RapidIO

White Paper

RapidIO Technology Solves the Communications Fabric Conundrum

The communications fabric silicon market is substantial – IDC projects that this market was $322M in 2003 and is expected to grow at a sturdy 9.5% Compound Annual Growth Rate (CAGR) through 2007. The Linley Group estimates that over 83% of this market is served by proprietary, in-house solutions. Switch fabrics represent the last major bastion of proprietary silicon in networking equipment, but recent dynamics are disrupting the status quo. Networking equipment companies have emerged from the recent industry downturn with smaller design staffs and a greater willingness to outsource activities that do not provide competitive differentiation. Merchant fabric silicon continues to mature and now competes favorably with in-house fabric solutions. These merchant fabric silicon vendors are also steadily turning away from proprietary backplane protocols and toward open standards. This last dynamic allows fabric silicon providers to better deliver on their presumed value proposition: lower technical and business risk due to outsourcing, the proliferation of end-points with integrated fabric interfaces, and economic leverage reaped from volume markets.

While business conditions are influencing equipment vendors to consider adopting an open standard fabric, multiple different technologies are vying to be the fabric technology of choice for networking and communications. The contenders include RapidIO interconnect, Advanced Switching Interconnect (ASI), Ethernet, and HyperTransport™. The proliferation of contenders has created something of a crisis in the embedded industry. OEMs can no longer afford to invest in proprietary fabrics, but until a clear winner emerges, the choice of an open fabric can seem risky. To be sure, this is a decision in which an OEM is *betting their company's future*. They absolutely cannot afford to be wrong.

In responding to these market developments, the RapidIO Trade Association has been extending the capabilities of the RapidIO architecture to enable it to serve as a fully functional, open communications fabric, replacing proprietary fabrics that dominate the communications space today.

Point-to-Point Connections vs. Communications Fabric

There is quite a bit of confusion surrounding the term *fabric*. In truth, many of the technologies that are touting themselves as fabric solutions are little more than simple serial interconnects with few of the features that are required by the demanding fabric applications of today.

To put it simply, an *interconnect* provides a means for exchanging data between a sender and a receiver. i^2C is an interconnect. Processor buses and peripheral buses like PCI are interconnects. Recently, the term interconnect has evolved to also include a new generation of high-speed serial buses that provide point-to-point connectivity between processor and peripheral devices. HyperTransport 2.0, for example, continues to maintain a processor bus focus even after adopting a higher speed physical layer. Similarly, PCI Express strictly adheres to PCI's host/peripheral load store DMA-based architecture on top of a serial physical and link layer. PCI Express is just a serial version of PCI, which, even with its serial physical layer, lacks basic features that are present in modern communications fabrics: source directed routing, a message passing protocol, classes of service, multicast, topological flexibility and much more.

A *communications fabric* must be able to function as a simple point-to-point interconnect and also scale to handle thousands of nodes. The term fabric derives its name from its topological representation. As the data paths between the nodes of a fabric are drawn out, the lines cross so densely that the topology map is analogous to a cloth. Figure 1 shows the breadth of application of a communications fabric.

