

Multicore Enablement for Embedded and Cyber Physical Systems

Edited by

Andreas Herkersdorf¹ and Michael Paulitsch²

1 TU München, DE, herkersdorf@tum.de

2 EADS – München, DE, michael.paulitsch@eads.net

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 13052 “Multicore Enablement for Embedded and Cyber Physical Systems”. During the seminar the participants from industry and academia actively discussed chances and problems of multicore processors in embedded in cyber-physical systems. The focus of the seminar was on the exchange of experiences and discussion of the challenges of reusable and transferable multicore technologies. Those were covered in the individual talks and plenum discussions. Beside that, working groups have been formed to discuss and present important topics in detail, which are also part of this report.

Seminar 27. January to 01. February, 2013 – www.dagstuhl.de/13052

1998 ACM Subject Classification C.1.4 Parallel Architectures

Keywords and phrases Multicore, hardware, software, platforms, embedded systems, security, real-time, safety, cyber physical systems

Digital Object Identifier 10.4230/DagRep.3.1.149


Edited in cooperation with Stefan Wallentowitz

1 Executive Summary

Andreas Herkersdorf

Michael G. Hinchey

Michael Paulitsch

License  Creative Commons BY 3.0 Unported license
© Andreas Herkersdorf, Michael G. Hinchey, and Michael Paulitsch

Multicore processors are a key enabling technology for solving grand societal challenges of the coming decades. Secure and ecological mobility, geographic coverage of high-tech healthcare, sustainable energy generation, distribution and management, and in general the development of our digitized society impose compute performance requirements on distributed embedded and cyber physical IT equipment which makes multicore technology indispensable. All leading processor vendors – ARM, Freescale, IBM, Infineon, Intel, MIPS, Nvidia – follow a strictly multicore-oriented strategy. Due to the paradigm shift from exploiting instruction level to process level parallelism, multicore processors are superior over single-core representatives with respect to computing performance and energy efficiency. Prerequisite is, processes can be balanced among parallel cores such that the nominally available computing performance can be utilized effectively, and cores can be set into sleep mode or power gated when not busy. As of today, the ability to efficient utilize the available resources depends to a large extent on the aptitude of experienced programmers and the inherent ability of being able to parallelize the computing problem.



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Multicore Enablement for Embedded and Cyber Physical Systems, *Dagstuhl Reports*, Vol. 3, Issue 1, pp. 149–182
Editors: Andreas Herkersdorf and Michael Paulitsch



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Embedded and Cyber Physical Systems exhibit demands for “non-functional requirements”, such as low(est) power and energy dissipation, reliability, availability and security, real-time and cost constraints, which are typically not found to the same extent in general purpose computing applications. The enablement of multicore technology for embedded and cyber-physical markets imposes serious challenges to industry and academia which can easily overwhelm the capabilities and capacities of individual corporations or even consortia. Industry and university research in Europe recognized early and invested significantly into the establishment of multicore know-how and competences. Examples of related projects at EU level and in Germany are: RECOMP – Reduced Certification Costs Using Trusted Multicore Platforms, ACROSS – ARTEMIS CROSS-Domain Architecture, SPES 2020 – Software Plattform Embedded Systems 2020, Cesar – Cost-efficient methods and processes for safety relevant embedded systems, MERASA – Multicore Execution of Hard Real-Time Applications Supporting Analysability (see Relationship to other seminars and projects for a more complete listing), and ARAMiS – Automotive, Railway and Avionics Multicore Systems.

The seminar brought together leading industry and university research groups from different fields of embedded system design and application development, multicore architecture and hardware/software design methodology & tools. The main objective of the seminar was on reporting experiences and discussing challenges of reusable and transferable multicore technologies among participants representing different application markets and scientific backgrounds. The technical focus of the agenda was on:

- Generic hardware/software building blocks for real-time performance, dependability, functional safety and security for embedded systems built around enhanced standard multicore solutions.
- System modeling, design and validation methods and tools for such platforms.

The seminar established new and strengthened existing ties between players and networks in the area of multicore and embedded technologies. Topical working groups were formed on the following topics:

- Specification & Interference
- Industrial Perspective on MultiCore Motivations and Challenges
- Certification of Safety-Critical Multicore Systems: Challenges and Solutions
- Network-on-Chip – Dependability and Security Aspects
- Multicore Ecosystem
- Secure Elements in future embedded multicore systems

The working groups compiled summaries reflecting the status and outlook on the respective topic. These summaries can be found in the sequel of this report.

2 Table of Contents

Executive Summary

Andreas Herkersdorf, Michael G. Hinchey, and Michael Paulitsch 149

Overview of Talks

The ARTEMIS ACROSS project <i>Christian El Salloum</i>	153
A Model-based Approach for Optimizing Existing Real-Time Software on Multicore Processors <i>Michael Deubzer</i>	153
IDAMC – A manycore architecture for mixed critical applications <i>Rolf Ernst</i>	154
Commerical Challenges of MultiCores in Automotive Domain <i>Glenn Farrall</i>	154
Timing Predictability of Multi-Core Processors <i>Christian Ferdinand</i>	154
Road to the use of multicore processors in space systems <i>Massimo Ferraguto</i>	155
Analysis of Embedded Software for Multicore in the Automotive Domain <i>Steffen Goerzig</i>	155
Multi core – Single bus <i>Rene Graf</i>	156
Distilling Programs for Multicore Architectures <i>Geoff Hamilton</i>	156
Necessity for & Feasibility of a Multicore Ecosystem <i>Andreas Herkersdorf</i>	157
“Heterogeneous Multiprocessing or Just a Bunch of Coprocessors?” – The case for unified programmability <i>Enno Luebbers</i>	158
Fault-Tolerant Time-Triggered Communication Infrastructure for Multi-Processor Systems-on-a-Chip <i>Roman Obermaisser</i>	158
Multi-Core in Avionics – On Problems and One Technical Approach Monitoring-Based Shared Resource Separation for Commercial Multi-Core System-On-Chip <i>Michael Paulitsch</i>	159
Talk on ARTEMIS Project RECOMP <i>Michael Paulitsch</i>	159
Talk on German Project ARAMiS – Automotive, Railway, and Avionics Multi-Core Systems <i>Michael Paulitsch</i>	160
Fine grained process migration for MPSoCs <i>Sri Parameswaran</i>	160

Task Mapping for Manycore-based Embedded Real-Time Systems <i>Stefan M. Petters</i>	161
Sustainable Development of Software in the Multi-Core Age <i>Matthias Pruksch</i>	161
Chances and risks for security in Multicore processors <i>Georg Sigl</i>	161
Isolation of Cores to Support Development of Mixed Critical Systems <i>Claus Stellwag</i>	163
Safe(r) Loop Computations on Multi-Cores <i>Jürgen Teich</i>	164
parMERASA- Multi-Core Execution of Parallelised Hard Real-Time Applications <i>Theo Ungerer</i>	164
OpTiMSoC – An Open Source Experimentation Platform for Multicore <i>Stefan Wallentowitz</i>	165
Efficient observation of Multicore SoCs <i>Alexander Weiss</i>	165
Many cores – many problems <i>Reinhard Wilhelm</i>	166
High-level Simulation-based Design Space Exploration on Multicore Virtual Platforms <i>Thomas Wild</i>	167
Working Groups	
Specification & Interference <i>Claus Stellwag, Michael Deubzer, and Glenn Farrall</i>	167
Industrial Perspective on MultiCore Motivations and Challenges <i>Glenn Farrall, Christian Ferdinand, Massimo Ferraguto, Steffen Görzig, Michael Paulitsch, Matthias Pruksch, Claus Stellwag, Sergey Tverdyshev, and Alexander Weiss</i>	169
Certification of Safety-Critical Multicore Systems: Challenges and Solutions <i>Stefan M. Petters and Rene Graf</i>	173
Network-on-Chip – Dependability and Security Aspects <i>Roman Obermaisser, Christian El Salloum, Theo Ungerer, and Thomas Wild</i> . . .	175
Multicore Ecosystem <i>Andreas Herkersdorf, Johan Lilius, Massimo Ferraguto, Christian Thiel, Stefan Wallentowitz, and Thomas Wild</i>	177
Secure Elements in future embedded multicore systems <i>Georg Sigl, Sri Paramareswaran, Michael Paulitsch, Stefan M. Petters, Matthias Pruksch, Sergey Tverdyshev, and Stefan Wallentowitz</i>	179
Inter-seminar workgroup: Software Certification & Multicore Processing <i>Michael Paulitsch</i>	181
Participants	182

3 Overview of Talks

3.1 The ARTEMIS ACROSS project

Christian El Salloum (TU Wien, AT)

License  Creative Commons BY 3.0 Unported license
© Christian El Salloum

The European ARTEMIS ACROSS project aims to overcome the limitations of existing Multi-Processor System-on-a-Chip (MPSoC) architectures with respect to safety-critical applications. MPSoCs have a tremendous potential in the domain of embedded systems considering their enormous computational capacity and energy efficiency. However, the currently existing MPSoC architectures have significant limitations with respect to safety-critical applications. These limitations include difficulties in the certification process due to the high complexity of MPSoCs, the lacking temporal determinism and problems related to error propagation between subsystems. These limitations become even more severe, when subsystems of different criticality levels have to be integrated on the same computational platform. Examples of such mixed-criticality integration are found in the avionics and automotive industry with their desire to integrate safety-critical, mission critical and non-critical subsystems on the same platform in order to minimize size, weight, power and cost. The main objective of ACROSS is to develop a new generation of multicore processors designed specially for safety-critical embedded systems; the ACROSS MPSoC. This talk will show how the ACROSS MPSoC overcomes the limitations of existing MPSoC architectures in order to make the multi-core technology available to the safety-critical domain.

