Developing Predictable Vehicular Distributed Embedded Systems on Multi-core*

Saad Mubeen[1], Thomas Nolte[1], and Kurt-Lennart Lundbäck[2]

[1] Mälardalen University Sweden, firstname.lastname@mdh.se
[2] Arcticus Systems AB Sweden, kurt.lundback@arcticus-systems.com

Abstract. In this paper we address the challenges related to supporting model- and component-based development of predictable software for vehicular distributed embedded systems, utilizing multi-core platforms. We present a research plan for the development of new methods and techniques to deal with these challenges. The techniques will support various aspects such as modeling of the software architecture; supporting multiple criticality levels; verifying predictability of the system using end-to-end timing analysis; code generation; and providing a predictable run-time support on multi-core platforms by virtualizing a certified single-core real-time operating system. As a proof of concept, we will implement the newly developed techniques in a commercial tool chain (Rubus-ICE). The efficacy of the newly developed techniques and the extended tool chain will be demonstrated on the industrial case studies.

1 Introduction

A large share of customer value in modern vehicles comes from computer-controlled functionality that is realized by distributed embedded systems. With the recent advancements in the vehicular domain, the size and complexity of software in such systems has drastically increased [4]. The traditional software development techniques for these systems are no longer effective with respect to handling the complexity; supporting reuse; reducing development costs, time to test and time to market. These challenges can be addressed by developing these systems using the principles of model- and component-based software engineering [7, 9, 11]. The developers of the these systems have to not only deal with their complexity but also guarantee their predictable timing behavior. This means, the timing behavior of these systems has to be verified, for instance, by using *a priori* schedulability analysis techniques such as end-to-end timing analysis [10, 12]. In addition, the execution platform should also provide predictable run-time support for these systems. There are several industrial tool chains, e.g. Rubus-ICE [2] and AUTOSAR [1] that support model-based development and predictable execution of the single-core systems. However, a lot of new challenges have emerged with the introduction of multi-core platforms for the execution of these systems [13]. A seamless tool chain for the development of vehicle software with *multiple criticality levels* and its predictable execution on partitioned single-core and multi-core platforms is missing in the state of the practice.

* The work in this paper is supported by the Swedish Foundation for Strategic Research within the project PRESS.

Research Goals and Challenges. We aim to develop techniques for model- and component-based software development of systems utilizing multi-core platforms. The techniques will support various development steps, i.e., from modeling of the software architecture to its synthesis and execution. The software in these systems can have multiple criticality levels. For instance, a high-critical function may co-exist with other low-critical functions on the same ECU. Multiple criticality levels in the vehicle software will be supported by means of virtual partitions in the core(s) of single-core as well as multi-core platforms. The main focus of this paper is on supporting predictable execution of these systems on such platforms. In this context, an end-to-end timing analysis framework will be developed to verify timing behavior of these systems. In order to provide a predictable run-time support for these systems on multi-core platforms, we aim to develop a virtualization technique that supports the reuse of a certified single-core Real-Time Operating System (RTOS) by means of a multi-core hypervisor.

Our goal is to implement the newly developed techniques in a tool chain that can be easily adopted by the vehicle industry. We plan to provide a proof-of-concept demonstrator by implementing these techniques in existing commercial tools that are actually used by the industry. The modeling and synthesis techniques will be implemented in the existing industrial model, the Rubus Component Model (RCM) and its tool suite Rubus-ICE [2]. The end-to-end timing analysis will be implemented as a plug-in for Rubus-ICE. Rubus RTOS, already certified in ISO 26262, currently supports single-core platforms. The newly developed multi-core hypervisor will reuse a separate instance of the Rubus RTOS per core. The efficacy of the extended tool suite will be demonstrated on the industrial use cases. The main research challenge can be formulated as follows. *"How to support model- and component-based development and predictable execution of software for vehicular distributed embedded systems on multi-core platforms?"*

2 Research plan and proposed contributions

We propose six phases to accomplish the research goals outlined in the paper.

1. A technique will be developed to support modeling of component-based vehicular distributed embedded systems on multi-core platforms.
2. The technique developed in the first phase will be extended to support modeling and development of systems that have more than one criticality level. This means, some parts of the software architecture (e.g., subsystems or chains of components) in these systems may have more stringent timing requirements compared to other parts. The motivation behind the need for multiple criticalities comes from the requirements on the certification of the system. The modeling of the software architecture for these systems will be implemented by means of virtual partitions in the core(s) of single-core and multi-core platforms respectively. The idea is to allocate the parts of the software architecture with different criticality levels to separate virtual partitions or cores. In this context, policies will be developed to handle dependencies and interferences among partitions.
3. In order to provide pre-runtime guarantees on the timing behavior of the system, the existing end-to-end timing analysis will be extended. Here, the focus

will be on analyzing the end-to-end delays (age and reaction delays) [10] on distributed cause-and-effect chains. The analysis framework will also consider partitioned systems and multi-core platforms.
4. In order to provide a predictable run-time support for these systems on multi-core platforms, we will develop a virtualization technique that supports the reuse of certified single-core RTOS. This objective will be realized by developing a multi-core hypervisor that will instantiate the certified single-core RTOS for each core. One of the benefits for this approach is that there is no need to recertify the RTOS if the multi-core hypervisor can be certified. Moreover, virtual partitions will be supported.
5. We will provide the proof of concept by assembling a complete tool chain that will support modeling, timing analysis, synthesis and predictable execution of the systems on multi-core platforms. The modeling and synthesis techniques will be implemented in RCM and Rubus-ICE. The end-to-end timing analysis will be implemented as a plug-in for Rubus-ICE. The multi-core hypervisor will reuse a separate instance of the certified Rubus RTOS per core.
6. In this phase the validation of the newly developed techniques will be carried out. The tool chain, implementing the new techniques, will be validated on several industrial case studies provided by the industrial partners.

