

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

Ron Sass, William V. Kritikos, Andrew G. Schmidt, Srinivas Beeravolu, Parag Beeraka,
Kushal Datta, David Andrews, Richard S. 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}@itcc.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 Platform 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. Microbenchmark results are reported to answer several pragmatic questions about key subsystems, including the system

software, network performance, memory bandwidth, power consumption of nodes in the cluster. Results suggest that the approach is sound.

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 [32] to ecologists simulating complex, multi-species ecosystems, to computational biologists predicting protein folds [21] to physicists studying Computational Fluid Dynamics (CFD) [16] — the performance of state-of-the-art high-end computers limits the resolution of experiments. This, in turn, dictates the scope of scientific endeavors. Indeed, many have suggested that access to cost-effective petascale computing would enable science that is simply not possible today [10].

The last two decades have seen the emergence of Beowulf-class computers (clusters of commodity off-the-shelf hardware combined with Open Source software) which — in conjunction with tremendous gains in microprocessors — 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 networking specifically designed for HPC, it is interesting to note that the most common networking technology on the list is commodity Ethernet [33]. Even in an arena where speed is king, the marketplace still reflects a sensitivity to cost.

However, despite their current prominence, it is not clear that commodity cluster technology will remain cost-

*This project was supported in part by the National Science Foundation under NSF Grants CNS 06-52468 (CRI) and CNS 04-10790 (EHS). 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

effective when scaled to the PetaFLOPs range. New issues — such as power, cooling, and physical infrastructure — are becoming increasingly important. Additional issues, such as bandwidth to primary and secondary storage, are emerging as significant challenges. (This point is expanded upon in section 2; section 2 of [3] outlines a similar argument.)

The goal of the Reconfigurable Cluster Computing (RCC) Project is to explore the feasibility of building cost-effective petascale HPC clusters from Platform FPGA nodes. It is expected that FPGA floating-point performance will continue to rise (both in absolute terms and relative to single-core processors [27]). While a number of current HPC projects are using FPGAs as “compute accelerators” for standard microprocessors, the proposed approach is fundamentally different in that nodes are composed exclusively of FPGAs. (This is possible because Platform FPGAs are capable of hosting an entire Linux-based system.) The central hypothesis of the project is that the proposed approach will lead to more cost-effective HPC system than commodity clusters in the petascale range.

Of course, there are a number pragmatic questions and potential pit-falls that could limit the expected performance gains. The ultimate aim of this multi-disciplinary project is to bring together engineers and computational scientists from several Universities to test real, computationally challenging science applications on a prototype cluster. Teaming computer engineers and domain scientists on every application is not an effective, long-term solution; however, it does provide invaluable hard data for computer engineers evaluating the design while advancing our collaborator’s science program.

The work reported here focuses on the initial design, construction of a prototype, and the preliminary investigations of key subsystems. Specifically, the contributions are three-fold. First, the report provides an outline of the challenges faced by current commodity clusters (section 2). Second, a set of system-wide performance metrics are proposed and a “straw man” petascale Platform FPGA cluster is described (section 3). Last, in section 4, a 64-node prototype cluster that is currently under construction at the University of North Carolina, Charlotte is described. This report includes results from a set of microbenchmark experiments that investigate the power, chip-to-chip bandwidth, and RAM bandwidth of the prototype cluster. It concludes with related work (section 5) and a summary in section 6.

2. Motivation

Technology and application trends are changing rapidly and this will have a significant impact on the scalability of commodity clusters. To put these trends in context, consider two representative computational science problems: Computational Fluid Dynamics (CFD) and Bioinformatics (specifically, sequence searching). Scientists working in

these domains are interested in solving ever-larger problems but as the problem size increases, so does the relative demands on various subsystems of the computer. Unfortunately, many of these demands are at odds with current technology trends.

2.1. Application and Technology Trends

By increasing the resolution of computer simulations, experiments reveal phenomena and insights into processes that are otherwise unobservable or impractical to experiment with in the physical world. For example, by increasing the resolution of their experiments, CFD scientists are able to observe very small-scale phenomenon using the first principles of physics. This can lead to better models for simulating large structures, such as jet engines. However, simply doubling the resolution of such experiments increases the computation 16-fold. The increase is due in part to the fact that the data set increases in three dimensions ($2^3 \times$ more data) and that smaller time-steps are required for mathematical stability.

