Cluster Comput (2014) 17:315–325
DOI 10.1007/s10586-013-0278-3

Architecture and applications for an All-FPGA parallel computer

Yamuna Rajasekhar · Ron Sass

Received: 17 February 2013 / Accepted: 13 May 2013 / Published online: 29 June 2013

Abstract The Reconfigurable Computing Cluster (RCC) project has been investigating unconventional architectures for high end computing using a cluster of FPGA devices connected by a high-speed, custom network. Most applications use the FPGAs to realize an embedded System-on-a-Chip (SoC) design augmented with application-specific accelerators to form a message-passing parallel computer. Other applications take a single accelerator core and tessellate the core across all of the devices, treating them like a large virtual FPGA. The experimental hardware has also been used for basic computer research by emulating novel architectures. This article discusses the genesis of the overarching project, summarizes results of individual investigations that have been completed, and how this approach may prove useful in the investigation of future Exascale systems.

Keywords FPGAs · Cluster · Architectures · Exascale

1 Introduction

For the past few decades, parallel computing architecture research has revolved around two axes: improving the performance of the (single core) von Neumann microprocessor and either low-cost (Ethernet) or specialized interconnects (IBM SP [26], Quadrics [15], Myrinet [4], InfiniBand [8]). However it is widely believed that the extreme scale of future parallel machines (circa 2020) will necessarily be constructed using a very different set of principles [3]. Unfortunately, there is no agreement yet on what those principles will be. This motivates the exploration of unconventional computer architectures and how to map extant and future applications to these emerging architectures.

In 2006, the Reconfigurable Computing Systems Laboratory (RCS Lab) at the University of North Carolina at Charlotte (UNC-Charlotte) began construction of *Spirit*, a parallel computer comprised of 64 FPGA-based nodes and a configurable, high-speed custom interconnect. With no discrete microprocessors, this parallel computer uses the configurable logic in the FPGA to implement multicore system-on-a-chip nodes, a network router, and application-specific hardware accelerators. Based on the past six years of experience with this very unconventional architecture, this articles explains some of the original motivations for the project, summarizes a number of individual projects, and explores the role of FPGAs in future parallel computing architecture research. This article provides a roadmap of the applications built for the cluster over the years and also describes some current projects that are under development. Although the expertise required to exploit *Spirit*'s capabilities is not widespread today, one of the major contributions of the overarching project is to show that it is not intractable and that tools under development now are making the technology increasingly accessible.

Y. Rajasekhar (✉) · R. Sass
Reconfigurable Computing Systems Laboratory, University of North Carolina at Charlotte, 9201 University City Blvd, Charlotte, NC 28223, USA
e-mail: yrajasek@uncc.edu

R. Sass
e-mail: rsass@uncc.edu

2 Motivation

Before presenting an overview of the research and applications that have been used to explore the architecture, we present a brief description of our original thinking and the details of the experimental hardware implementation.

2.1 View in 2004

This architecture was proposed in 2004 and was loosely inspired by two key technologies. The first was the success of the IBM BlueGene/L project [1]. It used a very large number of PowerPC 440 cores in a custom ASIC chip. This was particularly interesting because the PowerPC 440 model was commonly used in power-constrained embedded systems projects, not high-performance computing situations. The second technology was the introduction of FPGA devices with many integrated, multi-gigabit transceivers and PowerPC 405 (another embedded system processor). This allowed the construction of an all-FPGA cluster (no discrete microprocessors). However, unlike the BlueGene/L, the configurability of the FPGA allows application-specific accelerators to be configured and reconfigured into the computational platform. Moreover, these accelerators would be tightly coupled to the processor cores—sharing the same memory hierarchy and having their own interface to the system interconnection network. The other role envisioned for the proposed system was a flexible vehicle for computer architecture research.

2.2 Structure of spirit

The *Spirit* cluster (see [20] for complete description of the architecture) is a 64-node all-FPGA machine that was deployed in 2006. This message-passing parallel computer uses commercial, off-the-shelf FPGA developer boards from Xilinx (the ML410 [28]). The nodes in the cluster are connected by using a networking board developed in-house that brings the high speed transceivers out to eight bidirectional channels which are used to connect the FPGA nodes in a 4-ary 3-cube. The assembled system can be seen in Fig. 1. As shown, the system has been put together in two standard 19 inch racks. A custom network interface card that includes a serial to USB converter along with a JTAG interface was designed for this cluster. There also exists a sophisticated tool suite that acts as a management framework for efficient debug, development and production of designs for nodes in the cluster. MPI style applications have been developed to run collective communication tasks on the cluster. The next Sect. 3 gives a brief summary of the tools to use the cluster in an effective manner and applications that were accelerated using Spirit in the past.

3 Research projects

Over the last six years, the experimental equipment has been used for a variety of research projects. These include projects that provide basic infrastructure for the machine (such as management software and generic packet routing) to computational science applications (bioinformatics) to innovative architectures to support future extreme scale machines (collective communications, a hardware filesystem, advanced system resilience support).

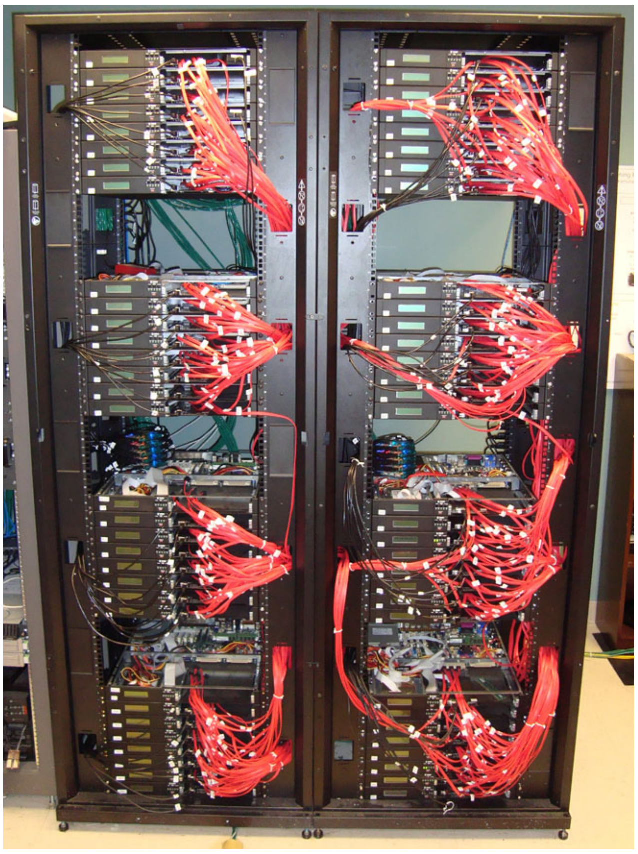


Fig. 1 Physical organization of *Spirit*

3.1 Infrastructure

FPGA session control Unfortunately, virtually all commercial tools assume there are relatively few FPGA devices on a single JTAG chain. When multiple (USB) JTAG cables are attached to a single host, it is up to the user to disambiguate which USB device is associated with which JTAG chain. (The association depends on how USB enumerates the end points.) This issue is further exacerbated when the number of JTAG chains, combined with UART serial consoles, exceeds the number of allowable end points on a USB network. In this case, the system has to resort to multiple hosts with multiple JTAG chains per host to support a large (>128) number of nodes. And, on top of this, these systems will need to be remotely accessible. As we started deploying this large system, it became increasingly obvious that not only will it need to be shared among multiple researchers (that maybe geographically separated) but also that there was a need for a comprehensive tool that would allow effective use of the entire system.

