"Novo-G: A View at the HPC Crossroads for Scientific Computing"

ERSA Keynote for Reconfigurable Supercomputing Panel

A. George, H. Lam, C. Pascoe, A. Lawande, G. Stitt
NSF Center for High-Performance Reconfigurable Computing (CHREC)
ECE Department, University of Florida, Gainesville, FL 32611-6200
{george, hlam, pascoe, lawande, gstitt}@chrec.org

Abstract *- High-performance computing for many science domains is at a major crossroads. Conventional HPC is ill-equipped to address escalating performance demands without resorting to massively large, energy-hungry, and expensive machines. This paper focuses upon the principal challenges for HPC for scientific computing, why and how reconfigurable supercomputing is poised to make a major impact on accelerating scientific applications.*

Novo-G, recently fielded by CHREC, the NSF Center for High-Performance Reconfigurable Computing, is believed to be the most powerful reconfigurable computer in the research community. Novo-G is an experimental testbed that serves as the centerpiece for a variety of research projects dedicated to understanding and advancing performance, productivity, and sustainability of future systems and applications. It is also an integration point for the Novo-G Forum, comprised of an international group of academic researchers and technology providers working collaboratively on applications and tools to establish and showcase advantages of reconfigurable computing at scale via Novo-G. Results from impactful applications and highlights from research projects being adapted for Novo-G by CHREC at the University of Florida will be presented.

Keywords - reconfigurable supercomputing, reconfigurable computing (RC), high-performance computing (HPC), FPGA, scientific computing

1. Introduction

High-performance computing for many science domains is at a major crossroads and in the center of a convergence of several technology megatrends of the last decade. First, technological advances in these areas have transformed data-starved science domains into data-driven ones. One such high-profile example is the Large Hadron Collider (LHC) project from CERN (European Organization for Nuclear Research). According to CERN's Web site [1], the LHC will produce roughly 15 Petabytes (15 PB) of data annually. Another example is the Large Synoptic Survey Telescope (LSST) project, which is scheduled to see first light in 2014, to begin doing science in 2015, and be in full survey operations by 2016. It is projected to produce 30 Terabytes (TB) of data per night, leading to a total database over the ten years of operations of 60 PB for the raw data, and 30 PB for the catalog database [2]. In the domain of genomics research, today's DNA sequencing instruments are capable of producing 225-250 million DNA reads per run, resulting in output files routinely in excess of 1 TB per instrument run. In the near future, instruments are projected to produce up to 3 billion reads per run, in which the output of a single run will approach 100-fold coverage of the entire human genome. Thus, it is with increasing recognition that the discordant trajectories growing between data production and capacity for timely analysis is threatening to impede new scientific discoveries and progress in many scientific domains, not because we cannot generate the raw data, but because we cannot analyze it.

In the second technology megatrend during the last decade, conventional computing no longer can depend upon exploiting increased clock rate and instruction-level parallelism to sustain performance. Performance is now improved through explicit parallelism using multicore and manycore processors. As a result, conventional HPC is ill-equipped to address escalating performance demands without resorting to massively large, energy-hungry, and expensive machines, where designers simply throw thousands (and soon millions) of x86 processor cores at each new and demanding problem.

Lastly, reconfigurable computing (RC) is finally ready for prime time. The research community has demonstrated small-scale but exciting successes in applying FPGA-based RC technology for accelerating scientific applications. Today's (and emerging) FPGAs finally have the computational horsepower (upper hundreds of thousands logic elements, soon millions, with tens of millions of memory bits, and other built-in functions) to enable high-performance computing. The opportunity is ripe for *reconfigurable supercomputing* (a.k.a., RC supercomputing), using scalable RC systems featuring a relatively large number of leading-edge FPGAs that can be configured specifically for high-intensity data processing for each application. The NSF Center for High-Performance Reconfigurable Computing (CHREC) recently fielded what is believed to be the most powerful reconfigurable computer in the research community. This machine, called Novo-G, is an experimental testbed that serves as centerpiece for a variety of research projects dedicated to understanding and advancing performance, productivity, and sustainability of future systems and applications. Novo-G features 192 new and extremely powerful FPGA accelerators, as well as potent subsystem and system architectures.

The remainder of the paper is organized as follows. A discussion of the three principal challenges for HPC for scientific computing, *performance, sustainability, and productivity*, is given in Section II. In Section III, the architecture and features of Novo-G are presented, along with the goals and organization of a Novo-G Forum. Sections IV and V highlight Novo-G research activities that address the three challenges outlined in Section II. In particular, Section IV summarizes the initial results of the design and scaling of two cornerstone bioinformatics applications on Novo-G, demonstrating how RC supercomputing is poised to make a major impact on accelerating scientific applications. Also, the energy consumption of Novo-G is profiled, providing some insight into the sustainability of RC supercomputing. Section V summarizes some key Novo-G research activities in productivity and tools for RC application development. Finally, Section VI provides a summary with conclusions.

2. Challenges for High-performance Computing

High-performance computing for many science domains is facing challenges in escalating performance requirements, with demands for massively large, expensive and energy-hungry machines, and concomitant challenges in using these systems productively. Performance and sustainability challenges will be discussed in Section II.A, while the productivity challenge will be discussed in Section II.B.

2.1 Performance and Sustainability Challenges

Although transistor density continues to follow Moore's Law, performance improvements from transistor scaling have decreased, and as a result conventional computing no longer can depend on exploiting clock rate and instruction level parallelism. Instead, performance is now improved through explicit parallelism using multicore and manycore processors; *but at what cost?* A 2007 joint report by the Environmental Protection Agency and the Lawrence Berkeley National Laboratory [3] showed that energy consumption from data centers had doubled between 2000 and 2006 and is expected to double again by 2011. In a press release from CERN in October 2008, an estimated 100,000 processors are needed (worldwide via the Worldwide LHC Computing Grid) to handle the computing for all the LHC experiments. Today, rather than continuing to add more computers and overextending the already limited power and cooling capacity of the IT infrastructure, CERN plans to divide its on-site 4,000 servers (which contain over 30,000 processors) into about 35,000 virtual servers by the end of the year [4]. Finally, as evidence of the growing concern on computing energy consumption, a major thrust at the 2009 Supercomputing Conference [5] was *sustainability*, with *energy efficiency* having the greatest number of sessions in the conference. As computer centers, using fixed-logic devices (e.g., CPUs and GPUs), grow to tens of megawatts in power, it is clear that the limiting factor for the total cost of ownership is shifting from the cost of system acquisition and maintenance (of hardware and software) to energy-related costs to sustain the system.

