Novo-G#: Large-Scale Reconfigurable Computing with Direct and Programmable Interconnects

Alan D. George*, Martin C. Herbordt†, Herman Lam*, Abhijeet G. Lawande*, Jiayi Sheng†, and Chen Yang†
*NSF Center for High-Performance Reconfigurable Computing (CHREC)
Department of Electrical and Computer Engineering, University of Florida
†Computer Architecture and Automated Design Lab (CAAD)
Department of Electrical and Computer Engineering, Boston University

***Abstract*—While High-Performance Computing is ever more pervasive and effective, computing capability is currently only a small fraction of what is needed. Three fundamental issues limiting performance are computational efficiency, power density, and communication latency. All of these issues are being addressed through increased heterogeneity, but the last in particular by integrating communication into the accelerator. This integration enables direct and programmable communication among compute components. Novo-G# is a large-scale FPGA-centric cluster being built to investigate and develop architectures, system and tool infrastructure, and applications for this model. In this report we discuss the motivation behind and particular objectives of Novo-G#, the work completed so far, the products of that work, and their potential impact. We end with a description of and an invitation to join the Novo-G# Forum, the project users group.**

I. Introduction

The overall goal of the Novo-G# project (pronounced Novo-Gee sharp) is to develop and demonstrate innovations that give the high-performance computing (HPC) user community the capability to conduct transformative research via scalable, cost effective, high-performance, general-purpose systems built from off-the-shelf components. The particular objective is to build a compute cluster and related infrastructure that facilitates computational science research and advances supporting computer systems. Specifically, Novo-G# is an enhancement of Novo-G, an accelerator-based reconfigurable compute cluster developed and housed at the University of Florida. A key aspect of the project is addressing the problem of strong scaling in high-performance computing through accelerator-centric clusters, that is, clusters where the accelerators themselves are the central components and communicate directly with one another. The specific accelerator technology we are investigating is the FPGA: they are currently the only COTS component that combines innate communication support, high-compute capability, low power, and an installed application base.

This work was supported in part by the National Science Foundation through Awards CNS-1405695, #CCF-1618303/7960, EEC-0642422, and IIP-1161022; by a grant from Microsoft by Altera through donated FPGAs, tools, and IP; by Gidel through discounted FPGA boards and daughter cards; and through tools provided by Mirabilis Design. This article is a progress report and outreach activity of the Novo-G# Community Resource project. All intellectual contributions have appeared elsewhere as cited in the text. Email: (jysheng|cyang90|herbordt)@bu.edu, (george|hlam|lawande)@chrec.org

The construction, demonstration, and dissemination of Novo-G#, together with building and organizing the associated communities, are the primary activities of this project. Novo-G# is designed to support research projects in various disciplines. These areas can broadly be classified into applications research (molecular dynamics, graph processing, bioinformatics, and artificial neural networks), and systems research (Exascale emulation, reconfigurable networks, and application-specific routing). Our goal is to reduce the barrier to entry of multi-FPGA design and enable researchers to use Novo-G# and similar systems as a platform for accelerating their research. In this report, we will discuss the motivation and particular objectives of the project, the work completed so far, the products of that work, and their potential impact. We end with a description of and an invitation to join the Novo-G# Forum, the project users group.

II. Context

A. Motivation and project overview

Together with theory and experiment, computer simulation now constitutes the third pillar of scientific inquiry, enabling researchers to build and test models of complex phenomena that either cannot be replicated or would be prohibitively expensive to be replicated in the laboratory. Applications range from the practical, such as designing more efficient aircraft and effective drugs, to basic research in understanding the molecular basis of diseases such as Alzheimer's. Yet computing capability is currently only a small fraction of what is needed: e.g., detailed biological simulations are limited to small numbers of macro-molecules; additional factors of millions are needed to simulate cells and far more than that for larger structures.

Three fundamental issues limiting performance are computational efficiency, power density, and communication latency. All of these issues are being addressed through increased heterogeneity, but the last in particular by integrating communication into the accelerator. This integration enables direct and programmable communication among compute components. Direct links enable the bypassing of CPU, network interface, and even device memory. Programmable communication enables data transfers to proceed with high efficiency even under substantial loads.

The Novo-G# infrastructure consists of the physical system and hardware, but also software and configurations, existing and under development, to enhance both general usability and the enabled research projects. Another aspect of this infrastructure, as with the Novo-G, is the community of collaborators (the Novo-G# forum) who are contributing applications, tools, evaluation, and feedback.

The end-goal of this project is to advance the capabilities of scientific computing. The nearer-term goal over the next few years is to provide a system testbed for transformative research in a variety of areas in Computer Science and Engineering including programmable network components, processor/network interfaces especially for accelerators, FPGA-based systems, applications in reconfigurable computing, architecture of clusters with direct and programmable communication, and libraries and tools to support such clusters.

