PCI Switching Fabrics

Overview

This multicomputing technology brief introduces some of the issues which must be addressed when connecting PCI bus segments together with a switching fabric. Typical systems today may not need the whole range of capabilities provided by high-speed switching fabrics, but these features will become more important in the future. As processors, peripherals, and system architectures evolve, it will be necessary for PCI to keep pace. Switching-fabrics topologies are the appropriate way to evolve core PCI bus segment-interconnect capabilities. The framework to implement bridging via switching fabrics, with only extensions rather than major changes, is already in place with the PCI bridge specification.

This paper defines a standard interface which can be used with different, and evolving switching fabrics, but does not define the switching fabrics themselves. The PCI-to-switching-fabric-to-PCI (PXP) bridge interface definition builds on the current PCI-to-PCI (P2P) bridge specification.

Goals

High-performance systems need connectivity to dozens of high-speed peripherals. The use of an industry standard interconnect method provides system integrators with a wide choice of I/O vendors, features, and cost points, promoting fast time-to-market. With performance levels that exceed system demands, PCI provides the best solution because of its large, rapidly increasing market share.

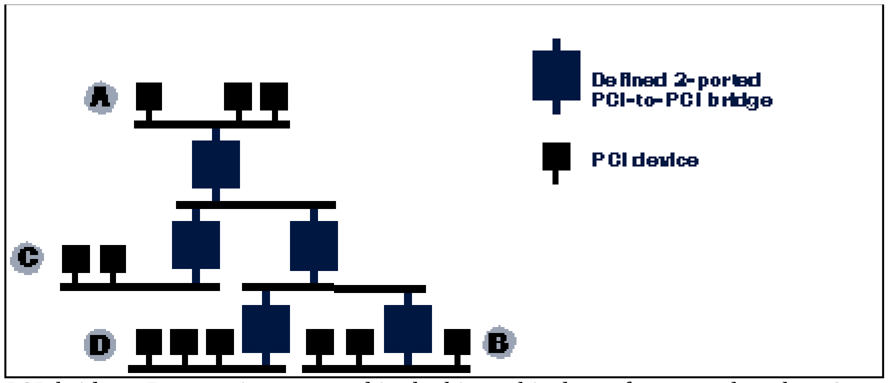
An ideal high-performance system provides the ability to:

- seamlessly connect many processors with many peripherals
- support the data throughput required to avoid bottlenecks
- scale connectivity upward as more processors and I/O are added

The goal, therefore, is to connect dozens of processors and peripherals together using PCI in a scalable, high-performance architecture.

Limitations

Unfortunately, PCI imposes electrical and physical constraints that limit its ability to efficiently scale to support large numbers of high-speed processors and peripherals.



A hierarchical tree using P2P bridges. Latency is protracted in the hierarchical tree for example, when A needs to communicate to B, arbitration must be completed on four bus segments. Contention is exacerbated for example, when A is communicating with B, C and D cannot, even though C's and D's bus segments are not busy.

Electrical limitations

Physically, PCI imposes electrical and mechanical constraints that ensure signal integrity. These include limits on loading, trace lengths, and connectors, typically limiting a single PCI bus segment to a total of 10 loads, with connectors considered as two loads. A motherboard or passive backplane is typically limited to four plug-in boards on a single PCI bus segment. Future 66 MHz PCI designs will be further constrained to only four loads per PCI bus segment or a single plug-in board on a motherboard (or two boards on a passive backplane). In less than five years, bus-frequency requirements will mandate point-to-point connections, necessitating a switching fabric interconnect between PCI agents.

Architectural limitations

On the logical side, P2P bridges are used to overcome the physical limitations of a single PCI bus segment. Bridging allows up to 256 PCI bus segments to be connected together. However, the bridge specification defines only dual-ported P2P bridges, so the 256 bus segments have to be connected as a hierarchical tree, as shown above. Poor contention, latency, and bisection-bandwidth characteristics make this is an inefficient interconnect topology for larger systems.