Our group at CHREC has studied device characterization issues in some depth over the past few years, including the characterization of device energy consumption. We have developed the first comprehensive set of device characterization methods [6] to quantitatively compare a broad range of programmable fixed-logic and reconfigurable-logic processing devices on the market. Metrics developed to evaluate these devices include computational density (CD), computational density per watt (CD/W), internal memory bandwidth (IMB), external memory bandwidth (EMB), and I/O bandwidth (IOB).

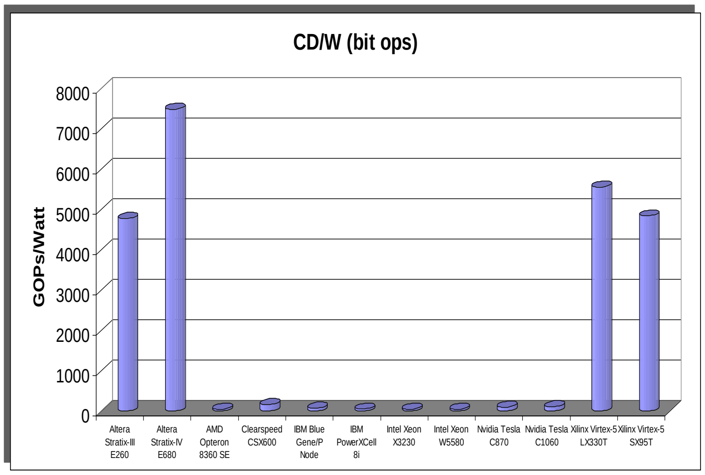


Figure 1(a) CD/W for bit operations

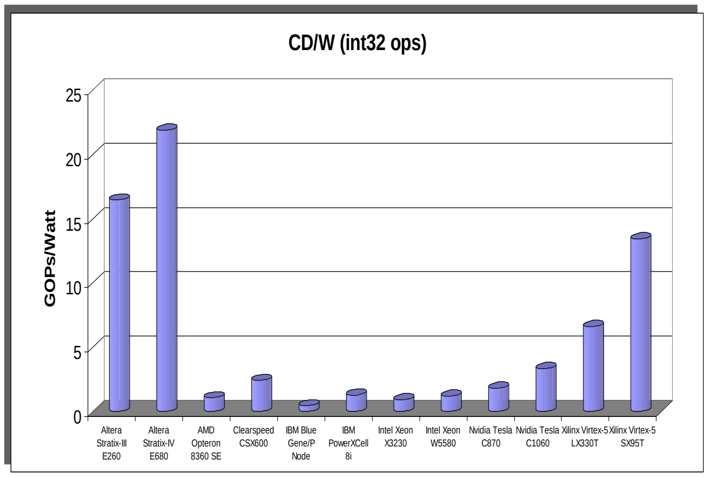


Figure 1(b) CD/W for 32-bit integer operations

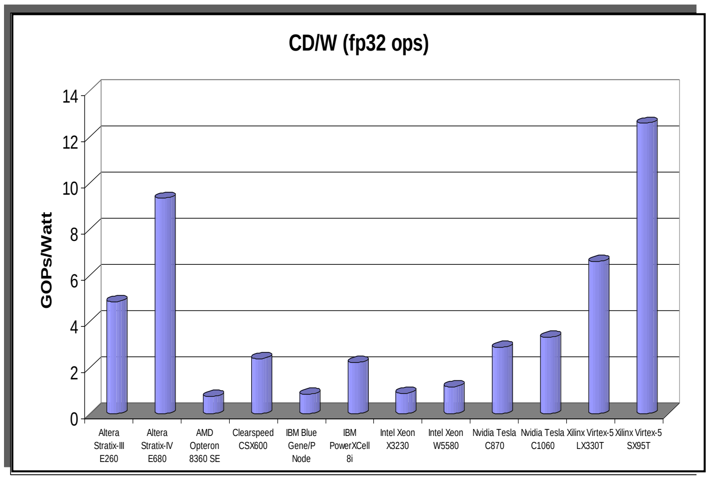


Figure 1(c) CD/W for 32-bit floating point operations

Figure 1. Computational density per watt characterization

Shown in Figure 1 are selected results for CD/W for bit-level, 32-bit integer, and 32-bit floating-point operations, respectively (Complete results can be found in [6]).

From detailed architectural analyses with these metrics on dozens of leading devices, RC device architectures were observed to inherently possess a higher degree of CD (computational density) than their fixed-logic counterparts, where both deep (pipeline) and wide (replication) parallelism can be emphasized, customized, and exploited. Moreover, RC devices run with more parallelism at lower clock speeds and thereby consume far less power, and thus were found to be inherently superior in CD/W (CD normalized by unit of energy) by varying degrees to CPUs, GPUs, Cells, etc. for all numerical formats, from bit-level operations to double-precision float and beyond.

In summary, the benefit of FPGAs for compute-intensive applications in scientific domains (e.g., genomics) comes from their reconfigurable structure. Unlike the use of fixed-logic processing devices (e.g., CPU, GPU) where applications must conform to their fixed structure (for better or for worse), with RC using FPGA devices the architecture conforms to the unique needs of each application. This adaptive nature enables FPGA devices to exploit higher degrees of parallelism while running at lower clock rates and thereby in many cases achieve better performance while consuming less energy. Thus, we claim it is the very nature of reconfigurable-logic devices that make them the superior multicore/manycore processing technology available today to process next-generation scientific data in terms of performance per unit of energy.

2.2 Productivity Challenge

Despite important advantages over conventional and alternative computing devices, reconfigurable computing, however, remains a niche technology due mainly to application design complexity that currently far exceeds conventional software design complexity. Until recently, FPGAs have been used primarily for ASIC replacement and prototyping. As a result, existing design methods and tools used for RC largely remain low-level and circuit-oriented and require hardware expertise. HDLs are generally used to specify the behavior and timing of the design in great detail, including clock-cycle accuracy of the interfaces and internal logic. Further, these systems are extremely difficult to debug and require sophisticated test equipment and/or verification methods for proper debugging. Performance analysis for RC systems is virtually non-existent outside of academic research. In addition, the compile times associated with FPGA place-and-route for large FPGAs are intolerably long for productive development.

Of course, this state of affair is not unexpected since RC is a relatively new field and the situation will no doubt improve as the field matures. Vendor products have emerged in the form of specialized high-level languages (HLLs) or electronic system level (ESL) tools to facilitate design at higher levels of abstraction. Examples of these include Impulse-C [7], Catapult-C [8], and Carte-C [9]. These tools automatically generate custom hardware circuits from a more abstract C-like specification. Other tools that aim to raise RC application design to even a greater level of abstraction are also in the mix. Simulink [10] can be used for model-based design and FPGA implementation. AccelDSP synthesis tool [11] is a high-level tool based on the MATLAB language for designing DSP blocks for Xilinx FPGAs. The LabVIEW FPGA Module from National Instruments (NI) uses LabVIEW embedded technology to extend the LabVIEW graphical development and target FPGAs on NI's reconfigurable I/O (RIO) hardware.