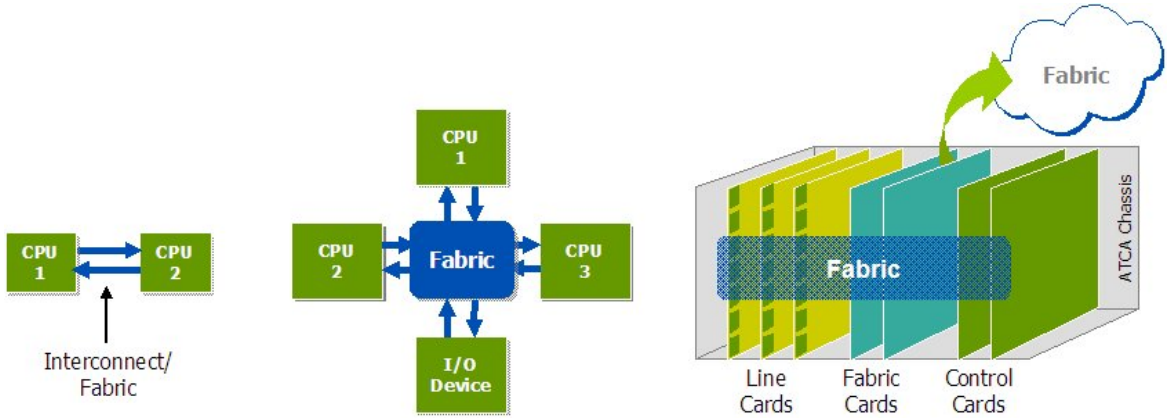

***Figure 1**: Communications Fabric Use – Chip-to-Chip through Chassis-to-Chassis Support*

Fabric architectures are used in many different market segments. To date, the architectural requirements for fabrics have been so demanding that equipment vendors have passed over commercially available technologies like PCI and Ethernet in favor of their own proprietary solutions. This has been the case for fabrics in both the high-end embedded computing and communications markets. While compute fabrics have begun to migrate to open standards, communications fabrics have resisted.

RapidIO Is the Embedded Fabric of Choice

There are a large cast of interconnect standards on the market today, but few even begin to address the requirements of an open communications fabric. In fact, the RapidIO standard is the only one that is mature and scaling to specifically address the full requirements of the communications fabric space. The new capabilities are seamlessly added into the elegant framework of the RapidIO specification architecture:

- **Flow Control** Logical Layer Extensions Specification
- **Data Streaming** Logical Layer Extension Specifications
  - -Interworking/Encapsulation
  - -Traffic Management
- **Multicast** Extensions Specifications
- **Serial Physical Layer** Specification
- ***Next Generation*** **Physical Layer** Specifications

Due to the RapidIO standard's modular and extensible architecture, these new specifications are designed to be fully interoperable with other parts of the RapidIO standard. Further, all of these new features may be adopted individually or not at all. Figure 2 shows the RapidIO layered architecture.

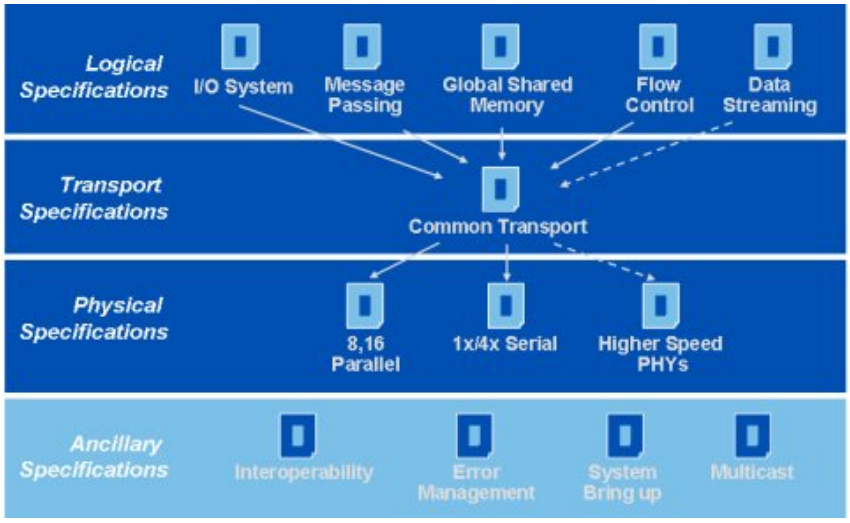

Figure 2: *RapidIO Architecture Layers and Associated Specifications*

The RapidIO Serial Physical Layer was ratified in late 2001. In September of 2003, the RapidIO Trade Association released a logical layer extension that added *Flow Control* functionality to the base specification. Flow control provides congestion control for medium-utilization data plane applications using the RapidIO interconnect architecture. The flow control extensions were driven by OEMs who were beginning to see a role for the RapidIO interconnect in wireless infrastructure, media gateways and other access equipment.

