An Overview of the BlueGene/L Supercomputer

Conference Paper · December 2002

DOI: 10.1109/SC.2002.10017 · Source: IEEE Xplore

CITATIONS
180

READS
159

115 authors, including:

Yariv Aridor
Intel
33 PUBLICATIONS **1,158** CITATIONS
SEE PROFILE

Rajkishore Barik
Intel
41 PUBLICATIONS **960** CITATIONS
SEE PROFILE

Ralph Bellofatto
IBM
31 PUBLICATIONS **672** CITATIONS
SEE PROFILE

Gyan V Bhanot
Rutgers, The State University of New Jersey
308 PUBLICATIONS **7,766** CITATIONS
SEE PROFILE

An Overview of the BlueGene/L Supercomputer

The BlueGene/L Team
IBM and Lawrence Livermore National Laboratory[1]

NR Adiga, G Almasi, GS Almasi, Y Aridor, M Bae, R Barik, D Beece, R Bellofatto, G Bhanot, R Bickford, M Blumrich, AA Bright, J Brunheroto, C Caşcaval, J Castaños, W Chan, L Ceze, P Coteus, S Chatterjee, D Chen, G Chiu, TM Cipolla, P Crumley, KM Desai, A Deutsch, T Domany, MB Dombrowa, W Donath, M Eleftheriou, C Erway, J Esch, B Fitch, J Gagliano, A Gara, R Garg, R Germain, ME Giampapa, B Gopalsamy, J Gunnels, M Gupta, F Gustavson, S Hall, RA Haring, D Heidel, P Heidelberger, LM Herger, D Hoenicke, RD Jackson, T Jamal-Eddine, GV Kopcsay, E Krevat, MP Kurhekar, AP Lanzetta, D Lieber, LK Liu, M Lu, M Mendell, A Misra, Y Moatti, L Mok, JE Moreira, BJ Nathanson, M Newton, M Ohmacht, A Oliner, V Pandit, RB Pudota, R Rand, R Regan, B Rubin, A Ruehli, S Rus, RK Sahoo, A Sanomiya, E Schenfeld, M Sharma, E Shmueli, S Singh, P Song, V Srinivasan, BD Steinmacher-Burow, K Strauss, C Surovic, R Swetz, T Takken, RB Tremaine, M Tsao, AR Umamaheshwaran, P Verma, P Vranas, TJC Ward, M Wazlowski
IBM Research

W Barrett, J Brown, D Krolak, T Liebsch, J Marcella, A Schram, G Ulsh, C Wait
IBM Rochester

K Dockser
IBM Microelectronics

L Kissel, MK Seager, JS Vetter, K Yates
Lawrence Livermore National Laboratory

Abstract

This paper gives an overview of the BlueGene/L Supercomputer. This is a jointly funded research partnership between IBM and the Lawrence Livermore National Laboratory as part of the United States Department of Energy ASCI Advanced Architecture Research Program. Application performance and scaling studies have recently been initiated with partners at a number of academic and government institutions, including the San Diego Supercomputer Center and the California Institute of Technology. This massively parallel system of 65,536 nodes is based on a new architecture that exploits system-on-a-chip technology to deliver target peak processing power of 360 teraFLOPS (trillion floating-point operations per second). The machine is scheduled to be operational in the 2004-2005 time frame, at price/performance and power consumption/performance targets unobtainable with conventional architectures.

[1] Part of this work was performed under the auspices of the U.S. Department of Energy by the University of California at Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

1. Introduction and Background

IBM has previously announced a multi-year initiative to build a petaflop scale machine for calculations in the area of life sciences. The BlueGene/L machine is a first step in this program, and is based on a different and more generalized architecture than IBM described in its announcement of the BlueGene program in December of 1999. In particular BlueGene/L is based on an embedded PowerPC processor supporting a large memory space, with standard compilers and message passing environment, albeit with significant additions and modifications to the standard PowerPC system.

Significant progress has been made in recent years mapping numerous compute-intensive applications, many of them grand challenges, to parallel architectures. This has been done to great success largely out of necessity, as it has become clear that currently the only way to achieve teraFLOPS-scale computing is to garner the multiplicative benefits offered by a massively parallel machine. To scale to the next level of parallelism, in which tens of thousands of processors are utilized, the traditional approach of clustering large, fast SMPs will be increasingly limited by power consumption and footprint constraints. For example, to house supercomputers in the 2004 time frame, both the Los Alamos National Laboratory and the Lawrence Livermore National Laboratory have begun constructing buildings with approximately 10x more power and cooling capacity and 2-4x more floor space than existing facilities. In addition, due to the growing gap between the processor cycle times and memory access times, the fastest available processors will typically deliver a continuously decreasing fraction of their peak performance, despite ever more sophisticated memory hierarchies.

The approach taken in BlueGene/L (BG/L) is substantially different. The system is built out of a very large number of nodes, each of which has a relatively modest clock rate. Those nodes present both low power consumption and low cost. The design point of BG/L utilizes IBM PowerPC embedded CMOS processors, embedded DRAM, and system-on-a-chip techniques that allow for integration of all system functions including compute processor, communications processor, 3 cache levels, and multiple high speed interconnection networks with sophisticated routing onto a single ASIC. Because of a relatively modest processor cycle time, the memory is close, in terms of cycles, to the processor. This is also advantageous for power consumption, and enables construction of denser packages in which 1024 compute nodes can be placed within a single rack. Integration of the inter-node communications network functions onto the same ASIC as the processors reduces cost, since the need for a separate, high-speed switch is eliminated. The current design goals of BG/L aim for a scalable supercomputer having up to 65,536 compute nodes and target peak performance of 360 teraFLOPS with extremely cost effective characteristics and low power (~1 MW), cooling (~300 tons) and floor space (<2,500 sq ft) requirements. This peak performance metric is only applicable for applications that can utilize both processors on a node for compute tasks. We anticipate that there will be a large class of problems that will fully utilize one of the two processors in a node with messaging protocol tasks and will therefore not be able to

utilize the second processor for computations. For such applications, the target peak performance is 180 teraFLOPS.

The BG/L design philosophy has been influenced by other successful massively parallel machines, including QCDSP at Columbia University. In that machine, thousands of processors are connected to form a multidimensional torus with nearest neighbor connections and simple global functions. Columbia University continues to evolve this architecture with their next generation QCDOC machine [QCDOC], which is being developed in cooperation with IBM research. QCDOC will also use a PowerPC processing core in an earlier technology, a simpler floating point unit, and a simpler nearest neighbor network.

