# Overview of the Blue Gene/L system architecture

A. Gara
M. A. Blumrich
D. Chen
G. L.-T. Chiu
P. Coteus
M. E. Giampapa
R. A. Haring
P. Heidelberger
D. Hoenicke
G. V. Kopcsay
T. A. Liebsch
M. Ohmacht
B. D. Steinmacher-Burow
T. Takken
P. Vranas

*The Blue Gene®/L computer is a massively parallel supercomputer based on IBM system-on-a-chip technology. It is designed to scale to 65,536 dual-processor nodes, with a peak performance of 360 teraflops. This paper describes the project objectives and provides an overview of the system architecture that resulted. We discuss our application-based approach and rationale for a low-power, highly integrated design. The key architectural features of Blue Gene/L are introduced in this paper: the link chip component and five Blue Gene/L networks, the PowerPC® 440 core and floating-point enhancements, the on-chip and off-chip distributed memory system, the node- and system-level design for high reliability, and the comprehensive approach to fault isolation.*

## Introduction

A great gap has existed between the cost/performance ratios of existing supercomputers and that of dedicated application-specific machines. The Blue Gene*/L (BG/L) supercomputer was designed to address that gap by retaining the exceptional cost/performance ratio between existing supercomputer offerings and that obtained by dedicated application-specific machines. The objective was to retain the exceptional cost/performance levels achieved by application-specific machines, while generalizing the massively parallel architecture enough to enable a relatively broad class of applications. The goal of excellent cost/performance meshes nicely with the additional goals of achieving exceptional performance/power and performance/volume ratios.

Our design approach to accomplishing this was to use a very high level of integration that made simplicity in packaging, design, and bring-up possible. This follows the approach of a number of previous special-purpose machines, such as QCDSP [1], that succeeded in achieving exceptional cost/performance. Advances include the areas of floating-point, network, and memory performance, as described in the QCDSP/QCDOC paper [2] in this issue of the *IBM Journal of Research and Development* dedicated to BG/L. To achieve this level of integration, we developed the machine around a processor with moderate frequency, available in system-on-a-chip (SoC) technology. The reasons why we chose an SoC design point included a high level of integration, low power, and low design cost. We chose a processor with modest performance because of the clear performance/power advantage of such a core. Low-power design is the key enabler to the Blue Gene family. A simple relation is

$$\frac{performance}{rack} = \frac{performance}{watt} \times \frac{watt}{rack}.$$

The last term in this expression, *watt/rack*, is determined by thermal cooling capabilities and can be considered a constant of order 20 kW for an air-cooled rack. Therefore, it is the *performance/watt* term that determines the rack performance. This clearly illustrates one of the areas in which electrical power is critical to achieving rack density.

We have found that in terms of *performance/watt*, the low-frequency, low-power, embedded IBM PowerPC* core consistently outperforms high-frequency, high-power microprocessors by a factor of 2 to 10. This is one of the main reasons we chose the low-power design point for BG/L. **Figure 1** illustrates the power efficiency of some recent supercomputers. The data is based on total peak floating-point operations per second divided by total system power, when that data is available. If the data is not available, we approximate it using *Gflops/chip power*.

Using low-power, low-frequency chips succeeds only if the user can achieve more performance by scaling up to

0018-8646/05/$5.00 © 2005 IBM

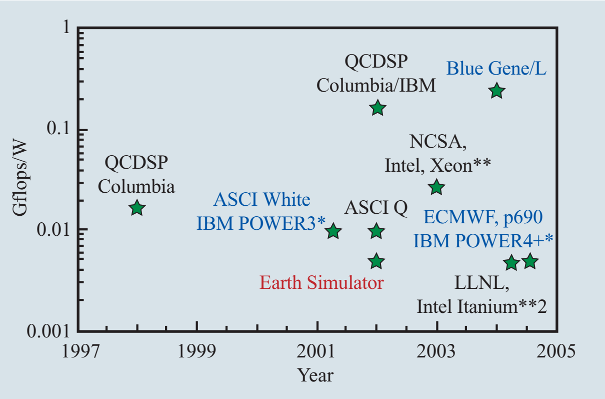

**Figure 1**

Power efficiencies of recent supercomputers. (Blue = IBM machines, black = other U.S. machines, red = Japanese machine.)

a higher number of nodes (processors). Our goal was to address applications that have good scaling behavior because their overall performance is enhanced far more through parallelism than by the marginal gains that can be obtained from much-higher-power, higher-frequency processors.

The importance of low power can be seen in a number of ways. The total power of a 360-Tflops computer based on conventional high-performance processors would exceed 10 megawatts, possibly approaching 20 megawatts. For reference, 10 megawatts is approximately equal to the amount of power used in 11,000 U.S. households [3]. Clearly, this is a fundamental problem that must be addressed. This power problem, while easy to illustrate for a 360-Tflops system, is also of great concern to customers who require high-performance computing at almost all scales. The rate of electrical infrastructure improvements is very slow and the cost is high, compared with those of the computing performance enhancements that have been achieved over the last four decades. Across the industry, technology is leading to further density improvements, but scaling improvements in the power efficiency of computing are slowing dramatically. This portends a difficult future, in which performance gains will have to be made through enhancements in architecture rather than technology. BG/L is an example of one approach to achieving higher performance with an improved power/performance ratio.

A number of challenges had to be overcome to realize good performance using many processors of moderate frequency. These were addressed by assessing the impact on application performance for a representative set of applications. The BG/L networks were designed with extreme scaling in mind. Therefore, we chose networks that scale efficiently in terms of both performance and packaging. The networks support very small messages (as small as 32 bytes) and include hardware support for collective operations (broadcast, reduction, scan, etc.), which will dominate some applications at the scaling limit.

The other critical issue for achieving an unprecedented level of scaling is the reliability, availability, and serviceability (RAS) architecture and support. A great deal of focus was placed on RAS support for BG/L so that it would be a reliable and usable machine, even at extreme scaling limits. Dealing with the sheer scale of supercomputers, whether based on clusters or on custom solutions, has long been one of the most difficult challenges for the entire industry and is likely to become more difficult as the scale of these systems grows. Since we were developing BG/L at the application-specific integrated circuit (ASIC) level, we were able to integrate many features typically found only on high-performance servers. This is an area in which BG/L can be clearly differentiated from commodity cluster solutions based on nodes that were not designed to reach the levels of scalability of supercomputers and therefore do not have the necessary RAS support for extreme scaling.

Another area of critical importance is system software and monitoring. The scalability of the system software and the availability of standard libraries and performance tools is essential for users to achieve the full potential of a massively parallel computer [4].

BG/L was designed to efficiently utilize a distributed-memory, message-passing programming model. While there are a number of message-passing models, Message Passing Interface (MPI) [5] has emerged as the dominant one. Many applications have now been ported to, or developed for, the MPI model, making optimization of its performance a necessity for the BG/L architecture. Hardware features have been added, and functional parameters tuned, to give good performance for MPI applications. The BG/L MPI implementation and its performance are described in [6] in this issue. Application development is a significant portion of the cost of supercomputing. Therefore, BG/L support of the MPI standard and of the standard programming languages C, C++, and Fortran significantly helps keep the cost/performance ratio low when total costs are taken into account.

One of the fundamental sources of complexity in supercomputers is power. By targeting a low-power design point, we dramatically reduced complexity in many areas. This low-power focus has been critical to the success of BG/L in that it simplified the design, verification [7, 8], and bring-up [9] of the machine, and enabled a dense, efficient packaging design.

The packaging constraints in a supercomputer are typically severe as a result of the need for extensive

**Table 1** Limitations to application scaling.

| *Application scaling behavior* | *Scaling limitations* | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | *Amdahl* | *Problem segmentation limits* | *Surface-to-volume communication dominates* | *Load imbalance* | *Small messages* | *Global communication dominates* | *Memory footprint* | *File I/O* |
| Strong scaling | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Weak scaling | ✓ | ✓ | | ✓ | | ✓ | ✓ | ✓ |

internode bandwidth coupled with a great number of high-performance nodes. These constraints typically result in a large demand for overall power, a large number of racks, and a large communication switch infrastructure. Low power has enabled us to integrate 1,024 dual-processor compute nodes into a rack 0.9 m wide, 0.9 m deep, and 1.9 m high that consumes 27.5 kW of total power. Because of the large number of nodes in a single rack, more than 85% of the internode connectivity is contained within the racks. The corresponding dramatic reduction in connectivity across racks allows for higher density, higher reliability, and a generally more manageable system.

From a user perspective, the full-size BG/L supercomputer appears as $2^{16} = 65{,}536$ nodes. Each node has its own 512-MB memory (architecturally expandable to 2 GB) and a complete set of network resources. The architectural size limit to a BG/L machine is, in fact, much larger than 65,536.

## Application-based architectural approach

The BG/L architecture was developed by addressing the needs of applications. It was very important to choose those applications and architectural features that we could support in a cost-effective way [10–13]. The fundamental goal of BG/L was to develop an extremely cost-effective supercomputer. In this pursuit, we chose not to build a machine that would necessarily be appropriate for all applications; rather, our goal was to build an architecture that would achieve superior cost-effectiveness for a broad class of applications. Our approach, therefore, was to restrict ourselves to innovations that enhance scalability with little cost impact. In this way, we were able to hold the cost for the machine at a level competitive with those for cluster and other commodity solutions, while offering many features typically found only in high-end computing environments. Two good examples of this are the integrated, high-performance networks and the RAS system.

### *Limits to scalability*

One of the fundamental issues addressed was scaling. Because we were targeting an unprecedented level of scaling, it was important to address the different scaling behaviors of applications.

In general, scaling is bounded by two different scaling limits: the weak and the strong. Weak scaling relates to scaling the application in such a way that the local data volume remains relatively fixed as the number of nodes is increased. Strong scaling refers to holding the total problem size fixed by reducing the local volume in each node as the node count is increased. In practice, most applications fall somewhere between these limits. Applications from the national laboratories tend more toward the weak scaling limit, while commercial high-performance computing (HPC) applications tend more toward strong scaling.