Solution

The solution is to connect PCI bus segments using a switching fabric instead of a bus hierarchy. A switching fabric provides point-to-point interconnects with the following features:

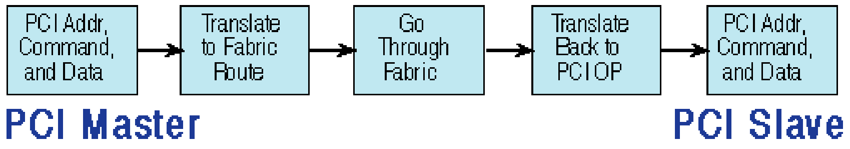
- high bandwidth
- low latency
- multiple simultaneous transactions
- scalable interconnect topologies
- realtime features desirable for I/O (priority-based preemption)

Switching Fabrics

A switching fabric is an interconnection architecture which uses multiple stages of switches to route transactions between an initiator and a target. One benefit of switching fabrics is that each connection is a point-to-point link. This inherently provides better electrical characteristics allowing higher frequencies and greater throughput than bus architectures. The use of multi-stage switching also allows flexibility and scalability in the size and topology of the interconnect. Examples of prevalent switching-fabrics standards are ATM at the WAN and LAN levels, and RACEway at the board and chassis levels.

Each stage of a switching fabric typically comprises an intelligent, multiport crossbar switch. The switch device recognizes a data-stream header message to dynamically route the interconnect transaction through the appropriate port to the next stage. The PCI to switching-fabric interface must forward transactions from one PCI bus segment to another. In the fabric interface, PCI addresses are translated to crossbar-switch route and fabric addresses, and then back to the original addresses at the destination PCI bus, as shown below.

Switching fabrics typically provide redundant interconnection resources to allow multiple transactions to proceed at the same time. This improves aggregate throughput by reducing contention and latency, and can also provide improved fault resiliency.


PCI Addr, Command, and Data
Translate to Fabric Route
Go Through Fabric
Translate Back to PCIOP
PCI Addr, Command, and Data
PCI Master
PCI Slave

The underlying principles of a standard switching fabric interface are:

1. Normal run-time software should not need to know that a switching fabric is present.
2. The only software that should need knowledge about the presence of a switching fabric is the Power-On Self Test (POST) code in the BIOS and operating system.
3. The only software that should need knowledge of the internal switching fabric mechanism should be loadable from an Expansion ROM by the POST code.

The tasks needed to standardize the interface to switching fabrics are:

1. Define addressing constraints and additional configuration registers to make the interfaces transparent, scalable and cost effective.
2. Define a workable approach to access Expansion ROM on bridges.
3. Define how interrupts and errors are handled through the switching fabric.
4. Provide optional extensions and guidelines that enable hardware and software designers to achieve optimal switching-fabric performance.

PXP Evaluation Issues

Some of the issues to be considered when evaluating the merit of a switching fabric for PCI bridging are:

- performance in terms of bandwidth, latency, and contention

- shared memory vs. message passing operations
- packet switching versus circuit switching
- flow control and buffering requirements
- transparency
- scalable distributed arbitration capability
- packaging and power

Performance: bandwidth

Since one of the main goals of PXPs is to enhance overall system performance, they should provide features that improve aggregate and bisection bandwidth and minimize contention and latency. Improved aggregate and bisection bandwidth is important when multiple processors and/or peripherals transfer data directly to each other and not just to system memory or a host processor located on a single PCI bus segment. Aggregate and bisection bandwidth is improved by using point-to-point interconnects that allow multiple, simultaneous transactions to occur throughout the switching fabric. Point-to-point fabric bandwidth should be greater than or equal to the bandwidth of the PCI bus segments being connected in order to achieve optimal system performance.

Performance: latency

Transfer of the first data in a switching fabric packet typically incurs a longer latency time than for a single P2P bridge. To minimize these effects, PXPs must be able to combine sequential data phases into bursts, write posting and read prefetching. In addition, initiator and target devices must amortize connection overhead over larger transfers by supporting bursting of large blocks of data. Latency counts! A small 32 byte block transfer at 132 MB/sec with 2us latency gives a net throughput of only about 14 MB/sec. A larger transfer of 1024 bytes at 132 MB/sec, however, greatly reduces the effect of the same 2 us latency, yielding a net throughput rate of 104.9 MB/sec.

Performance: contention

Data path contention results when multiple transactions try to use the same physical resource at the same time. Contention causes long and potentially unbounded transfer latencies," a condition incompatible with real-time data delivery. The PCI 2.1 spec addresses this problem with response time rules and delayed operations. The switching fabric must address this problem by supporting delayed operations and by providing arbitration that prevents starvation and deadlocks. Preemptive arbitration, where higher priority transactions automatically suspend lower priority transactions, may be needed to bound latencies for real-time performance.

