The Promise of High-Performance Reconfigurable Computing

Tarek El-Ghazawi, Esam El-Araby, and Miaoqing Huang, George Washington University
Kris Gaj, George Mason University
Volodymyr Kindratenko, University of Illinois at Urbana-Champaign
Duncan Buell, University of South Carolina

Several high-performance computers now use field-programmable gate arrays as reconfigurable coprocessors. The authors describe the two major contemporary HPRC architectures and explore the pros and cons of each using representative applications from remote sensing, molecular dynamics, bioinformatics, and cryptanalysis.

In the past few years, high-performance computing vendors have introduced many systems containing both microprocessors and field-programmable gate arrays. Three such systems—the Cray XD1, the SRC-6, and the SGI Altix/RASC—are parallel computers that resemble modern HPC architectures, with added FPGA chips. Two of these machines, the Cray XD1 and SGI Altix, also function as traditional HPCs without the reconfigurable chips. In addition, several Beowulf cluster installations contain one or more FPGA cards per node, such as HPTi's reconfigurable cluster from the Air Force Research Laboratory.

In all of these architectures, the FPGAs serve as coprocessors to the microprocessors. The main application executes on the microprocessors, while the FPGAs handle kernels that have a long execution time but lend themselves to hardware implementations. Such kernels are typically data-parallel overlapped computations that can be efficiently implemented as fine-grained architectures, such as single-instruction, multiple-data (SIMD) engines, pipelines, or systolic arrays, to name a few.

Figure 1 shows that a transfer of control can occur during execution of the application on the microprocessor, in which case the system invokes an appropriate architecture in a reconfigurable processor to execute the target operation. To do so, the reconfigurable processor can configure or reconfigure the FPGA "on the fly," while the system's other processors perform computations. This feature is usually referred to as runtime reconfiguration.[1]

From an application development perspective, developers can create the hardware kernel using hardware description languages such as VHDL and Verilog. Other systems allow the use of high-level languages such as SRC Computers' Carte C and Carte Fortran, Impulse Accelerated Technologies' Impulse C, Mitrion C from Mitrionics, and Celoxica's Handel-C. There are also high-level graphical programming development tools such as Annapolis Micro Systems' CoreFire, Starbridge Systems' Viva, Xilinx System Generator, and DSPlogic's Reconfigurable Computing Toolbox.

Readers should consult *Computer*'s March 2007 special issue on high-performance reconfigurable computing for a good overview of modern HPRC systems, application-development tools and frameworks, and applications.

HPRC ARCHITECTURAL TAXONOMY

Many early HPRC systems, such as the SRC-6E and the Starbridge Hypercomputer, can be seen as attached processors. These systems were designed around one node of microprocessors and another of FPGAs. The two

Published by the IEEE Computer Society
0018-9162/08/$25.00 © 2008 IEEE

nodes were connected directly, without a scalable interconnection mechanism.

Here we do not address these early attached processor systems but focus instead on scalable parallel systems such as the Cray XD1, SRC-6, and SGI Altix/RASC as well as reconfigurable Beowulf clusters. These architectures can generally be distinguished by whether each node in the system is homogeneous (uniform) or heterogeneous (nonuniform).[2] A uniform node in this context contains one type of processing element—for example, only microprocessors or FPGAs. Based on this distinction, modern HPRCs can be grouped into two major classes: uniform node nonuniform systems and nonuniform node uniform systems.

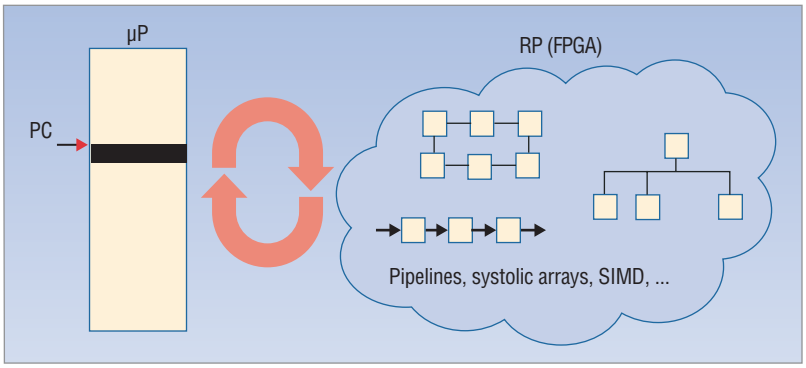

Figure 1. In high-performance reconfigurable computers, field-programmable gate arrays serve as coprocessors to the microprocessors. During execution of the application on the microprocessor, the system invokes an appropriate architecture in the FPGA to execute the target operation.

