Solving High-Bandwidth, High-Compute-Density, Low-Latency Problems with ATCA

MERCURY WHITE PAPER

Mercury Computer Systems, Inc.

TOUGH COMPUTING PROBLEMS

Advances in technology have brought with them ever-increasing demands for more compute power in multiple application areas. Many, but not all, of these high-performance applications are related to telecommunications, and they face similar computing problems:

- They must deal with unrelenting streams of data, possessing I/O bandwidths that exceed the capacity of standard computing systems, and the data often arrives in a unique format from a specialized sensor.

- Stringent latency requirements must be applied to process these streams of data, making it necessary to bring multiple processors to bear on the problem. Sometimes the processing is complex, involving several phases of computation and requiring the use of multiple, different processor types.

- To maximize overall system performance, compute cycles must be matched by bandwidth capacity, both I/O bandwidth and, for multiprocessor systems, bandwidth among processors. Balance is essential.

- A balanced, high-performance computing system can be difficult to design and time-consuming to integrate. Developing it within a business environment of tight cost constraints and aggressive time-to-market windows is even more challenging.

This white paper delivers a broad technical view of these issues and discusses a standards-based approach to achieving successful solutions.

APPLICATION REQUIREMENTS

High-performance data-stream processing applications, such as voice and video processing applications or industrial control systems, must contend with a continuous stream of high-bandwidth data, often in differing pre-packaged formats that must be processed in real time with low latency and high reliability. A more detailed examination of these characteristics reveals the challenges they represent to application design and deployment.

High-Bandwidth Data Streams

In today's data-stream processing parameters, bandwidths reaching 10 Gigabits per second (Gbps) are considered to be high-bandwidth. For voice-processing applications, 10 Gbps is equivalent to more than 150,000 standard voice channels under the G.711 standard, although data compression techniques can reduce this requirement. Video-processing applications require even more.

With sensor data, such as radar or visual camera data, the incoming data stream can be unrelenting, as, for example, when a camera scans an area. The data must be processed in real time, because there is too much of it to be stored. For example, if 10 Gbps of data is coming in and there is even a minimal delay in processing it, storage (buffering) capacity runs out very quickly. This is also true for incoming voice and video data from wireless networks or base stations, where hundreds of thousands of simultaneous calls are brought from base stations into central processing.

Latency and Determinism

Latency and determinism are critical characteristics of data-stream processing requirements. Processing latency—where latency is units of time measured from ingress port to egress port—must be very low. The parameters that define a low-latency response depend on the application. For example, voice-processing applications must control the signal processing delay across the entire network, including both satellite transmission delays and intra- and inter-system processing delays, to make a phone call understandable. The goal is 200 to 250 milliseconds of maximum delay end-to-end. For industrial control-loop applications, existing in a smaller physical environment, latencies are measured in microseconds rather than milliseconds.

These very low latencies must also be reliable. For the application to perform properly, each data processing step must be performed within a well-known, extremely small window of time, and this window must be the same each time the step is performed. This characteristic is referred to as determinism.

Computational Density

Computational density—the processing power that these applications need to manage their high-bandwidth, low-latency requirements—depends on how much processing must be performed on the data. In general, an application that has a lot of data coming in most likely needs a lot of processing power. However, the amount of processing power it requires can vary by multiple orders of magnitude, depending on what the application needs to do to the data.

For example, an application that receives raw data, and then packages or packetizes it, is not particularly compute-intensive. However, an application that performs complex collation or compression algorithms on a continuous stream of data in real time requires a lot of processing power.

Another factor in processing power requirements is the match between the application and the underlying processor technology. Standard general-purpose processors, digital signal processors (DSPs), and field-programmable gate array (FPGA) processors each specialize in a particular type of processing, and some are better than others for a given processing problem. If an application is processing-intensive, but does not have enough properly matched processing technology, it may need more processing power than it otherwise would.