In the research arena, design productivity for reconfigurable systems is an important topic that has attracted the interest of many research groups, who have identified and demonstrated techniques and tools for improving RC application design productivity. We will highlight some of the on-going productivity and tools projects related to Novo-G in Section V.

Interestingly, the trend towards highly parallel computing devices, such as multicore microprocessors and GPUs, is currently facing similar productivity challenges. Consequently, mainstream designers are now forced to consider parallelism explicitly, which has recently led to a heavy focus on new languages that contain constructs for explicit parallelism [12-18]. These overlapping challenges between current mainstream application design and RC design make it an opportune time to bring RC into the mainstream.

3. Novo-G Reconfigurable Supercomputer

Novo-G (G stands for "Genesis," first of a series of Novo platforms) is an experimental testbed operational at the University of Florida since July 2009 and serves as the centerpiece for a variety of research projects dedicated to address the challenges outlined in the previous section. These projects include the development of impactful scalable kernels and applications in key science areas (see Section IV) and an innovative suite of productivity tools for application development (see Section V). The emphases for Novo-G can be summarized as *performance* (system), *productivity* (methods/tools), and *impact* (applications).

The current Novo-G platform (shown in Figure 2(a)) consists of 24 compute nodes, each a standard 4U server with a quad-core Xeon (E5520) Nehalem processor, memory, disk, etc. A single 1U server with two Xeons functions as the head node. Compute nodes communicate via Gigabit Ethernet and a non-blocking fabric of DDR InfiniBand, the latter supporting data transfers up to 20 Gb/s. Each of the 24 compute nodes houses two PROCStar-III accelerator boards from GiDEL (Figure 2(b)) via an 8-lane PCI-Express bus.

The computing power of Novo-G is derived from these PROCStar-III boards, each containing four Stratix-III E260 FPGAs from Altera. Originally designed with one board per server, a recent upgrade has increased the number of boards to two per server, with a system total from 24 (96 FPGAs) to

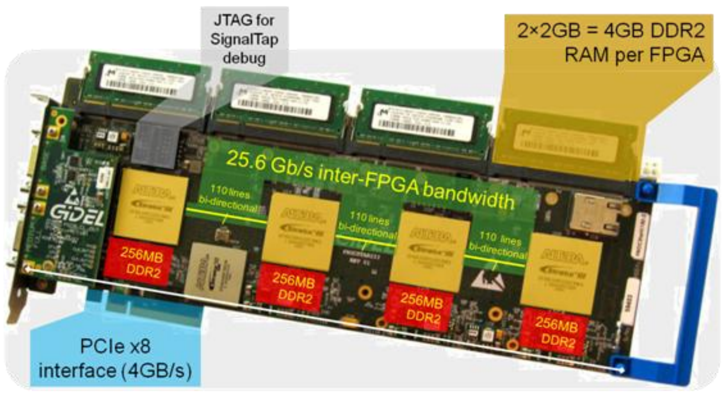


Figure 2. (a) Novo-G reconfigurable supercomputer (b) GiDEL PROCStar-III accelerator board

48 (192 FPGAs) boards. While the FPGAs are capable of providing massive computing power for the system, often it is memory or communication capacity that can limit performance if not balanced. As shown in Figure 2(b), each of the four FPGAs on each board is connected to 4.25 GB of dedicated memory. Data transfer between adjacent FPGAs can be made directly through a wide, bidirectional bus at rates up to 25.6 Gb/s and latencies of a single clock cycle up to 300MHz. Data transfer between FPGAs across the two boards on the same server can also be made directly through a high-speed cable. In the case where all I/O pins are used to send data from one board to the other, the maximum achievable clock frequency is 160MHz. If half of the I/O is partitioned to transfer data in one direction and the other half in the opposite direction, then the maximum achievable clock frequency is 250MHz between boards in each pair.

By supplying each FPGA with a large amount of dedicated memory, as well as high bandwidth and low latency for inter-FPGA data transfer, the system allows for more "self-sufficient" designs. Processing engines on the FPGAs can execute with minimal control from the software running on the host CPU, enabling the maximum utilization of computing power of the FPGAs.

3.1 Novo-G Forum

To establish and showcase the advantages of reconfigurable computing at scale, a Novo-G Forum was formed in early 2010. The Novo-G Forum is an international group of academic researchers and technology providers working collaboratively on experimental applications and tools with a common goal of realizing the promise of RC supercomputing by demonstrating unprecedented levels of performance, productivity, and sustainability.

Faculty and students in each academic research team are committed to contribute innovative applications and/or tools research on the Novo-G machine based upon their unique expertise and interests. The commitment for each team consists of four steps.

1. One or more quad-FPGA boards of the exact type in Novo-G will be procured for local research.
2. One or more promising applications and/or tools activities will be completed and optimized for maximum speedup on the local board(s).
3. The preceding achievements will then be moved and scaled onto the full Novo-G machine.
4. Finally, applications successfully optimized for the full system will be measured (in terms of performance, productivity, and sustainability) and compared versus conventional supercomputers to showcase overall achievement.

Currently committed academic participants in the Novo-G Forum include Boston University, University of Florida (CHREC), George Washington University (CHREC), Clemson University, Imperial College (UK), Northeastern University, Federal University of Pernambuco (Brazil), University of South Carolina, University of Tennessee, and Washington University at St. Louis.

4. Novo-G Performance/Sustainability

In this section, we highlight Novo-G research activities that address performance and sustainability challenges discussed in Section II. In Section IV.A, we summarize initial results of the design and scaling of two cornerstone bioinformatics applications on Novo-G, demonstrating the performance advantage of reconfigurable computing at scale. In Section IV.B, the energy consumption of Novo-G is profiled, providing some insight into the sustainability for RC supercomputing.

4.1 Novo-G Applications

The first application scaled to run on Novo-G was the well-known Smith-Waterman (S-W) application, which uses a dynamic programming algorithm to produce optimum, local sequence alignment of DNA, RNA, or protein sequences to identify regions of similarity in computational biology. For two sequences of length A and B, optimum alignment requires the calculation of A×B scores. We have

Table 1. S-W app: Novo-G speedup vs. single core

Execution Time of Serial Baseline on Single 2.4 GHz Opteron Core = 743,460 Seconds (~8.6 Days)	Number of Novo-G FPGAs in Execution					
	4	16	32	48	64	96(est)
Execution Time (Sec) of Novo-G	279	70.4	35.6	24.2	18.2	12.38
Novo-G Speedup vs. Single Core	2665	10561	20884	30721	40849	60053

succeeded in accelerating S-W targeted for Novo-G by designing novel hardware cores to exploit massive parallelism that allows up to A of the A×B scores of a single alignment to be calculated simultaneously in a single 125MHz clock cycle, reducing time complexity of this algorithm from O(A×B) to O(A+B).