3.2 A Model-based Approach for Optimizing Existing Real-Time Software on Multicore Processors

Michael Deubzer (Timing Architects Embedded Systems GmbH, DE)

License  Creative Commons BY 3.0 Unported license
© Michael Deubzer

In many upcoming real-time system projects multicore processors are an integral part of the roadmap. Till today, software in the embedded industry has been mostly developed for single-core processor systems and represents a high investment for a company. To protect their investment those companies are now faced with the challenge to migrate the software to multicore processor systems. The approach presented in this talk describes a methodology to migrate single-core software to multicore processor systems by regrouping of functions and adding a hardware abstraction layer. The methodology is applied on the architectural granularity of functions which are directly called in tasks of a multitasking system and described by dataflow, execution time and resource demands. In the first step a partitioning heuristic groups functions, considering the dataflow and execution time demands, to tasks and allocates them to cores. In real-time systems two major requirements have to be guaranteed, namely coherency and consistency. This is solved by evaluating data dependencies and creating a middle-ware which copies shared data items of functions to a local memory and writes back data after execution. In order to avoid conflicts at the access to shared resources, a protection mechanism is applied which manages exclusive access and therefore limits the degree of interference between applications. For the timing evaluation of a certain allocation

of tasks to cores, an explorative simulation is applied and timing constraints are checked. This process has been automated and is used for a multi-objective optimization algorithm, searching for best partitioning of functions and allocation of tasks in terms of predefined criteria like reaction times, memory usage or bus traffic. By splitting tasks in smaller subtasks and allocating those to cores a set of software allocations is generated and solutions, fitting the system requirements and configuration best, can be selected.

3.3 IDAMC – A manycore architecture for mixed critical applications

Rolf Ernst (TU Braunschweig, DE)

License  Creative Commons BY 3.0 Unported license
© Rolf Ernst

Joint work of Ernst, Rolf; Jonas Diemer; Philip Axer

URL <http://www.ida.ing.tu-bs.de/en/research/projects/aramis/>

Mixed critical systems integrate application functions of different safety and time criticality. For such systems, safety standards require that the critical functions must adhere to the reliability requirements and are not be affected by the non-critical functions (“freedom from interference”) while the less critical tasks shall be implemented with maximum efficiency. This talk will present a many-core architecture based on a Network-on-chip that provides core isolation and supports dynamic flow control for bounded timing interference. A formal timing model allows formal verification of performance constraints. First results will be presented.

3.4 Commerical Challenges of MultiCores in Automotive Domain

Glenn Farrall (Infineon – Bristol, GB)

License  Creative Commons BY 3.0 Unported license
© Glenn Farrall

While there are many challenging aspects to the deployment of Multicore devices, this talk takes a step back and considers some of the basic implementation issues of Multicore SoCs.

As well as additional die area – many characteristics of devices targeting the Automotive market (especially safety requirements) add to the power budget working against the major driver of multicore usage, namely higher performance per \$ and per Watt. This talk briefly covers several of these issues.

3.5 Timing Predictability of Multi-Core Processors

Christian Ferdinand (AbsInt – Saarbrücken, DE)


License  Creative Commons BY 3.0 Unported license
© Christian Ferdinand

All contemporary safety standards require to demonstrate the availability of sufficient resources to sustain correct functioning of the system. This includes determining safe upper bounds on the worst-case execution and response time of real-time tasks. In mixed-criticality

systems the entire system is subject to the highest occurring safety integrity level unless the independence of all safety functions can be demonstrated in the spatial and temporal domain. These requirements are imposed, e.g., by DO-178B, DO-178C, ISO-26262, IEC-61508, and EN-50128. Since FDA regulations and the German Medizinproduktegesetz require to take into account the state of the art they also pertain to software for medical devices. Spatial independence can be ensured by using partitioned operating systems, or can be proven by static analysis tools which, e.g. can demonstrate the absence of stack overflows or other runtime errors. However, many multi-core processors exhibit characteristics that make it difficult or even impossible to ascertain predictable performance: it may be hard to ensure freedom of interference and to determine safe worst-case execution time bounds. We give an overview of hardware features leading to interference and predictability problems, shows examples of predictability-oriented multi-core configurations, and describe a tool-based methodologies to ensure the correct timing behavior.

3.6 Road to the use of multicore processors in space systems

Massimo Ferraguto (Space Systems Finland Ltd – Espoo, FI)


License  Creative Commons BY 3.0 Unported license
© Massimo Ferraguto

The use of multicore in the space domain can be beneficial in terms of greater processing capability (concentration of multiple functions in one single computer, with partitioning by criticality level and/or function; more payload data processing on-board), weight, power and fuel reduction which ultimately lead to longer lifetime and cost efficiency.

The European space industry is developing the enabling technology to reach the necessary readiness level to be able to use multicore processors in real space missions. The main enabling technologies considered and under development include: multicore processors (Leon 4, etc.), Time and Space Partitioning approach of the integrated Modular Avionics for Space (started from single-core and inspired from the ARINC 653 standard), hypervisor technology (XtratuM, etc.) and SW architecture (SAVOIR-IMA). In particular the Time and Space partitioning of resources is considered to be an essential driver to ensure the predictability needed for critical missions.

3.7 Analysis of Embedded Software for Multicore in the Automotive Domain


Steffen Goerzig (Daimler AG – Böblingen, DE)

License  Creative Commons BY 3.0 Unported license
© Steffen Goerzig

Multicore technology promises reduced energy consumption, reduced package dimensions, and higher performance. But the road to multicore is covered with hazards – race hazards. This is especially true for most of the software as it will be ported rather than re-implemented for multicore platforms. The talk presents current approaches to avoid race hazards in embedded software including technology transfer from academia to industry. First results of automotive applications are shown.

3.8 Multi core – Single bus

Rene Graf (Siemens AG – Nürnberg, DE)

License  Creative Commons BY 3.0 Unported license
© Rene Graf

The use of multi core processors in embedded systems is essential in future designs. Though, system architects have to think about the implications porting a former single core application to a multi core hardware, since most of the peripherals will still remain a single resource.

Even if the application can be easily splitted into independent parts, which work on different peripherals, the bus, e.g. PCI, between processor cores and peripherals will become a bottle neck. Assuming one part of the distributed application running with real time conditions, the latencies that were met on a single core processor may not be met any more due to the interfering bus accesses of the other parts.

The modeling and analysis of such a system in the early phase of development can help to find these implications both in a qualitative and quantitative manner.

The method to analyse those systems is described using a real system with a multi core processor and different peripherals, which are connected by a single PCI bus. Finally, the simulation results are compared with real measurement figures.

3.9 Distilling Programs for Multicore Architectures

Geoff Hamilton (Dublin City University, IE)

License  Creative Commons BY 3.0 Unported license
© Geoff Hamilton

The proliferation of increasingly parallel architectures will have a significant impact on software developers; they can no longer develop software for a sequential architecture and expect performance to improve as the underlying architecture becomes faster. There is therefore a need to develop software that harnesses the power of parallel architectures directly. The development of parallel software is inherently more difficult than the development of sequential software; parallelization of programs by hand is very difficult, tedious and error-prone. By automatically introducing parallelism into programs, the programmer can be freed from explicitly implementing parallelism and can therefore concentrate on algorithmic issues. However, producing automatically parallelized code which is comparable in performance to code which has been parallelized by hand is very difficult, particularly for imperative programming languages.

There has been a recent upsurge of interest in the parallelization of functional programming languages. Functional programs are claimed to be better suited to parallelization than their sequential counterparts for a couple of reasons. Firstly, computations do not involve a shared state, which is problematic for parallel implementations; secondly, execution orders are solely constrained by data dependencies, as opposed to the unnecessary dependencies caused by statement sequences. Also, it is claimed that functional programs are easier to analyze and more amenable to transformation. However, functional programs also have the disadvantage that expressions are often combined using intermediate data structures, which would result in costly inter-process communication if these expressions were to be evaluated in parallel.

In this work, we show how our own program transformation algorithm which we call distillation can be used to transform programs into a form which makes functional programs

more amenable to parallelization and execution on multicore architectures. Distillation is a very powerful source-to-source program transformation algorithm for removing intermediate data structures which can achieve superlinear improvement in the run-time of sequential programs. Programs produced by transformation are in a specialised form called distilled form in which most functions are tail recursive and there are very few intermediate data structures. We show how distillation can be used to convert programs defined over sequential data structures to equivalent programs defined over data structures which are more easily partitioned to facilitate parallel execution. We then show how the resulting programs can be parallelised using Glasgow Parallel Haskell. We argue that this has the advantage over alternative techniques that fewer intermediate data structures are created in the resulting programs, so they can be executed more efficiently.

3.10 Necessity for & Feasibility of a Multicore Ecosystem

Andreas Herkersdorf (Technische Universität München, DE)


License © Creative Commons BY 3.0 Unported license
© Andreas Herkersdorf

Multicore technology overcomes the bottleneck of sequential task execution and provides superior processing performance and power efficiency compared to sophisticated single-core ancestors. Multicore technology also lets industry and academia face entirely new challenges with respect to coping with system complexity. For the time being, the efficient utilization of vast amounts of parallel processing resources relies predominantly on the skills of expert programmers and system architects, but isn't yet accessible for the broad community of software engineers. In the field of embedded and cyber physical systems, multicore processors must satisfy tough requirements with respect to real-time, power efficiency, reliability, safety and security. Methods for multicore system modeling, verification and software debugging, if existing, are specific to an individual processors, but not generically applicable to classes of multicore systems (Would we need a Sync-Point as an enhancement to a Break-Point?).

Finding generic, flexible and scalable solutions to these problems in order to enable multicore on an even broader scale for embedded systems applications may be beyond the skills and capacities of individual enterprises. Therefore, the "Working Group Multicore" within the Bavarian ICT Innovation Cluster BICCNNet proposed establishing a research and development network to jointly tackle these challenges. Through mutual exchange of knowledge and (partially) providing access to solutions in the specific domain of expertise of partners, a multicore ecosystem would gradually evolve. My objective for this Dagstuhl seminar is to stimulate discussions on the feasibility of such an ecosystem, to hear what reservations industry might have and how approaches for an initial instantiation could look like.

