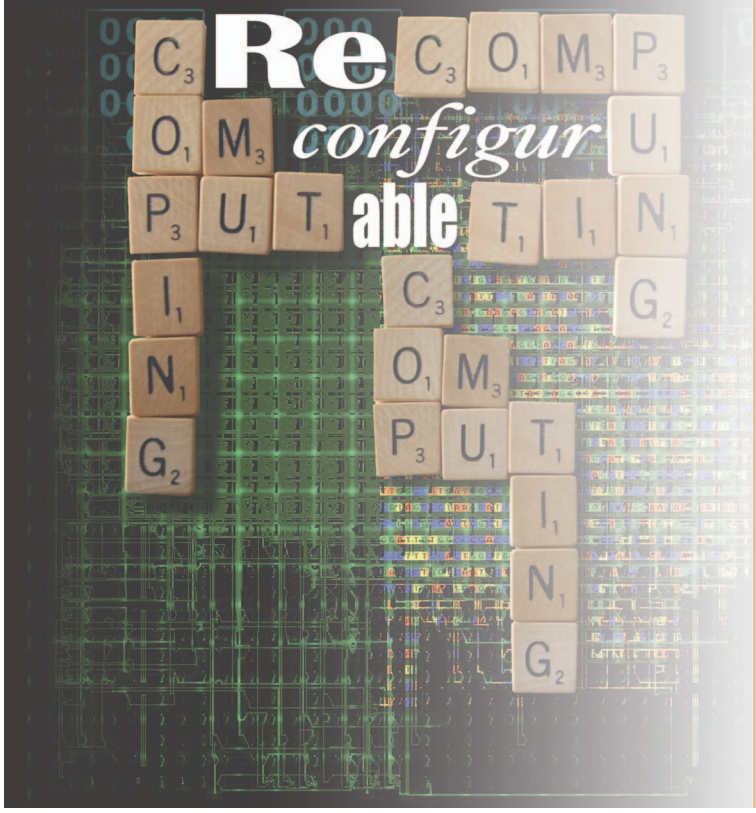

High-Performance Reconfigurable Computing

Duncan Buell, *University of South Carolina*
Tarek El-Ghazawi, *George Washington University*
Kris Gaj, *George Mason University*
Volodymyr Kindratenko, *University of Illinois at Urbana-Champaign*

High-performance reconfigurable computers have the potential to exploit coarse-grained functional parallelism as well as fine-grained instruction-level parallelism through direct hardware execution on FPGAs.

High-performance reconfigurable computers (HPRCs)[1,2] based on conventional processors and field-programmable gate arrays (FPGAs)[3] have been gaining the attention of the high-performance computing community in the past few years.[4] These synergistic systems have the potential to exploit coarse-grained functional parallelism as well as fine-grained instruction-level parallelism through direct hardware execution on FPGAs.

HPRCs, also known as reconfigurable supercomputers, have shown orders-of-magnitude improvement in performance, power, size, and cost over conventional high-performance computers (HPCs) in some compute-intensive integer applications. However, they still have not achieved high performance gains in most general scientific applications. Programming HPRCs is still not straightforward and, depending on the programming tool, can range from designing hardware to software programming that requires substantial hardware knowledge.

The development of HPRCs has made substantial progress in the past several years, and nearly all major high-performance computing vendors now have HPRC product lines. This reflects a clear belief that HPRCs

0018-9162/07/$25.00 © 2007 IEEE

have tremendous potential and that resolving all remaining issues is just a matter of time.

This special issue will shed some light on the state of the field of high-performance reconfigurable computing.

WHAT ARE HIGH-PERFORMANCE RECONFIGURABLE COMPUTERS?

HPRCs are parallel computing systems that contain multiple microprocessors and multiple FPGAs. In current settings, the design uses FPGAs as coprocessors that are deployed to execute the small portion of the application that takes most of the time—under the 10-90 rule, the 10 percent of code that takes 90 percent of the execution time. FPGAs can certainly accomplish this when computations lend themselves to implementation in hardware, subject to the limitations of the current FPGA chip architectures and the overall system data transfer constraints.

HPRCs are parallel computing systems that contain multiple microprocessors and multiple FPGAs.

In theory, any hardware reconfigurable devices that change their configurations under the control of a program can replace the FPGAs to satisfy the same key concepts behind this class of architectures. FPGAs, however, are the currently available technology that provides the most desirable level of hardware reconfigurability. Xilinx, followed by Altera, dominates the FPGA market, but new startups are also beginning to enter this market.