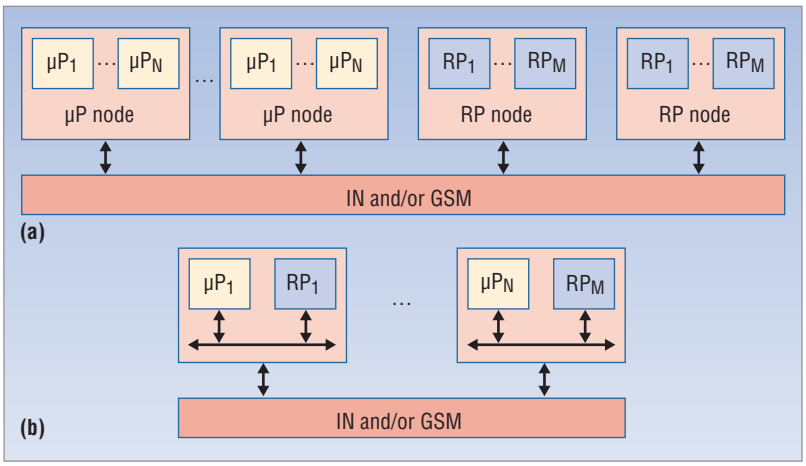

Figure 2. Modern HPRCs can be grouped into two major classes: (a) uniform node nonuniform systems (UNNSs) and (b) nonuniform node uniform systems (NNUSs).

Uniform node nonuniform systems

In UNNSs, shown in Figure 2a, nodes strictly have either FPGAs or microprocessors and are linked via an interconnection network to globally shared memory (GSM). Examples of such systems include the SRC-6 and the Altix/RASC. The major advantage of UNNSs is that vendors can vary the ratio of reconfigurable nodes to microprocessor nodes to meet the different demands of customers' applications. This is highly desirable from an economic perspective given the cost difference between FPGAs and microprocessors, and it is particularly suitable for special-purpose systems.

On the downside, having the reconfigurable node and the microprocessor node interact over the shared interconnection network makes them compete for overall bandwidth, and it also increases the latency between the nodes. In addition, code portability could become an issue even within the same type of machine if there is a change in the ratio between the microprocessor nodes and the FPGA nodes.

A representative example of the UNNS is the SRC-6/SRC-7, which consists of one or more general-purpose microprocessor subsystems, one or more MAP reconfigurable subsystems, and global common memory (GCM) nodes of shared memory space. These subsystems are interconnected through a Hi-Bar switch communication layer. The microprocessor boards each include two 2.8-GHz Intel Xeon microprocessors and are connected to the Hi-Bar switch through a SNAP interface. The SNAP card plugs into the dual in-line memory module slot on the microprocessor motherboard to provide higher data transfer rates between the boards than the less efficient but common peripheral component interconnect (PCI) solution. The sustained transfer rate between a microprocessor board and the MAP processors is 1,400 Mbytes per second.

The MAP Series C processor consists of one control FPGA and two user FPGAs, all Xilinx Virtex II-6000-4s. Additionally, each MAP unit contains six interleaved banks of onboard memory (OBM) with a total capacity of 24 Mbytes. The maximum aggregate data transfer rate among all FPGAs and OBM is 4,800 MBps. The user FPGAs are configured such that one is in master mode and the other is in slave mode. A bridge port directly connects a MAP's two FPGAs. Further, MAP processors can be connected via a chain port to create an FPGA array.