3.11 “Heterogeneous Multiprocessing or Just a Bunch of Coprocessors?” – The case for unified programmability

Enno Luebbbers (Intel GmbH – Feldkirchen, DE)

License  Creative Commons BY 3.0 Unported license
© Enno Luebbbers

Heterogeneous multiprocessor systems combine general-purpose processors with specialized (and thus highly efficient) processing units for the acceleration of application-dependent functionality. The increased overall efficiency, however, comes at the cost of programmability, as usually, the application developer needs to be an expert in the respective programming models for the individual accelerators (FPGAs, GPUs, DSPs, ...) in order to exploit the heterogeneous elements to their full potential.

Many approaches exist to cover heterogeneous elements with new or extended existing programming languages and models. In the face of upcoming challenges in embedded systems like openness, extensibility, safety and security requirements and ever-increasing complexity, the question rises whether there is actually a silver bullet, at least for certain application domains, or if we should look at different programming models which at least allow the integration of heterogeneous parts, developed by domain experts, into heterogeneous applications that fulfill the promise in terms of efficiency that heterogeneous platforms have made.

3.12 Fault-Tolerant Time-Triggered Communication Infrastructure for Multi-Processor Systems-on-a-Chip

Roman Obermaisser (Universität Siegen, DE)

License  Creative Commons BY 3.0 Unported license
© Roman Obermaisser

The ongoing technological advances in the semiconductor industry make MPSoCs more attractive, because uniprocessor solutions do not scale satisfactorily with increasing transistor counts. However, higher integration causes more sensitivity w.r.t. energy variations which requires new fault-tolerance measures to overcome the transient fault rates that have significantly increased. In the transient tolerant time-triggered system-on-chip architecture, fault-tolerance mechanisms for application components, communication interfaces and the time-triggered network-on-a-chip are introduced. In addition, a fault injection framework was developed to compare state-of-the-art integrated architectures (e.g., hypervisors such as XtratuM) and the transient tolerant time-triggered system-on-chip architecture. Experiment evaluations have provided evidence for the reliability of the architecture in the presence of soft-errors.

3.13 Multi-Core in Avionics – On Problems and One Technical Approach Monitoring-Based Shared Resource Separation for Commercial Multi-Core System-On-Chip

Michael Paulitsch (EADS – München, DE)

License © Creative Commons BY 3.0 Unported license
© Michael Paulitsch

Multi-core computer architectures are the first choice in consumer electronics. Their performance and power efficiency are also attractive features for safety-critical applications, as in avionics. But increased integration and optimizations for average case performance poses challenges when deploying them for such domains. In the Dagstuhl presentation, we first present visions and problems of multi-core processors. In an exemplary approach towards solving a specific solution, we focus on the problems of temporal indeterminism and fault containment introduced by shared resources such as network on chip and shared memory. Pursuing previous work that quantified the impact of concurrent usage of shared resources, targeting the integration of mixed-criticality applications on the same platform, we propose a partitioning approach to control those interferences. We present a partitioning concept, which is further used to develop a modified worst-case analysis for multi-core processors. For evaluation we use representative benchmarks of the EEMBC Autobench benchmark suite on the Freescale 8-core PowerPC P4080.

3.14 Talk on ARTEMIS Project RECOMP

Michael Paulitsch (EADS – München, DE)

License © Creative Commons BY 3.0 Unported license
© Michael Paulitsch

“RECOMP” stands for Reduced Certification Costs Using Trusted Multi-core Platforms and is a European funded project from ARTEMIS JOINT UNDERTAKING (JU). The project started April 1st of 2010 and has a duration of 36 months.

RECOMP research project pretend to form a joint European task force contributing to the European Standard Reference Technology Platform for enabling cost-efficient certification and re-certification of safety-critical systems and mixed-criticality systems, i.e. systems containing safety-critical and non-safety-critical components. The aim is establish methods, tools and platforms for enabling cost-efficient (re-)certification of safety-critical and mixed-criticality systems. Applications addressed are automotive, aerospace, industrial control systems, and lifts and transportation systems.

RECOMP recognizes the fact that the increasing processing power of embedded systems is mainly provided by increasing the number of processing cores. The increased numbers of cores is a design challenge in the safety-critical area, as there are no established approaches to achieve certification. At the same time there is an increased need for flexibility in the products in the safety-critical market. This need for flexibility puts new requirements on the customization and the upgradability of both the non- safety-critical and safety-critical parts. The difficulty with this is the large cost in both effort and money of the re-certification of the modified software-

3.15 Talk on German Project ARAMiS – Automotive, Railway, and Avionics Multi-Core Systems

Michael Paulitsch (EADS – München, DE)

License  Creative Commons BY 3.0 Unported license
© Michael Paulitsch

ARAMiS (Automotive, Railway and Avionic Multicore Systems) aims at the development of concepts for multi-core processors in automotive, railway and avionics to reach a gain in safety, comfort and efficiency. In current aircrafts or cars only single-core processors are used since only their functionality can be certified according to domain specific safety-standards. But these single-core architectures cannot reach the performance needed for future applications and are getting obsolete. To develop efficient multi-core architectures several research institutions and manufacturers from the automotive, railway, and avionics domain, their suppliers as well as hardware and software producers are working together in the ARAMiS project.

The project involves the following partners: AbsInt, Airbus, Audi, BMW, Bosch, Casidian, Continental, Daimler, Diehl, EADS, Freescale, Infineon, Intel, Liebherr, OpenSynergy, Symta Vision, Vector, Wind River and various research institutions (Technische Universität München, Technische Universität Braunschweig, Karlsruher Institut für Technologie, Universität Stuttgart, Technische Universität Kaiserslautern, Christian-Albrechts-Universität zu Kiel, Universität Paderborn, Fraunhofer IESE AISEC, Offis, Fortiss).

3.16 Fine grained process migration for MPSoCs

Sri Parameswaran (UNSW – Sydney, AU)

License  Creative Commons BY 3.0 Unported license
© Sri Parameswaran

Process migration (PM) is a method used in Multi-Processor System on Chips (MPSoCs) to improve reliability, reduce thermal hotspots and balance loads. However, existing PM approaches are limited by coarse granularity (i.e. can only switch at application or operating systems boundaries), and thus respond slowly. Such slow response does not allow for fine control over temperature, nor does it allow frequent migration which is necessary in certain systems.

In this work, we showcase Thor, an approach which is a fine-grained reliable PM scheme, for Embedded MPSoCs, to overcome the limitations of existing PM approaches. Our approach leverages custom instructions to integrate a base processor architecture, with PM functionality. We have proposed three schemes, Thor-BM (migration at basic block boundaries), Thor-BM/CR (migration at basic block boundaries with checkpoint and recovery), and ThorIM/CR (migration at instruction level with checkpoint and recovery). Our main motivation is to realize a fine-grained PM approach beneath the OS level within the local processor architecture. Performing locally, and without the use of the OS, results in short initiation Time. If such a scheme were to be implemented, then this would let Dynamic Thermal Management techniques (especially those with policies relying on frequent task migrations, improve overall performance while maintaining low peak temperatures. Such a scheme would allow for a fast process migration in the presence of faults. It is also possible for fast load balancing scenarios to take place. Our experiments show that the execution time overhead is less than 2%, while the additional area cost and power consumption costs are approximately 50% (excluding main memories, which if taken into account would substantially decrease this overhead). The average migration time cost is just 289 cycles.

3.17 Task Mapping for Manycore-based Embedded Real-Time Systems

Stefan M. Petters (ISEP-IPP – Porto, PT)

License © Creative Commons BY 3.0 Unported license
© Stefan M. Petters

Joint work of Nikolic, Boirslav; Petters, Stefan M.;

Main reference B. Nikolic, S.M. Petters, “Application Mapping In NoC-Based Many-Cores,” Technical Report, HURRAY-TR-121201, 2012.

URL <http://www.cister.isep.ipp.pt/docs/733/>

Manycores-based processors are clearly on the agenda for their eventual deployment in embedded systems. While the widespread usage of manycores is still a few years into the future, it is worth spending now some effort in considering implications and requirements for this deployment. A current research activity in the area of operating systems for such processors is the Barrelfish OS. Barrelfish operates on a limited migrative model, where a task can only migrate within a subset of cores. Within the talk, various challenges in the mapping process have been discussed, when it comes to the communication within the dispatcher entities of an application, as well as between applications. In particular the notion of proxies to simplify the analysis process of the communication delay has been introduced. Since an exhaustive search for feasible mappings is of exponential complexity, the talk presented a heuristic, which allowed via one parameter to control the mapping complexity and via another parameter, the greediness of high priority applications when it comes to chip area. A set of experiments showcase the complexity of the approach, as well as the impact of the two parameters on the mapping result.

3.18 Sustainable Development of Software in the Multi-Core Age

Matthias Pruksch (sepp.med – Röttenbach, DE)

License © Creative Commons BY 3.0 Unported license
© Matthias Pruksch

In order to benefit from progress in hardware, software development faces a paradigm change to parallelism. The impact is even more important, since life-cycle of software is much longer than that of hardware: investments in software last for decades. In addition, software now plays a crucial role for system design. Model based methods for development and quality assurance show the prospect to master the increasing complexity of such systems. Notably, if you are working in a context of products that have to be certified.

3.19 Chances and risks for security in Multicore processors

Georg Sigl (Technische Universität München, DE)

License © Creative Commons BY 3.0 Unported license
© Georg Sigl

The main security challenges in future systems in the application areas such as automotive, transport, industry automation, health care, smart grid are:

- Security for 10 and more (30) years.
- Secure autonomous interaction of heterogeneous machines (M2M).
- Protection against manipulation and misuse.

- Fulfilling security requirements while keeping real time requirements.
- Consider resource limitations.
- Managing increasing complexity in embedded systems.
- Protection of intellectual property (hardware and software) in embedded systems against counterfeiting.
- Support of adaptation of cyber physical systems through securely adaptable embedded systems.

In order to fulfill these requirements in the future, secure elements will be integrated in the systems in order to provide security services while still being resistant even against hardware attacks. Hardware attacks can be classified into probing attacks, which try to extract information out of a chip by probing internal signals or by forcing values. Side channel attacks observe power consumption or electromagnetic radiation and fault attacks inject faults into the chip, e.g. through spikes on the current supply or light. Examples for systems with secure elements are cars with secure elements as investigated in the German funded project SEIS, the smart meter gateway solution as specified by the German BSI (Bundesamt für Sicherheit in der Informationstechnik) or mobile phones with up to three secure elements, the SIM a secure element soldered in the phone for NFC payment and a SD-card with secure element provided by a bank.

