



PCI Express* to PCI-X* Bridge Architecture:

Where Interface Standards Meet

Contents

Abstract	3
Background – Motivation and System Context	3
Key PCI Express to PCI-X Bridge Architecture Features	4
Performance Optimization	6
Summary	7

Abstract

As PCI Express technology deploys into computing and communications systems and peripherals, a means of supporting existing PCI and PCI-X deployment is required. Some add-in card manufacturers may desire a means of developing PCI Express endpoint application devices that plug into platforms that exclusively support PCI and PCI-X system slots. The PCI Express to PCI-X bridge architecture is a new industry-standard specification developed by the PCI-SIG. It is being introduced to satisfy the requirements for system and component architecture transition to bridge between the PCI Express and the existing PCI-X parallel busses.

This paper describes the nature of PCI Express to PCI-X bridges as implemented in both client and server computing platforms. Several factors that motivated the creation of this bridge architecture are presented and the key architectural requirements are outlined. Finally, significant aspects of the architecture are summarized and several tips on optimizing system performance are suggested.

Background – Motivation and System Context

PCI Express architecture promises to propel computing and communications I/O technology to an improved cost and performance paradigm. The improved bandwidth per pin, routing characteristics, reliability, and reduced pin count per interface that PCI Express provides offers system and add-in card developers immediate benefits upon migration. However, as the industry moves to PCI Express, a clean transition strategy and architecture is required to support hardware and software backward compatibility. This enables investment preservation by facilitating device interface and system compatibility (i.e. slot-level) so that current PCI and PCI-X add-in card and device solutions can be used in PCI Express-enabled systems. Backward compatibility can be achieved using the bridge elements shown in Figure 1.

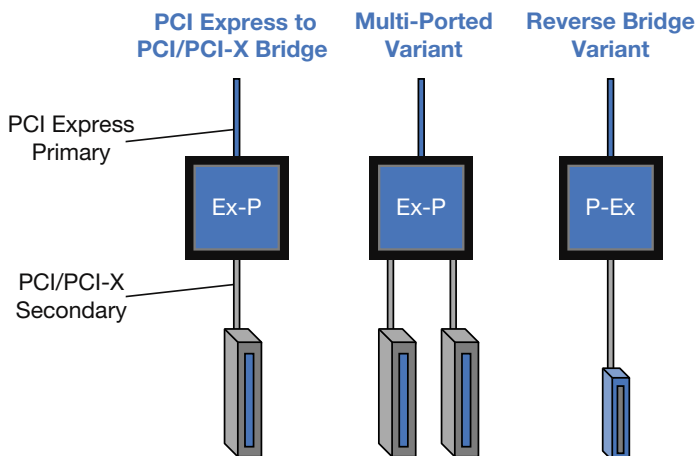


Figure 1: PCI Express bridge implementation examples.

The most basic bridge topology is that of a two-ported bridge that features a PCI Express primary interface and a conventional PCI or PCI-X (Mode 1 or 2) secondary interface. A multi-ported variation (i.e., a bridge with more than one secondary interface) can be used in systems that have significant I/O fan-out

requirements (Figure 2a). A PCI-X to PCI Express bridge, or reverse bridge, may be used to enable PCI Express add-in card solutions using existing PCI-X based application building blocks (Figure 2b). The basic bridge topology may also be used to enable new native PCI Express application building blocks to plug into existing PCI-X based systems (Figure 2c).

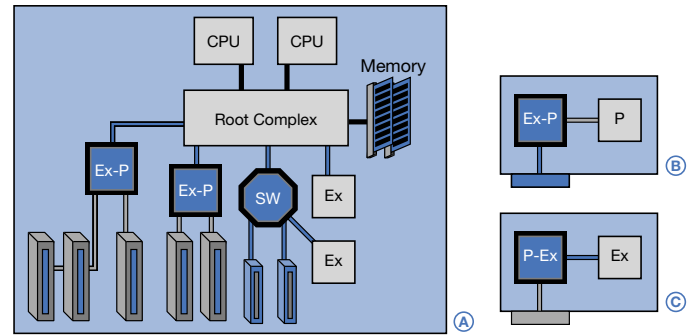


Figure 2: PCI Express bridge implementation examples for a) system board, b) PCI Express add-in card, and c) PCI-X add-in card.