Nonuniform node uniform systems

NNUSs, shown in Figure 2b, use only one type of node, thus the system level is uniform. However, each node contains both types of resources, and the FPGAs are connected directly to the microprocessors inside the node. Examples of such systems are the Cray XD1 and

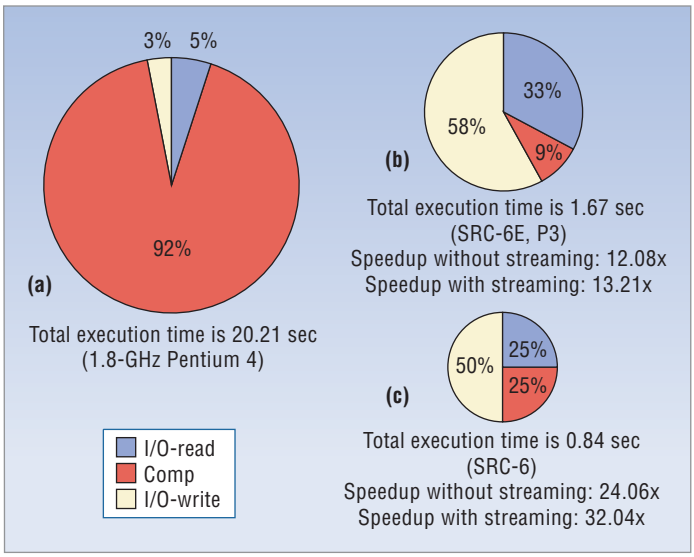

Figure 3. Execution profiles of hyperspectral dimension reduction. (a) Total execution time on 1.8-GHz Pentium 4 microprocessor. (b) Total execution time on SRC-6E. (c) Total execution time on SRC-6.

reconfigurable clusters. NNUSs' main drawback is their fixed ratio of FPGAs to microprocessors, which might not suit the traditional vendor-buyer economic model. However, they cater in a straightforward way to the single-program, multiple-data (SPMD) model that most parallel programming paradigms embrace. Further, the latency between the microprocessor and its FPGA coprocessor can be low, and the bandwidth between them will be dedicated—this can mean high performance for many data-intensive applications.

A representative example of the NNUS is the Cray XD1, whose direct-connected processor (DCP) architecture harnesses multiple processors into a single, unified system. The base unit is a chassis, with up to 12 chassis per cabinet. One chassis houses six compute cards, each of which contains two 2.4-GHz AMD Opteron microprocessors and one or two RapidArray Processors (RAPs) that handle communication. The two Opteron microprocessors are connected via AMD's HyperTransport technology with a bandwidth of 3.2 GBps forming a two-way symmetric multiprocessing (SMP) cluster. Each XD1 chassis can be configured with six application-acceleration processors based on Xilinx Virtex-II Pro or Virtex-4 FPGAs. With two RAPs per board, a bandwidth of 8 GBps (4 GBps bidirectional) between boards is available via a RapidArray switch. Half of this switch's 48 links connect to the RAPs on the compute boards within the chassis, while the others can connect to other chassis.

NODE-LEVEL ISSUES

We have used the SRC-6E and SRC-6 systems to investigate node-level performance of HPRC architectures in processing remote sensing[3] and molecular dynamics[4] applications. These studies included the use of optimization techniques such as pipelining and data transfer overlapping with computation to exploit the inherent temporal and spatial parallelism of such applications.

Remote sensing

Hyperspectral dimension reduction[3] is representative of remote sensing applications with respect to node performance. With FPGAs as coprocessors for the microprocessor, substantial data in this data-intensive application must move back and forth between the microprocessor memory and the FPGA onboard memory. While the bandwidth for such transfers is on the order of GBps, the transfers are an added overhead and represent a challenge on the SRC-6 given the finite size of its OBM.

This overhead can be avoided altogether through the sharing of memory banks, or the bandwidth can be increased to take advantage of FPGAs' outstanding processing speed. Overlapping memory transfers—that is, streaming—between these two processing elements and the computations also can help. As Figure 3a shows, such transfers (I/O read and write operations) take only 8 percent of the application execution time on a 1.8-GHz Pentium 4 microprocessor, while the remaining 92 percent is spent on computations.