In multicore systems we have alternative solutions for solving security problems which are nowadays solved with a secure element. In order to detect fault attacks a very proper means is redundancy, which can be implemented in multicore systems very well. Security critical tasks can be parallelized and the results can be checked afterwards for correctness. A very good countermeasure against side channel attacks is randomization, which increases the effort for the attacker to observe critical operations. In multicore systems the execution of tasks is randomized by default and can be even increased by actively assigning parts of tasks to different cores with changing degrees of parallelization. Another way to counteract side channel attacks is the implementation of secret sharing schemes which avoid the use of a complete secret key on one core but distribute it to many cores. Separation is another important security measure, which may be easier on multicores as long as the underlying architecture supports that. If there are too many shared resources this could however be also a security risk. With multicores it may be even possible to assign the role of a secure element flexibly to one or more of the standard cores, which the responsibility to monitor the behavior of the system and to provide security services.

Multicores enable creation of much more complex systems than today's processors. Increasing complexity usually increases the risk of vulnerabilities like denial of service, undetected malware, or buffer overflows. The reason for these vulnerabilities is resource sharing, lack of monitoring or control and badly separated software. Another risk is side channel attacks which may be executed on multicore systems through software which is executed on the same system on chip.

Overall multicore may offers opportunities to improve the security of embedded systems, where we have currently a lot of open and unsolved problems.

3.20 Isolation of Cores to Support Development of Mixed Critical Systems

Claus Stellwag (Elektrobit Automotive – Erlangen, DE)

License  Creative Commons BY 3.0 Unported license
© Claus Stellwag

The major issue when using multi-core controller in embedded devices is the huge amount of legacy code developed in former times. This code is normally well suited (“proven in use”) but not aware of execution parallelism and therefore cause problems when being executed on multi-core controller. Reasons for this behavior are: fast interrupt locks, implicit communication (e.g. activation order of threads) or cooperative scheduling (thus avoiding the use of explicit locks for shared resources) among others. Additionally new requirements have to be considered in the design (e.g. minimize energy consumption) and the use of standards is forced (AUTOSAR). If safety relevant software has to be executed on the same device with standard QM (or legacy) software the designer has to make sure that the safety part is not influenced from non-safety parts.

One common idea when migrating software from single to multi-core is to perform a redesign and split up threads to the different cores. Experience showed that this is possible but causes lot of work, e.g. all implicit communication or locking have to be considered and sometimes must be made explicit. Also the proof that the new partitioning of the redesign is free of errors is difficult - especially from the safety viewpoint. Therefore a different approach is presented here.

Central idea is to focus on the separation and isolation of cores. This means that existing applications should not be split but kept together. Still multiple applications can run on one device, but each one mapped to its own core. Communication between the cores shall be minimized in order to get the maximum performance. Safety applications get with this mapping their own core and (if the hardware supports isolation) can be completely protected from non-safety software. One controller family which supports this is the Infineon AURIX. On this chip the cores and their RAM can be configured in way that only read but no write access from other cores is possible.

As an example just consider an AUTOSAR system where normally the safety and non-safety applications (SWCs) can be separated, but all the basic software is shared. This means that all basic software must be developed according the related safety standard. If isolation is used one core can use a standard AUTOSAR system with all the non-safety code. The other core can be used for safety related software only. If required a core-to-core communication module can offer an exchange mechanism between the cores. The safety core might also need some basic software, but typically the amount of such modules for safety is very low. In general it is always better to limit the amount of safety related software to handle the complexity. Within one EU funded project (RECOMP) Elektrobit implemented this isolation approach for the AURIX. The implementation was used by Delphi (automotive supplier) to demonstrate an electrical steering column lock. The demonstrator showed that the isolation method works.

3.21 Safe(r) Loop Computations on Multi-Cores

Jürgen Teich (Universität Erlangen-Nürnberg, DE)

License © Creative Commons BY 3.0 Unported license
© Jürgen Teich

Main reference J. Teich, W. Schröder-Preikschat, A. Herkersdorf, “Invasive Computing - Common Terms and Granularity of Invasion,” arXiv:1304.6067v1 [cs.OS]

URL <http://arxiv.org/abs/1304.6067v1>

The necessity of satisfaction of non-functional constraints such as guaranteed data processing throughputs, deadline reactive processing or safety properties on the correctness of computational results is of utmost importance for the successful introduction of multi-core technology in many future embedded system products.

In this visionary introduction, we treat the problem of architectures, methods and tools that allow a developer to specify a certain safety level for a quite general and important class of loop computations. Loop programs are known to be quite amenable to parallel processing and are typically also quite scalable. However, no existing work is known to us how to make loop computations safe so to guarantee the correctness of the corresponding computed results of a loop program at run-time.

In this realm, we propose first ideas how, dependent on a specified safety level, the core allocation might be properly controlled for allowing concepts such as DMR and TMR known for single processor systems to loop computations on multi-cores including the way how deterministic voting may be efficiently implemented on a class of domain-specific multi-core architectures called tightly-coupled processor arrays (TCPAs).

We conclude how these concepts of redundant in-sync loop computations might be nicely supported by the recent parallel computing concept of invasive computing.

3.22 parMERASA- Multi-Core Execution of Parallelised Hard Real-Time Applications

Theo Ungerer (Universität Augsburg, DE)

License © Creative Commons BY 3.0 Unported license
© Theo Ungerer

URL <http://www.parmerasa.eu/>

Providing higher performance than state-of-the-art embedded processors can deliver today will increase safety, comfort, number and quality of services, while also lowering emissions as well as fuel demands for automotive, avionic and automation applications. Such a demand for increased computational performance is widespread among European key industries. Engineers who design hard real-time embedded systems in such embedded domains express a need for several times the performance available today while keeping safety as major criterion. A breakthrough in performance is expected by parallelising hard real-time applications and running them on an embedded multi-core processor, which enables combining the requirements for high-performance with time-predictable execution.

The talk will discuss preliminary results of the EC FP-7 project parMERASA (Multi-Core Execution of Parallelised Hard Real-Time Applications Supporting Analysability, started Oct. 1, 2011). The project targets timing analysable systems of parallel hard real-time applications running on a scalable and predictable multi-core processor with up to 64 cores. We target in particular future complex control algorithms by parallelising hard real-time

application programs to run on multi-/many-core processors. Application companies of avionics, automotive, and construction machinery domains cooperate with tool developers and multi-core architects to reach the project objectives.

3.23 OpTiMSoC – An Open Source Experimentation Platform for Multicore

Stefan Wallentowitz (Technische Universität München, DE)

License © Creative Commons BY 3.0 Unported license
© Stefan Wallentowitz

Joint work of Stefan Wallentowitz, Philipp Wagner, Michael Tempelmeier, Thomas Wild, Andreas Herkersdorf
Main reference S. Wallentowitz, P. Wagner, M. Tempelmeier, T. Wild, A. Herkersdorf, “Open Tiled Manycore System-on-Chip,” arXiv:1304.5081v1 [cs.AR].
URL <http://arxiv.org/abs/1304.5081v1>

Future System-on-Chip will integrate an increasing amount of processing elements. Tiled manycore System-on-Chip are a promising approach to organize future platforms. In such platforms a Network-on-Chip connects “tiles” of processing elements, memories and I/O, often as a mesh. For example Intel has presented the “Single Chip Cloud Computer”.

Research of such platforms and especially prototyping rely on building such a platform or is bound to simulation. At LIS we develop a framework for prototyping of such tiled manycore System-on-Chip: Open Tiled Manycore System-on-Chip (OpTiMSoC). It is based on open source components and itself freely available. Essential elements and target platforms are part of the library and a platform generator tool is in development to allow for fast creation and implementation of platforms.

This talk gives an overview of OpTiMSoC and presents the current status and roadmap of the project.

3.24 Efficient observation of Multicore SoCs

Alexander Weiss (Accemic GmbH & Co. KG – Kiefersfelden, DE)

License © Creative Commons BY 3.0 Unported license
© Alexander Weiss

Comprehensive observability of multicore System-on-Chip (SoC) is the basis for efficient debugging, especially for the analysis of root causes of non-deterministic failures. Furthermore, it is also important for the detection of race conditions, the measurement of WCET, cache and memory layout optimization as well as different kinds of coverage measurements. Solutions for multicore SoC observation can be rated by the completeness of accessible information, which includes executed instructions, clock cycle accurateness, data access (value, address, direction), cache and bus operations. This information should be captured in parallel for multiple CPUs and other bus masters. Observation should be long-time, non-intrusive, available in real-time and applicable for mass-produced SoCs. By using multicore SoCs the traditional computation-centric observation strategy of single core SoCs has to be amended by communication-centric observation. Today’s solutions are software instrumentation and embedded trace, both with limitations especially in the multicore area. The bottleneck is the required bandwidth for trace data output, which increases superlinear with the number of CPU cores. With hidICE a new observer based approach for full visibility of internal states

of multicore SoCs was developed. The fundamental idea behind hidICE is to equip the SoC with a facility that allows the synchronization with an external emulator. Thus, only data is communicated from the SoC to the emulator that is required to reconstruct all internal data and the program flow. All other system responses are defined by the program code. The advantage of this approach is the fact, that in most cases the bandwidth required for synchronization is significantly lower than the bandwidth required for traditional trace data output. The emulation provides full access to all internal CPU and bus states, including CPU register trace or bus trace with the deep view as known from a logic state analyzer. The hidICE approach was evaluated for different CPU architectures, including a LEON3 based multicore system. This implementation requires a very low gate count and provides full, real-time, continuous, and concurrent observation of all CPU cores. Another challenge in multicore SoC observation is an efficient strategy to handle the huge amount of trace data, accessible from multicore SoCs. The traditional offline analysis has to be complemented and replaced by online analysis approaches, such as runtime verification.