By replacing today’s proprietary chip-to-chip system board interconnects with PCI Express, a top-to-bottom industry-standard I/O solution is possible that supports slotted and embedded versions of both PCI Express and PCI-X applications.

Utilizing PCI Express for the first level of fan-out in the I/O subsystem leads to simplified system board routing with fewer layers (Figure 3) and a degree of fan-out from the Root Complex (or memory controller) component that would not be possible with more pin-intensive I/O interconnects.

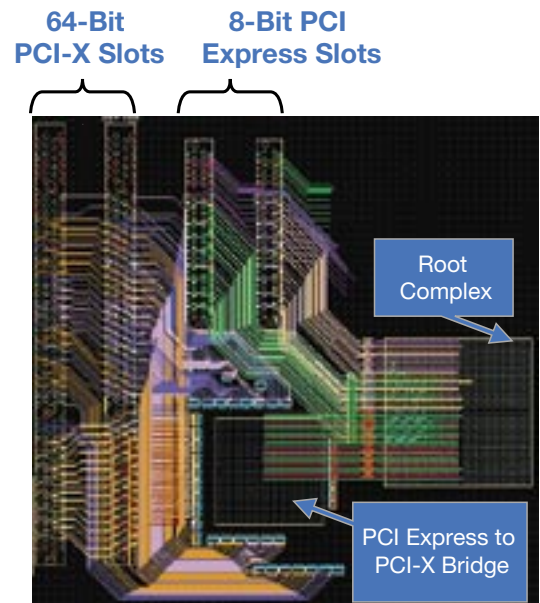


Figure 3: I/O subsystem layout example.

Providing the benefits of I/O backward compatibility, fan-out, and flexibility via an industry-standard bridge architecture means that the system software, system firmware, and device drivers can take maximum advantage of the capabilities of the I/O interconnects involved and avoid a functional “pinch-point” at the translation boundary, all in a standardized, dependable fashion.

Key PCI Express to PCI-X Bridge Architecture Features

The PCI Express to PCI-X bridge architecture is capable of translating PCI Express Base Specification, Revision 1.0 (PCI Express Base 1.0), compliant protocol and electricals to existing PCI and PCI-X interface standards, and doing so with a fully backward-compatible programming model. All bridges support conventional PCI, as defined in PCI Local Bus Specification, Revision 3.0 (PCI 3.0), but PCI-X Mode 1 and Mode 2, as defined in PCI-X Protocol Addendum to the PCI Local Bus Specification, Revision 2.0 (PCI-X PT 2.0), and the associated electrical/mechanical addendum, are optional. The key features of this novel bridge architecture are explored in the following sections.

Backward-Compatible Programming Model

The PCI Express to PCI-X bridge architecture is fully-compliant with today's PCI system software. Software backward compatibility is ensured by taking the same programming interface approach that PCI Express root devices, switches, and endpoint devices do, which is to leverage existing PCI device and PCI-to-PCI bridge configuration header data structures (Figure 3). The PCI Express to PCI-X bridge architecture preserves the familiar Type 1 bridge configuration header introduced in 1994 by the PCI-to-PCI Bridge Architecture Specification, Revision 1.0. A limited number of reserved register locations are invoked for new purposes. Most of the advanced features, including the PCI Express interface functionality, are added using PCI Capabilities List items and PCI Express Extended Capabilities List items. For example, PCI-X operation can be incrementally accommodated by adding the PCI-X Capabilities List item, as can secondary bus hot-plug support. Optional Advanced Error Reporting is added using an Extended Capabilities List item.

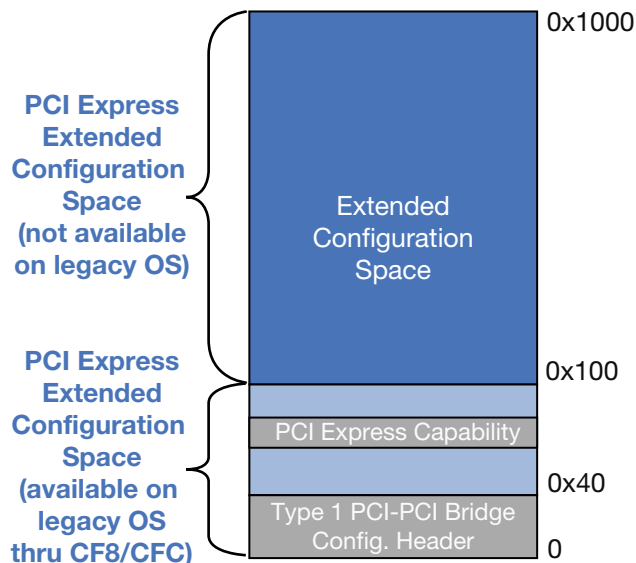


Figure 4: An example of a PCI Express to PCI-X bridge device configuration space.

