2010 18th IEEE Symposium on High Performance Interconnects

The Gemini System Interconnect

Robert Alverson, Duncan Roweth, Larry Kaplan Cray Inc

***Abstract*—The Gemini System Interconnect is a new network for Cray's supercomputer systems. It provides improved network functionality, latency and issue rate. Latency is reduced with OS bypass for sends and direct user completion notification on receives. Atomic memory operations support the construction of fast synchronization and reduction primitives.**

I. Introduction

GEMINI is the new network for Cray's supercomputer systems. It enhances the highly scalable Seastar design used to deliver the 225,000 core Oak Ridge National Laboratory Jaguar system, improving network functionality, latency, and issue rate. Gemini uses a novel system-on-chip (SoC) design to construct direct 3D torus networks (Figure 1) that can scale to in excess of 100,000 multi-core nodes. Gemini is designed to deliver high performance on MPI applications and filesystem traffic; in addition it provides hardware support for global address space programming. Gemini enables efficient implementation of programming languages such as Chapel, UPC, and Co-Array Fortran on massively parallel systems.

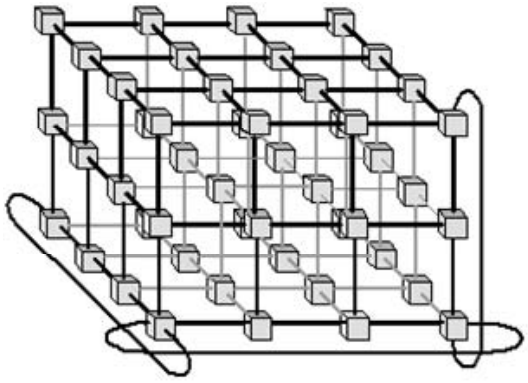

Figure 1: 3D Torus network

Each Gemini ASIC provides two network interface controllers (NICs), and a 48-port router. Each of the NICs has its own HyperTransport™ 3 host interface, enabling Gemini to connect two Opteron nodes to the network. This 2-node building block provides 10 torus connections, 4 each in two of the dimensions ('x' and 'z') and 2 in the third dimension ('y'), as shown in Figure 2. Traffic between the two nodes connected to a single Gemini is routed internally. The router uses a tiled design, with 8 tiles dedicated to the NICs and 40 (10 groups of 4) dedicated to the network.

The Gemini ASIC is implemented in the TSMC 90nm process and has a die size of 232.8 mm^2. The block structure of the Gemini design is illustrated in Figure 3. The Netlink block connects the NICs to the router. Traffic from both NICs is distributed over all eight of the router's inputs enabling injection bandwidth to be load balanced on a packet-by-packet basis. The Netlink also handles changes in clock speed between the NIC and router domains.

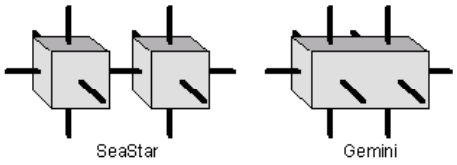

Figure 2: Seastar and Gemini

The supervisor block connects Gemini to an embedded control processor (L0) for the blade and hence the Cray Hardware Supervisory System (HSS) network, used for monitoring the device and loading its routing tables.

Gemini is designed for large systems in which failures are to be expected and applications must continue to run in the presence of errors. Each torus connection comprises 4 groups of 3 lanes. Packet CRCs are checked by each device with automatic link-level retry on error. The failure of a single lane is tolerated by shunting its data to the remaining 2 lanes in the group. In the event of the complete failure of a link, the router will select an alternate path for adaptively routed traffic. Gemini uses ECC to protect major memories and data paths within the device.

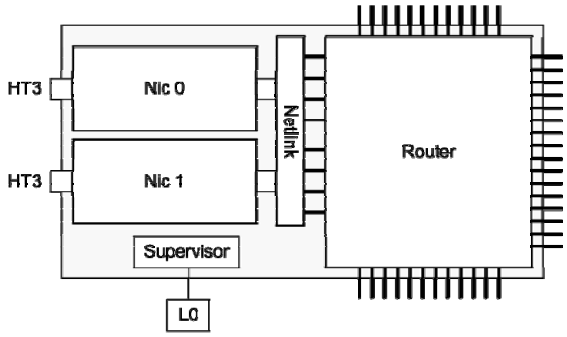

Figure 3: Gemini block structure

For traffic designated as adaptive, the Gemini router performs packet by packet adaptive routing, distributing traffic over lightly loaded links. With 8 links connecting each Gemini to its neighbors in the 'x' and 'z' directions and 4 links in the 'y' dimension, there are multiple paths available. Hashed deterministic routing can be selected as an alternative when a sequence of operations to the same cache line must be performed in order.