As Figure 3b shows, the first-generation SRC-6E achieves a significant speedup over the microprocessor: 12.08× without streaming and 13.21× with streaming. However, the computation time is now only 9 percent of the overall execution time. In the follow-up SRC-6, the bandwidth between the microprocessor and FPGA increases from 380 MBps (sustained) to 1.4 GBps (sustained). As Figure 3c shows, this system achieves a 24.06× speedup (without streaming) and a 32.04× speedup (with streaming) over the microprocessor.

These results clearly demonstrate that bandwidth between the microprocessor and the FPGA must be increased to support more data-intensive applications—an area the third-generation SRC-7 is likely to address. It should be noted, however, that in most HPRCs today, transfers between the microprocessor and FPGA are explicit, further complicating programming models. These two memory subsystems should either be fused into one or integrated into a hierarchy with the objective of reducing or eliminating this overhead and making the transfers transparent.

Molecular dynamics

Nanoscale molecular dynamics (NAMD)[4] is representative of floating-point applications with respect to node performance. A recent case study revealed that when porting such highly optimized code, a sensible approach is to use several design iterations, starting with

the simplest, most straightforward implementation and gradually adding to it until achieving the best solution or running out of FPGA resources.[5]

The study's final dual-FPGA-based implementation was only three times faster than the original code execution. These results, however, are data dependent. For a larger cutoff radius, the original CPU code executes in more than 800 seconds while the FPGA execution time is unchanged, which would constitute a 260× speedup. The need to translate data between the C++ data storage mechanisms and the system-defined MAP/FPGA data storage architecture required considerable development effort. When creating code from scratch to run on an FPGA architecture, a programmer would implement the data storage mechanisms compatible between the CPU and FPGA from the beginning, but this is rarely the case for existing code and adds to the amount of work required to port the code.

Although the "official benchmark" kernel employs double-precision floating-point arithmetic, the NAMD researchers applied algorithmic optimization techniques and implemented their kernel using single-precision floating-point arithmetic for atom locations and 32-bit integer arithmetic for forces. Consequently, the final design occupies most available slices (97 percent), yet utilization of on-chip memory banks (40 percent) and hardware multipliers (28 percent) is low. The fact that the slice limit was reached before any other resource limits suggests that it might be necessary to restructure code to better utilize other available resources. One possible solution is to overlap calculations with data transfer for the next data set to use more available on-chip memory.

Despite the relatively modest speedup achieved, the NAMD study clearly illustrates the potential of HPRC technology. FPGA code development traditionally begins with writing code that implements a textbook algorithm, with little or no optimization. When porting such unoptimized code to an HPRC platform and taking care to optimize the FPGA design, it is easy to obtain a 10×-100× speedup. In contrast, we began with decade-old code optimized to run on the CPU-based platform; such code successfully competes with its FPGA-ported counterpart. It is important to keep in mind that the study's 100-MHz FPGA achieved a 3× application performance improvement over a 2.8-GHz CPU, and FPGAs are on a faster technology growth curve than CPUs.[6]

Lessons learned

Optimization techniques such as overlapping data transfers between the microprocessors and FPGAs with computations are useful for data-intensive, memory-bound applications. However, such applications, including hyperspectral dimension reduction and NAMD, can only achieve good performance when the underlying HPRC architecture supports features such as streaming or overlapping. Streaming can be enabled by architectures that are characterized by high I/O bandwidth and/or tight coupling of FPGAs with associated microprocessors. New promising examples of these are DCP architectures such as AMD's Torrenza initiative for HyperTransport links as well as Intel's QuickAssist technology supporting front side bus (FSB) systems. Large enough memory bandwidth is another equally important feature.

By memory bandwidth we mean that the memory system has sufficient multiplicity as well as speed, width, or depth/size. In other words, because FPGAs can produce and consume data at a high degree of parallelism, the associated memory system should also have an equal degree of multiplicity. Simply put, a large multiple of memory banks with narrow word length of local FPGA memory can be more useful to memory-bound applications on HPRCs than larger and wider memories with fewer parallel banks.