FPGAs are based on SRAM, but they vary in structure. Figure A in the "FPGA Architecture" sidebar shows an FPGA's internal structure based on the Xilinx architecture style. The configurable logic block (CLB) is the basic building block for creating logic. It includes RAM used as a lookup table and flip-flops for buffering, as well as multiplexers and carry logic. A side-by-side 2D array of switching matrices for programmable routing connects the 2D array of CLBs.

PROGRESS IN SYSTEM HARDWARE AND PROGRAMMING SOFTWARE

During the past few years, many hardware systems have begun to resemble parallel computers. When such systems originally appeared, they were not designed to be scalable—they were merely a single board of one or more FPGA devices connected to a single board of one or more microprocessors via the microprocessor bus or the memory interface.

The recent SRC-6 and SRC-7 parallel architectures from SRC Computers use a crossbar switch that can be stacked for further scalability. In addition, traditional high-performance computing vendors—specifically, Silicon Graphics Inc. (SGI), Cray, and Linux Networx—have incorporated FPGAs into their parallel architectures. In addition to the SRC-7, models of such HPC systems include the SGI RASC RC100 and the Cray XD1 and XT4. The Linux Networx work focuses on the design of the acceleration boards and on coupling them with PC nodes for constructing clusters.

On the software side, SRC Computers provides a semi-integrated solution that addresses the hardware (FPGA) and software (microprocessor) sides of the application separately. The hardware side is expressed using Carte C or Carte Fortran as a separate function, compiled separately and linked to the compiled C (or Fortran) software side to form one application.

Other hardware vendors use a third-party software tool, such as Impulse C, Handel-C, Mitrion C, or DSPlogic's RC Toolbox. However, these tools handle only the FPGA side of the application, and each machine has its own application interface to call those functions. At present, Mitrion C and Handel-C support the SGI RASC, while Mitrion C, Impulse C, and RC Toolbox support the Cray XD1. Only a library-based parallel tool such as the message-passing interface can handle scaling an application beyond one node in a parallel system.

RESEARCH CHALLENGES AND THE EVOLVING HPRC COMMUNITY

FPGAs were first introduced as glue logic and eventually became popular in embedded systems. When FPGAs were applied to computing, they were introduced as a back-end processing engine that plugs into a CPU bus. The CPU in this case did not participate in the computation, but only served as the front end (host) to facilitate working with the FPGA.

The limitations of each of these scenarios left many issues that have not been explored, yet they are of great importance to HPRC and the scientific applications it targets. These issues include the need for programming tools that address the overall parallel architecture. Such tools must be able to exploit the synergism between hardware and software execution and should be able to understand and exploit the multiple granularities and localities in such architectures.

The need for parallel and reconfigurable performance profiling and debugging tools also must be addressed. With the multiplicity of resources, operating system support and middleware layers are needed to shield users from having to deal with the hardware's intricate details. Further, application-portability issues should be thoroughly investigated. In addition, new chip architectures that can address the floating-point requirements of scientific applications should be explored. Portable libraries that can support scientific applications must be

FPGA Architecture

Ross Freeman, one of the founders of Xilinx (www.xilinx.com), invented field-programmable gate arrays in the mid-1980s. [1] Other current FPGA vendors include Altera (www.altera.com), Actel (www.actel.com), Lattice Semiconductor (www.latticesemi.com), and Atmel (www.atmel.com).

As Figure A shows, an FPGA is a semiconductor device consisting of programmable logic elements, interconnects, and input/output (I/O) blocks (IOBs)—all runtime user-configurable—that allow implementing complex digital circuits. The IOBs form a ring around the outer edge of the microchip; each IOB provides individually selectable I/O access to one of the I/O pins on the exterior of the FPGA package. A rectangular array of logic blocks lies inside the IOB ring.

A typical FPGA logic block consists of a four-input lookup table (LUT) and a flip-flop. Modern FPGA devices also include higher-level functionality embedded into the silicon, such as generic DSP blocks, high-speed IOBs, embedded memories, and embedded processors. Programmable interconnect wiring is implemented so that it's possible to connect logic blocks to logic blocks and IOBs to logic blocks arbitrarily.

A slice (using Xilinx terminology) or adaptive logic module (using Altera terminology), which contains a small set of basic building blocks—for example, two LUTs, two flip-flops, and some control logic—is the basic unit area when determining an FPGA-based design's size. Configurable logic blocks (CLBs) consist of multiple slices. Modern FPGAs consist of tens of thousands of CLBs and a programmable interconnection network arranged in a rectangular grid.

