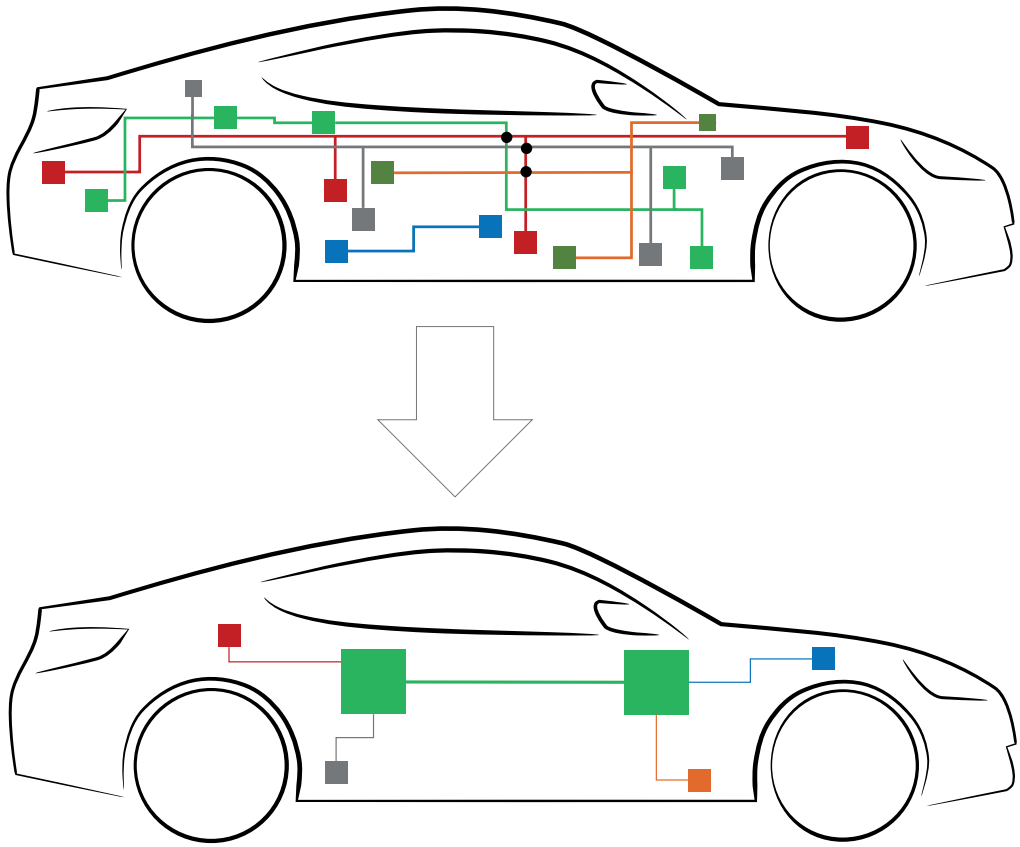
Mälardalen University Doctoral Dissertation 246

# Consolidating Automotive Real-Time Applications on Many-Core Platforms

**Matthias Becker**



MÄLARDALEN UNIVERSITY
SWEDEN

Mälardalen University Press Dissertations
No. 246

# CONSOLIDATING AUTOMOTIVE REAL-TIME APPLICATIONS ON MANY-CORE PLATFORMS

**Matthias Becker**

**2017**


School of Innovation, Design and Engineering

ISBN 978-91-7485-359-9
ISSN 1651-4238
Printed by E-Print AB, Stockholm, Sweden

Mälardalen University Press Dissertations
No. 246

# CONSOLIDATING AUTOMOTIVE REAL-TIME APPLICATIONS ON MANY-CORE PLATFORMS

Matthias Becker

Akademisk avhandling

som för avläggande av teknologie doktorsexamen i datavetenskap vid
Akademin för innovation, design och teknik kommer att offentligen försvaras
tisdagen den 19 december 2017, 09.00 i Kappa, Mälardalens högskola, Västerås.

Fakultetsopponent: Professor Marco Di Natale, Scuola Superiore Sant'Anna


Akademin för innovation, design och teknik

Abstract

Automotive systems have transitioned from basic transportation utilities to sophisticated systems. The rapid increase in functionality comes along with a steep increase in software complexity. This manifests itself in a surge of the number of functionalities as well as the complexity of existing functions. To cope with this transition, current trends shift away from today's distributed architectures towards integrated architectures, where previously distributed functionality is consolidated on fewer, more powerful, computers. This can ease the integration process, reduce the hardware complexity, and ultimately save costs.

One promising hardware platform for these powerful embedded computers is the many-core processor. A many-core processor hosts a vast number of compute cores, that are partitioned on tiles which are connected by a Network-on-Chip. These natural partitions can provide exclusive execution spaces for different applications, since most resources are not shared among them. Hence, natural building blocks towards temporally and spatially separated execution spaces exist as a result of the hardware architecture.

Additionally to the traditional task local deadlines, automotive applications are often subject to timing constraints on the data propagation through a chain of semantically related tasks. Such requirements pose challenges to the system designer as they are only able to verify them after the system synthesis (i.e. very late in the design process).

In this thesis, we present methods that transform complex timing constraints on the data propagation delay to precedence constraints between individual jobs. An execution framework for the cluster of the many-core is proposed that allows access to cluster external memory while it avoids contention on shared resources by design. A partitioning and configuration of the Network-on-Chip provides isolation between the different applications and reduces the access time from the clusters to external memory. Moreover, methods that facilitate the verification of data propagation delays in each development step are provided.

ISBN 978-91-7485-359-9
ISSN 1651-4238

# Sammanfattning

Fordonssystem har gått från att vara enkla transportmedel till att vara sofistikerade system. Genom den snabba ökningen av funktionalitet uppstår även en kraftig ökning av komplexitet. Detta manifesterar sig i en ökning av antalet funktioner och ökad komplexitet hos redan befintliga funktioner. För att hantera denna övergång till sofistikerade system ser vi ett skifte i nuvarande trender, bort från dagens distribuerade arkitekturer för att istället inrikta sig mot integrerade arkitekturer, där tidigare distribuerad funktionalitet konsolideras på färre och mer kraftfulla datorer. Detta skifte kan i sin tur underlätta integrationsprocessen, minska hårdvarukomplexiteten och slutligen minska kostnader.

En lovande hårdvaruplattform för dessa elektriska styrenheter är processorer med ett stort antal kärnor, även kallad many-core processorer. En many-core processor rymmer ett stort antal processorer som är partitionerade i rutnät och som är sammankopplade med varandra genom ett nätverk, ett Network-on-Chip. De naturliga partitioner som detta skapar möjliggör separata exekveringsutrymmen för olika applikationer, eftersom de flesta resurser inte delas mellan dem. Därför finns naturliga byggstenar för temporalt och rumsligt separerade exekveringsutrymmen som ett resultat av hårdvaruarkitekturen.

Förutom att applikationer som traditionellt har individuella tidskrav så finns för fordonsapplikationer ofta tidskrav för datautbredningen genom en kedja av semantiskt relaterade programdelar. Sådana krav skapar utmaningar för systemdesignern eftersom de endast kan verifiera dem efter systemsyntesen (dvs mycket sent i designprocessen).

I denna avhandling presenterar vi metoder som förvandlar komplexa tidskrav på fördröjningen hos dataöverföringen till ordningsrelationer mellan enskilda jobb. Vi har föreslagit ett exekveringsramverk för many-core-kluster som möjliggör åtkomst till externt minne samtidigt som det undviker att skapa flaskhalsar kring delade resurser. Partitionering och konfiguration av nätverk

ger isolering mellan de olika applikationerna och reducerar åtkomsttiden från klustren till externt minne. Dessutom tillhandahålls metoder som underlättar verifieringen av fördröjning av datautbredning i varje utvecklingssteg.

# Abstract

Automotive systems have transitioned from basic transportation utilities to sophisticated systems. The rapid increase in functionality comes along with a steep increase in software complexity. This manifests itself in a surge of the number of functionalities as well as the complexity of existing functions. To cope with this transition, current trends shift away from today's distributed architectures towards integrated architectures, where previously distributed functionality is consolidated on fewer, more powerful, computers. This can ease the integration process, reduce the hardware complexity, and ultimately save costs.

One promising hardware platform for these powerful embedded computers is the many-core processor. A many-core processor hosts a vast number of compute cores, that are partitioned on tiles which are connected by a Network-on-Chip. These natural partitions can provide exclusive execution spaces for different applications, since most resources are not shared among them. Hence, natural building blocks towards temporally and spatially separated execution spaces exist as a result of the hardware architecture.

Additionally to the traditional task-local deadlines, automotive applications are often subject to timing constraints on the data propagation through a chain of semantically related tasks. Such requirements pose challenges to the system designer as they are only able to verify them after the system synthesis (i.e. very late in the design process).

In this thesis, we present methods that transform complex timing constraints on the data propagation delay to precedence constraints between individual jobs. An execution framework for the cluster of the many-core is proposed that allows access to cluster external memory while it avoids contention on shared resources by design. A partitioning and configuration of the Network-on-Chip provides isolation between the different applications and reduces the access time from the clusters to external memory. Moreover, methods that facilitate the verification of data propagation delays in each development step are provided.

To my family.

*Ever tried. Ever failed. No matter.*
*Try again. Fail again. Fail better.*

Samuel Beckett

# Acknowledgements

Many people have supported me on the path that lead to this dissertation. First and foremost I would like to express my deepest gratitude to my supervisors, Professor Thomas Nolte, Associate Professor Moris Behnam, Dr. Saad Mubeen, and Adjunct Professor Kristian Sandström for their constant support, encouragement and expert guidance.

I am very grateful to Dakshina Dasari for her interest in my work, the enthusiasm, the many discussions, and the fruitful collaboration and friendship we developed during this time. I am also deeply indebted to Vincent Nélis for our collaboration and for providing me the possibility to visit the CISTER research centre in Porto, Portugal two times during this journey. These visits sparked many ideas that realize part of this thesis. In addition, I thank Borislav Nicolić and Benny Åkesson for all the discussions and their excellent feedback during our joint works.

A special thanks goes to Meng Liu for the close collaboration and the many discussions we had on the topic of Network-on-Chips.

I further thank all my co-authors with whom I had the pleasure to work with during this time: Adriaan Schmidt, Martin Orehek, Thomas Nolte, Moris Behnam, Kristian Sandström, Mohammad Ashjaei, Rafia Inam, Nima Khalilzad, Reinder J. Bril, Dakshina Dasari, Vincent Nélis, Luís Miguel Pinho, Saad Mubeen, Lingjian Gan, Xiaosha Zhao, Mikael Sjödin, Meng Liu, Benny Åkesson, Borislav Nicolić, Nandinbaatar Tsog, Fredrik Bruhn, and Marcus Larsson.

In addition, I thank all current and past members of the CORE Research Group: Thomas Nolte, Moris Behnam, Saad Mubeen, Kristian Sandström, Meng Liu, Alessandro Papadopoulos, Mohammad Ashjaei, Sara Afshar, Hamid Reza Faragardi, Daniel Hallmans, Reinder J. Bril, Luis Almeida, Nima Khalilzad, Rafia Inam, Mikael Åsberg, and Hang Yin.

I thank Kurt-Lennart Lundbäck, CEO of Arcticus Systems, for inviting me

to the company to discuss my research, to the valuable discussions with Mattias Gålnander and John Lundbäck and their important feedback, and for the possibility to validate parts of our research using their tool-suite.

I thank all lecturers and professor who provided many courses that I could attend throughout my time at MDH: Reinder J. Bril, Gordana Dodig-Crncovic, Hans Hansson, Hongyu Pei-Breivolt, Kristian Sandström, Moris Behnam, Margaret Obondo, Séverine Sentilles, Federico Ciccozzi, Antonio Cicchetti, Erik Dahlquist, Jan Gustafsson, Karin Molander Danielsson, Alessandro Papadopoulos, Mohammad Ashjaei, Helena Darnell-Berggren.

I also thank the administrative staff at MDH who were always there and made many things easier, especially Carola Rytterson and Susanne Fronnå.

Life at the university would not have been so pleasant without many of my colleagues and friends. I thank Predrag Filipovikj, Saad Mubeen, and Alessandro Papadopoulos for all the espresso trips to the software engineering department that had lead to many technical and non-technical discussions over the years. Thanks is also appropriate to Radu Dobrin, Antonio Cicchetti, and Federico Ciccozzi for letting us use the espresso machine constantly, it saved many late nights at the university.

Many others have also contributed to life at the university, to the fika breaks, and to the conference trips: Abhilash Thekkilakattil, Adnan Čaušević, Aida Čaušević, Alessio Bucaioni, Andreas Gustavsson, Ashalatha Kunnappilly, Ayhan Mehmed, Cristina Seceleanu, Dag Nyström, Daniel Hallmans, Eduard Paul Enoiu, Elaine Åstrand, Elena Lisova, Filip Markovic, Francisco Pozo, Gabriel Campaneau, Guillermo Rodriguez-Navas, Hamid Reza Faragardi, Hossein Fotouhi, Irfan Sljivo, Jakob Danielsson, LanAnh Trinh, Leo Hatvani, Marina Gutiérrez, Maryam Vahabi, Mehrdad Saadatmand, Meng Liu, Mikael Ekström, Mirgita Frasheri, Mohammad Ashjaei, Nandinbaatar Tsog, Nesredin Mahmud, Nils Müllner, Omar Jaradat, Patrick Denzler, Pablo Gutiérrez-Peón, Per Hellström, Raluca Marinescu, Rong Gu, Sahar Tahvili, Sara Afshar, Sara Abbaspour, Sara Abaspour(x), Séverine Sentilles, Simin Cai, Svetlana Girs, Tiberius Seceleanu, Tobias Holstein, Wasif Afzal.

During my visits to Porto I had the pleasure to meet many people that made life there memorable. I thank David Pereira for all the evening activities and weekend trips we had during this time, as well as Vincent Nélis, Patrick Meumeu Yomsi, Geoffrey Nellisen, Mitra Nasri, Hazem Ali, Borislav Nicolić, Shashank Gaur, João Loureiro, Cláudio Maia, Artem Burmyakov, and Konstantinos Bletsas for all the fun we had.

I would also like to thank Thomas Nolte, Moris Behnam, Saad Mubeen, Kristian Sandström, Jan Carlson, Dakshina Dasari, and Alessandro Pa-

padopoulos for their valuable feedback and comments on this thesis.

I thank Professor Marco Di Natale for accepting to by my faculty examiner, and to the committee members Associate Professor Enrico Bini, Associate Professor Martina Maggio, and Professor Petru Eles who kindly accepted to review and grade my thesis.

I am thankful to Min Kyung Song for her support and care, and for always being there for me. You make life beautiful!

In closing, I would like to express my sincere appreciation to my family who provided me with their unstinting support and trust throughout my life.

This work has been supported by the Swedish Knowledge Foundation (KKS) via the project PREMISE, and Mälardalen University.