Scalability is the largest concern for most parallel applications. The different scaling limitations for both strongly and weakly scaled applications are summarized in **Table 1**.

In the table, entries with a check mark indicate potential limitations to scaling. A survey of the applications has shown that while there are multiple possible limitations for the two scaling behaviors, only some limitations dominate the application space. The check marks in color indicate the limitations that are most commonly seen in the application space we have surveyed. It is important to understand how to address these scaling limitations so that we can achieve maximal scalability for this broad class.

A number of interesting observations can be formulated from the table. In particular, Amdahl's law (which states that even a small percentage of serial code can greatly inhibit the speed increase of a program as the number of parallel processors is increased), while certainly a concern, plays a small role in the limitation of current applications. This is likely due to evolution over time, which has resulted in a portfolio of applications that no longer retain serial components. To reach the levels of scalability that have already been achieved by other supercomputers, algorithms must already have reduced their serial computation component to less than 1/1,000, a level at which it is perhaps simpler to eliminate it completely.

Another concern is memory footprint. This constraint is more of a limitation for scaling down than for scaling up, since, in general, the working set per node decreases as one scales up. There are exceptions in which applications keep information or tables that scale with the number of nodes, thereby resulting in memory limitations at large scales. In general, this is seen in few applications, or it has been alleviated by the application developers.

For weakly scaling applications, we see that load imbalance and global communications dominate the scalability of most applications. Load imbalance is usually inherent in the algorithm, and there is little opportunity to ease this effect through machine architecture. The most assistance we can provide is an efficient network to redistribute the computational load, since the cost of redistribution can sometimes limit performance. With respect to global communications, we chose to add considerable support to significantly reduce and in some cases virtually remove the impact of such communications. This is particularly important for BG/L, since scaling to a large number of nodes with moderate performance stresses the global communication capabilities. To address this, we added two additional networks. The *collective network* allows for low-latency global communications, including global arithmetic operations, such as global summation; and the *global barrier network* allows for barriers and notifications over the entire machine, propagated through combinatorial logic, resulting in extremely low latency.

Small messages are both a bandwidth and a latency issue for strongly scaling applications. For such applications, one finds that the message size falls linearly for most clustered point-to-point communications, while it is closer to falling quadratically with respect to the node number for global all-to-all-type communication patterns. For example, doubling the number of nodes with an all-to-all-type communication pattern would translate into each node sending twice as many messages, each one-quarter the previous size, while the total volume would remain roughly unchanged. Ideally, one would like the communication time to fall linearly with the number of nodes in a manner similar to the behavior of the computation. For this reason, we have supported very small messages in hardware (as small as 32 bytes) with minimal latencies. We have also included hardware support for multicast, which can greatly reduce the number of messages that have to be sent for collective operations on groups of nodes.

For strongly scaling applications, we have the additional issue of surface-to-volume effects. This refers to a common application development model in which each node is responsible for a small subvolume of the full multidimensional problem. The subvolume is often represented by a multidimensional cell configuration. In this case, each node updates the parameters associated with its own local cells. This computational stage is usually proportional to the number of cells. The total number of cells can be referred to as the volume. After the computation, there is often a communication stage during which information about the cells on the surface of the multidimensional local volume is communicated to the neighbors. Because this communication is proportional to the number of cells on the surface, it has the effect that the communication-to-computation ratio increases as the problem is spread over more nodes.

### Classes of applications under consideration

The classes of applications that we addressed can be broadly separated into three categories:

- Simulations of physical phenomena.
- Real-time data processing.
- Offline data analysis.

The typical applications run in national laboratories and supercomputer centers fall predominantly into the first category: simulations of physical phenomena. Since the application investigations were done in close collaboration with national laboratories (Lawrence Livermore Laboratory in particular) and supercomputer centers (predominantly the San Diego Supercomputer Center), many of our early application evaluations were of the simulation type. Applications at the national laboratories differ from commercial HPC applications in that they are often scaled in a weak manner, as previously mentioned.

Simulations of physical phenomena are most frequently performed by representing the physical state variables on a lattice. The time evolution of the simulation is represented by a sequence of time steps during which the local state variables are updated according to physical laws. This updating is based on the values of the state variables on other lattice locations. For spatial decompositions, communications are often local in nature (with respect to lattice sites), mirroring the general short-range behavior of most real physical systems. Spectral methods often result in long-range communication requirements between lattice sites, but such methods sometimes display superior convergence behavior. Monte Carlo techniques are often used to generate statistical samples for various initial conditions.

The mapping of the multiple lattice sites onto the multiple processors of a parallel computer is a fairly straightforward process. Each node in the parallel processor is allocated a subset of the total problem. For spatial decomposition, this leads to each node communicating with a modest group of other nodes. This is representative of one of the communication patterns seen over a broad range of applications—namely, the communication of each node with a modest group of other nodes. Another common communication pattern is a global reduction associated with a normalization or decision step in an algorithm.

For BG/L, we chose to address these application requirements through the use of multiple well-suited networks. The torus network is especially efficient for local node communication. This is the case for many applications, especially those that are spatially decomposed. In a torus network, locality of communication can be accomplished through appropriately assigning processing tasks to processing nodes in a geometry that mirrors the geometry of the physical problem. Adaptive mesh refinement techniques can result in this locality being somewhat compromised, although a large degree of locality can be accomplished via a number of techniques, such as task migration, in which the tasks are incrementally reassigned to processing nodes when locality is compromised.

BG/L addresses the common need for global communications by supporting an independent network to handle global communications and some global operations. This network, called the *collective network*, provides arithmetic support to allow operations such as global summation and determining a global maximum to achieve latencies at orders of magnitude faster than most supercomputers. This is especially important because the scaling behavior of applications shows that these global operations are increasingly becoming the critical path and can limit overall scalability. Similarly, BG/L has an independent network to address the need for low-latency global barriers, another concern as applications reach unprecedented scale.

## System components

BG/L is a scalable supercomputer that, when completed, will be composed of 65,536 nodes, produced in 130-nm copper IBM CMOS 8SFG technology [14]. Each node is very simple, consisting of a single ASIC containing two processors and nine double-data-rate (DDR) synchronous dynamic random access memory (SDRAM) chips. Each node can also be assembled with 18 SDRAM chips per ASIC. The nodes are interconnected through five networks, the most significant of which is a 64 × 32 × 32 three-dimensional torus that has the highest aggregate bandwidth and handles the bulk of all communication. There are virtually no asymmetries in this interconnect; the nodes communicate with neighboring nodes that are physically close on the same board and with nodes that are physically far removed on a neighboring rack, with the same bandwidth and nearly the same latency. This allows for a simple programming model because there are no edges in a torus configuration. The SoC ASIC that powers the node incorporates all of the functionality needed by BG/L. It also contains 4 MB of extremely high-bandwidth embedded DRAM [15] that is of the order of 30 cycles from the registers on most L1/L2 cache misses.

The nodes themselves are physically small, allowing for very high packaging density. High density is important for reaching an optimum cost/performance point. As density decreases, system size and cost grow, while reliability suffers because the number of connectors and cables increases. Power is a critical parameter because the densities that we achieve are a factor of 2 to 10 greater than those available with traditional high-frequency uniprocessors. In addition, there are serious cost and reliability issues associated with high-power, high-density designs.

The system packaging for Blue Gene/L [16] calls for 512 processing nodes, each with a peak performance of 5.6 Gflops, on a doubled-sided board, or *midplane*, with dimensions of approximately 20 in. by 25 in. Each node contains two processors, which makes it possible to vary the running mode. For instance, each processor can handle its own communication (*virtual node mode*), or one processor can be dedicated to communication and one to computation (*communication coprocessor mode*).

In addition to the compute nodes, there are input/output (I/O) nodes that are used to communicate with the file system. Each compute node has a small operating system that can handle basic I/O tasks and all functions necessary for high-performance code. For file systems, compiling, diagnostics, analysis, and service of BG/L, an external host computer (or computers) is required. The I/O nodes contain a software layer above the layer on the compute nodes to handle communication with the host. The choice of host depends on the class of applications and their bandwidth and performance requirements.

Another important element of the BG/L architecture is the ability to handle multiple users simultaneously. This is accomplished by partitioning the machine space in a manner that enables each user to have a dedicated set of nodes for their application, including dedicated network resources. This partitioning is accomplished by using *link chips* that also redrive signals at the boundaries of a midplane. This partitioning is configured via host software and is static with respect to the user application. This partitioning is utilized by a resource allocation system which optimizes the placement of user jobs on hardware partitions in a manner consistent with the hardware constraints [17].

### Link chip overview

At the midplane boundaries, all BG/L networks pass through the BG/L link chip. Like the BG/L compute (BLC) chip, the link chip is also an ASIC in IBM Cu-11 technology. For the networks and other common

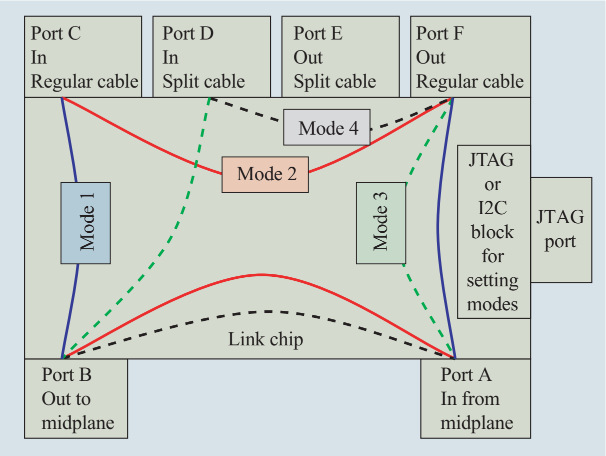

Figure 2

Blue Gene/L link chip switch function. Four different modes of using the link chip.