Note that at the time of the implementation of the S-W application, Novo-G had only one board per node and not all 24 nodes were operational yet. The basic design for the S-W application consists of a long systolic array of 512 processing elements (PEs) per FPGA with input FIFOs for streaming of database and query sequences, and various score reporting registers for feedback to the host CPU in each node. Each PE simultaneously performs all of the calculations required to produce a single score in a single clock cycle and makes extensive use of dedicated registers to store intermediate data. Because of the high bandwidth and low latency of the interconnect between FPGAs on each Novo-G board, this systolic array can be extended across all four FPGAs on the board with only a single clock cycle of delay from one FPGA to the next. Thus, with an array of 2048 PEs per node and 24 total nodes in Novo-G, 49152 scores can be simultaneously computed per clock cycle.

Experiments were performed for a 34MB chromosome sequence aligned with 16K 128-character sequences randomly extracted from a different chromosome. Novo-G performance was compared against an optimized software implementation executed on a single 64-bit, 2.4GHz AMD Opteron core. As shown in Table 1, a speedup of 2665 was measured for this dataset (though different datasets produce similar results) on a single Novo-G node with 4 FPGAs. Thus, on a conventional HPC machine (even assuming no overhead), 2665 of these Opteron cores working in parallel would be required to perform the same amount of work in the same period of time as a single Novo-G node. The same dataset and program were partitioned across 16 Novo-G nodes (64 FPGAs), using MPI for coordination, where a speedup of 40849 was observed. If application behavior is extrapolated to the full complement all 24 Novo-G nodes with 4 FPGAs each, a speedup of about 60053 can be expected. Thus, Novo-G (with only one board per node) can achieve in about 12 seconds what would require a fast processor core nearly 9 days! Further, after Novo-G completes its upgrade to 2 boards per node (total 192 FPGAs), the projected speedup approaches 120000, which is comparable to the speed of running the same application without overhead on 120000 Opteron cores in a conventional supercomputer, far more than any machine on the national NSF TeraGrid. Moreover, Novo-G is hundreds of times lower in cost, power, cooling, size, weight, etc. than any one of these massive machines.

A second bioinformatics application, Needle-Distance (N-D) has been scaled to run on Novo-G, also with excellent results. Needle-Distance is an extension of the better-known Needleman-Wunsch (N-W) application for global sequence alignment. Based on the N-W algorithm, the N-D application is used to calculate needle-distance of sequence pairs commonly needed in bioinformatics applications [19].

Table 2 shows the best-case performance obtainable by N-D on Novo-G, computing the needle-distance of 192×224 sequence pairs with average length 450. The serial baseline was extrapolated by averaging the time to compute 224 distances over five separate runs (all within 0.1% of each other), then multiplying by 192. Experimentally, the dataset was executed on up to 64 FPGAs in Novo-G. The times for 96 and 192 FPGAs are projected. It is important to note that the estimate for 192 FPGAs is not simply twice the estimate for 96. In our projections, we do account for the degraded performance observed (approximately 6× instead of 8× for the N-D application) when using two boards per node rather than one, which is attributed to contention for PCIe and the file system plus the slight disadvantage of using hyperthreading as opposed to eight independent cores.

Experimental results show speedup on a single FPGA in Novo-G in excess of 2700 versus software on a 2.4 GHz Opteron core, with measured speedup approaching 160000 with 64 FPGAs. Projected speedup when employing all 192 of the FPGAs of Novo-G exceeds 365000, which is comparable to the most optimistic scenario of running the same application on over 365000 Opteron cores in a conventional supercomputer. To put this into perspective, the two most massive, expensive, and powerful computers cited at www.top500.org (Jaguar at ORNL and Roadrunner at LANL), have a *combined* approximate total of 346000 cores.

Currently, the Smith-Waterman application is being upgraded based upon lessons learned from our more recent Needle-Distance effort. The N-D application itself is being modified to create a standalone Needleman-Wunsch application, so it can be used as a component in a pipeline for composite bioinformatics applications. Furthermore, there are many active projects within the Novo-G Forum in the design and scaling of a variety of impactful applications to run on Novo-G, including applications in computational sciences, DSP, remote sensing, image processing, quantum modular dynamics, information retrieval, finance, seismic processing, and data mining.

Table 2. Measured and projected performance of N-D on Novo-G

Execution Time of needledist on Single 2.4 GHz Opteron Core = 11,673 hrs (~1.3 years!)	Number of Novo-G FPGAs in Execution					
	1	4	16	64	96(est)	192(est)
Execution Time (Sec) of Novo-G	15549	4078	1032	265	177	115
Novo-G Speedup vs. Single Core	2703	10304	40719	158576	237417	365415

4.2 Novo-G Energy

The following series of tests were performed on a Novo-G node to determine the power consumption of the server of the node (containing a quad-core E5520 Xeon CPU), as compared to the two GiDEL ProcStar-III (PS-III) boards (altogether containing 8 Stratix-III E260 FPGAs and 40GB of RAM):

- Idle server, without any PS-III (RC) boards installed and without any application running
- Loaded server, without any RC boards, but running the S-W application in software
- Loaded server, running the S-W application in software, with the two RC boards installed but idle
- Idle server, with the two RC boards fully loaded and running the S-W application in hardware
- Loaded server running S-W application in software and loaded RC boards running S-W application in hardware