Matthias Becker
Västerås, November 14, 2017

# List of publications

## Papers Included in the Doctoral Thesis[1]

**Paper A** *Investigation on AUTOSAR-Compliant Solutions for Many-Core Architectures*, Matthias Becker, Dakshina Dasari, Vincent Nélis, Moris Behnam, Luís Miguel Pinho, Thomas Nolte. In the Proceedings of the 18th Euromicro Conference on Digital System Design (DSD), 2015, August.

**Paper B** *Synthesizing Job-Level Dependencies for Automotive Multi-Rate Effect Chains*, Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, Thomas Nolte. In the Proceedings of the 22nd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2016, August.

**Paper C** *A Generic Framework Facilitating Early Analysis of Data Propagation Delays in Multi-Rate Systems*, Matthias Becker, Saad Mubeen, Dakshina Dasari, Moris Behnam, Thomas Nolte. In the Proceedings of the 23rd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2017, August.
**Invited Paper.**

**Paper D** *End-to-End Timing Analysis of Cause-Effect Chains in Automotive Embedded Systems*, Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, Thomas Nolte. In Journal of Systems Architecture (JSA), Vol. 80 (Supplement C), Elsevier, 2017, October.

[1]The included articles have been reformatted to comply with the doctoral thesis layout and minor typos have been corrected and marked accordingly.

**Paper E** *Contention-Free Execution of Automotive Applications on a Clustered Many-Core Platform*, Matthias Becker, Dakshina Dasari, Borislav Nicolic, Benny Åkesson, Vincent Nélis, Thomas Nolte. In the Proceedings of the 28th Euromicro Conference on Real-Time Systems (ECRTS), 2016, July.

**Paper F** *Partitioning and Analysis of the Network-on-Chip on a COTS Many-Core Platform*, Matthias Becker, Borislav Nicolic, Dakshina Dasari, Benny Åkesson, Vincent Nélis, Moris Behnam, Thomas Nolte. In the Proceedings of the 23rd IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2017, April.

**Paper G** *Scheduling Multi-Rate Real-Time Applications on Clustered Many-Core Architectures with Memory Constraints*, Matthias Becker, Saad Mubeen, Dakshina Dasari, Moris Behnam, Thomas Nolte. In the Proceedings of the 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), 2018, January.

# Additional Peer-Reviewed Publications, not Included in the Doctoral Thesis

1. *Using Non-Preemptive Regions and Path Modification to Improve Schedulability of Real-Time Traffic over Priority-Based NoCs*, Meng Liu, Matthias Becker, Moris Behnam, Thomas Nolte. Real-Time Systems Journal, Springer, Vol 53, nr 6, 2017, November.

2. *Extending Automotive Legacy Systems with Existing End-to-End Timing Constraints*, Matthias Becker, Saad Mubeen, Moris Behnam, Thomas Nolte. In the Proceedings of the 14th International Conference on Information Technology : New Generations (ITNG), 2017, April.

3. *Buffer-Aware Analysis for Worst-Case Traversal Time of Real-Time Traffic over RRA-based NoCs*, Meng Liu, Matthias Becker, Moris Behnam, Thomas Nolte. In the Proceedings of the 27th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 2017, March.

4. *Consolidating Automotive Applications on Clustered Many-Core Platforms*, Matthias Becker. In the Proceedings of the SIGDA Student Research Forum at ASP-DAC, 2017, January.
   **Best Poster Presentation Award.**

5. *A Tighter Recursive Calculus to Compute the Worst-Case Traversal Time of Real-Time Traffic over NoCs*, Meng Liu, Matthias Becker, Moris Behnam, Thomas Nolte. In the Proceedings of the 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), 2017, January.

6. *Using Segmentation to Improve Schedulability of RRA-based NoCs with Mixed Traffic*, Meng Liu, Matthias Becker, Moris Behnam, Thomas Nolte. In the Proceedings of the 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), 2017, January.

7. *Real-Time Capabilities of HSA Compliant COTS Platforms*, Nandinbaatar Tsog, Matthias Becker, Marcus Larsson, Fredrik Bruhn, Moris Behnam, Mikael Sjödin. In the Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Work-in-Progress (WiP) Session, 2016, December.

8. *Analyzing End-to-End Delays in Automotive Systems at Various Levels of Timing Information*, Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, Thomas Nolte. In the Proceedings of the 4th IEEE International Workshop on Real-Time Computing and Distributed systems in Emerging Applications (REACTION), 2016, December. **Invited for an extension to the Journal of Systems Architecture.**

9. *Timing Analysis and Synthesis of Mixed Multi-Rate Effect Chains in MECHAniSer*, Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, Thomas Nolte. In the Proceedings of the Open Demo Session of Real-Time Systems located at the IEEE Real Time Systems Symposium (RTSS@Work), 2016, December.

10. *Tighter Time Analysis for Real-Time Traffic in On-Chip Networks with Shared Priorities*, Meng Liu, Matthias Becker, Moris Behnam, Thomas Nolte. In the Proceedings of the 10th IEEE/ACM International Symposium on Networks-on-Chip (NOCS), August, 2016.

11. *Scheduling Real-Time Packets with Non-Preemptive Regions on Priority-based NoCs*, Meng Liu, Matthias Becker, Moris Behnam, Thomas Nolte. In the Proceedings of the 22th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2016, August.
**Best Student Paper Award, invited for an extension to the Real-Time Systems Journal.**

12. *MECHAniSer - A Timing Analysis and Synthesis Tool for Multi-Rate Effect Chains with Job-Level Dependencies*, Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, Thomas Nolte. In the Proceedings of the 7th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS), 2016, July.

13. *Using Segmentation to Improve Schedulability of Real-Time Packets on NoCs with Mixed Traffic*, Meng Liu, Matthias Becker, Moris Behnam, Thomas Nolte. ACM SIGBED Review: Special Issue on the 14th International Workshop on Real-Time Networks (RTN), Vol 13, nr 4, 2016, September.

14. *A Dependency-Graph Based Priority Assignment Algorithm for Real-Time Traffic over NoCs with Shared Virtual-Channels*, Meng Liu, Matthias Becker, Moris Behnam, Thomas Nolte. In the Proceedings of the 12th IEEE World Conference on Factory Communication Systems (WFCS), 2016, May.

15. *Towards Automated Deployment of IEC 61131-3 Applications on Multi-Core Systems*, Saad Mubeen, Matthias Becker, Xiaosha Zhao, Lingjian Gan, Moris Behnam, Thomas Nolte. In the Proceedings of the 12th IEEE World Conference on Factory Communication Systems (WFCS), Work-in-Progress (WiP) Session, 2016, May.

16. *A Many-Core Based Execution Framework for IEC 61131-3*, Matthias Becker, Kristian Sandström, Moris Behnam, Thomas Nolte. In the Proceedings of the 41st Annual Conference of the IEEE Industrial Electronics Society (IECON), 2015, November.

17. *Improved Priority Assignment for Real-Time Communications in On-Chip Networks*, Meng Liu, Matthias Becker, Moris Behnam, Thomas Nolte. In the Proceedings of the 23rd International Conference on Real-Time Networks and Systems (RTNS), 2015, November.

18. *Adaptive Routing of Real-Time Traffic on a 2D-Mesh Based NoC*, Matthias Becker, Meng Liu, Moris Behnam, Thomas Nolte. In the Proceedings of the 21st IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Work-in-Progress (WiP), 2015, August.

19. *Partitioning the Network-on-Chip to Enable Virtualization on Many-Core Processors*, Matthias Becker, Dakshina Dasari, Vincent Nélis, Moris Behnam, Thomas Nolte. In the Proceedings of the 6th Real-Time Scheduling Open Problems Seminar (RTSOPS), 2015, July.

20. *Extended Support for Limited Preemptive Fixed Priority Scheduling for OSEK/AUTOSAR Compliant Operating Systems*, Matthias Becker, Nima Khalilzad, Reinder J. Bril, Thomas Nolte. In the Proceedings of the 10th IEEE International Symposium on Industrial Embedded Systems (SIES), 2015, June.

21. *Towards Improved Dynamic Reallocation of Real-Time Workloads for Thermal Management on Many-Cores*, Rafia Inam, Matthias Becker, Moris Behnam, Thomas Nolte, Mikael Sjödin. In the Proceedings of the 35th IEEE Real-Time Systems Symposium (RTSS), Work-in-Progress (WiP) Session, 2014, December.

22. *Challenges of Virtualization in Many-Core Real-Time Systems*, Matthias Becker, Mohammad Ashjaei, Moris Behnam, Thomas Nolte. ACM SIGBED Review: Special Issue on the 7th Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS), Vol 12, nr 2, 2015, April.

23. *Limiting Temperature Gradients on Many-Cores by Adaptive Reallocation of Real-Time Workloads*, Matthias Becker, Kristian Sandström, Moris Behnam, Thomas Nolte. In the Proceedings of the 19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2014, September.
**IEEE Industrial Electronics Society Scholarship Award.**

24. *Saving Energy by Means of Dynamic Load Management in Embedded Multicore Systems*, Matthias Becker, Adriaan Schmidt, Martin Orehek, Thomas Nolte. In the Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES), 2014, June.

25. *Mapping Real-Time Tasks onto Many-Core Systems Considering Message Flows*, Matthias Becker, Kristian Sandström, Moris Behnam, Thomas Nolte. In the Proceedings of the 20th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Work-in-Progress (WiP) Session, 2014, April.

26. *Dynamic Power Management for Thermal Control of Many-Core Real-Time Systems*, Matthias Becker, Kristian Sandström, Moris Behnam, Thomas Nolte. ACM SIGBED Review: Special Issue on the 6th Workshop on Adaptive and Reconfigurable Embedded Systems (APRES), Vol 10, nr 3, 2014, October.

27. *Increased Reliability of Many-Core Platforms Through Thermal Feedback Control*, Matthias Becker, Kristian Sandström, Moris Behnam, Thomas Nolte. In the Proceedings of the Workshop Performance, Power and Predictability of Many-Core Embedded Systems (3PMCES), 2014, March.

# Contents

# I

# Thesis

# Chapter 1

# Introduction

Over the past decades, computing devices became omnipresent in the way we perceive and interact with our environment. Nowadays, many application areas are unthinkable without the help of computing devices. In contrast to the personal computer, the computer in these systems is embedded and often not directly visible to the user. Hence, they are called embedded systems. Examples can be found from areas such as avionics or automotive to medicine. This becomes evident if considering that around 98% of all produced processors find their use in embedded systems [1]. This broad presence of embedded systems all around us warrants a deep study towards their safe and efficient usage [2]. As many of these systems interact with their environment to control or observe it, they can only perform correctly if the results of the performed computations are provided at correct times. This type of system is called real-time embedded system and is at the heart of this thesis.

Automotive systems constitute around 20% of the embedded systems market [3]. These systems underwent a surge in innovation and transitioned from basic transportation units to sophisticated systems [4]. The fast increase in functionality implemented in these systems lead to a large number of embedded computers, the Electronic Control Units (ECUs), that are connected by multiple networks [5]. A modern car for example hosts more than 100 ECUs [6, 7]. Recent efforts to circumvent this proliferation of the ECUs target the system architecture, going from distributed to integrated architectures [8, 9, 10]. This allows system designers to consolidate previously distributed functionality on fewer, more powerful, ECUs. The main advantages that come with this architecture are reduced system complexity, simplified integration of functionality,

and ultimately cost savings [11]. However, automotive software is highly complex and highly interconnected. This can be seen in, for example, the engine management system that requires timely interaction of thousands of *software components* that communicate over tens of thousands of communication signals [12].

A software component is the basic building block of software architecture, and may correspond to a task at runtime. Such applications are often modeled with chains of software components, e.g., a chain from sensor to actuator. Together, the software components of a chain perform a joint action. Integrating a number of these applications on the same hardware platform poses several challenges, as the functional and temporal integrity of the application must be maintained after integration. Additionally, these applications are subject to timing requirements such as end-to-end delay constraints on the data propagation through a chain of semantically related tasks [13]. Tasks in one such chain can be executed at different periods, together with the independence of tasks in regards to scheduling decisions. Several semantics can be defined for these constraints. For example, for control applications it is important that the data which is used for control decisions is not outdated, otherwise control performance is degraded [14, 15]. On the other hand, the first reaction of the system to an input event is important for so-called "*button to reaction*" applications.

One promising hardware platform to consolidate multiple automotive applications is the many-core processor [16, 17, 18, 19]. A many-core processor hosts tens or even hundreds of CPUs on a single chip [20]. The CPUs are typically arranged on so-called tiles, where one tile can host one or multiple CPUs, together with local memory. The Network-on-Chip (NoC) is the interconnect of choice for such systems as it provides a higher bisection bandwidth and better scalability than traditional bus- or ring-based interconnects [21]. Each of the tiles provides sufficient computational power to host an automotive application. We refer to a tile as "*compute cluster*", if the tile hosts a number of processing cores and local memory resources. This means, one many-core processor can host several automotive applications where each application can execute within one of these natural partitions.

In this thesis we investigate methods to consolidate automotive applications on a clustered many-core processor (where each tile hosts a number of CPUs and local memory), while maintaining their real-time properties. While several works advocate the suitability of many-core processors in the automotive domain [16, 17, 18, 19], to the best of our knowledge, this is the first work to provide a detailed investigation of methods to execute automotive real-time applications on many-core processors. Most related to our work is the work by

Perret et al. [22, 23, 24, 25] which addresses safety-critical avionics applications on a many-core platform.

Several factors affect a successful integration of applications on a many-core platform. While each application remains within a tile, the applications need to access the NoC to either communicate with each other, to access external memory, or to access other peripherals that connect the processor to its environment. A novel NoC organization based on symmetric partitioning is proposed that allows one to reduce the contention on the NoC and thus reduces message delays. Shared resources are also a challenge within a tile, as the access to shared memory is shown to be one of the main bottlenecks. A tile organization is proposed that is based on privatization of memory banks for local execution, together with sharing of memory banks for tile-local communication. A contention-free time-triggered scheduling is used to orchestrate the access to the shared memory resources. We propose an ILP formulation of the scheduling problem that is optimal in the sense that, if a solution exists it is found. Additionally, a memory-centric scheduling heuristic is devised that can generate the time-triggered schedule within a fraction of the time required by the ILP, while only sacrificing 0.5% of the average schedulable utilization.

An extension to the framework classifies tasks into two types, static and dynamic tasks, where static task's code is mapped to a dedicated memory bank within the tile. This utilizes the available memory better while it removes the need to pre-load the task's code. A Constrained Programming (CP) formulation is used to generate the classification of tasks, their mapping, and the generation of the time-triggered schedule. An improvement in schedulability of up to 19% is observed w.r.t. the contention-free execution framework.