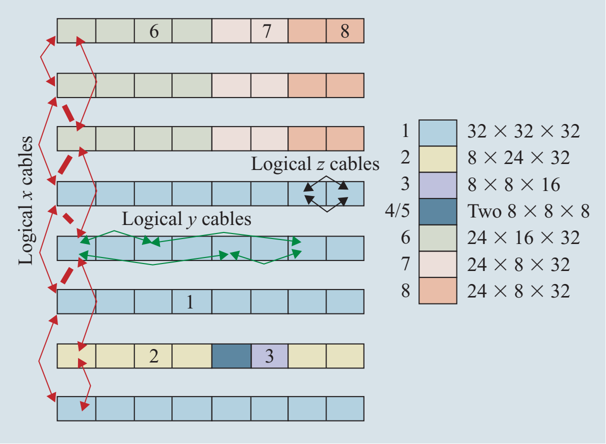

Figure 3

A 64-rack Blue Gene/L space-partitioned for eight users.

functions, the link chip and the BLC chip use the same logic. The link chip serves two functions. First, it redrives signals over the cables between the midplanes, restoring the high-speed signal shape and amplitude in the middle of a long lossy trace–cable–trace connection between compute ASICs on different midplanes. Second, the link chip can redirect signals between its different ports. This redirection function allows BG/L to be partitioned into multiple logically separate systems.

As illustrated in **Figure 2**, the link chip has six ports, four of which are in use at any time. Ports A and B are connected directly to nodes in a midplane. The other four ports are connected to cables. The logic inside the link chip supports arbitrary static routing of any port to any other port. This routing is set by the host at the time the partition is created and is static until another partition is created or reconfigured.

Each link chip port serves 16 unidirectional torus links entering and exiting the midplane. A few more signals serve the collective network and the barrier network, and provide spares and parity. The six $x+$, $x-$, $y+$, $y-$, $z+$, $z-$ faces of the 8 × 8 × 8 node midplane are each $8 \times 8 = 64$ nodes in size. Thus, each midplane is served by 24 link chips. This is implemented using four identical link cards, each containing six link chips.

The link chips are also used for reliability and availability of the BG/L machine by reconfiguring a system following a hardware fault in such a way that the midplane containing the faulty node is no longer in the user partition. This reconfiguring is accomplished by redirecting the port connections within the link chip. This allows a user application to restart from a checkpoint in such a manner that the machine appears and behaves in a way analogous to the way the user partition was originally configured. The ability of the link chip to interconnect multiple midplanes while bypassing a fixed set of midplanes allows for a flexible set of possible user partitions. More detail on the internal functionality of the link chips, including the modes illustrated in Figure 2, is presented in [16] in this issue.

**Figure 3** portrays a possible partitioning of the BG/L system. The lines illustrate some of the cables connecting the midplanes via the link chips. The thick lines illustrate cables allowing the $x$-dimension to be partitioned. The link chip provides the ability to divide the 64 racks (128 midplanes) into many user partitions, specified by the different colors in the figure.

### *Blue Gene/L networks*

The BG/L computer uses five interconnect networks for I/O, debug, and various types of interprocessor communication. Gigabit Ethernet is used to support file system access. Fast Ethernet (100 Mb/s) and JTAG (IEEE Standard 1149.1, developed by the Joint Test Action Group) are used for diagnostics, debugging, and some aspects of initialization. Three types of high-bandwidth, low-latency networks make up the interprocessor BG/L “fabric.” In this section, we briefly explain these networks. Considerably more detail may be found in the papers in this issue dedicated to the BG/L networks [18–20]. All network logic is integrated into the BG/L node ASIC. The three interprocessor networks are utilized via memory-mapped interfaces available from user space, allowing for minimal software overhead.

### Three-dimensional torus

The network used for the majority of application messaging is the torus network [19]. This network supports low-latency, high-bandwidth point-to-point messaging. Through this network, any node can communicate with any other node without restriction. Since the physical network comprises a nearest-neighbor interconnect, communication with remote nodes may involve transit through many other nodes in the system. This results in each node sharing its network bandwidth with cut-through traffic, resulting in a communication-pattern-dependent "effective bandwidth." For this reason, algorithms that keep much of the communication local, in a three-dimensional (3D) sense, make use of the available torus bandwidth most effectively. This also requires that a communication-intensive application be mapped to the physical BG/L machine in a way that preserves the locality as much as possible. This is addressed in the performance papers in this issue [10, 12, 21].

The physical machine architecture is most closely tied to the 3D torus. **Figure 4(a)** shows a 2 × 2 × 2 torus—a simple 3D nearest-neighbor interconnect that is "wrapped" at the edges. All neighbors are equally distant, except for generally negligible time-of-flight differences, making code easy to write and optimize. The signaling rate for the nearest-neighbor links is 1.4 Gb/s in each direction. Each node supports six independent, bidirectional nearest-neighbor links, with an aggregate bandwidth of 2.1 GB/s. The hardware latency to transit a node is approximately 100 ns. For the full 64Ki-node[1] machine configured as 64 × 32 × 32 nodes, the maximum number of node transits, or hops, is equal to $32 + 16 + 16 = 64$ hops, giving a worst-case hardware latency of 6.4 $\mu$s.

Considerable effort was put into the design of the torus routing, described in detail in [19]; a few highlights follow. The torus network supports cut-through routing, which enables packets to transit a node without any software intervention. In addition, the routing is adaptive, allowing for good network performance, even under stressful loads. Adaptation allows packets to follow any minimal path to the final destination, allowing packets to dynamically "choose" less congested routes. Four virtual channels are supported in the torus network, contributing to efficient, deadlock-free communication. Another property integrated in the torus network is the ability to do multicast along any dimension, enabling low-latency broadcast algorithms.

Each midplane is an 8 × 8 × 8 mesh. The surfaces of the mesh are all exposed, in terms of cabling, allowing this mesh to be extended between midplanes in all dimensions. A torus is formed in the usual way by interleaving midplanes to avoid the long cabling required at the end of a long succession of midplanes. The link chips are used to partition the BG/L machine into multiple user partitions, each of which has an independent torus.

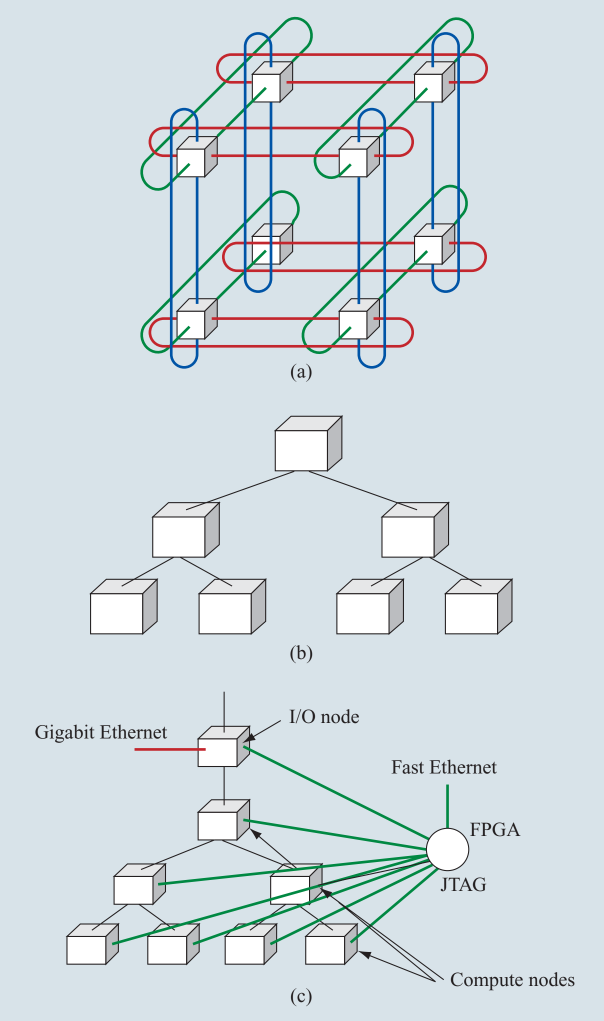

Figure 4

(a) Three-dimensional torus. (b) Global collective network. (c) Blue Gene/L control system network and Gigabit Ethernet networks.

### Collective network

The collective network extends over the entire BG/L machine, allowing data to be sent from any node to all others (broadcast), or a subset of nodes, with a hardware latency of less than 5 $\mu$s.[2] Every link of this collective network has a target bandwidth of 2.8 Gb/s, or 4 bits per processor cycle, in both the transmit and receive

[1] The unit "Ki" indicates a "kibi"—the binary equivalent of kilo (K). See *http://physics.nist.gov/cuu/Units/binary.html*.

[2] D. Hoenicke, M. A. Blumrich, D. Chen, A. Gara, M. E. Giampapa, P. Heidelberger, L.-K. Liu, M. Lu, V. Srinivasan, B. D. Steinmacher-Burow, T. Takken, R. B. Tremaine, A. R. Umamaheshwaran, P. Vranas, and T. J. C. Ward, "Blue Gene/L Global Collective and Barrier Networks," private communication.

directions. Each node has three links, with one or two or three physically connected to other nodes. A simple collective network is illustrated in **Figure 4(b)**.

Arithmetic and logical hardware is built into the collective network to support integer reduction operations including min, max, sum, bitwise logical OR, bitwise logical AND, and bitwise logical XOR. This was an important design element, since it is recognized that current applications spend an increasing percentage of their time performing collective operations, such as global summation. The latency of the collective network is typically at least ten to 100 times less than the network latency of typical supercomputers, allowing for efficient global operation, even at the scale of the largest BG/L machine.

The collective network is also used for global broadcast of data, rather than transmitting it around in rings on the torus. For one-to-all communications, this is a tremendous improvement from a software point of view over the nearest-neighbor 3D torus network. The broadcast functionality is also very useful when there are one-to-all transfers that must be concurrent with communications over the torus network. Of course, a broadcast can also be handled over the torus network, but it involves significant synchronization effort and has a longer latency. The bandwidth of the torus can exceed the collective network for large messages, leading to a crossover point at which the torus becomes the more efficient network.