B. *Previous and current work*

As of 2016 we are in an era where large FPGA clusters are not yet commonplace—outside niche areas such as circuit simulation, high-frequency trading, and security—but have been well-studied with perhaps dozens of examples.

The following is a by-no-means exhaustive list; we have somewhat arbitrarily begun this survey from 2004. An early FPGA cluster with 96 FPGAs was the Heterogeneous HPC computer (HHPC) at the Air Force Research Laboratory/ Information Directorate (AFRL/ID) Distributed Center [1]. Large research systems were also built at the University of Edinburgh (Maxwell) [2], UNCC by the Sass Group (RCC) [3], and Imperial College (Cube) [4] and (AXEL) [5]. COPACABANA is a large cluster of FPGAs built and commercialized for cryptanalysis [6]. The RAMP series of projects used FPGA clusters, e.g., for circuit simulation [7]. IBM has also been using FPGA clusters for circuit simulation [8]. Compared to the above, Novo-G# uses high-end Stratix V FPGAs and a 3D interconnect that can provide an aggregate bandwidth of 240 Gbps to and from each accelerator. Furthermore, we aim to upgrade Novo-G# in the near future with a new generation of FPGAs that will also feature hardware floating-point units.

Some FPGA cluster projects have had an emphasis on exploring issues with direct FPGA-FPGA communication, including RCC [3], and by a group at UCSD [9]. Bluehive is an FPGA cluster with direct communication built for modeling ensembles of neurons [10]. The Grape series of ASIC- and FPGA-based clusters has targeted N-Body and Molecular Dynamics most recently with the MDGrape-4 [11]. Other projects have investigated issues in FPGA-based communication including work at Toronto (TMD-MPI) [12] and Los Alamos (PetaFlops Router) [13]. In general, the use of FPGA in commercial routers is well-known [14] with Arista recently giving access to the internal FPGA [15]. NetFPGA is an educational platform for Gigabit switching and routing [16]. Novo-G# shares many of the advantages of these systems, but is also designed to be flexible at all levels of the communication and application stacks. With Novo-G# we aim to explore existing communication IP, network architectures, and application-aware communication protocols, therefore we have designed the system to be as reconfigurable as possible.

More recently, the Catapult system [17], deployed in their Cloud by Microsoft, encompasses 1632 servers with one mid-sized FPGA per server. FPGAs are interconnected directly through their transceivers in 6x8 tori. Catapult has been demonstrated to be cost-effective in a non-traditional application for FPGAs, page ranking, and also for security and deep learning. Extensive work was done to demonstrate not only performance, but holistic cost-effectiveness: connectivity, resilience, form factor, power, and cooling. Being an experimental system, Novo-G# is better suited for exploring cluster architecture as it is not wedded to any particular application or system integration.

As a major goal of the Novo-G# project is integrated communication and computation, we very briefly mention some of the prior work in that area. Certainly prior to the microprocessor, especially in the SIMD arrays of the 1980s, there was little distinction between the two. Dally's J-Machine (and its successors) was premised on communication latency of just a few cycles [18]. Physicists have long been building such systems for QCD, including the QCDOC [19], which was the basis of the first BlueGene [20]. Anton (now Anton 2) integrates communication and computation for Molecular Dynamics [21].

III. System

A. *Background and overview*

Novo-G [22], [23] began in 2009 as an effort to create a research cluster using high-density FPGA boards to accelerate scientific applications. The original machine began with a head node and 24 Linux servers, each featuring a quad-FPGA board from Gidel [24], for a total of 96 Altera Stratix III E260 FPGAs. Over subsequent years, the machine has been upgraded annually and now stands at 192 Stratix III E260 FPGAs in 24 servers, 192 Stratix IV E530s in 12 servers, 64 Stratix V GSMD8s in 16 servers, and a second set of 64 Stratix V GSMD8s in 16 servers under construction. Each server features dual Intel Xeon multicore processors. Server connectivity is provided by gigabit Ethernet and DDR/QDR InfiniBand within the system, and a 10 Gb/s connection to the Florida LambdaRail.

At CHREC (NSF Center for High-Performance Reconfigurable Computing), Novo-G has been used for a variety of application acceleration projects from the domains of bioinformatics [25], image processing [26], and financial computing [27]. The common factor among the above applications is that they are embarrassingly parallel and can therefore scale almost linearly with the available hardware resources. A greater challenge is accelerating communication-intensive applications like Molecular Dynamics. Traditionally, such communication makes use of centralized networks such as Ethernet or InfiniBand, and entails multiple interactions between the FPGA and the host. The increased latency, and communication bottlecknеck in such applications emphasize the need for a better solution.