Incorporating the end-to-end delay constraints on the data propagation through chains of tasks aggravates the generation of the time-triggered schedule. We address this challenge by first developing a novel method to calculate all possible data propagation paths that may be observed for such a chain (agnostic of hardware platform and scheduling decisions). These paths are then used to compute upper bounds on the different end-to-end delay semantics. For cases where the unconstrained system leads to missed end-to-end deadlines we augment the system model with job-level dependencies (a partial execution order of the task's jobs). We show that, as long as these job-level dependencies are met, the end-to-end delay constraints are always satisfied. Hence, the complex end-to-end delay constraints are translated into manageable job-level dependencies that can be used during the schedule generation.

To cater the needs to perform timing analysis for end-to-end delays at various stages of the system development process we extend the approach to con-

sider the concrete system information that is available at the respective stage. This then reduces the pessimism in the timing analysis by an increase in the system information.

## 1.1 Thesis Overview

This thesis consists of two main parts. The first part provides an introduction to the overall work, where Chapter 2 introduces background information on the research area. Chapter 3 presents research overview, consisting of the thesis goals, the research challenges, as well as corresponding technical contributions, followed by the research methodology. Finally, Chapter 4 draws the final conclusions and present an outlook on future work. A collection of seven research papers constitutes the second part of the thesis that describe the research results. A summary of the included research papers is as follows:

**Paper A:** *Investigation on AUTOSAR-Compliant Solutions for Many-Core Architectures*. Matthias Becker, Dakshina Dasari, Vincent Nélis, Moris Behnam, Luís Miguel Pinho, Thomas Nolte. Published in the Euromicro Conference on Digital System Design (DSD), 2015, August.

**Abstract:** As of today, AUTOSAR is the de facto standard in the automotive industry, providing a common software architecture and development process for automotive applications. While this standard is originally written for singlecore operated Electronic Control Units (ECU), new guidelines and recommendations have been added recently to provide support for multicore architectures. This update came as a response to the steady increase of the number and complexity of the software functions embedded in modern vehicles, which call for the computing power of multicore execution environments. In this paper, we enumerate and analyze the design options and the challenges of porting AUTOSAR-based automotive applications onto multicore platforms. In particular, we investigate those options when considering the emerging many-core architectures that provide a more scalable environment than the traditional multicore systems. Such platforms are suitable to enable massive parallel execution, and their design is more suitable for partitioning and isolating the software components.

**Paper B:** *Synthesizing Job-Level Dependencies for Automotive Multi-Rate Effect Chains*. Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, Thomas Nolte. Published in the 22nd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2016, August.

**Abstract:** Todays automotive embedded systems comprise a multitude of functionalities, many with complex timing requirements. Besides task specific timing requirements, such applications often have timing requirements for the propagation of data through a chain of tasks. An important metric for control applications is the data age, which is addressed in this work. The analysis of such systems is non-trivial because tasks involved in the data propagation may execute at different periods, which leads to over and undersampling within one chain. This work presents a novel method to compute worst- and best-case end-to-end latencies for such systems. A second contribution synthesizes job-level dependencies for such task sets in a way that data paths which exceed the age constraint are eliminated. An extensive evaluation is performed on synthetic task sets and the applicability to industrial applications is demonstrated in a case study.

**Paper C:** *A Generic Framework Facilitating Early Analysis of Data Propagation Delays in Multi-Rate Systems*. Matthias Becker, Saad Mubeen, Dakshina Dasari, Moris Behnam, Thomas Nolte. Published in the 23rd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2017, August.

**Abstract:** A large majority of multi-rate real-time systems are constrained by a multitude of timing requirements, in addition to the traditional deadlines on well-studied response times. This means, the timing predictability of these systems not only depends on the schedulability of certain task sets, but also on the timely propagation of data through the chains of tasks from sensors to actuators. In the automotive industry, four different timing constraints corresponding to various end-to-end data propagation delays are commonly specified on the systems. This paper identifies and addresses the source of pessimism as well as optimism in the calculations for one such delay, namely the reaction delay, in the state-of-the-art analysis that is already implemented in several industrial tools. Furthermore, a generic framework is proposed to compute all the four end-to-end data propagation delays, complying with the established delay semantics, in a scheduler and hardware-agnostic manner.

This allows analysis of the system models already at early development phases, where limited system information is present. The paper further introduces mechanisms to generate job-level dependencies, a partial ordering of jobs, that need to be satisfied by any execution platform in order to meet the end-to-end timing requirements. The job-level dependencies are first added to all task chains of the system and then reduced to its minimum required set such that the job order is not affected. Moreover, a necessary schedulability test is provided, allowing for varying the number of CPUs. The experimental evaluations demonstrate the tightness in the reaction delay with the proposed framework as compared to the existing state-of-the-art and practice solutions.

**Paper D:** *End-to-End Timing Anaysis of Cause-Effect Chains in Automotive Embedded Systems*. Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, Thomas Nolte. In Journal of Systems Architecture (JSA), Vol. 80 (Supplement C), Elsevier, 2017, October.

**Abstract:** Automotive embedded systems are subjected to stringent timing requirements that need to be verified. One of the most complex timing requirement in these systems is the data age constraint. This constraint is specified on cause-effect chains and restricts the maximum time for the propagation of data through the chain. Tasks in a cause-effect chain can have different activation patterns and different periods, that introduce over- and under-sampling effects, which additionally aggravate the end-to-end timing analysis of the chain. Furthermore, the level of timing information available at various development stages (from modeling of the software architecture to the software implementation) varies a lot, the complete timing information is available only at the implementation stage. This uncertainty and limited timing information can restrict the end-to-end timing analysis of these chains. In this paper, we present methods to compute end-to-end delays based on different levels of system information. The characteristics of different communication semantics are further taken into account, thereby enabling timing analysis throughout the development process of such heterogeneous software systems. The presented methods are evaluated with extensive experiments. As a proof of concept, an industrial case study demonstrates the applicability of the proposed methods following a state-of-the-practice development process.

**Paper E:** *Contention-Free Execution of Automotive Applications on a Clustered Many-Core Platform*. Matthias Becker, Dakshina Dasari, Borislav Nicolić, Benny Åkesson, Vincent Nélis, Thomas Nolte. Published in the 28th Euromicro Conference on Real-Time Systems (ECRTS), 2016, July.

**Abstract:** Next generations of compute-intensive real-time applications in automotive systems will require more powerful computing platforms. One promising power-efficient solution for such applications is to use clustered many-core architectures. However, ensuring that real-time requirements are satisfied in the presence of contention in shared resources, such as memories, remains an open issue. This work presents a novel contention-free execution framework to execute automotive applications on such platforms. Privatization of memory banks together with defined access phases to shared memory resources is the backbone of the framework. An Integer Linear Programming (ILP) formulation is presented to find the optimal time-triggered schedule for the on-core execution as well as for the access to shared memory. Additionally a heuristic solution is presented that generates the schedule in a fraction of the time required by the ILP. Extensive evaluations show that the proposed heuristic performs only 0.5% away from the optimal solution while it outperforms a baseline heuristic by 67%. The applicability of the approach to industrially sized problems is demonstrated in a case study of a software for Engine Management Systems.

**Paper F:** *Partitioning and Analysis of the Network-on-Chip on a COTS Many-Core Platform*. Matthias Becker, Borislav Nicolić, Dakshina Dasari, Benny Åkesson, Vincent Nélis, Moris Behnam, Thomas Nolte. Published in the 23rd IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2017, April.

**Abstract:** Many-core processors can provide the computational power required by future complex embedded systems. However, their adoption is not trivial, since several sources of interference on COTS many-core platforms have adverse effects on the resulting performance. One main source of performance degradation is the contention on the Network-on-Chip, which is used for communication among the compute cores via the off-chip memory. Available analysis techniques for the traversal time of messages on the NoC do not consider many of the architectural features found on COTS platforms. In this work, we target a state-of-the-art many-core processor, the Kalray MPPA®. A novel partitioning strategy for reducing the contention on the

NoC is proposed. Further, we present an analysis technique dedicated to the proposed partitioning strategy, which considers all architectural features of the COTS NoC. Additionally, it is shown how to configure the parameters for flow-regulation on the NoC, such that the Worst-Case Traversal Time (WCTT) is minimal and buffers never overflow. The benefits of our approach are evaluated based on extensive experiments that show that contention is significantly reduced compared to the unconstrained case, while the proposed analysis outperforms a state-of-the-art analysis for the same platform. An industrial case study shows the tightness of the proposed analysis.

**Paper G:** *Managing Complex Automotive Applications on Clustered Many-Core*. Matthias Becker, Saad Mubeen, Dakshina Dasari, Moris Behnam, Thomas Nolte. In Proceedings of the 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), 2018, January.

**Abstract:** Access to shared memory is one of the main challenges for many-core processors. One group of scheduling strategies for such platforms focuses on the division of tasks' access to shared memory and code execution. This allows to orchestrate the access to shared local and off-chip memory in a way such that access contention between different compute cores is avoided by design. In this work, an execution framework is introduced that leverages local memory by statically allocating a subset of tasks to cores. This reduces the access times to shared memory, as off-chip memory access is avoided, and in turn improves the schedulability of such systems. A Constraint Programming (CP) formulation is presented to selects the statically allocated tasks and generates the complete system schedule. Evaluations show that the proposed approach yields an up to 19% higher schedulability ratio than related work, and a case study demonstrates its applicability to industrial problems.

# Chapter 2

# Background

This chapter provides the required background information of the relevant work included in the thesis.

## 2.1 Embedded Systems

Embedded systems can be found in almost all electronic products around us. They enhance existing systems by replacing more costly or complex mechanical solutions, or they add new features to the existing systems. All this while (mostly) staying hidden from the end users. Bar and Massa define an embedded system as follows:

**Definition 2.1.** *"An embedded system is a combination of computer hardware and software – and perhaps additional parts, either mechanical or electronic – designed to perform a dedicated function."*
*Michael Barr, Anthony Massa [26]*

The embedded industry is constantly growing, with an estimated market size of 258,72 billion USD by 2023 [3]. This is not surprising, as the vast majority of all produced processors is intended for the usage in embedded systems [27]. According to the study in [3], three main industries comprise the majority of the embedded systems market, namely, *automotive*, *healthcare*, and *military and aerospace*. Alone in the automotive domain there is a clear shift towards Electric/Electronics (E/E) solutions, with more than 90% of all innovations brought by embedded solutions already in 2010 [4].

## 2.2 Real-Time Embedded Systems

Due to their intrinsic connection with the physical world, many embedded systems are subject to real-time requirements. It is of importance that the embedded system reacts to stimuli events, either produced by the user or the environment it controls, within a prescribed time. Thus, the importance of the relation of computation and physical time is elevated compared to traditional computer systems one might be familiar with. A definition is provided by Stankovic and Ramamritham:

**Definition 2.2.** *"Real-time systems are computing systems that must react within precise time constraints to events in the environment. As a consequence, the correct behavior of these systems depends not only on the value of the computation but also on the time at which the results are produced."*
*John A. Stankovic, Krithi Ramamritham [28]*

Hence, such systems are subject to strict timing requirements. In order to obtain an efficient and fault-free development of such systems it is important to avoid ad-hoc methods. Instead, well-studied design principles that rely on well-defined application models which allow for static code analysis should be used. Together with operating system mechanisms that are tailored for these systems it is possible to analyze the systems timing properties [2].

### 2.2.1 Real-Time Basic Task Model

Having their origin in control applications, where a control function needs to be executed periodically, real-time systems are generally designed for one or multiple periodic tasks. Other task models such as the sporadic task model or the aperiodic task model are conceived to better represent characteristics of the systems. In this thesis we focus on periodic tasks. A task $\tau_i$ comprises certain functionality which is repeated in periodic time intervals $T_i$. Since the hardware platform is potentially shared by multiple tasks, the exact interval the task executes is unknown from the system model alone. Each task further has a deadline $D_i$ by which the computation needs to be completed. In this work we consider *implicit deadlines*, meaning that the deadline is equal to the task's period ($D_i = T_i$). The execution time of the task in isolation is upper bounded by $C_i$. Successive executions of a task are called *jobs*. The $j^{th}$ job of a task $\tau_i$ is denoted by $\tau_{i,j}$. The absolute release time of this job can be

defined as $rel_{i,j} = T_i \cdot (j-1)$. Fig. 2.1 depicts the task and job parameters.

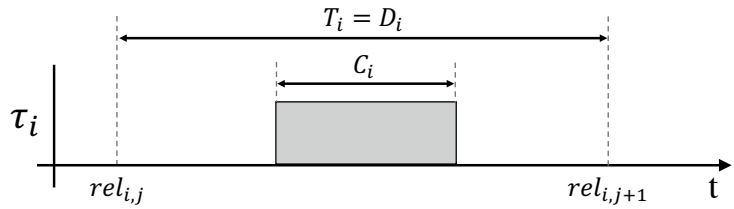



Figure 2.1: Illustration of the task and job parameters.

### 2.2.2 Worst-Case Execution Time Analysis

Finding the *Worst-Case Execution Time* (WCET), $C_i$, of a task on a specific platform is crucial in order to validate timing properties of the real-time embedded system. Several factors affect the WCET of a task. Different execution paths within the code may lead to different execution times, for example if a certain code region is visited only if a physical input to the embedded system exists. Also the architectural elements of the processor may affect the required execution time of the task (i.e., caches, pipelines, branch prediction, shared resources, etc.). Wilhelm et al. provide an overview of methods developed for WCET analysis in [29]. In this thesis we assume that WCET estimates are given.

### 2.2.3 Task Scheduling in Real-Time Systems

The scheduler is responsible to grant access to the computational resource (i.e., the CPU) to the tasks. The scheduler can be classified using several properties. *Non-preemptive* systems are those systems where a task's job runs to completion once it gets granted access to the computational resource. On the other hand, *preemptive* systems are systems where the scheduler can temporarily stop the execution of a task's job in order to grant access to the computational resource to other tasks.

Scheduling decisions can either be made *online*, or they can be precomputed *offline* and stored on the device for execution. Typically, online

scheduled systems are more flexible than offline scheduled systems. Since offline schedules are pre-computed it is possible to use powerful servers to generate the schedule, taking all application details into account. Online schedulers on the other hand need to use the available computational power of the embedded device to take decisions. An overview on single-core scheduling techniques is provided in [30], and multiprocessor scheduling techniques is reviewed in [31].

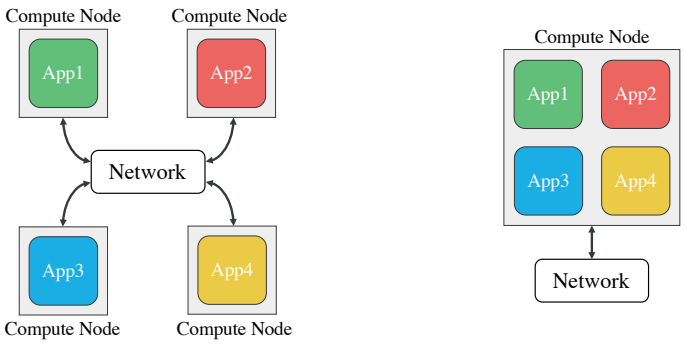
### 2.2.4 Timing Analysis

In order to validate the temporal correctness of a real-time system timing analysis needs to be performed. Timing analysis provides an analytical means to compute safe upper bounds on the response times of the tasks in a system. The response time of a task's job is defined as the difference between the job's finishing time and its release time. The worst-case response time of a task is the maximum response time among all its jobs. If response times of all tasks in a task set are smaller than or equal to their corresponding deadlines, the task set is said to be schedulable.