TECHNICAL APPROACHES TO MEETING THESE REQUIREMENTS

Several components for building systems meet the high-bandwidth, low-latency, and deterministic requirements of high-end data-stream processing applications. A successful system architecture:

- Partitions the application across multiple processors, so that each processing step is handled by the right type of processor
- Uses a switch fabric to move the processing stream from processor to processor at high speeds with great efficiency
- Builds on a framework that maximizes cost control and maintainability over time to satisfy real-world business drivers

Partitioning the Processing Job

The processing that an application performs drives the architecture of the underlying system. Deciding upon the optimal multiprocessor architecture for a specific application's processing requires consideration of the type of processing needed and the amount of data to be processed.

Processing can be pipelined (Figure 1) or carried out in parallel (Figure 2), using a multiprocessor/multicore approach. The type of processing that dominates the application—parallel or pipeline—defines the system architecture. In most cases where a system deploys multiple processors of any type, the system architecture is a mix of both pipelined and parallel processing (Figure 3).

An application (or a part of the application) that runs more effectively using parallel processing needs to send incoming data to multiple processors at the same time. This kind of data movement—transferring data from one point to multiple points in the system at the same time—typically leads to a star topology combined with broadcast/multicast techniques, or can be executed using a mesh infrastructure on the backplane.

If the application relies more on pipeline processing and needs to move data among different boards in the system without actually reaching the system hub, a mesh topology may be more appropriate. However, if the application relies heavily on a single ingress/egress point in the system (the hub becomes a central distribution point of the system), a star topology is more effective.

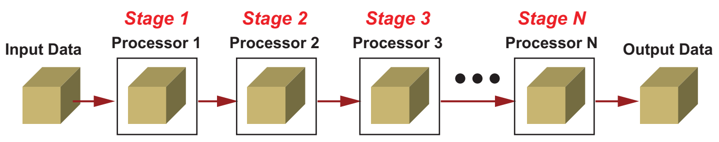

Figure 1. Pipelined processing

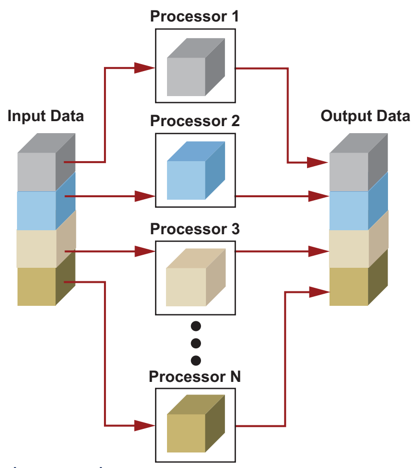

Figure 2. Parallel processing

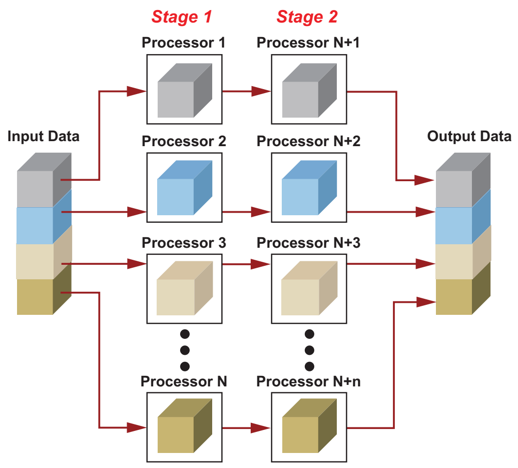

Figure 3. Parallel pipelined processing

Matching Processor Types to Processing Requirements

Different processing elements, such as FPGAs, DSPs, and network processors, are designed to handle different processing requirements.

Field-programmable gate arrays (FPGAs) are most effective for simple mathematical operations like add/multiply, because they perform the operations at the gate level as the data moves through rather than moving the data from memory to a computational unit and back, as other types of processing elements must do. For beamforming applications, which require an enormous number of simultaneous mathematical calculations, FPGAs are far superior to other processor types. Operations that can be performed with Boolean logic are best done on FPGAs. FPGAs are also good for filtering operations, which extract desirable data from an incoming data stream or remove unwanted data. For example, in applications that process antenna-generated data, an FPGA can efficiently filter out carrier information from incoming data channels.