Gemini provides the ability for user processes to transfer data directly between nodes without OS intervention. For example, one process in a parallel job can initiate a put directly from its memory to that of another process. To do this it specifies the data (or source address), the destination virtual

978-0-7695-4208-9/10 $26.00 © 2010 IEEE
DOI 10.1109/HOTI.2010.23

addresses, the destination process id, and the size of the transfer. Additional hardware primitives include remote get, atomic operations, block transfer and completion notification. The Gemini NIC is a hardware pipeline that maximizes the performance of these simple operations. More complex communication protocols such as message passing and TCP/IP are implemented using these primitives.

User space communication is supported by the User Gemini Network Interface (uGNI) and Distributed Memory Application (DMAPP) APIs. These libraries are called by Cray MPI and Shmem. DMAPP is also used in the run time for Cray Chapel, UPC, and Co-Array Fortran compilers. Inter-kernel communication is provided using the Kernel Gemini Network Interface (kGNI) which provides both messaging and RDMA. The Lustre filesystem is supported via a Lustre Network Driver (LND) for kGNI. Other filesystems such as NFS, GPFS and Panasas are provided via DVS, the Cray Data Virtualization Service layered over LND. TCP/IP communication over the Gemini fabric is provided by the IP over Gemini Fabric (IPoGIF) module.

II. GEMINI NIC

A. Overview

Each Gemini ASIC has a pair of NICs, each with its own HyperTransport 3 interface (known as the HT Cave and shown on the left hand side of Figure 3). The NICs are connected to the Gemini router via the Netlink block (on the right hand side of Figure 4). The NIC is a hardware pipeline. The node issues commands, writing them across the HyperTransport interface. The NIC packetizes these requests and issues the packets to the network, with output flowing from left to right at the top of Figure 4.

Packets are routed across the network to a destination NIC. The input pipeline flows from right to left at the bottom of Figure 4. The Gemini network employs a 3-tuple, the Network Address, to specify a logical address in a user process on a remote node. The address consists of a processing element identifier (or PE), a Memory Domain Handle (MDH) associated with a memory segment registered at the remote node, and an offset into this segment. This 58-bit network address extends the physical address space of the node, enabling global access to all of the memory of a large system.

Gemini supports both virtual addressing and virtual PEs. The MDH is combined with the offset to generate a user virtual address in the remote process. Virtual PEs (ranks in MPI parlance) used by the application are translated on output by the Node Translation Table (NTT) to obtain the physical PE. Constraints on physical resources limit the size of the NTT; only the top 12 bits of the PE are translated. Very large jobs are laid out in a regular fashion with low bits of the virtual and physical PEs being equal; alternately, they can use physical PEs

B. Fast Memory Access (FMA)

Fast Memory Access is a mechanism whereby user processes generate network transactions, such as puts, gets and atomic memory operations (AMO), by storing directly to the NIC. The FMA block translates stores by the processor into fully qualified network requests. FMA provides both low latency and high issue rate on small transfers. On initialization the user process is allocated one or more FMA descriptors and associated FMA windows. Writes to the FMA descriptor determine the remote processing element and the remote address associated with the base of the window. A write of up to 64 bytes to the put window generates a remote put. Storing an 8 byte control word to the get window generates a get of up to 64 bytes or a fetching AMO. FMA supports scattered accesses by allowing the user to select which bits in an FMA window determine the remote address and which determine the remote PE. Having set the FMA descriptor appropriately one can, for example, store a unique cacheline of data to each process in a parallel job by simply storing a contiguous block of data to the FMA window. The DMAPP library provides a lightweight wrapper around this functionality for the Cray compilers and libraries.