A global floating-point sum over the entire machine can be done in approximately 10 $\mu$s by utilizing the collective network twice. Two passes are required because the global network supports only integer reduction operations. On the first pass, the maximum of all exponents is obtained; on the second pass, all of the shifted mantissas are added. The collective network partitions in a manner akin to the torus network. When a user partition is formed, an independent collective network is formed for the partition; it includes all nodes in the partition (and no nodes in any other partition).

The collective network is also used to forward file-system traffic to I/O nodes, which are identical to the compute nodes with the exception that the Gigabit Ethernet is wired out to the external switch fabric used for file-system connectivity.

The routing of the collective network is static but general in that each node contains a static routing table that is used in conjunction with a small header field in each packet to determine a *class*. The class is used to locally determine the routing of the packet. With this technique, multiple independent collective networks can be virtualized in a single physical network. Two standard examples of this are the class that connects a small group of compute nodes to an I/O node and a class that includes all compute nodes in the system. In addition, the hardware supports two virtual channels in the collective network, allowing for nonblocking operations between two independent communications.

### Barrier network

As we scale applications to larger processor and node counts, the latency characteristics of global operations will have to improve considerably. We have implemented an independent barrier network to address this architectural issue. This network contains four independent channels and is effectively a global OR over all nodes. Individual signals are combined in hardware and propagate to the physical top of a combining tree. The resultant signal is then broadcast down this tree. A global AND can be achieved by using inverted logic. The AND is used as a global barrier, while the OR is a global interrupt that is used when the entire machine or partition must be stopped as soon as possible for diagnostic purposes. The barrier network is optimized for latency, having a round-trip latency of less than 1.5 $\mu$s for a system size of 64Ki nodes. This network can also be partitioned on the same midplane boundaries as the torus and collective networks.

### Control system networks

The 64Ki-node Blue Gene/L computer contains more than 250,000 endpoints in the form of ASICs, temperature sensors, power supplies, clock trees, fans, status light-emitting diodes, and more, and all must be initialized, controlled, and monitored [20]. These actions are performed by an external commodity computer, called the *service node*, which is part of the host computer. The service node accesses the endpoints through a commodity intranet based on Ethernet. At the board level, a field-programmable gate array (FPGA) called the control–FPGA chip converts 100-Mb Ethernet packets to various control networks, such as I2C. As illustrated in **Figure 4(c)**, the control–FPGA also converts from Ethernet to serial JTAG. As described in [20] and [22] in this issue, JTAG is used for initial program load and debug access to every node, which makes host control of the BG/L nodes very simple and straightforward. JTAG also allows access to the registers of every processor through, for example, the IBM RiscWatch software running on the host.

### Gigabit Ethernet network

As illustrated in Figure 4(c), I/O nodes also have a Gigabit Ethernet interface used to access external Ethernet switches [18]. These switches provide connectivity between the I/O nodes and an external parallel file system, as well as the external host. The number of I/O nodes is configurable, with a maximum

I/O-to-compute node ratio of 1:8. If BG/L is configured to a 1:64 ratio with 64 racks, it results in 1,024 I/O nodes, with an aggregate I/O bandwidth of more than one terabit per second.

## Blue Gene/L node overview

The BLC ASIC that forms the heart of a BG/L node is a SoC built with the IBM Cu-11 (130-nm CMOS) process. Integrating all of the functions of a computer into a single ASIC results in dramatic size and power reductions for the node. In a supercomputer, this can be further leveraged to increase node density, thereby improving the overall cost/performance for the machine. The BG/L node incorporates many functions into the BLC ASIC. These include two IBM PowerPC 440 (PPC440) embedded processing cores, a floating-point core for each processor, embedded DRAM, an integrated external DDR memory controller, a Gigabit Ethernet adapter, and all of the collective and torus network cut-through buffers and control. The same BLC ASIC is used for both compute nodes and I/O nodes, but only I/O nodes utilize the Gigabit Ethernet for host and file system connectivity.

The two PPC440s are fully symmetric in terms of their design, performance, and access to all chip resources. There are no hardware impediments to fully utilizing both processors for applications that have simple message-passing requirements, such as those with a large compute-to-I/O ratio or those with predominantly nearest-neighbor communication. However, for some other applications, one processor is dedicated to message handling and the other executes the computation of the application.

### *BG/L ASIC block diagram*

A block diagram of the BLC ASIC is shown in **Figure 5**. The green blocks in the diagram are cores that are available from the standard IBM ASIC library for use by internal and external customers. The boxes marked "Double-hummer FPU" are new cores for which there exist related, previous-generation devices. The double-hummer FPU consists of two coupled standard floating-point units (FPUs), giving a peak performance of four floating-point operations per processor cycle. The tan blocks represent new additions and were developed using standard design methodology.

### *PPC440 core description*

Each of the two cores in the BLC ASIC is an embedded PPC440 core, designed to reach a nominal clock frequency of 700 MHz (1.4 giga-operations per second). The core is illustrated in **Figure 6**. The PPC440 is a high-performance, superscalar implementation of the full 32-bit Book-E Enhanced PowerPC Architecture*. The power target of 1 W is internally achieved using extensive power management. The PPC440 has a seven-stage, highly pipelined microarchitecture with dual instruction fetch, decode, and out-of-order issue. It also has out-of-order dispatch, execution, and completion. A branch history table (BHT) provides highly accurate dynamic branch prediction. A branch target address cache (BTAC) reduces branch latency. The PPC440 contains three independent pipelines: a load/store pipeline, a simple integer pipeline, and a combined complex integer, system, and branch pipeline. The 32 × 32 general-purpose register (GPR) file is implemented with nine ports (six read, three write) and is replicated. Multiply and multiply–accumulate have single-cycle throughput. The independent 32-KB L1 instruction and data caches have a 32-byte line and 64-way associativity with round-robin replacement. The L1 data cache supports write-back and write-through operations and is nonblocking with up to four outstanding load misses. The memory management

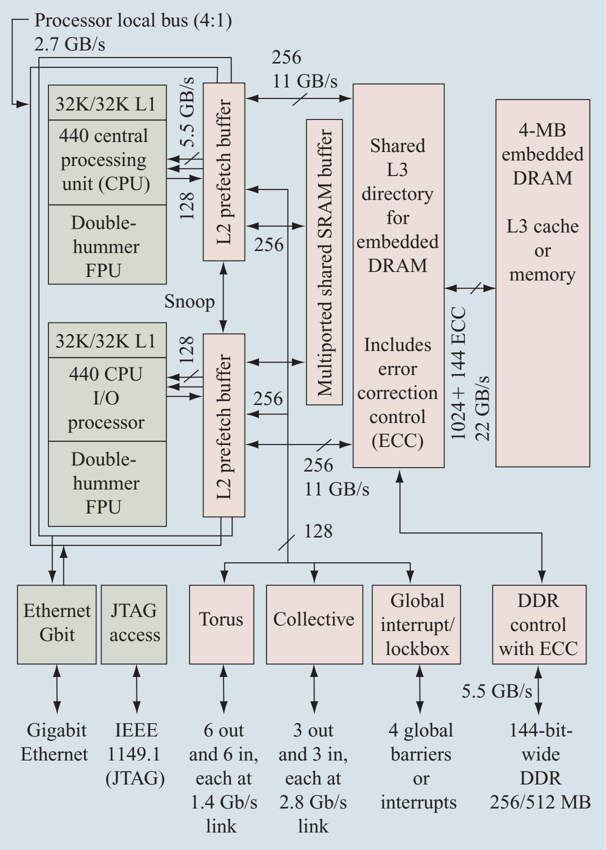

Figure 5

Blue Gene/L compute (BLC) chip architecture. Green shading indicates off-the-shelf cores. ©2002 IEEE. Reprinted with permission from G. Almasi et al., "Cellular Supercomputing with System-on-a-Chip," *Digest of Technical Papers*, 2002 IEEE International Solid-State Circuits Conference.

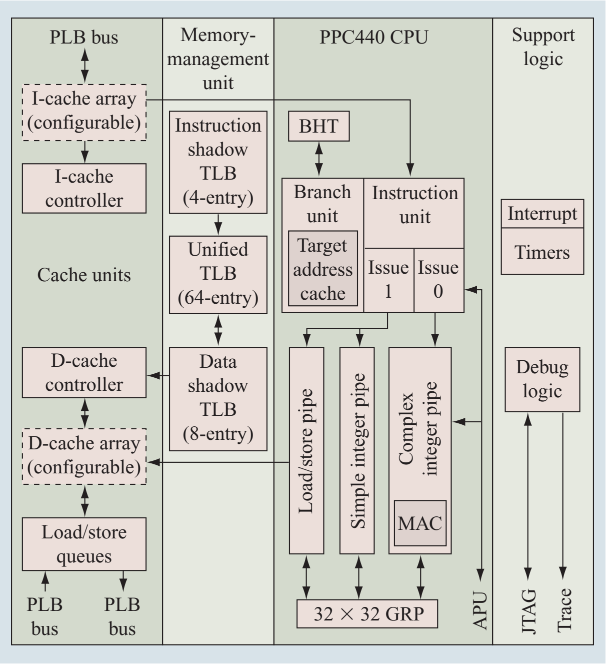

Figure 6

PowerPC 440 core.

unit is a 64-entry fully associative unified translation-lookaside buffer (TLB) supporting variable page sizes and four user-definable storage attributes. The PPC440 has three independent 128-bit interfaces for instruction reads, data reads, and data writes, collectively known as the processor local bus (PLB). BG/L uses a central-processing-unit-(CPU)-to-PLB frequency ratio of 2:1, the highest supported. The BG/L double FPU, described in the next section, attaches to the auxiliary processor unit (APU) port of the PPC440 and uses its 128-bit load/store capability.