Although the number of function units per die and memory capacity continue to grow very quickly, memory speed is growing much slower. (For example, DRAM capacity is doubling every 18 to 24 months while DRAM speed is only doubling every 48 months.) In other words, technology may provide the computational and memory capacities to solve larger CFD problems, but performance will be limited by the memory interface. Without corresponding improvements to the bandwidth between the processor and primary storage, one cannot rely on performance to scale with Moore’s Law. Indeed, as Underwood and Hemmert [28] show, there is ample computing resources on FPGAs for dense matrix algebra operations but the FPGA’s speed is limited by memory bandwidth.

For scientists interested in investigating the function of genes or proteins, current trends are especially challenging. This is because, 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 [12] suggest a more modest 6% growth rate. Regardless, the consequence is profound: 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, much faster than the bandwidth of I/O subsystems.

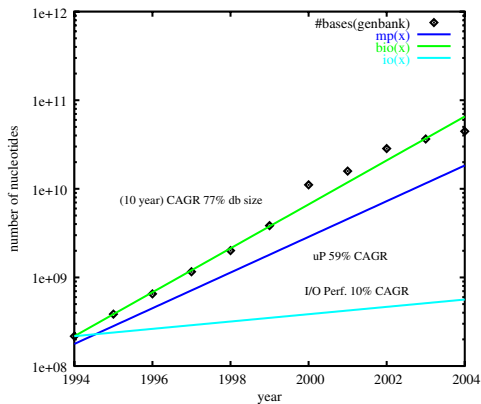


Figure 1. compound annual growth rate of problem size, single processor performance, and I/O subsystem performance

2.2. Commodity Cluster Solutions

Of course, commodity clusters are changing as well. As PC prices continue to drop, one option is to increase the number of nodes. Similarly, the introduction of multi-core processors provides another dimension in which to scale. Both of these options are considered individually below.

Multi-core processors will allow the on-chip rate-of-computation to continue to grow exponentially with Moore's Law. However, as just mentioned, this does not address the issue of primary memory speed. This so called "Memory Wall" was predicted many years ago [37] but microprocessors have been able to avoid the growing disparity by using ever-larger on-chip caches. Thus it possible that some clever research will emerge that again postpones the bottleneck issue; however to date the only solution has been to create ever-larger data caches. There are other unanswered research questions as well, such as how hundreds of cores on a single chip will communicate (on-chip and across the nodes of cluster). The point is not to say that multi-chip processors have insurmountable problems. Rather just to note that the technology is not without risk.

An alternative to a multi-core solution is to simply buy more (presumably less expensive) processors with fewer cores per chip. This introduces new problems, such as sheer size and power requirements. Consider a modern (circa Fall 2006) commodity cluster. For about \$200,000, one can buy 64 dual core, dual processor nodes in two racks with InfiniBand networking. Each node would typically 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 around 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 elec-

trical system would have to support the delivery of 40 MW of power to the system and even more energy to cool the machine (as the machine is converting all of that energy into heat). And, of course, there is the practical question of what commodity switch would allow 78,000+ nodes to efficiently communicate.

There is an additional, subtle cost issue associated with scaling a commodity microprocessor solution. A corollary of Moore's Law is that the cost/transistor is cut in half every technology cycle. Commodity clusters have been leveraging this to keep their larger and larger systems cost-effective. However, the rest of the node is not made of silicon and does not follow Moore's Law or its corollary. These components are made from commodities (steel, copper, FR4) 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 is that the peripheral components of the nodes (cases, cables, and so) will take an increasing amount of the budget. It is the Law of Diminishing Returns applied to the cost of a node.

2.3. Alternative Solutions

A full discussion potential PetaFLOP solutions is beyond the scope of this report. However, it is instructive to consider the growing set of alternative HPC solutions.

Two approaches have attracted a lot of attention recently. At 280.6 teraflops, the IBM BlueGene/L installed at Lawrence Livermore National Laboratory is the fastest computer in the world today. This architecture uses a very large number (hundreds of thousands) of relatively slow processors in System-on-a-Chip modules. Assuming its performance continues to scale, which is a reasonable assumption, it will need roughly a half-million processors to hit a PetaFLOP. A second alternative approach is to use high-performance custom chip designs — either as "compute accelerators" or as a basic node in a cluster. This exemplified by the those building clusters of Cell Broadband Engine chips, researchers using Graphics Processing Units for general-purpose applications (GPGPUs), and vendors offering special-purpose co-processors. Some of these custom chip solutions are relatively young technologies (none have the prominence of, say, BlueGene/L) and are under rapid development. Generally speaking, these are still custom solutions which are not as cost-effective as commodity clusters.