FMA supports source-side synchronization methods for tracking when put requests have reached a globally ordered

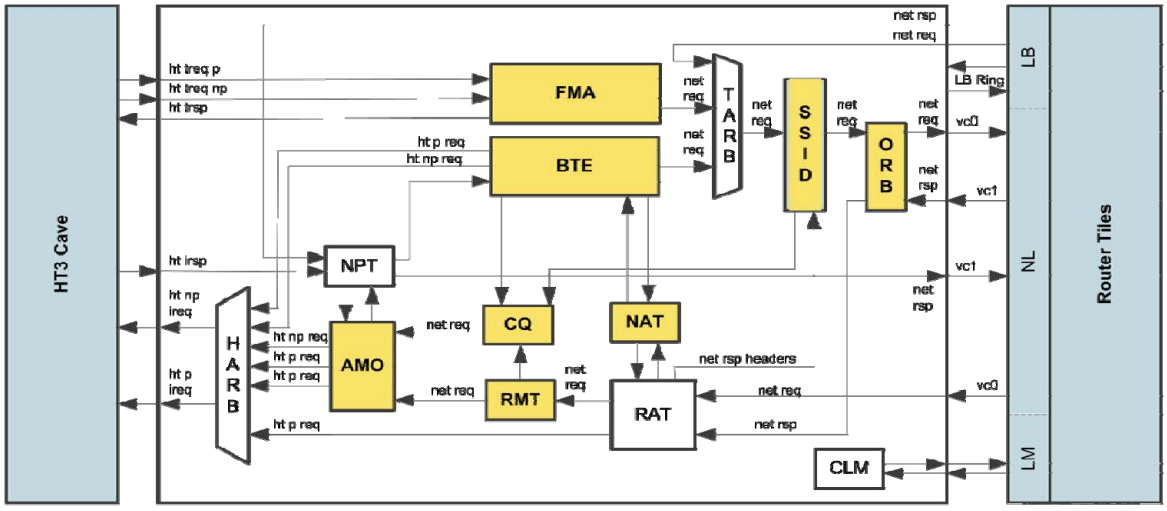

Figure 4: Gemini NIC

point at the target and when responses to get requests have reached a globally ordered point in the local node. It is also possible to issue puts that generate destination-side synchronization events at the target node, enabling a process on that node to be notified of new data, or to poll a single completion queue for its arrival.

C. Block Transfer Engine (BTE)

The Block Transfer Engine (BTE) supports asynchronous transfers between local and remote memory. Kernel software writes block transfer descriptors to a queue and the Gemini hardware performs the transfers asynchronously. The BTE supports memory operations (put/get) where the user specifies a local address, a network address and a transfer size. In addition the BTE supports channel operations (send) where the user specifies a local address and a target, but no target address. Channel semantics require the user to have pre-posted a receive buffer with the target BTE. By default there is no guarantee of completion ordering in block transfers issued by a given Gemini. Fence operations are used where necessary to ensure that one transfer is completed before another starts.

In general FMA is used for small transfers and BTE for large. FMA transfers are lower latency. BTE transfers take longer to start, but once running can transfer large amounts of data (up to 4GB) without CPU involvement.

D. Completion Queue (CQ)

Completion queues provide a lightweight event notification mechanism. The completion of a BTE or FMA transaction can generate an event in a user (or kernel thread) specific queue. Completion events can be generated on either the source or the target node. They include both user data and transaction status information.

E. Atomic Memory Operation (AMO)

Gemini supports a wide range of atomic operations, those with put semantics such as atomic add and those with get semantics such as conditional swap. Gemini maintains an AMO cache, reducing the need for reads of host memory when multiple processes access the same atomic variable. Host memory is updated each time the variable is updated (lazy update mechanisms are also provided to reduce load on the host interface), but network atomics are not coherent with respect to local AMD64 memory operations - all processes must use a Gemini application interface to update an atomic variable.

F. Synchronization Sequence Identification

Gemini uses a mechanism known as Synchronization Sequence Identification to track the set of packets that make up a transaction. Every packet in the sequence contains the same Synchronization Sequence Identifier (SSID). Packets can be delivered in arbitrary order; each contains a network address and can be committed to memory as soon as it arrives. There is no need for a reorder buffer. The sequence as a whole completes and CQ events are generated when all packets have been delivered. This mechanism is implemented using the SSID and Output Request Buffer (ORB) blocks on the output side and the Receive Message Table (RMT) block on the input side. The RMT caches active SSID state avoiding a network round trip for performance critical operations. It also matches BTE send requests to queued receive descriptors.