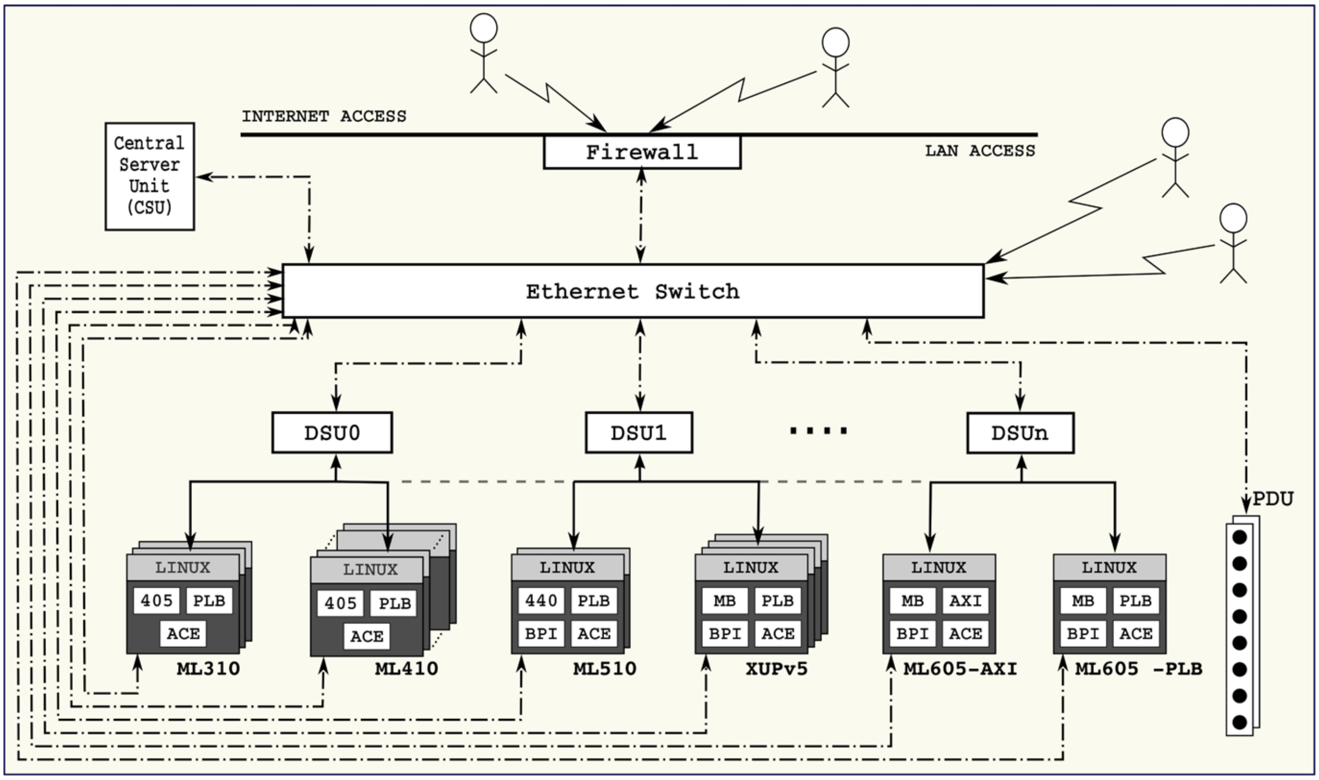


Fig. 2 Overview of FSC mechanism indicating abstraction of FPGA platforms from the Central Management Unit (Here, PLB and AXI denote the Interconnects; 405/440 are PowerPC4xx Processor and MB is the Microblaze Processor; BPI and ACE are the BPI and SysACE Compact Flash; DSUx are the Distributed Service Units and PDU is the Power Distribution Unit.)

FPGA Session Control (FSC) [17] was first developed in 2007 with just the bare minimum functionality required—remote access (power on/off the nodes), the ability to upload and run a design (with a ⟨*.ace*⟩ file) on the node. Its main purpose was to serve as a pedagogical aid. Since then, an array of new features have been added to make FSC a more advanced tool. FSC 2.0 [29] has been modified to include job scheduling, the ability to effectively manage multiple bitstreams and advanced features to debug designs remotely.

At the heart of the FSC suite is a client-server application. The client capabilities allow design, debug and deployment of designs to a large set of remote FPGA nodes. With the projected hundreds of nodes as explained in Sect. 4, each with a USB-UART in addition to a USB-JTAG, it is impossible to manage and control the system with just a single server (as in FSC 1.0) so the server functionality has been decomposed into a Central Server Unit (CSU) and multiple Distributed Server Units (DSUs). The CSU is the portal through which clients can access the nodes while the DSUs are wired to the FPGA nodes and directly interact with them. When a client initiates a call using the FSC protocol, the CSU validates the call and either handles it locally or forwards it to the appropriate DSU. As shown in Fig. 2, the FPGA nodes, CSU and the DSUs are all connected by an Ethernet switch. Typically, the Ethernet switch resides behind a firewall to protect a system from unauthorized access.

Packet router One of the pivotal factors that went into realizing *Spirit* is the design of the network. Toward that, an integrated on-chip and off-chip network was established—AIREN (Architecture Independent Reconfigurable Network) [24]. This network was designed in a way that it has the ability to support node-to-node communication as well as core-to-core communication. The most important component of the AIREN network is the AIREN router. The AIREN Router comprises of a crossbar switch in addition to routing modules. A distinct feature of the AIREN router is that the crossbar switch has configurable radix and data widths. This configuration is achieved by the switch controller module. Routing decisions are made using a dimension ordered routing algorithm. The block diagram of the AIREN router is as shown in Fig. 3. In addition to the router and the switch controller, the AIREN network consists of an AIREN interface. The AIREN interface uses the LocalLink standard to connect with the network. The plot of the measured bandwidth versus the message length is as shown in Fig. 4.