The other significant additions that enables system vendors to take full advantage of the RapidIO interconnect as a communications fabric is the *Data Streaming* logical layer specification and the *Multicast* specification. The Data Streaming logical layer consists of two parts: Phase I delivers the Interworking specification and Phase II will deliver the Traffic Management specification. Phase I was released in August of 2004. The Multicast extensions specification provides a defined mechanism to use RapidIO device IDs to serve as multicast group identifiers allowing switches to elaborate packets to any set of one or more of their output ports. The elegance of device ID-based routing, as opposed to other schemes like path-based routing, is that a single routing architecture can be used for both unicast and multicast traffic.

The RapidIO Trade Association will also continue to leverage the physical layer standards established by the communication infrastructure ecosystem such as those being defined in the Optical Internetworking Forum (OIF). This will scale RapidIO serial physical links to OC-192 rates and beyond. The RapidIO architecture has no inherent limitations preventing it from scaling indefinitely into the future following the industry requirements. The *Physical Layer* specifications will not only scale in speed, but also in their feature set as they evolve to meet the needs of demanding, high-performance embedded computing applications.

How RapidIO Technology Meets the Requirements of a Communications Fabric

In identifying the requirements of the RapidIO architecture as an open standard fabric, the RapidIO Trade Association examined what features were needed that would provide more extensive technical benefits than the proprietary solutions in production today, plus provide the benefits of an open standard. These features included:

- Architectural independence
- Carrier grade
- Advanced traffic management
- High performance
- Scalability
- The right ecosystem

These fabric features – and how the RapidIO architecture in general and the new extensions specifically address them – are discussed in the following sections.

Fabric Requirement #1: Architectural Independence

A fabric must not be tied to a particular hardware or software architecture. Ethernet is an example of a technology with a high-level of software dependence. An application generally cannot access an Ethernet packet without running a networking stack. This often requires the addition of a processor – not an issue when the end-point node is a desktop PC, but in embedded applications, additional processors raise system cost. Similarly, some architectures intrinsically suffer from a high-level of hardware architecture dependency. For example, transmitting an architecturally independent entity (like a packet/cell/frame) over an architecturally dependent memory mapped bus (that uses a common address space for all devices) runs counter to the natural *forward* evolution of communications architectures.

A communications fabric must also support direct peer-to-peer transactions and not be tied to a particular network topology (such as dual-star, mesh, ring, daisy-chain, or tree). In particular, interconnects that function in a tree topology force all transactions to flow through a common switch or CPU complex, which is not optimal for a fabric. A fabric must also support the spectrum of chip-to-chip, board-to-board, mezzanine, backplane and chassis-to-chassis mechanical standards.

Architectural independence also means that a fabric should be protocol agnostic. In the past, there was a belief that communications applications would converge around a single protocol. In reality, convergence has meant that OEMs are forced to support many protocols within a single system design: Ethernet/ Point-to-Point Protocol (PPP), UTOPIA/ATM, Packet over SONET (PoS), CSIX, and so on. A fabric has to provide the semantics to support the encapsulation, transformation and transport of all major networking protocols. This starts with heterogeneous traffic support but also includes support for both variable and fixed-size payloads, segmentation and reassembly (SARing) of large Protocol Data Units (PDUs), and multicast traffic.

Finally, a fabric must have a simple and clean separation between its physical, transport and logical layers. It must be easily extendible so that new features can be added without breaking the integrity of the original architecture. In particular, a fabric must be easily adaptable to parallel and serial physical layer architectures, as well as different physical mediums: copper traces, optical cable or whatever the application requires.

RapidIO technology was designed to be flexible and agnostic in terms of network architectures and protocol support. In addition, the layered RapidIO architecture is extensible and adaptable, enabling new features and physical layer technologies to be implemented without disrupting the integrity of the architecture. The following table describes how the new extensions in particular support the RapidIO value-proposition of architectural independence for communications applications.