2. System Overview

BlueGene/L is a scalable system in which the maximum number of compute nodes assigned to a single parallel job is $2^{16} = 65,536$. BlueGene/L is configured as a 64 x 32 x 32 three-dimensional torus of compute nodes. Each node consists of a single ASIC and memory. Each node can support up to 2 GB of local memory; our current plan calls for 9 SDRAM-DDR memory chips with 256 MB of memory per node. The ASIC that powers the nodes is based on IBM's system-on-a-chip technology and incorporates all of the functionality needed by BG/L. The nodes themselves are physically small, with an expected 11.1-mm square die size, allowing for a very high density of processing. The ASIC uses IBM CMOS CU-11 0.13 micron technology and is designed to operate at a target speed of 700 MHz, although the actual clock rate used in BG/L will not be known until chips are available in quantity.

The current design for BG/L system packaging is shown in Figure 1. (Note that this is different from a preliminary design shown in [ISSCC02] as are certain bandwidth figures that have been updated to reflect a change in the underlying signaling technology.) The design calls for 2 nodes per compute card, 16 compute cards per node board, 16 node boards per 512-node midplane of approximate size 17"x 24"x 34," and two midplanes in a 1024-node rack. Each processor can perform 4 floating point operations per cycle (in the form of two 64-bit floating point multiply-add's per cycle); at the target frequency this amounts to approximately 1.4 teraFLOPS peak performance for a single midplane of BG/L nodes, if we count only a single processor per node. Each node contains a second processor, identical to the first although not included in the 1.4 teraFLOPS performance number, intended primarily for handling message passing operations. In addition, the system provides for a flexible number of additional dual-processor I/O nodes, up to a maximum of one I/O node for every eight compute nodes. For the machine with 65,536 compute nodes, we expect to have a ratio one I/O node for every 64 compute nodes. I/O nodes use the same ASIC as the compute nodes, have expanded external memory and gigabit Ethernet connections. Each compute node executes a lightweight kernel. The compute node kernel handles basic communication tasks and all the functions necessary for high performance scientific code. For compiling, diagnostics, and analysis, a host computer is required. An I/O node handles communication between a compute node and

other systems, including the host and file servers. The choice of host will depend on the class of applications and their bandwidth and performance requirements.

The nodes are interconnected through five networks: a 3D torus network for point-to-point messaging between compute nodes, a global combining/broadcast tree for collective operations such as MPI_Allreduce over the entire application, a global barrier and interrupt network, a Gigabit Ethernet to JTAG network for machine control, and another Gigabit Ethernet network for connection to other systems, such as hosts and file systems. For cost and overall system efficiency, compute nodes are not hooked directly up to the Gigabit Ethernet, but rather use the global tree for communicating with their I/O nodes, while the I/O nodes use the Gigabit Ethernet to communicate to other systems.

In addition to the compute ASIC, there is a "link" ASIC. When crossing a midplane boundary, BG/L's torus, global combining tree and global interrupt signals pass through the BG/L link ASIC. This ASIC serves two functions. First, it redrives signals over the cables between BG/L midplanes, improving the high-speed signal shape and amplitude in the middle of a long, lossy trace-cable-trace connection between nodes on different midplanes. Second, the link ASIC can redirect signals between its different ports. This redirection function enables BG/L to be partitioned into multiple, logically separate systems in which there is no traffic interference between systems. This capability also enables additional midplanes to be cabled as spares to the system and used, as needed, upon failures. Each of the partitions formed through this manner has its own torus, tree and barrier networks which are isolated from all traffic from all other partitions on these networks.

System fault tolerance is a critical aspect the BlueGene/L machine. BlueGene/L will have many layers of fault tolerance that are expected to allow for good availability despite the large number of system nodes. In addition, the BlueGene/L platform will be used to investigate many avenues in autonomic computing.

3. System Packaging

The BG/L system is a cost/performance design, focused on fault tolerance, high density, low power and thus achieving low acquisition and runtime cost. The hardware cost is dominated by the ASIC and DRAM devices themselves. To manage circuit card costs, the interconnect was developed from the outside in. After identifying a system package based on standard 19" racks modified for high power transverse air cooling, we arranged these racks to minimize the longest cable as this is the bandwidth limiter in rack to rack communication. The differential cables are 26 AWG , and the longest length is 8 meters. Thicker conductors gave negligible improvement in attenuation for the added bulk. Rather, focus was on mechanical footprint, robustness and avoidance of large impedance discontinuities. We chose connector pin assignments for cables and circuit cards to minimize card wiring layers and avoid high priced, higher risk fine geometries, while also minimizing the worst case wire length. Circuit cards and ASIC packages have just 4 internal wiring layers except the 6 wiring layer midplane. Minimum card line width and

space is ~110μm long traces are oversized to ~200μm width while maintaining 50Ω impedance to reduce DC and skin effect losses. ASIC floorplanning and I/O assignments were iterated until a minimum layer ASIC package could be built. To avoid large numbers of capacitors for proper image current return at connectors, and to reduce switching noise, we used all differential signaling for the 1.4 Gb/s torus and tree links, and complete ground referencing on all nets. To reduce connector failure all DRAMs and DC-DC power supplies are directly soldered, cable connectors have screwed lockdowns, and all connectors are very reliable pin and socket multi-contact interfaces. Each 22 differential pair cable contains a spare pair which can be swapped in by the link ASIC, in much the same way that the 9 chip DRAM system has a spare 4 bits (nibble) that can be spared in if required. The DRAM additionally supports through ECC (error correcting code) the ability to run uninterrupted without error when losing a consecutive byte of the data bus as well as the usual single bit correct/ double bit detect functionality provided by ECC. Cables are removed only to service the link cards.

The BlueGene/L power and cooling network is an example of a cost/performance fault tolerant design. The system is air cooled, designed to operate in standard raised floor machine rooms, and assumes standard fault tolerant 220V rack feed and failover air chillers. Racks are designed to be on a 3-foot pitch in a row and a 6-foot pitch between rows. Chilled air to cool the expected 20kW/rack is drawn from an opening in the raised floor beneath the rack by a wall of fans on the left side of the rack. As shown in Figure 2, thirty 100mm diameter high speed, DC "smart fans" arranged on pluggable fans cards cool a midplane. Fan speed is monitored and adjusted with feedback from thermal sensors to maintain constant chip temperature in the face of coherent system-wide power demands. This reduces the effect of mini-cycles that can cause thermal cycle fatigue of chip to card connections. If a fan slows or stops, the others increase rotation to maintain ASIC junction temperature and an error condition is reported to the control host. A damaged fan card can be replaced with the system running. Fan power is supplied by the same 208V AC->48V DC N+1 redundant supplies that power the rack electronics. The 48V DC is further regulated to the 1.5V and 2.5V required respectively by the BG/L ASICs and external DRAMs by either commercially available long-lived converters with a mean time between failures (MTBF) of more than 2 Million hours, or better, redundant supplies. Both design points are being considered. The desired MTBF of the system is at least 10 days.