III. Gemini Router

The building block for the Gemini router is the tile (see Figure 5). Each tile contains all of the logic and buffering associated with one input port, one output port, an 8x8 switch, and associated buffers. In Gemini, each tile's switch accepts inputs from six row buses that are driven by the input ports in its row, and drives separate output channels to the eight output ports in its column. Using a tile-based micro-architecture facilitates implementation, since each tile is identical and produces a very regular structure for replication and physical implementation in silicon.

The tile-based design is best understood by following a packet through the router. A packet arrives in the input link of a tile. When the packet reaches the head of the input buffer, a routing decision is made to select the output column for the packet. The packet is then driven onto the row bus associated with the input port and buffered in a row buffer at the input of the 8x8 switch at the junction of the packet's input row and output column (at the cross-point tile). At this point the routing decision must be refined to select a particular output port within the output column. The switch then routes the packet to the column channel associated with the selected output port. The column channel delivers the packet to an output buffer (associated with the input row) at the output port multiplexer. Packets in the per-input-row output buffers arbitrate for access to the output port and, when granted access, are switched onto the output port via the multiplexer.

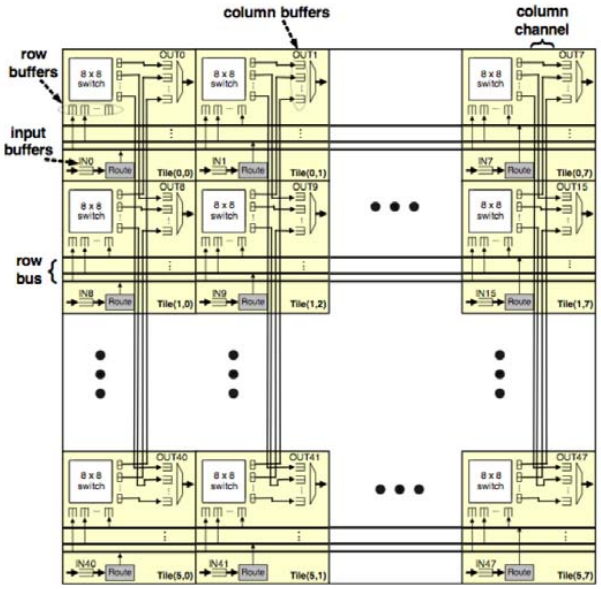

Figure 5: Gemini router

Gemini uses virtual cut-through flow control across the network links, but uses wormhole flow control internally due

to buffer size constraints. Network link input buffers are deep enough to account for credit round-trip latency and the maximum packet size.

Packets have a variable size and are divided into 24-bit phits (physical units) for transmission over network links. Write request packets have a 7-phit header, up to 24 phits of data and a single phit end-of-packet that denotes the last phit of a packet and contains status bits for error handling. A 2-phit response packet is generated for each request (3 phits on error). Get responses include a payload of up to 24 phits.

The header phit controls routing; it specifies the destination, the virtual channel, and details of how the packet is to be routed (see Figure 6).