3 Related work

The research community has produced a large body of research on schedulability analysis of mixed-criticality systems since the seminal paper by Vestal [14]. Burns and Davis [5] provide a survey on mixed criticality systems for both single-core and multi-core platforms. However, none of the existing works has addressed end-to-end path delay analysis for distributed systems with multiple criticality levels on multi-core platforms. We will address this analysis in our work. The focus of the survey and related works is on scheduling and schedulability analysis. These works can be regarded as complementary to our work because schedulability analysis is one of our sub-goals in providing a model- and component-based development and predictable execution support for mixed criticality systems on partitioned single-core as well as multi-core platforms.

There are several ongoing research projects that explore prospects of multi-core platforms for embedded systems. EMC2, a large European project[3], aims to utilize multi-core technology for the execution of mixed-criticality systems. In this context, the focus of the project is to support adaptive systems in dynamic and changeable real-time environments by means of a service-oriented architecture approach. On the other hand, our focus is on the development of a seamless tool chain for model-based development and predictable execution of such systems. Moreover, we focus on a component-based architecture. Unlike EMC2, we aim to develop a virtualization technique by reusing the certified single-core RTOS to support predictable multi-core run-time support.

The MCC project[4] focuses on development of mixed-criticality systems on many-core platforms [3]. Whereas the focus of our work is on both partitioned

[3] http://www.artemis-emc2.eu/
[4] https://www.cs.york.ac.uk/research/research-groups/rts/mcc/

single-core and multi-core platforms. Also, MCC does not provide the model- and component-based development support. SMARTCore [6] and other related works [8] exploit model-driven engineering for the development of multi-core systems. They focus on run-time monitoring, deployment optimization and back propagation of extra-functional properties from run-time to the system models. Whereas, our aim is to develop a seamless tool chain that supports various activities including modeling, timing analysis, synthesis and predictable execution of such systems. Also, these works do not address multiple criticality systems.

Similarly, there are other projects that address various aspects of mixed criticality and multi-core systems, e.g., RECOMP, CERTAINTY, PROXIMA, CONTREX and DREAMS. To the best of our knowledge, none of these projects seemingly address our sub-goals that include focus on the vehicular domain; model- and component-based software development; predicting the timing behavior with respect to the end-to-end path delays; and virtualization of certified single-core RTOS to support predictable multi-core run-time support.

References

1. AUTOSAR Techincal Overview, Release 4.1, Rev.2, Ver.1.1.0., http://autosar.org
2. Rubus models, methods and tools, http://www.arcticus-systems.com
3. Bate, I., Burns, A., Davis, R.: A bailout protocol for mixed criticality systems. In: 27th Euromicro Conference on Real-Time Systems (2015)
4. Broy, M., Kruger, I., Pretschner, A., Salzmann, C.: Engineering automotive software. Proceedings of the IEEE 95(2), 356 –373 (Feb 2007)
5. Burns, A., Davis, R.: Mixed criticality systems - a review. Tech. rep., Dept. of Computer Science, University of York (2015)
6. Ciccozzi, F., Corcoran, D., Seceleanu, T., Scholle, D.: Smartcore: Boosting model-driven engineering of embedded systems for multicore. In: 12th International Conference on Information Technology - New Generations. pp. 89–94 (April 2015)
7. Crnkovic, I., Larsson, M.: Building Reliable Component-Based Software Systems. Artech House, Inc., Norwood, MA, USA (2002)
8. Feljan, J., Ciccozzi, F., Carlson, J., Crnkovic, I.: Enhancing model-based architecture optimization with monitored system runs. In: 41st Euromicro Conference on Software Engineering and Advanced Applications. pp. 216–223 (Aug 2015)
9. Henzinger, T.A., Sifakis, J.: The Embedded Systems Design Challenge. In: 14th International Symposium on Formal Methods (2006)
10. Mubeen, S., Mäki-Turja, J., Sjödin, M.: Support for end-to-end response-time and delay analysis in the industrial tool suite: Issues, experiences and a case study. Computer Science and Information Systems 10(1) (2013)
11. Mubeen, S., Mäki-Turja, J., Sjödin, M.: Communications-Oriented Development of Component- Based Vehicular Distributed Real-Time Embedded Systems. Journal of Systems Architecture 60(2), 207–220 (2014)
12. Tindell, K., Clark, J.: Holistic schedulability analysis for distributed hard real-time systems. Microprocess. Microprogram. 40, 117–134 (April 1994)
13. Vector Informatic GmbH: Autosar goes multi-core the safe way. In: Techncal Article (June 2014), https://vector.com
14. Vestal, S.: Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In: 28th IEEE International Symposium on Real-Time Systems. pp. 239–243 (Dec 2007)