Guest Editors' Introduction: Advances in Configurable Computing

Patrick Lysaght
Xilinx Research Laboratories

P.A. Subrahmanyam
CyberKnowledge

AT TIMES, it appears that the many definitions of configurable computing are every bit as configurable as the technology itself. For example, Wikipedia—the free, online, user-editable encyclopedia—defines configurable computing (or, synonymously, reconfigurable computing) as " … computer processing with highly flexible computing fabrics. The principal difference when compared to using ordinary microprocessors is the ability to make substantial changes to the data path itself in addition to the control flow" (http://en.wikipedia.org/wiki/Configurable_computing).

Apparently, the original idea might be attributable to Gerald Estrin, who, in a paper published in 1960, described how to use a computer to control an array of reconfigurable hardware.[1] This might be an acceptable working definition, but compare it to the proposal by DeHon and Wawrzynek, which defines reconfigurable computing as "computing via a postfabrication and spatially programmed connection of processing elements."[2]

Although it's arguable that neither definition is entirely complete, the latter clearly emphasizes two of the key components of reconfigurable computing:

- The architecture used in the computation is determined postfabrication and can therefore adapt to the characteristics of the executed algorithms.
- The computation is spatial, in contrast to the more temporal style associated with microprocessors.

The technology to realize Estrin's reconfigurable computer did not exist in 1960, but the vision survived intact and has become possible with the development of FPGA technology.

FPGAs

Almost two decades ago, the embryonic FPGA industry emerged. Its central value proposition was a new type of programmable-logic architecture predicated on the observation that *silicon is free*. It did not matter that making a single logic gate required as many as 100 transistors. What did matter were the convenience and time-to-market advantages that reconfigurable FPGAs offered.

The initial *glue logic* components were considered low-capacity, low-speed, high-cost parts, characteristics that limited them to prototype development and low-volume production runs. Today, FPGAs are large and fast enough for use in multimillion-gate designs running at hundreds of megahertz. According to independent industry analysts such as Dataquest, FPGAs now exceed the capacity and speed requirements of the vast majority of ASIC design starts. The cost per FPGA gate has, as expected, continuously eroded over time. Meanwhile, the total design cost, including nonrecurring engineering charges and design tool costs, has escalated for ASICs. The result is that FPGAs are now sufficiently competitive for designers to use them in high-volume designs that ship in hundreds of thousands of units per year.

FPGAs have established themselves as the third programmable platform after microprocessors and DSPs. They have ridden Moore's law relentlessly, even emerging as the process drivers of choice for some of the world's leading fabrication plants. Entering the era of the billion-transistor FPGA, it is clear that the premise at the foundation of FPGAs—that silicon is free—has been vindicated.

It is well understood, of course, that silicon is never really free—simply looking at the cost of today's large chips will confirm this understanding. The intent of the

0740-7475/05/$20.00 © 2005 IEEE Copublished by the IEEE CS and the IEEE CASS

slogan is to emphasize a style of microarchitectural design that rejects the once predominant philosophy of optimizing for minimum silicon area at all costs. Instead, area is, relatively speaking, freely tradable for other advantages, as FPGA architectures exemplify.

Configurable computing

As FPGAs matured, researchers were quick to recognize the potential of the third programmable platform. Indeed, in many ways, the term *configurable computing* is synonymous with a set of expanded expectations and opportunities that extend the role of FPGAs beyond that of traditional programmable-logic devices. The pioneers of configurable computing envisaged a new class of application-specific computing that combined the programmability of microprocessors and the implementation efficiencies of hardware. These computers were customizable after fabrication to solve any problem; they would include microprocessors and reconfigurable-logic arrays. Consider this excerpt from Estrin's original paper describing the goals of his work:

> … to permit computations which are beyond the capabilities of present systems by providing an inventory of high speed substructures and rules for interconnecting them such that the entire system may be temporarily distorted into a problem oriented special purpose computer.[3]

We can appreciate just how much the emergence of FPGAs contributed to revitalizing these ideas.

Motivation

Configurable computing achieves superior computation density by transforming problems from the temporal programming domain to spatially programmed implementations. The benefits are now widely reported for applications in disciplines as diverse as watermarking in image processing and gene sequencing in bioinformatics.[2,4]

Several factors contribute to the overall speedup. In many cases, configurable computing requires the complete reformulation of algorithms, transforming them from the original sequential encoding in a high-level language to a much more concurrent representation. The new algorithms are then mapped to the FPGA with highly specialized data paths, customized memory interfaces, and optimized interconnection topology. Extensive pipelining and retiming of circuits, to minimize the delays inherent in programmable interconnect, further enhance the resulting concurrency and parallelism.

