Sub-microsecond interconnects for processor connectivity—The opportunity

Sam Fuller - May 22, 2013

As Moore's Law has continued to drive the performance and integration of processors ever higher, the need for higher-speed interconnects has continued to grow as well. Today's interconnects commonly sport speeds ranging from 1 to 40 Gigabits per second and have roadmaps leading to hundreds of gigabits per second.

In the race to faster and faster speeds for interconnects what is often not discussed are the types of transactions supported, the latency of communications, the overhead of communications and what sorts of topologies can be easily supported. We tend to think of all interconnects being created equal and having a figure of merit based solely on peak bandwidth. Reality is quite different. Much as there are different forms of processors that are optimized for general purpose, signal processing, graphics and communications applications, interconnects are also designed and optimized to solve different connectivity problems.

Typically an interconnect will solve the problems it was designed for very well and can be pressed into service to solve other problems, but it will be less efficient in these applications. It is instructive to review three important interconnects in this context. These interconnects are PCI Express in the Gen 2 and Gen 3 form, Ethernet in the increasingly popular 10 Gigabit form and the second generation and third generation RapidIO technology introduced in 2008.

Each of these technologies has moved to a multi-lane SerDes physical layer using 8B/10B line coding or more efficient line encodings like 64B/66B line coding for the higher speed offerings. While PCI Express and RapidIO offer wider interfaces than 4 lanes, wider interfaces will not typically be used across backplanes or on to Fiber or Cable connections. The Gen 3 RapidIO standard extends the 64B/66B scheme of the 10G Ethernet KR standard with an extra polarity inversion bit (64B/67B) that guarantees continuing DC balance of the transmitted bitstream.

The following table presents the typical bandwidth and lane configurations for PCI Express, RapidIO and 10 Gig Ethernet as used in processor connectivity applications.

	Gen 2 PCI Express	Gen 3 PCI Express	10 Gig Ethernet	10 Gig Ethernet	Gen 2 RapidIO	Gen 3 RapidIO
Lane Count	1, 2, 4+	1, 2, 4+	4 (XAUI)	1 (KR)	1, 2, 4+	1, 2, 4+
Signaling Speed	5 GHz	8 GHz	3.125 GHz	10.312 GHz	6.25 GHz	10.312 GHz
Line Coding	8b/10b + scramble code	128b/130b w/ scramble code	8b/10b	64b/66b w/ scramble	8b/10b w/ scramble	64b/67b w/ scramble
Bandwidth for four lanes (Except for 10G KR)	16 Gbps	32 Gbps	10 Gbps	10 Gbps	20 Gbps	40 Gbps
Latency	Sub microsecond	Sub microsecond	Tens of microseconds	Tens of microseconds	Sub microsecond	Sub microsecond

This article will focus, not on the raw bandwidths of the interconnect technologies, but rather on the inherent protocol capabilities, supported topologies and latency design targets for each of these interconnects. By doing this we gain a better understanding of where it makes sense to use

each technology.