The Novo-G# system [28], which has been under development for the last two years, features high-density Stratix V FPGAs connected by a 3D torus network that connects the FPGAs, enabling direct, low latency, and high-speed communication among the FPGAs. The hardware is supported by an easy-to-use, but efficient, protocol stack that packetizes, routes and buffers data, allowing coomunication-intensive, multi-FPGA applications to be developed rapidly. We have also recently added support for OpenCL-based, multi-FPGA apps that can also utilize the inter-FPGA communicaiton links.

B. Architecture

The Novo-G# system is part of our effort to create an FPGA cluster that can handle communication-intensive applications. In keeping with that theme, the system features ProceV boards, which are PCIe-based accelerator boards from Gidel populated with Stratix V GSMD8 FPGAs from Altera. The GS-series devices are optimized for high performance, high bandwidth applications with support for up to 36 on-chip transceivers that can operate up to 12.5 Gbaud. Each FPGA is connected to two 8GB DDR3 SODIMM and two 36-Mbit SRAM memory banks and communicates with the host CPU via PCIe v3. The FPGAs are housed in a 4U chassis with two Xeon E5-2620V2 (Ivy Bridge) processors per server. The servers themselves are interconnected via Gigabit Ethernet and QDR InfiniBand.

Our FPGA platform vendor Gidel has provided invaluable assistance by designing a custom daughterboard that allows external access to 24 high-speed transceivers. The transceivers are grouped into six bidirectional links, each link consisting of four parallel channels, enabling the construction of a 3D torus of arbitrary size. Physical connectivity between the boards is provided by a COTS CXP-3QSFP+ split cable that enables each FPGA to be connected in six different directions. Initial deployment of the Novo-G# system was completed in the second half of 2014 with 32 Stratix V boards housed in eight chassis and supporting up to a $2 \times 4 \times 4$ torus, and an upgrade to 64 nodes ($4 \times 4 \times 4$ torus) was completed in August 2015.

C. Protocol stack

A part of the hardware resources on each FPGA is used to implement a network stack that services the 3D torus network. The network stack is responsible for accepting data from the application logic, packetizing the data, routing packets across the 3D torus network by the shortest route, and delivering the data to the application logic at its destination. These functions represent a subset of the services provided by the lowest three layers of the OSI reference model (i.e. physical, data link, and network layers).

Figure 1 depicts our initial implementation of the 3D torus network stack on Novo-G#, and the network services associated with each component. The IP cores used for the transceiver interfaces are primarily supplied by Altera and interface directly with hardware resources attached to each transceiver channel. The remaining blocks are implemented as RTL code and therefore do comprise a resource overhead for each FPGA. Data generated by the application logic is packetized by the internal receiver block and stored in a FIFO. Similarly, the external receiver blocks accept packets or streaming data from the transceiver IP. One or more routers are used to route packets from the receiver to the transmitter blocks, which transmit the data further along the network or to the application at the destination node.

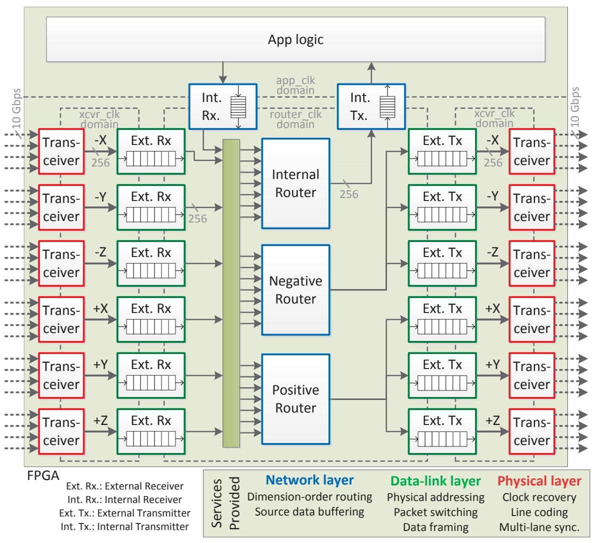

Fig. 1. Novo-G# node architecture detailing the 3D torus network stack. Services provided to the user through RTL or third-party IP are also shown.

Traditionally, the router block is the bottleneck in such network architectures and much research has been done on optimizing on-chip network architectures. In our previous work on Novo-G# performance prediction [29] we observed that the internal bandwidth provided and latency incurred by the network architecture has the greatest effect on execution time. Our approach uses a centralized single-level router network that can be customized based on hardware resources available and expected utilization of the 3D torus network. Figure 1 shows one such configuration with three routers, each servicing a different subset of output ports. In this manner more routers can be instantiated (up to the number of output ports) to increase the internal bandwidth available without increasing latency, but at the expense of hardware resources. Further optimization of the router and network design will be explored in future work.