Shared memory vs. message-passing operations

PCI addressing uses a flat address space and a shared memory model, but the insertion of a switching fabric contradicts this model. Support for interrupts and error notification through the fabric, therefore may require the use of message passing primitives.

Packet switching versus circuit switching

Write posting is a form of packet switching, but reads, locked accesses, and other PCI operations with ordering constraints imply circuit switching. A switching fabric has to be flexible enough to support the switching type needed for a particular PCI operation. To guarantee deterministic PXP transfer times, cross nodes must establish a logical circuit, and fabrics supporting write posting, read prefetching, delayed operations, multiple concurrent transactions, and priority preemption will remove or alleviate the contention problems associated with simpler circuit-switched topologies.

Flow control and buffering requirements

PCI transactions require flow control and adequate buffering to ensure lossless data transmission. Implementing this requirement may result in poor performance across a large switching fabric if not handled correctly. Data-flow latency occurs while a PCI sender waits for flow control acknowledgement to occur on a per-transfer basis. Instead, destination acknowledgements can be pipelined so the acknowledgement latency time is only incurred on the first transfer of a block.

Transparency

In order to be viable, PCI bridges must be completely transparent to both initiator and target locations during normal operations. This applies for both P2P and PXP bridges. In other words, it must be possible to initiate a transaction from one PCI bus, through the switching fabric, to a target PCI bus without having run-time knowledge of the switching fabric's existence or functional characteristics. PXP support must not require any modification of run-time device-driver software. In addition, the PCI interfaces must be hardware compatible with the standard PCI specification.

Bridge-specific knowledge is necessary during initialization to configure the bridges for transparent operation. However, PXPs should be mostly compatible with P2Ps to simplify their support. Thus the Power On Self Test (POST) and BIOS code should be as similar as possible to the standard code supporting PCI bridging. In other words, the PXP specification should only be an extension of the P2P specification, and not a redefinition.

Scalable, distributed arbitration capability

Centralized arbitration doesn't work for large systems with many endpoints all trying to perform transactions simultaneously. Distributed arbitration is needed, with built-in deadlock and starvation prevention.

Packaging and power

Multiport switches need packaging that supports the number of wires per port times the number of ports per chip. These switches quickly become pin and power limited if the number of wires per port is large. However, by designing a PXP switch port limited to 40 signals, a six-way silicon crossbar switch can be implemented in about one square inch, consuming about one watt of power.

The RACEway Interlink draft specification discusses these issues in greater detail. The specification is available via anonymous FTP as /pub/spec1.5.ps from ftp.mc.com.

Other Issues

Other functional issues which must also be addressed include:

- Addressing
- Configuration
- Strong Ordering and Buffer Flushing
- Deadlocks and Interlocks
- Error Handling
- Interrupts

Addressing

Memory space, I/O space, and configuration space operations must be supported as defined in the PCI 2.1 specification, to allow PXP compatibility with standard PCI devices, memory space, I/O space, and configuration space operations must be supported as defined in the PCI 2.1 specification.

This compatibility is achieved by emulating multiple levels of standard PCI to PCI (P2P) bus bridges with the switching fabric. The multiple levels of bus segments are treated as up to 32 virtual P2Ps (PXPs) hanging off a single virtual PCI bus segment (this is an architectural limit today). This method accomplishes compatibility with standard PCI devices while keeping the underlying switching fabric transparent.

Another compatibility scheme attaches up to 8 virtual bridge functions to one PCI device, each decoding its own set of address and configuration-space windows. This requires multiple sets of configuration registers and logic to mimic the behavior of up to 8 P2P bridges. Since each device only supports 8 virtual P2Ps as the fabric endpoints, however, this method does not scale well for connecting larger fabrics.