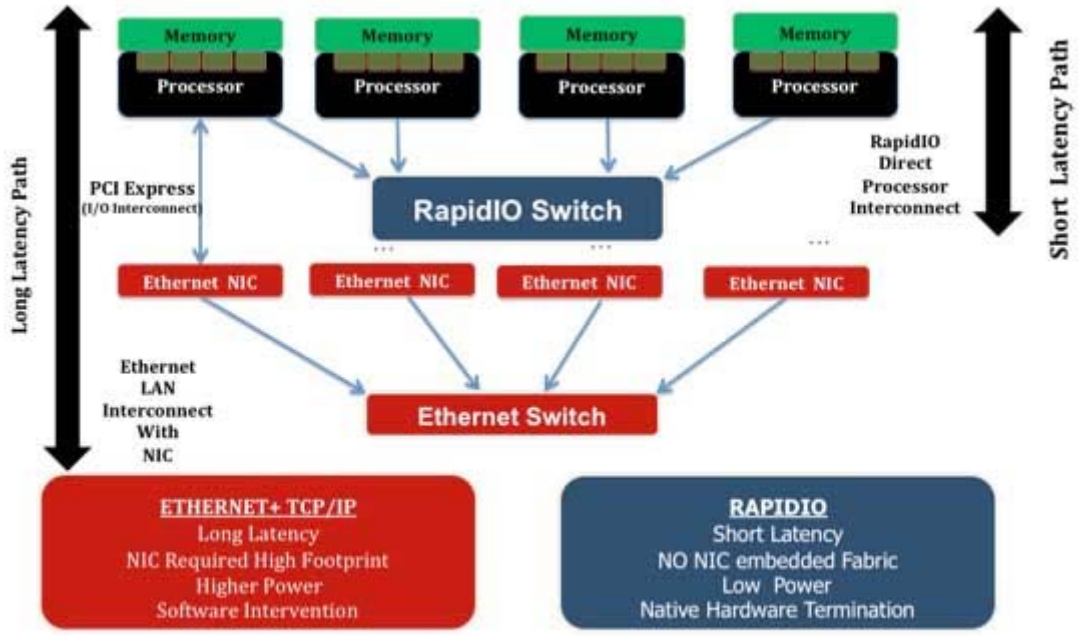

Figure 1. Different connectivity solutions between processors, I/O and systems.

PCI Express Transactions and Topology

PCI Express was designed, in 2003, to connect peripheral devices, typically slave devices like Ethernet NICs and graphics chips to a main host processor. It was not designed as a processor to processor interconnect but rather as a serialized version of the PCI bus. The acronym PCI stands for peripheral component interconnect. PCI Express retains the same programming model and approach to connectivity. Topologically PCI Express can support a hierarchy of buses with a single root complex. PCI Express switches have explicit upward (towards the root complex) and downward (towards attached devices) directions. Switches are primarily designed to expand peripheral device connectivity in systems.

Natively PCI Express does not support peer-to-peer processor connectivity. Using PCI Express for this sort of connectivity can be exceedingly complex. When you try to build a multi-processor interconnect out of PCI, you, of necessity, must step beyond the base PCI specification and create new mechanisms to map address spaces and device identifiers among multiple host or root processors. To date none of the proposed mechanisms to do this -- Advanced Switching (AS), Non-transparent Bridging (NTB) or Multi-Root – I/O Virtualization (MR-IOV) -- have been commercially successful nor do they support arbitrary topologies.