3.25 Many cores – many problems

Reinhard Wilhelm (Universität des Saarlandes, DE)


License  Creative Commons BY 3.0 Unported license
© Reinhard Wilhelm

The embedded-systems industry goes multi-core. The main reason is the good performance per consumed-energy ratio. For time-critical systems, this transition is problematic. No sound and efficient method for the verification of timing constraints of embedded systems executed on multi-core platforms exists. This problem is difficult and can not be solved by the established methods due to the interference on shared resources. There is a proposed method for timing analysis: 1. analyze an application by established single-core methods assuming that the access to shared resources happens instantaneously, then 2. add a safe bound on the delays of all accesses as indicated by a given abstraction of the access behavior of all co-running applications.

Current core designs do not allow this approach since they are not timing compositional. In addition, the currently used abstractions seem to be too coarse as to allow precise bounding of access delays. The MULCORS report, Use of Multicore Processors in airborne systems, submitted by Thales Avionics to the European Aviation Safety Agency is considered a document of capitulation in the face of the problem. It recommends to use measurement-based, unsound methods for timing verification of safety-critical and mixed-critical avionics systems implemented on COTS multi-core architectures. They ignore both the state of the art in single-core timing analysis as well as fundamental problems of measurement-based approaches. For example, they recommend using corrective factors to unsafe measurement-based WCET estimations based on an identification of worst-case perturbances, ignoring the high likelihood of domino effects on such architectures.

3.26 High-level Simulation-based Design Space Exploration on Multicore Virtual Platforms

Thomas Wild (Technische Universität München, DE)

License  Creative Commons BY 3.0 Unported license
© Thomas Wild

For an efficient simulation based performance assessment, instruction set simulators (ISSs) cannot meet the simulation performance requirements to be applied in the exploration of the macroarchitecture of multicore systems-on-chip (MPSoCs). In order to speed up the exploration it is essential to use more abstract simulation models, which limit the number of simulation events and yet deliver accurate and indicative insights into the architecture to assist the designer in finding an optimal solution.

This talk presents McSim, a SystemC based high level simulation tool, which allows the use of trace driven as well as compiled binary level simulation models. In a trace model, internal structure and functionality of a processor core are abstracted in time interleaved reads and writes, which are replayed and superimposed on the shared resources of the MPSoC architecture. A binary simulation model includes in addition the execution of the functionality, allowing the simulation of flexible input stimuli for the processor. Experiments with various benchmark applications from MiBench and MediaBench show - compared to SimpleScalar ISS - an increase of the simulation performance of a factor of up to 200 and 30, respectively. The maximum error of the two models relative to SimpleScalar are 15% and 10%. Architecture exploration with McSim consists of two phases: A one time generation of either the trace or the binary level model (which includes running the ISS). During the actual exploration phase, in each iteration the trace and binary level models are executed thus profiting from the acceleration compared to the ISS. The adaptation of HW/SW architecture and of task mapping/scheduling for the solution alternative under investigation is done via XML files.

4 Working Groups

4.1 Specification & Interference

Claus Stellwag, Michael Deubzer, and Glenn Farrall

License  Creative Commons BY 3.0 Unported license
© Claus Stellwag, Michael Deubzer, and Glenn Farrall

The workgroup focuses on the topics specification and interference but during the discussions also other topics were touched. The following topics (A, B, C) were considered to be most critical for a wide use of multi-core architectures in cyber physical systems.

A) Application Specification

It is of central importance that an application specification exists which details different aspects of the application design. Especially the document should:

1. include all requirements include non-functional ones (e.g. requirements on timing).
2. enable legacy code integration and shall explain how this is performed/which rules are applied.
3. help to avoid and safely bound interferences on shared resources.

Furthermore, as a base a well-defined software component model is needed. Basic characteristics shall include dependencies between components and interface to e.g. global variables which are accessed by the component. An example for such a model can be found in AUTOSAR. Depending on the design stage the information will be updated or extended. Example: During top down design the timing constraints are given and budgets are assigned to software components, which can be later verified when the executable of the application is available (e.g. by static analysis or tests on the final hardware). Standard component models talk about functional interactions. But the models shall also be able to include meta information e.g. by specifying memory interference, WCET, options to map the components to different cores, or energy behavior. This would greatly simplify the composition of applications in multicore architectures because it is assumed that the component developer and the integrator of the system are different persons. This way also parts of the design knowledge can be saved within the model and later reused e.g. to find the best mapping of components to cores.

Requirements shall contain all necessary properties like timing constraints, safety requirements, data loss constraints, and criticality information.

It would be very helpful to use specification guidelines which help the software architect to structure the specifications and to make sure that no important aspects gets lost. Such guidelines should

- be built upon a well-defined model of computation (interacting components)
- make all component dependencies explicit
- classify variables into configuration, start-up and volatile
- provide memory partitioning on an appropriate level of abstraction (local to process, shared between subsets of processes, global)

A compiler can map this to NUMA shared memory architectures. Depending on the real needs security aspects shall be listed. (E.g. a security architecture could add monitors)

B) Legacy Code

There is no easy way to migrate existing code to multicore controllers. Most code has implicit assumptions about the single core behavior, e.g. implicit communication between cooperative threads. Ideally depending on the original design the effort of the changes can be estimated. Unfortunately in many cases the know-how of this original design is lost (e.g. responsible people have left or retired) and therefore the fear of changes is huge. (“don’t touch it, it’s a piece of art”). The following four levels have been discussed (effort grows from low to high)

1. Small changes are often possible because their impact can be foreseen. (E.g. software unit internal behavior which is not visible from outside). Additional abstraction layers and interfaces to shared resources help to limit the degree of interference between software components.
2. Mapping of existing SW to cores. This approach maps parts of the existing software to separate cores and tries to minimize (or even omit) the communication between cores and avoid sharing resources with other cores. If resources are shared (e.g. I/O) it has to be handled in special way. Since the software runs only on a single core most of the old behavior can be established without modifications (e.g. implicit communication will work since all relevant communication processes are on the same core)
3. Redesign. Completely redesign (central) parts of the application based on a clear model of computation and interaction that exposes (at least) the required degree of concurrency.

4. New development. Some industries (e.g. signal processing, mobile communications) have to rewrite completely the application in a well-defined way, e.g. by model of computation that reveals concurrency and interaction (e.g. data flow models). In these domains, clearly defined design flows are available for multiprocessor platforms that optimize and guarantee timing, energy, and memory.

C) Virtualization

Virtualization could help to migrate software to multicore architectures if it would talk about real-time and not virtual time. In cyber physical systems programs typically have dependencies to real-time, this is different to IT systems (e.g. servers) where the timing within a virtualized software is not that important. Practically virtualization is quite established in PC like environments, but only very rarely used in embedded systems.

As an example: just consider an algorithm which performs an analog to digital conversation. This piece of software must know when it is time to convert the next analog value. If this time is missed there will be no result. Supporting real-time in virtualized environments is currently not considered in existing solutions and might also impact the used hardware (e.g. when peripherals are virtualized).

4.2 Industrial Perspective on MultiCore Motivations and Challenges

Glenn Farrall, Christian Ferdinand, Massimo Ferraguto, Steffen Görzig, Michael Paulitsch, Matthias Pruksch, Claus Stellwag, Sergey Tverdyshev, and Alexander Weiss

License © Creative Commons BY 3.0 Unported license
© Glenn Farrall, Christian Ferdinand, Massimo Ferraguto, Steffen Görzig, Michael Paulitsch, Matthias Pruksch, Claus Stellwag, Sergey Tverdyshev, and Alexander Weiss

There were two questions asked of industrial participants to the “Multicore Enablement for Embedded and Cyber Physical Systems” seminar.

For each question there is a summary of the response, and then for completeness the full set of text is provided.

Q1: What is the major motivation for *using* multicore devices — both today and in the future? (where using can be “implement a product with” or “supply tools and/or services to support”)

There were two main answers to the first question. This is not surprising since the motivations for multicore usage have high commonality and are aligned with the realities working against performance of single core devices increasing ad infinitum.

Availability Lack of availability of single core devices was one strong answer. There is a clear expectation that eventually there will only be multicore devices available (above a certain performance threshold) — this answer has been summarised as TINA (There Is No Alternative)!

Derived from product requirements The drive for new features, or step changes in capability demand the extra performance or the extra performance per unit (of power, weight, volume, etc.) that multicore devices offer. As should be obvious from the considered markets (those with Cyber Physical aspects) included in features required are safety and availability, and these also are attractively enhanced with multicore devices.

Collated feedback to question 1 with attribution (in alphabetical order):

Christian Ferdinand (AbsInt) Even for highly safety-critical applications, multi-core processors seem to promise better performance. Therefore, many of AbsInt's potential customers are exploring the possibilities. Typical COTS multi-core processors use shared memory and shared memory buses/networks. Such architectures introduce a high degree of resource sharing that would not be there in a distributed memory architecture. This additional resource sharing can lead to large interference effects between cores. This complexity has created a new demand for timing verification tool support not only for the highest criticality classes.

Massimo Ferraguto (Space Systems Finland) The use of multicore in the space domain can be beneficial in terms of greater processing capability (concentration of multiple functions in one single computer, with partitioning by criticality level and/or function; more payload data processing on-board), weight, power and fuel reduction which ultimately lead to longer lifetime and cost efficiency.

Steffen Görzig (Daimler) TINA (There Is No Alternative). When you can only buy multicore processors in the market, your only choice is to switch cores off...

Michael Paulitsch (EADS)

- Quest for more compute power due to new application demands.
- Wish to reduce size, weight and power of computing for improved performance of aircraft and improved environmental performance.
- Tighter integration leads to the need of more powerful central compute platforms
- wish to use COTS chips for reasons like possible lower cost (COTS are likely multicore devices or SoC)

Matthias Pruksch (sepp.med) Customers, who are the customers of sepp.med customers, demand ever increasing value of products in terms of quality, functionality and interoperability. To achieve this, many products gain from smart acquisition, control and handling of information. Savings in space, weight and power as well as added value by cyber physical systems are just two important drivers to name. Multicore devices offer the prospect to sustain the increasing demand for computing and communication performance. The fundamental switch to multicore architectures lead to a paradigm shift in software development and has a tremendous impact on the installed base of legacy software and how new software has to be written in order to be sustainable. Specifically, regulated domains like medical or avionics are challenged by current multicore implementations that neither fulfil stringent requirements for predictability and absence of interference, nor enable certified legacy software to be migrated without expensive reengineering and costly re-certification. sepp.med is highly interested to provide qualified services for their customers and partners from consulting, over development and quality assurance up to certification.

