Reconfigurable Computing Cluster (RCC) Project: Investigating the Feasibility of FPGA-Based Petascale Computing *

Ron Sass, William V. Kritikos, Andrew Schmidt, Srinivas Beeravolu, Parag Beeraka,
Kushal Datta, David Andrews, Richard Miller, and Daniel Stanzione, Jr.
Reconfigurable Computing Systems Lab†
University of North Carolina at Charlotte
9201 University City Blvd. / Charlotte, NC 28223–0001
{rsass,kdatta}@uncc.edu, {willk,aschmidt,dandrews}@ittc.ku.edu,
Srinivas.beeravolu@xilinx.com, beeraka@gmail.com, rm@ces.clemson.edu, dstanzi@asu.edu

Abstract

While medium- and large-sized computing centers have increasingly relied on clusters of commodity PC hardware to provide cost-effective capacity and capability, it is not clear that this technology will scale to the PetaFLOP range. It is expected that semiconductor technology will continue its exponential advancements over next fifteen years; however, new issues are rapidly emerging and the relative importance of current performance metrics are shifting. Future PetaFLOP architectures will require system designers to solve computer architecture problems ranging from how to house, power, and cool the machine, all-the-while remaining sensitive to cost.

The Reconfigurable Computing Cluster (RCC) project is a multi-institution, multi-disciplinary project investigating the use of FPGAs to build cost-effective petascale computers. This paper describes the nascent project's objectives and a 64-node prototype cluster. Specifically, the aim is to provide an detailed motivation for the project, describe the design principles guiding development, and present a preliminary performance assessment. Several core, pragmatic questions are enumerated and micro-benchmark results are reported. These results characterize key subsystems — including the system software, network performance, memory bandwidth, power consumption of nodes in the cluster. Results suggest that the approach is sound.

*This project was supported in part by the National Science Foundation under NSF Grants CNS 06-52468 (EHS) and CNS 04-10790 (CRI). The opinions expressed are those of the authors and not necessarily those of the Foundation.

†UNC-Charlotte is the lead institution. Will Kritikos, Andrew Schmidt, and David Andrews are at the University of Kansas, 2335 Irving Hill Road, Lawrence, KS 66045–7523; Srinivas Beeravolu is at Xilinx, Inc., 2100 Logic Drive , San Jose, CA 95124-3400; Parag Beeraka is at Qualcomm, Inc. 5775 Morehouse Drive, San Diego, CA 92121; Richard Miller is at Clemson University, Department of Mechanical Engineering, 210 Engineering Innovation Building, Clemson University, Clemson, SC 29634-0921; Daniel Stanzione is at Arizona State University, High Performance Computing Initiative, GWC 182, Box 875206, Tempe, AZ 85287-5206

1. Introduction

During the Age of Discovery, experimentalists were limited by the power of their instruments. Improvements in microscopic and telescopic resolution led to scientific discoveries that no one at the time could have imagined. For computational scientists today, *the computer is their instrument*. From computational chemists investigating biomolecular reactions [33] to ecologists simulating complex, multi-species ecosystems, to computational biologists predicting protein folds [22] to physicists studying Computational Fluid Dynamics (CFD) [14] — the performance of state-of-the-art high-end computers limits the resolution of experiments which in turn dictates the scope of scientific endeavors. Many computational scientists have suggested that access to cost-effective petascale computing would enable science that is simply infeasible today.

The last two decades have seen the emergence of Beowulf-class computers (clusters of commodity off-the-shelf hardware and Open Source software) which — in combination with tremendous gains in single-chip processors — has had a profound effect on High-Performance Computing (HPC). Over half of the fastest 500 computers on the TOP500 list identify themselves as "clusters." And while many of the fastest clusters use special-purpose networking designed for HPC, it is interesting to note that the most common networking technology on the list is Ethernet [34]. Even in an arena where speed is king, the marketplace still reflects a sensitivity to cost.

Despite the exponential growth of microprocessors, it is not clear that current technology will scale to a PetaFLOP/s. Issues beyond simply raw computational speed — such as power, cooling, and physical infrastructure — will become increasingly important. Just as important are other issues that arise from the performance of key subsystems, such as the bandwidth to primary storage, secondary storage, and networking.

1.1. Motivation

Consider three computational science problem domains: Computational Fluid Dynamics (CFD), Molecular Dynamics (MD), and Bioinformatics (sequence searching). Scientists working in these domains are interested in solving ever-larger problems. Increasing the resolution of computer simulations, reveal phenomena that are otherwise unobservable and insight into processes that are otherwise impractical to experiment with in the physical world.

CFD scientists want to increase the resolution of their experiments which will allow them to observe phenomenon and make better models. However, simply doubling the resolution increases the computation 16-fold. The increase is due in part to the fact that there is 2^3 more data that needs to be processed and that smaller time-steps (needed for mathematically stability) are required.

In terms of MD, we simply do not have the computing power to simulate the behavior of 100s of thousands of atoms for the length of time needed to understand the mechanical processes that happen. In both of these applications, not only does the number of floating-point operations increase but so does the memory requirements. Hence the need for additional memory bandwidth. Simply adding function units without associated improvements to the memory bandwidth will be fruitless.

For scientists interested in uncovering the function of genes or proteins, it is uphill battle. Even though Moore's Law has processor performance doubling every 18 months (a compound annual growth rate of 59%), biological databases are growing even faster. Figure 1 shows the growth rates of several key indicators since 1994 on a semi-log graph. The data points come from GenBank [18], a public collection sequenced genomes. A line fitted to this data shows a compound annual growth rate of 77%. Also important to note is the growth rate of I/O subsystem (disk and interface). The most aggressive estimates [1] suggest a 10% compound annual growth rate in performance while others [10] suggest a more modest 6% growth rate. Regardless, these trends have an important consequence: the same question (e.g., is this gene similar to any known gene?) will take longer every year. In other words, the problem size is growing faster than single processor performance and much faster than the bandwidth of I/O subsystems.

Similar arguments for industry users of high-performance computing: from animation, to finance, to product development [13]: these users not only need more computation but also more bandwidth to primary and secondary storage. If more bandwidth is not possible, then it is critical to fully utilize data on-chip, use the most efficient transfer mechanisms, and minimize the number of transfers.

While applications needs are trending towards large data sets and more computations, semiconductor technology trends are changing as well. There are two ways one might leverage Moore's Law: (1) try to scale with technology by buying more cores per chip and (2) scale the number of nodes with many slower, less expensive chips. Unfortunately, not all of the consequences of Moore's Law benefit computational scientists and each of these approaches face challenges that have not been an issue in the past.