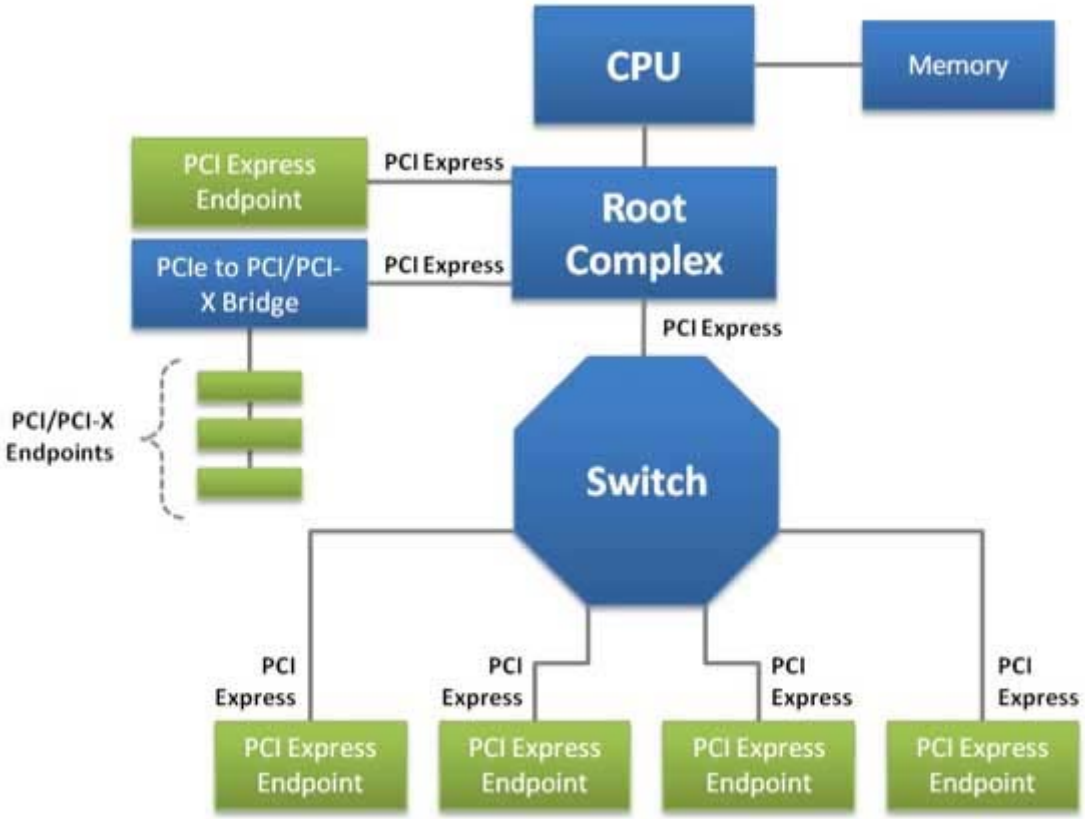

Figure 2: Typical PCI Express System Topology

PCI Express is not a routable protocol, like Ethernet or RapidIO. It defines a single large address space that devices are mapped into. Performing load or store operations to addresses in to the address range associated with a specific device is the most common way to communicate across PCI Express. PCI Express bridge or switch devices must detect the targeted device by comparing the 32-bit or 64-bit address contained in the packet against a set of base and limit values and forward the packet to the device or downstream switch that is associated with the address contained in the packet. A separate ID routing scheme is also support where devices are identified by bus number, device number and function number. This ID routing scheme is typically used for configuration and for message based communication. This scheme is not useful for transferring data. The bus number, device number and function numbers for ID routing, like the address space allocations are assigned during system bring up and discovery.

PCI Express packet routing uses three different algorithms, depending on the packet type. All of the algorithms assume that the system has a tree topology, with a root complex at the top and a global address map managed by the root complex:

- Address based: Base-and-limit registers associate address ranges with ports on a PCIe switch. There are three to six sets of base-and-limit registers for each switch port.
- ID based: Each PCIe switch port has a range of bus numbers associated with it. Packets are routed according to their bus number, device number, and function number.
- Implicit: PCIe Message packets make use of “implicit” routing, where the routing is determined by the message type.

PCIe has evolved to support “non-transparent bridging”, which allows separate root complexes to send transactions to each other. Typically, non-transparent bridging requires packet addresses and bus/device/function numbers to be translated in order to resolve conflicts between the global address maps of the different root complexes. There are no standards for implementing this translation capability.

Sub-microsecond interconnects--Page 2.

Without enhancing the capabilities of the base PCI Express protocol there is no natural way to support peer-to-peer processor connectivity. Products that advertise multi-host PCI Express switching are essentially moving an Ethernet-like NIC functionality into the “switching” device. While there can be value in doing this, the result is a more complex switching device and a proprietary single-vendor solution. The ability to cascade switches or support topologies like mesh or fat-tree connectivity are very problematic with this approach as well. Interoperability between different vendor’s products would be difficult to assure.

It should be noted that both PCI Express and RapidIO are designed to be tightly integrated with the memory subsystem in a SOC device. This allows for significantly reduced latency and higher bandwidth communications. External attached fabric interfaces like Infiniband or Ethernet will always require an extra stage of communication to move data to or from processor memory.

In summary, for systems where there is a clear single host device and other processors and accelerators operate as slave devices, PCI Express is quite a good choice for connectivity. However, for connecting many processors together in more complex systems, PCI Express has significant limitations in topology and support for peer-to-peer connectivity.