In their seminal work, Liu and Layland [32] provided utilization bounds for dynamic schedulers (namely Earliest Deadline First (EDF) and Rate Monotonic (RM)), and Joseph and Pandya developed the recursive response time calculations in [33], which provide an upper bound on the maximum response time a task can experience. For large task sets where the analysis time becomes intractable Bini et al. propose the hyperbolic bound [34]. A detailed survey is provided in [30].

### 2.2.5 From Federated to Integrated Architectures

Safety-critical embedded systems are generally designed in a federated manner, where each functionality is hosted on its own compute platform (see Fig. 2.2(a)). Bus-based interconnects or real-time networks are used to connect the different platforms to form the overall system. Prominent examples of application domains that are traditionally based on federated architectures can be found in the automotive domain, where functionality is spread over more than 100 ECUs [8, 6, 7], or the avionics domain [35, 36], and the industrial automation domain [37, 38]. The benefits of this architecture type are found in the safety properties of the system. As functionalities do not share local resources, fault propagation between them is minimal [39]. However, the




(a) Federated Architecture.

(b) Integrated Architecture.

Figure 2.2: Four applications in a federated and integrated system architecture.

growth of system complexity leads to systems that are difficult to maintain, where adding new functionality becomes increasingly challenging [40]. Integrating functionality on fewer computing platforms (see Fig. 2.2(b)) significantly reduces the required hardware (and thus reduces cost) and simplifies the system integration. This trend can be observed in the avionics domain, where the Integrated Modular Avionics (IMA) approaches allow system designers to replace distributed processors with fewer more powerful processors [35, 36]. Also in the automotive domain this trend is visible, where powerful processors are utilized to consolidate functionality of multiple, formerly distributed, ECUs [8, 41, 9, 10, 16, 17, 18].

## 2.3 Many-Core Processor

This section provides an overview of the many-core processor followed by a discussion of its distinctive hardware features.

### 2.3.1 Architectural Overview

Many-core processors are the newer manifestation of the traditional multi-core design. The two hardware paradigms have in common that they host a number of processing units, thus they provide the possibility to execute applications in parallel. A drastic change from multi- to many-core exists in the way the different processing units are connected. Where the multi-core utilizes bus/ring-based interconnects, the many-core implements the Network-on-Chip (NoC) [21]. With this shift in the design paradigm, the scalability problems of the bus/ring-based interconnect are tackled as the NoC provides a higher bisection bandwidth, and scales up to a large number of connected components [42, 43]. In this work, we define many-core processors as follows:

**Definition 2.3.** *A many-core processor hosts a large number of processing units (cores). The different processing units are allocated on so-called tiles, the tiles themselves are connected by on-chip networks.*

In Fig. 2.3, a typical architecture of the many-core processor is illustrated. The NoC connects each tile and the tiles themselves host processing elements and local memory.

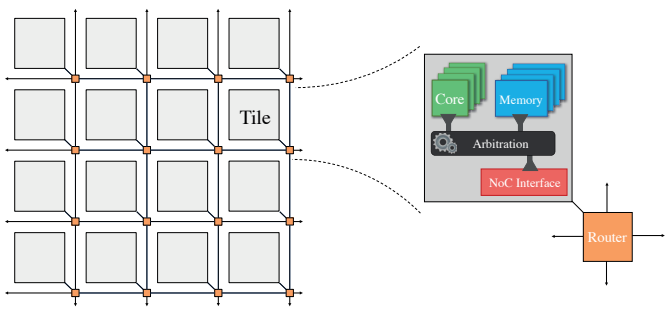


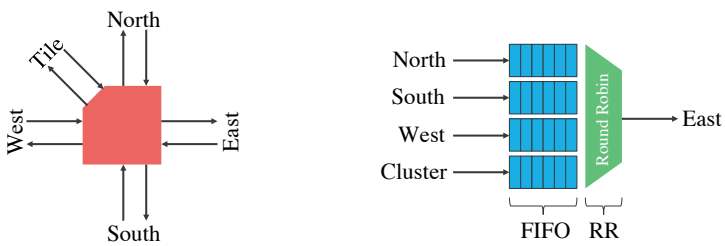

Figure 2.3: Illustration of a typical architecture of the many-core processor.

### 2.3.2 The Network-on-Chip as Backbone Network

**Wormhole Switching:** The NoC utilizes wormhole switching for moving data through the network [44]. In wormhole switching, a packet is divided into flow control digits (flits). A flit constitutes the elementary unit of transmission on the NoC and has a fixed size. During each clock cycle, each link on the NoC can transmit one flit. In addition to the packet payload, each NoC message includes a header flit that provides the necessary routing information. During transmission the header flit propagates through the NoC. Once the header proceeds from one router to the next, the remaining body flits can follow in a pipelined manner. In order to avoid buffer overflow in the router link-level flow control is commonly implemented. This means, a flit is not transmitted over a link if the required buffer space on the receiving side is not provided. During the transmission, the flits of one message can thus be spread out over multiple routers. Hence the name wormhole switching. One main advantage of this switching technique compared to store and forward switching is the reduced buffer requirements on routers, as not the whole message needs to be stored on a router [44].

**NoC Router:** The NoC router on a many-core processor dynamically arbitrates the access to links between the different messages. The most common dynamic arbitration mechanisms are Round Robin (RR) [45, 46], or Fixed Priority (FP) arbitration [47]. However, also Earliest Deadline First (EDF) was studied as arbitration policy [48]. In commercially available platforms the RR arbitration policy is most dominant [20, 49].

Buffers are used to store incoming flits before they can be transmitted on



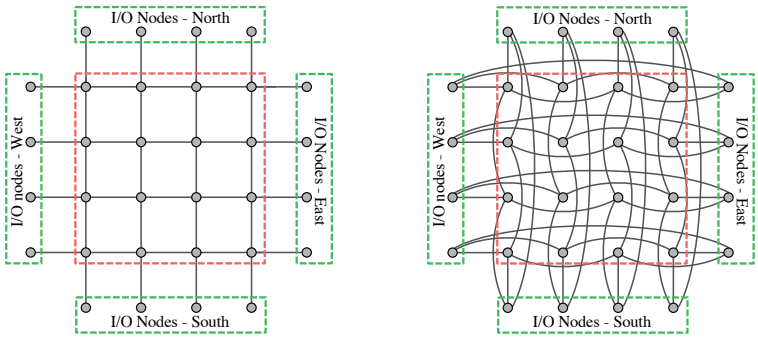

(a) NoC Router with communication links.

(b) Link arbitration on a router with output buffer.

Figure 2.4: Architecture of a NoC router with round robin based link arbitration.

the next link. Different schemes exist, for example Kalray's MPPA processor implements output buffer [20] (as illustrated in Fig. 2.4(b)), while Tilera's Tile Processor implements input buffer [49]. In [50] we showed that, from timing analysis point of view, the two designs are identical.

**NoC Topology:** The topology of the network dictates how the different network elements are connected to each other. Several network topologies are possible, see Fig. 2.5. On the many-core processor, a router generally has five ports. One for each cardinality plus an additional port to connect to the tile. Two topologies can be found, the 2D-mesh based NoC, where tiles are arranged on a 2D-grid and routers of neighboring tiles connect to each other, forming the NoC. A second topology that is found on many-core processors is the 2D-torus topology. Also here the tiles are arranged on a 2D-grid, but each row and column is connected by a torus, i.e. neighboring tiles do not share a direct connection (as seen in Fig. 2.5(b)).




(a) 2D mesh topology. (b) Torus topology.

Figure 2.5: Examples of most common NoC topologies on the many-core processor.

**Timing Analysis:** When used in the context of real-time embedded application, the messages that are transmitted on the NoC are subject to timing constraints. It is crucial to compute the Worst-Case Traversal Time (WCTT) of a NoC packet, i.e, the time it takes to transmit the NoC packet from source node to its destination node under consideration of all other packets that are transmitted on the same NoC. It is crucial to incorporate buffer effects in these computations. For a FP arbitrated NoC it was shown in [51] that the established real-time analysis based on [47] produces optimistic results (i.e., the computed upper bounds for the WCTT are smaller than the actual upper bound). We later

showed in [50] the same problem for the RR arbitrated NoC when the Recursive Calculus (RC) timing analysis of [45] is applied. The RC analysis was revised in [50] and the real-time analysis for the FP based NoC was revised in [52, 53].

### 2.3.3 The Tiles to Host the Computational Power

Connected by the NoC, the tiles provide a platform to host the processing units. Additionally a tile implements a network interface to connect to the NoC, as well as local memory. Typically, a many-core processor that is intended for real-time workloads does not implement shared cache between cores on the same tile [20].

Generally, a tile can host any element, such as processing cores, memory, or special purpose units. On the many-core processor tiles typically host one or multiple CPUs, and one or multiple memory banks. In this thesis we refer to tiles that host a number of compute cores, together with local memory, as *cluster*.

### 2.3.4 COTS Many-Core Processors

In this section we provide a brief description of a number of COTS many-core processors. Additionally, a more detailed description of the Kalray MPPA® is provided, as this platform is chosen as reference for the architectural features of a many-core processor in this thesis.

Several many-core processors are commercially available today. Examples are Mellanox's TileGx™ series of processors that host up to 72 identical processing cores [54], PEZY's PEZY-SC that hosts 1024 RISC cores as well as 2 ARM926 cores [55], Adapteva's Epiphany™ that is available with up to 64 RISC cores [56], or Kalray's Massively Parallel Processor Array (MPPA®) that hosts 256 processing cores, 16 resource management cores, and 16 I/O cores [20, 57]. Noteworthy is also Intel's research processor, the Single-Chip Cloud Computer (SCC) which hosts 48 P54C cores [58, 59].

The majority of today's many-core processors are developed for the high-performance computing domain where it is important that the average computational performance of the processor is maximized. This generally leads to uncertainties and thus difficulties in their timing analysis as many architectural elements used to improve average performance introduce unpredictability [60]. The MPPA® is designed with real-time applications in mind. Each

of the compute cores is designed in such a way that timing anomalies are eliminated. This means, the core-local worst-case is also the worst-case timing in the global scenario [61], and design decisions are made to best suite static timing analysis [62].

The MPPA® is a clustered many-core processor that hosts 256 compute cores that are available to the programmer for general-purpose computations. The compute cores are arranged on so-called compute cluster (tiles) in groups of 16. Hence one processor hosts 16 compute cluster. Additionally, each cluster contains one resource management core that is solely intended to manage processor resources on behalf of the entire cluster. In addition to the 16 compute cluster there are 4 so-called I/O subsystems. Each subsystem connects to the NoC over 4 independent NoC routers. The I/O subsystems connect to external resources, such as Ethernet or DDR memory.

The NoC on the MPPA® implements the 2D-torus topology. No flow control exists on link level. That is, the buffer inside the router can potentially overflow if more flits arrive than depart over a certain time period. To avoid such effects and to provide a means to guarantee service on the NoC to the different applications the MPPA® implements *flow regulation on source nodes* in the form of a packet shaper and traffic limiter that work in tandem. This means, the injection of messages on the NoC is regulated by the flow regulation on each node, thus making sure that buffers never overflow. Such an architecture then allows to apply timing analysis based on Network Calculus (NC) [63, 64]. However it was shown that these techniques lead to pessimistic estimates [65]. Additionally aggravating is the fact that the parameters to configure the flow regulation on each node need to be determined, which is not trivial.

In addition to the resource management core and the 16 compute cores, each of the compute cluster hosts a memory subsystem consisting of 16 SRAM memory banks (128 KB each) that are shared among all cores of the cluster. Memory request arbitration for each memory bank is done in a 4 stage process. The first three stages are RR based while the last stage is priority based, giving highest priority to the incoming NoC messages. A detailed discussion can be found in [66].

### 2.3.5 Challenges of Many-Core Processors in Real-Time Embedded Systems

The large number of shared resources on the many-core processor poses several challenges for real-time embedded systems [67]. Already multi-core archi-

tectures face multiple sources of unpredictability that aggravates their timing analysis [60]. This is amplified on the many-core processor due to the increase of clients that access the shared resources [68]. Approaches that assume an unconstrained system need to consider all possible interferences for each resource access.

Two approaches exist for the organization of the on-chip memory on a many-core processor, interleaved or blocked address mapping. In blocked address mapping consecutive memory addresses are mapped to the same bank, while in interleaved address mapping sequential addresses are spanning all memory banks evenly. Thus, requests to consecutive memory addresses are evenly distributed on all banks. While the interleaved address mapping can improve the average performance, the worst-case access times to the memory are often overly pessimistic as access patterns to the memory are difficult to determine. On the Kalray MPPA for example, a single memory request of a compute core to a cluster local memory bank has a worst case access time that is 63 times larger than the access time in isolation [66]. An investigation of the two variants is provided in [69].

Similar observations can be made on the NoC, where the worst case pattern of messages needs to be considered to upper bound the WCTT of messages.

## 2.4 Automotive Software Applications

Software is one of the driving forces for innovation in the automotive industry. While the number of implemented software functions is exponentially increasing to incorporate new features [4], the complexity of existing functionality is increasing at the same time [5]. A prime example is the Engine Management System (EMS), where complexity of implemented control functions is increasing to achieve the low fuel consumption dictated by the markets as well as to meet the strict emission guidelines imposed by governmental bodies [70, 71]. In this section we discuss the related standards that define how real-time automotive software applications are constructed, before the application model is discussed with a look into the data propagation delays, a specific type of end-to-end delay that is found in automotive applications.

### 2.4.1 Standards and Specifications in the Automotive Domain

To cope with the increasing system complexity, several abstraction levels were conceived by academia and industry [72, 73]. The abstraction levels are designed with the principle of separation of concerns, where each abstraction level provides a complete definition of the system in the context of its intended concern.

*EAST-ADL* [72] is an architecture description language that divides the development process of an automotive system into four layers, while the fifth layer is used for operational concerns. With feature modeling in the *Vehicle Model*, where the functionality is described in a solution independent way, followed by the *Analysis Level*, where the main focus lies on the consistency of requirements. The functional design of the system is addressed in the remaining two levels. The *Design Level* describes the functional deign and allocates the software components to hardware elements. Finally, the *Implementation Level* yields the concrete implementation of the functionality on the actual hardware platform. Only at the implementation level it is possible to analyze all system properties. A detailed description of the different levels can be found in [72]. The EAST-ADL methodology proposes to use AUTOSAR [73] at the implementation level.

