



RAPIDIO[®] CONNECTIONS

RAPIDIO TECHNOLOGY & QUALITY OF SERVICE

By Tom Cox, Executive Director, RapidIO Trade Association

Quality of service (QoS) is an inherent part of the RapidIO specification, implemented directly in hardware and enabling traffic to be classified into as many as six prioritized logical flows. While the mechanism for forward progress in the fabric relies upon ordering rules at the physical layer to give responses higher priority, the degree to which prioritization results in lower average latency or jitter for a particular flow is specific to the actual implementation. For example, more aggressive switches might make ordering decisions based upon a flow's priority, source, and destination ID fields while less aggressive designs might only utilize the priority field.

QoS is also affected by specific fabric arbitration policies. While the specification explicitly defines prioritized flows, developers are free to choose the particular arbitration policies to put into place to prevent starvation of lower-priority flows, such as the well-known leaky-bucket scheme. As even the least aggressive design must support these mechanisms, higher-priority flows are guaranteed to demonstrate better lower-average latency.

For applications requiring even more aggressive and effective QoS, advanced flow control and data plane capabilities are available. The RapidIO protocol defines multiple flow control mechanisms at the physical and logical layers. By managing physical layer flow control at the link layer, short-term congestion events are effectively managed for serial and parallel applications using both receiver- and transmitter-controlled flow control. Longer-term congestion is controlled at the logical layer using XOFF and XON messages which enable the receiver to stop the flow of packets when congestion is detected along a particular flow.

Receiver-only flow control, where the transmitter does not know the state of receiver buffers and the receiver alone determines whether packets are accepted or rejected based on receiver buffer availability, results in packets being resent, creating wasted link bandwidth. Additionally, ordering rules require a switch to send higher-priority packets before resending any packets associated with a retry, aggravating worst-case latency for lower priority packets.

Transmitter-based flow control avoids bandwidth wasting retries by enabling the transmitter to decide whether to transmit a packet based on receiver buffer status. Through

receiver buffer status messages sent to the transmitter using normal control symbols, the transmitter is able to limit transmissions within the maximum number of buffers available at the receiver. In general, priority watermarks at the various buffer levels are used to determine when the transmitter can transfer packets with a given priority.

A third link-level mechanism is available within the parallel PHY specification which enables the receiver to throttle the packet transmission rate by requesting that the transmitter insert a selectable number of idle control symbols before resuming transmission of packets.

Revision 1.3 of the RapidIO specification achieves further efficiency and higher throughput through the introduction of data plane extensions. Since data plane fabrics can carry multiple data protocols, these extensions enable the encapsulation of virtually any protocol using a data streaming transaction type with a payload up to 64 Kbytes. Hardware-based SAR support is expected for most implementations, with up to 256 classes of service and 64 K streams.

The upcoming 2.0 revision of the specification builds on revision 1.3 capabilities, introducing a new 5.0 Gbaud and 6.25 Gbaud PHY, lane widths up to 16x, 8 virtual channels with either reliable or best-effort delivery policies, enhanced link-layer flow control, and end-to-end traffic management with up to 16 million unique virtual streams between any two endpoints.

The RapidIO protocol is a simple and efficient interconnect designed specifically for high-speed embedded applications and appropriate to serve as a system-level fabric. By implementing protocol processing in hardware, many quality of service and flow control mechanisms are an inherent part of the PHY, maximizing efficiency and throughput while minimizing latency and switch complexity. Backed by new data plane extensions which enable RapidIO switches to encapsulate virtually any data protocol, the RapidIO specification is an ideal interconnect technology, enabling developers to consolidate interconnect layers, as well as both control and data planes, into a single fabric, reducing cost while increasing overall system reliability. For more information: <http://www.rapidio.org/specs/current>.