Ethernet Transactions and Topology

Many developers have looked to Ethernet as a solution for connecting processors together in systems. Ethernet has evolved significantly over the past 35 years and similar to the growth in computer processing speeds its peak bandwidth has grown steadily. Currently available Ethernet cards can support 40 Gigabits per second operating over four pairs of SerDes with 10Gbps signaling. Such cards contain significant processing resources on their own to be able to support the transmission and reception of packets at these speeds.

To use Ethernet, which is a best effort, lossy interconnect, as an efficient processor interconnect requires significant transaction acceleration to avoid the overhead and latency of software, and also enhancements to the Ethernet MAC and to the Ethernet switch devices themselves to reduce latency. Even with these enhancements interprocessor communications operations are typically only used for large block transactions in order to amortize the overhead and latency of using Ethernet.

Two important standards that have been developed to specify mechanisms for performing remote direct memory transfer operations (RDMA) over Ethernet. The iWARP RDMA protocol was first introduced in 2007 and has had limited success. The iWARP protocol relies on TCP and IP addressing to provide a reliable transport mechanism.

Most RDMA over Ethernet development is now focused on the recently introduced RoCE technology, RoCE, pronounced Rocky, is the acronym for RDMA over Converged Ethernet. The underlying transport for RoCE is made more reliable through the addition of a number of protocol enhancements collectively know as Data Center Bridging (DCB). These enhancements provide mechanism to avoid dropping packets due to network congestion.

It is important to note that DCB does not guarantee that packets will not be dropped in an Ethernet network, it just provides mechanisms that a network can use to communicate congestion

information and avoid situations where packets would be forced to be dropped due to congestion. DCB also does not guarantee that packets will not be lost due to packet errors or poor buffer management. This makes the Ethernet an almost lossless technology, almost lossless because packets may still be lost due to errors or poor buffer management. But systems designed using DCB will still need to be tolerant of lost or out of order packets. Because TCP/IP is not used with the RoCE protocol, the technology also is restricted to a single layer-2 Ethernet network, it can't cross router boundaries.

Both iWARP and RoCE are typically implemented through acceleration processors resident in the NIC card itself from vendors like Chelsio and Mellanox. Even with DCB support, RDMA transactions must still be carefully managed to reduce communications overhead. Ethernet Payload Efficiencies are good for very large packets, and very poor for small packets. If your communications model is based on small messages then Ethernet is quite inefficient. For example, for 100 Byte payloads, Ethernet transactions using UDP are only 60% efficient in terms of payload to bits on the wire and this efficiency gets significantly worse for payloads smaller than 100 Bytes. Transactions using TCP would be less efficient.