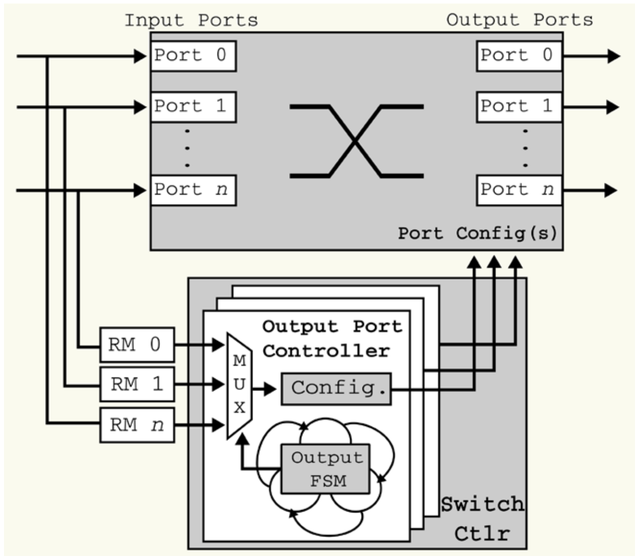


Fig. 3 Block diagram of the AIREN router

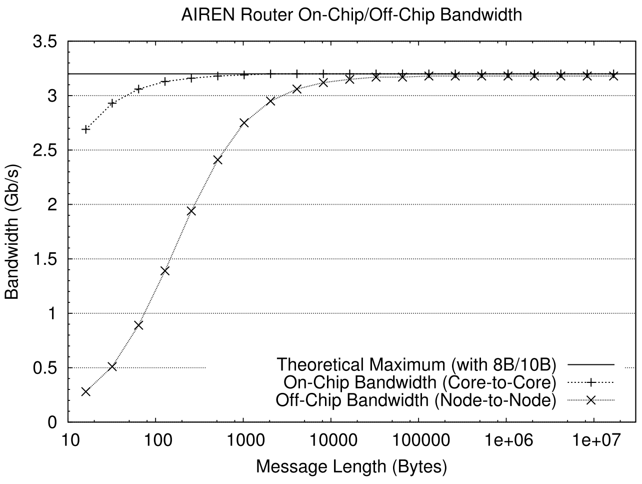


Fig. 4 On-Chip and Off-Chip bandwidth of AIREN network with hardware based routing

The AIREN network was also used to increase the productivity of I/O bound streaming applications and maintained the results when these applications were scaled. The results demonstrated linear speedup across numerous cores when the applications were scaled up to 512 cores on the Spirit cluster. Detailed results can be seen in [22].

3.2 Applications

RC-BLAST For a long time, the primary role of FPGAs has been to provide as accelerators. One of the applications developed in the RCS lab was to leverage FPGAs to serve as accelerators for Basic Local Alignment Search Tool (BLAST). BLAST is one of the most widely used bioinformatics tools today. It is a tool that performs sequence matching for DNA analysis. BLASTn runs nucleotide based queries over existing databases and calculates the significance of the matches. This is basically an I/O bound streaming operation. The hardware developed in our lab, implemented accelerated Scan and Ungapped Extension functions which were based on the National Center for Biotechnology Information's (NCBI) software implementation for the same. This approach produced the same results as the NCBI implementation. The block diagram of the hardware is shown in Fig. 5.

Experiments were run with variable number of cores and different Query Set sizes and significant speedup was observed. This can be observed in Figs. 6 and 7. While detailed results can be found in [22], it was found that the speedup was most pronounced for the largest query set (12,216 bytes).

DBES Ad hoc Mobile networks have been exhaustively studied via simulation. This has resulted in a very complex set of protocols that modern radios use. Unfortunately, when radios from different manufacturers do not behave correctly together, it is difficult to determine the source of the problem. Moreover, to develop better protocols, emulations with fewer assumptions and more realistic data will be necessary. Towards this end, an application is being developed that allows a cluster of FPGAs (augmented with an analog/digital mezzanine card) to recreate real world scenarios that can exercise commercial radios in real-time [5].

3.3 OS HW/SW co-design

A number of projects that are unique to this unconventional architecture are related to refactoring the operating system or system libraries. That is, these projects explore the effects of migrating software functionality from a library (such as MPI) or the OS (such as the filesystem) into hardware cores.

Hardware file system The block diagram of the hardware file system core developed is shown in Fig. 8. The hardware file system is based on the UNIX filesystem but has several distinctions as explained in [13]. This HWFS implements Open, Read, Write and Delete file functions. Various tests were conducted using different block sizes and the best performance was observed for a 1024 bytes block size which was found to consume just 7 % of the slices on a Virtex 4 FPGA board.

Collective communications The construction of a 64-node FPGA cluster like *Spirit* called for some kind of support for a communication mechanism. Toward this, support of MPI type collective communication tasks was enabled that helped in escalating the successful use of the cluster. Hardware implementations of collective communication operations like MPI barrier [9] and MPI reduce [10] were success-