### *Floating-point enhancements*

For Blue Gene/L, there were additional requirements for floating-point performance: namely, increased floating-point data bandwidth and the ability to perform two floating-point multiply–adds per cycle [23, 24]. This allowed for an additional level of power-efficient parallelism, achieving a processor with a peak double-precision performance of 2.8 Gflops. The performance/watt ratio of this approach compares favorably with the result that could be achieved by a higher-frequency approach. Although this approach could be pushed further to include a wider degree of parallelism, we did not feel that this would result in significant performance improvement because of memory system limitations and the inherent low degree of fine-grained parallelism in many applications of interest.

As described in [24], the double-hummer FPU implemented in BG/L has a primary and a secondary side, each with its own arithmetic pipe and register file. The 128-bit datapath allows for quadword loads and stores with single-cycle throughput for floating-point computations or for general data movement, such as filling and draining the torus and collective networks. Two floating-point multiply–add operations can be dispatched and executed in one cycle by splitting these execution units into a real and imaginary complex pair. Newly architected single-instruction multiple-data-like (SIMD-like) instructions feed this complex pair with the ability to execute two floating-point multiply–adds per cycle, while the normal scalar instruction utilizes only the real or primary side. These microarchitecture changes are instruction-set extensions beyond the PowerPC Book E specification and require compiler enhancements and support. Library routines and ambitious users can also exploit these enhanced instructions through assembly language. Detailed use of the double-hummer FPU is described in [23–26] in this issue.

The FPU specifications are as follows:

- High-performance, dual-issue, superscalar FPU with SIMD extensions.
- ANSI/IEEE 754-1985-compliant FPU.
- PowerPC Book-E compliant for scalar operations.
- Single-cycle throughput for most instructions.
- Independent load/store and execution units.
- Out-of-order execution and completion.
- 128-bit load/store capability.
- 2.8 Gflops peak performance single-precision/double-precision at 700 MHz.

## Memory system overview

This section describes the architecture of the BG/L distributed memory system [18], which includes an on-chip cache hierarchy, an off-chip main store, and optimized on-chip support for locking and communication between the two processors. The memory system is architected for peak performance over the range of target applications described in the application papers in this issue and elsewhere [10, 11, 13].

The aggregate memory of the machine is completely distributed in the style of a multicomputer, with no hardware sharing between nodes. For the 64Ki-node BG/L, each node has 512 MB of physical memory, resulting in a total of 32 TB. The 512 MB is a compromise between the cost of memory and the demand for memory exhibited by applications. The physical memory of a node is shared by the two processors within the ASIC.

**Figure 7** shows the basic components of a single-node memory system. The first-level caches (L1) are contained within the PPC440 core macro. The second-level caches (L2R and L2W) are very small and basically serve as prefetch and write-back buffers for L1 data. The third-level cache (L3) is large and is expected to provide high-bandwidth, low-latency access. It is shared by instructions and data. Finally, the main store is composed of off-chip DDR SDRAM. The following sections briefly describe each of the memory system components and various attributes of the memory system in more detail. These are covered in much more detail in the memory system paper [18] in this issue.

### Memory system

A great deal of effort was applied to the design of the BLC memory system. We strove to provide as balanced a memory system as possible within the constraints provided by the existing PPC440 core. Our goal was to provide both high bandwidth and low latency, and the SoC solution is well suited to this goal. Each PPC440 processor has independent 16-byte read and write data buses and an independent 16-byte instruction bus. The bandwidth through the second- and third-level caches and to external memory is commensurate with the maximum bandwidth achievable by the PPC440 processor. This nearly flat memory system is very attractive for many memory-intensive applications.

The sustained bandwidth for random reads from the various levels of the memory hierarchy is limited by the PPC440 limit of four outstanding loads before stall. For sequential access, the L2 can prefetch from any level of the memory hierarchy nearly as fast as the PPC440 can consume data from it. The latencies and bandwidths for these two reference patterns are listed in **Table 2**. The latency numbers correspond to the latency associated with pointer chasing.

The L2 is relatively small (2 KB) and consists of multiple prefetch engines along with a modest number of prefetch buffers. The line width of the L2 and L3 is 128 bytes. The L2 detects streamed data access patterns automatically and supports multiple simultaneous prefetching streams. An access from the PPC440 that is a hit in the L2 returns in approximately two processor cycles from the time the L2 observes the request. The L3 also supports prefetching. The prefetching modes of the L2 and L3 can be controlled via user-definable bits in the PPC440 TLB.

The L3 is composed of embedded DRAM, described in [15] in this issue. This 4-MB embedded DRAM can be segmented into two sections, one of which is memory-mapped, with the other used as an L3 cache. The relative size of the sections is adjustable, allowing for additional flexibility that can be leveraged to enhance overall memory system efficiency.

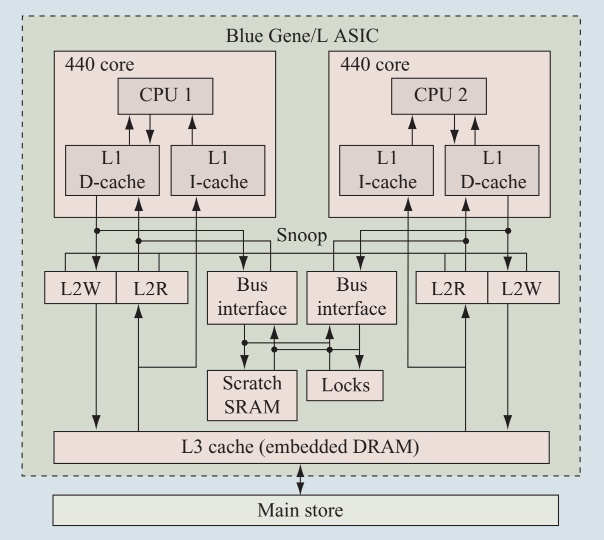

Figure 7

Single-node memory system.

A low-latency, high-bandwidth datapath is provided to a 16-KB SRAM shared between the processors. This memory is not coherent with respect to the two on-chip processors, but is expected to be used in conjunction with a dedicated hardware lock mechanism to exchange packet descriptors between processors. This dedicated hardware lock mechanism allows for low-latency locks in hardware.

### Coherence

The coherence of the memory system on a BG/L node warrants some discussion in this overview. Since the PPC440 cores do not have support for coherence, software must assist in managing coherence at the L1 level. The L2, L3, and main store levels are sequentially consistent, with coherence provided by hardware.

The L2 cache is not inclusive of the L1 because the L2 is so small. Furthermore, the L3 cache is not inclusive of the L1 because it is shared, so that one processor can cause evictions of lines cached on behalf of the other. Therefore, both the L2 and L3 caches are designed to handle higher-level evictions of modified lines that they may no longer be caching. Because the L1 and L3 caches can be operated in write-back mode, it is possible for a modified line to exist at any cache level.

The hardware is designed to support the following two operational modes, neither of which requires user applications to manage L1 coherence:

- *Communication coprocessor mode:* In this mode, one of the processors is dedicated to messaging and one is

**Table 2** Latency and sustained bandwidths for the BLC memory system.

| *Attribute* | *L1* | *L2* | *L3 embedded DRAM* | *Scratch SRAM* | *Main memory* |
|---|---|---|---|---|---|
| Size | 32 KiB (I)<br>32 KiB (D)<br>per processor | 2 KiB per processor | 2 banks of 2 MiB/bank = 4 MiB total shared by both processors | 16 KiB shared by both processors | 512 MiB shared by both processors |
| Latency (pclk) | 3 | 11 | 28/36/40 (hit/miss precharged/missed busy) | 15 | 86 (L3 cache enabled) |
| Sustained bandwidth: random quad load access (B/pclk) | NA | NA | 1.8/1.2 (hit/miss) | 2.0 | 0.8/0.5 (single/dual processor) |
| Sustained bandwidth: sequential access (B/pclk) | 16.0 | 5.3 | 5.3/5.3 (hit/miss) | 5.3 | 5.1/3.4 (single/dual processor) |
| Line width (B) | 32 | 128 | 128 | | |
| Number of lines | 1,024 | 16 | 32,768 | | |
| Coherent | No | Yes (weakly) | Yes | Yes (weakly) | Yes |
| Associativity | 64 way | Fully associative | 8 way/bank<br>2 banks | NA | NA |

available for application computation. L1 coherence can be managed in system-level libraries.

- *Virtual node mode:* In this mode, the node is logically separated into two nodes, each of which has a processor and half of the physical memory. Each processor can read and write its own memory area and can read the memory area of the other. This sharing of read-only data can avoid duplication of large application tables. In this mode, the node will run two application processes, one on each processor.

Utilizing the node in a symmetric multiprocessor fashion has not been ruled out, but the noncoherence at the L1 clearly presents some difficulty.

## Designing BG/L for high reliability

For a machine that is to scale to 64Ki nodes, reliability issues are paramount and must be considered at all levels of the system architecture [16].

### System-level RAS

Overall system reliability is achieved, first, through a focus on simplicity. By reducing the number of different components in the system, we have significantly improved reliability while significantly reducing cost. In addition,

- Redundancy is included in many areas of the system design, including the power supplies, which are $N+1$-redundant, and the cooling fans.
- All DRAM is soldered down to avoid failures associated with double inline memory module (DIMM) connectors.
- The cables between racks have spare signals that can be used to avoid isolated faulty wires.
- The clock distribution comprises a single low-jitter 700-MHz clock source that is fanned out, utilizing positive emitter-coupled-logic (PECL) transceivers.
- The positive and negative ends of the differential clock signal are swapped at every layer of the fan-out, reducing any systematic duty cycle distortion.
- The clock network is very simple, and there is no redundancy on its path.