Claus Stellwag (Electrobit Automotive) The increasing complexity and the rise of new functions (Automotive: e.g. advanced driver assistance functions which build the base for autonomous cars in the future) requires a lot more performance. Typical embedded controllers with only some megahertz will not be able to solve this performance gap. On the other hand the embedded environment (no active cooling of controllers, EMV, etc.) forbids the usage of standard (PC like) processors. So the only way to deliver the required performance and keeping the price affordable are multicore controllers.

Sergey Tverdyshev (SYSGO) The competition on the market drives companies to innovate. This innovation includes developing new functionalities, higher utilisation of available resources, sinking costs. Probably these are the main three reasons driving industry to

adopt multicore microprocessors.

SYSGO is one of the leading embedded RTOS providers and is constantly researching and improving support for multi-core systems. The following are some of the reasons behind this work:

- Customer demands for support multicore systems including system-level architectural support (e.g. AMP, SMP)
- OS support to maximise utilisation of HW resources
- Increasing RTOS/OS performance which is transparent to the user
- Hope to increase dependability for high-criticality systems with multicore

Alexander Weiss (Accemic)

- traditional computation-centric observation has to be amended by communication-centric observation
- multicore possible increases the amount of non-deterministic failures
- traditional storage and offline analysis of trace data gets more and more limited, online trace data decompression and computation (for run-time verification, WCET, race conditions, profiling, ...) seems to be the next step in tool evolution → new generation of tools are required

Q2: What would be a key enabler to making their usage easier, or more prolific or perhaps more profitable?

The second question had a much wider range of answers — as the challenges are not as neatly encompassed as the first question on motivations was by physics.

- The timing behaviour of multicores is consistently raised as a challenge which needs to be made easier to cope with. So mechanisms for interference reduction (or elimination as a goal) are definitely required for enhanced usage in Avionics and other safety critical domains.
- Documentation of existing behaviour (especially of interference or arbitration conditions) is also clearly in a poor state today and improvements in this area would help (or at least add confidence) to any safety case made on COTS based systems.
- Architectures that would allow software to port seamlessly from singlecore to multicore (of any number) are also on the wish list. This has a clear economic advantage — most systems are not created from scratch, but involve reuse of existing code. The investment in this existing code can be very significant and very few products will start from a clean sheet. So either an architecture or tooling to enable this migration would enhance deployment.

Collated feedback to question 2 with attribution (in alphabetical order):

Christian Ferdinand (AbsInt) Interferences complicate timing verification. The adoption of multi-core processors in highly safety-critical applications could be helped by providing support for processor configurations that reduce interferences and a clear documentation of the resource conflict resolution mechanisms

Massimo Ferraguto (Space Systems Finland) The main enabling technologies considered include: multicore processors (Leon 4, etc.), Time and Space Partitioning approach of the integrated Modular Avionics for Space (started from single-core and inspired from the ARINC 653 standard), hypervisor technology (XtratuM, etc.) and SW architecture (SAVOIR-IMA). In particular the Time and Space partitioning of resources is considered to be an essential driver to ensure the predictability needed for critical missions.

Steffen Görzig (Daimler) Methods and tools to migrate old software to multicore without adding errors.

Michael Paulitsch (EADS) Helping would be anything that addresses the limiting points below

- tight integration of function blocks on SoC with limited ability to control and monitor their behaviour
- complexity of SoC and fear of design faults and possible limitations of mitigation or getting detailed design info to argue correctness
- complexity of SoC and ability to holistically understand it
- gap between average and worst-case performance increases
- less ability to control at SoC system level (guaranteed switch off cores, controlled access to shared resources ...)
- increasing gap between COTS SoC design environment and avionics design environment

Making use easier:

- having WCET in mind (does not necessarily have to be the centre of focus)
- access to essential details of chip design

Matthias Pruksch (sepp.med) First objective is to get legacy programmes running without touching the code. Therefore, a mandatory key enabler is access to sustainable multicore devices that provide predictability without sacrificing performance: lockstep mode is no solution. This enables the consolidation of previously separate devices of mixed criticality into one multicore system. Second objective is to define requirements and design guidelines for sustainable software development, e.g. scalability by number of cores. This means, by the increasing number of cores, performance improvements can be realized. Integrated development environments and tool chains are needed to tackle the complexity in terms of an increasing number of software components and constraints and to support design space exploration. Model driven development (correctness by design, formal methods ...) and model based testing show the prospect to handle those aspects. In our opinion, solutions will bring us to the next level of system creation towards hardware/software co-design, with the benefit of faster time-to-market, improved reuse and conservation of resources.

Claus Stellwag (Electrobit Automotive) Key enabler will be the handling of the new complexity (real parallelism). Devices with a clear partitioning approach can help as well as good tools. Some areas need to be developed further (e.g. shared resources and the access times to it).

Sergey Tverdyshev (SYSGO) However, there are obstacles which hinder wide adoption of multi-core in safety critical domains:

- The state-of-the-art COTS multi-core design is driven by cost reduction and average performance optimisation. This lead to the gap between worst and best performance is increasing and predictability of the system behaviour is decreasing. Lesser predictability mitigates advantages of MC in medium to high critical systems.
- The state-of-the-art COTS multi-core architecture focus on “simply adding” new cores. This increases communication load on interconnects and peripheral devices making them the truly bottlenecks for safety and sometimes security.
- The lack of in-depth documentation on COTS hardware makes it impossible to mitigate HW deficits in software on OS, middleware, RTE, or application levels.
- The lack of acceptance by certification agencies

The current situation in *safety* critical area is similar to a round-dance around a huge-camp fire where dancers see or believe to see something very precious in the middle and

while dancing trying to figure how to get it out with being burned.

Interestingly the security critical domains are not that hard affected by safety issues. In these domains MC is widely used (at least in prototypes), especially in the systems without physical access to the hardware for attackers. The most of the problems are similar to single core systems, e.g. lack endurance/evidences that the produced COTS hardware is indeed the one which is described in documentation and does not contain malicious changes.

Alexander Weiss (Accemic) All the observation issues are very important. Not only the common instruction and data traces, also

- cycle accurate trace options for all devices
- trace of all bus masters (not only CPUs)
- easy differentiation on data trace between read and write access without the need for computing the instruction trace in parallel (ARM!)
- access to all information to observe scenarios as listed in Michael's paper [1, table 1]
- high bandwidth trace ports (in combination with smart port replacement approaches)

References

- 1 O. Kotaba, M. Paulitsch, J. Nowotsch, S. Petters, H. Theiling, *Multicore In Real-Time Systems – Temporal Isolation Challenges Due To Shared Resource* Workshop on Industry-Driven Approaches for Cost-effective Certification of Safety-Critical, Mixed-Criticality Systems (WICERT); part of DATE 2013. Grenoble, France. 2013

4.3 Certification of Safety-Critical Multicore Systems: Challenges and Solutions

Stefan M. Petters and Rene Graf

License © Creative Commons BY 3.0 Unported license
© Stefan M. Petters and Rene Graf

Certification in general is the process of a certification agent attesting to certain properties. Depending on the domain, the procedures vary substantially. In most cases certification applies to a complete system, which includes hardware, software and tools to build and validate the artefact. However, for example in the automotive domain, the concept of “certification-out-of-context” implies that some components may be certified in isolation. In general the current landscape in the area of systems certification within the automotive domain, industry automation and aerospace domains is dominated by a very conservative mind-set. Anecdotal evidence suggests that at the time of writing at least one dual-core system is in the certification process and the positive outcome of the process is not yet assured. This is driven by a number of reasons. Most notably there is, for obvious reasons, non-existent to negligible experience with the certification of multicore systems. This is exacerbated by the lack of answers to some of the technological challenges and consequently the lack of a reasonable safety argument. In particular the technological challenges with the deployment of multicores often presents substantial hurdles for the migration of legacy systems on Commercial-of-the-Shelf (COTS) based multicore platforms. The latter is driven primarily by the changed paradigm of performance gains which are now achieved via parallelism. This parallelism is in most cases not exposed in legacy systems. Even legacy applications using multiple threads are not verified running on a multicore processor.

One of the challenges identified within the working group is the increase of complexity in multicore systems. On the one hand this stems from the concurrency, as well as the fine grained resource sharing present in multicore, on the other hand the integration of additional components into the die which were previously accessible individually. Both concerns mean the traditional divide and conquer approach of looking at individual components and reasoning about the interaction between components in the analysis and safety argument are no longer possible. In order to mitigate the issue, it is of paramount importance to identify all resource sharing whether it's explicitly or implicitly shared. Once this identification is complete, platforms need to provide isolation as far as reasonably possible. The isolation properties can be achieved by either separation or duplication of the respective shared resource. For example, for caches, it would be of large benefit to avoid shared caches but rather have the possibility to partition this or use it as scratchpad, which is at least in some COTS processors available. However, shared caches or scratchpad memories may be needed in real SMP applications, where communication and synchronisation of threads need to be performed across several cores. In general it is also preferable to have the ability to control parameters and access to the shared resources in a fine grained manner.

For those resources, where strict isolation is not possible, like for example the memory interface, it is important to assure a fair and/or deterministic arbitration scheme. This would enable to reason about the maximal interference suffered by the core under investigation that is caused by applications running on the other cores. To validate the interference the platform needs to expose enough observability. With current technology, the integration of non-intrusive debug interfaces on individual cores and shared resources presents the most promising path to achieve the required observability.