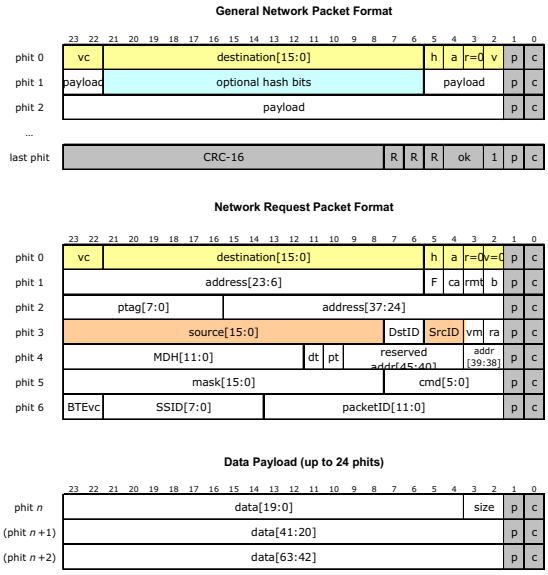

Figure 6: Network Packet Formats

Each Gemini chip has a unique 16-bit identifier, specified by the destination field within each packet. NICs and hence Opteron nodes are specified using a 2-bit identifier for the source (SrcID) and destination (DstID). The combined 18-bit address uniquely identifies every node in the system. The *v* field specifies the virtual channel; Gemini uses one virtual channel for requests and another for responses. The *r*, *a*, and *h* fields control routing. If the *r* bit is set, then the packet will be source routed and must contain a routing vector in the payload. Source routing is only used for diagnostics. If the *a* (adapt) bit is set, the packet is routed adaptively, otherwise the packet is routed using a deterministic hash constructed from the source and destination ids and (optionally if the *h* bit is set) the remote address. The fields shaded gray in Figure 6 contain side band data used by the link control block.

An 8-byte write requires an 11-phit request (7 header, 3 data and 1 EOP) and a 2-phit response. A 64-byte cache-line write requires 32 request phits (7 header, 24 data, and 1 EOP) and 2 response phits. A 64-byte get comprises an 8-phit request (7 header plus EOP) and a 27-phit response (2 header, 24 data and EOP).

IV. Gemini Fault Tolerance

Gemini provides a 16-bit packet CRC, which protects up to 64-bytes of data and the associated headers (768 bits max). Within each Gemini, major memories are protected using ECC.

Gemini links provide reliable delivery using a sliding window protocol. The receiving link checks the CRC as a packet arrives, returning an error if it is incorrect. The sending link retransmits on receipt of an error. The link block includes a send buffer of sufficient size to cover the round trip.

The CRC is also checked as a packet leaves each Gemini and as it transitions from the router to the NIC, enabling detection of errors occurring within the router core. If the checksum is incorrect, the packet is marked as bad and passed on; it will be dropped by the destination Gemini.

Completion events include details of the status of each transaction, allowing software to recover from errors. HSS/OS interfaces allow the reporting of any of these errors at the point of occurrence.

Each 3D Torus connection is made up of 4 Gemini links. The Gemini adaptive routing hardware will spread packets over the available links. If a link fails, the adaptive routing hardware will mask it out. In the event of losing all connectivity between two Gemini chips, it is necessary to route around the problem (this also happens when a board is removed). To handle this, the management software quiesces the network, computes new routing tables and then re-enables the network.

V. Gemini Performance

A. Clock Speed

The Gemini NIC operates at 650 MHz, the router at 800MHz, and the link SERDES at 3.125 to 6.25 GHz. The speed of the HyperTransport interface ranges from 1600 MHz to 2600 MHz depending on the node type.

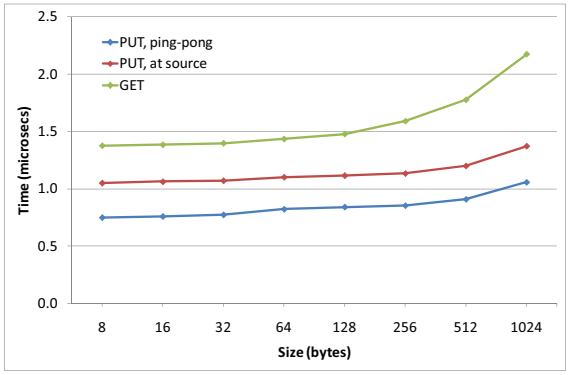

Figure 7; Gemini put and get latencies as a function of transfer size

A. *Latency & Bandwidth*

End-to-end latency in a Gemini network is determined by the end point latencies and the number of hops. On a quiet network, the end-point latency is less than 700 nanosecond for a remote put (see Figure 7), and 1.5 microseconds or less for a small MPI message. The low overheads of HyperTransport reads enable Gemini to achieve get latencies of less than 1.5 microseconds. The latency per hop is typically 105 ns on a quiet network.