RapidIO Extensions Addressing "Architectural Independence"	
Transport Very Large and Very Small Protocol Data Unit (PDU) Sizes	The Data Streaming specification extends the maximum PDU size supported by the RapidIO fabric to match that of the ubiquitous IP protocol: 64kB. Equally important, is the RapidIO architecture's ability to work with small PDUs: 32 bytes. For those who have designed latency-sensitive systems, the requirement for a small PDU is clear. When large packets are used, it may take so long to fill up the payload with real data that the data expires before it is sent. An alternative is to send a packet partly empty, but this compromise degrades fabric efficiency. This is one of the challenges facing designers who implement VoIP, which has a minimum packet size of 64 bytes encapsulated in a relatively large IP and Ethernet header. While a single Ethernet interface may have plenty of bandwidth for a single point-to-point VoIP flow, the fabric system architect has to consider the aggregation of millions of these flows and how payload inefficiency affects network utilization deeper in the network.
Support for a Maximum Transfer Unit (MTU) Size	Large PDUs can require a long time to send over a fabric link. This is especially a problem if smaller packets are blocked behind a large packet. The ATM standard created a fixed 53-byte cell precisely to avoid this and other problems associated with variable sized packets. Modern fabrics need to be able to divide large PDUs into segments and reassemble them when necessary. The Data Streaming Logical Layer supports SARing using an application defined segment, or MTU, size. The segment size is 32 bytes to 256 bytes (in increments of 4 bytes) and may be configured on a per-end-point basis if required by the application. This flexibility allows the application, by convention, to determine whether the fabric will transmit variable or fixed length packets. The RapidIO architecture allows the application to assign a specific ID to each segmentation. This ID, or segmentation context, allows a destination end-point to separate incoming flows of different PDU segments. The maximum size of 256 bytes was chosen because this is the point at which the protocol reaches peak efficiency. Transferring more would only add the burden of larger packet buffers, more bits spent on error coverage, and longer periods of fabric blocking in the case of mixed sized traffic.
Low Overhead	Segmented PDUs are transmitted using a *start*, *continuation* and *end* packet logical protocol. There is a maximum of one start and one end segment, but possibly many continuation segments. For this reason, the architects of the Data Streaming standard took special care to minimize the header overhead of the continuation segment. The header of the continuation segment is only 20-bits in length. This competes favorably with other interconnect protocols in terms of maximizing useful throughput and minimizing overhead.
Support for Encapsulation and Interworking	The Data Streaming Logical Layer provides an architectural framework that supports both encapsulation and transformation of common networking protocols like CSIX, Ethernet, UTOPIA-2/3, SPI-3/PL-3, SPI-4, and so on. The RapidIO architecture is protocol agnostic. The payload has no inherent semantic. This allows the system designer flexibility in locating protocol specific support either intrinsically in the fabric or at the edge of the fabric in end-points.

Fabric Requirement #2: Carrier Grade

A fabric must be reliable and robust. It must support performance management features, which allow a fabric manager (typically a host processor) to investigate and monitor the status of the fabric. The fabric must contain semantics for event notification and handling. The fabric must support common fault management scenarios such as failure detection, hot swap, redundancy and fault tolerance. Service providers typically have stringent requirements pertaining to planned and unplanned maintenance and these restrictions must be explicitly supported in the fabric architecture around which the system is designed.

RapidIO technology embodies the notion of reliability. It supports robust error detection with hardware-based recovery mechanisms. At the physical layer, each packet is explicitly acknowledged on a link-by-link basis. Packets are covered end to end with a Cyclic Redundancy Check (CRC). None of the bits covered under this CRC are changed thus making it invariant across the system. The RapidIO interconnect has a hardware-based recovery mechanism, which attempts to retransmit a bad packet or resynchronize a link that is out of sequence. Even when a link is idle it is always sending link status information. This means that failure of a link is immediately detected and corrective action taken even when idle.

