Concepts for improved availability and computational power in automated driving

S. Grubmüller, G. Stettinger, D. Nešić, D. Watzenig

Automated vehicles are required to operate on highways and in complex urban scenarios. To safely handle these complex environmental influences, sophisticated automated driving functions demand a high availability of all involved components in combination with increased computational power. Particular multi-core platforms are deployed to cope with these demands. To achieve higher system availability for SAE level 3 and higher, fail operational concepts from system level down to Microcontroller Unit (MCU) level are needed. These concepts include hardware as well as software requirements and are discussed in this paper. For an increased computing performance, the idea and further the model of a parallel computation method for driving functions and their control algorithms is introduced. For that a stabilizing controller is implemented on different cores of the multi-core processor. Finally, this resulting closed-loop system is modeled as a hybrid system which will serve as an input for further stability analysis.

Keywords: fail-operational; parallel computation; waveform relaxation; multi-core processor

Konzepte für höhere Verfügbarkeit und Rechenleistung im Bereich autonomes Fahren.

Autonome Fahrzeuge sollen in Zukunft auf Autobahnen sowie in komplexen urbanen Situationen funktionieren. Damit solche hochentwickelten autonomen Fahrsysteme diese komplexen Umwelteinflüsse sicher verarbeiten können, bedarf es einer hohen Verfügbarkeit dieser Systeme sowie einer hohen Prozessorleistung. Dieser Bedarf wird durch den Einsatz von Multi-Core-Prozessoren bewältigt. Höhere Verfügbarkeit von Fahrfunktion mit SAE Level 3 wird durch fehlertolerante Konzepte auf System- bis hin zur Mikrocontroller-Ebene erreicht. Diese Konzepte bestehen aus Hardware- und Software-Anforderungen. Zur Steigerung der Rechenperformance werden hier die Idee und ein Modell für parallele Rechenmethoden für Fahrfunktionen und Regler vorgestellt. Dafür wird ein stabilisierender Regler auf verschiedene Prozessorkerne des Multi-Core-Prozessors implementiert. Schließlich wird die resultierende geschlossene Regelschleife als hybrides System modelliert und kann für weitere Stabilitätsanalysen verwendet werden.

Schlüsselwörter: Fehlertolerant; paralleles Rechnen; Multi-Core-Prozessor

Received March 19, 2018, accepted June 5, 2018, published online July 4, 2018
© The Author(s) 2018

1. Introduction

Automated driving will significantly influence our future mobility to make it cleaner, more comfortable and safer. According to [6], it is assumed that vehicles are fully automated by 2050. To achieve a full automation of vehicles, the authors in [19] identified a number of research challenges, two of them are separately discussed in this paper: 1) availability, reliability and robustness and 2) constraint computing power.

In terms of end-user acceptance and passenger safety, a high availability [2], reliability and robustness [19] of (highly) automated systems is mandatory even under adverse environmental conditions. To achieve these requirements, failures need to be detected entirely as well as properly handled and a redundancy of all involved components has to be provided. This leads to a fail-operational behavior, where the automated driving function operates even when faults occur [9]. At the moment, fail-operational concepts exist for X-by-wire systems (e.g. [15]) but not for automated driving functions and their active safety systems, which we discuss here.

For a successful implementation of automated vehicles three building blocks are mandatory [19]: i) Sensing, ii) processing and decision making and iii) vehicle control. Based on this functional architecture, an autonomous driving function represents a control system. A classical control system consists of sensors, actuators, a controller and a plant. One example is the cruise control, where the vehicle represents the plant, the actual speed is measured and

the difference between required and measured speed is controlled to be zero by continuously adjusting the vehicle's acceleration. In the future such control systems consist of complex algorithms, rich software component interfaces and have to process amounts of data [11]. Thus, the higher the level of automation, the more computational power is required by the control function implemented on an Electronic Control Unit (ECU). A conservative implementation approach is the one function per ECU, which leads to a decentralized architecture and a high number of ECUs [13] especially when a fail-operational behavior is required. Consequently, the vehicle's power consumption will significantly increase. To solve this problem, multi-core ECUs are considered to replace single-core platforms in automotive domains [13]. The deployment of multi-core ECUs will improve the computing performance by parallel computation of different driving functions [11]. To further increase the computing performance, a distribution of the computing workload of one control function on a number of processor cores can be implemented. This idea is inspired by software development, where one task is parti-