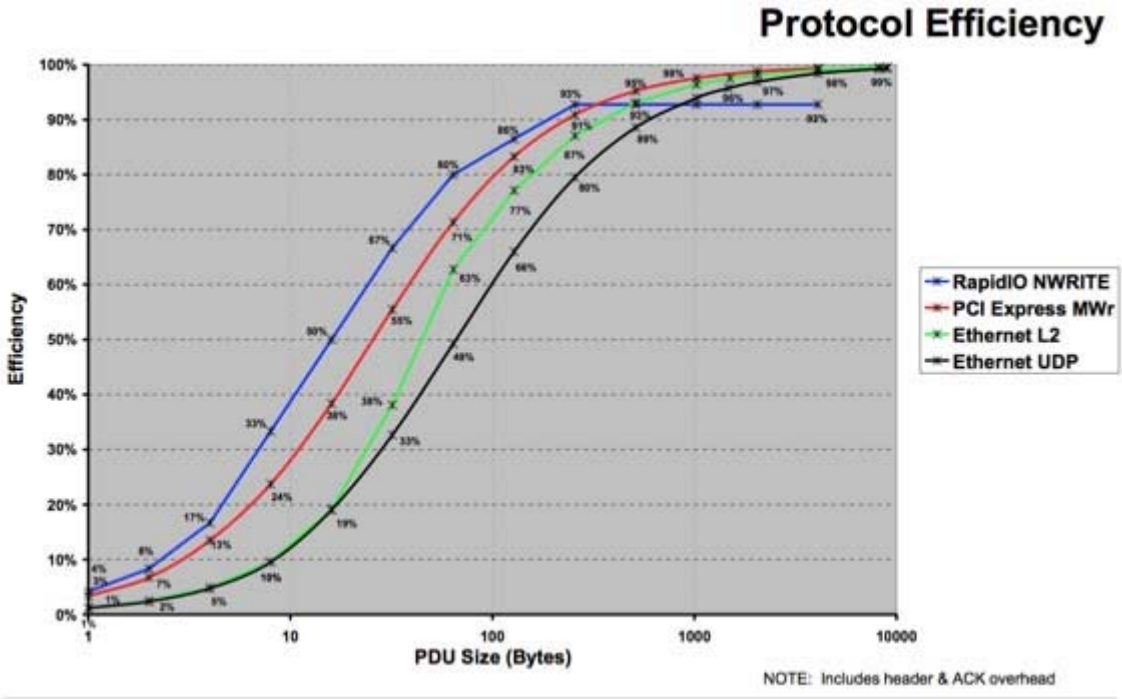

Figure 3: Comparisons of Protocol Efficiency

While the previous chart shows the efficiency of the bits on the wire, the following figure shows the latency of message transactions for 100 Byte messages implemented using RoCE and iWARP running on 10Gigabit Ethernet. This data shows, first the high variability of latency for iWARP due to the use of TCP with transaction latency varying between 10 uSeconds and over 60 uSeconds.

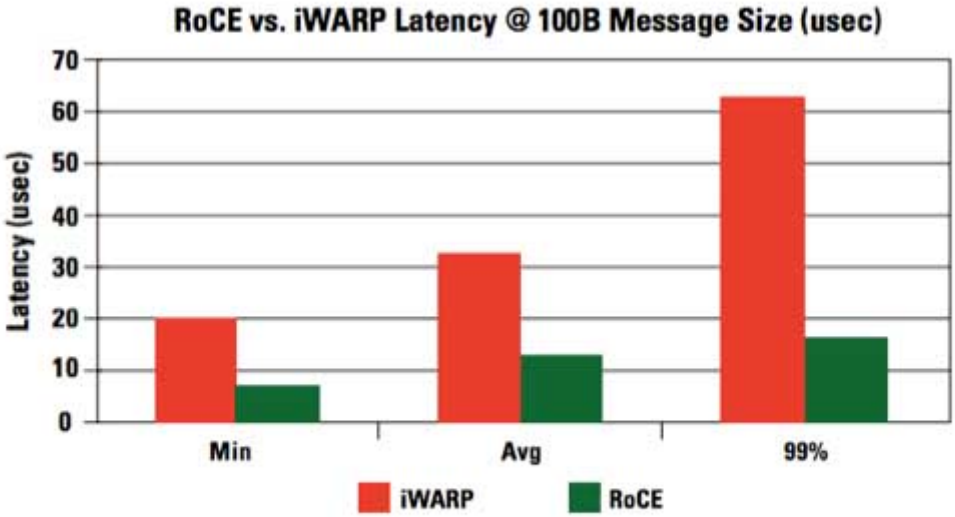

Figure 3. Comparing iWarp to RoCE Latency for message transaction. Source: Mellanox RoCE vs. iWARP Competitive Analysis Brief, November 2010.

Sub-microsecond interconnects--Page 3.
Ethernet Switching

In data center environments, Ethernet is ubiquitous. Ethernet originally made use of a "longest possible match" algorithm when routing IP packets. This supported Ethernet's spanning tree protocol and other networking functionality, however, it was complicated, power hungry and significantly increased latency through switches. Ethernet switches have evolved to support various flavors of Virtual LAN (VLAN) tags, which allow packets to be routed using a simpler indexing method. Software-defined Networking (SDN) approaches such as OpenFlow, work to establish common approaches to forwarding tables that all switches can support. This makes it easier for data centers and carrier networks to support more sophisticated functionality such as firewalls, prioritization, routing and virtual LANs. Technologies such as SDN drive up the complexity of Ethernet Switch silicon to provide more programmability and capabilities for data center networks. While these features are quite useful for data center computer-to-computer communications they add unnecessary power, cost and complexity when Ethernet is used as an internal point-to-point or fabric solution inside embedded computers.