Digital signal processors (DSPs), on the other hand, are very effective for data compression and decompression known as codec operations. In addition, compression and decompression of data is often combined using a pipelined process, with echo cancellation operations, which are another strength of DSPs. Echo cancellation is a critical component of voice-over Internet Protocol. For codec operations, DSPs provide much better performance than standard processors or FPGAs.

Network processors, which are specialized programmable ASICs, are best for in-depth packet analysis. The format of a packet is clearly defined, so the desired information can be extracted efficiently. A network processor in a router, for example, defines in real time where to send an incoming packet, what its priority is, and, therefore, how promptly it needs to be processed. These and many other similar functions of network processors are based on in-depth analysis of the packet content.

When different types of processing must be applied to a data stream, it makes sense to match processing elements to specific processing needs. For example, in voice and video applications, the DSP engine compresses the data, while the network processor on the router identifies packets as voice or data-only, assigns a higher priority to voice packets, and sends them out on the network. Or, in a voice-processing application, an FPGA does waveform processing of the input signal from the antennae, while a network processor behind the FPGA performs packet-level processing.

Using a Switch Fabric for Extremely Fast Data Transfers
To make a partitioned processing job run at optimal speed, the data stream must move between processors with maximum efficiency. The application must be able to rely on data transfers that occur with very low latency in a very deterministic manner. A switch fabric (Figure 4) is superior to bus architecture for this purpose in several ways:

- A switch fabric is fundamentally point-to-point, avoiding bus contention.

- The current generation of switch fabrics is well suited to high-bandwidth applications. At 10 Gbps bandwidth, a serial switch fabric has substantially better performance than a bus.

- Switch fabrics are easier to implement. Backplane routing is simplified, and switch fabrics are significantly better in terms of fault detection and redundancy implementation.

- Using a full mesh backplane allows for contention-less traffic from any source to any destination in the system, allowing data to be distributed from multiple sensor inputs to multiple processing units.

- Latency is very low. For example, using serial RapidIO®, latency is under 1 microsecond for a one-way trip across the backplane between any two endpoints in the system. In comparison, using Gigabit Ethernet, latency is in the 1 millisecond range and up.

- Reliability is high, because switch fabric consists of many point-to-point links. A single failing node does not bring down the entire system.

- A switch fabric is deterministic. Latencies are reliable, so that the arrival of data from point to point is predictable. A bus is undeterministic, unless a sophisticated set of priority control algorithms is built in.

- A switch fabric supports parallel transactions between two elements. With 10 elements, for example, a switch can have 5 simultaneous transactions, while a bus can have only one. A switch itself has potential blocking issues, but they are fewer than those of a full bus, and certain architectures allow for the creation of non-blocking switches.

- A switch fabric is less costly. For a bus, the connector cost—the number of pins on the connector and how to create a backplane with that many lines—is expensive.

- Multicast is easy to implement. Nearly all switching silicon supports this functionality. Multicast is particularly useful when data is processed in parallel. The application can use multicast to send the same data simultaneously to several different processing elements. For example, an application that performs different types of filtering on data arriving from an antenna can multicast the data to the multiple filter processors in the system. A router can use multicast to send a block of packets out for parallel processing. Multicast can also be used in a control plane for system-wide shutdown in the face of heat-related problems or other types of errors, because all affected processing elements simultaneously receive the shutdown message.

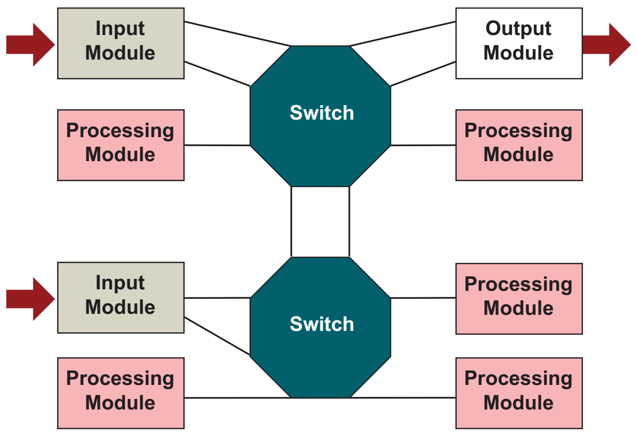

Figure 4. Basic switch fabric architecture

Using a Standards-Based Framework for System Design
Using a standards-based approach, as opposed to proprietary architectures, addresses the real-world business needs of cost control and maintainability. In the competitive business world, a system must be built at minimum cost, and its designers must be confident that they can extend, expand, and modify it over time. This is where computing standards are essential. When a standard is adopted across an industry, competition among suppliers who implement the standard automatically drives down cost, and the openness and transparency provided in a standard assures its users of improvement and change over time.