The Gemini NIC can transfer 64 bytes of data in each direction every 5 cycles. Thus the maximum bandwidth per direction is 64 x 650 / 5 = 8.3 GBytes/s. Injection bandwidth depends on the speed of the HyperTransport interface and the method of transfer. The interface is 16 bits wide and transfers data on both edges of the clock, giving a raw bandwidth of 9.6 GB/sec in each direction at 2400 MHz. With FMA Put, the HyperTransport overhead is 12 bytes for up to 64 bytes of data, limiting the peak bandwidth to 8 GBytes/sec. Figure 8 shows FMA put bandwidth for a sweep of transfer sizes. The bandwidth saturates progressively more easily as more processes per node participate. For BTE transfers there is a 12-byte read request followed by a 76-byte posted write for every 64-byte data packet. For symmetric BTE traffic the peak bandwidth of the host interface is 7 GBytes/sec in each direction after accounting for protocol overhead. The Netlink block injects packets into the router, distributing traffic across the 8 processor tiles. Each 64-byte write is transferred as 32 x 24-bit request phits (7 header, 24 data and 1 end of packet) with a 2-phit response, with each processor tile transferring one 64-byte packet in each direction every 32 cycles.

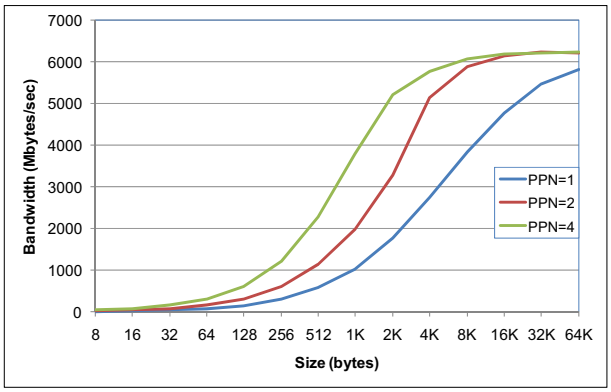

Figure 8: Gemini FMA put bandwidth as a function of transfer size for 1, 2 and 4 processes per node

Each of the 10 Gemini torus connections is comprised of 12 lanes in each direction operating at 3.125 to 6.25 GHz. Link bandwidths are 4.68 to 9.375 GBytes/sec per direction. Efficiency on 64-byte transfers is 63%, significantly higher than competing products. Bandwidth after protocol is 2.9 to 5.8 GB/sec per direction. The torus network provides multiple links between nodes. Packets are adaptively distributed over all of the available links, enabling a single transfer (a point-to-point MPI message for example) to achieve bandwidths of 5 GBytes/sec or more. Higher bandwidths can be achieved between processes on the same node or processes on nodes connected to the same Gemini. Where multiple user processes or kernel threads are sending data at the same time (the common case for multi-core nodes), the Gemini NIC can inject packets at host interface bandwidth, with the router spreading traffic out over all of the available links. Note that the bandwidth at which a node can send data to multiple destinations or receive data from multiple destinations exceeds the point-to-point bandwidth between any pair of nodes. The former is limited by injection bandwidth and the latter is limited by link bandwidth.

Figure 9 shows the performance achieved on many-to-one atomic memory operations (atomic add in this case). With a single AMO, all operations hit in the Gemini AMO cache; with 8192 variables selected at random by the source processes, all operations should miss. Gemini can deliver AMO rates in excess of 45 million/sec. When there is good cache reuse, as potentially with a barrier implementation, the AMO rate can reach 100 million updates per second.

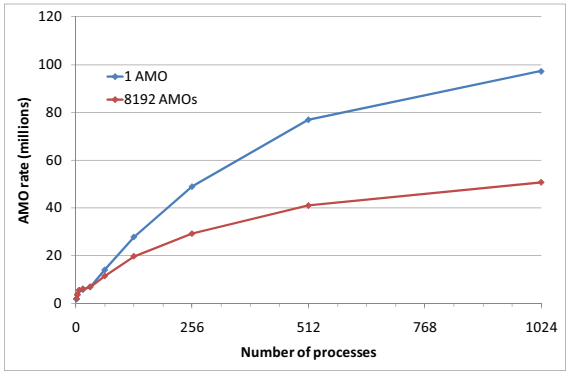

Figure 9; Gemini AMO performance

VI. Conclusion

The Gemini system interconnect provides all the essential components to build a highly efficient MPI communication system for supercomputers. In addition it supports emerging global address space programming with fine-grain single-sided remote put, get, and AMO operations. Coupled with its fault-tolerance features, these allow Gemini systems to scale to over 100,000 cores.

VII. Acknowledgement

This material is based upon work supported by the Defense Advanced Research Projects Agency under its Agreement No. HR0011-07-9-0001