If an individual node on the system fails, the primary RAS strategy is to isolate and replace the failing node while restarting the application from a checkpoint on a set of midplanes that does not contain the faulty node. Specifically, each 512-node midplane is on a separate power boundary, with a separate power domain for the link chips, enabling it to be powered down without affecting any other midplane. Once powered down, the two-way card containing the faulty node can be replaced, and the midplane can be restarted and brought online for the job scheduling software. Thus, a node failure can temporarily bring down a 512-node midplane.

However, powering down a midplane is not always required. For example, the connectively of the collective network has a great deal of built-in redundancy, allowing isolated bad links to be avoided. Some degree of fault tolerance in the torus network can be achieved in software by ensuring that packets are injected in a manner that forces them to avoid failed nodes; this requires non-minimal routing and can handle up to three concurrent failures in a partition provided they are not colinear. This has both software and performance impacts, and therefore is not intended for general application usage. A "bad" node ASIC that still has a viable torus network interface can be left in the network. This provides a good solution for a system that has a high node-failure rate. There are some software restrictions in this mode of operation, but very little performance impact. Additional error-detecting mechanisms that allow us to monitor and isolate faults include the power-supply monitoring and additional link cyclic redundancy checks (CRCs).

The availability of BG/L is achieved through the use of a flexible partitioning scheme and job scheduler along with the previously mentioned reliability features. The link chips allow the entire machine to be partitioned in segments of size 8 along any dimension. This mechanism allows the BG/L machine to be "sliced" up, while also maintaining a torus in each of the user partitions. Midplanes (of size $8 \times 8 \times 8$) need not be physical neighbors within a single user partition, allowing for further partitioning flexibility. One can also reduce the torus requirement and allow for mesh partitions. Because the placement restrictions for a mesh partition are less restrictive than those for a torus, the job scheduler has to be very sophisticated in order to avoid fragmenting the system in a manner in which only mesh partitions are available. Because the effective performance for a torus is greater than twice that of a mesh for some communication patterns, care must be taken to avoid this situation. As with all large supercomputers, when applications are running, checkpoints are regularly written to disk, allowing for the state of the previous "good" checkpoint to be restored upon a machine failure.

The cornerstone of serviceability for the BG/L system is, again, simplicity. We have a very simple, homogeneous packaging structure. Most significantly, compute and I/O nodes can be replaced without removal or movement of any cabling. The removal of a compute node or an I/O node involves first removing the parent node card from the midplane and then removing the appropriate compute or I/O card. The compute and I/O cards plug into the node card in a manner that does not require the movement of any cards other than the appropriate compute or I/O card. The time associated with a compute, I/O, or node card replacement is approximately ten minutes, allowing for very fast service.

### BG/L node reliability

The design of the BLC ASIC warrants special mention. Because it is the basic building block, much of the overall system relies on the node being very robust. The methodology we used to develop the BLC ASIC is closely aligned with that used in high-end servers. Techniques we have utilized include the following:

- All SRAMs are either protected by error checking and correction (ECC) or by a protocol that allows for retry upon a detected error. Both the Ethernet subsystem and the L1 cache arrays in the PowerPC 440 processor fall into the latter category. The vast majority of arrays fall into the former category.
- Network packet transmissions are protected by multiple levels of error detection combined with hardware packet resend, thereby providing guaranteed, reliable network delivery.
- The register arrays in the FPU have been designed utilizing structures with exceptional rejection of soft errors.
- The embedded DRAM utilized as an L3 cache has ECC protection.
- All buses in the ASIC between the functional units are covered by parity.
- There is extensive use of self-checking mechanisms within the different functional modules to detect "illegal" states.
- Latches that are otherwise not protected use a higher power level and are therefore more resilient with respect to soft errors.
- The external DRAM is protected by ECC with the capability to correct any four-bit symbol, detect all double-symbol errors, and detect most other symbol errors.
- DRAM hardware scrub capability allows for the continuous reading and writing back of the entire DRAM memory. This eliminates the likelihood that two random errors will accumulate in the same DRAM read line.
- The external DRAM interface contains an extra four bits (one symbol) of data bus connectivity that can be used to spare out a known bad symbol. This can be done in conjunction with counters that trigger sparing once the number of correctable errors on a given symbol exceeds a threshold.
- Correctible errors of all types are counted and monitored, allowing for extensive predictive error analysis.

These techniques allow for both a low rate of fatal errors and an excellent fault-isolation capability in the case of a failure. In addition, having extensive detection and monitoring of correctable and uncorrectable errors allows for a quantitative analysis of the likelihood of "silent" errors.

### Fault isolation

Fault isolation is one of the most critical aspects of the BG/L computer. A comprehensive approach toward fault isolation has been developed that allows for an extraordinary ability to detect and isolate failed components. The approach we adopted has many layers:

- Extensive data integrity checks allow for immediate isolation for most fails.
- An additional running CRC is maintained at both ends of all node-to-node links of the torus and collective networks, allowing for an independent check of link reliability and data integrity.
- Link chips contain a parity signal that allows for determination of the data integrity of each cable hop, or even multiple cable hops, between neighboring nodes.
- All compute nodes can accumulate checksums in hardware for all traffic injected into the torus and collective networks. These checksums, together with a user-invoked system call, provide a mechanism for finding a faulty node by rerunning a failed application and comparing the checksums. Since the instrumentation causes little performance degradation, it is enabled in all runs, including any failed run. Therefore, fault isolation can be achieved by a single application run following the failed run. In the past, for many machines there was a great deal of difficulty determining the cause of an incorrect run, often resulting in running them for many days of diagnostics to try to reproduce the error after instrumentation had been added.
- The private Fast Ethernet/JTAG control network provides unfettered access through the "back door" of every BG/L compute and I/O node, allowing the entire state of any or all nodes to be dumped without affecting the current state.
- The BLC ASIC and DDR memory of a BG/L node can be synchronously started for cycle-reproducible execution. This provides a very controlled environment for analyzing a fault within a node.
- Prompt system-wide global interrupts allow an entire user partition as large as 64Ki nodes to be halted in less than 1.5 $\mu$s. This avoids the situation in which the system continues to run on and leaves virtually every node in an error state.

Together, these features allow BG/L to achieve fault isolation at virtually arbitrary scale.

## Conclusion

The Blue Gene/L supercomputer was designed to dramatically improve cost/performance for a relatively broad class of applications with good scaling behavior. At a given cost, such applications achieve a dramatic increase in computing power through parallelism. The machine supports the needs of parallel applications, especially in the areas of floating-point, memory, and networking performance.

Compared with other supercomputers, BG/L has a significantly lower cost in terms of power (including cooling), space, and service, while doing no worse in terms of application development cost. Our approach to improving the cost/performance was to utilize an exceptionally high level of integration, following the approach of a number of previous special-purpose machines, such as QCDSP. The high level of integration reduced cost by reducing the overall system size, power, and complexity, and was achieved by leveraging SoC technology. This technology provided two main benefits. First, all of the functionality of a node was contained within a single ASIC chip plus some external commodity DDR memory chips. The functionality includes high-performance memory, networking, and floating-point operations. Second, the PowerPC 440 embedded processor that we used has a dramatically better performance per watt than typical supercomputer processors. Admittedly, a single PowerPC 440 core has relatively moderate performance, but the high level of integration efficiently allows many, many cores to provide high aggregate performance.

The future promises more and more applications using algorithms that allow them to scale to high node counts. This requires nodes to be connected with a sufficiently powerful network. Low-latency communication becomes especially important as the problem size per node decreases and/or as node frequency increases. If such networks can be provided, users will care less about the absolute performance of a node and care more about its cost/performance. Owing to many existing effects, such as complexity, and new effects, such as leakage, the highest absolute node performance is diverging from the best cost/performance. Thus, if sufficiently powerful networks can be provided, scalable applications will be met by supercomputers offering more and more nodes.

An emerging trend in computing is one of focusing on power/performance with respect to the single-node architecture and design point. Blue Gene/L provides a clear example of the benefits of such a power-efficient design approach. Because of future technology scaling limitations, systems similar to Blue Gene/L are likely to become commonplace and are likely to replace the conventional approach toward supercomputing based on power-inefficient, high-performance nodes.

## Acknowledgment

The Blue Gene/L project has been supported and partially funded by the Lawrence Livermore National Laboratory on behalf of the United States Department of Energy under Lawrence Livermore National Laboratory Subcontract No. B517552.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Intel Corporation in the United States, other countries, or both.