A standards-based framework has advantages for system flexibility and interoperability:

- Because processing elements from different manufacturers are designed to the same standard, designers have the flexibility to build systems using multiple sources for these processing elements.

- When standards are well defined, equipment integrated into a system from multiple sources can interoperate, either immediately or after some minimal amount of effort. Without a standard, there is no starting point for interoperability nor the ability to take advantage of equipment from different suppliers.

EXAMPLE SOLUTION

High-performance data–stream processing applications typically need a lot of processing power and a lot of I/O capacity. A logical approach to designing a system that meets these demands is to build on a standard that addresses these needs. The Advanced Telecom Computing Architecture (Advanced TCA® or ATCA) was developed precisely for these reasons.

The next step is to couple the ATCA standard with both serial RapidIO and Gigabit Ethernet fabrics to solve the different problems that they address, and then to populate the system with multiple processor types to achieve the various processing goals needed within the application.

ATCA Standard

The ATCA standard was conceived to specify a carrier grade-based system infrastructure. It was built from the ground up to support a range of processors. Its backplane, and the front and rear panels are designed to deal with an unprecedented amount of I/O, and its system management infrastructure is designed to be separate from data-processing activities and to support a complex, layered environment.

I/O capacity. The form factor and architecture of ATCA boards allow an unprecedented amount of I/O connections for front and rear panels. The carrier blades at the front panel allow optical or other types of I/O connections to bring data in and out, while the rear transition module (RTM) form factor provides a large amount of physical space for I/O connections from the rear.

Computational power and flexibility. The ATCA carrier blade form factor supports well-balanced systems delivering teraOPS of processing power in a single sub-rack, and the architecture is flexible as to the types of processors that can co-exist in the system. An advanced mezzanine card (AMC), which connects to a carrier blade, can contain processing elements of very different natures. Processor types can include standard processors—for example, Freescale 8641D PowerPC or Intel single or dual-core processors—DSP engines, FPGAs, or digital-to-analog (DA)/analog-to-digital (AD) converters, if the application needs to handle incoming analog signals or signals that come from some type of measuring equipment. The AMC concept even allows for the creation of highly specialized compute engines for particular applications.

If application requirements change over time, as they often do, a previously deployed AMC can be replaced with one whose processing element more effectively matches the new requirements. AMCs provide the flexibility to process the data and to match it to what the application really needs to do with that data.

System management infrastructure. ATCA specifies a sophisticated intelligent platform management interface (IPMI)-based infrastructure that allows for the construction of a consistent system management environment for alarms, configuration, and diagnostics that can be run on a completely different medium from the application's data and control planes.

Serial RapidIO and Gigabit Ethernet Both Have a Place

The ATCA system defines two distinct fabrics: the data plane and the control plane. ATCA supports a fabric interface for the data plane, and 1 Gigabit Base Ethernet for the control plane. ATCA's data-plane fabric interface can support many different fabrics, including 1 and 10 Gigabit Ethernet and serial RapidIO, among others.

Serial RapidIO, running at 3.125 GHz, and 10 Gigabit Ethernet are very competitive with respect to high bandwidth. However, while their capabilities overlap in some areas, serial RapidIO and Ethernet were developed to solve different problems and their underlying architectures differ in many respects.

Serial RapidIO was designed for embedded applications, supporting chip-to-chip and board-to-board communications. Because most of its protocol is implemented in the hardware of its endpoints, serial RapidIO offers extremely low latency and deterministic performance, and it does not require software management to move the data. The latency of serial RapidIO switches is highly deterministic: 112 ns for unicast packets and 163 ns for multicast packets. For endpoints, the latency depends on endpoint design, but is likely to be under 40 ns. With serial RapidIO, an increase in latency occurs only when there is an enormous amount of traffic or when an endpoint in the network is too slow to manage its incoming traffic.