The system MTBF is calculated below assuming predicted failure rates for ASICs after burn-in, predicted DRAM hard failure rates, and manufacturer's suggested average failure rates for remaining components. Redundant power supplies would further increase MTBF. We expect to be dominated by DRAM hard failures. The expected DRAM failure rate is ~5x less than the raw DRAM rate of 25 FITs which accounts for the effects of bit sparing in the controller. We expect a maintenance policy that tracks software errors and replaces nodes at service intervals with DRAMs or compute ASICs with EDRAM showing increased soft errors, to further reduce this number.

System design methodology includes extensive use of parity and ECC to allow for the detection (and possible correction) for the vast majority of soft error events.

Component	FIT per component*	Components per 64k partition	FITs per system	Failure rate per week
ETH complex	160	3024	484k	
DRAM	5	608,256	3,041k	
Compute + I/O ASIC	20	66,560	1,331k	
Link ASIC	25	3072	77k	
Clock chip	6.5	~1200	8k	
Non-redundant power supply	500	384	384k	
Total (65,536 compute nodes)			5315k	0.89

* After burn-in and applied redundancy. T=60C, V=Nom, 40K POH. FIT = Failures in parts per million per thousand power-on hours. 1 FIT = $0.168*10^{-6}$fails/week if the machine runs 24 hrs/day.

Table 1: Uncorrectable hard failure Rates of BlueGene/L by major component.

4. Node Overview

The BG/L node ASIC, shown in Figure 3 includes two standard PowerPC 440 processing cores, each with a PowerPC 440 FP2 core, an enhanced "Double" 64-bit Floating-Point Unit. The 440 is a standard 32-bit microprocessor core from IBM's microelectronics division. This superscalar core is typically used as an embedded processor in many internal and external customer applications. Since the 440 CPU core does not implement the necessary hardware to provide SMP support, the two cores are not L1 cache coherent. A lockbox is provided to allow coherent processor-to-processor communication. Each core has a small 2 KB L2 cache which is controlled by a data pre-fetch engine, a fast SRAM array for communication between the two cores, an L3 cache directory and 4 MB of associated L3 cache made from embedded DRAM, an integrated external DDR memory controller, a gigabit Ethernet adapter, a JTAG interface as well as all the network link cut-through buffers and control. The L2 and L3 are coherent between the two cores.

In normal operating mode, one CPU/FPU pair is used for computation while the other is used for messaging. However, there are no hardware impediments to fully utilizing the second processing element for algorithms that have simple message passing requirements such as those with a large compute to communication ratio.

The PowerPC 440 FP2 core, shown in Figure 4, consists of a primary side and a secondary side, each of which is essentially a complete floating-point unit. Each side has its own 64-bit by 32 element register file, a double-precision computational datapath and a double-precision storage access datapath. A single common interface to the host PPC 440 processor is shared between the sides.

The primary side is capable of executing standard PowerPC floating-point instructions, and acts as an off-the-shelf PPC 440 FPU [K01]. An enhanced set of instructions include those that are executed solely on the secondary side, and those that are simultaneously executed on both sides. While this enhanced set includes SIMD operations, it goes well beyond the capabilities of traditional SIMD architectures. Here, a single instruction can initiate a different yet related operation on different data, in each of the two sides. These operations are performed in lockstep with each other. We have termed these type of instructions SIMOMD for Single Instruction Multiple Operation Multiple Data. While Very Long Instruction Word (VLIW) processors can provide similar capability, we are able to provide it using a short (32 bit) instruction word, avoiding the complexity and required high bandwidth of long instruction words.

Another advantage over standard SIMD architectures is the ability of either of the sides to access data from the other side's register file. While this saves a lot of swapping when working purely on real data, its greatest value is in how it simplifies and speeds up complex-arithmetic operations. Complex data pairs can be stored at the same register address in the two register files with the real portion residing in the primary register file, and the imaginary portion residing in the secondary register file. Newly defined complex-arithmetic instructions take advantage of this data organization.

A quadword (i.e., 128 bits) datapath between the PPC 440s Data Cache and the PPC 440 FP2 allows for dual data elements (either double-precision or single precision) to be loaded or stored each cycle. The load and store instructions allow primary and secondary data elements to be transposed, speeding up matrix manipulations. While these high bandwidth, low latency instructions were designed to quickly source or sink data for floating-point operations, they can also be used by the system as a high speed means of transferring data between memory locations. This can be especially valuable to the message processor.

The PowerPC 440 FP2 is a superscalar design supporting the issuance of a computational type instruction in parallel with a load or store instruction. Since a fused multiply-add type instruction initiates two operations (i.e., a multiply and an add or subtract) on each side, four floating-point operations can begin each cycle. To help sustain these operations, a dual operand memory access can be initiated in parallel each cycle.

The core supports single element load and store instructions such that any element, in either the primary or secondary register file, can be individually accessed. This feature is very useful when data structures in code (and hence in memory) do not pair operands as they are in the register files. Without it, data might have to be reorganized before being moved into the register files, wasting valuable cycles.

Data are stored internally in double-precision format; any single-precision number is automatically converted to double-precision format when it is loaded. Likewise, when a number is stored via a single-precision operation, it is converted from double to single precision, with the mantissa being truncated as necessary. In the newly defined instructions, if the double-precision source is too large to be represented as a single-

precision value, the returned value is forced to a properly signed infinity. However, round to single precision instructions are provided so that an overflowing value can be forced to infinity or the largest single precision magnitude, based on the rounding mode. Furthermore, these instructions allow for rounding of the mantissa.

All floating-point calculations are performed internally in double precision and are rounded in accordance with the mode specified in the PowerPC defined floating-point status and control register (FPSCR). The newly defined instructions produce the IEEE-754 specified default results for all exceptions. Additionally, a non-IEEE mode is provided for when it is acceptable to flush denormalized results to zero. This mode is enabled via the FPSCR and it saves the need to renormalize denormal results when using them as inputs to subsequent calculations.

All computational instructions, except for divide and those operating on denormal operands, execute with a five cycle latency and single cycle throughput. Division is iterative, producing two quotient bits per cycle. Division iterations cease when a sufficient number of bits are generated for the target precision, or the remainder is zero, whichever occurs first. Faster division can be achieved by employing the highly accurate (i.e., to one part in 2^{13}) reciprocal estimate instructions and performing software-pipelined Newton-Raphson iterations. Similar instructions are also provided for reciprocal square root estimates with the same degree of accuracy.