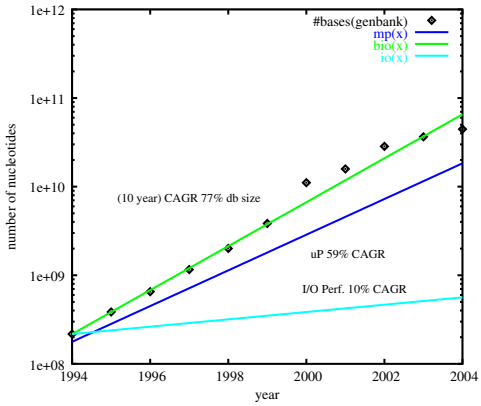

Figure 1. compound annual growth rate of problem size, single processor performance, and I/O subsystem performance

First, consider multi-core solutions. While computation rates and memory capacity are growing exponentially, the speed of DRAM is growing much slower (doubling every 48 months versus every 18 to 24 months). Moreover, the bandwidth between off-chip memory and computation units is relatively flat. This so called "Memory Wall" was predicted for general-purpose computing [37], but general-purpose processors have been able to delay this issue by compensating for the growing disparity with large on-chip caches. In many ways, FPGAs — which almost universally lack the sort of caches needed — arrived at this problem earlier, and (to novices) many application problems appear I/O bound. Nonetheless, there are limits to how long any solution can postpone the issue. A petascale machine built on the premise of 100s of cores on a multi-core processor will definitely have to address this issue.

The alternative is to buy more nodes. This introduces problems of sheer size, cost, and power. It is also has subtle cost implications. Consider a modern circa 2006 commodity cluster. For about $200,000, one can buy 64 dual core, dual processor nodes in two racks and InfiniBand networking. Each node will have a 500W power supply and the cluster as a whole (assuming 80% of theoretical peak performance) will deliver about 820 GFLOPS. To scale this system to a PetaFLOP, would require about 1220 of these 64-node clusters. That is roughly 1950 racks (packing 40

nodes into a 42U rack) which is an enormous footprint in a machine room! Moreover, the electrical system would have to support the delivery of 40 MW of power to the system and even more energy to cool the machine that is converting 40 MW into heat. And, of course, there is a huge question of what interconnect would allow 78,000+ nodes to efficiently communicate.

However, there is a subtle problem with simply scaling a cluster in terms of the number of nodes. A corollary of Moore's Law is that the cost/transistor is cut in half every technology cycle. Even though the transistors cost less and less, the rest of the components in the node do not follow the same trend. Many of the components are commodity items and their cost may rise or fall based on the market. Assuming that commodity prices are stable and that the budget is fixed, a consequence of scaling the number of nodes (with less expensive processors) is that cases, cables, components, etc. begin to take a disproportionate amount of the budget. It is the Law of Diminishing applied to decreasing node cost.

In summary, one can take advantages of gains in semiconductor technology to grow the number of cores on a single chip or grow the number of individual processors (albeit slower and cheaper) to scale to a PetaFLOP. However, the former will have to address the memory bandwidth issue and the latter has to forgo commodity packaging and address physical scaling issues. And, of course, neither directly addresses the secondary storage issue. Faced with growing needs of computational science and these technology trends, the fundamental question the project is investigating is: how build cost-effective petascale computers?

1.2. Overview

The hypothesis of the RCC project is that the answer to leverage FPGA technology. To test the premise, the project is considering a number of performance metrics — that incorporate energy, cost, physical size, as well as rate-of-computation and capital investment — to evaluate a proposed architecture. These metrics, described in subsection 2.1, are not new but not universal either. Moreover, the relative importance these metrics are changing. Next we present our reasoning for FPGAs in subsection 2.2. A straw man FPGA-cluster design based on these metrics is described in subsection 2.3. To test some of these ideas, a prototype cluster is being built at the University of North Carolina, Charlotte. section 3 describes this design and present we present raw performance data a number of key subsystems. The paper concludes with related work (section 4) and a summary in section 5.

2. Design of Petascale FPGA-Based Cluster

2.1. Performance Metrics

Rate-of-computation (speed) and cost have been the dominate metrics for the last two decades. Indeed, it has appeared that some system designers have been willing trade any amount of power to make an incremental improvement in speed. However, as [16] and others have clearly argued, power is now a first-class design metric. And, as argued in the introduction, petascale computing is not simply a matter of scaling up the number of computation units. To make rational, engineering-based decisions about a scalable design, we propose a collection of metrics to relate the many facets of system performance. In other words, in the past a system designer simply chose the fastest processor available (or, perhaps, the fastest processor the project could afford). In petascale design, the best design may actually use very modest processors. (Consider the design of BlueGene/L [20] and other proposals, such as [17].)

Implicitly, this study assumes that a petascale system will be a collection of parallel nodes where the number of nodes needed to reach a PetaFLOP, n, is different for different designs. Thus we focus on ratios that are independent of n. The first five metrics relate speed to various other performance metrics or cost. The last metric deals with networking bandwidth. All of the ratios are written so that higher is better.

For most medium- to large-sized compute centers, there is fixed, practical limit to how much current the equipment can draw. Moreover, the power used to drive a computer system is converted into heat which subsequently raises the ambient temperature and high temperatures causes equipment failures. If active cooling is needed to remove the heat, then that adds to the power required by the design. Furthermore, the GFLOPS/W of a design not only expresses the peak current required but also characterizes part of operating costs (in terms of an electricity bill).

Another important metric relates cost and speed. The GFLOPS/$ ratio of a design is the capitol investment cost required to deploy a node. Over a range of n, this ratio is monotonic but not necessarily continuous. (For example, adding one more node might require another switch or spill over to another rack.) Nonetheless, a GFLOPS/$ function will allow users to determine the performance available for fixed budget or, as our goal here is, find the estimated cost of a petascale system.

Another increasingly important metric is related to space. For many institutions, space comes at a premium and has to be factored into the cost. As suggested in the introduction, it is not feasible to simply scale current (Fall 2006) clusters to a PetaFLOP because of the number of racks needed. For this and other reason, GFLOPS/m^3 is important. However, as long as systems are designed around 19"

racks, the metric can usually be simplified to its "footprint" which ignores height and focuses on the square footage on a machine room floor.

Three- versus two-dimensional space aside, there is a more subtle issue that relates to volume. Namely, the performance of many interconnection networks are directly proportional to the physical distance of the longest cable. Standard networks are essentially a compromise between a maximum cable length, how much shielding to use on the conductors and receptacles, and maximum transmission rate. Some petascale designs that have large footprints (or low GFLOPS/m^3 ratios) will approach the maximum cable lengths. Solutions to such problems will either increase network costs, distort performance, or both as the system is scaled. It is important to note that keeping a petascale computer within a critical volume enables one to use a number of low-cost, short distance communication technologies.