Ethernet was originally designed as a way for multiple computers to communicate over a shared coaxial cable. The physical layer has evolved to point-to-point, but each endpoint is assumed to have a processor that is both available and capable of running software that implements the Ethernet (TCP/IP) protocol stack. Because its protocol is implemented in software, Ethernet implies higher latency, non-deterministic performance, and the need for software management. Ethernet is a "best effort" transmission, unless quality of service (QoS) is built in. There is no guarantee that data will arrive at any particular time, and packets can be dropped (and lost). With serial RapidIO, latency is in the hundreds of nanoseconds range. With Ethernet, it could be one microsecond or much more, depending on the amount of traffic on the network. Although the Ethernet stack can be implemented in hardware, this approach locks in a particular version of the stack, losing the flexibility that is one of its main advantages.

On the other hand, Ethernet has become the unchallenged communications interconnect for wide area networks, because its stack is highly flexible and supports essentially unlimited numbers of endpoints. Ethernet is also off-the-shelf technology. Nearly everyone knows how to deploy it, whereas serial RapidIO is less well known.

How Mercury Builds Solutions with ATCA
Mercury uses the ATCA chassis, carrier blade, and AMC form factors to build specialized computing engines that target specific compute-intensive processing requirements. These computing elements are linked via serial RapidIO, while Gigabit Ethernet is used for control communications and I/O to the outside world (Figure 5).

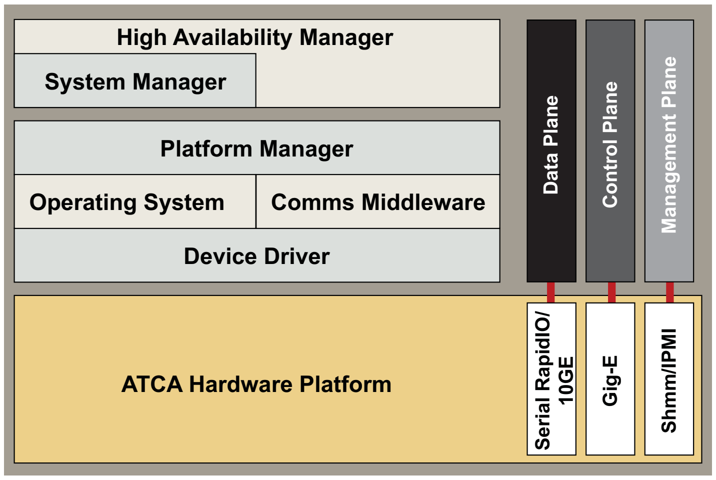

Figure 5. ATCA cross-platform software stack

Specialized Computing Engines with Carrier Blades and AMCs. Mercury uses two approaches to creating optimal computing engines for a particular application:

- Adding processors directly to a carrier blade to create a specialized carrier blade. Computing elements on a carrier blade can act as a hard-wired pool of processors, or they can be interconnected among different carrier blades in a flexible manner using switches on the carrier and in the hub. This design delivers maximum performance, but is less flexible in terms of re-use.

- Adding processors to an AMC, which then attaches to a standard carrier blade. Building a set of AMCs for different computational jobs is simpler and faster than designing specialized carrier blades, and AMCs designed for different applications can be removed and replaced, depending on application requirements. For example, if an application needs more codec processing, an AMC that does the code processing can be replaced with a higher-performance codec AMC.

Mercury designs its own compute engines only when no other reliable solution is available that properly addresses the application requirement. In most cases, Mercury builds ATCA systems with off-the-shelf AMCs and other components.

Serial RapidIO for the Data Plane. Mercury uses serial RapidIO to move data between computing elements on a carrier blade (intra-blade) and between computing elements on different blades in a chassis (inter-blade). Native format is also used, if raw data can be handled. As an ATCA data plane, serial RapidIO has two main advantages:

- It does not need a lot of data pre-packaging and requires minimal software to move it. With 1 or 10 Gigabit Ethernet, however, you need a TCP/IP stack to be able to move the data. Using serial RapidIO eliminates the need for stack processing, so the data moves quickly with low latency.