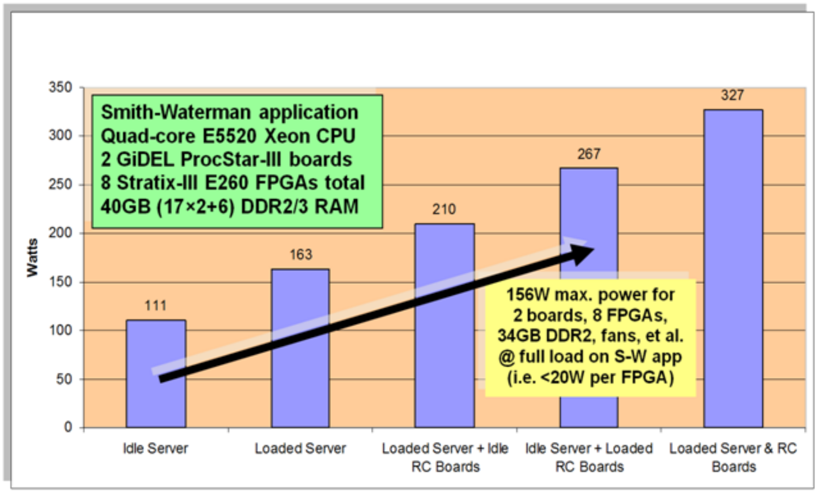


Figure 3. Power consumption of each Novo-G Node

The results of the tests are shown in Figure 3. Note that a maximum of 156W (i.e., 267W - 111W) is required to power two RC boards, 8 FPGAs, 34GB DDR2 RAM, fans, etc. at full load running the S-W application in hardware. Thus, each FPGA consumes less than 20W at full load and each node consumes 327W for full utilization of the resources of the quad-core CPU and eight FPGAs. The entire Novo-G machine consumes about 8 KW of power and is easily housed in three server racks with room to spare. Justifiably, the "G" in Novo-G can also represent "green" (Novo-Green).

5. Productivity: Novo-G Tools

Despite important advantages over conventional and alternative computing devices, reconfigurable computing will not become a widely accepted technology until the productivity challenge is overcome. In a recent DARPA-funded project [20], the four CHREC universities conducted studies to analyze limitations in existing FPGA tool flow and RC application design methods. An important outcome from the studies is an integrated research roadmap crafted by CHREC for DARPA to address the limitations. The essence of the roadmap is summarized in the concept diagram shown in Figure 4 and a set of productivity principles:

- Raise the level of abstraction throughout the entire development "stack" (Formulation, Design, Translation, and Execution), with an emphasis on Formulation-driven development. The reasoning is that the time spent up front in strategic design and design space exploration in the Formulation phase will pay major dividends and significantly reduce the time and the number of design iterations in the later phases.
- Exploit reuse throughout all phases to leverage pre-existing components.

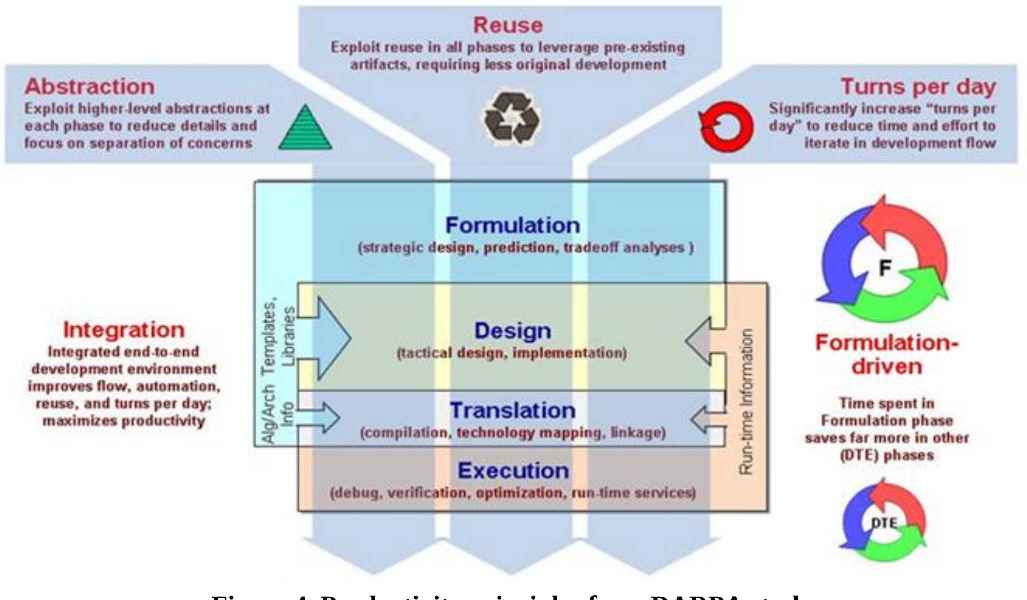

Reuse
Exploit reuse in all phases to leverage pre-existing artifacts, requiring less original development
Abstraction
Exploit higher-level abstractions at each phase to reduce details and focus on separation of concerns
Turns per day
Significantly increase "turns per day" to reduce time and effort to iterate in development flow
Formulation
(strategic design, prediction, tradeoff analyses)
Design
(tactical design, implementation)
Translation
(compilation, technology mapping, linkage)
Execution
(debug, verification, optimization, run-time services)
Alg/Arch Info
Templates, Libraries
Run-time Information
Integration
Integrated end-to-end development environment improves flow, automation, reuse, and turns per day; maximizes productivity
F
Formulation-driven
Time spent in Formulation phase saves far more in other (DTE) phases
DTE

Figure 4. Productivity principles from DARPA study

- Increase the number of "turns per day," making each design iteration more efficient and effective, and thus reducing the total number of design iterations.
- Tools should be integrated and interoperable, with some standard means for the tools to exchange information to support automation, reuse, and turns per day.

Application design productivity for reconfigurable systems is an important topic that has attracted the interest of many research groups and technology providers in the Novo-G Forum. For the remainder of this section, we will highlight some of the on-going productivity and tools projects from CHREC@UF (University of Florida) that are being adapted for Novo-G, including strategic design space exploration (Section V.A), system design and coordination techniques (Section V.B), virtual architectures for fast placement and routing (Section V.C), and performance analysis and verification (Section V.D).

5.1 Strategic Design Space Exploration

Unlike software designers, an RC application developer cannot productively evaluate the performance of different implementations by simply making rapid code changes, recompiling, and then executing the implementations. In an RC application design flow, the design-translate-execute (DTE) cycle (e.g., using RTL through HDL) is long and tedious due the lengthy synthesis, placement, and routing processes. Thus, there is a critical need for early design space exploration (DSE) to model and evaluate RC designs prior to going down implementation paths that can be potentially wasteful with unnecessary revisions. DSE is especially important as the size and complexity of RC applications scale up to RC supercomputing.

Shown in the left-hand side of Figure 5 is an RCML framework for Formulation which provides an integration point for several CHREC projects at the University of Florida that aim at supporting early DSE for RC systems. The framework is centered on an RC Modeling Language (RCML) [21], used for the estimation modeling of RC systems and applications. RCML is designed to allow users to efficiently model RC systems in the early stages of RC development. It enables users to separately model the *algorithm* and the execution *platform architecture* under study, providing specific constructs for defining parallelism, communication patterns, and other common aspects in RC applications. A *system model* is then created by mapping the algorithm model to the platform model. The mapping can be performed manually by the designer or can be assisted by an Automated Design-Space Exploration (ADSE) tool [22].