The so-called "Memory Wall" mentioned in the introduction presents an especially important challenge. No matter what the solution — multi-core, low-power processor, or FPGA-based — one cannot escape the bandwidth limits imposed by package pin counts. Barring any unexpected breakthrough, the approach which uses that bandwidth most efficiently will prevail. Current processors use a metric of mega-transfers per second, MT/s, to rate DIMM performance. (This abstracts away clock rates and the fact that some technology uses double or quad transfers per clock period.) Thus, for reasons we explain in the subsections, the metric (MT/s)/GFLOPS will be as important — if not more important — to an FPGA design than the raw floating-point performance of a design. This is because trends suggest that, ultimately, this metric will put an upper limit on the rate at which results can be produced. However, using MT/s/GFLOPS alone provides an imperfect picture. Modern processors have extensive caching structures — in some processors, most of the transistors are committed to caches. This can have the effect of amplifying effective memory bandwidth by avoiding off-chip communication. Likewise, FPGA designers routinely build various custom buffers to memory subsystems to accomplish the same. Hence, a related metric, effBW/GFLOPS — the effective bandwidth after any caching — may turn out to be an important metric when comparing two competing designs.

A final metric, intended to characterize the interconnection network, measures the achievable network bandwidth per node, NetBW/node, is intended to characterize both the network interface as well as as switch capabilities. This metric may not capture enough salient details to properly differentiate to FPGA cluster designs. Nonetheless, it is a starting point for the project. There are a number of questions we aim to answer with our prototype related to networking. Results of those investigations may indicate that other metrics — such as message latency may need to be incorporated.

The central goal of the Reconfigurable Computing Cluster project is not to select the best components in every category (i.e., performance at any price) but rather to select components that balance these metrics to achieve a cost-effective, practical petascale system. Specifically, we are targeting a machine that will consume 250 kW of power, costs about $\$10$M, has a 25×25 ft^2 footprint and is within a factor-of-2 of a PetaFLOP/second performance (0.5 PetaFLOPS).

2.2. FPGA Characteristics

The application and technology trends described in the introduction will present challenges to any HPC system designer. Given the need to solve larger problems, the challenge is how to address the technology trends.

It appears likely that we will achieve petascale computing in the near future; however, a more important question is how to make it cost-effective. One approach is to build 100s of thousands (or millions) of energy efficient SoC processors. Another is to build clusters with fewer nodes out of primarily commodity parts using the fastest multi-core chips available. The solution being investigated by the RCC project is to use clusters of FPGAs. This approach is a compromise between the two extremes: a balance between the computational power of a node and the number of nodes.

Although not in context of clusters, DeHon argued the computational density advantage of FPGAs in [8]. This sets up one fundamental compromise that has to be resolved by the system designer: An FPGA draws more power than a single chip SoC but has the ability of delivering a faster rate-of-computation. A critical question, that can only be answered by implement-and-test, is to determine the GFLOPS/W of a FPGA. Our premise is that through parallelism, specialization, and a synergy between the two — plus application-specific customizations — will give FPGA-based solutions a significant advantage in this metric. Also, we anticipate that there are significant potential gains to be made from the tight coupling of the network and computational components. With an FPGA, it is possible to continue to explore application-specific and general-purpose innovations — including collective communications and evolving programming models. Designs fixed at manufacture will not foster those innovations. Moreover, with a higher rate-of-computation, an FPGA-cluster will reach the petascale with fewer nodes. This has important systemic implications in that we are only just beginning to see the subtle problems that arise when coordinating 100,000s of independent processes [23]. The sensitivity of of 1,000,000s of independent processes to slight perturbations is yet to be seen.

By increasing the speed of the nodes, FPGAs exacerbate the memory bandwidth problem. As Underwood and Hem-

mert have illustrated, with FPGAs, many important operations are not compute-bound but limited by memory performance [29]. In this case, the question is whether FPGAs will perform better than multi-core high-end processors. Our premise is that FPGAs are better suited to accommodating more memory channels and an application-specific approach will be better suited to use the bandwidth available. In other words, given that neither approach can change the pin bandwidth, it becomes a question of which will be more efficient and we believe FPGAs have an advantage here.

A final aspect is cost: this is an especially difficult measure to predict since a state-of-the-art FPGA is typically orders-of-magnitude more expensive than a state-of-the-art commodity mainboards while value-priced FPGAs are available long after commodity mainboards are end-of-lifed. So an FPGA used in the prototype here might be $85 but, ultimately, the one found to offer the best characteristics is $8,500. With less confidence, we believe a balance between cost and performance can be struck.

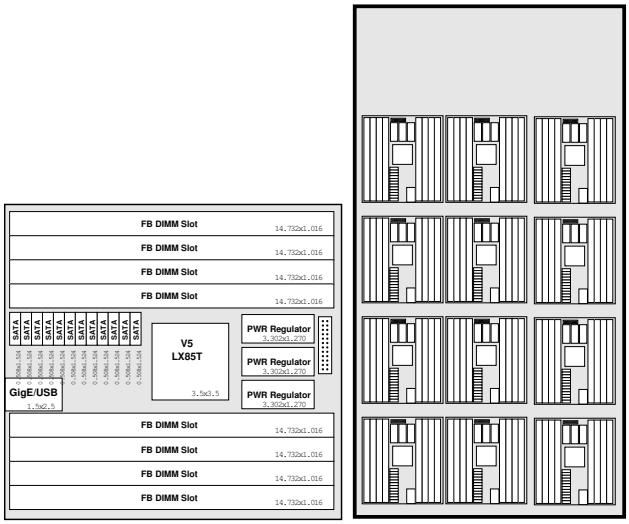

Figure 2. (a) straw man proposal of one FPGA node (b) twelve FPGA nodes in a 1U chassis

2.3. Straw Man Design

What would a petascale FPGA-cluster look like? There are too many outstanding issues to answer this question definitively. However, it is useful to have a starting point and a straw man design gives us a set of specific decisions to question and framework to test alternatives.

The emphasis on cost-effectiveness would suggest we use commodity components. However, it seems unlikely that a commodity node will meet the size, power, and other metrics discussed. The major components of custom-made node — drawn to scale and with appropriate spacing — are shown in Figure 2(a). Although more expensive, the size advantage gives us several system density advantages. One important is the ability to leverage high-speed, low-cost SATA cables. (In this design, we assume a direct connect network. However, there is a lively discussion on how best to handle networking and, in particular, whether the switch should be integrated into the FPGA.)

Twelve of these 15.25×14.25 cm^2 nodes will fit in a standard 19-inch deep rack mount chassis as shown in Figure 2(b). The configuration shown leaves room for two 300W, 5V power supplies in the rear. (Other required voltages come from regulators on each node.)