The transceiver blocks in Figure 1 are low-level IP blocks provided by Altera that instantiate various portions of the hardwired transceiver channel. In our aim to make Novo-G# as flexible and reconfigurable as possible, we provide support for the use of the Low Latency, Custom, and Interlaken PHYs from Altera, which can be substituted without affecting the rest of the network architecture. The Low Latency PHY IP provides the lowest inter-FPGA latency, but has no word alignment or DC balancing features, making it difficult to use without additional hardware. Conversely, the Interlaken PHY IP does automatic word alignment, 64b/67b encoding, scrambling, and multi-lane synchronization, but the inter-FPGA latency is the highest, and varies due to the addition of framing words into the datastream. The Custom PHY provides word alignment through 8b/10b encoding and considerably lower inter-FPGA latency than the Interlaken PHY, but the 25% overhead of 8b/10b encoding only makes this PHY useful for small packets.

D. Alternative routing support

We now describe another of our routing mechanisms. Our goal is to create general infrastructure that supports a variety of routing modalities. For example, for a specific application, its decomposition and communication is often known *a priori* (see, e.g., [30]). Therefore the routes and timing of each packet could be optimized prior to execution, i.e., *statically* or *offline*. We adopt table-based routing to support offline routing [31], [32] including collectives [33]; we show this and the internals of the switches.

1) Routing Mechanism: Table-based routing can be implemented in two ways: source routing and node-table routing [34]. Since source routing needs to carry the table indexes along with the packets, which consumes extra bandwidth, we adopt node-table routing. The routing table preserves a table entry for each incoming packet (see Figure 2).

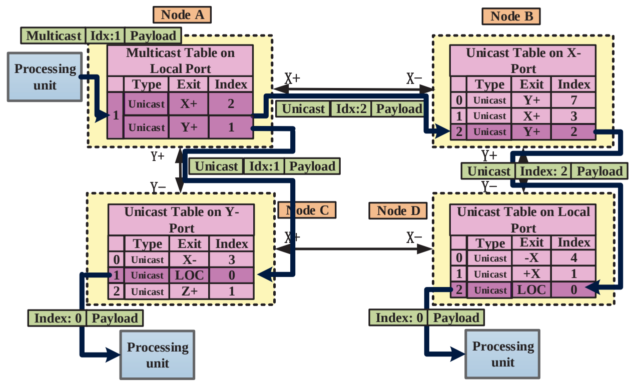

Fig. 2. The table-based routing scheme

In this example, node A dispatches a multicast packet that carries three fields: packet type, table index, and payload. The router routes the packet to either a unicast, multicast, or reduction table based on the packet type. In the corresponding table, multicast in this example, the router looks up the table entry based on the index field in the packet. The multicast table entry has slots for each of the six possible fan outs. In this example, the multicast packet has two fan outs. For the first fan out, the table entry shows that it is a unicast packet, that it should be routed to the X+ port, and that its table index for next node is 2. The router then generates a unicast type packet and routes it to the X+ port, after which it goes to X- port on node B. On node B, since it is a unicast packet, the router looks up in the unicast rather than the multicast table. The table entry shows that this packet should be routed to the Y+ port of node B. In the same manner, the router on the Node B sends it to the Y- port of Node D, at which point it is ejected. Similarly for the second fan out, the packet is routed to the Y- port of Node C, where it is ejected.

Table-based routing enables implementation of any oblivious routing scheme by simply entering values in the routing table. Examples include simple patterns, such as dimension-order routing, and complex patterns, such as certain optimal routes that achieve congestion-free communication even under heavy loads [35].

2) Switch Architecture: Our switch architecture is based on the classic four-stage pipelined Virtual-Channel switch [36]. By adding support for multicast and reduction, the four-stage pipeline is extended to seven-stage pipeline. Its architecture is illustrated in Figure 3

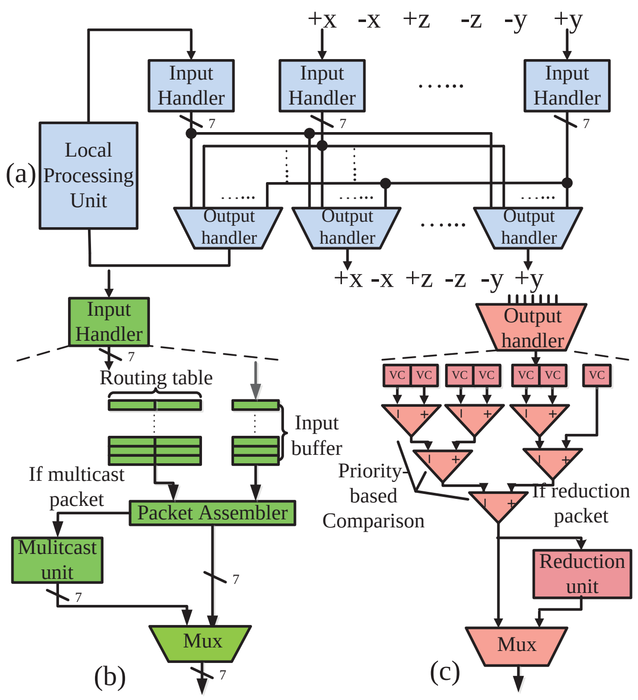

Fig. 3. Switch architecture: (a) The switch is connected by seven input handlers and seven output handlers. (b) The input handler has four stages: input buffer consumption, routing table lookup, multicast table lookup, and virtual channel allocation. (c) The output handler has three stages: switch allocation, reduction table lookup, and reduction table write-back.