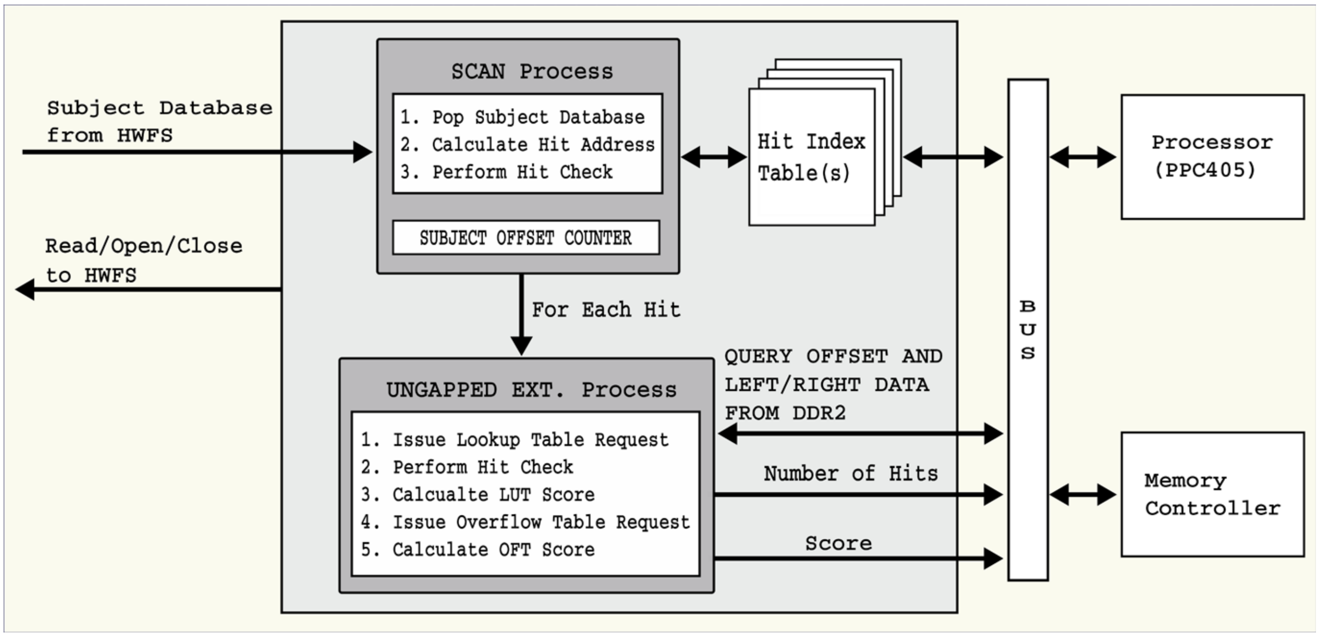
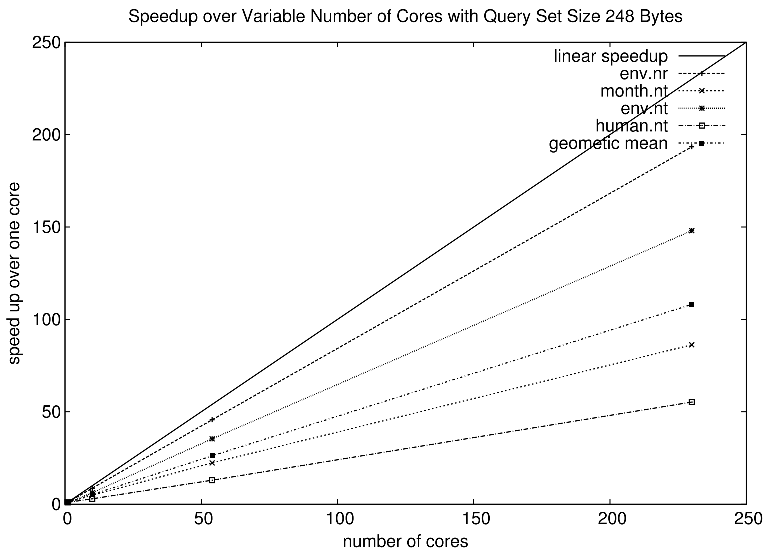


Fig. 5 Functional overview of the BLAST core

Fig. 6 Speedup observed for a query size of 248



fully rendered. Also, point-to-point implementations such as MPI send and MPI receive were implemented. These designs were tested not only on our FPGA cluster but also on a commodity cluster. The results for MPI Barrier proved to be more significant than the conventional software implementation. The MPI reduce design demonstrated a speedup of ≈2× to ≈800× while compared to a commodity cluster for small datasets.

These designs were also tested along with the AIREN network interface [23]. Different topologies were tested and the average time to complete a barrier was recorded as shown in Fig. 9. Figure 10 shows the measured MPI_reduce performance on the cluster for different topologies.

3.4 System resilience

A third class of projects has focused on using the machine to explore future architectural issues. For example, the scale and reliability of future machines have suggested that they will be required to tolerate a much higher rate of transient errors.

Fault injection In a short project, the reconfigurable fabric of the FPGA was used to introduce two types of additional hardware cores. The first type was designed to intentionally introduce interference in the system. For example, one core would sit on the bus and periodically (at a programmed rate)