In summary, 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? One approach is to continue to use commodity clusters by taking advantage of gains in semiconductor technology to grow the number of cores on a single chip or grow the number of individual processors to

scale to a PetaFLOP. However, the former will have to address the memory bandwidth issue and the latter will have to get around commodity packaging to address physical scaling issues. And, of course, neither approach directly addresses the secondary storage issue. FPGA characteristics enable a new approach worthy of investigation which is described next.

3. Design of Petascale FPGA-Based Cluster

Until recently, Field-Programmable Gate Arrays have had the reputation of being slow, power hungry, and not very good for floating-point applications. However, these generalizations are based on a snap shot in time and depend heavily on the context in which the FPGA is employed. This issue, along with a set of system metrics and a straw man system design, are explored in this section.

3.1. FPGA Characteristics

For HPC systems, floating-point performance is fundamental. However, it has only been in the last decade that FPGAs have had enough resources to even implement an IEEE floating-point unit on a single device [36]. Fortunately, as Moore's Law provides exponentially more transistor's — and since FPGA devices have begun including more diffused IP such as multipliers — the floating-point capabilities of FPGAs are growing dramatically [27, 28]. Peak floating-point performance is expected to continue to grow and as Underwood and Hemmert [28] point out, FPGA floating-point performance is not limited by computing resources. Rather, floating-point performance is limited by memory speed which is the issue. One important characteristic of FPGAs is that there are a wide range of memory technologies/ potential memory organizations available to explore with FPGA designs.

Another characteristics where FPGAs seem counter-intuitive is power consumption. While FPGAs generally consume $12\times$ as much power as an ASIC [15], in this domain it is more appropriate to compare the power consumption of a Platform FPGA against the power consumption of a node in a commodity cluster. Using just a rough estimate (power supply wattage), it is worth noting that modern processors generally require 500W power supplies while FPGA developer boards ship with 250W power supplies. Although hardly a conclusive argument, we present data later that suggests a Platform FPGA requires an order-of-magnitude less power than a typical microprocessor-based node.

Again, compared to ASICs, an equivalent FPGA design is generally $3\times$ slower and is $40\times$ larger [15]. However, the clock rate is not as important to HPC as is the rate of computation. As long the rate (floating-point operations per second) is sustained then slower clock rates can provide a power advantage.

There are other aspects to Platform FPGAs that make them especially useful for petascale clusters. One advantage is that, because a single FPGA is now large enough (and has a rich enough set of on-chip IP), it is able to host an entire system. This eliminates many of the peripheral components found on a standard commodity PC board which in turn offers size and weight advantages. Also, by reducing the size of the node, there is a secondary advantage in that it reduces the distance between nodes enabling novel networking options.

3.2. Performance Metrics

To make rational, engineering-based decisions about a scalable design, we propose a collection of metrics. These metrics aim to relate several components that overall comprise system performance.

Most end-user installations will have a practical limit to how much electrical current the equipment can draw. Moreover, the power used to drive a computer system is converted into heat which subsequently raises the ambient temperature. High temperatures cause equipment failures and can increase the power that a device consumes. Most situations require active cooling which draws even more power. Furthermore, as systems scale up, the GFLOPS/W metric is of interest because it is a major component of operating cost.

Another important metric relates cost and speed. The GFLOPS/\$ ratio of a design is the capital 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 may require adding another switch or the use of an additional 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.

Space is another increasingly important metric. For many institutions, lab 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 clusters to a PetaFLOP due to the number of additional racks. For this and other reason, GFLOPS/m³ 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.

Also related to space is the networking technology that can be deployed. Specifically, 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/m³ 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.

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. However, using MT/s/GFLOPS alone provides an incomplete 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 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 \$10M, has a 25×25 ft² footprint and is within a factor-of-2 of a PetaFLOP/second performance (0.5 PetaFLOPS).

3.3. Straw Man Design

What would a petascale FPGA-cluster look like? At this point in the project, any answer would be highly speculative. However, having a starting point is useful and, by establishing a set of specific decisions, a straw man design

provides an initial baseline from which to test alternatives.