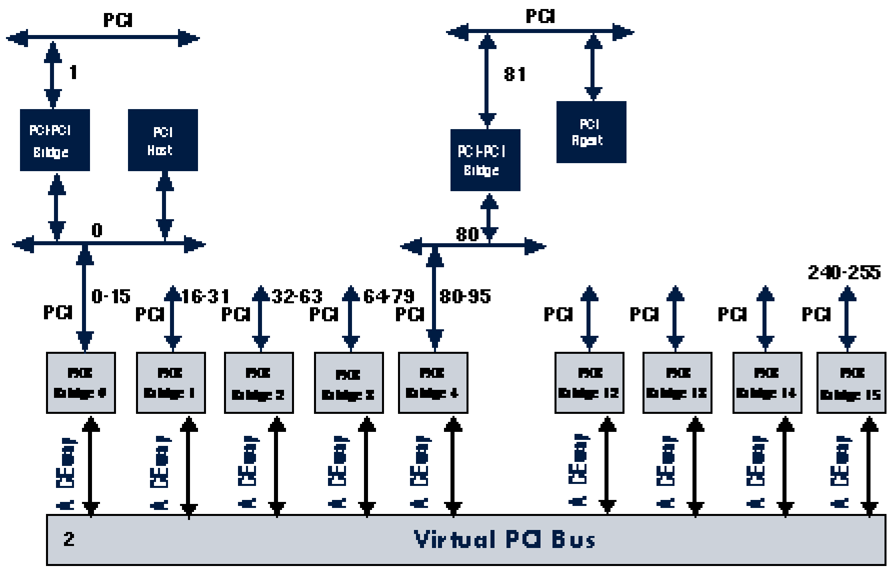
Mapping the address windows in the three spaces into routes through the fabric can be done cleanly and efficiently within the framework of PCI and P2P bridges. However, this is a complex issue which is discussed in a separate white paper.

Configuration

The standard boot sequence for PCI buses is as follows:

1. The host uses configuration operations to probe for additional PCI bus segments using a depth-first algorithm. At the end of this step, the POST code has created a map of all of the PCI bus segments to which it is attached, and has configured each P2P to respond to configuration operations using bus numbers as addresses.
2. The host then uses configuration operations to probe for devices on all the attached PCI bus segments. At the end of this step, the POST code knows about all the attached devices and their expansion ROM, memory and I/O address needs.
3. At this point, the host assigns expansion ROM, memory, and IO address spaces to the devices, enabling them to respond to memory and IO commands. It also sets up the PXPs to pass through operations addressed to the address-mapped PCI devices behind the PXP bridges.
4. In the next step, the POST code reads from the expansion ROMs, and executes the device specific initialization code (x86 or Open Firmware) for each device.

To allow flexible initialization of the fabrics and their interfaces, it is desirable to use the expansion ROM mechanism to load and execute PXP-specific code. The bridges to the switching fabrics need to be initialized during step 1 to allow the configuration operations to traverse them transparently. The problem is that the expansion ROMs map into memory space, which isn't allocated until step 3. The POST code therefore needs to assign temporary addresses to access the expansion ROMs on bridges.



Typical Bus Numbering with RACEway Fabric

Strong Ordering and Buffer Flushing

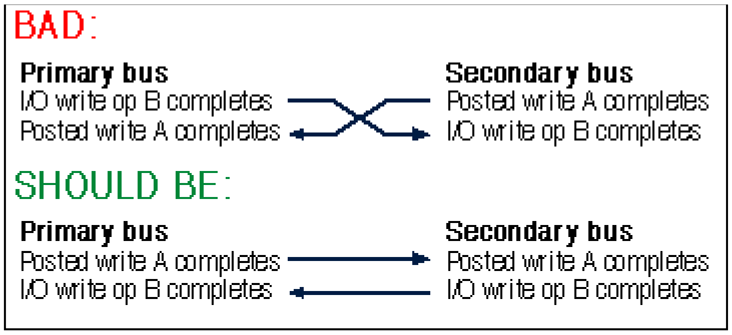
I/O, configuration, and locked operations are coupled operations - this means that they can't be posted and "must" flush the posted write buffer in the opposite direction before completing. (Note that with delayed operations they can still proceed through the fabric, they just can't complete until the conditions are met.)

The reason that the I/O and configuration operations are required to behave this way is to ensure that both:

1. The operations complete on the destination bus before completing on the initiating bus, and
2. Operation order on the destination bus is the same as operation order on the initiating bus.

EXAMPLE: A SCSI device on PCI writes data through the switching fabric to a processor's memory. For performance reasons these are posted writes, which means that the SCSI device recognizes them as a complete transaction before all the data actually reaches the memory (some of the data can still be in buffers in the switching fabric). At some point, the device driver running on the processor may perform an I/O write across the switching fabric to set the SCSI device off-line. The driver will see the off-line command complete, and then possibly reassign or free its memory allocation.

The I/O command must be a coupled operation to ensure that it does not complete until all of the posted write buffers between the processor and the I/O device have been flushed. If the I/O command were not a coupled operation, there might still be data in the switching fabric buffers, en route to the processor memory. This data would get written to the original memory locations after the driver thinks the device is already off-line, resulting in data loss or other errors.



