Introduction

Jürgen Becker

Institut für Technik der Informationsverarbeitung (ITIV),
Universität Karlsruhe (TH), Karlsruhe, Germany
E-mail: becker@itiv.uni-karlsruhe.de
WWW: http://www.itiv.uni-karlsruhe.de

Serge Vernalde

Interuniversity MicroElectronics Center (IMEC), Leuven Belgium
E-mail: Serge.Vernalde@imec.be
WWW: http://www.imec.be

Biographical notes: Jürgen Becker is full Professor for embedded electronic systems and head of the Institute for Information Processing (ITIV) at Universität Karlsruhe (TH). His actual research is focused on industrial-driven System-on-Chip (SoC) integration with emphasis on adaptivity and reconfigurability in hardware/software development for automotive and communication systems. He is Vice-President ('Prorektor') for Studies and Teaching at Universität Karlsruhe (TH). He is author and co-author of more than 150 scientific papers, and active as chairman of international conferences, Chair of GI/ITG Technical Committee of Architectures for VLSI Circuits, Associate Editor of IEEE Transactions on Computers, and Senior Member of the IEEE.

Serge Vernalde received the Electrical Engineering Degree in 1990 at the University of Leuven, Belgium. In 1990 he joined the IMEC-laboratory, focusing on design technology for wireless and multimedia systems. Between 1995 and 2004, he has been heading multiple research groups in the fields of digital broadband transceivers, reconfigurable systems and design technology for heterogeneous multiprocessor platforms. He is currently technical business director at IMEC, responsible for the partner relations in the context of IMEC's Multi-Mode Multi-Media (M4) program. He is author or co-author of over 60 scientific publications in international conferences and journals and was the general chair of the FPL 2004 conference.

In the design of embedded systems, Application-Specific Integrated Circuits (ASICs) have been common components by providing the high-performance and/or low-power budget that many systems require at the expense of long and difficult design cycles. In the middle of the 1980s the use of reprogrammable components, in particular, Field Programmable Gate Arrays (FPGAs) was introduced. Since the early years of programmable logic arrays, FPGA research has experienced tremendous growth in both academic and industrial organisations. As a result, chip programmability has now reached a high level of sophistication. Today's FPGAs provide cost-effective solutions for complex system designs by means of simple, homogeneous architectures. Thus, they are increasingly popular as a realistic alternative to ASIC and full-custom design. 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 are a promising technology for developing high-performance embedded systems. The density and performance of FPGAs have drastically improved over the past few years and FPGAs now exceed the capacity and speed requirements of the vast majority of ASIC design starts. Furthermore, the cost per FPGA gate has continuously declined over time. The total design cost, including non-recurring engineering charges and design tool costs, has escalated for ASIC-based designs. The trend in both industry and academia is to develop chips that include either embedded components in them such as memory, I/O controllers and multiplier blocks, or both system reconfigurable components and programmable cores. The resulting processors/chips, which are not anymore a single part of an embedded system but rather can be used to develop the whole system, are known by various names ranging from hybrid architectures to Systems-on-Chip (SoC), Configurable System-on-Chip (CSoC), Reconfigurable Systems-on-Chip (RSoC), and Systems on Programmable Chip (SoPC), among others. Thus, FPGAs and in particular reconfigurable devices are the integral parts in future embedded system design. As characteristic of an emerging technology, current FPGA research is taking diverse directions. At the lowest level, chip design and manufacturing issues such as interconnect technology and testing continue to demand attention. Fundamental questions about the internal organisation and the relative use of logic and routing resources in a chip remain only partially

addressed. FPGA research plays a substantial role at system level too. The rapid development of these chips has provided the technological basis for attaining configurable computing solutions with relative ease.

This issue is the second of two special issues, which features 11 papers out of 23 papers in total, selected from 38 submissions that represent the diverse problems being addressed today by the FPGA research community. The published papers are extended special editions of outstanding papers, which have been presented at the *11th Reconfigurable Architectures Workshop (RAW)* as part of the *18th Annual International Parallel and Distributed Processing Symposium (IPDPS, 2004)*. With authors from around the world, these papers bring us an international sampling of significant work.