## References

1. P. A. Boyle, C. Jung, and T. Wettig, "The QCDOC Supercomputer: Hardware, Software, and Performance," *Proceedings of the Conference for Computing in High Energy and Nuclear Physics (CHEP03)*, 2003; see paper at *http://xxx.lanl.gov/PS_cache/hep-lat/pdf/0306/0306023.pdf*.
2. P. A. Boyle, D. Chen, N. H. Christ, M. A. Clark, S. D. Cohen, C. Cristian, Z. Dong, A. Gara, B. Joó, C. Jung, C. Kim, L. A. Levkova, X. Liao, G. Liu, R. D. Mawhinney, S. Ohta, K. Petrov, T. Wettig, and A. Yamaguchi, "Overview of the QCDSP and QCDOC Computers," *IBM J. Res. & Dev.* **49**, No. 2/3, 351–365 (2005, this issue).
3. "Electrical Energy," *The New Book of Popular Science*, Grolier Incorporated, Danbury, CT, 2000.
4. X. Martorell, N. Smeds, R. Walkup, J. R. Brunheroto, G. Almási, J. A. Gunnels, L. DeRose, J. Labarta, F. Escalé, J. Giménez, H. Servat, and J. E. Moreira, "Blue Gene/L Performance Tools," *IBM J. Res. & Dev.* **49**, No. 2/3, 407–424 (2005, this issue).
5. Message Passing Interface Forum, "MPI: A Message-Passing Interface Standard," University of Tennessee, 1995; see *http://www.mpi-forum.org/docs/mpi-11-html/mpi-report.html*.
6. G. Almási, C. Archer, J. G. Castaños, J. A. Gunnels, C. C. Erway, P. Heidelberger, X. Martorell, J. E. Moreira, K. Pinnow, J. Ratterman, B. D. Steinmacher-Burow, W. Gropp, and B. Toonen, "Design and Implementation of Message-Passing Services for the Blue Gene/L Supercomputer," *IBM J. Res. & Dev.* **49**, No. 2/3, 393–406 (2005, this issue).
7. A. A. Bright, R. A. Haring, M. B. Dombrowa, M. Ohmacht, D. Hoenicke, S. Singh, J. A. Marcella, R. F. Lembach, S. M. Douskey, M. R. Ellavsky, C. G. Zoellin, and A. Gara, "Blue Gene/L Compute Chip: Synthesis, Timing, and Physical Design," *IBM J. Res. & Dev.* **49**, No. 2/3, 277–287 (2005, this issue).
8. M. E. Wazlowski, N. R. Adiga, D. K. Beece, R. Bellofatto, M. A. Blumrich, D. Chen, M. B. Dombrowa, A. Gara, M. E. Giampapa, R. A. Haring, P. Heidelberger, D. Hoenicke, B. J. Nathanson, M. Ohmacht, R. Sharrar, S. Singh, B. D. Steinmacher-Burow, R. B. Tremaine, M. Tsao, A. R. Umamaheshwaran, and P. Vranas, "Verification Strategy for the Blue Gene/L Chip," *IBM J. Res. & Dev.* **49**, No. 2/3, 303–318 (2005, this issue).
9. M. E. Giampapa, R. Bellofatto, M. A. Blumrich, D. Chen, M. B. Dombrowa, A. Gara, R. A. Haring, P. Heidelberger, D. Hoenicke, G. V. Kopcsay, B. J. Nathanson, B. D. Steinmacher-Burow, M. Ohmacht, V. Salapura, and P. Vranas, "Blue Gene/L Advanced Diagnostics Environment," *IBM J. Res. & Dev.* **49**, No. 2/3, 319–331 (2005, this issue).
10. R. S. Germain, Y. Zhestkov, M. Eleftheriou, A. Rayshubskiy, F. Suits, T. J. C. Ward, and B. G. Fitch, "Early Performance Data on the Blue Matter Molecular Simulation Framework," *IBM J. Res. & Dev.* **49**, No. 2/3, 447–455 (2005, this issue).
11. M. Eleftheriou, B. G. Fitch, A. Rayshubskiy, T. J. C. Ward, and R. S. Germain, "Scalable Framework for 3D FFTs on the Blue Gene/L Supercomputer: Implementation and Early Performance Measurements," *IBM J. Res. & Dev.* **49**, No. 2/3, 457–464 (2005, this issue).
12. F. Suits, M. C. Pitman, J. W. Pitera, W. C. Swope, and R. S. Germain, "Overview of Molecular Dynamics Techniques and Early Scientific Results from the Blue Gene Project," *IBM J. Res. & Dev.* **49**, No. 2/3, 475–487 (2005, this issue).
13. G. Almási, S. Chatterjee, A. Gara, J. Gunnels, M. Gupta, A. Henning, J. E. Moreira, B. Walkup, A. Curioni, C. Archer, L. Bachega, B. Chan, B. Curtis, M. Brodowicz, S. Brunett, E. Upchurch, G. Chukkapalli, R. Harkness, and W. Pfeiffer, "Unlocking the Performance of the BlueGene/L Supercomputer," *Proceedings of SC'04*, 2004; see paper at *http://www.sc-conference.org/sc2004/schedule/pdfs/pap220.pdf*.
14. See *http://www-306.ibm.com/chips/techlib/techlib.nsf/techdocs/05D0405273F1C1BD87256D6D0063CFB9*.
15. S. S. Iyer, J. E. Barth, Jr., P. C. Parries, J. P. Norum, J. P. Rice, L. R. Logan, and D. Hoyniak, "Embedded DRAM: Technology Platform for the Blue Gene/L Chip," *IBM J. Res. & Dev.* **49**, No. 2/3, 333–350 (2005, this issue).
16. P. Coteus, H. R. Bickford, T. M. Cipolla, P. G. Crumley, A. Gara, S. A. Hall, G. V. Kopcsay, A. P. Lanzetta, L. S. Mok, R. Rand, R. Swetz, T. Takken, P. La Rocca, C. Marroquin, P. R. Germann, and M. J. Jeanson, "Packaging the Blue Gene/L Supercomputer," *IBM J. Res. & Dev.* **49**, No. 2/3, 213–248 (2005, this issue).
17. Y. Aridor, T. Domany, O. Goldshmidt, J. E. Moreira, and E. Shmueli, "Resource Allocation and Utilization in the Blue Gene/L Supercomputer," *IBM J. Res. & Dev.* **49**, No. 2/3, 425–436 (2005, this issue).
18. M. Ohmacht, R. A. Bergamaschi, S. Bhattacharya, A. Gara, M. E. Giampapa, B. Gopalsamy, R. A. Haring, D. Hoenicke, D. J. Krolak, J. A. Marcella, B. J. Nathanson, V. Salapura, and M. E. Wazlowski, "Blue Gene/L Compute Chip: Memory and Ethernet Subsystem," *IBM J. Res. & Dev.* **49**, No. 2/3, 255–264 (2005, this issue).
19. N. R. Adiga, M. A. Blumrich, D. Chen, P. Coteus, A. Gara, M. E. Giampapa, P. Heidelberger, S. Singh, B. D. Steinmacher-Burow, T. Takken, M. Tsao, and P. Vranas, "Blue Gene/L Torus Interconnection Network," *IBM J. Res. & Dev.* **49**, No. 2/3, 265–276 (2005, this issue).
20. R. A. Haring, R. Bellofatto, A. A. Bright, P. G. Crumley, M. B. Dombrowa, S. M. Douskey, M. R. Ellavsky, B. Gopalsamy, D. Hoenicke, T. A. Liebsch, J. A. Marcella, and M. Ohmacht, "Blue Gene/L Compute Chip: Control, Test, and Bring-Up Infrastructure," *IBM J. Res. & Dev.* **49**, No. 2/3, 289–301 (2005, this issue).
21. G. Bhanot, A. Gara, P. Heidelberger, E. Lawless, J. C. Sexton, and R. Walkup, "Optimizing Task Layout on the Blue Gene/L Supercomputer," *IBM J. Res. & Dev.* **49**, No. 2/3, 489–500 (2005, this issue).
22. J. E. Moreira, G. Almási, C. Archer, R. Bellofatto, P. Bergner, J. R. Brunheroto, M. Brutman, J. G. Castaños, P. G. Crumley, M. Gupta, T. Inglett, D. Lieber, D. Limpert, P. McCarthy, M. Megerian, M. Mendell, M. Mundy, D. Reed, R. K. Sahoo, A. Sanomiya, R. Shok, B. Smith, and G. G. Stewart, "Blue Gene/L Programming and Operating Environment," *IBM J. Res. & Dev.* **49**, No. 2/3, 367–376 (2005, this issue).
23. S. Chatterjee, L. R. Bachega, P. Bergner, K. A. Dockser, J. A. Gunnels, M. Gupta, F. G. Gustavson, C. A. Lapkowski, G. K. Liu, M. Mendell, R. Nair, C. D. Wait, T. J. C. Ward, and P. Wu, "Design and Exploitation of a High-Performance SIMD Floating-Point Unit for Blue Gene/L," *IBM J. Res. & Dev.* **49**, No. 2/3, 377–391 (2005, this issue).
24. C. D. Wait, "IBM PowerPC 440 FPU with Complex-Arithmetic Extensions," *IBM J. Res. & Dev.* **49**, No. 2/3, 249–254 (2005, this issue).
25. J. Lorenz, S. Kral, F. Franchetti, and C. W. Ueberhuber, "Vectorization Techniques for the Blue Gene/L Double FPU," *IBM J. Res. & Dev.* **49**, No. 2/3, 437–446 (2005, this issue).
26. R. F. Enenkel, B. G. Fitch, R. S. Germain, F. G. Gustavson, A. Martin, M. Mendell, J. W. Pitera, M. C. Pitman, A. Rayshubskiy, F. Suits, W. C. Swope, and T. J. C. Ward,

"Custom Math Functions for Molecular Dynamics," *IBM J. Res. & Dev.* **49**, No. 2/3, 465–474 (2005, this issue).

*Received October 29, 2004; accepted for publication December 3, 2004; Internet publication April 7, 2005*

**Alan Gara** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (alangara@us.ibm.com)*. Dr. Gara is a Research Staff Member at the IBM Thomas J. Watson Research Center. He received his Ph.D. degree in physics from the University of Wisconsin at Madison in 1986. In 1998 Dr. Gara received the Gordon Bell Award for the QCDSP supercomputer in the most cost-effective category. He is the chief architect of the Blue Gene/L supercomputer. Dr. Gara also led the design and verification of the Blue Gene/L compute ASIC as well as the bring-up of the Blue Gene/L prototype system.

**Matthias A. Blumrich** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (blumrich@us.ibm.com)*. Dr. Blumrich is a Research Staff Member in the Server Technology Department. He received a B.E.E. degree from the State University of New York at Stony Brook in 1986, and M.A. and Ph.D. degrees in computer science from Princeton University, in 1991 and 1996, respectively. In 1998 he joined the IBM Research Division, where he has worked on scalable networking for servers and the Blue Gene supercomputing project. Dr. Blumrich is an author or coauthor of two patents and 12 technical papers.

**Dong Chen** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (chendong@us.ibm.com)*. Dr. Chen is a Research Staff Member in the Exploratory Server Systems Department. He received his B.S. degree in physics from Peking University in 1990, and M.A., M.Phil., and Ph.D. degrees in theoretical physics from Columbia University, in 1991, 1992, and 1996, respectively. He continued as a postdoctoral researcher at the Massachusetts Institute of Technology from 1996 to 1998. In 1999 he joined the IBM Server Group, where he worked on optimizing applications for IBM RS/6000 SP systems. In 2000 he moved to the IBM Thomas J. Watson Research Center, where he has been working on many areas of the Blue Gene/L supercomputer and collaborating on the QCDOC project. Dr. Chen is an author or coauthor of more than 30 technical journal papers.