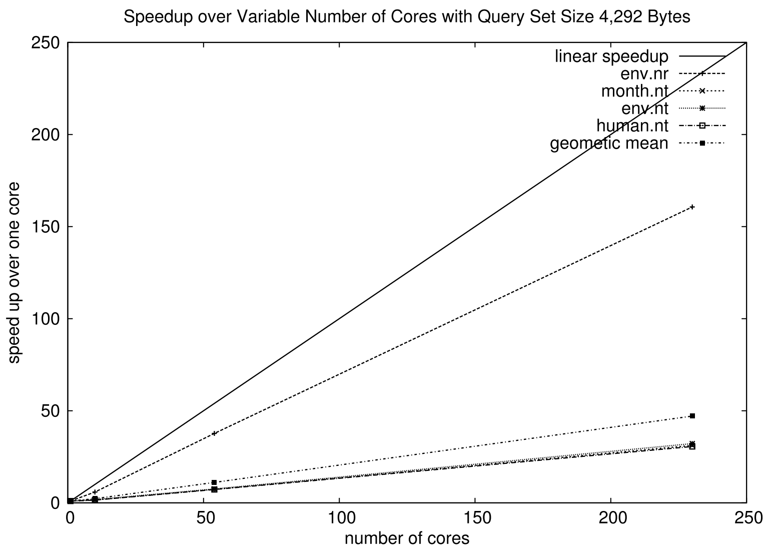


Fig. 7 Speedup observed for a query size of 4292

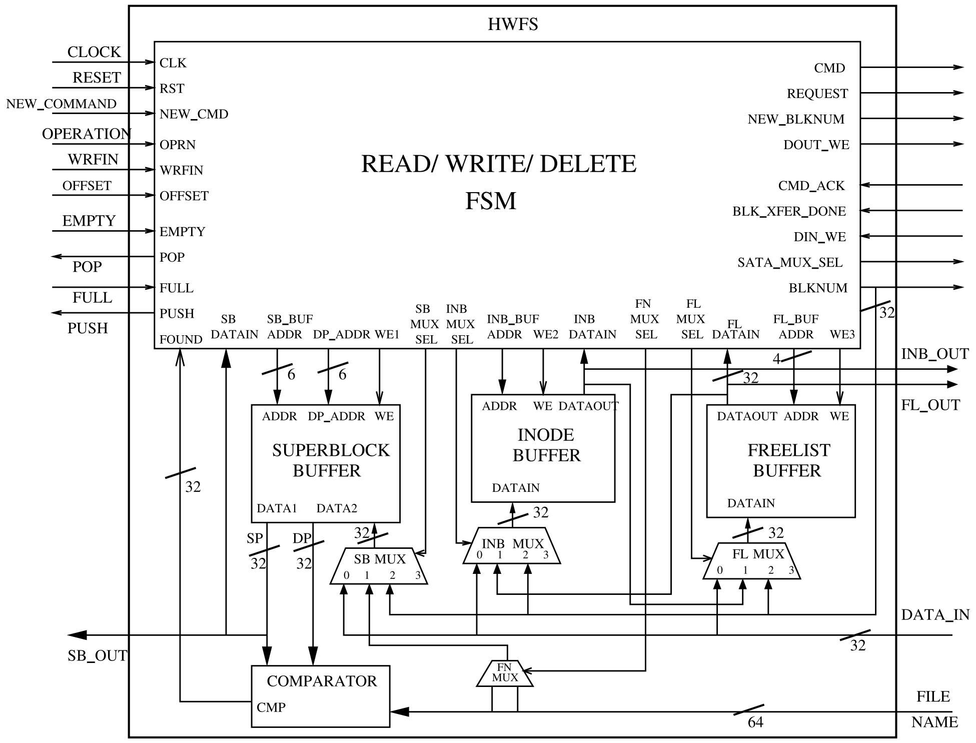


Fig. 8 Hardware Filesystem block diagram

request arbitrary bus transactions. Other cores of this type injected transient errors in the system (at the network or in RAM). The other type of core was designed to observe the behavior of the system. The idea of this project was to determine if subtle changes would lead to systemic degradations and, if so, could the situation be detected [21].

Performance monitoring and checkpoint/restart FPGA designs consist of a lot of intricacies that often tend to stump even experienced engineers. The Hardware Performance Monitoring Infrastructure (HwPMI) [25] tool is designed to ease these complexities during the design and debug stage. This tool has evolved significantly in the past year. The initial design started out as a mechanism to detect a node failure. Then, along with improving the System Monitoring infrastructure, dedicated hardware cores were added that provided *checkpoint/restart* capabilities. This was enabled by augmenting a character device driver to the system monitor hardware. The HwPMI tool collects status information in order to detect faults, has the ability to recover and restore context for checkpoint/restart, and can help in improving performance by detecting stalls.

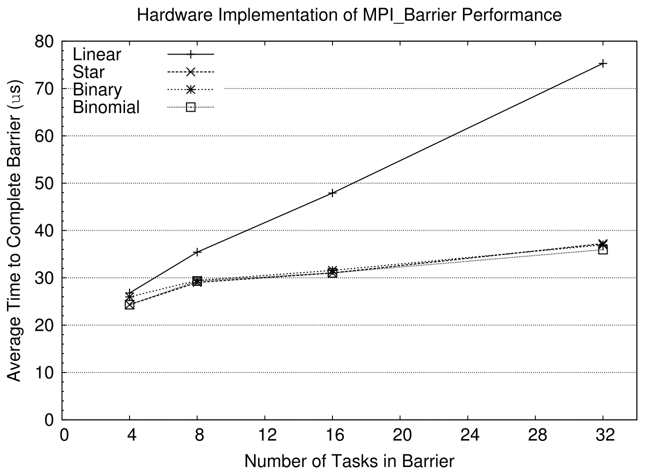


Fig. 9 Measured `MPI_Barrier` performance of different network topologies on *Spirit*

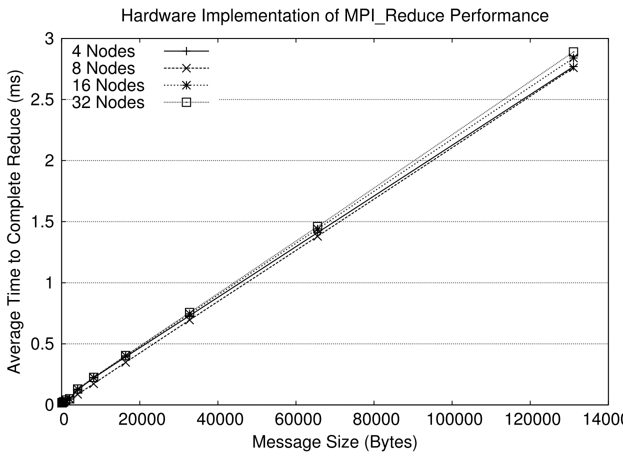


Fig. 10 Measured `MPI_Reduce` performance of different network topologies on *Spirit*

To inspect the behavior of the HwPMI, a 8 × 2 Multiply and Accumulate (MAcc) was realized and HwPMI was inserted into its hardware. A larger MAcc array 16 × 2 was developed first and tested on a Fast Ethernet system. It had to be downsized to include eight network channels for the AIREN network. The performance achieved for a 16 × 2 Multi-node Matrix Multiplication (MMM) system and 8 × 2 MMM system is shown in Fig. 11.

4 Path to exascale

Going forward, this unconventional architecture may have a larger role to play in both the development of and the realization of extreme scale computing machines. Below, we describe the challenges of designing an Exascale computing machine and then suggest that this architecture has the ability to play a role in the development of such machines.

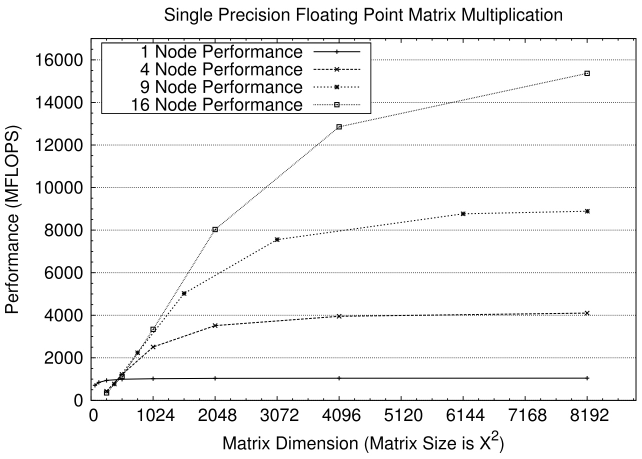


Fig. 11 Performance of a 8 × 2 MAcc array on the AIREN network

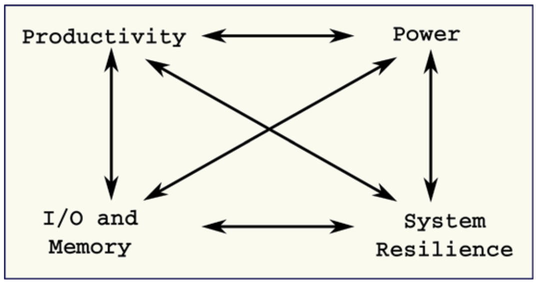


Fig. 12 Challenges in tension

4.1 Abstract machines

In the past, there was a virtuous circle where computer architects could focus on a single core microprocessor: advances in node performance led to scalable, COTS parallel computing which in turn generated more demand for (single core) microprocessors. This cycle was broken when power concerns slowed the rate of increase in clock frequency and abundance of transistors resulted in explicitly parallel, multi-core devices. As a result, the goal of reaching Exascale computing faces a multi-faceted set of challenges. It varies depending on the report, but several studies have identified four core challenges moving forward: power, programmability, the primary and secondary storage hierarchies, and system resilience. Previously, it was possible to advance these challenges individually but, in the future, this will become more difficult because these challenges are in tension. Figure 12 graphically illustrates this. For examples, one cannot propose advances in programmability if it dramatically increases power. Similarly, many techniques that improve system resilience decrease programmability or put additional stress on the I/O subsystem.