With 32 1U chassis in a rack, 36 racks can be arranged in three rows of 12 and will fit into a $24'$ by $15'$ footprint. This organization would allow for a 3-D torus to be implemented with approximately 864 1-meter cables and (a lot of) standard SATA cables.

As any good straw man design should, this proposal raises many more questions than it answers, such as the effective memory bandwidth that can be expected, the GFLOPS/W that will actually be achieved, and the NetBW for realistic cable lengths is possible. The prototype being built at UNC Charlotte (and described next) is designed to help answer some of the questions.

3. RCC Prototype and Measurements

Complex systems defy purely analytical evaluations and the straw man proposed in the previous section is no exception. In addition to the large number of subsystems that — when combined — may exhibit unexpected behavior, there are subtle synchronization issues when scaling loosely coupled, independent systems by orders-of-magnitude. Similarly, real-world concerns — such as EMI generated by the system — can change the bit-error rates of communication channels and have a significant impact system performance. These effects are nearly impossible to model on paper or in a discrete event simulator. Consequently, our approach is to construct a two-rack, scale model and take direct measurements.

3.1. Prototype Cluster

A block diagram of the 64-node cluster under construction at the University of North Carolina, Charlotte is illustrated in Figure 3. FPGA-based compute nodes, a novel data network, and two 48-port Ethernet network switches comprise the core components of the cluster. A Head node has a high-speed disk I/O subsystem and also functions as a router, connecting the private subnet of the cluster to the campus network (and the Internet).

The Xilinx ML-310 [39] is familiar ATX form factor development board. It has PCI slots, IDE controllers, the stan-

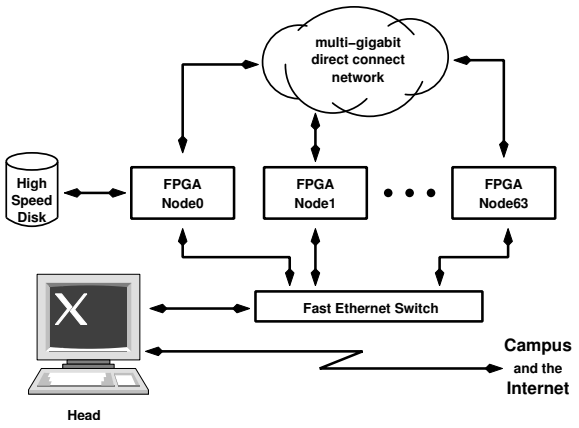

Figure 3. block diagram of proposed cluster

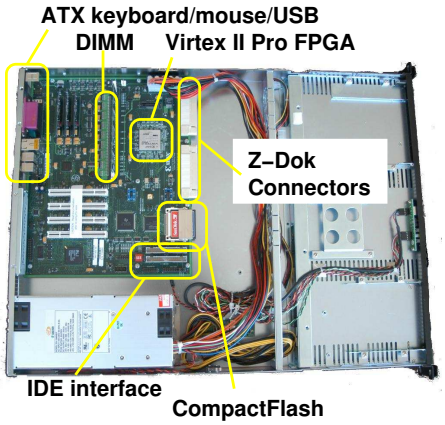

Figure 4. one ML-310 node

dard keyboard/mouse/USB connections, and — instead of a microprocessor — a Xilinx Virtex II-Pro (V2P30) FPGA. Figure 4 shows an ML-310 board installed in a typical 19-inch 1U chassis. The features relevant to the proposed projects are highlighted.

A logical representation of a typical system architecture, called a Base System Platform or BSP, is shown in Figure 5(a). The solid lines represent cores that are implemented in silicon and the dashed lines represent cores implemented in CLBs. The base system platform is synthesized into a bitstream using vendor-provided tools: Xilinx Platform Studio (XPS)/Embedded Development Tool Kit (EDK) and Integrated Software Environment (ISE).

A novel aspect of the proposed cluster its use of the eight 3.125 Gb/s bidirectional Multi-Gigabit Transceivers (MGTs a/k/a RocketIOs). In the proposed cluster, these communication channels directly connect the FPGAs. By integrating the network switching onto the FPGA, the cluster not only eliminates the need for an external switch (lowering costs) but also enables the tight coupling of key high-performance computing services. A logical representation of a potential BSP designed for MPI message-passing programs is shown in Figure 5(b). (Note: not all hardware accelerators need a network connection. For example, one core that computes e^{-x} is simply a hardware-assist for a local processor.)

Using Aurora protocol cores, a network switch core, and a network layer core, much of the data communication protocol processing can be handled in hardware rather than software. Aurora is a freely-available link-layer protocol that adds services, such as channel bonding, and greatly simplifies the interface to the multi-gigabit transceivers (see [38]). The network layer core was developed as part of the NSF-sponsored Adaptable Computing Cluster project (see [11] and [12] for details of the configurable network layer protocol). It adds reliable packet and stream-oriented services.

Several on-chip network switches options are available. One, developed by a colleague at ASU, is a candidate router for this project. Details of this router can be found in [35] and [3]. [19] has also developed a complex router for ATM-like communications. However, it requires a large amount of configurable logic and only a heavily stripped down version would be a likely candidate for HPC communications. A third option under consideration is to use an external network switch.

Access to the high-speed network transceivers presents a technical challenge. High-speed signaling is very sensitive to noise and distance, so the cabling, and cable-to-FPGA interface must be carefully designed. The serial transceivers are bidirectional and use Low-Voltage Differential Signaling (LVDS). Seven conductors (two differential signal pairs and three drains) are needed. At the physical layer (and in terms of performance) the transceivers are very similar to InfiniBand, PCI-Express, and other high-speed serial transceivers. However, InfiniBand cables, for example, are very expensive and Serial ATA (SATA) cables provide the same data rates for distances for short distances. So a benefit of keeping the nodes physically close, is the ability to use the lower-cost SATA cables.[1]

Both of these challenges are met with a custom 4-layer printed circuit board that routes the eight MGT signals through a Z-Dok connector on the ML-310 board to eight SATA receptacles. Two version of the board has been developed. The first, shown in Figure 6, revealed some design flaws in testing. The second revision resolved all of the known issues and greatly improved the network performance.

The physical components reside in two 19-inch racks, each equipped with a switched, metered power delivery unit (PDU). The PDUs will enable us to (a) measure the power used to run an application and (b) individually power nodes.

[1]The cost difference between SATA and IB cables for this 64-node cluster would have been around $16,000!