Vendor-provided transparent transfers can enhance performance by guaranteeing the most efficient transfer modes for the underlying platform.

In addition, further node architecture developments are clearly necessary to support programming models with transparent transfers of data between FPGAs and microprocessors by integrating the microprocessor memory and the FPGA memory into the same hierarchy. Vendor-provided transparent transfers can enhance performance by guaranteeing the most efficient transfer modes for the underlying platform. This will let the user focus on algorithmic optimizations that can benefit the application under investigation rather than data transfers or distribution. It also can improve productivity.

SYSTEM-LEVEL ISSUES

We have used the SRC-6 and Cray XD1 systems to investigate system-level performance of HPRC architectures in bioinformatics[7] and cryptanalysis[8-10] applications. These applications provide a near-practical upper bound on HPRC potential performance as well as insight into system-level programmability and performance issues apart from those associated with general high-performance computers. They use integer arithmetic, an area where HPRCs excel, are compute-intensive with lots of computations and not much data transfer between the FPGAs and microprocessors, and inherit both spatial and temporal parallelism.

We distributed the workload of both types of applications over all nodes using the message passing interface (MPI). In the case of DNA and protein analysis, we broadcast a database of reference sequences and scatter sequence queries. The application identified matching scores locally and then gathered them together. Each

				Expected		Measured	
				Throughput (GCUPS)	Speedup	Throughput (GCUPS)	Speedup
FASTA (ssearch34)	Opteron 2.4 GHz	DNA		NA	NA	0.065	1
		Protein		NA	NA	0.130	1
SRC-6 100 MHz (32x1)		DNA	1 Engine/chip	3.2	49.2×	3.19 → 12.2 1 → 4 chips	49 → 188 1 → 4 chips
			4 Engines/chip	12.8	197×	12.4 → 42.7 1 → 4 chips	191 → 656 1 → 4 chips
			8 Engines/chip	25.6	394×	24.1 → 74 1 → 4 chips	371 → 1,138 1 → 4 chips
		Protein		3.2	24.6×	3.12 → 11.7 1 → 4 chips	24 → 90 1 → 4 chips
XD1 200 MHz (32x1)		DNA	1 Engine/chip	6.4	98×	5.9 → 32 1 → 6 chips	91 → 492 1 → 6 chips
			4 Engines/chip	25.6	394×	23.3 → 120.7 1 → 6 chips	359 → 1,857 1 → 6 chips
			8 Engines/chip	51.2	788×	45.2 → 181.6 1 → 6 chips	695 → 2,794 1 → 6 chips
		Protein		6.4	49×	5.9 → 34 1 → 6 chips	45 → 262 1 → 6 chips

Figure 4. DNA and protein sequencing on the SRC-6 and Cray XD1 versus the open source FASTA program. An FPGA with one engine produced a 91× speedup, while eight cores on the same chip collectively achieved a 695× speedup.

FPGA had as many hardware kernels for the basic operation as possible. In the case of cryptanalysis, we broadcast the ciphertext as well as the corresponding plaintext; upon finding the key, a worker node sent it back to the master to terminate the search.

Bioinformatics

Figure 4 compares DNA and protein sequencing on the SRC-6 and Cray XD1 with the open source FASTA program running on a 2.4-GHz Opteron microprocessor. We used giga cell updates per second (GCUPS) as the throughput metric as well as to compute speedup over the Opteron. With its FPGA chips running at 200 MHz, the XD1 had an advantage over the SRC-6, which could run its FPGAs at only 100 MHz.

By packing eight kernels on each FPGA chip, the Cray XD1 achieved a 2,794× speedup using one chassis with six FPGAs. An FPGA with one engine produced a 91× speedup instead of the expected 98× speedup due to associated overhead such as pipeline latency, resulting in 93 percent efficiency. On the other hand, eight cores on the same chip collectively achieved a 695× speedup instead of the expected 788× speedup due to intranode communication and I/O overhead. The achieved speedup for eight engines/chip was 2,794× instead of the estimated (ideal) of 4,728× due to MPI internode communications overhead, resulting in 59 percent efficiency.