Additionally, one cannot increase system resilience by asking programmers to incorporate ever finer-grain checkpoint/restart mechanisms into their application code because the programmability challenge is so significant. Similarly, using automated checkpoint/restart techniques requires significantly more I/O bandwidth and consumes energy that does not directly contribute to the science question at hand.

As mentioned earlier, computer architecture research has focused narrowly on the microprocessor. As a key component of the "Beowulf-style" (commodity components and Open Source software) computer, this research benefited the high-performance computing sector. Indeed, in 2009, more than 80 % of computers on the TOP500 list fit this category and PetaFLOP (10^{15} operations/second) machines exist today. However, it is clear from numerous, exhaustive reports that the path from PetaFLOP to Exascale will be dramatically different [16].

As a result, the methods of computer architecture research will need to change. In the past, researchers could start with an existing micro-architecture, propose an innovative change, and then—using a cycle-accurate software simulator—test a set of benchmark applications. If the change resulted in faster execution, then the hypothesis was affirmed. Future chips are going to be much more complex with many heterogeneous cores, complex communication paths, and memory characteristics. Moreover, the node architecture of Exascale machines will be far more complex which means that one cannot simply simulate a single core in isolation and then try extrapolate system performance.

Hence, many computer architecture researchers are turning to FPGA-based technology to *emulate* integrated chips or whole nodes. In doing so, the hope is to increase the speed of experiments, even as the number of cores increase. However, we argue in this article that simply using the old paradigm with FPGAs will be ineffective. As the reports cited suggest, the chip and system architectures will be dramatically different and studying these architectures in an evolutionary way will not achieve much. What is needed is the ability to (a) *rapidly* implement novel chip and system architectures on multiple FPGA devices and (b) easily collect *actionable* information from the emulation of the system. This means computer architecture investigations will need a strong suite of research tools to effectively use multi-FPGA systems. Some researchers refer to this as implementing an abstract machine and this is an increasingly important area that is a very good fit for this type of architecture.

4.2 Proposed exascale research tools

Recently, the RCS lab has been investigating the role of a cluster of FPGAs as an abstract machine. Based on earlier work on system resilience, it was observed that 64 nodes is too small for even a scale model. This has motivated us to propose building a 625-node FPGA cluster, organized as a 5-ary 4-cube. The idea is to have custom AMC-FPGA cards that will have FPGA Mezzanine Connectors (FMC) that will eventually be used for a storage subsystem (SSD) card. These FPGA nodes will be contained in standard MicroTCA chassis that can hold up to twelve FPGA cards. The plan is to populate one rack with 125 nodes, and thus five racks are required to build the entire machine.

The RCS Lab is currently working toward building a smaller scale prototype of the 625-node cluster previously described. The main aim of this machine is to support a digital battlefield environment simulator application as explained in Sect. 3.2. The main components that have been used to build this cluster are described in detail below.

AMC boards The FPGA nodes are Nutaq Perseus 6113 AMC boards. They are double-wide and mid-size advanced mezzanine cards. The boards include a Virtex-6 Xilinx FPGA and two high-pin-count FPGA Mezzanine Card (FMC) sites. Additionally, support for RTM expansion is also included. The cards use the Xilinx Virtex-6 SX475 part. In the larger cluster, the Xilinx Virtex-6 LX550 part will be preferred because of the additional resources. The boards also have a DDR3 SODIMM interface that enables upgrades to system memory. These boards have an Intelligent Protocol Management Interface (IPMI) controller and IPMI and FPGA JTAGs for easy debugging.

Rear transition modules (RTM) The Rear Transition Modules house the SSD cards. It is also instrumental in establishing dedicated communication links between multiple FPGA nodes. The RTM provides different mechanisms (PCIe, SAS, SATA, etc.) to move data between nodes and/or from external components. Mini-SAS cables are used to connect multiple FPGA nodes. The AMC boards can route 32 Gigabit High-Speed Transceivers in groups of four over eight mini-SAS HD connectors using the RTM. A total of 20 Gbps network throughput can be achieved to and from each FPGA through the mini-SAS HD connectors. A simple ping-pong test was done using two AMC boards and two RTMs and it was observed that the interconnect adds 132 nanoseconds latency for communication from one AMC board to another.

Analog radio cards In order to accomplish radio research with the proposed machines, Nutaq Radio 422x FMC cards are required. This is a transceiver module that supports broadband coverage as well as TDD and FDD full duplex modes of operation.

Radio peripherals To establish radio communication, we require Universal Software Radio Peripheral (USRP) devices. The USRP N210 has the flexibility to support the implementation of custom functions in the FPGA or the onboard RISC softcore. This opens up different possibilities for new and exciting research. Another factor in choosing the N210 kits over their predecessors (USRP N200) was that these kits have the processing capability of 100 MS/s in both directions whereas the older kits had only a 50 MS/s streaming capability.

Chassis To hold the AMC boards, a μ TCA chassis is required. This is a 8U μ TCA .4 chassis that provides 12 slots to hold the AMCs and the RTMs described above. The chassis supports dual Micro Carrier Hubs (MCHs), dual cooling units and quad power modules. The chassis has the provision to route cables from the front to the back that enables easy connectivity.

AC-DC power supply Each of the chassis described above contain two power modules that can take up to 800 Watts of power. To power the machine we need to provide up to 1600 Watts of power to each chassis, and we have six chassis in total. Therefore, a DC power plant that takes in 208 Volts AC and has seven power bays that accept 2000 Watt modular rectifiers is chosen for this cluster. This DC Power plant has the capability of SNMP monitoring and maintaining logs, both of which are very crucial for managing such a large system.

Head node/server In order to achieve Digital Battlefield Environment Simulation, several simulations of different scenarios need to be calculated. This can take on the order of 24 hours to perform all of the calculations for a scenario on a typical workstation. To improve productivity, this bottleneck is eliminated by performing the calculations on a dedicated high performance server. This will reduce the calculation time down approximately an hour per scenario. This server will also need a large amount of disk space as the output from each scenario is several gigabytes. The Server is also used for the administration and management of all the FPGA nodes.