In the classic switch, there are four stages: routing computation (RC), virtual channel allocation (VA), switch allocation (SA), and switch traversal (ST). The first change is that we divide RC into three stages: input buffer consumption, routing table lookup, and multicast table lookup. When a packet is injected into the switch input buffer, we spend one cycle to fetch it from the buffer. By examining its header, we get its routing table index so in the routing table lookup stage, we find its routing table entry. We then know whether it is a multicast packet or not. If it is, then in the next stage we look up its multicast table entry, based on the index from the routing table. Our switch also contains a VA stage, in which we allocate VCs to all the multicast children generated in the previous stage. If VA fails, then the router generates backpressure to stall the pipeline. In the SA stage, there might be multiple packets (up to 7) requesting the same output port. The packet with the highest priority wins the contention. The priority field is attached to each packet at the routing table lookup stage; our priority scheme is typically farthest-first.

Another change from the classic switch is that we divide ST into two stages: reduction table lookup and reduction table write-back. If the packet is not a reduction packet, it spends two cycles going through the two stages and leaving the switch. If the packet is a reduction packet, it is routed to the reduction unit, which contains a reduction table. We allocate one entry in the table for each reduction operation. In the reduction table lookup stage, the reduction packet finds its corresponding entry. It then checks whether the expected number of downstream packets has arrived or not. If not, then the reduction operation is executed, and the reduction table entry is updated. If all of the expected downstream packets have arrived, then the reduction unit dispatches a new packet and injects it into the upstream link.

IV. Sample Application

The 3D FFT is a core application across diverse scientific and engineering domains. Of particular interest to the Novo-G# project, the 3D FFT can be on the critical path for long timescale Molecular Dynamics (LTMD) simulations on large clusters [37]. Since MD has previously been shown to be highly amenable to acceleration with single FPGAs [38]–[40], demonstrating the efficiency of the 3D FFT on FPGA-centric clusters is a critical step in implementing LTMD on the Novo-G#. Due to its importance to our respective research programs, both BU [31], [32] and UF groups [29] have implemented solutions; both the approaches and results are similar.

The difficulty with accelerating the relatively small 3D FFTs used in LTMD, typically 32^3 to 128^3, is scalability. Few published results show much speed-up beyond a small number of processors; one exception is the ASIC-based Anton processor family, which was designed with this application in mind [37]. We show that the Novo-G# can perform comparably to Anton, and an order of magnitude better than the best published results for non-ASIC-based clusters.

Our previous work proposed a generalized mapping method for the 3D FFT of an arbitrary size onto FPGA clusters with arbitrary numbers of nodes. By knowing the routing information, it is possible to schedule each packet to minimize the congestion in the network. In the BU version, we have so far tested on a subset of nodes (2×2) and with a cycle-accurate simulator (for large clusters). The simulation results, and the comparisons with other technologies, are shown in Table I. The results demonstrate that even with conservative simulation settings (see [31], [32] for details), the Novo-G# 3D FFT is able to achieve the same order of magnitude latency as the ASIC implementation [37].

In the UF version, the IP cores are fed data by a Nios II soft-core processor. The processor is also connected to the internal transmitter and receiver nodes, and is responsible for aggregating FFT inputs coming in from other FPGAs, dispatching FFT data to available IP blocks, and consolidating 1D FFT outputs into packets for transmission to other FPGAs. The design has been implemented on 8 FPGAs connected in a $2\times2\times2$ network configuration and the performance measured and validated using the UF Novo-G# performance model. On scaling up the system size to 64 nodes, the model predicts a speedup of 6-19 times that of a BlueGene/Q system of the same size, for FFT sizes ranging from $32 \times 32 \times 32$ to $128 \times 128 \times 128$ elements.

V. Work in progress

In this Section we give an overview of a selection of the many projects currently under way under the direction of the authors and collaborators.

A. OpenCL

The push for better HLS tools for FPGAs in the past few years indicates that programmability and usability have become important issues for the RC community to address. Altera OpenCL (AOCL) has presented an end-to-end solution that tries to address both issues. This tool allows GPU-based OpenCL code to be readily ported to FPGAs, while also simplifying the FPGA design process and reducing turnaround time. While Altera OpenCL currently only provides support for single-FPGA design, we are in the process to extending it to support our 3D torus network, and thereby enable productive use of the Novo-G# network. This extension would also allow Novo-G# users to leverage the existing OpenCL framework and tools to verify, debug and profile their designs.