Unlike a standard application-specific integrated circuit that performs a single specific function for a chip's lifetime, an FPGA chip can be reprogrammed to perform a different function in a matter of microseconds. Typically, either source code written in a hardware description language, such as VHDL or Verilog, or a schematic design provides the functionality that an FPGA assumes at runtime.

As Figure B shows, in the first step, a synthesis process generates a technology-mapped netlist. A map, place, and route process then fits the netlist to the actual FPGA architecture. The process generates a bitstream—the final binary configuration file—that can be used to reconfigure the FPGA. Timing analysis, simulation, and other verification methodologies can validate the map, place, and route results.

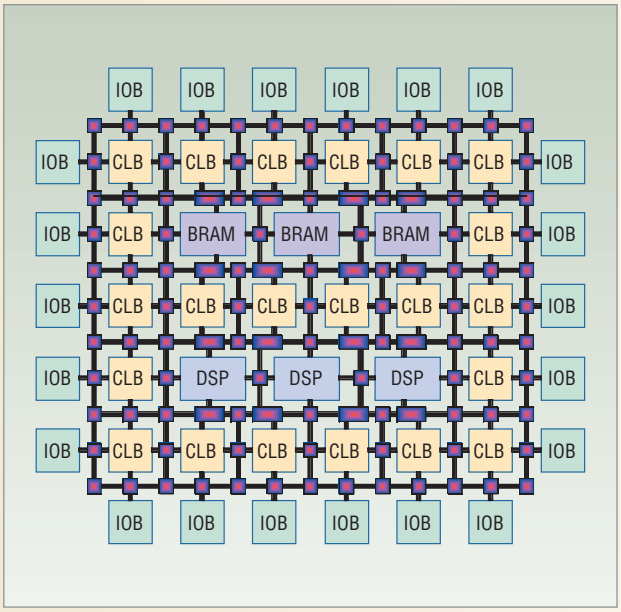

Figure A. FPGA internal structure based on the Xilinx architecture style. An FPGA can be described as "islands" of (reconfigurable) logic in a "sea" of (reconfigurable) connectors.

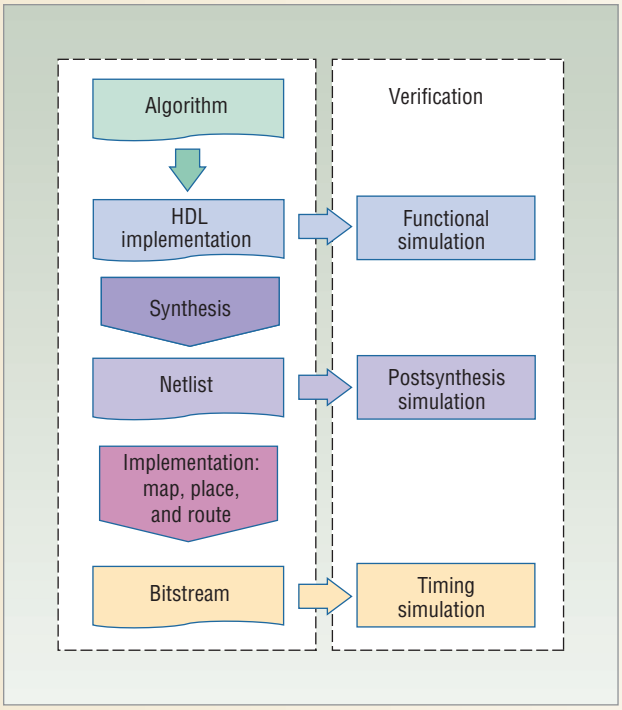

Figure B. Typical FPGA design flow.

Reference

1. S.M. Trimberger, ed., *Field-Programmable Gate Array Technology*, Kluwer Academic, 1994.

sought, and the need for more closely integrated microprocessor and FPGA architectures to facilitate the data-intensive hardware/software interactions should be further studied.

As researchers pursue developments to meet a wide range of HPRC requirements, the failure to incorporate standardization into some of these efforts would be detrimental. It can be particularly useful if academia, industry, and government work together to create a community that can approach these problems with the full intellectual intensity it deserves, subject to the needs of the end users and the experience of the implementers.

Some of this community-forming has been already observed. On the one hand, OpenFPGA (www.openfpga.org) has recently been formed as a consortium that mainly pursues standardization. On the other, the NSF has recently granted to the University of Florida and George Washington University an Industry/University Center for High-Performance Reconfigurable Computing (http://chrec.ufl.edu) award. The center includes more than 20 industry and government members who will guide the university research projects.

High-performance reconfigurable computing has demonstrated its potential to accelerate demanding computational applications.