In the first contribution on 'Runtime reconfigurable interfaces: the RTR-IFB approach', the authors S. Ihmor and W. Hardt address the critical aspects of inter-module communication in reconfigurable architectures because dependencies between reconfigured interacting computation modules in real-time environments lead to massive reconfiguration efforts. Their approach introduces runtime reconfigurable interface blocks (RTR-IFB), which can be a solution for solving those dependencies. The reconfiguration of modules during runtime, synchronisation and inter-module communication is handled. As one effect, reconfigured modules can share the same execution resources without having a communication gap. Their RTR-IFB methodology extends actual concepts for inter-module communication and explains the integration of RTR-IFBs in the IFB-Flow and the partial reconfiguration design flow. Another interesting topic on high-level synthesis is described in the second paper 'Overlapping memory operations with circuit evaluation in reconfigurable computing' by Y. Ben-Asher, D. Citron and G. Haber. Their contribution considers the problem of compiling programs, written in a general high-level programming language, into hardware circuits executed by an FPGA (Field Programmable Gate Array) unit. In particular, they consider the problem of synthesising nested loops that frequently access array elements stored in an external off-chip memory. They propose an aggressive profile-based compilation scheme, based on loop unrolling and code flattening techniques, where array references from/to the external memory are overlapped with an uninterrupted hardware evaluation of the synthesised loop's circuit. Additionally, the authors implemented a restricted programming language called DOL based on the proposed compilation scheme. Their experimental results provide preliminary evidence that aggressive compilation can be used to compile large code segments into circuits, including overlapping of hardware operations and memory references.

The third contributed paper on 'Dynamic reconfiguration for management of radiation-induced faults in FPGAs' by M. Gokhale, P. Graham, M. Wirthlin, D.E. Johnson and N. Rollins treats the problems, which may occur if embedded reconfigurable systems are deployed in radiated environments. Their paper describes novel methods of exploiting the partial, dynamic reconfiguration capabilities of Xilinx Virtex V1000 FPGAs to manage Single-Event Upset (SEU) faults due to radiation in space environments. The presented on-orbit fault detection scheme uses radiation-hardened reconfiguration controllers to continuously monitor the configuration bitstreams of nine Virtex FPGAs and to correct errors by partial, dynamic reconfiguration of the FPGAs while they continue to execute. In order to study the SEU impact on signal processing example applications, they introduce a novel fault injection technique to corrupt configuration bits, thereby simulating SEU faults. By using dynamic reconfiguration, they are able to run the corrupted designs directly on the FPGA hardware, giving many orders of magnitude speed-up over purely software techniques. Their work highlights the benefits of dynamic reconfiguration for space-based reconfigurable computing.

The fourth paper, 'Tuning adaptive microarchitectures' by A.S. Dhodapkar and J.E. Smith, presents an adaptive microarchitecture that reconfigures itself during runtime to match changing program requirements. The architecture employs four multi-configuration units – instruction and data cache, unified L2 cache and a branch predictor. The goal of adaptation is to achieve power efficiency without suffering from significant performance degradation. A light-weight profiling-hardware collects working-set signatures and cache miss-rates for detecting program phase changes and tuning each unit separately. The introduced tuning algorithms use these signatures to accurately detect program phase changes and decouple the tuning of each unit using a novel technique.

The paper 'Dynamic reconfiguration of Distributed Arithmetic designs' by K. Danne and C. Bobda addresses aspects of distributed arithmetic in runtime reconfigurable FPGA environments. Their paper explores various solutions to implement an application using runtime reconfigurable FPGA. The studied example is a mechatronic control system, which has to adapt its behaviour during system runtime. The described example application is modelled as a task-graph, in which every task is associated with a hardware module, which is characterised by its required FPGA resource and its computation time. In order to execute the application by using a runtime reconfigurable FPGA, a placement of the distributed arithmetic tasks, which defines how the tasks share the FPGA over time and a trade-off between execution time and required FPGA area, has to be found. The received estimated values are used to compute execution times and FPGA areas of the over-all system, for each of the proposed placements. The results show which mapping is optimal for the given application timing constraints, the order of controllers and reconfiguration speed of the used FPGA.