While there are a number of academic approaches available to provide predictable and analysable HW platforms, their deployment in real-certified systems is very limited. This is driven by the differences at both ends of the spectrum, where current certification practice is focused on single core platforms while industrial pre-development is eyeing the computing capabilities of modern COTS based multicore systems, to implement a plethora of new system functionalities. The COTS issues can be mitigated to some degree by a working close communication with the HW vendors. However, such a communication base is not achieved on short notice but needs to be developed over years. The alternative is an individual development of a platform, however, this is neither a likely path to a high performant computer system, nor appears to be a cost effective solution.

Furthermore, the presence of dynamic and potentially non-controllable features in many COTS present further issues. Examples for such features are the NMI interrupt in Intel based systems or commonly used clock throttling for temperature and/or power management. Similar to the resource sharing, the identification of such features requires great care to be taken. Secondly, if such mechanisms may not be switched off in a guaranteed manner or provide no way to reason about their behaviour and impact it will make the deployment of such an architecture likely impossible in a certified environment.

Besides the HW architecture features discussed above, the software stack running on top of a multicore platform has the potential to mitigate some of the issues occurring on the HW platform. This reaches from the application driven pipelining of image processing segments to concentrate memory accesses at the start and end of the execution and thus avoiding interference on the shared read and write paths. On the operating system level tasks may be pinned to specific cores and thus avoiding the indeterminism of migrating applications.

One fundamental asset in the certification process of new HW/SW platforms would be to obtain a set of guidelines identifying "best practice" in the area. However, such guidelines

would naturally have to be developed by the certification agencies and are usually a product of experience. Consequently the process faces a similar paradigm shift, as the introduction of the first processors in certified systems delivered, with little clear idea on how to approach the problem. The first system mentioned in the introduction is certainly a step in this direction, even in this case though the process cannot be successfully completed, as it means first testing the waters and then refining with future attempts.

4.4 Network-on-Chip – Dependability and Security Aspects

Roman Obermaisser, Christian El Salloum, Theo Ungerer, and Thomas Wild

License © Creative Commons BY 3.0 Unported license
© Roman Obermaisser, Christian El Salloum, Theo Ungerer, and Thomas Wild

A major requirements for NoCs in embedded systems in *predictability*. Techniques for predictability range from static scheduling (e.g., time-triggered) to dynamic scheduling (e.g., priority-based). Also, NoCs provide solutions for monitoring and enforcement of resource budgets (e.g., AETHEReal). Predictable application behavior for a given NoC also requires suitable modeling and timing analysis techniques.

The second challenge for NoCs is *composability*. Composability refers to a framework that supports the integration and reuse of independently developed components in order to increase the level of abstraction in the design process. Prior services of components must not be invalidated by integration, which is facilitated by temporal and spatial isolation based on precise interface specifications. In addition, the goal of composability is to avoid unintended emerging side effects at the system-level. Of particular importance in NoCs is deadlock freedom. Deadlock freedom can be ensured by isolation (i.e., control of dependencies), suitable routing strategies without deadlocks and formal analysis methods for routing cycles. Furthermore, resource guarantees such as bandwidth, jitter and latency must be maintained upon component integration. A key mechanism are specifications with explicit resource and memory requirements.

A third challenge for NoCs is *fault-tolerance* and *robustness* to support the reliable operation in the presence of faults. NoCs need to support the provision of an acceptable level of service on an MPSoC despite the occurrence of transient and permanent hardware faults of resources. For permanent hardware faults, important techniques are active redundancy or migration and reconfiguration of services exploiting spare resources. Transient hardware faults require the recovery in predictable time with state recovery. In mixed-criticality systems, containment of design faults is the primary concern, which is supported by NoCs with strong temporal and spatial isolation. The error detection mechanisms for operational faults and design faults can be based on a priori knowledge, information redundancy, analytical redundancy or replication. Recent advances in fault-tolerance of NoCs focus on proactive fault-tolerance. For example, wear-out specific scheduling takes into account temperature variations or increasing fault-rates.

Security is rapidly gaining significance in the field of embedded systems. In particular in safety-critical systems, security has to be considered as a safety aspect in scenarios where a malicious attack can lead to unspecified system behavior with catastrophic consequences (e.g., sabotage or terroristic attacks). The NoC in a multi-core processor provides the perfect opportunity to implement security mechanisms directly in hardware in order to enforce specific inter-core security policies. Considering the role that a NoC has, namely establishing communication among the individual cores and other entities like on-chip device controllers, a security-enabled NoC should establish the following properties:

- **Authenticity of the sender:** A receiving core on the NoC should be able to reliably determine the core from which a message was sent. It should be not possible for any core to forge the sender address without being detected.
- **Message integrity:** It should be not possible for any core to modify, delete or duplicate the messages sent by any other core. In a real-time system, the integrity of a message does not only depend on the message content, but also on the timing of the message. Therefore the integrity requirement has to be extended, such that it also should be not possible for any core to change the timing of messages that where sent by any other core.
- **Message confidentiality:** Only the intended receivers of a message should be allowed to read the message contents.
- **Availability of guaranteed communication resources:** In a (hard) real-time system, guaranteed communication resources have to be given to the individual cores, in order to assure that end-to-end deadlines are always met. From a security perspective it must be prevented, that the behavior of a malicious core (e.g., due to a compromised program running on a core) can lead to a violation of such guarantees (i.e. Denial-of-Service DoS attacks).
- **Prevention of side-channel attacks:** For some applications, it is required that there are no other possible means of communication than the explicitly defined communication channels. Such other means of communication are called side channels. An example of a side channel can be found in a NoC where the temporal properties of a given communication channel depend (even slightly) on the communication activities on another communication channel. Imagine two malicious nodes located on two explicitly defined distinct communication channels. Since the communication channels are defined as distinct, it should be not possible to leak confidential information from one channel to the other. The problem in a NoC that is not free from temporal interference, is that the two malicious cores can use that interference to illegally exchange information in a Morse-Code like manner. One core can induce a specific temporal interference pattern (by sending a pattern of channel-local messages) which can be observed and interpreted by the core on the other channel.

An example of a NoC satisfying all the above stated requirements is the time-triggered NoC (TTNoC) in the ACROSS architecture. In ACROSS the cores have no direct access to the NoC, but only via the Trusted Interface Sub System (TISS) which acts as a guardian. The TISS stores a statically defined time-triggered message schedule, which holds for each message the sent instance as well as the route and the set of receivers. Thus, the time-triggered schedule holds the entire topology which defines to which receivers a given message will be forwarded. All other cores will never see the message. The statically defined topology ensures message integrity and authenticity as well as confidentiality. Furthermore, the TISS ensures that messages are only sent according to the time-triggered schedule such that there is absolutely no temporal interference among different messages. Thereby the TTNoC establishes availability of the communication resources and prevents hidden side-channel attacks.

Adaptiveness is a challenge for NoCs to support system evolution, context adaptation and resource variation. In long-lived systems, the integration of new components, services and resources is needed to cope with changed application requirements. Technique for adaptiveness include predictable, fault-tolerant and secure configuration of the NoC. A prerequisite are standardized interfaces supporting configuration. Recent techniques for self-optimization in NoCs are a promising approach to autonomously and continuously adapted to the application behavior and resource availability. The extension of these feedback techniques for safety-critical embedded systems is a future research challenge.

4.5 Multicore Ecosystem

Andreas Herkersdorf, Johan Lilius, Massimo Ferraguto, Christian Thiel, Stefan Wallentowitz, and Thomas Wild

License © Creative Commons BY 3.0 Unported license
© Andreas Herkersdorf, Johan Lilius, Massimo Ferraguto, Christian Thiel, Stefan Wallentowitz, and Thomas Wild

Multicore as an ICT Key Technology

Multicore processors are a key technology for coping with the important challenges our society will face in the upcoming decades. Secure and sustainable mobility, comprehensive healthcare, universal power management and the development of a digital society pose great demands on a distributed and powerful information and communication technology (ICT). These demands on embedded and cyber physical systems can only be met with multicore processors. All leading processor vendors – Intel, IBM, ARM, Nvidia, Freescale, Infineon, MIPS, TI – pursue a multicore architecture strategy. Such multicore processors are superior to their single-core ancestors with respect to processing performance and power efficiency, as they can execute different tasks concurrently on less complicated but parallel processor cores. On the other hand, industry and academia are facing entirely new challenges with respect to system complexity. The efficient utilization of parallel processing resources currently relies predominantly on the individual skills of the programmers. In the field of embedded and cyber physical systems, multicore processors must adhere to much stronger demands of real-time, power efficiency, reliability, safety and security when compared to standard desktop machines. Furthermore, multicore-enabled test and debugging tools are often missing along with universal methods for the modeling, design and validation of system issues. Various industrial and academic institutions in Europe have identified the relevance of multicore as a key ICT technology from the very beginning, and have established a competitive knowledge base for multicore technologies. However, finding flexible and scalable solutions for non-functional requirements, performance and power efficiency in increasingly demanding embedded system applications will soon be beyond the capabilities of large-scale enterprises or even networks of companies.

Roles and Benefits of a Multicore Ecosystem

The “Working Group Multicore” within the Bavarian ICT Innovation Cluster BICCNNet proposes the establishment of a research and development network to jointly tackle these challenges [1]. In particular, topics such as parallelization support for non-functional requirements, migration of existing software and the development of sophisticated tools for debugging, testing and validation of multicore systems need to be addressed jointly. By achieving their individual goals, partners in this research network will also contribute to the above-mentioned topics. The results from publicly funded projects can be designed with compatibility in mind by using standard interfaces, and are available to all partners, allowing a growing *multicore ecosystem* to develop. Along with software and hardware components, this ecosystem will also contain models, methods and tools for multicore solutions; to the advantage of all contributing partners.

A multicore ecosystem can have a number of beneficial aspects. It can be an innovation ecosystem, where the idea is to encourage the interaction among of the actors to create new innovations, or its goal can be to create new business. In the first the main goal is the creation of new ideas, while in the latter the goal is to create new economic value. In

addition, the mere information exchange among different players and recognizing, who can bring what asset to the table and looks for filling what gaps in the own portfolio, may bring together new partners and represents a value by itself.