Transaction Forwarding and Address Space Support

The PCI Express to PCI-X Bridge architecture supports the memory, configuration, and I/O address spaces defined in today's PCI and PCI-X standards. The bridge efficiently translates the PCI Express packetized protocol into PCI-X transactions (and vice versa) that are conveyed on a parallel interface using various control, address/data, and other signal bundles (Figure 4). In the downstream direction (i.e., PCI Express to PCI-X directed requests), a bridge may support memory-mapped (prefetchable, non-prefetchable, and exclusive), configuration, and I/O space transactions. The same support is available in the upstream direction, with the exception of configuration transactions and exclusive accesses, which are only permitted to be generated by the PCI Express Root Complex.

In some cases, it may be desirable to use PCI Express message commands for a vendor-specific purpose. PCI Express to PCI-X bridges may optionally support the translation of PCI Express Vendor-Defined Messages to/from Device ID Messages on a PCI-X secondary interface. These message transactions are routed by destination Bus Number, Device Number, and Function Number, instead of by a memory-mapped or I/O address. The usage model for these messages in the system context is implementation-specific but they may find use in a number of peer-to-peer I/O applications.

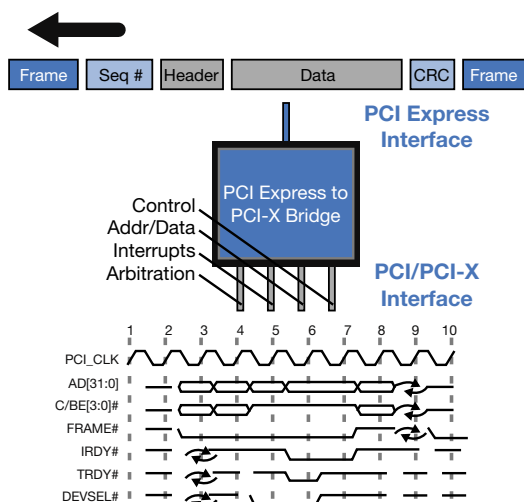


Figure 5: Bridge transaction forwarding between serial (PCI Express) and parallel (PCI/PCI-X) I/O interfaces.

Extended Bridge Configuration Space and Hierarchical Forwarding

PCI Express removes the 256-byte device configuration space restriction imposed on today's deployed PCI and PCI-X programming models. The specification permits extension of device configuration space to 4 KB using 12-bit device register addressing (see Figure 4). A PCI Express to PCI-X bridge accepts PCI Express packets that target its extended configuration space (Type 0 configuration) region and seamlessly converts Type 1 configuration requests that target its secondary interface or subordinate buses into PCI/PCI-X configuration transactions. Hardware interlocks prevent configuration corruption due to potential aliasing of 12-bit configuration register addresses that could be forwarded to downstream interfaces that support

only traditional 8-bit configuration register index decoding. This support enables native PCI Express applications to migrate to extended configuration support while existing PCI/PCI-X based platform infrastructure and supported applications are gradually converted over in a reliable fashion.

Interrupt Support

PCI-based system architectures support two styles of standardized interrupts: PCI INTx interrupts and Message-Signaled Interrupt (MSI/MSI-X) transactions. MSI/MSI-X transactions are the preferred method of signaling interrupts in new systems and their forwarding is supported by all bridges since these interrupts are memory write transactions. However, the PCI Express to PCI-X bridge architecture also provides for both the forwarding of sideband interrupts (INTA-D#) collected from the PCI-X interface and interrupts (both INTx and MSI/MSI-X) generated by sources internal to the bridge (such as by a Standard Hot-Plug Controller) as PCI Express “virtual wire” interrupt messages.

