Editorial

Jürgen Becker*

Multicore technology in the mobility domains

https://doi.org/10.1515/itit-2017-0026

1 Introduction

Although multicore technology is well established in many best effort driven systems, in the context of safety-critical or mixed-critical embedded systems, multicore technology still persists in early stages of development. Even though a high demand for computational processing power exists, the transition from sequential to parallel execution introduces various challenges including spatial and temporal interferences, synchronization issues or the deployment of software on multiple cores.

In the project ARAMiS, leading industrial companies and research institutions joined their forces and developed cross domain solutions, which enable the use of multicore devices in safety-critical and mixed-critical applications. The developed approaches implicate deployment and system architecture analysis, safety aspects like monitoring, resource sharing, segregation and virtualization as well as tooling support.

2 Background

Embedded systems play an ever increasing role in almost any field of daily life, including the mobility domain taking massive benefits from using software in their products. In fact, electronic components and systems are the key factors for the integration of novel functionality in any technical system. Nowadays, more than 90% of all innovations in modern vehicles are possible through the use of electronics and software. In consequence, the vast majority of modern technology contains to some extend electronics and very often software running on an embedded processor.

It is a general observation, that every new generation of a product comes with an increasing amount of software, being executed on processors with additional performance. In fact it is the only way to master the increased complexity, meet environmental challenges, enhance competitiveness, and to improve cost efficiency. In other words, the performance of next generation's embedded systems will be a key factor for the success of novel applications such as highly automated systems.

For these reasons singlecore devices as presently used come to their performance limits. Further raising the clock frequency is not an option, because of technical reasons i.e. power dissipation density. Consequently, an alternative approach to singlecore devices is needed. Otherwise, novel functionality might not be realizable, and the limitation of singlecore might turn out to be a major impediment to innovation.

Multicore technology, being state of the art in standard ICT for a couple of years now, seems to be the most promising way and probably is the key enabler for future systems. However, the emergence of truly parallel execution of cooperating applications requires new design and engineering approaches. Analytical support of the required software alteration or generation by engineering processes and software tools is desirable and increasingly mandatory.

In general, embedded systems have to fulfill a huge amount of functional and non-functional requirements, which – especially in the case of safety critical systems – go far beyond the standards of best effort driven systems. Here the most important characteristics are: real-time capabilities, performance, reliability and availability, functional safety, robustness against malicious attacks, compatibility to existing concepts (legacy code), and user friendliness and apps (third party code).

Obviously developing embedded systems, which fulfill all requirements that result from the aforementioned characteristics is a challenging task, not even mentioning the architectural complexity within products.

The main multicore challenges in the context of safety-critical or mixed-critical embedded systems can be summarized as follows:

- **Segregation in space and time:** Concepts are necessary, that allow sharing resources with clear limits in memory space and time to reduce interferences. This affects all kind of resources, including memory, coprocessors or peripherals.

***Corresponding author: Jürgen Becker,** Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, e-mail: becker@kit.edu

- **Efficient application software distribution:** The distribution of software needs to be efficient and minimize total execution costs in order to benefit from the potential performance of the multicore.
- **Legacy code on multicore:** Concepts and mechanisms are needed, that allow deploying legacy singlecore code on multicore platforms. This also includes parallelization of legacy software.
- **Efficient platform software distribution:** Concepts for the distribution of platform infrastructure software, including operating systems and middleware are necessary.
- **Analysis of multicore platforms and software:** Analysis methodologies and tools are needed that allow detecting potential multicore-specific software errors (e.g., race conditions, WCET).
- **Configuration complexity:** Multicores have an enormous configuration space that needs to be managed to find valid and robust configurations for safety critical systems.
- **Certifiability/quantifiability:** Achieving certifiability is a major challenge, due to the multicores complexity and interferences.

3 Topics of this special issue

Based on the achieved results in the ARAMiS project, a comprehensive overview of challenges and contributions is given in the following articles:

System-Level considerations: An important focus of work conducted in ARAMiS was targeted towards the overall multicore system. The most important reason for that decision is that safety, methodology and tooling are not limited to hardware or software only. Moreover system efficiency requires optimization across the hardware/software boundary. The article "Multicore System Architecture Aspects and Deployment" briefly discusses the most important activities, which have been performed on system architecture level, including improvements of modelling existing architectures and deployment strategies. Furthermore, several design techniques concerning reliability, independence, segregation and determinism for the deployment of software components to multicore processors are exemplarily described as "Core Usage Patterns".

Monitoring: In a complex Multiprocessor Systems-On-Chip (MPSoC) it is one of the most important challenges to know the system behavior. As the complexity of today's MPSoCs rises to another level, simple predictions and a deterministic behavior are not valid anymore. Therefore a collection of information within the System-on-Chip is a necessary basic for all monitoring approaches. The article "Online Monitoring for safety-critical Multicore Systems" presents and evaluates online monitoring approaches for safety-critical multicore systems.

Virtualization and hardware extensions: A promising approach for using multicores is the implementation of embedded virtualization. As virtualization still is new for embedded multicore devices and safety critical applications, several challenges had to be addressed. Although ARAMiS was neither a hardware project nor had the goal to develop its own multicore chip, some investigation on architectural level took place with the goal to provide suggestions to semiconductor industry. Proof of concepts were implemented on FPGAs as the extension of coprocessors and peripherals with multicore specific interfaces. These have been also extended towards hardware virtualization support to benefit from the usage of embedded virtualization technology. A multicore specific HSM has been implemented in ARAMiS to provide a clear separation among users and is also presented in the article "Hardware/Software Trade-offs for Shared Resources Virtualization in Mixed-Criticality Automotive Multicore Systems".

Tooling: The solution of the aforementioned challenges not only affects the system architecture but also the way it is developed, thus resulting in challenges for tools and methods. The challenges are twofold: First some new multicore specific tools or tool enhancements are needed, which include the detection of race-conditions, parallelization support, or determination of software deployments. Second tooling landscapes tend to be fragmented, providing a huge amount of very specific solutions without any interactions. Thus it was one of the goals to develop the basis for a coherent multicore tooling structure, enabling the interconnection of various tools for different aspects. Both aspects are summarized in the article "Coherent Multicore Methodology and Tooling".

In addition to the contributions from the ARAMiS context, also related tooling aspects for embedded multicore technology in mobility domains are considered in the article "APP4MC: Application Platform Project for Multi- and Many-core Systems". APP4MC is an open source Eclipse platform that provides AUTOSAR compliant common data models namely AMALTHEA, basic parallelization features, visualizations, and the possibility to add any existing tooling. The platform enables the creation and management of complex tool chains including performance simulation and validation.