An ecosystem is often recognised post-facto, when one realises that there are strong activities around an issue. In the area of multi-core, the classical example of a business ecosystem is the ARM ecosystem, that has grown around ARM processors. Other examples of ecosystems are e.g. the activities around the AUTOSAR standard (which maybe is not as clearly identified as an ecosystem yet), or the activities around the eclipse tools (for which there is not necessarily big economic gain for the participants). Characteristic for these ecosystems is that there is a central entity around which the actors of the ecosystems place themselves to achieve added value.

In order to get started, the basic set of entities for a multicore ecosystem could center around could be a set of tools or hardware and software intellectual property building blocks (such as, e.g., elements of the AUTOSAR stack) that are either difficult to obtain (portability, licensing), very expensive to buy, and would be too complex to build oneself. Identifying such a set of tools and building blocks, and providing them for use to the community could be an interesting foundation to start ecosystem building activities. Providing this set of tools as open-source is crucial, since as noted by Riehle [3] this provides an avenue not only for users of the software, but also for system integrators and other actors to increase profit. Finally participation in the further development of the tools also becomes crucial for companies, as this will allow them to participate in the decision processes and influence the tool evolution.

As a proposal the OpTiMSoC [2] tools could form a starting point.

Many-core Monday

An ecosystem needs a platform for interaction and for attracting new participants. One such platform is the regular BICCnet AK Multicore meetings in Munich. Another interesting concept is *Mobile Monday*¹. This is an open community platform of mobile industry visionaries, developers and influential individuals fostering brand neutral cooperation and cross-border P2P business opportunities through live networking events to demo products, share ideas and discuss trends from both local and global markets. Mobile Monday started as an informal gathering in Helsinki, with the aim of bringing together persons in the mobile industry. Initially it was just a group of people inviting friends and colleagues to an informal drink in a bar on Mondays. Often there were one or two presentations about something interesting, but the emphasis was on informal discussion and networking. The movement has grown and is organised into chapters that have organised events in over 140 cities worldwide.

Open Innovation

Open Innovation is an idea promoted by Henry Chesbrough [4], where companies use both internal and external ideas to create new products. An open innovation ecosystem consists of a group of actors that share both risk and rewards, creating growth for everybody. Central in this idea is that it is possible to build on top of other ideas. In the multicore area, open innovation could help create larger toolflows if tool vendors would make their tools interoperable, and would build new tools based on these toolflows. Open innovation is promoted by many large companies, and e.g. Nokia has been working successfully with a number of Universities (EPFL Lausanne, Berkley, Aalto), by forming “tablets” small research

¹ <http://www.mobilemonday.net/> – “Mobile Monday”

groups at the University campus. This makes it easy for the industrial and the academic researchers to interact.

References

- 1 A. Herkersdorf et al., *Relevanz eines Multicore-Ökosystems für künftige Embedded Systems*, BICC-net, 2011, <http://www.bicc-net.de/nachrichten/artikel/multicore-oekosystem/>.
- 2 S. Wallentowitz, P. Wagner, M. Tempelmeier, T. Wild, A. Herkersdorf, *Open Tiled Manycore System-on-Chip*, arXiv:1304.5081, <http://arxiv.org/abs/1304.5081>
- 3 D. Riehle, *The economic motivation of open source software: Stakeholder perspectives*, *Computer*, vol. 40, no. 4, pp. 25–32, 2007.
- 4 H. W. Chesbrough, *Open Innovation*, Harvard Business Press, 2006.

4.6 Secure Elements in future embedded multicore systems

Georg Sigl, Sri Paramareswaran, Michael Paulitsch, Stefan M. Petters, Matthias Pruksch, Sergey Tverdyshev, and Stefan Wallentowitz

License © Creative Commons BY 3.0 Unported license
© Georg Sigl, Sri Paramareswaran, Michael Paulitsch, Stefan M. Petters, Matthias Pruksch, Sergey Tverdyshev, and Stefan Wallentowitz

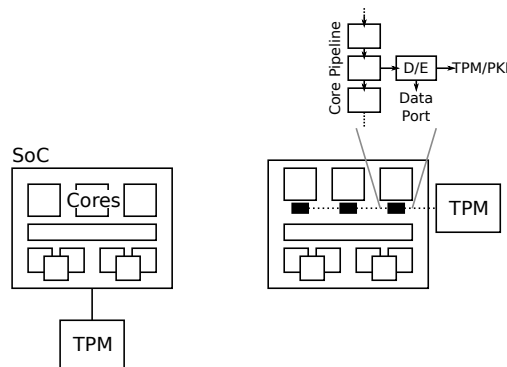
There is currently a trend that a specifically designed hardware attack resistant very well isolated secure element is integrated in systems on chip (SoC). The purpose of this secure element is:

- Integrity check of software that is executed on the system during boot
- Remote Attestation to confirm to a remote party that integrity of system is ok
- Provide identity and authentication for communication with other parties (PKI)
- Access control to resources and configuration registers of the SoC
- Key storage and secure memory

One example where these features are used is the boot process in a system. Normally in a microprocessor system using secure boot technology as specified by the Trusted Computing Group, the boot code calculates a hash value of its own executable, sends this to the secure element where the value is compared to an expected value. Then the secure element would recognize if the boot code has been changed. After that the boot core calculates the hash value of configuration code, e.g. BIOS, and afterwards of the operating system kernel and sends these values to the secure element as well for comparison. After completion of this process the secure element can confirm that the system has been started with trusted software. In a multicore embedded system this start up procedure is executed by one core (core 0) which then starts the boot process of other cores.

This security solution is based on the assumption that no hardware attacks are performed on the system on chip and a software security layer, e.g. a hypervisor, provides sufficient security and separation services for the applications running on the multicore system. For a better separation of cores a new architecture with an encryption and decryption (D/E) unit at each core could be helpful as shown in Figure 1. All data leaving a core would be encrypted and could be put into common memories without the chance to access it from other cores unless the correct key is known.

If we assume hardware attacks on a multicore system we currently have no hardware means in the multicore to detect them. The only solution in a system with a secure element is to move all security critical tasks into the secure element and establish an end-to-end



■ **Figure 1** An encryption and decryption unit at each core can help with better separation of cores.

protected communication channel between the secure element and the remote application requesting the security service. If we have security services, which need a high bandwidth such as car2car communication with hundreds of signatures to be computed within seconds, there may be a need to integrate many secure elements as well. Otherwise there may be a bandwidth problem in an architecture as shown above.


There are both chances and risks in multicore systems concerning security (see presentation of Georg Sigl). One example where we see even both advantages and disadvantages in multicore systems is the chance to implement monitoring services in multicores. One core could be used to monitor the behavior of others in order to detect misbehavior created by malware running. On the other side the monitoring could be used to perform side channel attacks with a much better measurement accuracy compared to an external measurement of the cache behavior, e.g..

A very good countermeasure against many attacks, such as side channel attacks, is randomization. Multicores give plenty of opportunity for randomization, which is exactly the most severe concern of safety-critical system design engineers. Designers and certification authorities insist in deterministic behavior for these systems in order to determine, e.g., worst case execution times and to guarantee certain timings. As a solution to resolve this conflict for secure safety-critical systems, it would be very interesting to investigate implementations, which accept random behavior and still guarantee a timely execution with high probability. The project Proartis² goes into this direction. Synergies between this approach and the needs and solutions developed in the security domain could be a very interesting research direction and may be also a topic for a future Dagstuhl seminar.

² <http://www.proartis-project.eu/>

4.7 Inter-seminar workgroup: Software Certification & Multicore Processing

Michael Paulitsch

License  Creative Commons BY 3.0 Unported license
© Michael Paulitsch

Multicore processing for safety-critical and security-relevant and safe deployment strongly depends on the ability to certify software running on multicore processors in the system context. The workshop “Multicore Enablement for Embedded and Cyber Physical Systems” has been incidentally running in parallel to the workshop “Software Certification: Methods and Tools (Seminar 13051)”. Both groups realized the link between the two topics and organized an open common exchange and discussion session that addressed both topics in some detail. The common session increased the understanding of each other’s workshop topic and made participants realize the complexity of certification involving multicore processors. An exemplary common observation of both seminar participant groups was the ever increasing gap and wishes of simplicity of processing in certified safety-critical environments for deterministic execution of critical software and the increasing modern multicore processor complexity.

Participants

- Michael Deubzer
Timing Architects Embedded
Systems GmbH, DE
- Christian El Salloum
TU Wien, AT
- Rolf Ernst
TU Braunschweig, DE
- Glenn Farrall
Infineon – Bristol, GB
- Christian Ferdinand
AbsInt – Saarbrücken, DE
- Massimo Ferraguto
Space Syst. Finland Ltd –
Espoo, FI
- Steffen Görzig
Daimler AG – Böblingen, DE
- René Graf
Siemens AG – Nürnberg, DE
- David Gregg
Trinity College Dublin, IE
- Geoff Hamilton
Dublin City University, IE
- Andreas Herkersdorf
TU München, DE
- Johan Lilius
Abo Akademi University, FI
- Enno Lübbers
Intel GmbH – Feldkirchen, DE
- Roman Obermaisser
Univ. Siegen – Feldkirchen, DE
- Sri Parameswaran
UNSW – Sydney, AU
- Michael Paulitsch
EADS – München, DE
- Stefan M. Petters
ISEP-IPP – Porto, PT
- Matthias Pruksch
sepp.med – Röttenbach, DE
- Georg Sigl
TU München, DE
- Claus Stellwag
Elektrobit Automotive –
Erlangen, DE
- Jürgen Teich
Univ. Erlangen-Nürnberg, DE
- Christian Thiel
BICCnet – München, DE
- Lothar Thiele
ETH Zürich, CH
- Sergey Tverdyshev
Sysgo AG – Mainz, DE
- Theo Ungerer
Universität Augsburg, DE
- Stefan Wallentowitz
TU München, DE
- Alexander Weiss
Accemic GmbH & Co. KG –
Kiefersfelden, DE
- Thomas Wild
TU München, DE
- Reinhard Wilhelm
Universität des Saarlandes, DE