By extending the existing capabilities of AOCL on the ProceV boards, we have created a multi-FPGA OpenCL framework that allows AOCL kernels to communicate with each other within the OpenCL model. We have also established and tested an MPI-based framework to run AOCL programs on multiple OpenCL devices. Finally, we have benchmarked and evaluated the framework through three varied case studies and devised the most effective manner in which to make use of the channels efficiently, achieving a measured data rate of 24 Gbps on every inter-FPGA channel.

There are a number of improvements that can be made to this framework in the future. Adding the ability to tune the transceiver parameters at runtime could potentially allow the inter-FPGA channels to operate closer to the 40 Gbps line rate. Additionally, integrating the network-layer router from [28] would allow for transparent packet routing within the 3D torus network. Both of these improvements, however, are unlikely to fit within the established AOCL model or the OpenCL 2.0 specification and would therefore require significant changes to the AOCL hardware and runtime to support them.

B. Middleware integration

Another approach is to provide middleware support for FPGA developers. Again, FPGAs offer attractive power consumption, flexibility, and performance compared with CPUs and GPUs and much lower cost than ASICs. But programmability and portability are still obstacles: it is often challenging

TABLE I
RESULTS FOR VARIOUS TECHNOLOGIES AND PROBLEM SIZES

Implementation Technology				Perf. in μs		
Model	Parallelism	Date	Code	16^3	32^3	64^3
			2005 era technology			
Blue Gene/L	512 or 4096	04/Q4	FFTW	NA	100	200
			2008 era technology			
Intel Nehalem	4 cores	09/Q1	MKL	38	116	983
NVIDIA Tesla	240 SPs	08/Q3	CUFFT	54	66	257
Altera StratixIII	single FPGA	08/Q2	report	4.5	NA	NA
DE Shaw Anton	512 PEs	08/Q3	report	NA	4	13
			2012 era technology			
Intel Sandy Bridge	8 cores	12/Q1	MKL	22	55	288
NVIDIA Kepler	2688 SPXs	12/Q4	CUFFT	25	29	92
Xilinx Virtex-7	single FPGA	12/Q2	report	3.6	21	216
Altera StratixV	64 or 512 FPGAs	12/Q3	report	1.63	2.24	6.72

Anton is fixed point, other results are for single-precision floating point. All times are in microseconds. Release dates are from corporate announcements of availability in quantity. Stratix V times are our simulation results calibrated with our prototype cluster. Anton results are from [37]. Blue Gene/L results are from [41]. All others are from our previous work [32], [42].

to get a complex application working on a single FPGA, let alone a large FPGA cluster. The LEAP environment from Intel [43] is one of several active projects that builds an FPGA OS that allows users more focus on algorithm level while retaining full flexibility in the design. LEAP builds on the idea of latency-insensitive channels (LI) as the primary communication primitive to abstract the communication inside one FPGA and among multiple FPGAs.

Current work by one of the authors (Sheng) has the goal of providing a similar interface that hides the low-level details in inter-FPGA communication, but includes flexibility in modes of synchronization, timing, and handshaking. The link should be as transparent as a UNIX-style pipe to users. Our current progress, described in Section 3, has already facilitated part of this goal. The difference is that the latency is not insensitive; for HPC applications, we do care the latency and so we would like the latency to be manageable by users. One solution is to provide several communication link abstractions that provide different latencies and QoS. For example, a heavyweight abstraction ensures lowest error probability, but with longest latency, while a lightweight abstraction provides the lowest latency but the QoS is not 100% reliable. In the latter case, some error correction at a higher level would be necessary. Another project focuses on portability by integrating the higher level routing mechanism (e.g., the global communication described above) with LEAP or other layer that facilitates abstract pipes.

VI. OUTREACH: THE NOVO-G# FORUM

Given recent advances, such as the introduction of FPGAs into the cloud [17], we are entering a time where clusters such as Novo-G# will be ubiquitous. We have organized the Novo-G# Forum based on the highly successful Novo-G Forum model, which had over 20 members across four continents, as a mechanism to foster participation and collaboration. The Novo-G# Forum is comprised of an international group of academic researchers and technology providers working collaboratively on applications and tools to establish and showcase advantages of reconfigurable supercomputing at scale with unprecedented levels of performance, productivity, and sustainability.

Faculty and students in each academic research team can contribute innovative applications and/or tools research on the Novo-G# machine based on their unique interests. This commitment will consist of four steps. 1) One or more FPGA boards of the type in Novo-G# will be procured for local research use via help from Altera (donated or discounted devices) and Gidel (discounted boards). 2) One or more promising applications and/or tools activities will be completed and optimized for maximum speedup (versus a common CPU reference) on the local board(s). 3) The preceding achievements will then be moved and scaled onto multiple nodes in the Novo-G# machine. 4) Finally, applications successfully optimized for Novo-G# will be measured (in terms of performance, productivity, and sustainability) and compared versus conventional supercomputers to showcase overall achievement. Each technology provider will provide equipment and/tools (as donation or at discounted cost) to the academic research teams for this effort, including a reasonable level of technical support for their products, and directly participate in forum activities and discussions as they choose.