The steps involved in forwarding a PCI INTx interrupts from the secondary interfaces of a multi-ported PCI Express bridge possessing two secondary interfaces are illustrated in Figure 6. The multi-ported bridge collects the level-sensitive interrupt inputs from each secondary interface, logically ORs (i.e. collapses) them into a single set of internal signals, and then maps/converts the signals to the appropriate Assert_INTx/Deassert_INTx messages for forwarding on the PCI Express interface.

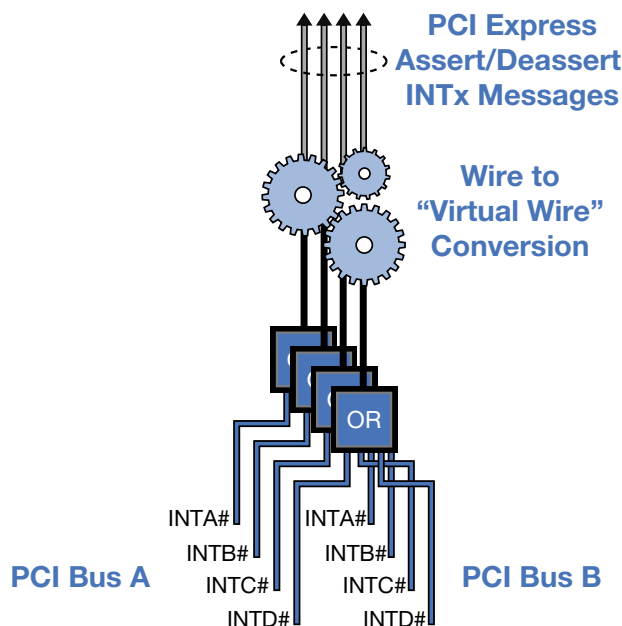


Figure 6: PCI INTx# collapsing and conversion to PCI Express INTx messages in a multi-ported bridge implementation.

Error Handling

PCI Express supports error signaling of correctable, (uncorrectable) non-fatal, and (uncorrectable) fatal events via in-band messages. PCI Express to PCI-X bridges must map PCI-X error occurrences to one of the PCI Express error classifications and, depending on configurable masks, transmit the appropriate PCI Express error message. Software backward

compatibility is maintained by reporting PCI-X error occurrences using the PCI-X status bits defined in the existing PCI-X specifications and by mimicking legacy system error signaling using the PCI Express error messages. An example of the mapping of PCI-X error events to PCI Express error messages is shown in Figure 7.

PCI/PCI Error	Mask	Express Message	Required/Optional
Received Master - Abort	MAM and (SERR Enable or NFERE)	Nonfatal	Req
Data Parity Error (Target)	SERR Enable or NFERE	Nonfatal	Optional
Address Parity Error	SERR Enable or NFERE	Fatal	Req

Figure 7: An example of the mapping of PCI error events to PCI Express non-fatal and fatal error messages.

Implementations that require more than a basic level of error handling capability from the bridge may add the PCI Express Advanced Error Reporting (AER) feature. The bridge-specific PCI Express Capability Structure for AER adds configuration registers that permit flexible reporting, masking, and message severity mapping of all potential PCI-X error occurrences on the secondary interface. When transaction-related errors are detected on PCI-X, address and attribute information may be logged for debug or recovery purposes.

In addition to signaling and logging errors, bridges are capable of forwarding PCI Express packets with poisoned data to PCI-X as transactions with bad parity and/or ECC. Packets received with end-to-end CRC (ECRC) errors are contained by the bridge since they cannot be reliably forwarded. Bridges are likewise capable of forwarding PCI-X Parity and/or ECC errors as PCI Express packets with poisoned data and may optionally forward these errors as bad ECRC.

Power Management

The PCI Express to PCI-X bridge architecture meets the requirements of the PCI Bus Power Management Interface Specification, Revision 1.1, the Advanced Configuration and Power Interface Specification, Revision 2.0, and supports the Active-State Power Management features introduced in PCI Express Base 1.0. All PCI Express devices, including PCI Express to PCI-X bridges, support a L0 low-power PCI Express link state and may optionally support L1 for greater power savings at the expense of longer exit latency.