Design space exploration in the RCML framework is enabled by the ability to perform fast and reasonably accurate performance estimation of candidate RCML models of an RC system. RCML is designed to feed any number of analysis tools to enable efficient design-space exploration. Currently, an analytic methodology called the RC Amenability Test (RAT) [23] and a script-based simulation framework for RC [24] have been integrated within the framework. Case studies using a pair of image-processing applications show that, using RCML models that required only several minutes to create, the automated RAT tool was able to produce performance predictions within 0.6% to 6.2% of the actual execution time for a single-device application, while the simulation environment produced performance predictions within 2.3% to 7.4% of the actual execution time for a multi-device application [21].

Other tools can be developed to further extend the usefulness of the RCML framework. For example, a Core-level Modeling and Design (CMD) framework for RC algorithms [25] is being developed to support DSE with

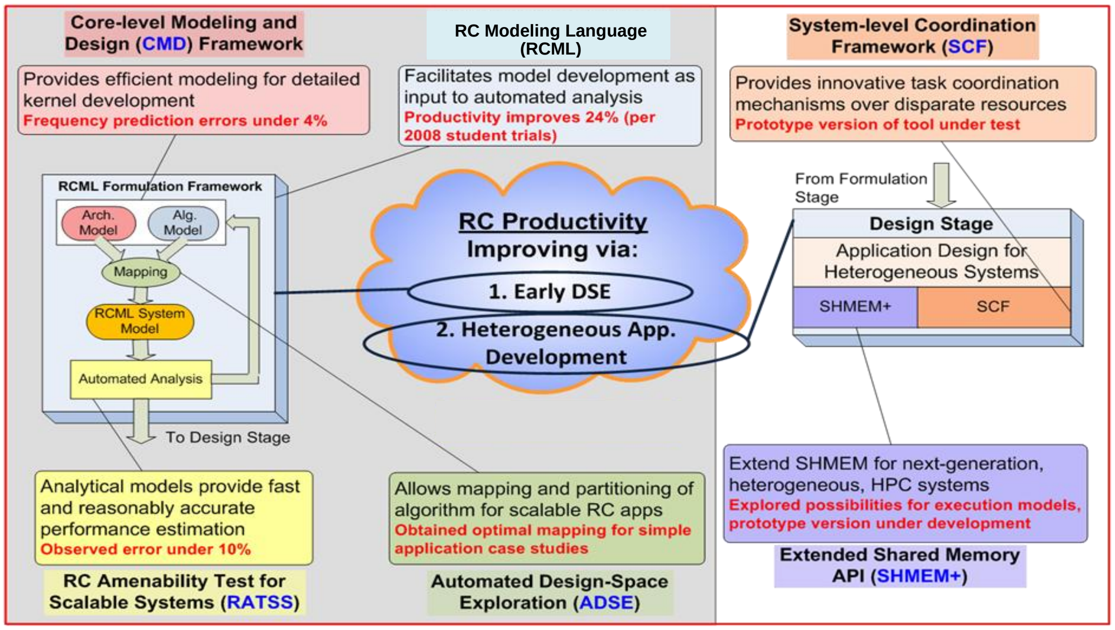

Core-level Modeling and Design (CMD) Framework
Provides efficient modeling for detailed kernel development
Frequency prediction errors under 4%
RC Modeling Language (RCML)
Facilitates model development as input to automated analysis
Productivity improves 24% (per 2008 student trials)
System-level Coordination Framework (SCF)
Provides innovative task coordination mechanisms over disparate resources
Prototype version of tool under test
RCML Formulation Framework
Arch. Model
Alg. Model
Mapping
RCML System Model
Automated Analysis
To Design Stage
RC Productivity Improving via:
1. Early DSE
2. Heterogeneous App. Development
From Formulation Stage
Design Stage
Application Design for Heterogeneous Systems
SHMEM+
SCF
Analytical models provide fast and reasonably accurate performance estimation
Observed error under 10%
RC Amenability Test for Scalable Systems (RATSS)
Allows mapping and partitioning of algorithm for scalable RC apps
Obtained optimal mapping for simple application case studies
Automated Design-Space Exploration (ADSE)
Extend SHMEM for next-generation, heterogeneous, HPC systems
Explored possibilities for execution models, prototype version under development
Extended Shared Memory API (SHMEM+)

Figure 5. Frameworks for RC productivity via early DSE and system-level design and coordination

more details (in contrast to abstract task-level modeling of RCML), without the need of coding (in contrast to RTL modeling) for algorithms on RC devices. Performance prediction, such as maximum clock frequency, supports core-level DSE and can help system-level modeling and design tools to achieve more accurate system-level DSE. Furthermore, CMD can be used for rapid bridging to the Design and Translation phases by generating code templates and design constraints that feed translation tools to rapidly obtain the predicted performance.

5.2 System Design and Coordination

Much of the advances in languages and tools for FPGAs (mainly by vendors) have simplified device-level design for FPGAs. However, system-level design issues (e.g., communication and synchronization between multiple devices in RC systems) have largely been unaddressed. Unlike traditional HPC systems, RC systems lack integrated, system-wide, parallel-programming models and languages, thus limiting most RC applications to small systems. Currently, application developers for large-scale RC systems have to resort to employing ad-hoc methods and multiple libraries and APIs to incorporate inter- and intra-node communication and synchronization for such systems, severely limiting development productivity.

Two on-going projects at CHREC@UF address system-level design for large-scale RC systems, using two different approaches. The first project, shown at the bottom-right corner of Figure 5, introduces a multi-level Partitioned Global Address Space (PGAS) model and a communication library for scalable, heterogeneous RC systems. The multi-level PGAS model is based on the traditional PGAS models designed for conventional HPC systems, leveraging their simplicity, syntax, and semantics, but extending them to capture the unique characteristics of RC systems. Based upon this multi-level PGAS model, we extend and adapt the SHMEM programming language [18] to become what we call SHMEM+ [26]. Using SHMEM+, designers can create scalable, parallel applications that execute over a mix of microprocessors and FPGAs. The higher level of abstraction provided by SHMEM+ can yield significant improvement in developer productivity.

In the second project (shown at the top-right corner of Figure 5), a System-Level Coordination Framework (SCF) uses a message-passing approach to enable transparent communication and synchronization between tasks running on heterogeneous processing devices in a system [27]. SCF consists of:

- A library of message-passing coordination primitives that allows an application to be expressed as a static task-graph. Each task can be coded in a different language and defined independently of the other tasks, devices, and communication architecture.
- A set of tools that can generate customized communication methods for a given system architecture based on the mapping of tasks to devices.