In addition, the RapidIO architecture supports hot-swap and redundant links. Multiple host system discovery and maintenance can be done using in-band maintenance transactions. This allows system implementers to have redundancy in their fabric maintenance. The RapidIO standard defines an error management programming model (including error logging registers, error threshold counters, and so) that allows software implementers to be able to rely on uniformity of specific error reporting registers across endpoints and switches.

Fabric Requirement #3: Advanced Traffic Management

A fabric must be able to support Classes of Service (CoS). Traffic classes have unique requirements: some classes are sensitive to latency (voice and video), some classes are bursty (data), other classes have minimum or maximum throughput profiles (service level agreements). A fabric architecture must support 256 classes in order to capture the full semantic of the class fields of common networking protocols (IPv4 Type_of_Service, IPv6 Traffic_Class, CSIX Class, etc). Interconnect architectures that do not support 256 classes, may be quickly outgrown as protocols like IP continue to evolve new uses for their traffic class semantics.

Additionally, a fabric must support millions of flows. Flows can be used for a number of important purposes; they can represent anything from PHYs to traffic types to individual users. Modern systems have many PHYs. For example, UTOPIA extended addressing provides support for up to 124 unique MPHYs per port. In a multi-slot system, this could mean supporting several hundreds or even thousands of MPHYs. The fabric architecture needs to support the concept of PHY identification without burdening these relatively simple devices with the overhead of full end-point address support. The inability for an interconnect architecture to separate traffic into flows would prevent it from supporting the policing and shaping operations that a fabric switch must perform in order to provide intelligent non-blocking support. Further, flow-based management of traffic is required in order to support graceful degradation when a fabric becomes congested.

Lastly, a fabric must support end-to-end flow control. Traffic sources can, for example, send a large overwhelming burst of traffic to a PHY. In this canonical case, traffic can back-up within the various buffers and FIFOs within the fabric, and block critical pathways. A typical solution is for the PHY to send flow control messages to traffic sources before PHY resources are completely allocated. This signals transmitting devices to refrain from sending more traffic. Ethernet, for example, has no flow-based flow control semantic, so an Ethernet device has no way to receive notification of congestion within the fabric until the fabric blocks and a link-level flow control is issued. This may be one of the main reasons why Ethernet, despite its unparalleled success as a networking technology, has never been more than a niche player in the fabric market.

Figure 3 summarizes how a simple message passing architecture can achieve a roughly 50% utilization rate of a link. Some implementations rely on over-provisioning (provisioning for peak bandwidth) to compensate for a lack of traffic management support. While this is acceptable in some applications, others want to squeeze every last bit of efficiency out of their network. Flow control for a small number of traffic classes can increase efficiency above 50%. To achieve 90% utilization of the network, the fabric must add advanced traffic management features like end-to-end, flow-based and class-based flow control and hundreds of traffic classes and thousands of flows.

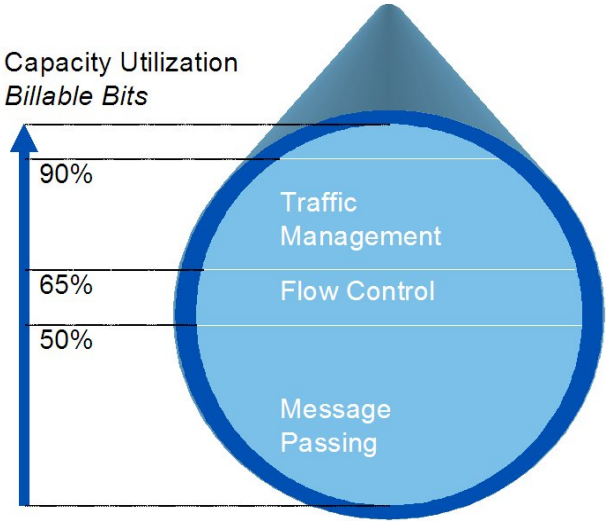

***Figure 3**: Getting the Most out of the Fabric*

The following table defines the traffic management capabilities of the new extensions.