- It allows for many more hops without a latency penalty. As a result, serial RapidIO enables system scalability without compromising latency. Because each processing element has deterministic latency (as opposed to best effort in Ethernet), it is still possible to define to nanosecond accuracy how much time it will take to get from one particular processing element to another.

Gigabit Ethernet for the Control Plane and Inter-System Communication. Mercury uses 1 Gigabit Ethernet as the control plane for intra-chassis control, as supported by the ATCA standard. For inter-chassis communication and data I/O, Mercury also uses 1 Gigabit Ethernet, because communication between different physical locations is typically over public Internet or a proprietary network that is most likely Ethernet-based.

SOME REAL-WORLD EXAMPLES

To help get a clearer idea of how these design parameters work together, this paper concludes with two examples of ATCA systems that Mercury has developed.

Example 1: Custom AMCs, Standard AMCs, and Serial Rapid IO Fabric

This example illustrates using AMCs, standard ATCA carrier blades, and a RapidIO fabric to meet the demanding low-latency control-loop requirements of an industrial equipment application.

The customer's application had hard, deterministic, real-time constraints. Breaking the input-to-output latency would result in machine failure, unacceptable performance, and down-time. Achieving a low mean latency was insufficient. The peak system-level latency also needed to be constrained. As the demands on the equipment increased, an existing bus-based computing system was running out of bandwidth and introducing unacceptable latency. Our customer needed a fabric-based solution that was cost-effective and scalable, yet still supported an existing proprietary I/O format.

The computing system was linked to other parts of the industrial equipment by a set of proprietary interfaces. There were so many such connections that replacing them with a standards-based interface was considered unrealistic. Mercury responded by developing a custom interface AMC to do the translation between RapidIO and the proprietary protocol.

The rest of Mercury's system design used off-the-shelf ATCA components: processing AMCs using the PowerQUICC™ III 8548, a fiber I/O AMC, standard carrier blades with RapidIO switches, and a 5-slot ATCA chassis.

The resulting computing system (Figure 6) met the customer's deterministic low-latency requirements in a physically small, high-density package. An additional benefit was that ATCA brought with it a set of system management capabilities through the IPMI network and shelf manager. These capabilities allowed machine health functionality to be added to the computing system, improving diagnostics and maintenance, which resulted in reduced down time and service costs.

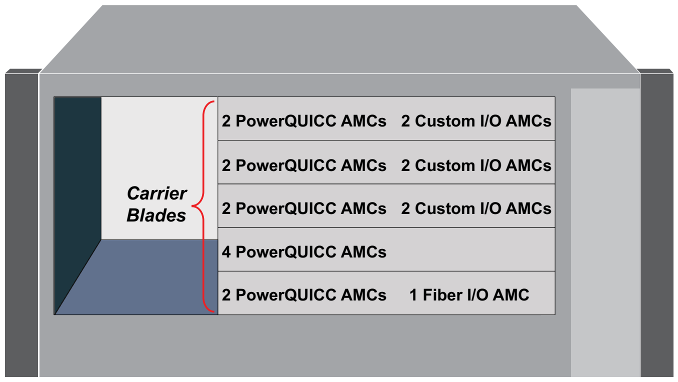
Carrier Blades
2 PowerQUICC AMCs 2 Custom I/O AMCs
2 PowerQUICC AMCs 2 Custom I/O AMCs
2 PowerQUICC AMCs 2 Custom I/O AMCs
4 PowerQUICC AMCs
2 PowerQUICC AMCs 1 Fiber I/O AMC

Figure 6. ATCA chassis with AMCs

Example 2: Specialized Blade without Serial RapidIO Fabric

This example illustrates a semi-customized processing-power solution for optimal application performance in a satellite communications system.

The solution addressed a high-bandwidth, high-compute-density, low-latency application that required 300 Gbps of I/O capacity in each direction and 15 TeraOPS of continuous computing per chassis. This example system performed analog-to-digital conversion in full-duplex mode. Because the number of processing components required to process the incoming data exceeded the physical capacity of a single chassis, processing was divided between two chassis that comprised the system (Figure 7).