**George Liang-Tai Chiu** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (gchiu@us.ibm.com)*. Dr. Chiu received a B.S. degree in physics from the National Taiwan University in 1970, a Ph.D. degree in astrophysics from the University of California at Berkeley in 1978, and an M.S. degree in computer science from Polytechnic University, New York, in 1995. From 1977 to 1980, he was a Research Staff Astronomer with the Astronomy Department at Yale University. Dr. Chiu joined the IBM Research Division at the IBM Thomas J. Watson Research Center as a Research Staff Member. He worked on high-speed testing for Josephson technology from 1980 to 1981, and became a manager of the Josephson Technology Group in 1981. During the years 1983 through 1988, he managed the Silicon Test Systems Group. From 1988 to 1989, he served as the Technical Assistant to Dr. Dean Eastman, IBM Research Vice President of Logic, Memory, and Packaging. From 1989 to 1992, he was the Senior Manager of the Optics and Optoelectronics Group. From 1992 to 1999, he was the Senior Manager of the Packaging and Optics Group. From 1999 to 2000, he was the Senior Manager of the Packaging Technology Group. Since April 2000, he has been the Senior Manager of the Advanced Hardware Server Systems Group in the IBM Systems Department. His research has encompassed VLSI device and internal node characterization at picosecond resolution, laser-beam and electron-beam contactless testing techniques, functional testing of chips and packages, optical lithography, optoelectronic packaging, thin-film transistor liquid crystal displays, optical projection displays, head-mounted displays, and cinema projectors. His current research interests and management responsibility include supercomputer architecture and computer systems packaging. Dr. Chiu has published more than 100 papers, and he has delivered several short courses in the areas mentioned above. He holds 22 U.S. patents. He has received an IBM Outstanding Technical Achievement Award and eight IBM Invention Achievement Awards. Dr. Chiu is a member of the IEEE and the International Astronomical Union.

**Paul Coteus** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (coteus@us.ibm.com).* Dr. Coteus received his Ph.D. degree in physics from Columbia University in 1981. He continued at Columbia to design an electron–proton collider, and spent from 1982 to 1988 as an Assistant Professor of Physics at the University of Colorado at Boulder, studying neutron production of charmed baryons. In 1988, he joined the IBM Thomas J. Watson Research Center as a Research Staff Member. Since 1994 he has managed the Systems Packaging Group, where he directs and designs advanced packaging and tools for high-speed electronics, including I/O circuits, memory system design and standardization of high-speed DRAM, and high-performance system packaging. His most recent work is in the system design and packaging of the Blue Gene/L supercomputer, where he served as packaging leader and program development manager. Dr. Coteus has coauthored numerous papers in the field of electronic packaging; he holds 38 U.S. patents.

**Mark E. Giampapa** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (giampapa@us.ibm.com).* Mr. Giampapa is a Senior Engineer in the Exploratory Server Systems Department. He received a B.A. degree in computer science from Columbia University. He joined the IBM Research Division in 1984 to work in the areas of parallel and distributed processing, and has focused his research on distributed memory and shared memory parallel architectures and operating systems. Mr. Giampapa has received three IBM Outstanding Technical Achievement Awards for his work in distributed processing, simulation, and parallel operating systems. He holds 15 patents, with several more pending, and has published ten papers.

**Ruud A. Haring** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (ruud@us.ibm.com).* Dr. Haring is a Research Staff Member at the IBM Thomas J. Watson Research Center. He received B.S., M.S., and Ph.D. degrees in physics from Leyden University, the Netherlands, in 1977, 1979, and 1984, respectively. Upon joining IBM in 1984, he initially studied surface science aspects of plasma processing. Beginning in 1992, he became involved in electronic circuit design on both microprocessors and application-specific integrated circuits (ASICs). He is currently responsible for the synthesis, physical design, and test aspects of the Blue Gene chip designs. Dr. Haring has received an IBM Outstanding Technical Achievement Award for his contributions to the z900 mainframe, and he holds several patents. His research interests include circuit design and optimization, design for testability, and ASIC design. Dr. Haring is a Senior Member of the IEEE.

**Philip Heidelberger** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (philiph@us.ibm.com).* Dr. Heidelberger received a B.A. degree in mathematics from Oberlin College in 1974 and a Ph.D. degree in operations research from Stanford University in 1978. He has been a Research Staff Member at the IBM Thomas J. Watson Research Center since 1978. His research interests include modeling and analysis of computer performance, probabilistic aspects of discrete event simulations, parallel simulation, and parallel computer architectures. He has authored more than 100 papers in these areas. Dr. Heidelberger has served as Editor-in-Chief of the *ACM Transactions on Modeling and Computer Simulation*. He was the general chairman of the ACM Special Interest Group on Measurement and Evaluation (SIGMETRICS) Performance 2001 Conference, the program co-chairman of the ACM SIGMETRICS Performance 1992 Conference, and the program chairman of the 1989 Winter Simulation Conference. Dr. Heidelberger is currently the vice president of ACM SIGMETRICS; he is a Fellow of the ACM and the IEEE.

**Dirk Hoenicke** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (hoenicke@us.ibm.com).* Mr. Hoenicke received a Dipl. Inform. (M.S.) degree in computer science from the University of Tuebingen, Germany, in 1998. Since then, Mr. Hoenicke has worked on a wide range of aspects of two prevalent processor architectures: ESA/390 and PowerPC. He is currently a member of the Cellular Systems Chip Development Group, where he focuses on the architecture, design, verification, and implementation of the Blue Gene system-on-a-chip (SoC) supercomputer family. In particular, he was responsible for the architecture, design, and verification effort of the collective network and defined and implemented many other parts of the BG/L ASIC. His areas of expertise include high-performance computer systems and advanced memory and network architectures, as well as power-, area-, and complexity-efficient logic designs.

**Gerard V. Kopcsay** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (kopcsay@us.ibm.com).* Mr. Kopcsay is a Research Staff Member. He received a B.E. degree in electrical engineering from Manhattan College in 1969, and an M.S. degree in electrical engineering from the Polytechnic Institute of Brooklyn in 1974. From 1969 to 1978, he was with the AIL Division of the Eaton Corporation, where he worked on the design and development of low-noise microwave receivers. He joined the IBM Thomas J. Watson Research Center in 1978. Mr. Kopcsay has worked on the design, analysis, and measurement of interconnection technologies used in computer packages at IBM. His research interests include the measurement and simulation of multi-Gb/s interconnects, high-performance computer design, and applications of short-pulse phenomena. He is currently working on the design and implementation of the Blue Gene/L supercomputer. Mr. Kopcsay is a member of the American Physical Society.

**Thomas A. Liebsch** *IBM Engineering and Technology Services, 3605 Highway 52 N., Rochester, Minnesota 55901 (liebsch@us.ibm.com).* Mr. Liebsch has worked for IBM as an electrical engineer and programmer since 1988. He received B.S. and M.S. degrees in electrical engineering from South Dakota State University in 1985 and 1987, respectively, along with minors in mathematics and computer science. He has had numerous responsibilities involving IBM server power-on control software design, built-in self-test designs, clocking designs, and various core microprocessor logic design responsibilities. He was the system technical owner for several IBM iSeries* and pSeries* servers. Mr. Liebsch is currently the chief engineer working on the Blue Gene/L system.

**Martin Ohmacht** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (mohmacht@us.ibm.com)*. Dr. Ohmacht received his Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from the University of Hannover, Germany, in 1994 and 2001, respectively. He joined the IBM Research Division in 2001 and has worked on memory subsystem architecture and implementation for the Blue Gene project. His research interests include computer architecture, design and verification of multiprocessor systems, and compiler optimizations.

**Burkhard D. Steinmacher-Burow** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (steinmac@us.ibm.com)*. Dr. Steinmacher-Burow is a Research Staff Member in the Exploratory Server Systems Department. He received a B.S. degree in physics from the University of Waterloo in 1988, and M.S. and Ph.D. degrees from the University of Toronto, in 1990 and 1994, respectively. He subsequently joined the Universitaet Hamburg and then the Deutsches Elektronen-Synchrotron to work in experimental particle physics. In 2001, he joined IBM at the Thomas J. Watson Research Center and has since worked in many hardware and software areas of the Blue Gene research program. Dr. Steinmacher-Burow is an author or coauthor of more than 80 technical papers.

**Todd Takken** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (taken@us.ibm.com)*. Dr. Takken is a Research Staff Member at the IBM Thomas J. Watson Research Center. He received a B.A. degree from the University of Virginia and an M.A. degree from Middlebury College; he finished his Ph.D. degree in electrical engineering at Stanford University in 1997. He then joined the IBM Research Division, where he has worked in the areas of signal integrity analysis, decoupling and power system design, microelectronic packaging, parallel system architecture, packet routing, and network design. Dr. Takken holds more than a dozen U.S. patents.

**Pavlos Vranas** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (vranasp@us.ibm.com)*. Dr. Vranas is a Research Staff Member in the Deep Computing Systems Department at the IBM Thomas J. Watson Research Center. He received his B.S. degree in physics from the University of Athens in 1985, and his M.S. and Ph.D. degrees in theoretical physics from the University of California at Davis in 1987 and 1990, respectively. He continued research in theoretical physics as a postdoctoral researcher at the Supercomputer Computations Research Institute, Florida State University (1990–1994), at Columbia University (1994–1998), and at the University of Illinois at Urbana–Champaign (1998–2000). In 2000 he joined IBM at the Thomas J. Watson Research Center, where he has worked on the architecture, design, verification, and bring-up of the Blue Gene/L supercomputer and is continuing his research in theoretical physics. Dr. Vranas is an author or coauthor of 59 papers in supercomputing and theoretical physics.