*AUTOSAR*, the AUTomotive Open System ARchitecture, was created to provide a standardized software architecture for ECUs, promoting the scalability, transferability, collaboration among different vendors, and simplification of the system integration process. The software architecture can be divided into

three layers, the Basic Software, Runtime Environment, and the Application Software layer. At its heart, AUTOSAR deploys a Fixed-Priority scheduler which is built on the OSEK Operating System [74].

AUTOSAR is one possible option for the implementation layer of EAST-ADL. Besides AUTOSAR, the Rubus Component Model can be utilized at the implementation level [75].

First conceived without support for real-time requirements, the TIMMO-2-USE [76] project was started to develop a timing model for AUTOSAR. Outcome is the TADL2 language [77] that is used to describe timing constraints in both, AUTOSAR and EAST-ADL.

To meet verification requirements, an automotive application needs to adhere to safety norms stipulated by the ISO26262 safety standard [78]. Each function is assigned an ASIL (Automotive Safety Integrity Level) that is assigned based on the possible consequences of a failure (determined by hazard analysis [79]), the likelihood of occurrence of the failure, and the possible provision of means to handle the failure [80]. ASIL A is the lowest safety integrity level, whereas ASIL D is the highest safety integrity level in the standard.

### 2.4.2 Automotive Real-Time Applications

Automotive real-time applications are highly interconnected and highly complex. An EMS for example connects to 40-50 sensors and multiple actuators [70]. The source code defines around 2000 atomic software components[1], over 5000 source files and half a million lines of C-code. The most complex variants can require nearly 4000 runnables (the elementary schedulable units in AUTOSAR) that communicate over up to 60000 data labels [66, 12].

In this thesis we assume a one-to-one mapping of runnables to tasks, where on task is described by a period $T$, a WCET $C$, and an implicit deadline. Further, each task requires a set of data labels that need to be read during execution $\mathcal{R}$, and a set of data labels that are written during execution $\mathcal{W}$.

The data labels are used for communication between the tasks. Such a communication form does not require signaling between tasks and thus decouples their execution from communication. The most common mechanism that is used for the communication is the *implicit communication* [12]. In this communication paradigm, the task's execution is divided into three parts, *read, execute, write*. During the read phase, a local copy is created for all data labels $\in \mathcal{R}$. During the write phase all data labels $\in \mathcal{W}$ are written. During execution

[1] An atomic software component cannot be further divided into sub-components and represents the elementary parts of execution.

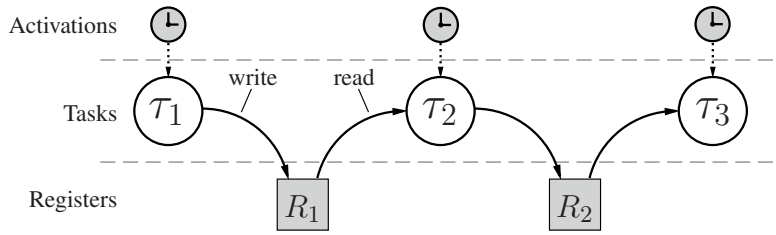



Figure 2.6: Illustration of register-based communication.

the task accesses the local copies of the data labels. This communication has the advantage that the task keeps a consistent view of the data labels during its execution, even if another task would change the global value of the data label in the meantime.

**Register Communication:** The communication is implemented using register-based communication [12]. Each data label is represented by a shared register (i.e., a global variable). A task sends and receives values by writing to and reading from the register respectively. This simplifies the communication between tasks that execute at different rates (i.e., periods), as there is no direct signaling between tasks in order to communicate. An example is shown in Fig. 2.6, where three tasks communicate over two shared registers.

### 2.4.3 End-to-End timing Requirements on Data Propagation Delays

In addition to the task's local deadlines, it is often important to meet timing constraints that are specified on data propagation delays across a semantically related chain of tasks [12, 71]. These delay semantics are defined in AUTOSAR, EAST-ADL, and TADL2 [81, 72, 77].

Different delay semantics can be defined for the data propagation through a chain of tasks. Fig. 2.7 illustrates the two main semantics, *data age* and *reaction delay*, at the example of a chain consisting of two tasks. The data propagates through the chain in the order $\tau_A \rightarrow \tau_B$, where $\tau_A$ has a period of 8 time units and $\tau_B$ has a period of 4 time units. Thus, oversampling can be observed between the two tasks. This leads to $\tau_B$ consuming the same input value multiple times. The data propagation is visualized by red arrows that origin in the sending job and terminate in the consuming job.

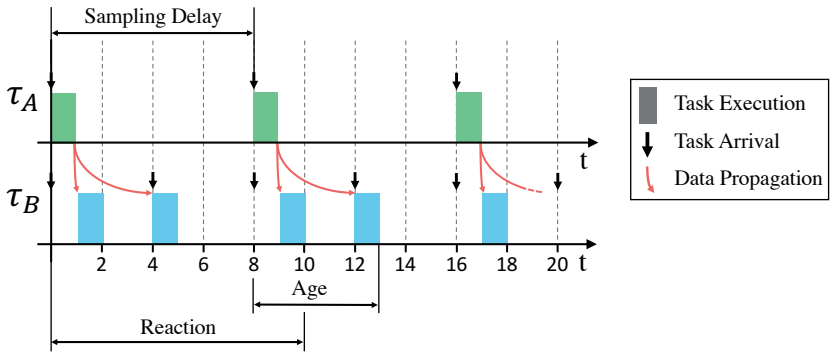



Figure 2.7: Overview of the data age and reaction delay semantics in a schedule of a multi-rate task chain.

The two delay semantics are defined in TADL2, where the reaction delay constraint is specified as follows:

**Definition 2.4.** *"A **reaction constraint** defines how long after the occurrence of a stimulus a corresponding response must occur."*
*TADL-2 [77]*

Such a delay is most important for "button to reaction" applications. Where an action needs to be performed within a certain time window after a trigger event. Note that the start of this delay is at the arrival of the event itself. The data age constraint is defined in TADL2 as:

**Definition 2.5.** *"An **age constraint** defines how long before each response a corresponding stimulus must have occurred."*
*TADL-2 [77]*

This delay type is most important for control applications, where it is important how old input data is. If the data in which the control decision is based on is too old the physical system might be in a different state already and the computed control action has reduced or even has negative effect on the control quality [14, 82].

**Timing Analysis:** Timing analysis is used to verify the timing requirements of the data propagation delays in register communication-based systems.

A method to compute upper bounds on the data propagation delays in these systems was first proposed in [13], and later in [83] it was shown how to compute such delays using model checking techniques. Mubeen et al. [84] discuss the implementation of end-to-end delay analysis in an industrial tool suite. Since then, the analysis of [13] is implemented and adopted in several industrial tools [85, 86, 87, 88]. All these implementations have in common that they require information that is only available at the implementation level. Thus, violations are discovered at a late development stage, where changes in the software are magnitudes more expensive to correct than at earlier stages [89, 90]. Few works address the timing analysis at higher abstraction levels, such as [91, 92, 93]. These works either rely on the reuse of previously developed software, or model transformations. Moreover, these works still leverage the timing analysis engines from the implementation level.

End-to-end delays are also considered during the system synthesis. In [94], task parameters are synthesizes to guarantee end-to-end deadlines in single core systems, and in [95] the approach is extended to distributed systems. The selection of task activation events (periodic or event-driven activation) is considered in [96]. Distributed systems are further considered in [97, 98, 99, 100].

In [101] we provide an analysis method for the data age delay that is applicable at the design level. This allows for an early detection of design flaws. This analysis method is implemented in MECHAniSer[2], an academic tool [102, 103] that targets the analysis, synthesis, and design of task chains at higher abstraction levels (see also Appendix A). This analysis method is later extended to support all end-to-end delay semantics [104][3] that are identified in [13].

[2] www.mechaniser.com

[3] Note that we use the terms *end-to-end delay* and *data propagation delay* interchangeable in this thesis.

# Chapter 3

# Research Overview

This chapter provides an overview of the doctoral thesis. First, the thesis goal is formulated, before smaller subgoals are derived. After a holistic discussion of the proposed process, the individual contributions of the thesis are presented before they are related to the subgoals. A discussion of the research process and methodology that is applied concludes this chapter.

## 3.1 Goal of the Thesis

With the increasing complexity of automotive systems, and the transition from federated to integrated architectures, many-core processors gain in relevance for this category of real-time embedded systems. While this highly parallel hardware platform can be used either by one application that executes in parallel on all available hardware resources, or by multiple applications that are consolidated on the different tiles. In this thesis, we focus on the latter as the consolidation of applications is a relevant industrial problem that is in line with current automotive trends [41, 9, 10].

In addition, a many-core processor provides multiple natural partitions, the tiles, that can be used to provide exclusive execution spaces for applications. This is further amplified by the tile-local memory that cannot be accessed by cores on a different tile. Hence some natural building blocks towards temporally and spatially separated execution spaces for the different applications are given by the hardware platform, which is important for standards such as ISO 26262 [78].

This thesis aims to enable the consolidation of real-time embedded applications from the automotive domain on a many-core processor. Under such a target, we define the main research goal as follows:

> ***To provide methods for resource efficient consolidation of automotive applications on clustered many-core architectures, while maintaining their real-time properties.***

Considering the high number of shared resources on such a platform, and thus their potentially large impact on real-time performance, together with the highly complex and interconnected automotive applications, that must adhere to complex timing requirements, different research challenges can be defined as follows:

**RC1: Identify important aspects of automotive software on a many-core processor**
This research challenge focuses on the suitability of many-core processors to host automotive applications, specifically issues of parallelism, safety, and analysability are of interest, both for execution on the many-core but also for communication on the NoC that connects the different tiles. The following research challenges are based on these aspects.

**RC2: Achieve mechanisms for predictable execution within a tile of the many-core processor**
The tile of the many-core processor was identified in RC1 as a natural partition. Hence it is ideally suited to host one automotive application. This research challenge focuses on the development of mechanisms to execute automotive applications in a predictable manner, while minimizing the otherwise substantial contention on tile-local resources such as local memory or NoC access which is required to access external resources (such as external memory or communication with other applications).

**RC3: Develop methods to support predictable deployment of multiple applications on the many-core**
In order to consolidate multiple applications on the many-core processor it is crucial to address the NoC interconnect. While the applications are mapped to individual tiles, the NoC remains a shared resource. Typically the tile on a many-core processor is resource constrained and hosts limited local memory. Thus, applications on one tile need to access the off-

chip memory to preload code at runtime over the NoC, but also to access other external resources or to send communication messages. Hence, this research challenge focuses on methods that address the challenges that arise when multiple applications share the common interconnect in order to guarantee predictable, safe, and efficient usage of this shared resource.

**RC4: Develop methods to guarantee complex timing requirements on data propagation delays**
Automotive applications are subject to complex timing requirements that add constraints on the data propagation delays through a chain of independently triggered tasks. Such tasks can be executed at different periods which introduces over- and under-sampling effects. Thus, it is not known at design time over which instances of the tasks the data propagates. Consequently, the data propagation delays are complicated to analyze without the knowledge of a concrete schedule. This challenge focuses on the generation of a partial ordering of the task instances that are involved in the data propagation, such that the data propagation delay constraints are met. Such an ordering allows to account for these delay types already during schedule construction.

**RC5: Develop techniques to perform timing analysis of the application under the developed framework**
The time-critical nature of the considered class of automotive applications mandates timing analysis to verify the specified timing requirements. This requires methods to verify the timing properties for task's local timing requirements, requirements on the data propagation through a chain of semantically related tasks, as well as the traversal times of messages on the NoC.

## 3.2 Technical Contributions

This section provides an overview of the contributions included in this thesis. First, an overview is provided to present a holistic view of the different contributions and how they together work towards the main goal of consolidating automotive real-time applications on clustered many-core platforms. After, the high-level connections are presented and the individual contributions are discussed in detail.

### 3.2.1 Overview of the Proposed Approach

The different methods that are proposed in this thesis contribute to a holistic framework that targets the consolidation of automotive real-time applications on a clustered many-core platform such as the Kalray MPPA®.

As the objective of the approach is to consolidate existing applications on a clustered many-core platform it is important that different applications do not negatively influence each other. Therefore, in our approach one application is mapped to one cluster. A cluster provides a dedicated execution space for the application with isolated compute and memory resources [20, 105]. A one-to-one mapping of applications to clusters thus exploits this natural partitioning of the hardware resources and it further facilitates incremental changes in the system (i.e., add or remove an application) without the need to revisit previously allocated applications.

Fig. 3.1 illustrates the proposed process to consolidate automotive real-time applications on a clustered many-core platform.

(i) Each application is represented by a set of tasks and a set of cause-effect chains that represent a semantic relation on the data propagation among the tasks. As the applications represent real-time applications, both tasks and the data propagation through the chains are subject to specified timing constraints.

(ii) Considering temporal constraints on the data propagation over a chain of independently triggered tasks (of possibly different periods) during scheduling is a non trivial problem. Thus, a first step translates the timing constraints on the data propagation delay to precedence constraints of task's jobs (represented by job-level dependencies). This is done in a way such that, if the precedence constraints are met, the data propagation delay constraints are met as well. Thus, the later execution on the

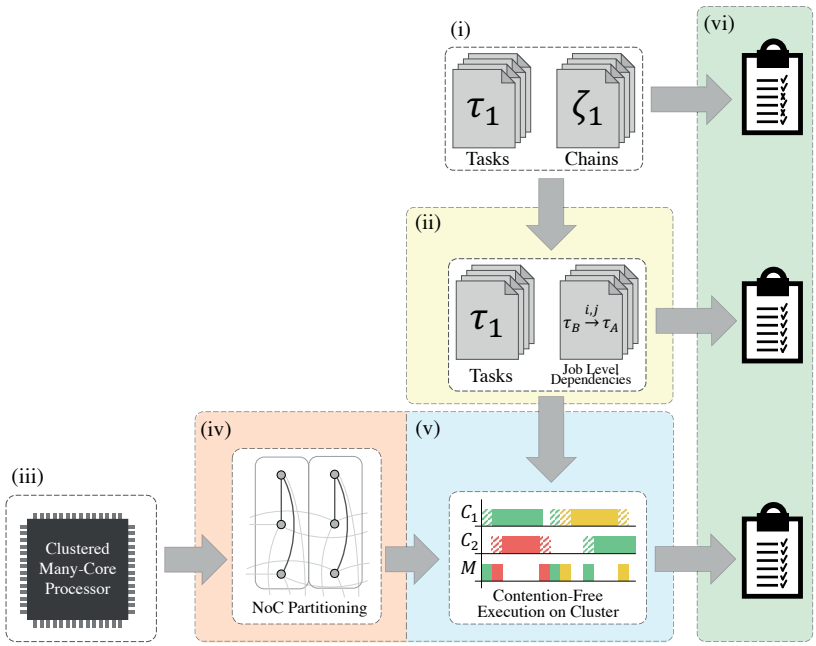



Figure 3.1: Holistic overview of the proposed methods to consolidate automotive real-time applications on a clustered many-core platform. From top to bottom, the integration process of one application on one cluster of the many-core processor is visualized. From left to right, the methods to provide predictable execution on the clustered many-core platform are shown, from the global NoC partitioning over the cluster-local contention-free execution framework.

hardware platform does only need to consider the specified precedence constraints.