Grubmüller, Stephanie, Virtual Vehicle Research Center, Inffeldgasse 21a, 8010 Graz, Austria (E-mail: Stephanie.Grubmueller@v2c2.at); **Stettinger, Georg**, Virtual Vehicle Research Center, Inffeldgasse 21a, 8010 Graz, Austria; **Nešić, Dragan**, Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville, 3010 VIC, Australia; **Watzenig, Daniel**, Institute of Automation and Control, Graz University of Technology, Inffeldgasse 21b, 8010 Graz, Austria

Fig. 1. (a) Triple modular redundancy (2-out-of-3 voting); (b) Dynamic redundancy with hot standby (solid lines) and cold standby (including dashed lines) according to [9]

tioned into smaller sub-tasks and allocated to different computing resources [1, 13].

In this work dynamic control systems are considered. A decomposition of the controller state-space model and parallel computation may lead to instabilities of the closed-loop system. Thus, to address this issue an appropriate analysis method is needed which requires a model of the actual closed-loop system. In this paper we present a first idea for a modeling approach for parallel computation of controllers which is the basis for further stability analysis in the future.

First we discuss the fail-operational concepts in Sect. 2. In Sect. 3 parallel computing and the modeling framework is presented. Finally, the future work and conclusions are given.

2. Fail-operability in automated driving

This section presents fail-operational strategies on hardware and software level and shows the importance of fault-tolerant systems in automated driving, especially for active safety systems.

A system is called fail-operational, when after one internal failure it still operates with full or degraded functionality until the system enters a safe state [9]. For example, if an error is detected and the vehicle is brought to a safe standstill by the remaining active system components, then the automated vehicle shows fail-operational behavior.

Different fault-tolerant concepts for hardware, such as sensors, actuators, ECUs etc., exist and are presented in [9]. One is the static redundancy structure, where the system is implemented redundantly including a majority voter. In Fig. 1(a) is shown a triple modular redundancy, where the 2-out-of-3 voting allows one system to fail.

This hardware concept leads to a high number of systems as well as system components. Consequently the weight of the vehicle may increase.

A dynamic redundancy reduces the amount of systems, but a raise in data processing may be expected. It mainly consists of a primary and backup system which may have a degraded functionality. In case of a failure detected by the fault detection module, the primary

system transfers to a safe state and the backup system takes over control. According to [9], faults are either detected by model-based or signal-based methods.

In Fig. 1(b) two dynamic redundancies for a duplex system are presented. In solid lines a hot standby of the backup system is depicted where both systems run in parallel. If a failure in System 1 is detected, then System 2 is ready to take over without any reconfiguration. But the wear out rate is higher. A cold standby (including dashed lines) represents a backup system which is initially deactivated. In case of a failure in System 1 and after reconfiguration System 2 takes over.

For example, for a fail-operational 360° environmental awareness designed for nearly all weather conditions, the field of view of different sensors should overlap. This leads to the deployment of a static redundancy. On the other hand, for actuators such as braking systems, a dynamic redundancy is appropriate including two duplex systems consisting of two duplex systems with fault detection.

When we consider automated driving functions, the control function is implemented on an ECU. Per definition in fault-tolerant control systems, failures are detected in sensors, actuators and the plant which lead to a reconfiguration of the controller [16]. But the controller itself is not fail-operational. To change it, the ECU and the controller implementation itself need to meet a number of requirements. The authors in [10] differentiate between fail-operational concepts on ECU and on MCU level, where the MCU represents the central component of the ECU. On ECU level the following concepts are presented: (i) Triple modular redundancy. (ii) Symmetric two-out-of-two diagnosis fail safe (2oo2DFS), where the whole system is designed either as fully redundant system or as a system consisting of a primary ECU with a backup ECU. Both are fail-safe, detect their own errors and monitor the other. (iii) Asymmetric 2oo2DFS: where the architecture works as a master-slave strategy similar to the dynamic redundancy including a cold standby ECU with degraded functionality.

Depending on the performance and implementation, the MCU hosts safety related and comfort functions. According to [10], a fail-operational MCU provides fault tolerance, fault containment and fault diagnosis on register level such as the multi-core MCU presented in the paper.