Device drivers (and devices) should use memory commands instead of I/O commands for performance and IO address limitation reasons. They must then use other methods to ensure that any data from the device is no longer still buffered between the device and the destination. For instance, the driver may wait until it sees a completion status write from the device after a data write. If the completion status write comes through then all of the data in front of it must have been written as well. (A PCI switching fabric must maintain the sequential ordering of posted writes originating from a given bus segment.)

Deadlocks/Interlocks

Interlocked buses such as EISA and VME have the potential to cause deadlocks when performing a transaction to a PCI device through a switching fabric. This can happen because they may not be able to retry an operation to let another operation complete first. To avoid deadlock, bridges to these buses traditionally perform transactions only with agents on their nearest PCI bus segment, where the legacy bus bridge uses arbitration on both buses and/or sideband signals on PCI. With PCI switching fabrics (or multiple levels of P2Ps) these methods can not be used.

EXAMPLE:A SCSI device on PCI writes data through the switching fabric back to a memory buffer on an interlocked bus. At the same time, a master on the interlocked bus tries to read from the same remote PCI bus segment. The data is write-posted in the bridge (for performance) on its way to the interlocked bus memory. However, it gets hung up in a bridge buffer waiting for the master on the interlocked bus to complete its read, which is also hung up waiting for that buffer to flush. In the normal PCI case, the read would be retried to allow the write to complete and get out of the way. With interlocked buses, the read may not be able to be retried, which results in a deadlock.

One solution to this requires an extension called Read-Around Bypass. In a Read-Around Bypass, data being returned for a read goes around data stuck in a posted write buffer. The advantages of this are that it prevents deadlocks, it doesn't require purging of valid posted write data, and it doesn't require losing performance by needing to turn off posted writes. The disadvantage is that if the posted write data is going to the interlocked device, then strong operation ordering is lost. This is probably acceptable, since a read still follows any write data from the interlocked bus going in the same direction (Ordering of operations originating on a given bus is still maintained.) The only thing which can't be guaranteed is strong ordering of operations between the two buses. A posted write may complete on bus A (its originating bus) before a read coming the other way, while completing after the same read on bus B (its destination bus.)

Error Handling

PCI has three error handling categories:

1. In PC compatibility mode, the PCI bridge sets a status bit indicating an error, then proceeds as if no error occurred, leaving it up to the software to poll for errors.
2. In transaction error handling, the PCI bridge notifies the master that an error occurred while the

transaction is still in progress. This may allow the process which initiated the transaction to either handle the error or terminate with an error indication.

3. In system error handling, an error is detected after the transaction completes at the master. In this case, since there is no transaction associated with the error, the only recourse is to notify the host that an error occurred. The host probably won't know which process had the error, and may perform a system reboot to try to clean up after the error.

The PXP detects several different kinds of errors, including Master Abort, Retry Counter Overflow, Target Abort, and Parity Errors. The error handling for each of these proceeds as outlined in the PCI Bridge Spec. SERR# and PERR# notification across the fabric can be handled the same way as interrupts.

Interrupts

Interrupts may be passed across the switching fabric using message passing. Upon sampling a change on an interrupt signal (INTA#, INTB#, INTC# or INTD#) from deasserted to asserted, the appropriate message is sent to the primary PXP, which then sets the appropriate interrupt line on its PCI bus. On sampling a change on an interrupt signal (INTA#, INTB#, INTC# or INTD#) from asserted to deasserted, another appropriate message is sent to the primary PXP, to reset the appropriate interrupt line on its PCI bus.

Counters are needed in the primary side PXP for each of the 4 interrupt lines. Each one increments on receiving a set message, and decrements on receiving a reset message. The appropriate interrupt line is asserted while the counter greater than 0. This allows multiple devices on the secondary side of the switching fabric to share the same interrupt line. Additional support may be desirable to minimize the interrupt handler's search time with shared lines in a large system.

Final Note

A transparent PCI switching fabric poses many technical challenges. A solution to the challenges is being formulated by Mercury and Cypress Semiconductor using the existing RACEway switching fabric. With the addition of a RACEway to PCI bridge chip, RACE crossbars will act as a transparent PXP fabric. In the future, RACEway will evolve to support 64-bit features and higher frequencies needed for supporting future PCI devices. Please contact Mercury or Cypress for the most current information.