(iii) The target hardware platform consist of a clustered many-core. On this type of platform, clusters host a number of CPUs as well as a local memory. The different clusters communicate with each other and access the external memory over a NoC.

(iv) The NoC provides a scalable interconnect that caters the need of many-core processors. However, messages from one cluster might interfere

with messages sent by another cluster when they share common resources on the NoC. Changing one application on one cluster thus potentially impacts on the WCTT of messages sent by another cluster. To preserve the isolation between different applications, a symmetric NoC partitioning is proposed that divides the NoC into isolated partitions, each having access to the external memory. Thus, this is an incremental part to provide isolation between clusters as it allows to upper bound the access time to external memory without knowledge of other applications behavior.

(v) An execution framework on the cluster is responsible for a predictable execution of the application on the hardware platform. By taking the tasks execution semantics into account the framework orchestrates the memory access using a time-triggered schedule. This is done in a way such that no two cores access the shared memory (local or external to the cluster) at a time. As tasks are executed based on a time-triggered schedule, the verification of timing constraints is given by construction. That is, if a time-triggered schedule can be constructed, all timing constraints are met.

(vi) Timing analysis is crucial to verify that all temporal constraints are met. The verification of the more complex constraints on the data propagation through cause-effect chains is supported at various levels of system information (with increasing precision) and can thus be performed throughout the development process.

### 3.2.2 Discussion of Individual Contributions

The technical contributions presented in this thesis can be grouped into five main contributions[1].

**C1: Identifying challenges of automotive applications on many-core platforms**
The increased complexity of today's automotive applications, together with the increased number of software applications that are implemented in a modern car, require powerful compute platforms. We investigate the many-core architecture as one possible solution for such systems.

[1]Note hat the technical contributions C2 to C5 are visualized in Fig. 3.1 by green, yellow, red and blue boxes respectively.

In [105], we analyze AUTOSAR, as de facto standard in the automotive industry, on multi- and many-core platforms. Adapting the AUTOSAR environment to multi- or many-core platforms is not trivial and direct solutions result in performance penalties that diminish the gain of the additional compute cores, as shown by Böhm in [106]. We propose a number of possible design options of the AUTOSAR software architecture on such platforms and discuss their possible performance impacts.

Additionally to the software architecture, the hardware architecture impacts on several properties of the aplications. In particular, we investigate the suitability of scenarios where multiple applications are consolidated on a single platform. Issues such as parallelism, safety and analyzability are highlighted. It is observed that the many-core processor provides natural partitions (compute clusters) with compute cores and local memory that can be used to host applications in isolation. The NoC, however, is a shared resource between the different clusters as it is used for inter-application communication as well as to access off-chip resources such as external memory, actuators, and sensors.

The temporal properties of messages sent over the NoC are an integral part of the framework to guarantee the temporal correctness of the applications on the clusters. Architectural features of the NoC such as buffer or flow regulation often exacerbate the timing analysis [50]. Additionally, the traversal time of messages on the NoC depends on other messages. Thus, changing one application may impact the timing behavior of a second application on a different cluster solely due to the impact on NoC messages [107].

Finally, the compute clusters themselves pose challenges as multiple compute cores are available that access shared resources such as local memory or the network interface. For example, access to the local memory banks of the the Kalray MPPA® can take up to 63 times longer if the worst case access pattern is observed compared to access in isolation [66].

**Targeting Research Challenge: RC1**
**Included Paper: Paper A, E, F, G**

**C2: Timing analysis of data propagation delays during various development stages**
For many applications it is not only crucial that individual tasks finish

their execution within their set deadlines, but also that data is propagated through a chain of semantically related tasks within a certain time. The timing analysis for these data propagation delay constraints was first presented by Feiertag et al. [13] and a model-checking based approach is presented in [83]. In contrast to previous work we present a timing analysis for data propagation delays in multi-rate systems under register communication that already applies at the design level. This is done first for the data age [101], as one of the main data propagation delay semantics, and later generalized to all data propagation delay semantics in [104]. The timing analysis is based on the concept of Data Propagation Trees (DPT), which represent a tree structure with task's jobs as nodes and edges between jobs that may communicate data, depending on the concrete schedule.

The information required to perform the timing analysis is very limited in the early development stages and increases throughout the development process. From the software architecture, to response times, and an exact schedule. Alternatively, execution semantics such as the Logical Execution Time (LET) model provide additionally information on the data communication. In [108] we show how this additional information can be used to augment the timing analysis of [101], which decreases the pessimism in the timing analysis when additional system information is present.

**Targeting Research Challenge: RC5**
**Included Paper: Paper B, Paper C, Paper D**

**C3: Transforming data propagation delay constraints into precedence constraints**

The integration of data propagation delay constraints into the scheduling level is cumbersome as these constraints focus on the timely propagation of data through a chain of semantically related tasks that can be triggered at different periods, while real-time scheduling generally focuses on the timely execution of consecutive instances of the same task [30].

Job-level dependencies, i.e., precedence constraints between selected jobs of two tasks, are one way to restrict the data propagation between dedicated task instances, which in turn can affect the data propagation. This is observed in [101], where it is shown how job-level dependencies can be used to prune branches of the DPT. We further present a

heuristic solution that augments an application with job-level dependencies in a way such that the specified data age constraints are met as long as the job-level dependencies are met. Hence, during scheduling, only the specified job-level dependencies need to be considered while all data propagation delay constraints are met by design. The advantage of systems that take the produced job-level dependencies into account during scheduling, compared to an approach that only focuses on the tasks local deadlines, is shown in [109].

In [104] we extend the mechanisms to augment an application model with job-level dependencies such that timing constraints on all different data propagation delay semantics are supported. Additionally, properties are observed under which specified job-level dependencies can be omitted without altering the specified ordering of jobs. This reduces the number of required job-level dependencies that need to be considered during scheduling.

**Targeting Research Challenge: RC4**
**Included Paper: Paper B, Paper C**

**C4: NoC partitioning to provide isolation between different clusters**
The NoC, which connects the clusters to each other as well as to external memory, is one shared resource, and thus the main source of interference that remains. Applications need to access external memory during their execution due to the limited amount of memory that is available within the cluster [66, 107, 110]. In [107], we propose a symmetric NoC partitioning, that effectively creates identical NoC islands. Different NoC partitions cannot impact on the traversal time of messages sent by clusters of other partitions. This allows to compute tight upper bounds on the traversal time of messages on the NoC and thus on the access time to external memory. It is demonstrated that messages that travel under the proposed NoC organization experience less contention on the NoC and thus travel faster than under a generic NoC usage. This, in turn, reduces overall access latencies to external resources.

The timing analysis of Round Robin arbitrated NoC is well studied [45], however, hardware elements such as buffer on NoC routers is often not considered in the timing analysis which makes the results unsafe to use in these architectures [50]. Augmenting the existing solutions to consider these effects increases the analysis complexity and may lead to overly

pessimistic results [50]. The NoC on the Kalray MPPA® implements a buffer on the NoC router, as well as flow regulation on each source node to limit the nodes injection rate. A general timing analysis for this NoC is presented in [63, 64]. Perret [24] propose a time-triggered transmission of messages on the NoC of the MPPA® in order to remove contention by design. In our scenario, a time-triggered solution to schedule the NoC traffic would require to regenerate the NoC schedule when a single application is changed. In contrast, we present a timing analysis for the proposed NoC partitioning that considers the hardware effects of flow regulation on source nodes and limited buffer size on the NoC routers. As the timing analysis is tailored to the partitioned case it provides tighter timing estimates than general timing analysis, while it has a constant time complexity.

Lastly, we show how to configure the flow regulation on source nodes for the partitioned scenario such that the buffers on the NoC routers cannot overflow. This is done in a way such that the minimal WCTT of messages is guaranteed.

**Targeting Research Challenge: RC3, RC5**
**Included Paper: Paper F**

**C5: Predictable execution of an automotive application on one cluster**

Contention on shared resources within one compute cluster of the many-core platform can impact the execution of tasks, and thus result in inflated execution times up to the point that the performance gain of the architecture is lost. Using dedicated memory access phases is a way to remove contention while accessing shared memory [111]. Such an access model is particularly suitable for industrial applications as automotive applications generally operate on a read-execute-write semantic [12], and a three phase task model is investigated as promising candidate for avionic applications [112, 113]. Both models divide the execution in three distinct non-preemptive phases, where read operations are carried out in the first phase, and write operations are carried out in the last phase. During the execution phase, only private memory is accessed.

In [66], we propose a novel cluster organization that divides the cluster-local memory banks in private and shared memory. Each compute core is assigned a private bank, while data that is shared among the differ-

ent tasks is located on the shared bank. To circumvent the memory limitations that prevent an allocation of the complete applications footprint on the cluster local memory, each task accesses external memory to prefetch the tasks code before execution of a job into the local memory banks. The execution is time-triggered in nature, where the time-triggered schedule is generated in a way that no two tasks on any compute core are in a memory phase at the same time. Hence, contention on shared memory is avoided by design. We propose an Integer Linear Programming (ILP) formulation that generates an optimal assignment of tasks jobs to cores and provides the time-triggered schedule. To overcome the scalability issues of the ILP formulation we propose a memory-centric heuristic that generates the task's jobs assignment and the time-triggered schedule in a fraction of the time of the ILP such that industrial sized problems can be solved.

Prefetching of task's code in every execution creates significant overheads, as access to external memory is expensive [107]. On the other hand, local memory banks are generally large which allows to store a number of task's footprints statically. While this reduces the prefetching overhead, the footprint's locality of these tasks binds the execution of all their jobs to the core that is associated with the respective memory bank, while other task's jobs can be executed on arbitrary cores. Additionally, we consider job-level dependencies that may be specified on tasks of the system. The execution framework is extended in [110] to additionally map task's code to local memory banks, if possible. A Constraint Programming (CP) formulation is presented that assigns task footprints to external or local memory, maps the task's jobs to cores and schedules all execution phases such that memory access phases never overlap. Experimental results show that this framework improves the schedulability ratio by up to 19%, compared to the original framework.

**Targeting Research Challenge: RC2, RC5**
**Included Paper: Paper E, Paper G**

**My Contribution:** I am the main driver and main contributor of all papers included in this thesis. The presented research work is done in collaboration with Dr. Dakshina Dasari, Dr. Vincent Nélis, Dr. Borislav Nicolić, Dr. Benny Åkesson, Professor Luís Miguel Pinho, Dr. Saad Mubeen, Associate Professor Moris Behnam, and Professor Thomas Nolte.

## 3.3 Research Process and Methodology

The scientific method can be interpreted as a concrete set of rules that define how to proceed in posing new relevant questions and how to formulate successful research problems, as stated by Dodig-Crnkovic in [114]. To structure this process, a generic framework can be defined as depicted in Fig. 3.2. This process portrays a framework for most research in the domain of computer science and consists of four main parts [115, 116].

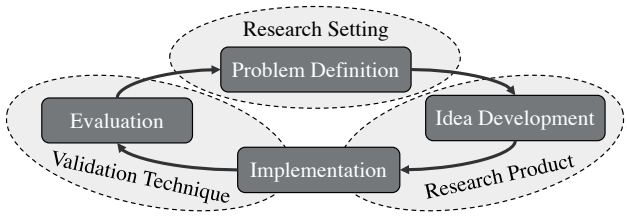



Figure 3.2: Research Process in Computer Science.

1) **Problem Definition:** This phase concerns itself with the problem formulation and can include steps like a state-of-the-art study or a systematic survey of the scientific field.

2) **Idea Development:** This phase focuses on the way the idea, to answer the stipulated problem, is developed.

3) **Implementation:** The implementation phase is used to transfer the theoretical approach developed during the idea development into an analyzable artefact (i.e., simulation, implementation on real hardware, implementation of analysis techniques, etc.).

4) **Evaluation:** In the evaluation phase data is collected to evaluate the proposed solution, draw conclusions, and identify limitations of the approach.

In [116] Shaw identifies commonly applied *research settings*, *research products*, and *validation techniques* that are used in the field of computer science. These groups can be mapped to the phases of the research process such that the research setting describes the problem class and the type of research question posed. The research product illustrates the means to how the research

is approached, including the possible solution types. Finally, the research technique describes the means of validating the software and maps to the remaining two steps in the research process (see Fig. 3.2).

Shaw writes that a successful research strategy "*identifies a research setting that addresses a fundamental issue in the problem and a matching approach and validation technique*" [116]. Hence, it is crucial to select a combination of research setting, research product, and research technique that are well suited to be applied together.

The identified research challenges presented in Section 3.1 can be classified in two types of research settings. RC1 is of type *Characterization* as it targets the identification of challenges when combining a specific type of software system with a specific type of hardware architecture. The remaining research challenges RC2 to RC5 are of type *Method/Means*. Here it is important to design, or improve existing methods for certain systems.

The research products used to address the research challenges are threefold. RC1 utilizes *Qualitative or Descriptive Model* as the main focus lies on the identification of challenges. The system is carefully analyzed and the problem area is structured. Observations are reported and examples are provided. RC2 and RC3 focus on the development of a new execution framework for the many-core processor, and thus can be categorized as *Technique*. RC4 and RC5 target the timing analysis of automotive applications and fall in the category of *Analytic Model*, as models are developed that allow to formally analyze the desired timing properties of the system.

Also the selected validation techniques vary between the different research challenges. RC1 can be categorized as *Persuation*, as the system and its implications are analyzed based on reasoning upon which conclusions are drawn. RC2 to RC5 are *Evaluation/Quantitative Model* as the evaluation focuses on comparison of many samples that are generated in a random way to represent a representative range of applications, or by performing several measurements on real hardware to draw conclusions. RC4 further falls in the category *Analysis*, since the proposed approach is proven to be correct in a rigorous way.

# Chapter 4

# Conclusions and Future Work

## 4.1 Summary and Conclusion

In this thesis we introduce methods that facilitate the consolidation of automotive real-time applications on clustered many-core platforms. Our approach can be divided into two main parts. First, the transformation of complex timing requirements into scheduling constraints, including their timing analysis. Second, an execution framework that reduces contention on shared resources, while providing isolated execution spaces for applications that are mapped to the platform.

In the automotive domain, timing constraints on the propagation of data through the system are crucial to ensure predictable behavior of the system. Such timing constraints are, however, cumbersome to verify at the level of software architecture, or to integrate in scheduling decisions, as tasks can be activated at different periods and the scheduler typically focuses on the task's local deadlines. Our proposed timing analysis is applicable at various levels of system information, and can already give upper bounds on these delays purely based on the application model. To ease the scheduling of the application, we propose methods that augment an application with job-level dependencies such that specified data propagation delay constraints are met as long as these precedence constraints between task's jobs are met.