The role of the bridge in power management event signaling is straightforward. Power Management Event (PME) capable PCI-X devices assert the PME# pin to signal a power management event. The collection of physical PME# signals from the secondary interface is required of bridges except when the bridge is specifically designed for system board applications in which the PME#

signals are routed around the bridge and directly to the Root Complex. Bridges convert the wired-OR PME# signal from each secondary interface into in-band PCI Express PME Messages (Figure 8). The same messages are used to convey PMEs signaled by sources integrated into the bridge.

Bridges are required to use the PCI Express Requester ID to either identify the internal source or the secondary interface from which the PME# event was collected. This identification permits software to determine the path to the device that signaled a request for service. Since, as with PCI-X bridges, the PME# signal input to the bridge is potentially a wired-OR collection of PME# signals from different PCI devices, the bridge must take care not to “lose” PME assertions. The problematic case is illustrated in Figure 9 and demonstrates the potential for a bridge to not detect the assertion of PME# from Device B if the wired-OR PME# input isn’t periodically polled. In the example, the assertion of PME# by Device A is conveyed to the system by the first PM Message. Because the bridge polled and found the PME# input later asserted, a second PM Message is generated that will bring Device B to the system’s attention. Redundant PM Messages may result from this scheme but they will have a benign effect on the system.

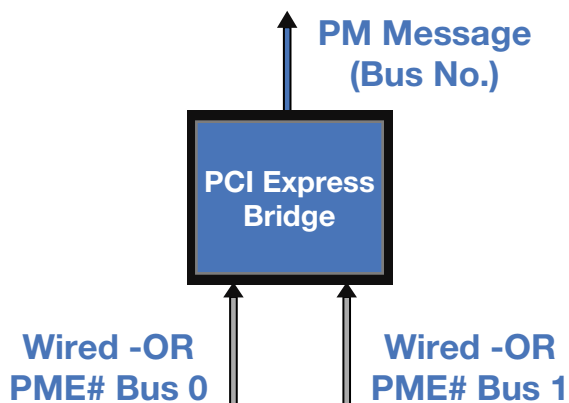


Figure 8: Collection and conversion of PME# signals from the secondary interfaces of a dual-headed bridge.

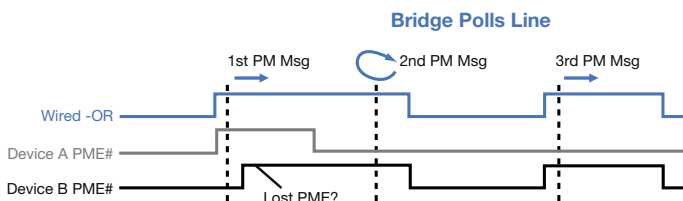


Figure 9: Schematic waveform that demonstrates potential for unidentified (i.e. lost) PME# signal assertions.

Performance Optimization

The following implementation recommendations will help ensure that a PCI Express to PCI-X bridge supports scalable and robust I/O subsystem performance. In some cases, these guidelines will also minimize bridge design complexity.

All bridge implementations contain a finite amount of buffer space that can be dedicated to the storage of transaction-related state information. The design goal, then, is to minimize or eliminate the storage requirements for state information related to requests forwarded by the bridge so as to take advantage of the trade-off depicted in Figure 10.

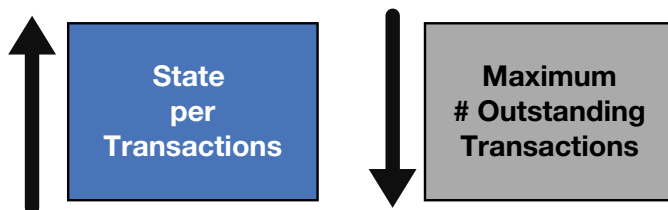


Figure 10: The simple relationship between the quantity of state stored per transaction and the maximum number of outstanding transactions that may be forwarded by a bridge.

Many requests may be “transparently” forwarded through a bridge without the need for storing any transaction-specific information; only the total outstanding header and data quantities are updated and stored to support the PCI Express flow control mechanism and PCI-X split transaction commitment limits. Some requests, though, require the bridge to take “ownership” of the transaction on behalf of the original requester. In these instances, the bridge will forward the request using its own Requester ID and store transaction-related state information that is used to subsequently return the completion to the original requester.