The application's computational requirements—performing simultaneous simple mathematical calculations on thousands of data points—drove the design to multiple FPGAs on each carrier blade. Each FPGA was programmed to perform a different calculation, and the data was moved among FPGAs for processing. Figure 8 illustrates the possible instances of intra-blade, inter-blade, and inter-chassis data traffic for a carrier blade.

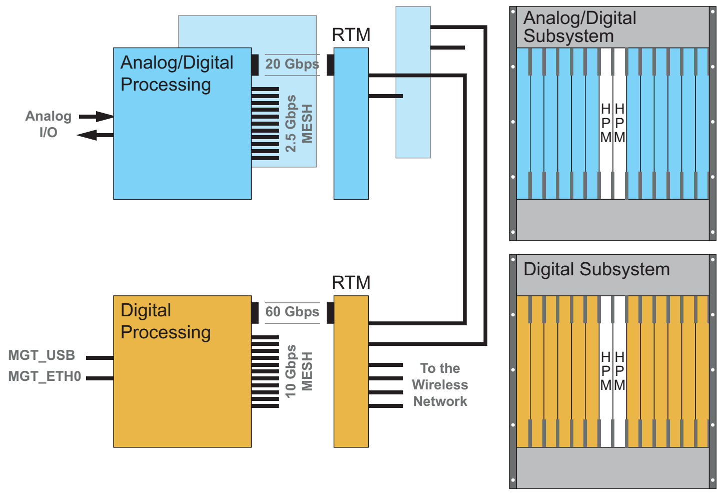

Figure 7. Data flow in 2-chassis high-density ATCA system

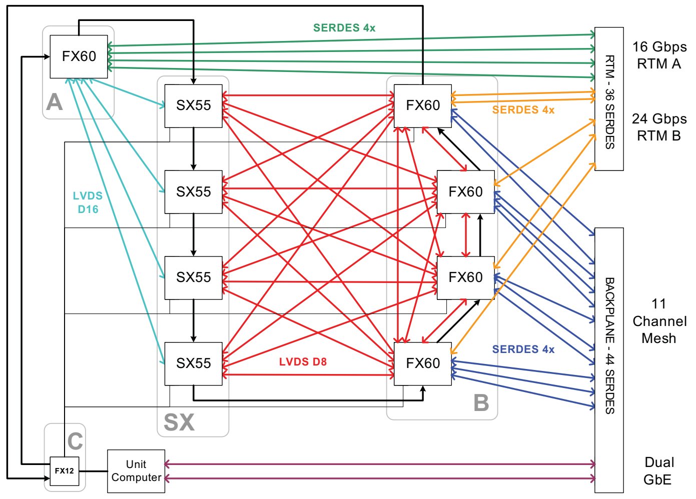

Figure 8. Raw data movement on ATCA carrier blade

Because this system used FPGA-to-FPGA communication, it was possible to transmit raw data. No package formatting was required. Latency was extremely low, because data was sent from blade to blade in the exact time needed for transmission of the electrical signal, as no protocols or data formats were involved. For this application, requiring extreme bandwidth but no data packaging, a raw data format was even more suitable than the efficient RapidIO protocol.

The system followed the ATCA standard of a control plane and a data plane. It used a mesh topology for the data plane and a star topology for the control plane. The data plane provided extremely high throughput: 20 Gbps of data between the two chassis.

For this application:

- I/O bandwidth was up to 720 Gbps (blade-dependent)

- Board-to-board bandwidth was 110 Gbps (blade-dependent)

- Processing power was up to 19.2 TOPS (blade-dependent)

Mercury Computer Systems, Inc.
Challenges Drive Innovation™

Corporate Headquarters
199 Riverneck Road
Chelmsford, MA 01824-2820 USA
+1 (978) 967-1401 • +1 (866) 627-6951
Fax +1 (978) 256-3599
www.mc.com

Worldwide Locations
Mercury Computer Systems has R&D, support and sales locations in France, Germany, Japan, the United Kingdom and the United States.

For office locations and contact information, please call the corporate headquarters or visit our Web site at **www.mc.com**.