As pointed out in the introduction, the importance of cost-effectiveness cannot be understated. This would suggest that the cluster should use commodity components. However, it seems unlikely that a commodity FPGA node will meet the project's goals. The major components of a hypothetical custom node — drawn to scale and with appropriate spacing — are shown in Figure 2 (right side of figure). Although the NRE costs are more expensive, there are several advantages to using a custom, small board: it enables the infrastructure advantages of Platform FPGA (shorter cables and less node infrastructure).

Twelve of these 15.25×14.25 cm² nodes will fit in a standard 19-inch rack mount chassis as shown in Figure 2 (left side). 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 racks each and will fit into a 24' by 15' footprint. This organization would allow for a 13,824 node 3-D torus to be implemented with approximately 864 1-meter cables and about 12,960 standard SATA cables. This paper design, with a very large Platform FPGA, has a theoretical, peak floating-point performance of a PetaFLOP.

As any good straw man design should, this proposal raises more questions than it answers. To answer these questions, the project has a two-step approach. First, a 64-node scale model is under construction at the University of North Carolina at Charlotte. This model uses commodity developer boards to answer basic feasibility questions. If successful, the second step would be build prototype nodes, as described in the straw man, and test scalability with a ≈ 700 node cluster.

4. RCC Prototype and Measurements

Complex systems defy purely analytical evaluations and the straw man just proposed is no exception. 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 ratio of communication channels and have a significant impact on system performance. These effects are nearly impossible to model with a discrete event simulator, let alone on paper. Consequently, the approach taken by the RCC Project is to construct a two-rack scale model using commodity components and take direct measurements. The commodity boards are an imperfect match for the straw man proposed and this cluster is much too small to ever approach a PetaFLOP. However, it does allow us to investigate key subsystems and evaluate performance.

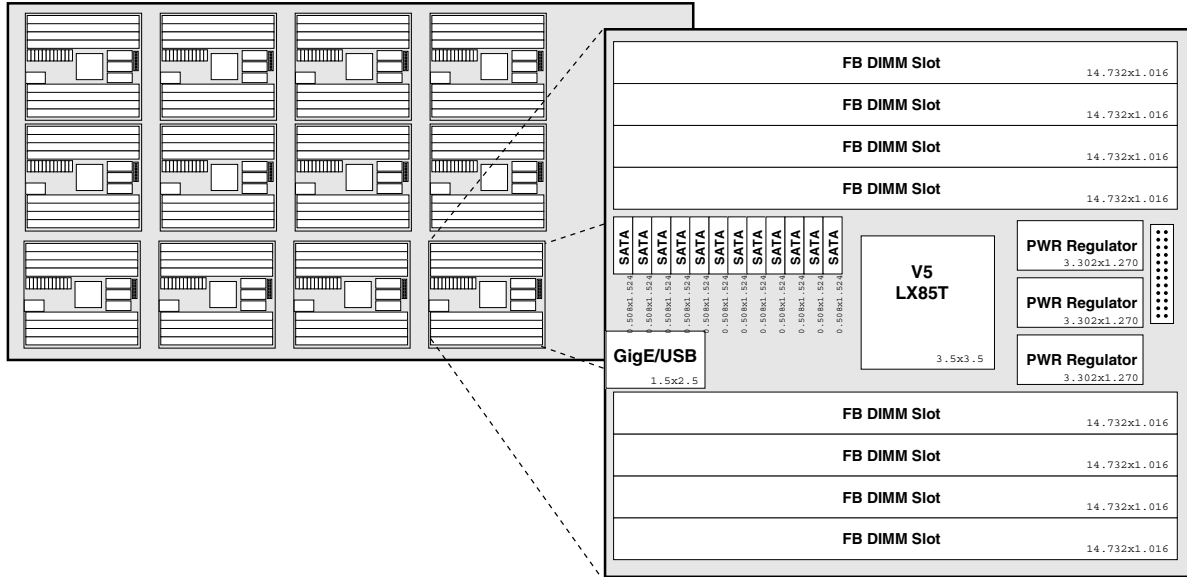


Figure 2. straw man proposal of one Platform FPGA node; twelve FPGA nodes in a 1U chassis

