Editorial

Jürgen Teich

Invasive computing

DOI 10.1515/itit-2016-0043

Multi-core computing is not the privilege of a few anymore, but available mainstream. It is not only deployed in high performance computing (HPC) centers, but also in our PCs and even in mobile devices such as tablet computers and mobile phones. Whereas computer architecture research has spent decades (and still does) in improving single core performance through the introduction of concepts such as deep instruction pipelining, multi-level cache architectures, achieving higher clock rates through progress in technology and efficiency through multi-threading, multi instead of single processor systems-on-a-chip, so called MPSoCs determine the future of computer technology. As an example, general purpose processing on Graphics Processing Units (GPUs) is a domain in which the core number on a chip has already surpassed the sonic barrier of 1,000. The semiconductor roadmap confirms this trend as unbroken.

Yet, in order to exploit such massively parallel processing capabilities also in our main-stream applications, and moreover, in the growing field of embedded applications and everyday devices, a rethinking in application development, exploitation of concurrency, and assignment of program pieces to such an abundant number of processors becomes necessary. As can be imagined, application programmers must be able to express concurrency, and there must be methods to dynamically distribute concurrent computations to many cores for speeding up program execution. Also, new techniques how to assign and manage 1,000 and more cores on a chip must be developed. Another question is how long centralized control will survive rather than new techniques for distributed workload distribution and management emerge.

Invasive Computing denotes a quite new parallel programming paradigm in which a multi-core programmer may reason about resources and qualities of program execution rather than just functional correctness. Notably, a seminal article has appeared in a 2008 issue of this journal on the main ideas and benefits: Invasive Computing provides a programmer new commands to request or literally invade a by-default exclusive claim of processors, bandwidth, and memory for individual regions of the program. Subsequently, the region of code is scheduled in parallel on the claim of resources such as a tile of RISC processors, a GPU or a domain-specific accelerator. This execution phase is called infection of the claim. Once the program terminates or wants to resume sequential execution, a retreat command releases the resources and returns them back to the pool.

Apart from increasing the efficiency of parallel computations by only occupying resources when needed, the goal of invasive computing is to enforce non-functional qualities including timeliness, security, and reliability. These are of utmost importance in many applications in embedded and cyber-physical systems today such as automotive, mobile communication, and real-time industrial control. As can be imagined, sharing and scheduling of resources in a best-effort way – as is the practice today – does not provide the necessary isolation of multiple concurrently executing applications for guaranteeing a *-predictable execution. Here, "*" stands for different qualities such as timeliness, fault-freeness, or a power-limited execution.

The DFG Transregio 89 Invasive Computing

Using invasive computing, resource-aware programming becomes possible such as temperature- or utilization-driven programming. Yet, to support the novel programming paradigm, new programming concepts, language integrations, compilers and operating systems as well as architectural changes in the design of MPSoCs to efficiently support invade, infect and retreat operations have been necessary. These were developed during the first four-year funding phase of the DFG Transregional Research Center 89 between the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), the Karlsruhe Institute of Technology (KIT) and the Technical University of Munich (TUM) starting in 2010. Since its start, major achievements include the definition of required programming language elements for

Jürgen Teich: Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Hardware/Software Co-Design, Cauerstr. 11, 91058 Erlangen, Germany, e-mail: juergen.teich@fau.de

the specification of invasion operations as well as a set of constraints to argue about number, types, and state of resources that may be invaded (see, e.g., www.invasic.de, project area A). A first invasive language based on the language X10 by IBM as well as a compiler for translation of invasive X10 programs (project area C) onto a heterogeneous invasive multi-tile architecture that has also been successfully jointly architected (project area B) is used for experimentation on an FPGA-based prototype (project Z2). The compiler interfaces to the invasive run-time support system *iRTSS* that provides for dedicated operating-system support for invasive computing. First invasive applications exploiting different types of processor and communication resources of an invasive network-on-chip (*i-NoC*) are running successfully and have shown considerable gains in resource utilization and computational efficiencies in comparison with their non-invasive counterparts.

*-predictability

A unique jewel of invasive computing, however, has not been exploited so far: By the fact that resources are temporally claimed (by default) in an exclusive manner, interferences due to multiple applications sharing the same resources – being the reality on today's multi-core platforms – may be reduced if not avoided completely. Moreover, *run-to-completion* is the default mode of thread execution. Finally, memory reconfiguration and isolation as well as bandwidth guarantees on the designed network-on-chip allow us also to provide predictable QoS also for communication.

In this special issue, four peer-reviewed and thoroughly revised papers have been accepted that describe results of the second phase of the Transregio 89 that has begun in 2014. These four presentations all deal with the problems of **-predictability of multi-objective execution qualities* of parallel invasive programs and including their *optimization* and *exploration of design space*. Joint investigations include new language constructs to define so-called *requirements* on desired, respectively qualities to be enforced during execution. Application-specific qualities do not only consider performance (e.g., execution time, throughput, etc.), but also aspects of *security* and *power*. Through analysis of application requirements from different domains including stream processing and malleable task applications, not only efficiency but also predictable execution may be enforced for applications stemming from robotics, imaging, as well as HPC.

The first chapter of this special issue is on **timing predictability**. This joint work proposes a hybrid application mapping methodology that combines design-time analysis of application mappings with run-time management. Design space exploration delivers several resource reservation configurations with verified real-time guarantees for individual applications. These timing properties can be guaranteed at run-time, as long as dynamic resource allocations comply with the offline analyzed resource configurations. A prerequisite for this are composable hardware concepts that enable the dynamic isolation of real-time applications from interference with other workloads. The article presents programming, optimization, analysis, and hardware techniques for enforcing timing predictability. A case study illustrates the timing predictable management of real-time computer vision applications in dynamic robot system scenarios.