Ethernet and the TCP/IP protocols that support it were designed to work in a lossy and dynamic environment. This is absolutely the right solution for connecting discrete computers together. However, when computers become virtualized and hosted on arrays of closely connected processors the overhead of Ethernet's software based communications protocols and lossy interconnects becomes apparent.

The largest issue relates to transaction latency. As discussed earlier, unidirectional communication across a switched Ethernet link using TCP/IP will typically take tens of microseconds. A traditional multicore processor can, during the time it takes to send and receive an interprocessor communications message, perform millions of operations. The Ethernet protocol does not support any sort of automatic packet acknowledgement, meaning that the target of an Ethernet packet does not automatically communicate back to the sender the receipt of the packet or the results of the request. By leaving the packet reliability confirmation to software, the use of Ethernet adds significant latency to low-level transactions when compared to protocols like PCI Express and RapidIO.

For example, if Ethernet were used to perform something like a remote memory read operation, a very common RDMA operation, the packet containing the read request, would need to be received by the Ethernet NIC hardware and interpreted by software. The packet would then need to be classified as it could conceivably be any type of packet, once identified as an RDMA read operation the request would be forwarded to the RDMA software which would interpret the request, identify the requested data and prepare the response Ethernet packet or packets. The response packet would then return the requested data back to the requestor through the TCP/IP and Ethernet stack and driver software. As demonstrated by the Mellanox data, best-case messaging latencies for 10 Gigabit Ethernet is in the range of 6 microseconds with average latencies closer to 13 microseconds when using the accelerated RoCE technology. In both PCI Express and RapidIO the handling of a data read request can be completely conducted in hardware with no software interaction, latency for transactions for both PCI Express and RapidIO will typically be under 1 microsecond and as low as 500 nanoseconds. This is much closer to the latencies seen when communicating with on-chip resources or with DRAM.

The Ethernet value proposition is well known, but there are significant questions related to its use as a transport for processor connectivity in tightly coupled systems. While Ethernet is clearly the right choice for connecting servers and computers together, as networking requirements grow, driven by software defined networking (SDN) requirements, what are sold as Ethernet switches are becoming capable of doing much more and approaching routers in terms of complexity. These new SDN switches are making switching decisions based on much deeper packet information including the IP address, the TCP port number and potentially even based on the associated application data that the packet is targeted at. In aggregate the combination of software transaction processing, Ethernet MAC processing (often with acceleration) and then Ethernet or SDN switching latency add complexity, cost and inevitably higher power dissipation inside systems where much more targeted and power efficient connection approaches could be utilized.

RapidIO Transactions and Topology

RapidIO was designed to serve as processor fabrics interconnect. RapidIO is most-often used in embedded systems that require high reliability, low latency (typically sub microsecond) and deterministic operation. It is designed to support connecting different types of processors from different manufacturers together in a single system. Because of this it has found widespread use in wireless infrastructure equipment where there is a need to combine general purpose, digital signal, FPGA and communication processors together in a tightly coupled system with low latency and high reliability.

The usage model of RapidIO created the need to support memory-to-memory transactions, including atomic read-modify-write operations. To meet these requirements RapidIO provides direct RDMA, messaging and signaling constructs that can be implemented without software intervention. For example in a RapidIO system, a processor can issue a load or store transaction or an integrated DMA engine could transfer data between two memory locations and these operations would be conducted across a RapidIO fabric where the sources or destination addresses for the operations are located across the RapidIO fabric. These operations will typically occur without any software intervention. Also, as viewed by the processor they are no different than common memory transactions.