But not only the hardware also the software need to be redundant. Software redundancy concepts cover fault-tolerance on bit and code level. On bit level, checksums detect errors in data (e.g. bit flips) [17]. On code level different redundancy strategies exist. One static redundancy is the repeated execution of code segments [17]. The most popular static redundancy concept is N-version programming. The algorithms are developed and implemented independently, provide the same functionality, run in parallel and the output is provided by a majority voting [8]. The recovery block concept is a dynamic redundancy structure. The safety critical code has got alternative implementations which are executed subsequently. If an error in a code is detected, the previous states will be restored and an alternative will be executed [8].

3. Parallelism of control functions

Section 2 proposes the deployment of special multi-core ECUs in order to increase the availability of driving functions. Here the deployment of multi-core ECUs is discussed to increase the computational performance and to discover new opportunities for computation especially in relation to automated driving. According to [1], parallel computing is defined as the concurrent usage of a number of processors to solve one computational problem. Therefore, the problem is decomposed in chunks of work and then allocated

Fig. 2. Processor communication architecture

to several computing resources. The processor communication architecture, the partitioning, the communication schedule between processors and finally the modeling of such problems is described in this section.

3.1 Processing structure and essentials

A parallel computing system consists of several processors (cores) which are located in small distance of each other and therefore the exchange of data between them is reliable and predictable [4]. Different processor interconnection architectures exist. One possibility is to exchange data via a shared memory. Another option is a message-passing architecture, where the network can either consist of directly connected links or of a shared bus. A combination of both architectures is called a hybrid architecture. For a first modeling approach and the sake of simplicity we assume to have a simplified architecture of synchronized processor cores as shown in Fig. 2. Each core is directly connected to a buffer which is directly connected to each buffer of the other cores. Values of inputs $\hat{u}$ and outputs $\hat{y}$ are also provided to the cores via buffers.

After the architecture has been identified, the control function can be decomposed. The decomposition method applied here is the functional decomposition where the computational workload is partitioned into less computing resource consuming sub-tasks [1].

To ensure that for every task a time interval for computation is allocated, a task scheduling on multi-core level is needed. Static scheduling policies where the execution order is determined off-line and unchanged during run-time (e.g. Round Robin) are used in real-time systems, due to their predictability. But dynamic schedulers, where the execution order is defined on-line are under research currently [13]. Important task parameters for hard real-time systems, such as control systems, are the start time and deadline of a task as well as the worst case execution time. For such systems the tasks are executed periodically [5]. The continuous time signals to and from the plant, such as the control input and the plant output, are sampled periodically and then held constant between sampling time instants [5].

3.2 Modeling and system description

For the modeling of the parallel computation of controllers we use the idea of the emulation design as in Networked Control Systems (NCS) [18] where the control system is a continuous time dynamic system. First a stabilizing controller for the plant is designed while the effects of parallel computation are ignored. The closed-loop system shown in Fig. 3(a) is defined by

$$\dot{x}_p = f_p(x_p, u), \quad y = g_p(x_p), \quad (1)$$

$$\dot{x}_c = f_c(x_c, y), \quad u = g_c(x_c), \quad (2)$$

a)

b)

Fig. 3. (a) Closed-loop control system; (b) Closed-loop control system including a parallel computed controller

where the plant state is $x_p \in \mathbb{R}^{n_p}$, the controller state is $x_c \in \mathbb{R}^{n_c}$, the control input is $u \in \mathbb{R}^{n_u}$ and the plant output is $y \in \mathbb{R}^{n_y}$, where the dimensions n_c , n_u , n_y and n_p are positive real numbers. The functions f_p , g_p , f_c and g_c are assumed to be continuous and sufficiently smooth. The time sequence t_k is defined by $t_0 < t_1 < \dots$ with $T = t_{k+1} - t_k$ and satisfies $T > 0$. The interval T is referred to as one computational interval and T_s with $T_s \geq T$ to the computing cycle. T_s is discretized in N_{stages} time instants with $N_{stages} \in \mathbb{Z}_{>0}$ and $T_s = t_{k+N_{stages}} - t_k$.