With SCF, many low-level architectural details are hidden from the application designer, allowing them to simply define each task in a language of their choice, while specifying coordination between tasks using message-passing primitives. Furthermore, SCF enables rapid exploration of different task-to-device mappings without modifications to task definition code, often resulting in both improved designer productivity and system performance. Finally, by allowing designers to define communication independently of the devices in a system, SCF improves application portability.

5.3 Intermediate Fabrics

In an RC application design flow, the design-translate-execute (DTE) cycle is long and tedious, commonly taking several hours, due largely to the placement and routing (PAR) process that maps a design to an FPGA. It is clear that in order to increase the turns per day for the DTE cycle, the bottleneck caused by PAR in existing device-vendor tools must be addressed.

To overcome this challenge, we are investigating *intermediate fabrics* (IFs), which are virtual reconfigurable fabrics specialized for fast PAR. As shown in Figure 6, IFs are implemented between user designs and the underlying physical FPGA (i.e., an intermediate translation layer).

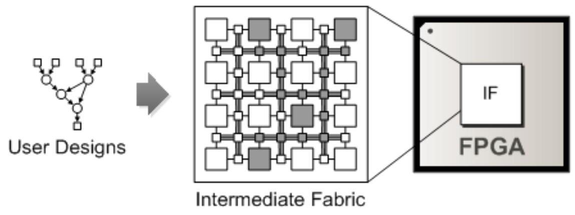
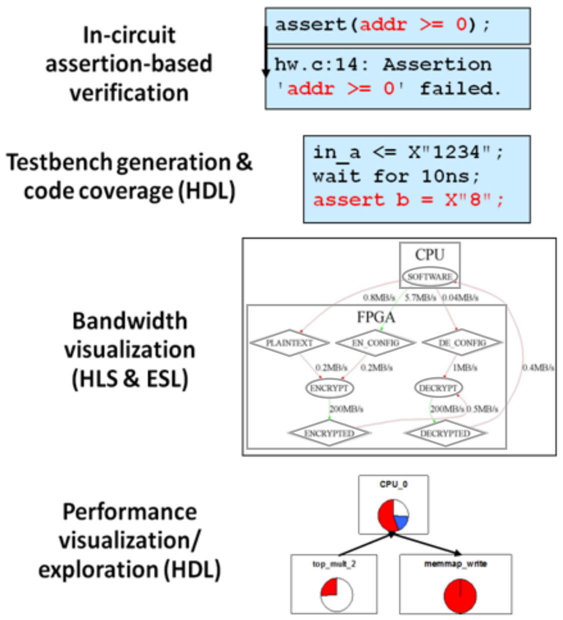


Figure 6. Intermediate Fabric for fast PAR

Unlike a physical device, whose architecture must support a wide range of applications, IFs can be specialized for particular application domains or even individual applications. Such specialization hides much of the complexity of fine-grained COTS devices, thus enabling fast placement and routing, while also enabling circuit portability across any devices that implement the intermediate fabric. The main challenge related to IF research is minimizing area and performance overhead introduced by a virtual fabric. Preliminary results show placement and routing speedups over 500×, with an average overhead of 10% of the slices on a Xilinx Virtex-4 LX200, and an average clock speed of 195 MHz.

5.4 Performance Analysis, Verification, and Debugging

While it is important to enable more turns per day, it is equally important to reduce the total numbers of turns (i.e., design iterations). An important factor in reducing the total number of design iterations is the effectiveness of the methods and tools for performance analysis, verification, and debugging of RC systems. To address these challenges, a Reconfigurable Computing Application Performance (ReCAP) framework is being developed, as illustrated in Figure 7, that provides performance analysis and visualization techniques, automatic bottleneck detection, in addition to simplified verification and debugging techniques for HDL [28], high-level synthesis (HLS) and electronic-system-level (ESL) tools [29-30].



```
assert(addr >= 0);
```

```
hw.c:14: Assertion
'addr >= 0' failed.
```

```
in_a <= X"1234";
wait for 10ns;
assert b = X"8";
```

Figure 7. ReCAP framework

For conventional parallel software applications (e.g., MPI), performance analysis tools are widely available. Unfortunately, traditional performance analysis tools only monitor application behavior from the CPU's perspective. Due to the complexity inherent in large-scale RC systems, unification of software and hardware performance analysis into a single tool is crucial to efficiently record and understand application behavior at runtime.

ReCAP analyzes performance by automatically inserting performance counters to track the behaviors of user-specified events and generate reports to support automatic bottleneck and hotspot detection. Furthermore, data representing increasingly large RC systems cannot be fully utilized without meaningful visualizations. At the HLS/ESL level, ReCAP provides bandwidth visualization to help designers in quickly pinpointing communication problems via bandwidth-annotated/colored design graphs. At the HDL level, performance visualization can be used to explore "what-if" scenarios concerning application changes and their performance effects. Thus, a designer can see the expected performance before going through the process of additional design iterations.

ReCAP also provides verification and debugging techniques targetting more mainstream RC designers. Although hardware verification techniques are well established, such techniques are largely focused on ASIC or safety-critical RC designs. For many potential RC users, such techniques are overly complicated and could instead be replaced by simpler techniques. One common problem is for RC designs to simulate correctly, but fail when executed on an actual FPGA. ReCAP assists with this problem by supporting in-circuit, assertion-based verification (ABV) for both HDL and high-level languages. Currently, ABV in ReCAP supports all 48 synchronous Open Verification Library (OVL 2.0) assertions [31]. In addition, ReCAP provides testbench generation to help determine where actual execution differs from simulation, while also performing automatic HDL code coverage.

6. Summary and Conclusions

This paper focused on the principal challenges of HPC for scientific computing and discussed why and how reconfigurable supercomputing is poised to make a major impact on accelerating scientific applications. Novo-G, believed to be the most powerful reconfigurable computer in the research community, provides an experimental testbed that serves as the centerpiece for a variety of research projects dedicated to understanding and advancing performance, productivity, and sustainability of future systems and applications. It is also an integration point for the Novo-G Forum, comprised of an international group of academic researchers and technology providers working collaboratively on applications and tools to establish and showcase advantages of reconfigurable computing at scale via Novo-G. Highlights from on-going research projects being adapted for Novo-G by CHREC@UF were also presented.

High-performance computing for many science domains is at a major crossroads and in the center of a convergence of several technology megatrends of the last decade. Technological advances in many science domains have resulted in the generation of massive amount of data that cannot be analyzed in a timely manner, impeding new scientific discoveries and progress. Conventional computing no longer can depend upon exploiting instruction-level parallelism and increased clock rate to improve performance. As a result, conventional HPC is inherently ill-equipped to address the escalating performance demands without resorting to massively large, energy-hungry, and expensive machines. The opportunity is ripe for *reconfigurable supercomputing* to satisfy emerging performance needs of scientific computing in a sustainable manner.