A recent example demonstrates how to easily assemble an ad hoc configurable computer and deploy the resulting compute performance with impressive effect. A group of researchers at Johns Hopkins University and RSA Laboratories broke the encryption used in Texas Instruments' digital signature transponder (DST) product.[5] The product is a radio-frequency identification (RFID) device that is common in many automotive immobilization systems.

These researchers first tried to crack the encryption using hand-optimized software running on a 3.4-GHz Pentium workstation. This implementation computed fewer than 200,000 encryptions per second. With a cluster of 10 computers, they estimated that it would take over two weeks to crack a single key.

Next, the researchers assembled a configurable computer from 16 commercially available boards, each of which featured one Xilinx XC3S1000 FPGA. Altogether, the configurable computer hardware costs approximately the same as a high-performance PC. In this implementation, each FPGA computed nearly 16 million encryptions per second—approximately two orders of magnitude faster than the software implementation. Therefore, one FPGA could crack a single key in less than 21 hours. Using their inexpensive configurable computer, the team succeeded in cracking five keys in less than *two hours*. It is worth emphasizing that this configurable computer was an ad hoc configuration, so the researchers spent little time optimizing these results.

High-performance computing

The computational power of custom computing is attracting interest from mainstream companies in high-performance computing. Cray has announced that it has incorporated the custom computing technology that it acquired in purchasing OctigaBay into its latest, entry-level XD1 supercomputer.[6] The XD1 combines AMD Opteron processors with FPGAs for compute acceleration in a Linux environment. This configurable supercomputer has achieved orders-of-magnitude speedups in benchmarks from cipher breaking and elliptic-curve cryptography to vehicular-traffic simulation. Cray has adopted a library-based approach for hardware-accelerated routines.

Meanwhile, Silicon Graphics also exploits FPGAs as part of its multiparadigm computing strategy for what the company calls *reconfigurable application-specific computing*.[7] The company is targeting applications in

areas such as genomics; audio, video, and image signal processing; format translation; encoding and decoding; compression; database searching; the digital watermarking of video; and vibration/seismic analysis.

Programming paradigms and design tools

One key element of the configurable computing vision is that it does not involve the traditional logic design process of design entry with VHDL or Verilog followed by synthesis, and place and route. Instead, it envisages that users will code their applications within familiar software development design environments using high-level design abstractions, and conventional programming languages such as C.

The development of effective programming abstractions for this new *spatial-computing* paradigm is proving to be a stubborn research challenge. This is not altogether surprising considering how many generations of computer architectures became obsolete before compiler technology and computers could coexist synergistically and efficiently. Hennessy and Patterson[8] note that the compilers produced for Fortran in the late 1960s and 1970s were of very high quality. However, almost two decades passed before optimizing compiler technology and reduced-instruction-set computer (RISC) architectures intersected to provide the highly efficient platforms in use today. Current research into configurable computing architectures and compilers aims to achieve the same type of ease-of-programming and efficiency for the new spatial computing paradigm.

Special issue

The three articles we selected for this special issue are quite diverse and offer significantly different perspectives on configurable-computing research. The covered topics include new architectures for embedded reconfigurable devices; new programming models for integrating processors and reconfigurable accelerators; and designing high-performance reconfigurable supercomputers. What these articles share is a common vision about the importance of reconfigurable computing, the need for new programming abstractions, and some very exciting directions for future work.

In the first article, Mei et al. present a flexible architectural template called architecture for dynamically reconfigurable embedded systems (Adres). Adres is an integrated hardware-software framework for the systematic architectural exploration of a class of coarse-grained arrays that are tightly coupled to a very long instruction word (VLIW) processor. Adres presents two complementary functional views of a unified architecture that overlap in their use of physical resources: first, the VLIW with its row of functional units and multiport register files; and second, the array processor, which includes all of the functional units and register files, including those of the VLIW row. Access to external data memory is available via load/store operations on a subset of the functional units in the VLIW row.

Adres targets the ultra-low-power embedded systems in multifunction, portable terminals that are typical of today's consumer convergence products. The reconfigurable array of functional units operates at word or subword granularity and efficiently executes only computationally intensive kernels. The authors want to trade flexibility to reduce delay, area, power, and configuration time compared with that of FPGAs. The intention is to reconfigure the array to adapt to the varying needs of the instantaneous compute load.