4.3 Evolving projects

Outlined below are some evolving projects in areas that will have a huge impact on Exascale research.

Network One of the main challenges toward scaling a system is to provide a stable and resilient network. Toward that, the RCS lab built a Radix Tree Router [12]. This router is intended to provide flexibility to include irregularities in the network. This router uses the 32-bit IPv4 addressing scheme to enable ease of portability. Each node will have a radix tree table which will contain the next hop information for that node to all other nodes in the network. This tree is generated in software, and this software is run at boot time on each node. Each time a node failure is detected, this software can be run to generate a new radix table. The time to build and update the tree is 13.93 µS. A newer version of this router is in the works, which has more sophisticated features to route around failed nodes and better mechanisms to deal with multi-node failures.

Memory There has been a steady increase in the density of the transistors used on a chip and with time, these transistors are faster. Unfortunately, the progression of memory technology is not on par with the ever-increasing clock rates. This has caused memory latency, measured in processor clock cycles, to grow dramatically and in the next decade it is expected to approach 1000 clock cycles to access off-chip DRAM [16]. In an effort to address this issue, we began investigating ways to improve memory bandwidth and reduce memory latency. This led to the implementation of a Dynamic Memory Allocation Controller (DMAC) [18]. The DMAC is a *message-store* unit that can handle variable number of variable-sized memory segments, independent of any software process or OS involvement. A binary tree data structure was implemented in hardware to hold the metadata of the system. The simplicity of a binary tree structure on an average, enabled a small system with relatively fast access but it also led to increased access times in the case of a bad data pattern. Current investigations include implementation of a balanced binary tree, and a caching policy to make the hardware more efficient.

Tools This paper presented a couple of tools (FSC and HwPMI) that enable deploying and instrumenting abstract machines on our cluster of FPGAs. While the tools may prove to be useful in other settings, we are particularly interested in addressing the needs of researchers investigating novel architectures as opposed to the more conventional use of FPGAs for embedded systems or computer-accelerator roles. As such, we do not see these tools replacing the commercial tools for designing hardware. Rather, the flow here involves using existing cores and combining them into systems with high-level entities. At this point, the HwPMI tool has the ability to automatically instrument these designs to collect performance data and bring it out of the system at run-time. On the other hand, FSC 2.0 is responsible for moving the bitstreams to nodes, managing multiple FPGAs, and allowing multiple users to share a collection of FPGA nodes. FSC 2.0 is Open Source and is available for download at Source Forge [19].

5 Related work

While the cluster described in the paper is fairly unique, it is not the only architecture to include FPGA-based nodes. Over the last dozen years, a number of cluster-of-FPGA type machines have been described. The Sepia project [14] assembled multiple PAMette nodes (an FPGA board in a workstation) into a system for large scale image processing. The FPGAs in Sepia also had direct access to a high-speed network. The Adaptable Computing Cluster [27] assembled a system of sixteen nodes using ACEcard (later GRIP2) FPGA boards; the FPGAs functioned as a network interface card to a Gigabit Ethernet switch as well as a compute accelerator. This allowed the FPGA to operate on messages in transit. It was found that small operations could have significant systemic effects. More recently, systems of FPGA nodes have been assembled from BEE2 [6] and BEE3 [7] boards; however, the focus is on cycle-accurate simulation of microprocessors rather than general-purpose parallel computing. Maxwell [2], and Novo-G [11] are also well known examples of the type of systems that include a large number of FPGAs. However, the architecture of these systems is much more conventional—these are traditional commodity clusters that have been augmented with FPGA accelerators on the peripheral bus of each node.

6 Conclusion

This article presents an unconventional cluster architecture comprising of FPGA nodes connected by a custom, high-speed network. The tools and a variety of specialized hardware designs have been presented. With technology growing at an extremely fast rate toward Exascale systems, it is likely that relatively small systems (like the one presented) will grow from tens of FPGA nodes to hundreds (or maybe even thousands) of FPGAs nodes. The challenges that will arise in designing these large systems are stated and nascent research that could prove to be a stepping stone toward embracing exascale computing is described.

Acknowledgements The authors would like to thank our friends and colleagues—William V. Kritikos, Andrew G. Schmidt, Shan Yuan Gao, Ashwin A. Mendon, and Bin Huang—who advanced many of the projects summarized in this report.

This project was supported in part by the National Science Foundation under NSF Grants CNS 05-51688 (CRI) and CNS 04-10790 (EHS). The opinions expressed are those of the authors and not necessarily those of the Foundation.