REFERENCES

[1] K. Tomko, "Feasibility of FPGA co-processor acceleration of FDTD codes," High Performance Computing Modernization Program, Tech. Rep. GSA Contract No. DAAD05-01-C-0033, 2004.

[2] R. Baxter, et al., "Maxwell - A 64 FPGA Supercomputer," in *Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2007, pp. 287–294.

[3] R. Sass, et al., "Reconfigurable computing cluster (RCC) project: Investigating the feasibility of FPGA-based petascale computing," in *Proc. IEEE Symp. on Field Programmable Custom Computing Machines*, 2007, pp. 127–138.

[4] O. Mencer, et al., "Cube: A 512-FPGA Cluster," in *Proc. Southern Programmable Logic Conference*, 2009.

[5] K. Tsoi and W. Luk, "Axel: A Heterogeneous Cluster with FPGAs and GPUs," in *Proc. ACM Symp. on Field Programmable Gate Arrays*, 2010.

[6] T. Guneysu, T. Kasper, M. Novotny, C. Paar, and A. Rupp, "Cryptanalysis with COPACABANA," *IEEE Trans. on Computers*, vol. 57, no. 11, 2008.

[7] J. Wawrzynek and K. Asanovic, "Field Programmable Gate Array (FPGA) Emulation for Computer Architecture," Air Force Research Laboratory, Tech. Rep. AFRL-RY-WP-TR-2009-1281, 2009.

[8] M. Kapur, "FPGA-based acceleration platform for chip verification," Talk at RAMP Retreat, August 2008.

[9] T. Bunker and S. Swanson, "Latency-Optimized Networks for Clustering FPGAs," in *Proc. IEEE Symp. on Field Programmable Custom Computing Machines*, 2013.

[10] S. Moore, P. Fox, A. Markettos, and A. Majumdar, "Bluehive–A Field Programmable Custom Computing Machine for Extreme-Scale Real-Time Neural Network Simulation," in *Proc. IEEE Symp. on Field Programmable Custom Computing Machines*, 2012.

[11] I. Ohmura, G. Morimoto, Y. Ohno, A. Hasegawa, and M. Taiji, "MDGRAPE-4: a special purpose computer system for molecular dynamics simulations," *Philosophical Transactions of the Royal Society A*, vol. 372, no. 20130387, 2014.

[12] M. Saldana, A. Patel, C. Madill, D. Nunes, D. Wang, P. Chow, R. Wittig, H. Styles, and A. Putnam, "MPI as a Programming Model for High-Performance Reconfigurable Computers," *ACM Trans. on Reconfigurable Technology and Systems*, vol. 3, no. 4, pp. 1–28, 2010.

[13] Z. Baker, T. Bhattacharya, P. Graham, R. Gupta, J. Inman, A. Klein, G. Kunde, A. McPherson, M. Stettler, and J. Tripp, "The PetaFlops Router: Harnessing FPGAs and Accelerators for High Performance Computing," in *Proc. High Performance Embedded Computing*, 2009.

[14] J. Bolaria and J. Byrne, *A Guide to FPGAs for Communications*. The Linley Group, 2009.

[15] Arista Networks, Inc., http://www.aristanetworks.com/en/products/7100series/7124fx/, accessed 10/2013.

[16] J. Lockwood, et al., "NetFPGA - An Open Platform for Gigabit-rate Network Switching and Routing," in *Proc. IEEE Int. Conf. on Microelectronic System Education*, 2007.

[17] A. Putnam, et al., "A reconfigurable fabric for accelerating large-scale datacenter services," in *Proc. Int. Symp. on Computer Architecture*, 2014, pp. 13–24.

[18] W. J. Dally and et al., "The Message-Driven Processor: A multicomputer processing node with efficient mechanisms," *IEEE Micro*, vol. 12, no. 2, pp. 194–205, 1994.

[19] Boyle, P.A., et al., "Status of the QCDOC project," in *Proc. Lattice 2001*, 2001.

[20] D. Chen, N. Eisley, and P. Heidelberger, "The IBM Blue Gene/Q Interconnection Network and Message Unit," in *Proc. ACM/IEEE Int. Conf. for High Performance Computing, Networking, Storage and Analysis – Supercomputing*, 2011.

[21] D.E. Shaw et al., "Anton 2: raising the bar for performance and programmability in a special-purpose molecular dynamics supercomputer," in *SC '14: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2014, pp. 41–53.