In addition to the VLIW/array template, Adres includes a retargetable simulator and compiler to enable the systematic evaluation of different template instances with respect to performance, energy efficiency, area, and flexibility. The dynamically reconfigurable embedded system compiler (Dresc) makes applications programmable in C. The key to the Dresc framework is a software pipelining technique called modulo scheduling that simultaneously solves the placement, routing, and scheduling problems for coarse-grained arrays. This technique helps investigate different array topologies and various dimensions of functional units and register files by exercising them with compute-intensive kernels from a representative subset of multimedia and telecommunications applications. At present, the results are essentially comparative. Further work is necessary to establish robust performance figures derived from comprehensive silicon estimates.

The second article, by Vuletic, Pozzi, and Ienne, describes the development of a general, parallel-programming paradigm for reconfigurable SoCs (RSoCs). Their reconfigurable computing environment is an Altera Excalibur device consisting of a hard-core ARM microprocessor and an Altera FPGA. The authors envisage the FPGA operating as a reconfigurable hardware accelerator for the processor. They seek to support this use model with a unified and transparent programming model that deemphasizes the differences between software and hardware development.

Their abstraction presents a unified virtual-memory image for the application's software and hardware com-

ponents. A standard multithreaded programming model makes this possible—multiple software threads execute within the context of a common process, relying on the thread library and operating system support for interthread communication. The authors extend thread support to the tasks being accelerated in hardware (and coded in a hardware description language) by developing a virtualization layer. It consists of a software part (virtual window manager) that provides standardized operating system services to the user space libraries and applications, and a hardware part (window management unit) that provides a standardized hardware interface to the hardware accelerators.

Whatever the task, communication and synchronization exploit the same primitives so software threads are oblivious to the fact that they might be communicating with other nonsoftware threads. This enhances ease of programming as well as code portability (achieved via recompilation and resynthesis) across systems supporting similar operating system and accelerator configurations. The authors evaluate their extended thread abstractions using the benchmarks for the International Data Encryption Algorithm and adaptive differential pulse code modulation (for voice decoding). The results indicate that integrating the window management unit with the processor block, rather than implementing it in FPGA fabric, could significantly reduce the time and area overheads while preserving the ease-of-programming and portability advantages.

The final article in this special issue comes from Chang, Wawrzynek, and Brodersen of the University of California, Berkeley. It describes, from a practical perspective, the design of a second-generation, high-end reconfigurable computer consisting of commercial-off-the-shelf components, such as DRAM modules and standard network interfaces. However, it relies exclusively on FPGAs as the processing elements. Called the Berkeley Emulation Engine 2 (BEE2) project, it succeeds the Berkeley Wireless Research Center's original emulation engine (BEE).

The BEE2 project aims to create a universal reconfigurable computing system that can target a wide range of application domains, from high-performance digital signal processing (where FPGAs are already successful) to scientific computing (where the use of FPGAs is less mature). The project goals include

- designing a processing module as the building block for a family of high-end reconfigurable computers;
- developing several programming models; and
- demonstrating the machine's efficiency in applications such as high-performance digital signal processing, communication systems, and traditional scientific computing.

BEE2 targets applications such as the design of novel wireless communications systems, high-performance real-time digital signal processing, real-time scientific computation and simulation, and the acceleration of CAD tools. The first deployment of BEE2 is in radio astronomy for beamforming, spatial correlation, and wide-band fine-resolution spectroscopy. The latter is necessary for large radio telescope antenna arrays such as the Allen telescope array used in the Search for Extra-Terrestrial Intelligence (SETI) project. Researchers have deployed FPGA solutions in related applications for many years. The BEE2 system aims to go beyond these attempts by providing a generic, cost-effective solution that is sufficiently scalable to address a range of high-performance radio telescope DSP applications.

The authors report that the BEE2 system currently uses a synchronous dataflow model to program the reconfigurable fabric and the microprocessors. The programming environment is based on the Mathworks Simulink and Xilinx System Generator tools. The authors and others at the Berkeley Wireless Research Center have augmented these commercial tools with homegrown tools to provide automatic mapping from high-level block diagrams and state machine specifications for FPGA configurations.

Given the spectrum of potential applications for the BEE2, additional domain-specific programming models are under investigation. One candidate is the Message-Passing Interface standard. Because of its popularity with existing microprocessor-based supercomputers, Chang, Wawrzynek, and Brodersen anticipate that an MPI library for the BEE2 would make it much easier to port applications from conventional supercomputers to the BEE2.