Two situations that require ownership are illustrated in Figure 11. Bridges must take ownership of PCI-X read requests that are fragmented into multiple read requests on PCI Express because the request straddled a naturally-aligned 4 K address boundary or because the maximum PCI Express read size is set smaller than the limit set on PCI-X (or vice versa). Bridges must also take ownership of PCI Express non-posted transactions that possess 8-bit tags since these tags may alias to an existing transaction’s 5-bit tag on PCI-X. Several steps can be taken to reduce the ownership burden on a bridge, including:

- Setting system values for PCI-X and PCI Express maximum read size to 512 bytes (the PCI Express default)
- Limiting the use of 8-bit tags on PCI Express to only when the number of potential outstanding transactions merits it
- Avoiding the issuing (by PCI-X devices) of requests that straddle naturally-aligned 4 K address boundaries

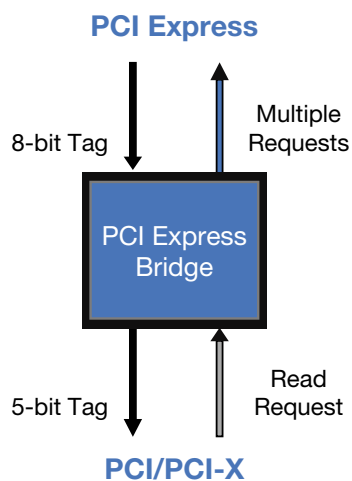


Figure 11: Two transaction forwarding situations that require a bridge to take ownership of a request.

A second important bridge implementation performance goal is the avoidance of inadvertent traffic backpressure. Backpressure mechanisms are an integral part of system architectures, preventing buffer overrun in data targets (e.g., main memory or an add-in card) under heavy traffic loading conditions. It is generally the objective of a properly designed I/O subsystem, however, to avoid being the throughput bottleneck. Potential sources of undesirable backpressure are identified in Figure 12.

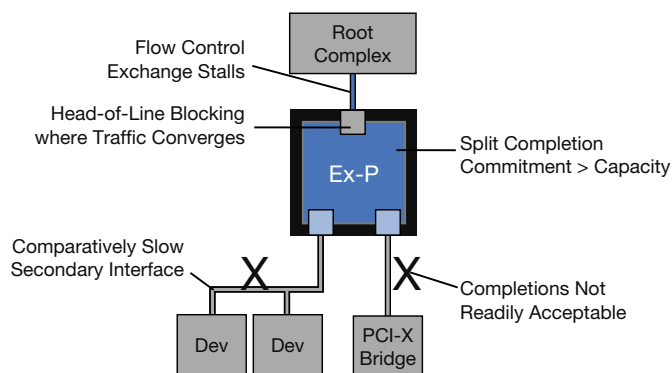


Figure 12: Potential sources of undesirable traffic backpressure in an I/O subsystem.

Although some sources of undesirable backpressure may be part of the basic cost-performance tradeoff in a system design, other sources can be mitigated through good design practice and proper consideration. For example, returning flow credits expeditiously on PCI Express will help avoid unnecessary flow control exchange stalls. Also, caution can be exercised when programming the PCI Express to PCI-X bridge's Split Transaction Commitment Limit greater than the Split Completion Capacity to service devices that request significant amounts of data or when the downstream devices support a lower aggregate throughput than the primary PCI Express interface. These situations can lead to the blocking of unrelated traffic at the PCI Express interface. Performance systems may take advantage of Relaxed Ordering and PCI Express Virtual Channels to reduce traffic stalls or the Commitment Limit and Capacity settings can simply be left at their default (equal) values.

Summary

The PCI Express to PCI-X bridge architecture offers full support of conventional PCI 2.3/3.0 and PCI-X 1.0b/2.0 while enabling systems to transition to native PCI Express. Compatibility with existing PCI system software and firmware is preserved while optional, new capabilities are supported that enhance the I/O subsystem value proposition. The scalable set of capabilities and high degree of implementation flexibility allow the bridge to have applicability in a multitude of computer market segments, from the value PC to the enterprise-class server. By observing proper, straightforward performance optimization guidelines, system and application developers will find that the bridge architecture taps the full potential of the interface standards supported.

Author Biography

Ted Wilke, Technical Editor, PCI Express Bridge Specification

For more information, visit the Intel web site at: developer.intel.com

Copyright© 2003 Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.