Since we extended the instruction set beyond the PowerPC architecture, we are developing the necessary compiler enhancements. Library routines and ambitious users can also exploit these enhanced instructions through assembly language, compiler built-in functions, and advanced compiler optimization flags. The double FPU can also be used to advantage by the communications processor, since it permits high bandwidth access to and from the network hardware.

Power is a key issue in such large scale computers, therefore the FPUs and CPUs are designed for low power consumption. Incorporated techniques range from the use of transistors with low leakage current, to local clock gating, to the ability to put the FPU or CPU/FPU pair to sleep. Furthermore, idle computational units are isolated from changing data so as to avoid unnecessary toggling.

The memory system is being designed for high bandwidth, low latency memory and cache accesses. An L2 hit returns in 6 to 10 processor cycles, an L3 hit in about 25 cycles, and an L3 miss in about 75 cycles. L3 misses are serviced by external memory, the system in design has a 16 byte interface to nine 256Mb SDRAM-DDR devices operating at a speed of one half or one third of the processor. While peak memory bandwidths, as indicated in Figure 2 are high, sustained bandwidths will be lower for certain access patterns, such as a sequence of loads, since the 440 core only permits three outstanding loads at a time.

The high level of integration of the BlueGene/L system-on-a-chip approach allows for latencies and bandwidths that are significantly better than those for nodes typically used in ASCI scale supercomputers.

5. Torus Network

The torus network is used for general-purpose, point-to-point message passing and multicast operations to a selected "class" of nodes. The topology is a three-dimensional torus constructed with point-to-point, serial links between routers embedded within the BlueGene/L ASICs. Therefore, each ASIC has six nearest-neighbor connections, some of which may traverse relatively long cables. The target hardware bandwidth for each torus link is 175MB/sec in each direction. Torus networks have been implemented in other machines such as the Cray T3E [ST96].

The general structure of the torus within each node is shown in Figure 5. Packets are injected into the network at one of the Local Injection FIFOs, and are deposited into a Local Reception FIFO upon reaching their destinations. The messaging coprocessor is responsible for injecting and removing packets to and from these FIFOs. Packets that arrive from another node are immediately forwarded in the absence of contention, or stored in a waiting FIFO in the corresponding input unit until the contention clears. Arbitration is highly pipelined and distributed (each input and output unit has its own arbiter), as is common in such switches, e.g., [DDHKX94]. Using a link-level CRC and a HIPPI-like retransmission protocol, the network will reliably deliver a single copy of every packet injected.

The torus network provides both adaptive and deterministic minimal-path routing, and is deadlock free. Throughput and hardware latency are optimized through the use of virtual cut-through (VCT) routing [KK79]. Messages can be composed of multiple packets, which are the atomic unit of routing. Therefore, adaptively routed packets from the same message can arrive out of order. Packets are variable in size, ranging from 32 bytes to 256 bytes with a granularity of 32 bytes.

Virtual channels (VCs) are used to provide deadlock-free adaptive routing and increase throughput [DS87, D92, D93]. We have developed a near cycle accurate simulator of the torus network that has been used for the detailed design of the network. Based on performance studies and a sizing of the hardware requirements, the network will have four VCs. Two VCs will be devoted to adaptive routing, and two will be devoted to deterministic routing. One of the deterministic VCs is used as a "bubble escape channel" [PGPBDI99] for the adaptive sub-network in order to guarantee deadlock freedom, and the other is reserved for high-priority packets. Because it is expected that most traffic will be adaptively routed, two adaptive VCs are provided in order to reduce head-of-line blocking and allow for the use of simple FIFO buffers within the routers. Flow control between routers is provided through the use of tokens. There is sufficient buffer space to maintain full link hardware bandwidth in the absence of contention.

A more detailed description of the network and the network simulator will be given elsewhere, however we now give a short example of its use. One of the most communications intensive operations in scientific computing is the MPI_Alltoall collective communications call in which every processor sends a different message to every other processor. We simulated a representative portion of this call on a 32K node machine and the results are displayed in Table 2. The table illustrates the advantage of dynamic routing over static routing and shows that, because of the highly pipelined nature of the network, the links can be kept busy essentially all the time with only 2 dynamic VCs. For full sized packets, we expect a 12% overhead due to the hardware packet header and trailer, the software packet header, and the token protocol; this accounts for the difference between the total link utilization and the payload link utilization.

	Average Link Utilization (Total)	Average Link Utilization (Payload Only)
Static Routing	76%	66%
1 Dynamic VC	95%	83%
2 Dynamic VCs	99%	87%

Table 2: Estimated link utilizations during the middle of an MPI_Alltoall operation on a 32K node BG/L. In all three cases, the total FIFO sizes are fixed and equal to 3 KB per link. There is no high priority traffic in this exchange.

6. Signaling

The BG/L torus interconnect, and the BG/L tree described below, rely on serial communication. A common system wide clock at the frequency of the processor is used to provide a reference for sending and receiving data. Data are driven on both clock edges from a differential, two bit pre-compensating driver designed to overcome the ~10 decibel of loss on the 8 meter differential cables and connectors. Data are captured by over-sampling using a string of time delayed latches, the location of the data bit is computed by a background state machine that monitors false transitions and tracks changes in arrival or sampling times. To reduce power and allow for minimum silicon delay, a variable delay line after the driver is auto-configured at power-on to optimize the location of the sampled datum in the instrumented delay line. Features such as byte serial/deserializing, parity and CRC generation and checking, message retry, and checksums for error localization are all provided by the hardware.

7. Global Trees

Message passing on the global combining tree is done through the use of a packet structure similar to that of the torus network. The tree network is a token-based network with two VCs. Packets are non-blocking across VCs. Setting programmable control registers flexibly controls the operation of the tree. In its simplest form, packets going up

towards the root of the tree can be either point-to-point or combining. Point-to-point packets are used, for example, when a compute node needs to communicate with its I/O node. The combining packets are used to support MPI collective operations, such as MPI_Allreduce, across all the nodes connected to the tree (e.g., on MPI_COMM_WORLD). All packets coming down the tree are broadcast further down the tree according to the control registers and received upon reaching their destination. For collective operations, the packet is received at each node. The tree has a target hardware bandwidth of 350 MB/sec and a target one-way hardware latency of about 1.5 microseconds on a 64K node partition.

The tree module, shown in Figure 6, is equipped with an integer ALU for combining incoming packets and forwarding the resulting packet. Packets can be combined using bit-wise operations such as XOR or integer operations such as ADD/MAX for a variety of data widths. To do a floating-point sum reduction on the tree requires potentially two round trips on the tree. In the first trip each processor submits the exponents for a max reduction. In the second trip, the mantissas are appropriately shifted to correspond to the common exponent as computed on the first trip and then fed into the tree for an integer sum reduction. Alternatively, double precision floating-point operations can be performed by converting the floating-point numbers to their 2048-bit integer representations, thus requiring only a single pass through the tree network.