INDUSTRY INSIGHTS

RAPIDIO MAKES ITS CASE IN THE MERCHANT MILITARY SUPPLIERS MARKET

By Chris Kissel, Research Analyst Emerging Technologies, In-Stat

In the United States (and probably in other countries) there are two distinct business models for the defense industry. The first business model is the formal military/industrial complex. These enterprises include dedicated contractors, like General Dynamics, Raytheon, and Lockheed Martin. Also as a part of this definition would be weapons and systems development done by the service branches themselves; the Army Corps of Engineers would be an example of this idea.

The second business model would be systems developed by private contractors. Companies that would be included in this category would be Mercury Computer Systems, DRS Technologies, L3 TRL Electronics, WB Electronics, and Motorola Computer Group. The formal term used to describe these companies is merchant military suppliers.

There are differences between the two technology groups. What differentiates the merchant military suppliers from the military/industrial complex is that the merchant military suppliers are more likely to use a combination of proprietary systems, as well as commercial-off-the-shelf (COTS) software and hardware. The bigger contractors gravitate toward the larger, closed systems that do not use standards-based protocols or components. A Booz-Allen model of the US military spending model suggests that the US defense industry will spend roughly US \$14 billion in COTS hardware and software. Of this total, somewhere between 5-10% of this spending is available to the merchant military suppliers.

There has been a unifying movement within the merchant military supplier market. Motorola and Mercury Computer Systems were instrumental in developing the RapidIO standard. Many of their solutions used the Serial RapidIO interconnect. A key competitor of those companies, Curtiss-Wright, has written a white paper advocating SRIO as the best IO for multiple processor applications. Serial RapidIO is becoming a more common interconnection in the US defense industry.

Derivatives of the Versa Module Eurocard are the most used board/connectors in the defense industry. The VPX format specification (comprising VITA 46 and its complement VPX-REDI/VITA 48), establishes a COTS standard that harnesses the performance of modern, high-speed serial interfaces. The standard defines VME- and CompactPCI-compatible 3U- and 6U-sized modules that use a modern backplane connector capable of handling the signaling rates of today's high-speed serial fabrics, such as Serial RapidIO. The VPX standard is based on the concept of a "core fabric" connector that is intended as the board-to-board communications medium, also known as the "switched serial backplane."

PowerPC computing components are the dominating processors for merchant military suppliers. Additionally, many of the applications for the defense industry use a LINUX operating system (OS). The convergence of board/connectors, PowerPC and Linux OS, all optimized for RapidIO, make for a compelling communications or logistics platform.

One of the publicly disclosed weapons systems provided by the merchant military suppliers is Global Hawk. The Global Hawk is an unmanned aerial vehicle (UAV). The UAV is used for reconnaissance. Using synthetic aperture radar (SAR), The UAV can provide high-resolution images even through cloud-cover and sandstorms. The Global Hawk can survey as much as 100,000 square kilometers (40,000 square miles) of terrain a day. Germane to this article, Global Hawk is using SRIO for its system I/O.

In-Stat estimates that Serial RapidIO is an interconnection used in roughly 20% of all processors provided in the defense industry. As compelling as Serial RapidIO is as an on-board interconnect and a board-to-board interconnection, in the defense industry, some board designs change over 5-10 year increments. Even so, In-Stat sees Serial RapidIO as an interconnection for 35% of all processors used in the military merchant defense industry by 2009. For more market share information

visit: http://www.rapidio.org/switchfabrics_whitepaper

TECHNICAL INSIGHTS

SOFTWARE CONSIDERATIONS AROUND RAPIDIO DESIGNS – PART ONE

By Jim Parisien, Fabric Embedded Tools (FET)

This part one of a two-part article provides an understanding of the basic information needed for successful board and system level bring up. Included is a discussion of the high availability features built in to RapidIO that aid in debugging both software and hardware in a RapidIO system.

Discovery in RapidIO is required to explore a network without knowing its configuration, and assign each “element” a unique device identifier or Device ID. RapidIO supports any system topology as well – Tree, Star, ring and full mesh. As such, the Discovery process involves identifying all of the Processing Elements (PEs) in a network, understanding the system topology, and enumerating each endpoint with a unique Destination ID. The Destination ID register is one of the fundamental maintenance registers within any endpoint, along with a number of others discussed below.