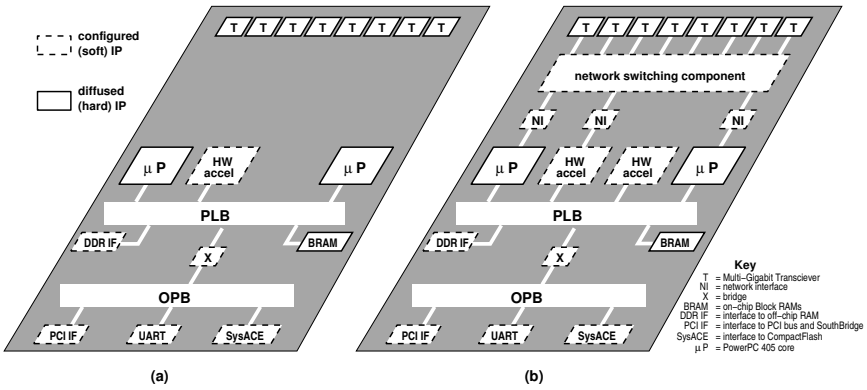

Figure 5. (a) typical base system platform (b) network-enhanced base system platform

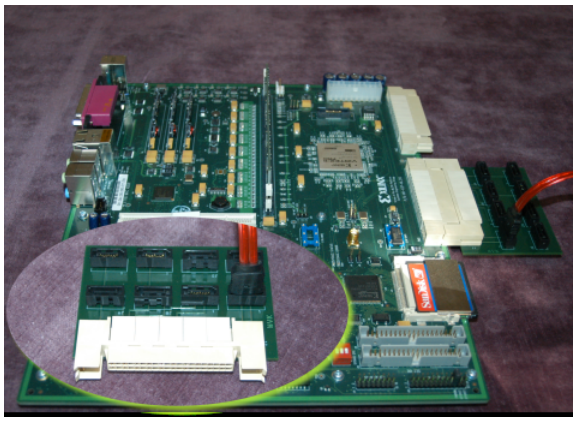

Figure 6. ML-310 with installed with our custom multi-gigabit transceivers to SATA adapter board (inset)

3.2. Investigations

3.2.1. Memory Performance

The bus structures commonly used in Platform FPGAs (such as the ones used in the prototype) are generally designed for Embedded Systems. But, like most buses, they provide ease-of-use and general connectivity at the expense of bandwidth. However, as identified earlier, memory bandwidth will be crucial for designs that have a GFLOPS per node metric. Hence, we have investigated several memory structures to evaluate: (1) which on-chip organizations provide the highest bandwidth and (2) test the feasibility of using existing bus structures in the petascale design.

Towards this end, a specialized core was developed that generates three common HPC memory access patterns (sequential, strided, and random). For sequential access, there is also the option of using burst mode transactions (for the bus combinations that support it).

The off-chip memory controller interacts with SDRAMs on a DDR DIMM package on the ML-310. There are memory controllers for both the OPB and PLB which gives us four combinations (HPC core on the OPB/memory controller on the OPB, HPC core on the OPB/memory controller on the PLB, and so on).

Theoretically, the bandwidth between the memory controller and the DIMMs is just below 2700 MB/s. As one might expect, the bus structures will degrade the effective bandwidth. However, our studies revealed that all combinations of memory controllers and bus HPC cores resulted in a small fraction of the theoretical speeds. Moreover, the bus protocol overhead was so large that it renders moot any attempt to intelligently access the SDRAMs based on their low-level layouts. Complete details on the fourteen combinations studied and how the experiments were conducted can be found in [26].

Consequently, we are investigating multi-ported memory controllers that are more likely to provided the desired bandwidth for HPC applications. The idea is to connect one port to a bus so that slower, general-purpose cores still have the convenience of a bus while carefully-designed HPC cores can move significant amounts of data to and from off-chip memory.

3.2.2. Power

Estimating power consumption is challenging. The power consumption of an FPGA design is largely determined by how often transistors switch which is function of the clock frequency, the design, and the run-time data. Also, for small circuits, FPGA designs exhibit discontinuities (for example, adding a gate does not necessarily increase the number of slices used). Thus, in order to get an accurate estimate of the energy needed per floating-point operation, one has to exercise several designs incorporating multiple floating-point units — especially as the device reaches its capacity. The following experiment was designed to do this.

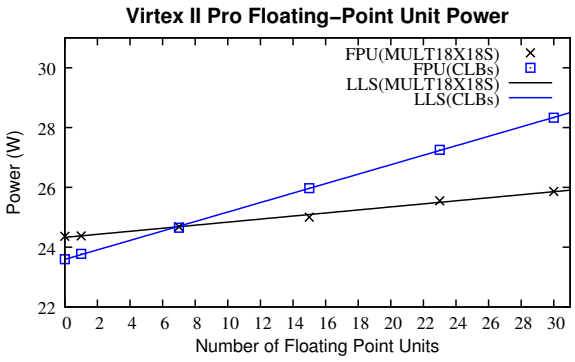

Figure 7. power consumed by implementing floating-point units with multiplier blocks and CLBs; points are measured, lines are fitted linear-least squares

First, a power meter (measuring total power into a node: FPGA, RAM, all components) was connected. An idle Linux system running draws about 24.74 Watts. Next, a number of designs involving 1, 7, 15, 23, and 30 IEEE 754 single-precision compatible Floating-Point Unit (FPU) were generated. In these experiments, two different FPUs were used: one uses block multipliers (MULT18X18S) present on the Virtex II Pro and the other was implemented exclusively using CLBs. Both are six-stage 100-MHz pipelined cores. As mentioned, an idle unit (or one that reuses the same inputs) is likely to give an unrealistic estimate of power. So two pseudo-random number generators were used to create a large set of ever-changing data to drive the multipliers. (However, it is worth noting that scientific applications exhibit a high-degree of "value locality" (0 and 1, for example, are very common inputs), so our estimates are conservative. Real applications may use less power.)

We were pleasantly surprised to see a nearly linear increase in power for both types of floating-point units and, using Linear Least Squares, we were able to fit two lines to the data as shown in Figure 7. From the slopes of these lines, we can approximate that a fully utilized (MULT18X18S) FPU will draw about 50.9mW/FPU. At 100 MHz, this is about 509 picojoules/floating-point operation. The all-CLB solution works out to be 158 mW on average; however, it appears that for sparsely packed designs the CLB-solution can use less power.

What does this mean in terms of the power feasibility of using FPGAs? First, the resources used to instantiate 30 FPUs is not practical but if one did and could effectively keep them busy, this would achieve about 3 GFLOPS per FPGA board. Unfortunately, this would require something like 333,334 boards to reach a petascale cluster and 8.44 MW to operate. Both of these are impractical and far from our targets. However, if one looks at just the floating-point units, then 509 pJ/operation translates into roughly 500 kW to power just the floating-point units needed to reach a PetaFLOP. While this number is still high, it is not unreasonable especially since our experimental FPU system designs are quite simple and no attempts to optimize for power have been made. Also, as we mentioned, using pseudo-random number generator for inputs is probably overly conservative.