A separate set of wires based on asynchronous logic form another tree that enables fast signaling of global interrupts and barriers (global AND or OR). The target latency to perform a global barrier over this network for a 64K node partition is approximately 1.5 microseconds.

8. Software Support

Scalable system software that supports efficient execution of parallel applications is an integral part of the BlueGene/L architecture. A BlueGene/L application is organized as a collection of compute processes, each executing on its own compute node from a partition of the system. I/O nodes provide additional services for the executing application. In this section, we describe the software services provided by compute and I/O nodes for the execution of applications. We also discuss the programming models we are investigating for BlueGene/L, other components of the system software, and certain autonomic features of the machine that we are developing.

8.1 Operating System Architecture

Our goal in developing the system software for BG/L has been to create an environment which looks familiar and also delivers high levels of application performance. The applications get a feel of executing in a Unix-like environment.

The approach we have adopted is to split the operating system functionality between compute and I/O nodes. Each compute node is dedicated to the execution of a single application process. The I/O node provides the physical interface to the file system. The I/O nodes are also available to run processes which facilitate the control, bring-up, job launch and debug of the full BlueGene/L machine. This approach allows the compute node software to be kept very simple.

The compute node operating system, also called the BlueGene/L compute node kernel, is a simple, lightweight, single-user operating system that supports execution of a single dual-threaded application compute process. Each thread of the compute process is bound to one of the processors in the compute node. The compute node kernel is complemented by a user-level runtime library that provides the compute process with direct access to the torus and tree networks. Together, kernel and runtime library implement compute node-to-compute node communication through the torus and compute node-to-I/O node communication through the tree. The compute node-to-compute node communication is intended for exchange of data by the application. Compute node-to-I/O node communication is used primarily for extending the compute process into an I/O node, so that it can perform services available only in that node.

The lightweight kernel approach for the compute node was motivated by the Puma and Cougar kernels at Sandia National Laboratory and the University of New Mexico. The BG/L compute kernel provides a single and static virtual address space to one running compute process. Because of its single-process nature, the BG/L compute kernel does not need to implement any context switching. It does not support demand paging and exploits large pages to ensure complete TLB coverage for the application's address space. This approach results in the application process receiving full resource utilization.

I/O nodes are expected to run the Linux operating system, supporting the execution of multiple processes. Only system software executes on the I/O nodes, no application code. The purpose of the I/O nodes during application execution is to complement the compute node partition with services that are not provided by the compute node software. I/O nodes provide an actual file system to the running applications. They also provide socket connections to processes in other systems. When a compute process in a compute node performs an I/O operation (on a file or a socket), that I/O operation (e.g., a read or a write) is shipped through the tree network to a service process in the I/O node. That service process then issues the operation against the I/O node operating system. The results of the operation (e.g., return code in case of a write, actual data in case of a read) are shipped back to the originating compute node. The I/O node also performs process authentication, accounting, and authorization on behalf of its compute nodes.

I/O nodes also provide debugging capability for user applications. Debuggers running on an I/O node can debug application processes running on compute nodes. In this case, the shipping occurs in the opposite direction. Debugging operations performed on the I/O node are shipped to the compute node for execution against a compute process. Results are shipped back to the debugger in the I/O node.

8.2 Programming Models

Message passing is expected to be the dominant parallel programming model for BG/L applications. It is supported through an implementation of the MPI message-passing library. In developing MPI for BG/L, we are paying particular attention to the issue of efficient mapping of operations to the torus and tree networks. Also important is the issue of efficient use of the second (communication) processor in a compute node. We are also investigating two approaches to the global address space programming model for BG/L: Co-arrays [NuRe98] and Unified Parallel C (UPC) [CDCYBW99].

8.3 Control System

The BG/L system software includes a set of control services that execute on the host system. Many of these services, including system bring up, machine partitioning, measuring system performance, and monitoring system health, are nonarchitected from a user perspective, and are performed through the backdoor JTAG network described in Section 2 (which is also nonarchitected from a user perspective).

The resource management system of BG/L provides services to create electrically isolated partitions of the machine and to allocate resources to jobs. Each partition is dedicated to the execution of a single job at a time. Job scheduling and job control is also performed from the host.

8.4 Autonomic Features

Given the scale of BG/L, there is clearly a need to recover from failures of individual components. Support for long-running applications will be provided through a checkpoint/restart mechanism. We are currently developing an application-assisted checkpoint infrastructure. In this approach, the application programmer is responsible for identifying points in the application in which there are no outstanding messages. The programmer can then place calls to a system checkpoint in those points. When executed, the checkpoint service will synchronize all tasks of the application and take a complete application checkpoint, writing the state of all compute processes to disk. Application state that resides on I/O nodes, particularly file pointers and list of open files, is also saved to disk. In case of unexpected termination, the application can then be restarted from its latest checkpoint.

In addition to being able to reconfigure the machine on a midplane boundary, we have flexibility in the routing hardware to allow for deadlock free routing in the presence of a limited number of isolated faulty nodes. This is accomplished through software by setting the routing parameters on a packet such that the faulty node or link is guaranteed to be avoided. There will be some impact on the network performance for this model. Additionally, this run model will not allow for some of the other hardware features such as hardware row multicast to operate as the faulty node may be in the row in which the

multicast is occurring. The design of the compute node that contains the network routing logic is such that the node can still operate from the network point of view even if there is a hard fault in the remaining compute portions of the node.

9. Validating the architecture with application programs

A wide variety of scientific applications, including many from DOE's NNSA laboratories, have been used to assist in the design of BlueGene/L's hardware and software. BlueGene/L's unique features are especially appealing for ASCI-scale scientific applications. The global barrier and combining trees will vastly improve the scalability and performance of widely-used collective operations, such as MPI_Barrier and MPI_Allreduce. Our analysis shows that a large majority of scientific applications such as SPPM (simplified piecewise-parabolic method), Sweep3D (discrete ordinates neutron transport using wavefronts), SMG2000 (semicoarsening multigrid solver), SPHOT (Monte Carlo photon transport), SAMRAI (Structured Adaptive Mesh Refinement Application Infrastructure) and UMT2K (3D deterministic multigroup, photon transport code for unstructured meshes) use these collective operations to calculate the size of simulation timesteps and validate physical conservation properties of the simulated system. (These programs are described at www.llnl.gov/asci/purple/benchmarks/ and www.llnl.gov/CASC/SAMRAI.) Most applications use MPI's nonblocking point-to-point messaging operations to allow concurrency between computation and communication; BG/L's distinct communication and computation processors will allow the computation processor to transfer overhead for messaging to the communication processor. In addition, we have identified several important applications whose high flops/loads ratio and alternating compute/communicate behavior will allow effective use of the second floating-point unit in each node. We are continuing to study application performance through tracing and simulation analysis, and will analyze the actual hardware as it becomes available. The results of analysis performed with collaborators at the San Diego Supercomputer Center and Cal Tech's Center for Advanced Computing Research will be reported elsewhere.