RapidIO was also designed to support peer-to-peer transactions. It always assumed that there would be multiple host or master processors in the system and that those processors needed to communicate with each other through shared memory, interrupts and messages. Multiple processors, up to 64K, can be configured in a RapidIO network, each with their own complete address space.

Sub-microsecond interconnects--Page 4

Mechanisms for packet routing have evolved in different directions for the different technologies. RapidIO uses a simple and flexible routing scheme based on a device ID, which is a value used to identify an endpoint in a RapidIO network. Endpoints may have more than one device ID. Each RapidIO packet has a destination ID, which is the device ID of the endpoint which should receive the packet, and a source ID, which is the device ID of the endpoint which originated the packet. RapidIO switches use a packets destination ID as the index in a table. Each table entry indicates the port, or the bit vector of ports for multicast, which the packet should be sent to. Once a packet reaches it's destination, the endpoint may formulate a response. The routing information in the response is formed by using the request source ID as the response destination ID, and the request destination ID as the response source ID. RapidIO switches are forbidden from modifying packets.

RapidIO also provides a very clean dividing line between the functionality of switches and of endpoints. RapidIO switches only make switching decisions based on explicit source/destination address pairs and explicit priorities. This allows RapidIO endpoints to add new transaction types without requiring changes or enhancements to the switch devices.

As more and more of the system becomes integrated on to a single piece of silicon, PCI Express and even Ethernet are being integrated within the processors -- now properly called SOCs. This integration however has not changed the nature of the transactions provided by these interconnects.

The new RapidIO 3.0 specification, also know as 10xN, increases single lane speed to 10 Gbit/s and support up to 16 lanes operating in parallel for up to 160 Gbps of unidirectional bandwidth for connectivity in systems. The RapidIO 3.0 specification is also designed to operate over the same reach and connector topologies as supported by the current 2.x specifications, meaning support for standard backplane and connector environments such as ATCA and VPX.

RapidIO is supported by a robust ecosystem of switch, IP, processor and FPGA vendors and is used widely in wireless infrastructure and high performance embedded equipment with tens of millions of ports shipped every year.

What does all this mean?

The RapidIO value proposition has been well known in the embedded market for many years. The opportunity now presents itself to take this same value proposition to more mainstream data-processing markets, which are evolving to demand many of the same system attributes that communications networks have long required.

With the introduction of virtualization, ARM-based servers and highly integrated SOC devices, we are now seeing the next stage of the evolution of high performance computing. This evolution is towards more tightly coupled clusters of processors that represent processing farms built to host hundreds to thousands of virtual machines. These processor clusters will be composed of up to thousands of multicore SOC devices connected though high performance, low latency processor interconnects. The more efficient this interconnect is the better it will be for the performance and economics of the system.

Technologies such as PCI Express and 10G Ethernet are certainly not going away any time soon, but they also will not be the foundation for these future tightly coupled computing systems. PCI Express is not a fabric and can really only support the connectivity of small numbers of processors and/or peripherals. It would only serve as a bridge to a fabric gateway device. 10G Ethernet can be used as a fabric, but has significant hardware and software protocol processing

requirements. Its widely variable frame sizes (46 Bytes to 9000 bytes for Jumbo frames) drive the need for both very fast processing logic to support lots of small packets and very large buffers to support very large packets in end-points and switches. The use of PCI Express or 10G Ethernet will either restrict the topologies and connectivity available or add cost and overhead to the solution. These drawbacks create opportunities for proprietary fabrics or for open solutions like RapidIO in this very interesting new market.

Implementing integrated server, storage and networking systems present an opportunity for OEMs to innovate. A key component of that innovation will be the internal system connectivity. RapidIO is a mature well-proven technology and has the attributes required for success in this market. As was the case for wireless infrastructure where RapidIO went from early innovators to becoming the de facto base station interconnect standard, RapidIO's biggest challenge in server, storage and HPC will be to cross the chasm from today's innovators and early adopter markets to mass market proliferation.