Even though the Virtex II Pro P30 is not going to deliver the characteristics needed for a petascale cluster, these experiments do bolster one of our design principles: Namely, that a significant amount of power goes to components that are present just to make system run and do not directly contribute to the node's rate-of-computation. So, fewer nodes, with more floating-point units per device is likely to improve power efficiency of the system.

One final note regarding the feasibility of our approach. As a power density comparison, consider the GFLOPS/W of the prototype cluster with ML-310 boards versus the IBM BlueGene/L. Our quick-and-dirty design is about 0.2 GFLOPS/W while BlueGene/L reports 0.228 GFLOPS/W [7]. This suggests that (a) cooling a petascale FPGA cluster will be feasible and (b) that it is possible to use design specialization and custom architectures to compensate for the inherent power disadvantages of FPGAs in place of an ASIC.

3.2.3. Chip-to-Chip Networking

The latency of the link is defined as the amount of time from when the "start of frame" signal is asserted by the transmitter to when the "start of frame" signal is then seen by the receiver. This latency can be split into five components: the latency of the aurora transmitting protocol engine, the latency of the transmitting transceiver, the propagation delay of the cable, the latency of the receiving transceiver, and the latency of the receiving aurora protocol engine.

The values for the five latency components shown in Table 1 and all of the data, except for the propagation delay, come from Appendix A of [40]. The aurora core used in the tests was a 2 byte, single lane design, meaning that the aurora core takes in 2 bytes in parallel each user clock cycle, and only 1 transceiver is used, ie, there were no bonded channels. The value for the propagation delay of the Serial ATA link was taken from the Amphenol SpectraStrip data sheet, which is the cluster's current candidate cable. The clock frequency used for these experiments was 156.25 MHz.

The efficiency of a link is defined as the ratio of correct user data sent to all data sent over the link. To easily find the effective bandwidth of the link, the overall efficiency of that link is multiplied by the raw data rate of the link. The

Table 1. Latency Components (clock cycles, cc)

Aurora TX protocol engine:	5 cc $= 32$ ns
MGT TX:	8.5 ± 0.5 cc = 54.4 ± 3.2 ns
1m SATA propagation delay:	4.3 ns
MGT RX:	24.5 ± 1 cc = 156.8 ± 6.4 ns
Aurora RX protocol engine:	5 cc $= 32$ ns
Total latency:	$279.5 \pm 9.6ns$

overall efficiency of the link is the product of efficiencies for each component of the network.

In finding the effective bandwidth of the aurora link, four efficiencies were used: each characterizes different parts of the system. The efficiencies used are for the framing aurora module, the header efficiency, the packet error and retransmit efficiency, and the 8B/10B encoding efficiency.

The efficiency of the aurora framing module is driven by the overhead of sending the start of frame, and end of frame bytes down the link, as well as the bytes inserted for clock correction. This efficiency is given in the following equation, where m is the length, in bytes, of the entire packet, including the header. Four byte overhead is due to sending the 2 byte "start of frame" and 2 byte "end of frame" for every frame on the link. The $12 \times m/9988$ factor represents the overhead of sending 12 clock correction bytes every 10,000 bytes sent.

Framing Aurora Efficiency: $m/(m + 4 + 12 \times m/9988)$

The efficiency of the header is defined as $n/n+h$, where n is the length of the user data to be sent in the packet, and h is the length in bytes of the header that must be sent with the data for control data, such as the sender, receiver, and checksum for this packet.

The efficiency of retransmitting a packet that was lost due to a bit error is defined as the number of packets that are sent over a link by that same number, plus the number of packets that will need to be retransmitted because of a bit error. This assumes that whenever a packet is effected by a bit error, the entire packet will be thrown away, and that the new packet will be retransmitted perfectly.

8B/10B encoding takes 8 user bits and then encodes them into a 10 bit sequence. It is frequently used in high speed communication systems to ensure a sufficient number of bit transitions to keep the transmitter and receiver synchronized. The efficiency of 8B/10B encoding is 8/10.

The bit error ratio (BER) of the networking link is was measured, which is the ratio of incorrect bits to correct bits transmitted in a period of time. In these tests, a 0.5 meter SATA cable was tested. This cable is representative for most of the links in the cluster. Several longer cables, 1 to 3 meters, will also need to be tested. Two revisions of a networking board were completed. The first revision of the board was not manufactured using a controlled impedance process, while the second revision was manufactured with controlled impedances.

Table 2. Bandwidth Calculations

User Data Length (n) =	1 KBytes
Header Length (h) =	32 Bytes
Packet Length (m) =	1056 Bytes
Aurora Efficiency =	$\frac{1056}{1056+4+12\times1056/9988}$
=	0.9950
Header Efficiency =	1024 Bytes / 1056 Bytes = 0.9967
8B/10B Efficiency =	8 bits / 10 bits = 0.8
Rev 1 Packet Retransmit Efficiency =	$36/(36+1) = 0.9730$
Rev 2 Packet Retransmit Efficiency =	$\frac{3\times10^9}{3\times10^9+1} = 1$
Rev 1 Overall Efficiency =	0.7712
Rev 1 Effective Bandwidth	3.125×0.7712 = 2.410 Gbps
Rev 2 Overall Efficiency =	0.7933
Rev 2 Effective Bandwidth =	3.125×0.7933 = 2.479 Gbps

The BER of the first revision was measured to be 3.4×10^{-6}, meaning that on average, 1 bit in 294,000 is wrong. While testing the BER of the second revision board, no bit errors were found in transmitting 2×10^{14} bits. If we assume that an error was going to occur at the instant that the test was stopped, a conservative assumption, the BER of the second revision board would be 5×10^{-15}. The bit errors are assumed to be randomly spaced - not tightly clumped. By making this assumption, the average number of good packets transmitted for every 1 bad packet can be found by 1/(Packet length in bits $\times$ BER).

This comparison of the effective bandwidth does not show a strong justification for the extra cost of the controlled impedance manufacturing of the Rev 2 board. However, if the user data size is increased to 8 KB, the effective bandwidth of the rev 2 board becomes 2.486 Gbps, while the effective bandwidth of the rev 1 board becomes 2.066 Gbps. This shows that the rev 2 board will become very useful as the length of the packets increases. On average, every packet will have an error on rev 1 board if the packet length is increased beyond 40 KB.