10. BlueGene science applications development

To carry out the scientific research into the mechanisms behind protein folding announced in December 1999, development of a molecular simulation application kernel targeted for massively parallel architectures is underway. For additional information about the science application portion of the BlueGene project, see [A01]. This application development effort serves multiple purposes: (1) It is the application platform for the Blue Gene Science program. (2) It serves as a prototyping platform for research into application frameworks suitable for cellular architectures. (3) It provides an application perspective in close contact with the hardware and systems software development teams.

One of the motivations for the use of massive computational power in the study of protein folding and dynamics is to obtain a microscopic view of the thermodynamics and kinetics of the folding process. Being able to simulate longer and longer time-scales is

the key challenge. Thus the focus for application scalability is on improving the speed of execution for a fixed size system by utilizing additional CPUs. Efficient domain decomposition and utilization of the high performance interconnect networks on BG/L (both torus and tree) are the keys to maximizing application scalability.

To provide an environment to allow exploration of algorithmic alternatives, the applications group has focused on understanding the logical limits to concurrency within the application, structuring the application architecture to support the finest grained concurrency possible, and to logically separate parallel communications from straight-line serial computation. With this separation and the identification of key communications patterns used widely in molecular simulation, it is possible for domain experts in molecular simulation to modify detailed behavior of the application without having to deal with the complexity of the parallel communications environment as well.

Key computational kernels derived from the molecular simulation application have been used to characterize and drive improvements in the floating point code generation of the compiler being developed for the BG/L platform. As additional tools and actual hardware become available, the effects of cache hierarchy and communications architecture can be explored in detail for the application.

11. References

A01: F. Allen et al., Blue Gene: A vision for protein science using a petaflop supercomputer, IBM Systems Journal, Volume 40, Number 2, 2001, p. 310 (http://www.research.ibm.com/journal/sj/402/allen.html)

CDCYBW99: W. Carlson, J. Draper, D. Culler, K. Yelick, E. Brooks, and K. Warren. "Introduction to UPC and Language Specification." IDA Center for Computing Sciences Technical Report CCS-TR-99-157, 1999.

D92: Dally, W.J. (1992). Virtual-Channel Flow Control. *IEEE Transactions on Parallel and Distributed Systems* 3, No. 2, 194-205.

DDHKX94: Dally, W.J., Dennison, L.R., Harris, D., Kan, K., and Xanthoppulos, T. "Architecture and Implementation of the Reliable Router," In Proceedings of HOT Interconnects II, pp. 122-133, Aug. 1994.

DS87: Dally, W.J. and Seitz, C. "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," IEEE Transactions on Computers, pp. 547-553, May 1987.

D01: K. Dockser. "Honey, I Shrunk the Supercomputer" - The PowerPC TM 440 FPU brings supercomputing to IBM's Blue Logic TM library. MicroNews, 7(4):29–31, November 2001. IBM Microelectronics.

D93: Duato, J. “A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks,” *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No. 12, pp. 1320-1331, Dec. 1993.

ISSCC02: Almasi et al. “Cellular Supercomputing with System-On-A-Chip.” In Proceedings of the 2002 IEEE International Solid-State Circuits Conference.

KK79: Kermani, P. and Kleinrock, L. “Virtual Cut-Through: A New Computer Communication Switching Technique,” *Computer Networks*, Vol. 3, pp. 267-286, 1979.

NuRe98: R. W. Numrich and J. K. Reid. “Co-Array Fortran for parallel programming.” Rutherford Appleton Laboratory Technical Report RAL-TR-1998-060, 1998.

PGPBDI99: Puente, V., Gregorio, J.A., Prellezo, J.M., Beivide, R., Duato, J., and Izu, C. “Adaptive Bubble Router: A Design to Balance Latency and Throughput in Networks for Parallel Computers,” In Proceedings of the 22nd International Conference on Parallel Processing, ICPP ‘99, Sept. 1999.

ST96: Scott, S.L. and Thorson, G.M. “The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus,” In Proceedings of HOT Interconnects IV, Aug. 1996.

QCDOC: A 10-TERAFLOPS SCALE COMPUTER FOR LATTICE QCD. Nucl. Phys. Proc. Suppl. 94: 825-832, 2001.