Is Discovery required for a RapidIO system? No. System Discovery is used only in systems where the configuration is unknown, such as a system where a user can put any set of RapidIO-based blades into a chassis. In a static system, however where PEs are always connected in the same manner through the same switches software can initialize switches and endpoints with a known configuration. Many PEs support power up configuration options where routing tables, Destination IDs, and other parameters are pre-set from non-volatile memory such as I2C EPROMs, eliminating the need for software.

System Discovery is the process of determining what types of PEs are in the system, how they are interconnected, and configuring them for operation. Let’s look at Discovery, Enumeration, and Initialization separately.

Discovery

In its simplest form Discovery is the process of interrogating each switch and endpoint in a system. From this, designers can determine the specific capabilities of each PE, how each is interconnected, and get an understanding of the network topology. If the first PE has another endpoint, then the Discovery process is completed. However if the first PE is a switch, more exploration is needed.

How do you interrogate the first PE in order to discover a RapidIO network from a processor somewhere within a network? This requires Maintenance Transactions and an understanding of how they are used.

RapidIO uses a Destination-based routing methodology. Therefore, virtually all packets contain a Destination ID in the header. For standard I/O transactions this represents the endpoint. For a Maintenance Transaction it represents a direction along the path to a specific endpoint. For Maintenance Transactions, a second parameter called a Hop Count is

decremented by each switch along the way and used to specify the packet's final destination. Routing tables in switches are simply a switch port number lookup table that uses Destination ID as an index.

Before Discovery, an unknown endpoint is an endpoint that is required to respond to any maintenance transaction using a Destination ID of Hex FF following power up. This is referred to as Promiscuous Mode. Small transport systems support 8-bit Destination IDs, while large transport systems support 16-bit Destination IDs. This article covers small transport systems; however, a large transport system is exactly the same, but uses Hex FFFF.

With a four-port switch and four endpoints in a small transport system, for example, the first transaction could be a maintenance read using a Destination ID 0xFF and a hop count of 0 to the Processing Element Features Capability Register (CAR).

CAR helps identify the type of PE – processor, switch, memory or bridge. In this example, it is a switch. One requirement of a Discovery algorithm is knowing whether or not it is intended to operate within a multi-host environment. If so, designers must first determine if another host has already discovered the device. If so, it's important to know if that host is a higher priority host, or if it has already enumerated the system.

The RapidIO specification facilitates this aspect of Discovery using a Host Based Device ID Lock register and the "Discovered" bit within the Port General Control and Status Registers (CSR). The Discovered bit is a simple status bit, that if set, will clearly indicate if another host has encountered this PE. The Host Base Device ID Lock register is a semaphore mechanism that allows a host to determine ownership of a PE or system.

For the purposes of the one switch / four endpoint example, assume that no other host has claim over the found switch, then lock the device and set the Discovered bit. The next steps require an understanding of how many ports the switch has, which port of the switch has been entered, and at which port to begin to exploration. To do this, a maintenance read is issued to the Switch Port Information CAR. The bit-fields within this register will provide the information. The port chosen is arbitrary. Assume the entry is through Port 1. Explore each of the other ports on the switch in sequential order from lowest to highest port, starting with what lies beyond Port 0. At this point in the Discovery algorithm, there may be a number of other capability registers in preparation for the third step of initialization.

In order to discover what exists out Port 0, issue a maintenance packet with a Destination ID of 0xFF and hop count of 1. First, configure the routing table in the switch to "steer" the next maintenance packet out Port 0. There are two ways to configure a switch's routing table, Standard and Optional Proprietary methods. The standard interface allows one to specify the output port number for any given Destination ID in one global lookup table. There are a number of other details associated with routing tables such as default port for non-programmed entries.