3.2.4. System Software & Programming Model

There are several aspects to the system and support software. First, users need a programming model and tools (compilers) to develop their applications. Second, the nodes need an operating system to provide system-level support to applications. Third, in a cluster of FPGAs — especially where each node is effectively an independent system — a mechanism to effectively manage and distribute bitstreams. Each is described below. Although not all of the system software is production-ready, we simply assert that the current state of these tools suggest that the software infrastructure is feasible.

The programming model we adopt for the feasibility

studies is MPI with one variation that was originally proposed as part of the Adaptable Computing Cluster Project [25]. The ML-310 nodes have IBM PowerPC 405 processor cores and, most likely, scientists will be developing software on some other processor. Similarly, the head node is likely to be a high-end workstation processor for the foreseeable future. Hence, there is a need for a cross-development environment that allows users to compile their MPI codes for the cluster. A PowerPC 405 cross compiler (and associated tools) has been built based on the GNU GCC software and Kegel's Cross-Tools scripts. The most recent version (1.1.2) of OpenMPI has been compiled with the cross-compiler as well. We intend to use OpenMPI's Modular Component Architecture (MCA) [21, 27] to directly interface to our high-speed network; presently we have simply compiled in standard TCP/IP support Another aspect that has to be finished is the OpenMPI Run-Time Environment which distribute jobs on a cluster.[2]

We currently are running two built-from-scratch distributions of Linux on the nodes (one uses a 2.4 kernel derived from MontaVista; the other is stock 2.6 kernel). Both distributions use a root file system that is populated with BusyBox utilities plus additional software to enable MPI. It remains to be seen whether a different or stripped version of Linux will be needed.

The final aspect of the system software is called `rboot`. This software infrastructure was developed to remotely manage configuration and boot options of individual FPGA nodes. It also enables users with ssh (secure shell) to access the cluster by routing the console output over TCP/IP to an Internet client. Each ML-310 board as a CompactFlash drive that holds FPGA configuration and the Linux root file system. The rboot system works by powering up the node into a default Linux system that runs an Internet dæmon. Through a custom protocol, bitstreams can be transferred and a remote client can configure the on-board SysACE chip to select a new configuration and reboot. A simple script can reconfigure the entire cluster. (The software also allows users to allocate a subset of nodes which makes it especially useful for education and development.)

4. Related Work

Almost since the Field-Programmable Gate Array was introduced in 1985, researchers have contemplated ways of using FPGAs to build high performance Custom-Computing Machines (CCMs). Several have been used in clusters or direct predecessors of clusters. Early researchers used large collections of FPGAs because individual FPGAs had relatively few resources; hence multiple FPGAs were required to implement a substantial design. Typical examples include the Splash-2 attached processor [2] and "The Virtual Computer" [4]. Although the goal of these systems was to provide a fast co-processor, a kernel of the idea of "clusters of interacting systems" is in these designs. The Splash-2 boards had a configurable interconnection network between the FPGAs and one model for programming these boards was to use a variant of data parallel C [9]. In this model, the processing elements are not independent systems; however, clearly this was approaching a cluster of communicating systems.

Other early CCMs include the Programmable Active Memories (PAM) project [36] which produced the DECPeRLe-1 and DEC PCI Pamette FPGA boards. The first to use FPGAs in a true cluster of independently operating systems was the project called Sepia [15]. Sepia used the DEC PCI Pamette FPGA board with a ServerNet [6] network interface implemented in the FPGA. This allowed the cluster to be a cost-effective interactive image processing platform that supports a number of visualization techniques, including ray tracing, volume visualization, and others.

A number of peripheral-bus cards have been developed over the years, usually to support a specific problem domain (such as the Ace2card [28] from TSI-Telsys which was designed for satellite telemetry). About the same time that Sepia was published, the Adaptable Computing Cluster project began. This project made an FPGA an integral component of the network interface card of a commodity cluster by first using an Ace2card with a commodity Gigabit Ethernet NIC mezzanine card and then later the ISE-East's GRIP2 card. The project investigated the system effects of introducing simple operations in the data path. (In several ways, the current project is direct descendant of the ACC.) By introducing a user-programmable FPGA into the network interface, the premise was that simple user operations performed on in-flight messages could make a significant impact on the performance of applications (in terms of speed and scalability) with modest increases in hardware cost (see [30, 32, 31]).

Other related projects have been built. For example, in 2003 AFRL Rome built a 48-node cluster of PCs with an off-the-shelf FPGA board on the PCI bus [24]. However, it is worth noting the difference between these style of clusters and the Adaptable Computing Cluster and the FPGA cluster described here. The basic node architecture of an off-the-shelf cluster with a PCI card is shown in Figure 8(a). The NIC and FPGA are peers on the peripheral bus. In contrast, the ACC only has *one* peripheral, the Intelligent NIC (INIC). As shown in Figure 8(b), the INIC replaces an ordinary Network Interface Card with one that has FPGA resources juxtaposed between the the peripheral bus and the media access controller. This subtle architectural change (which can be recreated in the current work) allows for hardware operations on data in-transit and does not suffer from having to share the I/O bandwidth of the peripheral bus.

[2] This is somewhat tricky: some code has to run on the head node and some has to be cross-compiled for the nodes.

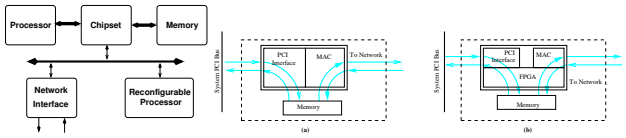

Figure 8. single node of a PCI-card type of cluster and the basic architecture of an ACC node

In 2006, [5] presents simulations of an architecture that is based a scalable direct network which has similarities with the work presented here.

5. Conclusion

The Reconfigurable Computing Cluster project is investigating the feasibility of cost-effective petascale clusters of FPGAs. This paper has argued that petascale computer designer will face challenges that are quite different from those of the last 15 years and put forth the hypothesis that FPGA-based clusters offer a competitive solution. A prototype cluster was described and preliminary data testing the performance of key subsystems was also presented. The results suggest that networking and power in the cluster are on target. However, using the conventional memory subsystem components for Platform FPGAs will not suffice. A higher bandwidth memory controller — one that achieves a higher percentage of the DRAM memory bandwidth — will be needed to hit the petascale targets outlined. Fortunately, options exist and warrant investigation. Overall, there was no "show stopper" in these early results and the proposed approach appears sound.

References

[1] T. Agerwala. System trends and their impact on future microprocessor design. In *MICRO 35: Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press. Invited keynote talk.

[2] J. M. Arnold, D. A. Buell, and E. G. Davis. Splash 2. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 316–324, June 1992.