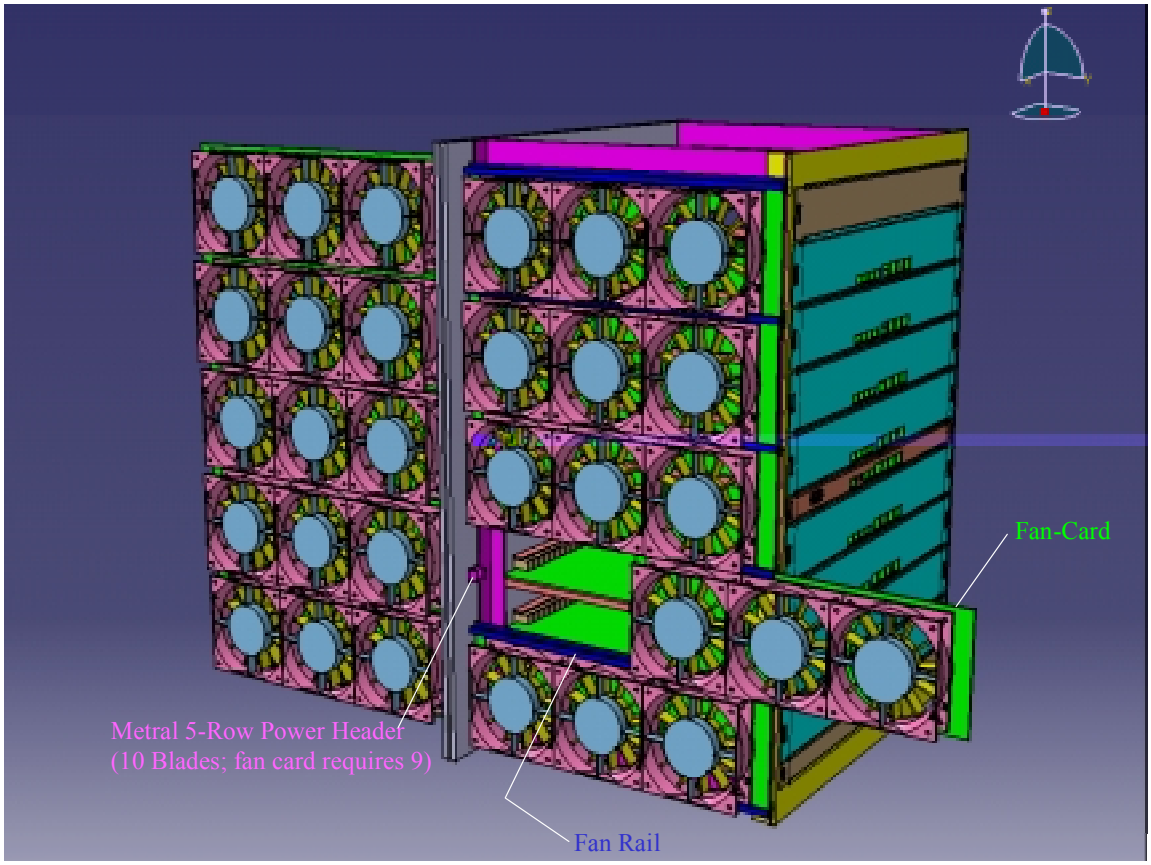

Figure 2: BlueGene/L midplane package.

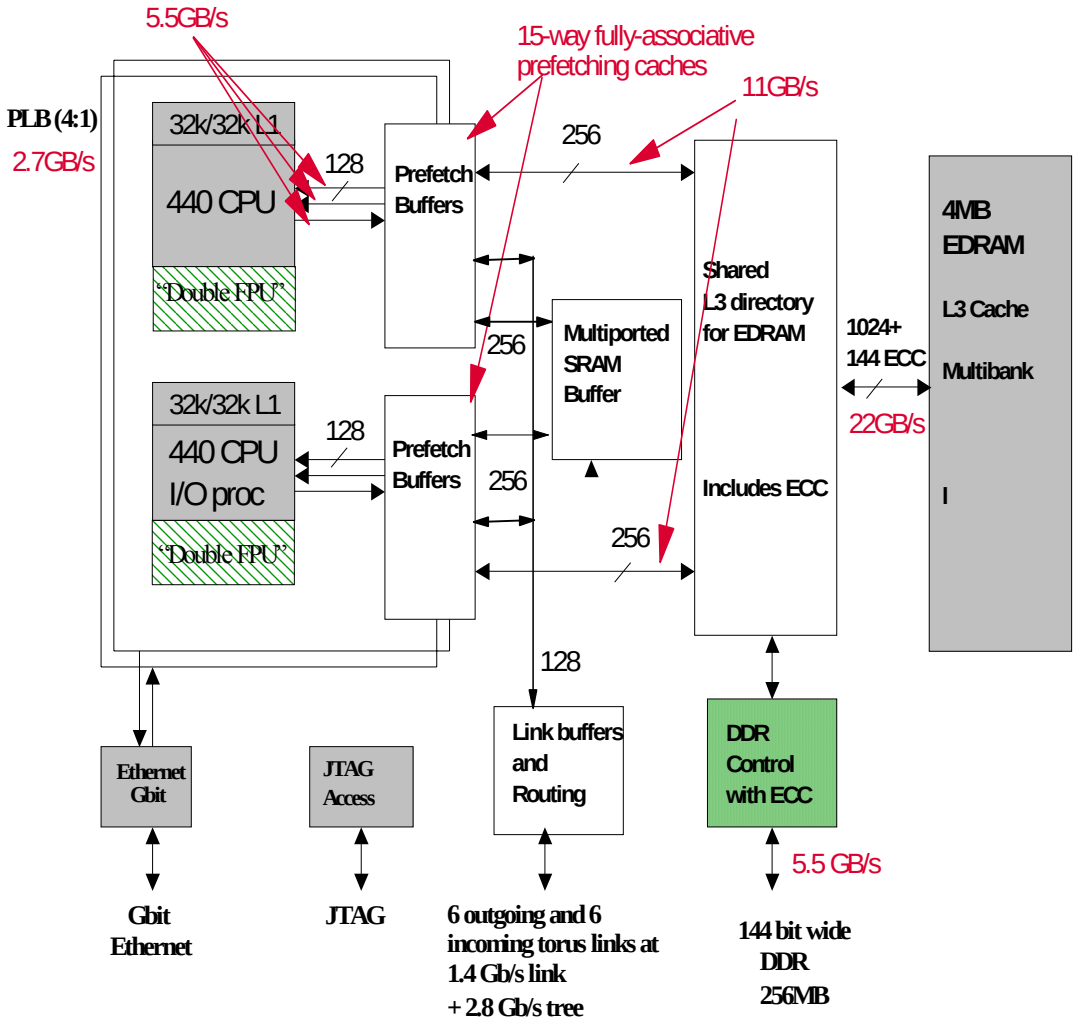

Figure 3: BlueGene/L node diagram. The bandwidths listed are targets.

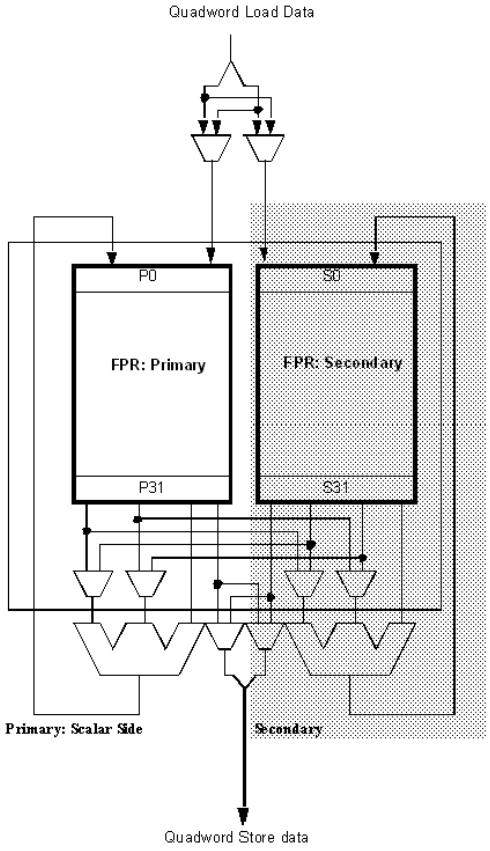

Figure 4: Double FPU architecture.