Figure 3(b) shows a closed-loop system including a controller which is distributed over several processors. The controller is decomposed into n subsystems $x_c = [x_{c1}^T, \dots, x_{cn}^T]^T$ and allocated to n different processor cores. On the i -th core the differential equations of x_i are solved sufficiently fast to be considered as a continuous time system. The variables $\hat{u}$, $\hat{y}$ and $\hat{x}_{ci}$ indicate buffer states containing the values of u , y and x_{ci} at the latest update time instants. These buffers operate as Zero-Order-Hold (ZOH) equivalents and satisfies $\dot{\hat{y}} = 0$, $\dot{\hat{u}} = 0$ and $\dot{\hat{x}_{ci}} = 0$ between update time instants. Inputs u and outputs y are periodically sampled every T_s and subsequently stored in $\hat{u}$ and $\hat{y}$; $\hat{x}_{ci}$ are the values of states exchanged with other processors at certain time instants. This leads to the following system for times $t \in [t_k, t_{k+1}]$

$$\dot{x}_p = f_p(x_p, \hat{u})$$

$$y = g_p(x_p)$$

$$\dot{x}_{c1} = f_{c1}(x_{c1}, \hat{x}_{c2}, \dots, \hat{x}_{cn}, \hat{y})$$

$$\vdots$$

$$\dot{x}_{cn} = f_{cn}(\hat{x}_{c1}, \dots, \hat{x}_{c(n-1)}, x_{cn}, \hat{y})$$

$$u = g_c(x_{c1}, \dots, x_{cn}). \quad (3)$$

To keep it simple g_c has unlimited access to x_c .

Fig. 4. (a) Parallel and (b) sequential scheduling policy

3.2.1 Task and inter-processor communication scheduling

Important for that approach is when and how processors exchange data. This partly depends on the task scheduling. Here a static task scheduling as Round Robin [18] is considered. The communication schedule policy itself is inspired by methods of parallel numerical computation of Ordinary Differential Equation (ODEs), especially by the parallelism across the system [3]. It is a way of decoupling the system's state equations to allow a parallel numerical integration [7]. Several methods were introduced over the years. Important for our work is the Waveform Relaxation (WR) [12]. The system of ODEs is decomposed into coupled subsystems and each subsystem is iteratively solved by its own numerical integration method for a given time interval. The solution of each subsystem for one iteration is called wave. After one iteration the waves are exchanged and the solver starts again until the numerical solution close enough to the unique solution [3].

Two iterative methods are distinguishable: Jacobi WR and Gauss-Seidel WR [4]. For the Jacobi WR waves generated an iteration before are used for the current integration. Therefore, values of the waves at specified time instants are exchanged and need to be stored for the whole integration time. On the other hand, for the Gauss-Seidel WR the waves of different subsystems are sequentially generated and exchanged. To implement a WR, multiple buffers for all controller states and specified exchange time instants would be necessary.

Hence, the scheduling policies used here are motivated by Jacobi WR and Gauss-Seidel WR. First we assume that after one integration the waves are close enough to the unique solutions and numerical integration effects can be ignored. Furthermore, the buffers $\hat{x}_{ci}$ contain the values of x_{ci} at the latest exchange time instant.

The resulting idea of the scheduling for a test system with controller states x_{c1} and x_{c2} is depicted in Fig. 4.

The resulting parallel computing scheme shown by Fig. 4(a) is inspired by the Jacobi WR, where x_{c1} and x_{c2} are exchanged simultaneously after every computational period T and instantaneously stored in the buffers $\hat{x}_{c1}$ and $\hat{x}_{c2}$. The processor cores are busy with computing all the time. It is assumed that read and write actions on the buffers are sufficiently fast, so that the caused delays can be ignored. After every $T_s = T$, u and y are sampled and stored in the buffers $\hat{u}$ and $\hat{y}$.

The sequential computing scheme given in Fig. 4(b) is inspired by the Gauss-Seidel WR. Within one computing cycle T_s , $\hat{x}_{c1}$ and

$\hat{x}_{c2}$ are updated sequentially. This leads to the definition of computational nodes. If the subsystems update their buffers in parallel, then they are assigned to the same computational node and $N_{stages} = 1$. Otherwise, the subsystems are assigned to different computational nodes and $N_{stages} = n$, where n is the number of subsystems updated sequentially.

Consequently, each scheduling policy discussed before can be expressed similar to the update function defined in [18]. Based on the considerations before, an impulsive system similar to that one presented in [14] can be introduced. In a follow-up paper we will describe the model and stability analysis of this approach.

4. Conclusions

Ideas to improve availability and computation performance in automated vehicles are discussed here.