These results demonstrate that, with FPGAs' remarkable speed, overhead such as internode and intranode communication must be at much lower levels in HPRCs than what is accepted in conventional high-performance computers. However, given the speed of HPRCs, very large configurations might not be needed.

Cryptanalysis

The cryptanalysis results, shown in Tables 1 and 2, are even more encouraging, especially since this application has even lower overhead. With the Data Encryption Standard (DES) cipher, the SRC-6 achieved a 6,757× speedup over the microprocessor—again, a 2.4-GHz Opteron—while the Cray XD1 achieved a 12,162× speedup. The application's scalability is almost ideal.

In the case of the Cray XD1, straightforward MPI application resulted in using all nodes. However, it made sense for the node program to run on only one microprocessor and its FPGA; the other microprocessors on each node were not used. On the SRC-6, MPI processes had to run on the microprocessors, and the system had to establish an association between each microprocessor and a MAP processor. Because the SRC-6 was limited to two network interface cards that could not be shared efficiently, two MPI processes were sufficient. This meant the program could only run on one microprocessor and one MAP processor.

Lessons learned

Heterogeneity at the system level—namely, UNNS architectures—can be challenging to most accepted

Table 1. Secret-key cipher cryptanalysis on SRC-6.

Application	Hardware		Software		Speedup
	Number of search engines	Throughput (keys/s)	Number of search engines	Throughput (keys/s)	
Data Encryption Standard (DES) breaking	40	4,000 M	1	0.592 M	6,757×
International Data Encryption Algorithm (IDEA) breaking	16	1,600 M	1	2.498 M	641×
RC5-32/12/16 breaking	4	400 M	1	0.351 M	1,140×
RC5-32/8/8 breaking	8	800 M	1	0.517 M	1,547×

Table 2. Secret-key cipher cryptanalysis on Cray XD1.

Application	Hardware		Software		Speedup
	Number of search engines	Throughput (keys/s)	Number of search engines	Throughput (keys/s)	
Data Encryption Standard (DES) breaking	36	7,200 M	1	0.592 M	12,162×
International Data Encryption Algorithm (IDEA) breaking	30	6,000 M	1	2.498 M	2,402×
RC5-32/8/8 breaking	6	1,200 M	1	0.517 M	2,321×

SPMD programming paradigms. This occurs because current technology utilizes the reconfigurable processors as coprocessors to the main host processor through a single unshared communication channel. In particular, when the ratio of microprocessors, reconfigurable processors, and their communication channels differs from unity, SPMD programs, which generally assume a unity ratio, might underutilize some of the microprocessors. On the other hand, heterogeneity at the node level does not present a problem for such programs.

Heterogeneity at the system level is driven by nontechnological factors such as cost savings, which developers can achieve by tailoring systems to customers using homogeneous node architectures. However, this is at least partly offset by the increased difficulty in code portability. NNUS architectures are more privileged in this respect than their UNNS counterparts.

HPRC PERFORMANCE IMPROVEMENT

To assess the potential of HPRC technology, we exploited the maximum hardware parallelism in the previously cited studies' testbeds at both the chip and system levels. For each application, we filled the chip with as many hardware cores as possible that can run in parallel. We obtained additional system-level parallelism via parallel programming techniques, using the MPI to break the overall problem across all available nodes in order to decrease execution time. After estimating the size of a computer cluster capable of the same level of speedup, we derived the corresponding cost, power, and size savings that can be achieved by an SRC-6, Cray XD1, and SGI Altix 4700 with an RC100 RASC module compared with a conventional high-performance PC cluster.