As the consumerization of electronics continues, the need to incorporate the benefits of reconfigurable computing in consumer devices becomes more apparent. The compute density and the postfabrication customization advantages are especially relevant for increasingly complex products with extremely short time-to-market windows and very flexible product requirement definitions. However, the cost and power consumption of FPGAs currently prevent their use in

ultra-low-power products such as mobile phones. The Holy Grail, then, is a stand-alone, extremely-low-cost, ultra-low-power reconfigurable compute fabric. Intermediate steps toward this end include ASIC platforms that can incorporate such a fabric in varying amounts as dictated by the particular applications. Further, this fabric must be programmable in a manner that is natural for the application domain.

One very recent response to the programmability challenge is the generalized open source programmable logic (Gospl) platform from STMicroelectronics (http://www.gospl.org/fpl/static/aboutgospl.jsp). The Gospl initiative is the world's first open source platform for FPGAs. It seeks to create a community of open source developers who will mimic the Linux community in developing open source platforms and tools for reconfigurable computing based on programmable logic. To kick-start the initiative, STMicroelectronics is proposing to release the source code for its internally developed FPGA architecture and its software design tool suite. This purportedly consists of a million lines of code representing a $50 million investment. It is unclear (at the time of this writing) how this effort will proceed. Certainly, an open source effort of this particular type and scale in the context of configurable fabrics is unprecedented. ■

Acknowledgments

We thank the many people who contributed to making this special issue possible. These include the authors who submitted articles, the reviewers, the editor in chief, and the editorial and production staff at the IEEE Computer Society.

■ References

1. G. Estrin, "Reconfigurable Computer Origins: The UCLA Fixed-Plus-Variable (F+V) Structure Computer," *IEEE Annals History of Computing*, vol. 24, no. 4, Oct.-Dec. 2002, pp. 3-9.
2. A. DeHon and J. Wawrzynek, "Reconfigurable Computing: What, Why, and Implications for Design Automation," *Proc. 37th Design Automation Conf.* (DAC 99), ACM Press, 1999, pp. 610-615.
3. G. Estrin, "Organization of Computer Systems—The Fixed Plus Variable Structure Computer," *Proc. Western Joint Computer Conf.*, 1960, pp. 33-40.
4. A. DeHon, "The Density Advantage of Configurable Computing," *Computer*, vol. 33, no. 4, Apr. 2000, pp. 41-49.
5. S. Bono et al., "Security Analysis of a Cryptographically-Enabled RFID Device," 28 Jan. 2005; http://www.rfidanalysis.org/DSTbreak.pdf.
6. "Cray XD1 Brings High-Bandwidth Supercomputing to the Mid-Market," D.H. Brown Associates, 2004; http://www.cray.com/downloads/dhbrown_crayxd1_oct2004.pdf.
7. "Extraordinary Acceleration of Workflows with Reconfigurable Application-specific Computing from SGI," Silicon Graphics Inc., 2004; http://www.sgi.com/pdfs/3721.pdf.
8. J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, 1st ed., Morgan Kaufmann, 1990, pp. 127-130.

Patrick Lysaght is a senior director at Xilinx Research Laboratories. His research interests include reconfigurable computing, embedded systems, system-level modeling, and emerging design technologies for FPGAs. Lysaght has a BSc in electronic systems from the University of Limerick, Ireland, and an MSc in digital techniques from Heriot-Watt University in Edinburgh, Scotland. He has coauthored more than 40 technical papers and coedited two books on programmable logic, and is chair of the steering committee for the International Conference on Field-Programmable Logic and Its Applications (FPL 2005). He is a member of the IEEE.

P.A. Subrahmanyam is chairman of CyberKnowledge and a consulting professor in the Department of Electrical Engineering at Stanford University. His research interests include configurable computing systems and applications; wireless/sensor networks and security; embedded systems; and design technology. Subrahmanyam has an MSc and an MTech in physics and computer science from the Indian Institute of Technology, and a PhD in computer science from the State University of New York at Stony Brook. He is the author of more than 150 technical papers, has coedited five books, has nine awarded or pending international patents, and was the founding editor in chief of *Formal Methods in System Design*. He is a member of the IFIP Working Group 10.2/10.5; a cochair of the Commercial Handset/Modeling/Communication Architecture working group of the Software Defined Radio Forum; a founding member of the IEEE Communications Design and Development Committee; and a Fellow of the IEEE.

■ Direct questions and comments about this special issue to P.A. Subrahmanyam, 45110 Pawnee Drive, Fremont, CA 94539; psubra@ieee.org.