IN THIS ISSUE

We have selected five articles for this special issue that represent the latest trends and developments in the HPRC field. The first two cover particularly important topics: a C-to-FPGA compiler and a library framework for code portability across different RC platforms. The third article describes an extensive collection of FPGA software development patterns, and the last two describe HPRC applications.

In "Trident: From High-Level Language to Hardware Circuitry," Justin Tripp, Maya Gokhale, and Kristopher Peterson describe an effort undertaken at the Los Alamos National Laboratory to build Trident, a high-level-language to hardware-description-language compiler that translates C language programs to FPGA hardware circuits. While several such compilers are commercially available, Trident's unique characteristics include its open source availability, open framework, ability to use custom floating-point libraries, and ability to retarget to new FPGA board architectures. The authors enumerate the compiler framework's building blocks and provide some results obtained on the Cray XD1 platform.

"V-Force: An Extensible Framework for Reconfigurable Computing" by Miriam Leeser and her colleagues and students from Northeastern University and the College of the Holy Cross outlines their efforts to implement the Vforce framework. Based on the object-oriented VSIPL++ standard, Vforce encapsulates hardware-specific implementations behind a standard API, thus insulating application-level code from hardware-specific details. As a result, as long as the third-party hardware-specific implementation is available, the same application code can run on different reconfigurable computer architectures with no change. The authors include examples of applications and results from using Vforce for application development.

In "Achieving High Performance with FPGA-Based Computing," Martin Herbordt and his students from Boston University share a valuable collection of FPGA software design patterns. The authors start with an observation that the performance of HPC applications accelerated with FPGA coprocessors is "unusually sensitive" to the quality of the implementation. They examine reasons for such a "sensitivity," list numerous methods and techniques to avoid generating "implementational heat," and provide a few application examples that greatly benefit from the uncovered design patterns.

"Sparse Matrix Computations on Reconfigurable Hardware," by Gerald Morris and Viktor Prasanna describes implementations of conjugate gradient and Jacobi sparse matrix solvers. In "Using FPGA Devices to Accelerate Biomolecular Simulations," Sadaf Alam and her colleagues from the Oak Ridge National Laboratory and SRC Computers describe an effort to port a production supercomputing application, a molecular dynamics code called Amber, to a reconfigurable supercomputer platform. Although the speedups obtained while porting these applications—highly optimized for the conventional microprocessors—to an SRC-6 reconfigurable computer are not spectacular, these articles accurately capture the overall trend.

Reconfigurable supercomputing has demonstrated its potential to accelerate computationally demanding applications and is rapidly entering the mainstream HPC world.

High-performance reconfigurable computing has demonstrated its potential to accelerate demanding computational applications. Much, however, must be done before this technology becomes a mainstream computing paradigm. The articles in this issue highlight a small subset of challenging problems that must be addressed. We encourage you to get involved with HPRC and contribute to this newly developing field.

References

1. D.A. Buell, J.M. Arnold, and W.J. Kleinfelder, eds., *Splash 2: FPGAs in a Custom Computing Machine*, IEEE CS Press, 1996.

2. M.B. Gokhale and P.S. Graham, *Reconfigurable Computing: Accelerating Computation with Field-Programmable Gate Arrays*, Springer, 2005.
3. S.M. Trimberger, ed., *Field-Programmable Gate Array Technology*, Kluwer Academic, 1994.
4. T. El-Ghazawi et al., "Reconfigurable Supercomputing Tutorial," *Int'l Conf. High-Performance Computing, Networking, Storage and Analysis* (SC06); http://sc06.supercomputing.org/schedule/event_detail.php?evid=5072.

Duncan Buell *is a professor in the Department of Computer Science and Engineering at the University of South Carolina, Columbia. Buell received a PhD in mathematics from the University of Illinois at Chicago. Contact him at buell@sc.edu.*

Tarek El-Ghazawi *is a professor in the Department of Electrical and Computer Engineering at the George Washington University, Washington, D.C. El-Ghazawi received a PhD in electrical and computer engineering from New Mexico State University. Contact him at tarek@gwu.edu.*

Kris Gaj *is an associate professor in the Department of Electrical and Computer Engineering at George Mason University, Fairfax, Virginia. Gaj received a PhD in electrical engineering from Warsaw University of Technology, Poland. Contact him at kgaj@gmu.edu.*

Volodymyr Kindratenko *is a senior research scientist at the National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Urbana. He received a DSc in analytical chemistry from the University of Antwerp, Belgium. Contact him at kindr@ncsa.uiuc.edu.*