RapidIO Extensions Addressing "Traffic Management"	
Support Classes of Service	Data plane architectures classify PDUs into traffic classes that have distinct transport requirements. Some classes are sensitive to latency like voice or video. Other classes can be bursty like data traffic, for example. In other examples, service providers may want to establish a service agreement whereby subscribers pay for differing levels of service. The fabric has to be able to police and shape traffic to conform to the desired requirements. Low-end access equipment, such as DSLAMs, are beginning to handle many traffic types (data, video, gaming, and so on) and in some cases are requesting that fabrics be able to shape 32 or more traffic classes. Applications that do not need that many classes today, may still prefer to choose an architecture that provides room to grow. One thing is certain when discussing next generation communications architectures, demand for traffic class support is likely to only increase – never decrease. As mentioned earlier, the CSIX protocol as well as IPv4/IPv6 contain protocol specific fields in their headers for 256 distinct traffic classes. The Data Streaming Logical Layer is able to transport these class values without losing information in its 8-bit *Class* field.
Provide Millions of Flows	A *Stream* is defined as a persistent relationship between unique source and destination devices. This relationship is application specific or even device specific. In practice, a stream may have specific traffic management requirements or be associated with a specific flow such as transmission to/from a PHY. The Data Streaming Logical Layer allows 64k streams to be defined between any source and destination pair. This effectively allows the fabric to represent millions of unique streams. Apart from traffic classes, streams are the fundamental unit of traffic management within the new extensions. They allow the fabric to manage congestion by separating the mass of flowing PDUs into thousands or perhaps millions of individual flows with specific priorities, latency requirements and throughput requirements. Streams allow a fabric to manage congestion intelligently by viewing traffic as a series of flows that can be policed and shaped according to any desired heuristic.
Enable End-to-end Flow Control	One basic mechanism that allows the fabric to avoid blocking is flow control. When fabric resources start to become oversubscribed, the fabric may issue flow control messages to traffic sources telling them to stop transmitting certain streams. Lower priority flows are signaled first. When fabric resources become available, flow control messages are sent to stopped flows to restart them.

Fabric Requirement #4: High Performance

Advanced traffic management features directly contribute to fabric performance. Traffic management can prevent a burst of packets from causing a fabric to block. This is critical to maintaining performance. Traffic management can also enforce performance guarantees for specific classes or flows.

Low-end fabrics that run without the benefit of flow control or traffic management features rely on brute-force over-provisioning. In other words, the fabric provides enough bandwidth to meet the peek requirements of the application. However, throughput isn't everything. Fabric switches must also control latency. This is critical for voice applications that have to deliver real-time data. Further, fabric architects pay a great deal of attention to the ratio of header to payload. For example, the overhead represented by Ethernet's MAC layer, the IP network layer, and the TCP transport layer represents a minimum of 86 bytes of "wrapper" around the data payload. Control plane traffic tends to consist of short data transfers. These small transfers are heavily penalized by the protocol overhead. You can be sure that service providers study this issue quite seriously – *bill-able bits* are their business. Every bit that is wasted on header or overhead is rightfully considered a lost revenue opportunity. As mentioned earlier, the Data Streaming logical layer has a deliberately compact header: 36 bits for the start and end segments and 20 bits for the continuation segment. When SAR'ing large PDUs, the continuation segment is reused so that efficiency increases as the PDU size increases.

RapidIO technology was also designed to provide ample bandwidth for fabric-quality forwarding. The RapidIO serial physical layer is based upon the XAUI (10 Gigabit Ethernet Attachment Unit Interface) specification. Serial RapidIO, runs at 3.125 Gbaud today, 25% faster than the 2.5 Gbaud clock supported by PCI Express and ASI and over three times faster than a Gigabit Ethernet interface.