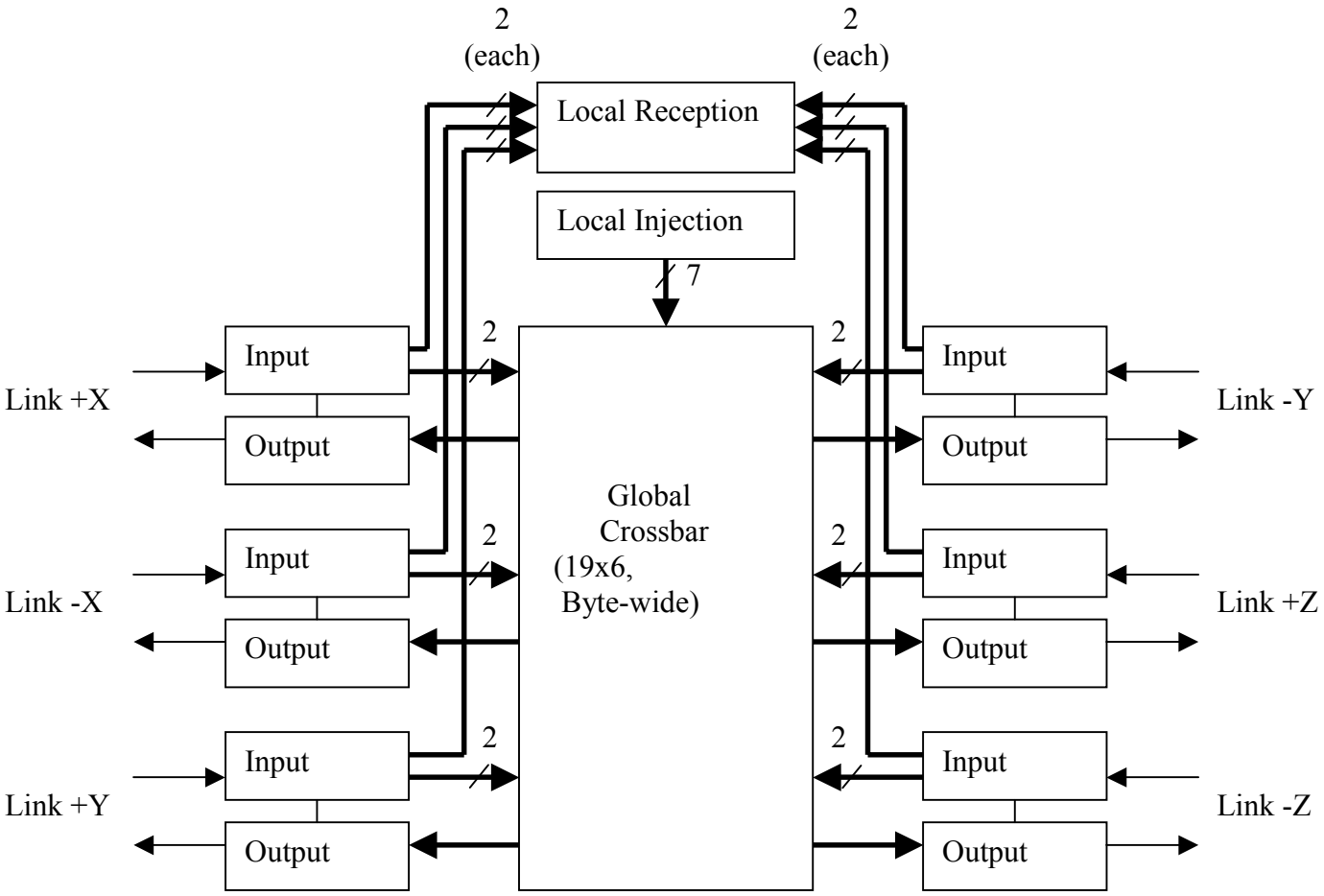

Figure 5: Basic architecture of the torus router.

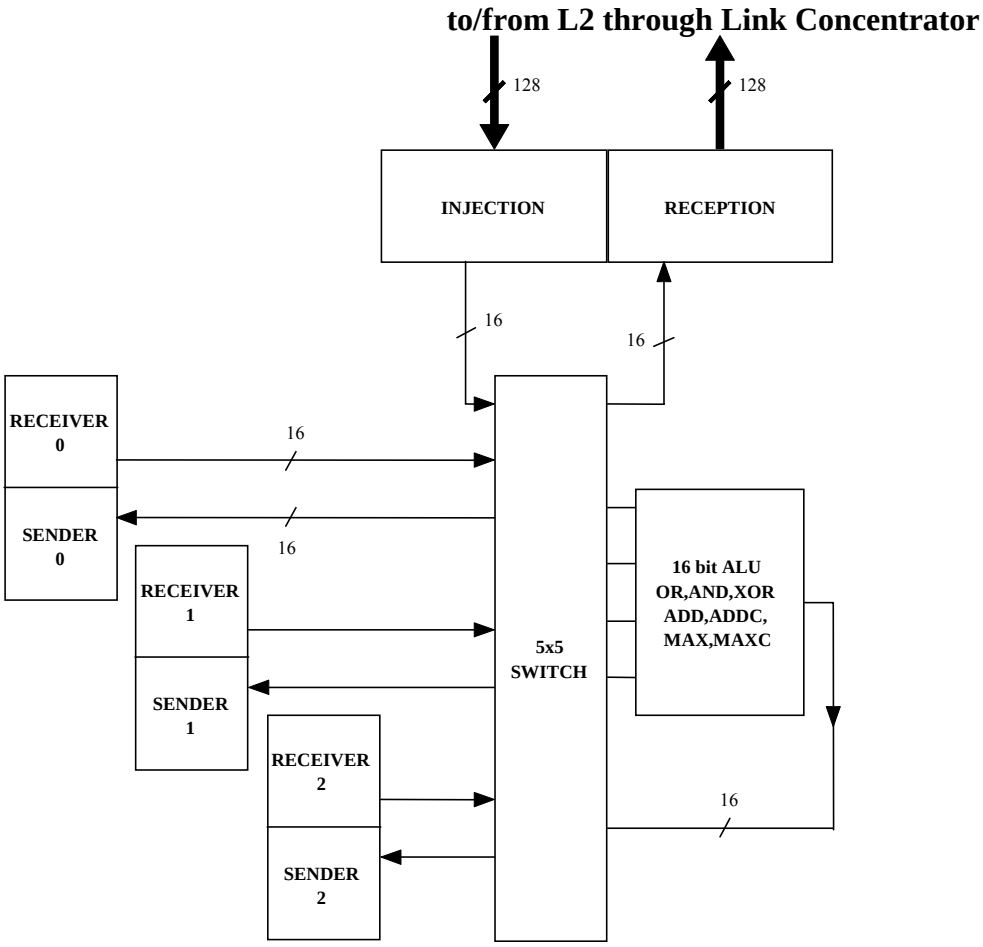

Figure 6: Basic architecture of the tree router.