As Tables 3-5 show, the improvements are many orders of magnitude larger. In this analysis, a 100× speedup indicates that the HPRC's cost, power, and size are compared to those of a 100-processor Beowulf cluster. The estimates are very conservative, because when parallel efficiency is considered, a 100-processor cluster will likely produce a speedup much less than 100×—in other words, we assumed the competing cluster to be 100 percent efficient. We also assumed that one cluster node consumes about 220 watts, and that 100 cluster nodes have a footprint of 6 square feet. Based on actual prices, we estimated the cost ratio to be 1:200 in the case of the SRC-6 and 1:100 in the case of the Cray XD1. The cost reduction is actually much larger than the tables indicate when considering the systems' associated power and size.

These dramatic improvements can be viewed as realistic upper bounds on the promise of HPRC technology because the selected applications are all compute-intensive integer applications, a class at which HPRCs clearly excel. However, with additional FPGA chip improvements in the areas of size and floating-point support, and with improved data-transfer bandwidths between FPGAs and their external local memory as well as between the microprocessor and the FPGA, a much wider range of applications can harness similar levels of

Table 3. Performance improvement of SRC-6 compared with a Beowulf cluster.

Application	Speedup	Savings		
		Cost	Power	Size
DNA and protein sequencing	1,138×	6×	313×	34×
DES breaking	6,757×	34×	856×	203×
IDEA breaking	641×	3×	176×	19×
RC5 breaking	1,140×	6×	313×	34×

Table 4. Performance improvement of Cray XD1 compared with a Beowulf cluster.

Application	Speedup	Savings		
		Cost	Power	Size
DNA and protein sequencing	2,794×	28×	148×	29×
DES breaking	12,162×	122×	608×	127×
IDEA breaking	2,402×	24×	120×	25×
RC5 breaking	2,321×	23×	116×	24×

Table 5. Performance improvement of SGI Altix 4700 with RC100 RASC module compared with a Beowulf cluster.

Application	Speedup	Savings		
		Cost	Power	Size
DNA and protein sequencing	8,723×	22×	779×	253×
DES breaking	28,514×	96×	3,439×	1,116×
IDEA breaking	961×	2×	86×	28×
RC5 breaking	6,838×	17×	610×	198×

benefits. For example, in the hyperspectral dimension reduction study, data transfer improvements between the SRC-6E and SRC-6, while using the same FPGA chips, almost doubled the speedup.

Our research revealed that HPRCs can achieve up to four orders of magnitude improvement in performance, up to three orders of magnitude reduction in power consumption, and two orders of magnitude savings in cost and size requirements compared with contemporary microprocessors when running compute-intensive applications based on integer arithmetic.

In general, these systems were less successful in processing applications based on floating-point arithmetic, especially double precision, whose high usage of FPGA resources constitutes an upper bound on fine-grained parallelism for application cores. However, they can achieve as high performance on embarrassingly parallel floating-point applications, subject to area constraints, as integer arithmetic applications. FPGA chips will likely become larger and have more integrated cores that can better support floating-point operations.

Our future work will include a comprehensive study of software programming tools and languages and their impact on HPRC productivity, as well as multitasking/multiuser support on HPRCs. Because porting applications from one machine to another, or even to the same machine after a hardware upgrade, is nontrivial, hardware architectural virtualization and runtime systems support for application portability is another good research candidate.