An application that is augmented with job-level dependencies is then

scheduled on the many-core platform, where we propose a one-to-one mapping between applications and clusters. This effectively utilizes the execution islands that are a result of the many-cores' intrinsic hardware architecture. Within one cluster an execution framework is proposed that removes contention to cluster local resources by privatization and sharing of dedicated memory banks between compute cores. A time-triggered schedule is used to orchestrate the access to these shared resources, as well as to external memory, in a way that no two tasks access shared resources at a time. Two versions of the framework are presented. The Contention-Free Execution Framework (CEF) introduces the basic pillars of the framework and uses an optimal Integer Linear Programming (ILP) formulation, and a memory-centric heuristic to generate the job mapping and the time-triggered schedule. To better utilize the potential of cluster local memory, and to reduce the required traffic to external memory, the CEF is extended to the Memory Aware Contention-Free Execution Framework (MCEF). This framework makes it possible to assign tasks statically to compute cores. Such static tasks can then allocate their entire memory within the cores memory bank, effectively reducing the access to external memory, as all required code and data is hosted within the cluster for these tasks. Hence, there exists a tradeoff. Static tasks experience reduced memory copy times, whereas dynamic tasks are not confined to execute all their jobs on the same core. A Constraint Programming (CP) formulation is presented that assigns tasks to the two classes, dynamic or static, and further generates the mapping and time-triggered schedule in a way that job-level dependencies, and thus data propagation delay constraints, are considered. An integral part of the execution on the many-core is access to cluster-external memory which is performed over the NoC. We propose a symmetric NoC organization that effectively divides the complete NoC into partitions, where two partitions do not share NoC resources, hence, they cannot affect each other. It is shown how to configure the traffic injection in a way that message transmission times are minimized, while buffers on the NoC router do not overflow. A tight timing analysis for the partitioned scenario is provided. This NoC partitioning reduces the contention on the NoC and leads to faster messages between the cluster and external memory, which in turn reduces the overall latency of accesses to external memory.

In summary, we have presented methods that transform complex timing constraints on the data propagation to constraint types that ease the scheduling problem. An execution framework for the cluster of the many-core is proposed that allows access to cluster external memory while it avoids contention on shared resources by design. The NoC partitioning and configuration provides

isolation between the different applications and reduces the access time from the clusters to external memory. Moreover, methods that facilitate the verification of data propagation delays in each development step are provided.

## 4.2 Future Work

Several research directions remain for the future work:

- The developed timing analysis for data propagation delays [101, 104, 108] is applicable to synchronized systems only. I.e. all tasks must operate on the same timebase. Traditional automotive systems often distribute functionality across different ECUs that are connected by real-time bus systems, such as the Control Area Network [117]. In this setting, different compute nodes are not synchronized. Thus, an extension of the proposed timing analysis to incorporate communication between unsynchronized nodes would further strengthen the applicability of the timing analysis to industrial requirements.

- The main focus of the NoC partitioning proposed in [107] is to facilitate the analysis of NoC traffic between clusters and external memory, taking architectural features of the NoC into account. This restricts the communication between different clusters to be performed only over the external memory. Allowing communication between clusters directly can improve the communication delays, however it may also introduce additional interference that needs to be considered in the timing analysis. Future work will investigate different methods to facilitate inter-application communication on the platform.

- Time-triggered scheduling requires to store the complete schedule within the cluster local memory. Typically, memory is a scarce resource within the cluster, while the number of jobs of industrially sized applications, that need to be scheduled within one hyperperiod, is large [66]. One approach to tackle this problem is to generate the time-triggered schedule in a way that techniques such as the offline-equivalence methods [118] can be used. Here, online scheduling is used, and only deviations from the online schedule are required to be stored in an offline table. On the other hand, online scheduling techniques can be explored [111, 113], and extended to incorporate the specified job-level dependencies into scheduling and schedulability tests. The schedule size can also be reduced by grouping of tasks to larger entities, where only the release of

groups needs to be saved in the schedule. Sequencing of tasks within the groups further allows to remove constraints for the schedule generation. Such a grouping can have several possible objectives, as it may affect the accumulated time to access data from different groups, or the number of job-level dependencies that remain for the schedule generation. On the other hand, it is known that larger memory access phases negatively impact the schedulability of the system [66, 112, 113].

- To this point, only periodic real-time tasks are considered. Automotive real-time applications may also include to other task types. Such as sporadic tasks that are triggered only at the arrival of a certain stimuli event, or rate-dependent tasks that are dependent on the rotational speed of the crankshaft and may also adjust their WCET depending on the engine speed. Extending the execution framework on the clusters to incorporate these task types, thus, widens the supported application range.

- Many of the software applications that are found in automotive systems are not subject to stringent timing requirements. Infotainment applications, or many components that are required for autonomous driving are rather subject to quality-of-service guarantees. Static configurations of the platforms become more and more challenging, as cars become connected to the internet which allows for continuous updates [119, 120]. AUTOSAR's adaptive platform for example provides linking between provided services and application clients during runtime [121]. Dedicated cluster on the many-core platform could be used to host these dynamic systems. Investigation on the interconnection and access to shared off-cluster resources is thus relevant future work.

# Bibliography


[1] Jim Turley. *Essential Guide to Semiconductors, the*. Prentice Hall Press, Upper Saddle River, NJ, USA, first edition, 2002.

[2] Giorgio Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Real-Time Systems Series. Springer US, 2011.

[3] Global Market Insights. Embedded system market size by application, by product. Industry outlook report, regional analysis, application development potential, price trends, competitive market share & forecast, 2016 to 2023. Technical Report GMI117, 2016.

[4] Stefan Rathgeber. AUTOSAR - A Standard in the course of time. In *EUROFORUM Automotive Software Development*, 2016.

[5] Alexander Pretschner, Manfred Broy, Ingolf H. Kruger, and Thomas Stauner. Software engineering for automotive systems: A roadmap. In *Proceedings of the Future of Software Engineering (FOSE)*, pages 55–71, 2007.

[6] Roland Berger Strategy Consultants. Need for consolidation in vehicle electronics, 2015.
Available at http://www.greencarcongress.com/2015/07/20150729-berger.html, last access September 2017.

[7] Freescale. Future advances in body electronics, 2013.
Available at http://cache.freescale.com/files/automotive/doc/white_paper/BODYDELECTRWP.pdf, last access September 2017.

[8] Marco Di Natale and Alberto Luigi Sangiovanni-Vincentelli. Moving from federated to integrated architectures in automotive: The role of

standards, methods and tools. *Proceedings of the IEEE*, 98(4):603–620, April 2010.

[9] Dominik Reinhardt and Markus Kucera. Domain controlled architecture - a new approach for large scale software integrated automotove systems. In *Proceedings of the 3rd International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS)*, pages 221–226, 2013.

[10] Dominik Reinhardt and Gary Morgan. An embedded hypervisor for safety-relevant automotive e/e-systems. In *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 189–198, 2014.

[11] Wolfgang Stolz, Robert Kornhaas, Ralph Krause, and Tino Sommer. Domain control units - the solution for future e/e architectures? In *SAE Technical Paper*. SAE International, April 2010.

[12] Simon Kramer, Dirk Ziegenbein, and Arne Hamann. Real world automotive benchmarks for free. In *Proceedings of the 6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2015.

[13] Nico Feiertag, Kai Richter, Johan Norlander, and Jan Jonsson. A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics. In *Proceedings of the 1st International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS)*, 2008.

[14] Anton Cervin, Dan Henriksson, Bo Lincoln, Johan Eker, and Karl-Erik Årzén. How does control timing affect performance? analysis and simulation of timing using jitterbug and truetime. *IEEE Control Systems*, 23(3):16–30, 2003.

[15] Anton Cervin. Stability and worst-case performance analysis of sampled-data control systems with input and output jitter. In *American Control Conference (ACC)*, pages 3760–3765, 2012.

[16] Paolo Gai and Massimo Violante. Automotive embedded software architecture in the multi-core age. In *Proceedings of the 21th IEEE European Test Symposium (ETS)*, pages 1–8, 2016.

[17] Selma Saidi, Rolf Ernst, Sascha Uhrig, Henrik Theiling, and Benoît Dupont de Dinechin. The shift to multicores in real-time and safety-critical systems. In *Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis (CODES-ISSS)*, pages 220–229, 2015.

[18] Kalray Inc. MPPA processors for autonomous driving. Technical report, 2016.

[19] Paolo Burgio, Marko Bertogna, Ignacio Sanudo Olmedo, Paolo Gai, Andrea Marongiu, and Michal Sojka. A software stack for next-generation automotive systems on many-core heterogeneous platforms. In *Proceedings of the 19th Euromicro Conference on Digital System Design (DSD)*, pages 55–59, 2016.

[20] Benoît Dupont de Dinechin, Duco van Amstel, Marc Poulhis, and Guillaume Lager. Time-critical computing on a single-chip massively parallel processor. In *Proceedings of the 17th Conference on Design, Automation & Test in Europe (DATE)*, pages 1–6, 2014.

[21] Luca Benini and Giovanni De Micheli. Networks on chips: a new SoC paradigm. *Computer*, 35(1):70–78, Jan 2002.

[22] Quentin Perret. *Predictable execution on many-core processors*. PhD thesis, University of Toulouse, 2017.

[23] Quentin Perret, Pascal Maurère, Éric Noulard, Claire Pagetti, Pascal Sainrat, and Benoit Triquet. Predictable composition of memory accesses on many-core processors. In *Proceedings of the 8th European Congress on Embedded Real Time Software and Systems (ERTS2)*, 2016.

[24] Quentin Perret, Pascal Maurère, Éric Noulard, Claire Pagetti, Pascal Sainrat, and Benoit Triquet. Temporal isolation of hard real-time applications on many-core processors. In *Proceedings of the 22nd IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 1–11, 2016.

[25] Quentin Perret, Pascal Maurère, Éric Noulard, Claire Pagetti, Pascal Sainrat, and Benoit Triquet. Mapping hard real-time applications on many-core processors. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems (RTNS)*, 2016.

[26] Michael Barr and Anthony Massa. *Programming Embedded Systems: With C and GNU Development Tools*. O'Reilly Media, 2nd edition, 2009.

[27] Jim Turley. The two percent solution. Technical Report Embedded Systems Design, TechInsights, 2002.

[28] John A. Stankovic and K. Ramamritham, editors. *Tutorial: Hard Real-time Systems*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1989.

[29] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat, and Per Stenström. The worst-case execution-time problem – overview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(3):36:1–36:53, May 2008.

[30] Lui Sha, Tarek Abdelzaher, Karl-Erik Årzén, Anton Cervin, Theodore Baker, Alan Burns, Giorgio Buttazzo, Marco Caccamo, John Lehoczky, and Aloysius K. Mok. Real time scheduling theory: A historical perspective. *Real-Time Systems*, 28(2):101–155, 2004.

[31] Robert I. Davis and Alan Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Computing Surveys (CSUR)*, 43(4):35:1–35:44, October 2011.

[32] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, January 1973.

[33] Mathai Joseph and Paritosh K. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, 1986.

[34] Enrico Bini, Giorgio C. Buttazzo, and Giuseppe M. Buttazzo. Rate monotonic analysis: the hyperbolic bound. *IEEE Transactions on Computers*, 52(7):933–942, Jul 2003.

[35] Christopher B. Watkins and Randy Walter. Transitioning from federated avionics architectures to integrated modular avionics. In *Proceedings of the 26th IEEE/AIAA Digital Avionics Systems Conference (DASC)*, pages 2.A.1–1–2.A.1–10, 2007.

[36] Thomas Gaska, Chris Watkin, and Yu Chen. Integrated modular avionics - past, present, and future. *IEEE Aerospace and Electronic Systems Magazine*, 30(9):12–23, 2015.

[37] Vijay P. Bhatkar and Dobrivoje Popovic. *Distributed Computer Control for Industrial Automation*. Marcel Dekker, Inc., New York, NY, USA, 1990.

[38] Peter Neumann. Communication in industrial automation – what is going on? *Control Engineering Practice*, 15(11):1332 – 1347, 2007. Special Issue on Manufacturing Plant Control: Challenges and Issues.

[39] Herman Kopetz. From a federated to an integrated architecture for dependable embedded systems. In *Proceedings of the 8th Annual High Performance Embedded Computing Workshop (HPEC)*, 2004.

[40] Herman Kopetz. An integrated architecture for dependable embedded systems. In *Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems (SRDS)*, pages 160–161, 2004.

[41] Samarjit Chakraborty, Martin Lukasiewycz, Christian Buckl, Suhaib Fahmy, Naehyuck Chang, Sangyoung Park, Younghyun Kim, Patrick Leteinturier, and Hans Adlkofer. Embedded systems and software challenges in electric vehicles. In *Proceedings of the 15th Conference on Design, Automation & Test in Europe (DATE)*, pages 424–429, 2012.

[42] William J. Dally and Brian Towles. Route packets, not wires: on-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference (DAC)*, pages 684–689, 2001.

[43] Pierre Guerrier and Alain Greiner. A generic architecture for on-chip packet-switched interconnections. In *Proceedings of the Conference on Design, Automation & Test in Europe (DATE)*, pages 250–256, 2000.

[44] Lionel M. Ni and Philip K. McKinley. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2):62–76, Feb 1993.

[45] Thomas Ferrandiz, Fabrice Frances, and Christian Fraboul. A method of computation for worst-case delay analysis on spacewire networks. In *Proceedings of the 4th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 19–27, 2009.

[46] Dakshina Dasari, Borislav Nikolić, Vincent Nélis, and Stefan M. Petters. NoC contention analysis using a branch-and-prune algorithm. *ACM Transactions Embedded Computing Systems (TECS)*, 13(3s):113:1–113:26, March 2014.

[47] Zheng Shi and Alan Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *Proceedings of the 2nd ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, pages 161–170, 2008.

[48] Borislav Nikolić and Stefan M. Petters. EDF as an arbitration policy for wormhole-switched priority-preemptive nocs: Myth or fact? In *Proceedings of the 14th International Conference on Embedded Software (EMSOFT)*, pages 28:1–28:10, 2014.

[49] David Wentzlaff, Patrick Griffin, Henry Hoffmann, Liewei Bao, Bruce Edwards, Carl Ramey, Matthew Mattina, Chyi-Chang Miao, John F. Brown III, and Anant Agarwal. On-chip interconnection architecture of the tile processor. *IEEE Micro*, 27(5):15–31, September 2007.

[50] Meng Liu, Matthias Becker, Moris Behnam, and Thomas Nolte. Buffer-aware analysis for worst-case traversal time of real-time traffic over RRA-based NoCs. In *Proceedings of the 25th Euromicro Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 567–575, 2017.

[51] Qin Xiong, Zhonghai Lu, Fei Wu, and Changsheng Xie. Real-time analysis for wormhole noc: Revisited and revised. In *Proceedings of the 26th Edition on Great Lakes Symposium on VLSI (GLSVLSI)*, pages 75–80, 2016.