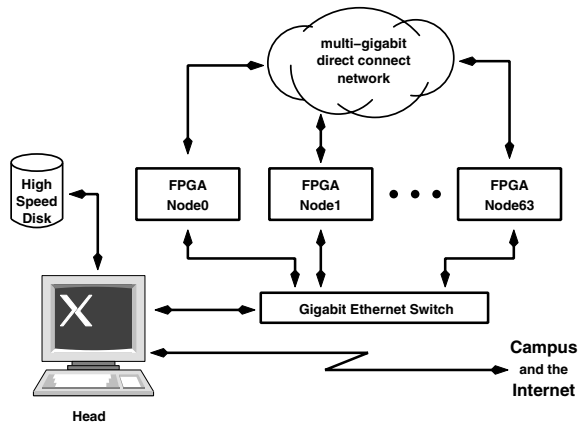


Figure 3. block diagram of proposed cluster

4.1. Prototype Cluster

A block diagram of the 64-node cluster under construction at the University of North Carolina at Charlotte is illustrated in Figure 3. FPGA-based compute nodes, a custom direct point-to-point network, and two 48-port Nortel 5510-48T GigE Ethernet switches comprise the core components of the cluster. The Head node has a high-speed SATA disk I/O subsystem. (A proposal currently under review will, if funded, augment every node of the cluster with multi-disk secondary storage.)

Nodes All of the experiments reported here were performed using Xilinx ML-310 Development boards[39], an ATX form factor board with a Virtex II Pro (V2P30) Plat-

form FPGA. However, the cluster under construction will be equipped with 64 Xilinx ML-410 Development boards[41] with a Virtex-4 (V4P60) FPGA and two gigabit Ethernet cores per node.

A logical representation of a typical system architecture, called Base System Platform or BSP, is shown in Figure 4(a). The solid lines represent diffused IP cores (implemented in CMOS transistors) and the dashed lines represent cores implemented in CLBs. A typical configuration used in the cluster is illustrated in Figure 4(b). 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).

Networks There are several communications networks in the cluster. One not shown in Figure 3 is a USB network the consolidates the serial console outputs of every node. This network plays an important role in debugging. The Gigabit Ethernet's role is largely administrative (providing TCP/IP access to individual nodes and for transferring configuration bitstreams to the nodes).

The third network uses the Multi-Gigabit Transceivers (MGTs), also known as RocketIOs, to provide a custom high-speed data network. 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. The BSP shown in Figure 4(b) includes an on-chip router that connects computing elements and to support MPI message-passing programs. (Note: not all computing ele-

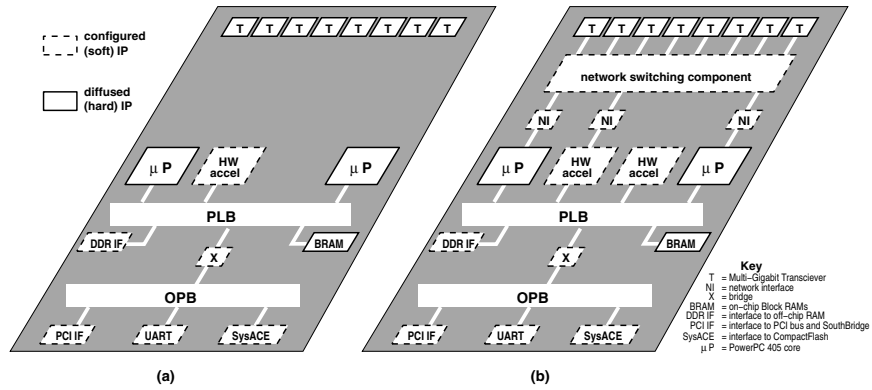


Figure 4. (a) typical base system platform (b) network-enhanced base system platform

ments need a network connection. For example, one core that computes e^{-x} is simply a hardware-assist on a local processor bus.)

Using Aurora protocol cores, a network switch core, and a network layer core, a majority of the data communication protocol processing can be handled in hardware rather than software. Aurora is a freely-available link-layer protocol which 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 [13] and [14] for details of the configurable network layer protocol), adding 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 [34] and [4]. Nelson [19] has also developed a Advanced Telecom Communications Architecture (ATCA) router for FPGAs which provides a rich set of functionality with ATM-like communications; however, it requires a large amount of configurable logic and only a stripped down version would be a likely candidate for HPC communications. A third option is to use an external network switch — the savings in FPGA resources may make it more attractive.