References

1. M. Taher and T. El-Ghazawi, "A Segmentation Model for Partial Run-Time Reconfiguration," *Proc. IEEE Int'l Conf. Field Programmable Logic and Applications,* IEEE Press, 2006, pp. 1-4.
2. T. El-Ghazawi, "Experience with Early Reconfigurable High-Performance Computers," 2006; http://hpcl.seas.gwu.edu/talks/Tarek_DATE2006.ppt.
3. S. Kaewpijit, J. Le Moigne, and T. El-Ghazawi, "Automatic Reduction of Hyperspectral Imagery Using Wavelet Spectral Analysis," *IEEE Trans. Geoscience and Remote Sensing,* vol. 41, no. 4, 2003, pp. 863-871.
4. J.C. Phillips et al., "Scalable Molecular Dynamics with NAMD," *J. Computational Chemistry,* vol. 26, no. 16, 2005, pp. 1781-1802.
5. V. Kindratenko and D. Pointer, "A Case Study in Porting a Production Scientific Supercomputing Application to a Reconfigurable Computer," *Proc. 14th Ann. IEEE Symp. Field-Programmable Custom Computing Machines,* IEEE CS Press, 2006, pp. 13-22.
6. K. Underwood, "FPGAs vs. CPUs: Trends in Peak Floating-Point Performance," *Proc. 12th ACM/SIGDA Int'l Symp. Field Programmable Gate Arrays,* ACM Press, 2004, pp. 171-180.
7. D.W. Mount, *Bioinformatics: Sequence and Genome Analysis,* 2nd ed., Cold Spring Harbor Laboratory Press, 2004.
8. O.D. Fidanci et al., "Implementation Trade-Offs of Triple DES in the SRC-6E Reconfigurable Computing Environment," *Proc. 5th Ann. Int'l Conf. Military and Aerospace Programmable Logic Devices,* 2002; www.gwu.edu/~hpc/rcm/publications/MAPLD2002.pdf.
9. R.L. Rivest, "The RC5 Encryption Algorithm," revised version, MIT Laboratory for Computer Science, Cambridge, Mass., 20 Mar. 1997; http://people.csail.mit.edu/rivest/Rivest-rc5rev.pdf.
10. A. Michalski, K. Gaj, and T. El-Ghazawi, "An Implementation Comparison of an IDEA Encryption Cryptosystem on Two General-Purpose Reconfigurable Computers," *Proc. 13th Ann. Conf. Field-Programmable Logic and Applications,* LNCS 2778, Springer, 2003, pp. 204-219.

Tarek El-Ghazawi *is a professor in the Department of Computer and Electrical Engineering, a founder of the High-Performance Computing Lab (HPCL) at George Washington University, and cofounder of the NSF Center for High-Performance Reconfigurable Computing (CHREC). His research interests include high-performance computing, parallel computer architectures, high-performance I/O, reconfigurable computing, experimental performance evaluations, computer vision, and remote sensing. El-Ghazawi received a PhD in electrical and computer engineering from New Mexico State University. He is a senior member of the IEEE and a member of the ACM. Contact him at tarek@gwu.edu.*

Esam El-Araby *is a doctoral student in the Department of Computer and Electrical Engineering and a research assistant in the HPCL at George Washington University. His research interests include reconfigurable computing, hybrid architectures, evolvable hardware, performance evaluation, digital signal/image processing, and hyperspectral remote sensing. El-Araby received an MSc in computer engineering from the George Washington University. Contact him at esam@gwu.edu.*

Miaoqing Huang *is a doctoral student in the Department of Computer and Electrical Engineering and a research assistant in the HPCL at George Washington University. His research interests include reconfigurable computing, high-performance computing architectures, cryptography, image processing, and computer arithmetic. Huang received a BS in electronics and information systems from Fudan University, Shanghai. Contact him at mqhuang@gwu.edu.*

Kris Gaj *is an associate professor in the Department of Electrical and Computer Engineering and leads the Cryptographic Engineering Lab at George Mason University. His research interests include reconfigurable computing, cryptography, computer arithmetic, hardware description languages, and software-hardware codesign. Gaj received a PhD in electrical and computer engineering from Warsaw University of Technology, Warsaw. He is a member of the International Association for Cryptologic Research. Contact him at kgaj@gmu.edu.*

Volodymyr Kindratenko *is a senior research scientist in the Innovative Systems Laboratory at the National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign. His research interests include high-performance, reconfigurable, and scientific computing. Kindratenko received a DSc in analytical chemistry from the University of Antwerp, Antwerp, Belgium. He is a senior member of the IEEE and the ACM. Contact him at kindr@ncsa.uiuc.edu.*

Duncan Buell *is a professor and chair of the Department of Computer Science and Engineering at the University of South Carolina. His research interests include high-performance computing applications, parallel algorithms and architectures, computer security, computational number theory, information retrieval, and algorithm analysis. Buell received a PhD in mathematics from the University of Illinois at Chicago. He is a senior member of the IEEE Computer Society and a member of the ACM and the American Mathematical Society. Contact him at buell@engr.sc.edu.*