[22] A. George, "Novo-G Overview," Presentation at CHREC: NSF Center for High-Performance Reconfigurable Computing, 16 June 2010, http://www.chrec.org/ george/Novo-G.pdf, 2010.

[23] A. George, H. Lam, and G. Stitt, "Novo-G: At the Forefront of Scalable Reconfigurable Computing," *Computing in Science and Engineering*, vol. 13, no. 1, 2011.

[24] "Gidel Products," retrieved on: 2015-02-25. [Online]. Available: http://www.gidel.com/Products.htm

[25] B. C. Lam, C. Pascoe, S. Schaecher, H. Lam, and A. D. George, "BSW: FPGA-accelerated BLAST-Wrapped Smith-Waterman aligner," in *2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*. IEEE, Dec. 2013, pp. 1–7. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6732273

[26] S. Craciun, G. Wang, A. D. George, H. Lam, and J. C. Principe, "A scalable RC architecture for mean-shift clustering," in *2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors*. IEEE, Jun. 2013, pp. 370–374. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6567603

[27] R. Sridharan, G. Cooke, K. Hill, H. Lam, and A. George, "FPGA-Based Reconfigurable Computing for Pricing Multi-asset Barrier Options," in *2012 Symposium on Application Accelerators in High Performance Computing*. IEEE, Jul. 2012, pp. 34–43. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6319189

[28] A. G. Lawande, A. D. George, and H. Lam, "Novo-g#: a multidimensional torus-based reconfigurable cluster for molecular dynamics," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 8, pp. 2374–2393, 2016, cpe.3565. [Online]. Available: http://dx.doi.org/10.1002/cpe.3565

[29] A. Lawande, H. Yang, A. George, and H. Lam, "Simulative Analysis of a Multidimensional Torus-based Reconfigurable Cluster for Molecular Dynamics," in *Proceedings of the International Conference on Parallel Processing Workshops (ICPP) 2014*. ACM Press, 2014.

[30] R. Cypher, A. Ho, S. Konstantinidou, and P. Messina, "Architectural requirements of parallel scientific applications with explicit communication," in *Proc. Int. Symp. on Computer Architecture*, 1993, pp. 2–13.

[31] J. Sheng, B. Humphries, H. Zhang, and M. Herbordt, "Design of 3D FFTs with FPGA Clusters," in *IEEE High Performance Extreme Computing Conference*, 2014.

[32] J. Sheng, C. Yang, and M. Herbordt, "Towards Low-Latency Communication on FPGA Clusters with 3D FFT Case Study," in *Proc. Highly Efficient and Reconfigurable Technologies*, 2015.

[33] ——, "Collective Communication on FPGA Clusters with Static Scheduling," in *Proc. Highly Efficient and Reconfigurable Technologies*, 2016.

[34] M. Kinsy, M. Cho, K. Shim, and M. Lis, "Optimal and Heuristic Application-Aware Oblivious Routing," *IEEE Trans. Computers*, vol. 62, no. 1, pp. 59–73, 2013.

[35] M. Herbordt and P. Swarztrauber, "Towards scalable multicomputer communication through offline routing," Department of Electrical and Computer Engineering, Boston University, Tech. Rep. TR2003-01, 2003.

[36] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Elsevier, 2004.

[37] C. Young, J. Bank, R. Dror, J. Grossman, J. Salmon, and D. Shaw, "A 32x32x32, spatially distributed 3D FFT in four microseconds on Anton," in *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009, pp. 1–11.

[38] M. Chiu and M. Herbordt, "Molecular dynamics simulations on high performance reconfigurable computing systems," *ACM Trans. Reconfigurable Tech. and Sys.*, vol. 3, no. 4, pp. 1–37, 2010.

[39] M. Chiu, M. Khan, and M. Herbordt, "Efficient calculation of pairwise nonbonded forces," in *Proc. IEEE Symp. on Field Programmable Custom Computing Machines*, 2011.

[40] A. Sanaullah, A. Khoshparvar, and M. Herbordt, "FPGA-Accelerated Particle-Grid Mapping," in *Proc. IEEE Symp. on Field Programmable Custom Computing Machines*, 2016.

[41] M. Eleftheriou, B. Fitch, A. Rayshubskiy, T. J. C. Ward, and R. Germain, "Performance measurements of the 3D FFT on the Blue Gene/L supercomputer," in *Proceedings of Euro-Par 2005 Parallel Processing*, 2005, pp. 795–803.

[42] B. Humphries, H. Zhang, J. Sheng, R. Landaverde, and M. Herbordt, "3D FFT on a Single FPGA," in *Proc. IEEE Symp. on Field Programmable Custom Computing Machines*, 2014.

[43] K. Fleming, Y. Jung, M. Adler, and J. Emer, "The LEAP FPGA operating system," in *Proc. IEEE Conf. on Field Programmable Logic and Applications*, 2014, pp. 1–8.