The second chapter describes how to establish **security on demand**: If applications are assumed to act maliciously on a multi-core platform, many security problems may arise. In this article, different ways to deal with security problems in a resource-aware way are discussed. First, the attacker model and the different security requirements that application may have in multi-core systems are formalized. Subsequently, different hardware and software security mechanisms that can be dynamically configured to guarantee security on demand for invasive applications are introduced and discussed.

The third chapter deals with issues of **power and temperature management** using a self-aware agent-based approach: Indeed, power density has become the major design constraint since Dennard Scaling does not apply any longer for current and future technology nodes. Since V_{dd} has almost reached its scaling limits, V_{dd}'s impact on (switching) power leads to a quadratic increase of the on-chip power density (power per chip area). This, in turn, leads to unsustainable on-chip temperatures that manifest as high peak temperatures, temporal thermal gradients, spatial thermal gradients, etc. High temperatures in all its flavors have a profound impact on dependability. It will therefore be infeasible to run all cores of a 1,000+ core chip at all times at full performance. This phenomenon is also known as Dark Silicon. Minimizing the portion of Dark Silicon or proactively exploiting Dark Silicon in order to mitigate dependability issues or circuit aging mechanisms has become a major focus of research in advanced multi-core architectures. Here, the article presents an innovative approach of managing the computing resources invasively in order to minimize the negative impact Dark Silicon otherwise has. An agent-based technique provides self-adaptation and self-awareness and it scales well with the rising number of on-chip cores. On-chip sensors provide a large amount of data that is therefore classified and

analyzed before the actual relevant portion of data is fed as input to the agent system. As a result, the invasive agent-based multi-core management is highly adaptive and self-aware leading to a high degree of computing efficiency even with the Dark Silicon restrictions.

The fourth chapter from authors of the Transregio 89 deals with achieving reliable multi-core execution through **fault-tolerance on-demand**: As a consequence of technology scaling, today's complex multi-processor systems have also become more and more susceptible to errors. In order to satisfy reliability requirements, such systems require methods to detect and/or tolerate errors. This entails two major challenges: providing a comprehensive approach that ensures fault-tolerant execution of parallel applications across different types of resources, and optimizing resource usage in the face of dynamic fault probabilities or with varying needs for fault tolerance of different applications. In this contribution, a *holistic* and *adaptive* approach to provide fault tolerance on multi-processor systems on a chip (MPSoCs) on demand of an application or environmental needs is provided based on invasive computing. It is shown how invasive computing may provide fault tolerance on a heterogeneous MPSOC including hardware accelerators, and communication infrastructure such as a Network-on-Chip (NoC). Here, compile-time transformations to implement well-known redundancy schemes such as Dual Modular Redundancy (DMR) and Triple Modular Redundancy (TMR) for fault-tolerant loop execution on a class of Coarse-Grained Reconfigurable Arrays (CGRAs) are introduced. Moreover, a methodology to detect and adaptively mitigate faults in invasive NoCs is presented.

Finally, the special issue presents an overview article on a recent international initiative called "Elastic Computing" that you will see exploits quite similar ideas on resource allocation on demand, yet for cloud computing, in order to dynamically tradeoff multiple qualities of services that may be observed in reaction. This contribution is authored by Prof. Shahram Dustdar and members of the Distributed Systems Group at TU Vienna, Austria.

Bionotes

Prof. Dr.-Ing Jürgen Teich
Friedrich-Alexander-Universität
Erlangen-Nürnberg (FAU),
Hardware/Software Co-Design,
Cauerstr. 11, 91058 Erlangen, Germany
juergen.teich@fau.de

Jürgen Teich is with Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany, where he is directing the Chair for Hardware/Software Co-Design since 2003. He received the M.S. degree (Dipl.-Ing.; with honors) from the University of Kaiserslautern, Germany in 1989 and the Ph.D. degree (Dr.-Ing.; summa cum laude) from the University of Saarland, Saarbruecken, Germany, in 1993. In 1994, he joined the DSP design group of Prof. E. A. Lee in the Department of Electrical Engineering and Computer Sciences (EECS), University of California at Berkeley (PostDoc). From 1995 to 1998, he held a position at the Institute of Computer Engineering and Communications Networks Laboratory (TIK), ETH Zürich, Switzerland with his Habilitation on the topic of "Synthesis and Optimization of Digital Hardware/Software Systems" in 1996. From 1998 to 2002, he was Full Professor in the Electrical Engineering and Information Technology Department, University of Paderborn, Germany.
His current research focuses on electronic design automation of embedded systems with emphasis on hardware/software co-design, reconfigurable computing and multi-core systems. Prof. Teich has organized various ACM/IEEE conferences/symposia as Program Chair including CODES+ISSS'07, FPL'08, ASAP'10, and DATE'16. He serves regularly as a TPC member of many program committees including DAC, ASP-DAC, ICCAD, FPL, ASAP, FPT, FPGA, RECONFIG, ESTIMEDIA, VLSI Design, GECCO, EMO, RTSS, etc. He also serves and has served in the editorial board of journals including ACM TODAES and JES and has edited two text books on Hardware/Software Co-Design (Springer).
Prof. Teich is involved in many interdisciplinary projects on basic research as well as industrial projects. From 2003–2009, he was an elected board member (Fachkollegiat) of the Deutsche Forschungsgemeinschaft (DFG) for the area of Computer Architecture and Embedded Systems. He has been the initiator and coordinator of the DFG priority programme 1148 on "Reconfigurable Computing". Since 2010, he has also been the principal coordinator of the Transregional Research Center 89 "Invasive Computing" funded by the DFG. In 2011, he was elected member of the Academia Europaea.