Access to the high-speed network transceivers presents a technical challenge. High-speed signaling is very sensitive to noise and distance requiring the cabling and cable-to-FPGA interface to 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 (IB), 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 over short distances. One benefit of keeping the nodes physically close is the ability



Figure 5. ML-310 with installed with our custom multi-gigabit transceivers to SATA adapter board (inset)

to use the less expensive 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 5, revealed some design flaws in testing. The second revision resolved all of the known issues and greatly improves the performance. The performance of both revisions are described below.

Power 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) control (turn on or off) power to individual nodes.

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

4.2. Microbenchmark Results

Initial investigations have focused on the memory performance, power, and link layer networking. Measurements have been taken on the ML-310 nodes. (The 64 ML-410 nodes are scheduled for delivery in June 2007).

4.2.1. Memory Performance

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

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 3200 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 operating characteristics.² Complete details on the fourteen combinations studied and how the experiments were conducted can be found in [24] and [25].

4.2.2. Power

Estimating power consumption is challenging. In general, the (dynamic) power consumed by a CMOS device is determined by how often transistors switch which is a function of the clock frequency, the design, and the run-time data. Thus, to estimate the power consumption of an FPGA, one has to use real designs with inputs that vary over time. Also, for small circuits, FPGA designs exhibit discontinuities: I.e.,

²SDRAMs do not have uniform access times; ideally, one would like to schedule memory accesses to best match SDRAM characteristics.

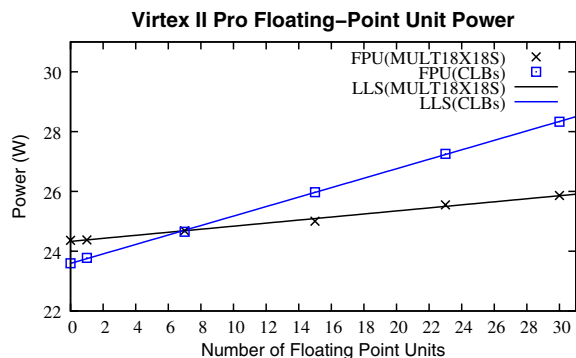


Figure 6. power consumed by implementing floating-point units with multiplier blocks and CLBs; points are measured, lines are fitted linear-least squares

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. It is especially important to test designs that approach the capacity of the device since PAR/MAP strategies change as resource utilization rises. The following experiment was designed to do this.

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 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.³

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 6. 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

³Scientific applications exhibit a high-degree of “value locality” — the integer values 0 and 1 are very common inputs — so these experimental results are probably conservative. Real applications may use less power.

of using FPGAs? First, the resources used to instantiate 30 FPU 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.

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 [8]. This suggests that (a) cooling a petascale FPGA cluster will be feasible and (b) specialization and custom architectures may be able to compensate for the inherent power disadvantages of FPGAs (compared to CMOS SoC technology).

4.2.3. Chip-to-Chip Networking

Until the cluster is assembled, the project is focusing on the link layer performance of the custom network. Below we evaluate the latency and bandwidth of this point-to-point network.

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 is a two byte, single lane design, meaning that the it takes in two bytes in parallel each user clock cycle, and only one transceiver is used (there were no bonded channels).

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 ± 9.6 ns |

The value for the propagation delay of the SATA 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 overall efficiency of the link is the product of efficiencies for each component of the network.

In order to measure 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.

First, 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. Equation 1 models Aurora framing efficiency.

$$m / (m + 4 + 12 \times m / 9988) \quad (1)$$

In this equation, m is the length, in bytes, of the entire packet, including the header. Four byte overhead is due to sending the two byte “start of frame” and two 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.

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.

The 8B/10B encoding takes eight user bits and then encodes them into a ten 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

Table 2. Bandwidth Calculations

| | |
|--------------------------------------|---|
| User Data Length (n) = | 1 KBytes |
| Header Length (h) = | 32 Bytes |
| Packet Length (m) = | 1056 Bytes |
| Aurora Efficiency = | $\frac{1056}{1056+4+12 \times 1056/9988}$ |
| | = 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 = | $\frac{3 \times 10^9}{3 \times 10^9 + 1} = 1$ |
| Rev 1 Overall Efficiency = | 0.7712 |
| Rev 1 Effective Bandwidth | $3.125 \times 0.7712 = 2.410$ Gbps |
| Rev 2 Overall Efficiency = | 0.7933 |
| Rev 2 Effective Bandwidth | $3.125 \times 0.7933 = 2.479$ Gbps |

synchronized. The efficiency of 8B/10B encoding is 80%.