Acknowledgments

This work was supported in part by the I/UCRC Program of the National Science Foundation under Grant No. EEC-0642422. The authors gratefully acknowledge support for equipment and tools provided by Altera and GiDEL that helped make this work possible.

7. References

[1] Large Hadron Collider (LHC) project, CERN (European Organization for Nuclear Research). http://public.web.cern.ch/public/en/lhc/Computing-en.html.

[2] Large Synoptic Survey Telescope (LSST) project, http://www.lsst.org.

[3] U.S. Environmental Protection Agency, "Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431", Aug 2, 2007, http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf.

[4] L. Greenemeier, "CERN Gears Up Its Computers for More Atom Smashing", Scientific American, February 8, 2010, http://www.scientificamerican.com/article.cfm?id=cern-lhc-cloud-computing.

[5] The 22nd. Annual International Conference on Supercomputing (SC'09), Portland, OR, November 14-20, 2009, http://sc09.supercomputing.org/.

[6] J. Williams, A. George, J. Richardson, K. Gosrani, C. Massie, and H. Lam, "Characterization of Fixed and Reconfigurable Multi-Core Devices for Application Acceleration," ACM Transactions on Reconfigurable Technology and Systems (TRETS), to appear.

[7] Impulse-C, Impulse Accelerated Technologies, http://www.impulseaccelerated.com.

[8] Catapult C, Mentor Graphics, http://www.mentor.com.

[9] Carte-C, SRC Computers, http://www.srccomp.com.

[10] Simulink, The MathWorks, http://www.mathworks.com/products/simulink.

[11] AccelDSP, Xilinx, http://www.xilinx.com/tools/acceldsp.htm.

[12] B. Chamberlain, D. Callahan, and H. Zima, "Parallel Programmability and the Chapel Language", Int. J. High Perform. Comput. Appl., 21(3):291–312, 2007.

[13] P. Charles, et al., "X10: An Object-oriented Approach to Non-Uniform Cluster Computing", Proc. of 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, New York, NY, USA, pp. 519–538, 2005.

[14] P. Husbands, C. Iancu, and K. Yelick, "A Performance Analysis of the Berkeley UPC Compiler", Proc. of 17th Annual Int. Conference on Supercomputing (SC'03), New York, NY, USA, pp. 63–73, 2003.

[15] Khronos Group, "OpenCL 1.0", 2009, http://www.khronos.org/opencl/.

[16] MPI, "The Message Passing Interface (MPI) Standard", 2008, http://www-unix.mcs.anl.gov/mpi/.

[17] OpenMP, "The OpenMP API Specification for Parallel Programming", 2008, http://openmp.org/wp/.

[18] Silicon Graphics, Inc., "SHMEM API for Parallel Programming", 2008, http://www.shmem.org/.

[19] Y. Sun, Y. Cai, L. Liu, F. Yu, M. L. Farrell, W. McKendree and W. Farmerie, "ESPRIT: estimating species richness using large collections of 16S rRNA pyrosequences," Nucleic Acids Res., vol. 37, pp. e76, 2009.

[20] T. El-Ghazawi, A. George, et al., "Exploration of a Research Roadmap for Application Development and Execution on Field-Programmable Gate Array (FPGA)-based Systems", October 2008, http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA494473&Location=U2&doc=GetTRDoc.pdf

[21] C. Reardon, B. Holland, A. George, G. Stitt, and H. Lam, "RCML: An Environment for Estimation Modeling of Reconfigurable Computing Systems," ACM Transactions on Embedded Computing Systems (TECS), to appear.

[22] C.Reardon, A. George, G. Stitt, and H. Lam, "An Automated Scheduling and Partitioning Algorithm for Large-Scale Reconfigurable Computing Systems", accepted to the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'10), Las Vegas, NV, July 12-15, 2010.

[23] B. Holland, K. Nagarajan, C. Conger, A. Jacobs, and A. George, "RAT: A Methodology for Predicting Performance in Application Design Migration to FPGAs," Proc. of High-Performance Reconfigurable Computing Technologies & Applications Workshop (HPRCTA) at SC'07, Reno, NV, Nov. 11, 2007.

[24] C. Reardon, E. Grobelny, A. George, and G. Wang, "A Simulation Framework for Rapid Analysis of Reconfigurable Computing Systems," ACM Transactions on Reconfigurable Technology and Systems (TRETS), to appear.

[25] G. Wang, G. Stitt, H. Lam, and A. George, "A Framework for Core-level Modeling and Design of Reconfigurable Computing Algorithms," Proc. of High-Performance Reconfigurable Computing Technology and Applications Workshop (HPRCTA) at SC'09, Portland, OR, Nov. 15, 2009.

[26] V. Aggarwal, A. George, K. Yalamanchili, C. Yoon, H. Lam, and G. Stitt, "Bridging Parallel and Reconfigurable Computing with Multilevel PGAS and SHMEM+," Proc. of High-Performance Reconfigurable Computing Technology and Applications Workshop (HPRCTA) at SC'09, Portland, OR, Nov. 15, 2009.

[27] V. Aggarwal, R. Garcia, G. Stitt, A. George, and H. Lam, "SCF: A Device- and Language-Independent Task Coordination Framework for Reconfigurable, Heterogeneous Systems," Proc. of High-Performance Reconfigurable Computing Technology and Applications Workshop (HPRCTA) at SC'09, Portland, OR, Nov. 15, 2009.

[28] S. Koehler, J. Curreri, and A. George, "Performance Analysis Challenges and Framework for High-Performance Reconfigurable Computing," Parallel Computing journal (special issue on HPRC), Vol. 34, No. 4, May 2008, pp. 217-230.

[29] J. Curreri, G. Stitt, and A. George, "High-Level Synthesis Techniques for In-Circuit Assertion-Based Verification," Proc. of 17th Reconfigurable Architectures Workshop (RAW) at IPDPS'10, Atlanta, GA, Apr. 19-20, 2010.

[30] J. Curreri, S. Koehler, A. George, B. Holland, and R. Garcia, "Performance Analysis Framework for High-Level Language Applications in Reconfigurable Computing," ACM Transactions on Reconfigurable Technology and Systems (TRETS), Vol. 3, No. 1, Jan. 2010, pp. 1-23.

[31] Open Verification Library (OVL) Technical Subcommittee, 2010, http://www.accellera.org/activities/ovl.