[3] N. Banerjee, P. Vellanki, and K. S. Chatha. A power and performance model for network-on-chip architectures. In *Proceedings of Design Automation and Test in Europe Conference (DATE 2004)*, pages 1250–1255, Paris, France, Feb. 2004.

[4] S. Casselman. Virtual computing and the virtual computer. In D. A. Buell and K. L. Pocek, editors, *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines*, pages 43–48, Napa, CA, Apr. 1993.

[5] C. L. Cathey, J. D. Bakos, and D. A. Buell. A reconfigurable distributed computing fabric exploiting multilevel parallelism. In J. Arnold and D. A. Buell, editors, *Proceedings of 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'06)*, pages 121–130, Napa, CA, Apr. 2006.

[6] Compaq. Compaq Servernet II SAN interconnect for scalable computing clusters, June 2000. From Whitepaper found at http://www.compaq.com/support/techpubs/whitepapers/-tc000602wp.html.

[7] P. Coteus. Packaging the blue gene/l supercomputer. *IBM Journal of Research and Development*, pages 213–248, 2005.

[8] A. DeHon. The density advantage of configurable computing. *Computer*, 33(04):41–49, 2000.

[9] M. Gokhale and B. Schott. Data parallel C on a reconfigurable logic array. *Journal of Supercomputing*, 9(3):291–313, 1994.

[10] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1996.

[11] R. Jaganathan. Configurable network protocol architecture for the adaptable computing cluster. MS thesis, Clemson University, Department of Electrical and Computer Engineering, May 2003.

[12] R. G. Jaganathan, K. D. Underwood, and R. R. Sass. Reconfigurable network protocol for an INIC. In *Proceedings of the 2003 SC Conference*, Nov. 2003.

[13] E. Joseph, A. Snell, C. G. Willard, S. Tichenor, D. Shaffer, and S. Conway. Council on competitiveness study of ISVs serving the high performance computing market: The need for better application software. http://www.compete.org/hpc/hpc_software_survey.asp, Aug. 2005. Sponsored by Defense Advanced Research Projects Agency.

[14] R. Miller and J. Bellan. Direct numerical simulation and subgrid analysis of a transitional droplet laden mixing layer. *Phys. Fluids*, 12(3):650–671, 2000.

[15] L. Moll, A. Heirich, and M. Shand. Sepia: scalable 3d compositing using PCI Pammette. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 146–155, Napa Valley, CA, April 1999.

[16] T. Mudge. Power: A first-class architectural design constraint. *Computer*, 34(4), Apr. 2001.

[17] H. Nakashima, H. Nakamura, M. Sato, T. Boku, S. Matsuoka, D. Takahashi, and Y. Hotta. Megaproto: 1 tflops/10kw rack is feasible even with only commodity technology. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 28, Washington, DC, USA, 2005. IEEE Computer Society.

[18] NCBI user services. Genbank overview, Aug. 2005. http://www.ncbi.nlm.nih.gov/Genbank/.

[19] M. Nelson. Mesh fabric switching with Virtex-II Pro FPGAs. *Xcell Journal*, 49(2):34–44, Summer 2004. http://www.xilinx.com/publications/xcellonline/xcell_49/xc_toc49.htm.

[20] e. a. NR Adiga. An overview of the bluegene/l supercomputer. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–22, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

[21] Open HPC, Inc. Open mpi: Open source high performance computing, 2005. http://www.open-mpi.org/papers/euro-pvmmpi-2004-overview/.

[22] V. Pande. Folding@home: Advances in biophysics and biomedicine from world-wide grid computing (invited keynote). In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - HiCOMB Workshop 7*, page 196.2, Washington, DC, USA, Apr. 2005. IEEE Computer Society.

[23] F. Petrini, D. J. Kerbyson, and S. Pakin. The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of asci q. In *SC '03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, page 55, Washington, DC, USA, 2003. IEEE Computer Society.

[24] V. W. Ross. Heterogeneous high performance computer. In *Users Group Conference*, pages 304–307, 2005.

[25] R. R. Sass, K. D. Underwood, and W. B. Ligon, III. Design of the adaptable computing cluster. In *Proceedings of the 4th Annual Military and Aerospace Applications of Programmable Devices and Technologies International Conference*, Laurel, Maryland, USA, Sept. 2001.

[26] A. Schmidt. Quantifying effective memory bandwidth in platform fpgas. Master's thesis, University of Kansas, May 2007.

[27] J. M. Squyres and A. Lumsdaine. The component architecture of open MPI: Enabling third-party collective algorithms. In V. Getov and T. Kielmann, editors, *Proceedings, 18th ACM International Conference on Supercomputing, Workshop on Component Models and Systems for Grid Applications*, pages 167–185, St. Malo, France, July 2004. Springer.

[28] TSI Telsys. Ace2 card manual, 1998.

[29] K. Underwood and K. S. Hemmert. Closing the gap: CPU and FPGA trends in sustainable floating-point blas performance. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 219–228, April 2004.

[30] K. D. Underwood. *An Evaluation of the Integration of Reconfigurable Hardware with the Network Interface in Cluster Computer Systems*. PhD thesis, Clemson University, Aug. 2002.

[31] K. D. Underwood, W. B. L. III, and R. Sass. Analysis of a prototype intelligent network interface. *Concurrency and Computation: Practice and Experience*, pages 751–777, 2003.

[32] K. D. Underwood, R. R. Sass, and W. B. Ligon. A reconfigurable extension to the network interface of beowulf clusters. In *Proceedings 2001 IEEE Conference on Cluster Computing*, pages 212–221, Newport Beach, CA, October 2001.

[33] U.S. Food and Drug Administration. Parkinson's disease new treatments slow onslaught of symptoms, June 2004. http://www.fda.gov/fdac/features/1998/498_pd.html.

[34] A. J. van der Steen and J. J. Dongarra. Overview of recent supercomputers.

[35] P. Vellanki, N. Banerjee, and K. S. Chatha. Quality-of-service and error control techniques for mesh based network-on-chip architectures. *INTEGRATION: The VLSI Journal*, 38:353–382, 2005.

[36] J. E. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. H. Touati, and P. Boucard. Programmable active memories: reconfigurable systems come of age. *IEEE Trans. Very Large Scale Integr. Syst.*, 4(1):56–69, 1996.

[37] W. A. Wulf and S. A. McKee. Hitting the memory wall: Implications of the obvious. *Computer Architecture News*, 23(1):20–24, 1995.

[38] Xilinx, Inc. Aurora reference design, 2004. http://www.xilinx.com/aurora.

[39] Xilinx, Inc. Ml310 documentation and tutorials. http://www.xilinx.com/ml310, June 2005.

[40] Xilinx, Inc. Logicore aurora v2.4 user guide. http://www.xilinx.com/, Dec. 2006.