The bit error ratio (BER) of the networking link is 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 is tested. This cable is representative of most of the links in the cluster. Several longer cables, 1 to 3 meters, also will 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.

The BER of the first revision was measured to be 3.4×10^{-6} , meaning that on average, one 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 one bad packet can be found by $1/(\text{Packet length in bits} \times \text{BER})$.

The comparison in Table 2 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.

4.2.4. 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 [23]. 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) [20] 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.⁴

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 a 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 and control the nodes of the cluster via 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. More information about this software and how it supports FPGA laboratory education can be found in [9].

⁴This is somewhat tricky: some code has to run on the head node and some has to be cross-compiled for the nodes.

5. 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” [5]. 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 [11]. 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 [35] 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 [17]. Sepia used the DEC PCI Pamette FPGA board with a ServerNet [7] 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 [26] 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 [29, 31, 30]).

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 [22]. However, this is different from the Adaptable Computing Cluster — where the FPGA was in the network data path — and very different from the current project where the Platform FPGA

is the node.

In 2006, [6] presents simulations of an architecture that is based a scalable direct network which has similarities with the work presented here.

6. 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] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, December 18 2006.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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>.
- [8] P. Coteus. Packaging the blue gene/l supercomputer. *IBM Journal of Research and Development*, pages 213–248, 2005.
 - [9] K. Datta and R. Sass. Rboot: Software infrastructure for a remote fpga laboratory. In *FCCM '07: Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, page ???, Washington, DC, USA, 2007. IEEE Computer Society.
 - [10] e. a. Earl Joseph. 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.
 - [11] M. Gokhale and B. Schott. Data parallel C on a reconfigurable logic array. *Journal of Supercomputing*, 9(3):291–313, 1994.
 - [12] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1996.
 - [13] R. Jaganathan. Configurable network protocol architecture for the adaptable computing cluster. MS thesis, Clemson University, Department of Electrical and Computer Engineering, May 2003.
 - [14] 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.
 - [15] I. Kuon and J. Rose. Measuring the gap between fpgas and asics. In *FPGA '06: Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*, pages 21–30, New York, NY, USA, 2006. ACM Press.
 - [16] 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.
 - [17] 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.
 - [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] Open HPC, Inc. Open mpi: Open source high performance computing, 2005. <http://www.open-mpi.org/papers/euro-pvmmmpi-2004-overview/>.
 - [21] 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.
 - [22] V. W. Ross. Heterogeneous high performance computer. In *Users Group Conference*, pages 304–307, 2005.
 - [23] 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.
 - [24] A. Schmidt. Quantifying effective memory bandwidth in platform fpgas. Master's thesis, University of Kansas, May 2007.
 - [25] A. G. Schmidt and R. Sass. Quantifying effective memory bandwidth of platform fpgas. In *FCCM '07: Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, page ???, Washington, DC, USA, 2007. IEEE Computer Society.
 - [26] TSI Telsys. Ace2 card manual, 1998.
 - [27] K. Underwood. Fpgas vs. cpus: trends in peak floating-point performance. In *FPGA '04: Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays*, pages 171–180, New York, NY, USA, 2004. ACM Press.
 - [28] 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.
 - [29] 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.
 - [30] 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.
 - [31] 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.
 - [32] U.S. Food and Drug Administration. Parkinson's disease new treatments slow onslaught of symptoms, jun 2004. http://www.fda.gov/fdac/features/1998/498_pd.html.
 - [33] A. J. van der Steen and J. J. Dongarra. Overview of recent supercomputers.
 - [34] 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.
 - [35] 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.
 - [36] I. W. B. Ligon, G. Monn, S. P. McMillan, K. Schoonover, F. Stivers, and K. D. Underwood. Implementation of ieee single-precision floating-point operations on fpgas. In *FPGA '98: Proceedings of ACM/SIGDA 6th International Symposium on Field Programmable Gate Arrays*, page 258, 1998.
 - [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. MI310 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.
 - [41] Xilinx, Inc. MI410 documentation and tutorials. <http://www.xilinx.com/ml410>, Jan. 2007.