[52] Leandro Soares Indrusiak, Alan Burns, and Borislav Nikolić. Analysis of buffering effects on hard real-time priority-preemptive wormhole networks. *Computer Research Repository (CoRR)*, abs/1606.02942, 2016.

[53] Qin Xiong, Fei Wu, Zhonghai Lu, and Changsheng Xie. Extending real-time analysis for wormhole NoCs. *IEEE Transactions on Computers (TC)*, PP(99):1–1, 2017.

[54] *Mellanox Technologies Ltd. TILEPro64 processor product brief*, 2015. Available at http://www.mellanox.com/related-docs/prod_multi_core/PB_TILE-Gx72.pdf, last access October 2017.

[55] *PEZY Computing Ltd. PEZY-SC*, 2014.
Available at http://pezy.co.jp/en/products/pezy-sc.html, last access October 2017.

[56] Adapteva Inc., Adapteva Inc. 1666 Massachusetts Ave, Suite 14 Lexington, MA 02420 USA. *Epiphany Architecture Reference*, 2012.

[57] David Kanter and Linley Gwennap. Kalray clusters calculate quickly. In *Linley Group - Microprocessor Report*, 2015.
Available at http://www.linleygroup.com/mpr/article.php?id=11353, last access October 2017.

[58] *Intel Corp. The SCC platform overview, revision 0.80*, 2012.
Available at https://communities.intel.com/docs/DOC-5512, last access October 2017.

[59] Max Baron. The single-chip cloud computer. In *Linley Group - Microprocessor Report*, 2010.
Available at https://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/intel-labs-single-chip-cloud-article.pdf, last access October 2017.

[60] Daksina Dasari, Benny Åkesson, Vincent Nélis, Muhammad Ali Awan, and Stefan M. Petters. Identifying the sources of unpredictability in cots-based multicore systems. In *Procedings of the 8th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 39–48, 2013.

[61] Reinhard Wilhelm, Daniel Grund, Jan Reineke, Marc Schlickling, Markus Pister, and Christian Ferdinand. Memory hierarchies, pipelines, and buses for future architectures in time-critical embedded systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(7):966–978, 2009.

[62] Jan Reineke, Daniel Grund, Christoph Berg, and Reinhard Wilhelm. Timing predictability of cache replacement policies. *Real-Time Systems*, 37(2):99–122, November 2007.

[63] Benoît Dupont de Dinechin, Yves Durand, Duco van Amstel, and Alexandre Ghiti. Guaranteed services of the noc of a manycore processor. In *Proceedings of the 7th International Workshop on Network on Chip Architectures (NoCArc)*, pages 11–16, 2014.

[64] Benoît Dupont de Dinechin and Amaury Graillat. Network-on-chip service guarantees on the kalray mppa-256 bostan processor. In *Proceedings of the 2nd International Workshop on Advanced Interconnect Solutions and Technologies for Emerging Computing Systems (AISTECS)*, 2017.

[65] Hamdi Ayed, Jerome Ermont, Jean-Luc Scharbarg, and Christian Fraboul. Towards a unified approach for worst-case analysis of Tilera-like and Kalray-like NoC architectures. In *Proceedings of the 12th IEEE World Conference on Factory Communication Systems (WFCS)*, pages 1–4, 2016.

[66] Matthias Becker, Dakshina Dasari, Borislav Nicolić, Benny Åkesson, Vincent Nélis, and Thomas Nolte. Contention-free execution of automotive applications on a clustered many-core platform. In *Proceedings of the 28th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 14–24, 2016.

[67] Reinhard Wilhelm and Jan Reineke. Embedded systems: Many cores – many problems. In *Proceedings of the 7th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 176–180, 2012.

[68] Vincent Nélis, Patrick Meumeu Yomsi, Luís Miguel Pinho, José Carlos Fonseca, Marko Bertogna, Eduardo Quiñones, Roberto Vargas, and Andrea Marongiu. The Challenge of Time-Predictability in Modern Many-Core Architectures. In *14th International Workshop on Worst-Case Execution Time Analysis*, volume 39 of *OpenAccess Series in Informatics (OASIcs)*, pages 63–72, 2014.

[69] Andreas Tretter, Georgia Giannopoulou, Matthias Baer, and Lothar Thiele. Minimising access conflicts on shared multi-bank memory. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(5s):135, 2017.

[70] Denis Claraz, Stefan Kuntz, Ulrich Margull, Michael Niemetz, and Gerhard Wirrer. Deterministic execution sequence in component based multi-contributor powertrain control systems. In *Proceedings of the 5th European Congress on Embedded Real Time Software and Systems (ERTS2)*, pages 1–7, 2012.

[71] Patrick Frey. Ulmer Informatik Berichte Nr 2010-03 - Case Study: Engine Control Application. Technical report, University Ulm, 2010.

[72] EAST-ADL - Domain Model Specification, V2.1.12, 2014.
Available at http://www.east-adl.info/Specification/V2.1.12/EAST-ADL-Specification_V2.1.12.pdf, last access September 2017.

[73] *AUTOSAR*.
Available at www.autosar.org, last access September 2017.

[74] OSEK/VDX operating system. Technical report, OSEK group, 2005.

[75] Kai Hanninen, Jukka Maki-Turja, Mikael Nolin, Mats Lindberg, John Lundback, and Kurt-Lennart Lundback. The rubus component model for resource constrained real-time systems. In *Proceedings of the 3rd International Symposium on Industrial Embedded Systems (SIES)*, pages 177–183, 2008.

[76] *TIMMO-2-USE*, 2012.
Available at https://itea3.org/project/timmo-2-use.html, last access September 2017.

[77] Timing augmented description language (TADL) syntax, semantics, metamodel. Ver. 2, Deliverable 11. Technical report, 2012.

[78] ISO/DIS 26262-1 - Road Vehicles – Functional safety, 2009.

[79] International Electronical Comission (IEC): 44:2006 Analysis Techniques for System Reliability – Procedure for Failure Mode and Effects Analysis (FMEA), 2006.

[80] Rolf Ernst and Marco Di Natale. Mixed criticality systems – a history of misconceptions? *IEEE Design & Test*, 33(5):65–74, 2016.

[81] AUTOSAR - Specification of Timing Extensions, Nr. 4.3.0, 2016.

[82] Tobias Klaus, Florian Franzmann, Maximilian Gaukler, Andreas Michalka, and Peter Ulbrich. Closing the loop: towards control-aware design of adaptive real-time systems. In *Proceedings of the 37th IEEE Real-Time Systems Symposium (RTSS)*, pages 363–363, 2016.

[83] A. C. Rajeev, Swarup Mohalik, Manoj G. Dixit, Devesh B. Chokshi, and S. Ramesh. Schedulability and end-to-end latency in distributed ecu networks: Formal modeling and precise estimation. In *Proceedings of the 10th ACM SIGBED International Conference on Embedded Software (EMSOFT)*, pages 129–138, 2010.

[84] Saad Mubeen, Jukka Mäki-Turja, and Mikael Sjödin. 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 (ComSIS)*, 10(1), 2013.

[85] Arcticus Systems. Rubus ICE.
Available at https://www.arcticus-systems.com/products/, last access May 2017.

[86] Rafik Henia, Arne Hamann, Marek Jersak, Razvan Racu, Kai Richter, and Rolf Ernst. System level performance analysis - the symta/s approach. *IEEE Proceedings - Computers and Digital Techniques*, 152(2):148–166, 2005.

[87] Symtavision GmbH. SymTA/S and TraceAnalyzer for ECUs.
Available at https://www.symtavision.com/products/ecu-timing/, last access May 2017.

[88] Timing Architects. Timing Architects Inspector.
Available at https://www.timing-architects.com/ta-tool-suite/inspector/, last access May 2017.

[89] The economic impacts of inadequate infrastructure for software testing (planning report 02-3). Gaithersburg, MD: National Institute of Standards and Technology., 2002.

[90] C.C. Michael, Ken van Wyk, and Will. Radosevich. Risk-based and functional security testing, 2005.
Available at: https://www.us-cert.gov/bsi/articles/best-practices/security-testing/risk-based-and-functional-security-testing, last access May 2017.

[91] Saad Mubeen, Mikael Sjödin, Thomas Nolte, John Lundbäck, Mattias Gålnander, and Kurt-Lennart Lundbäck. End-to-end timing analysis of black-box models in legacy vehicular distributed embedded systems. In *Proceedings of the 21st IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 149–158, 2015.

[92] Alessio Bucaioni, Antonio Cicchetti, Federico Ciccozzi, Romina Eramo, Saad Mubeen, and Mikael Sjödin. Anticipating implementation-level timing analysis for driving design-level decisions in east-adl. In

*Proceedings of the International Workshop on Modelling in Automotive Software Engineering (MASE)*, 2015.

[93] Saad Mubeen, Thomas Nolte, Mikael Sjödin, John Lundbäck, and Kurt-Lennart Lundbäck. Supporting timing analysis of vehicular embedded systems through the refinement of timing constraints. *Software & Systems Modeling*, pages 1–31, 2017.

[94] Richard Gerber, Seongsoo Hong, and Manas Saksena. Guaranteeing end-to-end timing constraints by calibrating intermediate processes. In *Proceedings of the 15th Real-Time Systems Symposium (RTSS)*, pages 192–203, 1994.

[95] Manas Saksena and Seongsoo Hong. Resource conscious design of distributed real-time systems: An end-to-end approach. In *Proceedings of the 2nd IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 306–313, 1996.

[96] Marco Di Natale, Wei Zheng, Claudio Pinello, Paolo Giusto, and Alberto Sangiovanni Vincentelli. Optimizing end-to-end latencies by adaptation of the activation events in distributed automotive systems. In *Proceedings of the 13th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS)*, pages 293–302, 2007.

[97] Jos Carlos Palencia Gutierrez, J. Javier Gutierrez Garcia, and Michael Gonzalez Harbour. On the schedulability analysis for distributed hard real-time systems. In *Proceedings of the 9th Euromicro Workshop on Real Time Systems*, pages 136–143, 1997.

[98] Qi Zhu, Yang Yang, Marco Di Natale, Eelco Scholte, and Alberto Sangiovanni-Vincentelli. Optimizing the software architecture for extensibility in hard real-time distributed systems. *IEEE Transactions on Industrial Informatics*, 6(4):621–636, 2010.

[99] Wei Zheng, Qi Zhu, Marco Di Natale, and Alberto Sangiovanni Vincentelli. Definition of task allocation and priority assignment in hard real-time distributed systems. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS)*, pages 161–170, 2007.

[100] Mohammad Ashjaei, Nima Khalilzad, Saad Mubeen, Moris Behnam, Ingo Sander, Luis Almeida, and Thomas Nolte. Designing end-to-end resource reservations in predictable distributed embedded systems. *Real-Time Systems*, 53(6):916–956, Nov 2017.

[101] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. Synthesizing job-level dependencies for automotive multi-rate effect chains. In *Proceedings of the 22nd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 159–169, 2016.

[102] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. MECHAniSer - a timing analysis and synthesis tool for multi-rate effect chains with job-level dependencies. In *Proceedings of the 7th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2016.

[103] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. Timing analysis and synthesis of mixed multi-rate effect chains in mechaniser. In *Proceedings of the Open Demo Session of Real-Time Systems located at the 37th IEEE Real Time Systems Symposium (RTSS@Work)*, 2016.

[104] Matthias Becker, Saad Mubeen, Dakshina Dasari, Moris Behnam, and Thomas Nolte. A generic framework facilitating early analysis of data propagation delays in multi-rate systems. In *Proceedings of the 23th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–11, 2017.

[105] Matthias Becker, Dakshina Dasari, Vincent Nélis, Moris Behnam, Luís Miguel Pinho, and Thomas Nolte. Investigation on autosar-compliant solutions for many-core architectures. In *Proceedings of the 18th Euromicro Conference on Digital System Design (DSD)*, pages 95–103, 2015.

[106] Niko Böhm, Daniel Lohmann, and Wolfgang Schröder-Preikschat. A comparison of pragmatic multi-core adaptions of the AUTOSAR system. In *Proceedings of the 7th Workshop on Operating Systems Platforms for Embedded Real-Time applications (OSPERT)*, 2011.

[107] Matthias Becker, Borislav Nikolić, Dakshina Dasari, Benny Åkesson, Vincent Nélis, Moris Behnam, and Thomas Nolte. Partitioning and analysis of the network-on-chip on a cots many-core platform. In *Proceedings of the 23rd IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 101–112, 2017.

[108] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. End-to-end timing analysis of cause-effect chains in automotive embedded systems. *Journal of Systems Architecture*, 80(Supplement C):104 – 113, 2017.

[109] Jonas Holmberg. Offline scheduling of task sets with complex end-to-end delay constraints. Master's thesis, Mälardalen University, Sweden, 2017.

[110] Matthias Becker, Saad Mubeen, Dakshina Dasari, Moris Behnam, and Thomas Nolte. Scheduling multi-rate real-time applications on clustered many-core architectures with memory constraints. In *Proceedings of the 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018.

[111] Ahmed Alhammad and Rodolfo Pellizzoni. Schedulability analysis of global memory-predictable scheduling. In *Proceedings of the 14th International Conference on Embedded Software (EMSOFT)*, pages 20:1–20:10, 2014.

[112] Cláudio Maia, Luis Miguel Nogueira, Luís Miguel Pinho, and Daniel Gracia Prez. A closer look into the aer model. In *Proceedings of the 21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2016.

[113] Cláudio Maia, Geoffrey Nelissen, Luis Miguel Nogueira, Luís Miguel Pinho, and Daniel Gracia Pérez. Schedulability analysis for global fixed-priority scheduling of the 3-phase task model. In *Proceedings of the 23rd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2017.

[114] Gordana Dodig-Crnkovic. Scientific methods in computer science. In *Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden (Promote IT)*, 2002.

[115] Hilary J. Holz, Anne Applin, Bruria Haberman, Donald Joyce, Helen Purchase, and Catherine Reed. Research methods in computing: What are they, and how should we teach them? *ACM SIGCSE Bulletin*, 38(4):96–114, June 2006.

[116] Mary Shaw. The coming-of-age of software architecture research. In *Proceedings of the 23rd International Conference on Software Engineering (ICSE)*, pages 656–, 2001.

[117] ISO 11898-1. Road Vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication, ISO Standard-11898, Nov. 1993.

[118] Mitra Nasri and Björn B. Brandenburg. Offline equivalence: A non-preemptive scheduling technique for resource-constrained embedded real-time systems (outstanding paper). In *Proceedings of the 23rd IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 75–86, 2017.

[119] Helena Holmström Olsson, Hiva Alahyari, and Jan Bosch. Climbing the "stairway to heaven"–a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In *Profeedings of the 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, pages 392–399, 2012.

[120] Sebastian Vöst. Vehicle level continuous integration in the automotive industry. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, pages 1026–1029, 2015.

[121] AUTOSAR - Adaptive Platform, 2017.
Available at: https://www.autosar.org/standards/adaptive-platform/, last access September 2017.