Fabric Requirement #5: Scalability

Scalability is related to performance, but it is broader. A fabric must be able to scale in the throughput domain: from very low-cost, low-power applications all the way up to very high-end, high-throughput systems. The Serial RapidIO physical layer supports three speeds: 1.25GHz, 2.5 GHz and 3.125 GHz. Engineers intuitively understand that higher speed clocks imply higher power consumption. RapidIO technology is unique in allowing the user to scale the speed of the physical layer to support power sensitive applications. To gain additional performance, at the high-end, using existing SerDes technology, serial lanes are combined to create a by-N port, often called *striping*. RapidIO technology supports a 4-lane or 10Gbps (after 8/10 encoding) link.

Fabrics must also support thousands of end-points in order to scale to the needs of high-end applications. Many years ago, when Ethernet battled Token Ring, there was much discussion surrounding device-based routing (the Ethernet MAC) versus Token Ring's path-based routing scheme. Obviously, Ethernet won the battle and the war. Architectures that use path-based routing may contend with some of the same issues that determined Token Ring's destiny at that time.

In path-based routing, end-points must maintain a real-time database of the entire topological map of the fabric. Multicast can be tricky to implement. Also, hot swap events which change the topology can result in a control plane storm to all end-points. Device-based routing is much more straightforward. A change in topology requires updates only to the nearest neighbors of the device that is changing. Further, multicast is easily implemented with bit masks that are associated with a particular destination device id.

Device-based routing simplifies end-points and switches. There are up to 64k RapidIO devices in a fabric. The switch often maintains a simple look-up table associating each destination ID with one or more ports. Worst case latency is, of course, implementation dependent, but RapidIO switches can forward a unicast packet in as little as a few hundred nanoseconds.

Fabric Requirement #6: The Right Ecosystem

One common myth is that embedded and communications systems design with the same parts as commercial desktops, laptops and servers. In fact, quite the opposite is true. If one were to compare the bills of materials of a commercial desktop PC with a CompactPCI motherboard, for example, the two would share very few actual components. The reason is that the embedded and communications markets have specific non-negotiable needs: long product life cycles, industrial qualification and reliability, and industrial temperature ranges. Solutions that are targeted at the commercial markets categorically do not address these requirements. Technologies from commercial computing may be used in embedded computing, but for the most part, the actual parts used are manufactured specifically for embedded customers. It is not enough to have any ecosystem, even if it is large. A technology has to have the *right ecosystem*. This means support from the broad community of vendors who are focused on the unique needs of the embedded systems market.

The RapidIO Trade Association comprises some of the leading system OEMs, silicon manufacturers, and software vendors in embedded computing and communications. These vendors are leading the charge with the RapidIO ecosystem. Processors, switches, boards and systems, FPGAs, and ASIC devices are available with RapidIO technology today, and several companies have shipped in production volumes.

Conclusion

The fabric market is dominated by proprietary solutions, but as merchant switch fabric suppliers turn to open fabric standards, they are likely to increase their market penetration. Ethernet has been available for many years, but has not achieved critical mass due to its lack of features like flow/class-based flow control and its high software architecture dependence. HyperTransport is an interconnect standard that focuses its value proposition on being a processor bus. PCI Express lacks class of service, peer-to-peer transaction support and architectural independence. ASI is targeted more for low-end backplanes, lacking architectural support for traffic management, which will prevent it from scaling to high-end fabrics. ASI is in truth more of a switched interconnect than a true fabric.

The RapidIO architecture presents a strong value proposition in seeking to win the backplanes and fabrics of future communications equipment. With the help of the new extensions, RapidIO technology can scale from very cost/power sensitive low-end local bus applications to high-performance fabrics. The new specifications add features that modern fabrics require including interworking, traffic management and multicast support. In addition, the new specifications solidify the RapidIO architecture's position as the premier open-standard interconnect and fabric for embedded and communications applications.

RapidIO®
www.rapidio.org