In this example, the entire routing table is not ready to configure as the entire system has not yet been discovered. Nonetheless, the next maintenance packets need to be forced out Port 0. Using a Destination ID of 0xFF, simply write an 0xFF into the Standard Route Configuration Destination ID Select CSR and an 0x0 into the Standard Route Configuration Port Select CSR. Again, look at the simplest form of routing table configuration so consult

the User Manual for the switch that may be used for more advanced configuration requirements as they all offer different sets of advanced capabilities. Make sure to enter a route for the return path, back to the host being used to discover the system. If the Destination ID of the Host is zero, write a 0 into the Standard Route Configuration Destination ID Select CSR and a 1 (because the switch on Port 1 was entered) into the Standard Route Configuration Port Select CSR.

Before leaving the switch, Discover what the width of the link is by using the Serial Port x Control CSR. Here, many devices can also discover the link speed. Both of these parameters are useful when trying to determine Throughput and Congestion.

Now the routing table is configured to allow the maintenance packet to traverse the switch if the Destination ID of 0xFF and a Hop Count of 0x1 are used. At this point, consider issuing the maintenance packet and attempt to read the Processing Element Features CAR in whatever device that may be connected to Port 0.

If nothing is connected to it a bus timeout will occur after a period of time as no response to the read request arrives. This approach will slow down the Discovery process due to cumulative timeout delays, causing appreciable delays for large systems with many 16-port switches. To make it more time efficient, before issuing any maintenance transactions out port 0, read the Serial Port 0 Error and Status (CSR) and determine if the port has trained. This could save costly bus timeouts and shorten system configuration time considerably.

Presuming the port is trained, issue the maintenance packet read to the Processing Element Features CAR in whatever device may be connected out Port 0. It is an Endpoint, and as with any PE discovered, the test described earlier should be repeated to see if another host has locked the PE and already discovered it. Assuming one has not, lock the PE and set the Discovered bit. Now enumerate it and initialize or configure certain capabilities within it.

The next step is to explore out of the next port of the switch. Simply overwrite the routing table entry of 0xFF to Port 0 with 0xFF to Port 2. Remember, the switch came in on Port 1, so the next Port to explore in the sequence is Port 2. The Discovery algorithm will want to track this. The return path route is the same as used for Port 0. In this process, keep track of the network topology as it is discovered. In some systems you will encounter multiple paths between PEs and many circular loops in full mesh systems. The Discovery algorithm will need to detect when the revisited parts of a network have already discovered and where in the existing known network map it happened to reconnecting to.

The next step is enumeration which will be discussed in the next newsletter. To hear the related podcast, visit <http://www.rapidio.org/wp/?p=25>

[IN THE NEWS](#)

News from RapidIO Trade Association members

[GE Fanuc Intelligent Platforms Announces Worlds First COTS VXS Serial RapidIO Managed Switch](#)

[Curtiss-Wright - New card speeds integration of embedded custom FPGA computing](#)

[IDT Introduces High Performance, Lowest Power Serial RapidIO® Switches for the Embedded Market](#)

[Tundra Semiconductor Introduces Multi-Standard Serial RapidIO Switch with PCI Bridging and FPGA Interface](#)

[CommAgility: AdvancedMC module for wireless baseband applications](#)

[Freescale demo's platform for wireless broadband gear](#)

[Tundra Semiconductor Announces Mercury Computer Systems Sublicenses RapidIO Tsi578 MicroTCA Switching Module](#)

[WiMAX OEMs Spur Deployment with Texas Instruments Wave 2 Software and DSP-based Development Tools](#)

[AMCC and Silicon Turnkey Express Establish Strategic Partnership for Advanced Mezzanine Cards](#)

[Mercury Computer Systems Brings 10th AMC to Market](#)

[Huawei Selects Tundra Semiconductor's Serial RapidIO\(R\) Switch for Next Generation Systems](#)

[Tundra Semiconductors Serial RapidIO® Switch Selected by RadiSys for High Density Processing in Next-Gen Communication Networks](#)

[IDT Pre-Processing Switch Passes RIOLAB Level 1 Device Interoperability Test](#)

RapidIO® is a registered trademark of the RapidIO Trade Association. Product and company names mentioned may be trademarks and/or registered trademarks of their respective holders.