The sixth paper, 'System-level parallelism and concurrency maximisation in reconfigurable computing applications' by E. El-Araby, M. Taher, K. Gaj, T. El-Ghazawi, D. Caliga and N. Alexandridis addresses the performance bottleneck caused by DMA transfer I/O time between processor and FPGA, which can be greater than the

computations time. In this paper, they perform a theoretical and experimental study of this specific performance limitation. The mathematical formulation of the problem has been experimentally verified on a state-of-the art reconfigurable platform where they demonstrate and quantify the possible solution to this problem that exploits the system-level parallelism within reconfigurable machines.

The next contribution on 'Dynamically configurable security for SRAM FPGA bitstreams' by L. Bossuet, G. Gogniat and W. Burleson deals with a topic, which will be of importance in future reconfigurable applications. Since FPGAs are important for the electronic industry, it becomes necessary to improve their security policy particularly for SRAM FPGAs, since they are more vulnerable than other FPGA technologies. This paper proposes a solution to improve the security of SRAM FPGAs through flexible bitstream encryption. This proposition is distinct from other works because it uses the latest capabilities of SRAM FPGAs like partial dynamic reconfiguration and self-reconfiguration. Additionally there is no need for an external battery to store the secret key, which is a great benefit.

The eighth paper on 'Non-contiguous linear placement for reconfigurable fabrics' by C. Ababei and K. Bazargan presents efficient solutions for the non-contiguous linear placement of data paths for reconfigurable fabrics. A strip-based architecture is assumed for the reconfigurable fabric, which can be applied to Xilinx FPGAs. Two very efficient algorithms are proposed to solve the simpler problem of non-contiguous placement with blockages but without core reuse for tree graphs. The linear ordering obtained with any of the above algorithms is used as input for a third efficient algorithm to solve the problem of non-contiguous placement with both active and inactive cores. A fourth algorithm is proposed to solve the problem of non-contiguous placement with both core and connectivity reuse.

As the ninth contribution from the neuro/machine learning domain, the paper 'Dynamically reconfigurable neuron architecture for the implementation of self-organising learning array' by J.A. Starzyk, Y. Guo and Z. Zhu describes a new dynamically reconfigurable neuron hardware architecture based on the modified Xilinx Picoblaze microcontroller and Self-Organising Learning Array (SOLAR) algorithm. This architecture is aiming at using hundreds of traditional reconfigurable FPGAs to build the SOLAR learning machine, which has many advantages over traditional neural network hardware implementation. Neurons can be optimised for area and speed, and the whole system is dynamically self-reconfigurable during the runtime. Additionally the system architecture is expandable to a large multiple-chip system. The tenth paper 'Functional programming of real-time reconfigurable embedded systems' by A.G. Strelzoff proposes a Hardware/Software Co-Design execution model and new language based on functional programming, which removes the distinction between hardware and software and supports statically analysable real-time system design. He introduces a language called 'V', which can be viewed as the synthesisable subset of Verilog with additional functional programming features. V syntax looks much like Verilog or C without pointers in order to facilitate adaptation. The V compiler generates a net-list of the elementary functions, which are supported by a particular array. The execution model is a cycle-based synchronous dataflow.

The last paper in this issue 'Mapping a class of dependence algorithms to coarse-grained reconfigurable arrays: architectural parameters and methodology' by F. Hannig, H. Dutta and J. Teich addresses the problems of mapping algorithms by exploiting their inherent parallelism and the possibilities of array shaped reconfigurable hardware. The paper presents an overview of constraints, which have to be considered when mapping applications to coarse-grained reconfigurable arrays. Furthermore, the authors show their design methodology for mapping regular algorithms onto massively parallel arrays, which is characterised by loop parallelisation in the polytope model, and finally, in a first case study, they adapt their design methodology for targeting reconfigurable arrays, which shows that the presented regular mapping methodology may lead to highly efficient implementations taking into account the constraints of the architecture.

Acknowledgements

We would like to thank all who contributed to making this special issue possible. They include the authors who submitted their papers, the reviewers, the Editor in chief and the editorial and production staff at the IJES. We hope the readers will enjoy reading this special issue as much as we enjoyed compiling it.