References

1. Almasi, G., Chatterjee, S., Gara, A., Gunnels, J., Gupta, M., Henning, A., Moreira, J.E., Walkup, B.: Unlocking the performance of the BlueGene/L supercomputer. In: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing (SC'04), p. 57. IEEE Comput. Soc., Washington (2004). doi:10.1109/SC.2004.63
2. Baxter, R., Booth, S., Bull, M., Cawood, G., Perry, J., Parsons, M., Simpsõn, A., Trew, A., McCormick, A., Smart, G., Smart, R., Cantle, A., Chamberlain, R., Genest, G.: Maxwell—a 64 FPGA supercomputer. In: Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007), pp. 287–294 (2007). doi:10.1109/AHS.2007.71
3. Booth, S., Campbell, D., Chien, A., Lethin, R., Mullin, L., Rodrigues, A., Sass, R., Shalf, J., Snir, M., Sterling, T.: Exascale and beyond: Configuring, reasoning, scaling. Tech. rep., US Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR) (2011). http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/ArchitecturesIIWorkshopReport.pdf
4. Buntinas, D., Panda, D.K., Sadayappan, P.: Fast NIC-based barrier over Myrinet/GM. In: Parallel and Distributed Processing Symposium, International, vol. 1, (2001). doi:10.1109/IPDPS.2001.924993
5. Buscemi, S., Sass, R.: Design of a scalable digital wireless channel emulator for networking radios. In: Military Communications Conference (MILCOM 2011), pp. 1858–1863 (2011). doi:10.1109/MILCOM.2011.6127583
6. Chang, C., Wawrzynek, J., Brodersen, R.: Bee2: a high-end reconfigurable computing system. Design test of computers. IEEE **22**(2), 114–125 (2005). doi:10.1109/MDT.2005.30
7. Davis, J.D., Thacker, C.P., Chang, C.: BEE3: revitalizing computer architecture research. Tech. rep., Microsoft research (2009)
8. Eddington, C.: InfiniBridge: an InfiniBand channel adapter with integrated switch. IEEE MICRO **22**, 48–56 (2002). doi:10.1109/MM.2002.997879
9. Gao, S., Schmidt, A.G., Sass, R.: Hardware implementation of mpi_barrier on an FPGA cluster. In: Proceedings of the 19th International Conference on Field-Programmable Logic and Applications (FPL'09) (2009)
10. Gao, S., Schmidt, A.G., Sass, R.: Impact of reconfigurable hardware on accelerating mpi_reduce. In: International Conference on Field Programmable Technology (FPT'10). IEEE Comput. Soc., Los Alamitos (2010)
11. George, A., Lam, H., Stitt, G.: Novo-g: at the forefront of scalable reconfigurable supercomputing. Comput. Sci. Eng. **13**, 82–86 (2011). http://dx.doi.org/10.1109/MCSE.2011.11. doi:10.1109/MCSE.2011.11
12. Kritikos, W.V., Rajasekhar, Y., Schmidt, A.G., Sass, R.: A radix tree router for scalable fpga networks. In: Proceedings of the 2011 21st International Conference on Field Programmable Logic and Applications (FPL'11), pp. 76–81. IEEE Comput. Soc., Washington (2011). http://dx.doi.org/10.1109/FPL.2011.24. doi:10.1109/FPL.2011.24
13. Mendon, A., Schmidt, A.G., Sass, R.: A hardware filesystem implementation with multi-disk support. Int. J. Reconfigurable Comput. (2009)
14. Moll, L., Shand, M., Heirich, A.: Sepia: scalable 3D compositing using PCI pamette. In: Proceedings of the Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '99), p. 146. IEEE Comput. Soc., Washington (1999). http://dl.acm.org/citation.cfm?id=795658.795874
15. Nieplocha, J., Tipparaju, V., Krishnan, M.: Optimizing strided remote memory access operations on the quadrics QsNetII network interconnect. In: Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA '05), p. 28. IEEE Comput. Soc., Washington (2005). http://dx.doi.org/10.1109/HPCASIA.2005.62. doi:10.1109/HPCASIA.2005.62
16. Kogge, P., et al.: Exascale computing study: technology challenges in achieving exascale systems. Tech. rep. TR-2008-13, DARPA Information Processing Techniques Office (IPTO) sponsored study (2008). www.cse.nd.edu/Reports/2008TR-2008-13.pdf
17. Rajasekhar, Y., Kritikos, W., Schmidt, A., Sass, R.: Teaching FPGA system design via a remote laboratory facility. In: International Conference on Field Programmable Logic and Applications (FPL 2008), pp. 687–690 (2008). doi:10.1109/FPL.2008.4630040
18. Rajasekhar, Y., Sass, R.: A first analysis of a dynamic memory allocation controller (dmac) core. In: Proceedings of the 2011

Symposium on Application Accelerators in High-Performance Computing (SAAHPC'11), pp. 64–67. IEEE Comput. Soc., Washington (2011). http://dx.doi.org/10.1109/SAAHPC.2011.23. doi:10.1109/SAAHPC.2011.23

19. Sass, R.: FPGA session control. Accessed March 2012 (2012). http://sourceforge.net/projects/fpga-session
20. Sass, R., Schmidt, A.G., Buscemi, S.: Reconfigurable computing cluster: a five-year perspective of the project. In: ParaFPGA2011: Parallel Computing with FPGAs (2011)
21. Sass, R., Sharma, R.R., DeBardeleben, N.: Towards a hardware fault-injection testbed to support reproducible resiliency experiments. In: Proceedings of the 2009 Workshop on Resiliency in High Performance (Resilience '09), pp. 15–22. ACM, New York (2009). http://doi.acm.org/10.1145/1552526.1552529. doi:10.1145/1552526.1552529
22. Schmidt, A.G., Datta, S., Mendon, A.A., Sass, R.: Investigation into scaling i/o bound streaming applications productively with an all-fpga cluster. Int. J. Parallel Comput. (2011)
23. Schmidt, A.G., Kritikos, W.V., Gao, S., Sass, R.: An evaluation of an integrated on-chip/off-chip network for high-performance reconfigurable computing. Int. J. Reconfigurable Comput. (2012)
24. Schmidt, A.G., Kritikos, W.V., Sharma, R.R., Sass, R.: AIREN: A novel integration of on-chip and off-chip FPGA networks. In: Proceedings of the 17th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'09). IEEE Comput. Soc., Los Alamitos (2009)
25. Schmidt, A.G., Sass, R.: Improving fpga design and evaluation productivity with a hardware performance monitoring infrastructure. In: International Conference on Reconfigurable Computing and FPGAs (ReConFig), pp. 422–427 (2011). doi:10.1109/ReConFig.2011.53
26. Shah, G., Bender, C.: Performance and experience with LAPI—a new high-performance communication library for the IBM RS/6000 SP. In: Proceedings of the 12th International Parallel Processing Symposium on International Parallel Processing Symposium (IPPS '98), p. 260. IEEE Comput. Soc., Washington (1998). http://dl.acm.org/citation.cfm?id=876880.879642
27. Underwood, K.D., Sass, R.R., Ligon, W.B. III: Cost effectiveness of an adaptable computing cluster. In: Proceedings of the 2001 ACM/IEEE Conference on Supercomputing (CD-ROM) (Supercomputing '01), pp. 54. ACM, New York (2001). http://doi.acm.org/10.1145/582034.582088. doi:10.1145/582034.582088.
28. Xilinx, Inc.: Ml410 virtex-4 fx embedded development platform. http://www.xilinx.com/ml410
29. Rajasekhar, Y., Sharma, R.R., Sass, R.: An extensible and portable tool suite for managing multi-node FPGA systems. In: Proceedings of the 20th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'12). IEEE Comput. Soc., Los Alamitos (2012)

Yamuna Rajasekhar Yamuna Rajasekhar is a PhD Candidate at the Reconfigurable Computing Systems Lab at the University of North Carolina at Charlotte. She has done her M.S in Electrical and Computer Engineering at the same university. Her B.S. degree is in Electronics and Telecommunication from the University of Mumbai. She is a member of the IEEE. Her interests include Reconfigurable Computing and Advanced Computer Architecture.

Ron Sass Ron Sass is an Associate Professor in the Electrical & Computer Engineering Department and the Levine Faculty Fellow at the University of North Carolina at Charlotte. The Reconfigurable Computing Systems Lab investigates FPGA-based custom computing systems for high performance and embedded applications. Recent projects include a multiple node digital wireless channel emulator for the US Navy, single chip test articles for a DARPA project, and the NSF-sponsored Reconfigurable Computing Cluster project. He is an author of the book 'Embedded System Design for Platform FPGAs' and over 60 peer-reviewed conference and journal papers. He is a member of the ACM and a senior member of the IEEE.