Fail-operational concepts from system level to component level are presented. These concepts include software as well as hardware. Fail-operability is a main driver for user acceptance of highly automated driving functions due to the increase availability, reliability and robustness.

To increase the computational performance and decrease the number of ECUs in a vehicle, a method for parallel computation of the driving functions and its inherent modeling approach was presented here. The method and the model depend on the processors architecture, the decomposition of the workload and the scheduling of the tasks and processor communication. For a first consideration, a simplified processor and communication architecture is assumed. At this stage of the work, a static scheduling is considered, where the scheduling order relies on specific realisations of WR. The main outcome is an impulsive model of the closed-loop system including a parallel computed controller. That model can be considered in further stability analysis to find an upper bound on T which guarantees stability.

5. Future work

Currently we are working on a specification for a fail-operational active safety systems for SAE level 3 and higher. Our specific use case is an automated emergency brake assistant that works under adverse weather conditions. It includes the fail-operational field of view of the sensors, sensor fusion and decision making. The goal is to demonstrate a fail-operational behavior of the active safety system under real road conditions.

The second part of the future work involves the parallel computing of control functions. The decomposition and allocation of the controller may lead to instabilities. Therefore a stability analysis is under investigation at the moment. It will provide a maximum allowable computational interval for each task so that the closed-loop system is uniformly asymptotically stable. It is further planned to integrate idle states of the processors in the model which will also cause an extension of the stability analysis. Besides, only static scheduling is considered here. It would be beneficial to investigate the deployment of dynamic scheduling in such parallel control systems.

Acknowledgements

Open access funding provided by Graz University of Technology. This project received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No. 737469 (AutoDrive Project). This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Austria, Spain, Italy, Latvia, Belgium, Netherlands, Sweden, Finland, Lithuania, Czech Republic, Romania,

Norway. In Austria the project was also funded by the program "IKT der Zukunft" and the Austrian Federal Ministry for Transport, Innovation and Technology (bmvit). The publication was written at VIRTUAL VEHICLE Research Center in Graz and partially funded by the COMET K2 – Competence Centers for Excellent Technologies Programme of the Federal Ministry for Transport, Innovation and Technology (bmvit), the Federal Ministry for Digital, Business and Enterprise (bmdw), the Austrian Research Promotion Agency (FFG), the Province of Styria and the Styrian Business Promotion Agency (SFG).

D. Nešić work was supported under the Australian Research Council under the Discovery Project DP170104099.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Barney, B. (2017): Introduction to parallel computing. URL: https://computing.lnl.gov/tutorials/parallel_comp/.
2. Becker, J., Helmle, M., Pink, O. (2017): System architecture and safety requirements for automated driving (pp. 3–16). Cham: Springer. https://doi.org/10.1007/978-3-319-31895-0_11.
3. Ben Khaled, A., Ben Gaid, M. E. M., Pernet, N., Simon, D. (2014): Fast multi-core co-simulation of cyber-physical systems: application to internal combustion engines. *Simul. Model. Pract. Theory*, 47, 79–91. <https://doi.org/10.1016/j.simpat.2014.05.002>. <https://hal-ifp.archives-ouvertes.fr/hal-01018348>.
4. Bertsekas, D. P., Tsitsiklis, J. N. (1997): Parallel and distributed computation: numerical methods. Belmont: Athena Scientific.
5. Derler, P., Lee, E. A., Toengren, M., Tripakis, S. (2013): Cyber-physical system design contracts. In *ACM/IEEE international conference on cyber-physical systems (ICCP)* (pp. 109–118).
6. Road Transport Research Advisory Council (ERTRAC) (2017): Automated Driving Roadmap.
7. Gear, C. W. (1993): Massive parallelism across space in odes. *Appl. Numer. Math.*, 11(1), 27–43. [https://doi.org/10.1016/0168-9274\(93\)90038-S](https://doi.org/10.1016/0168-9274(93)90038-S). URL: <http://www.sciencedirect.com/science/article/pii/016892749390038S>.
8. Hecht, H. (1979): Fault-tolerant software. *IEEE Trans. Reliab.*, R–28(3), 227–232. <https://doi.org/10.1109/TR.1979.5220573>.
9. Isermann, R., Schwarz, R., Stolz, S. (2002): Fault-tolerant drive-by-wire systems. *IEEE Control Syst.*, 22(5), 64–81. <https://doi.org/10.1109/MCS.2002.1035218>.
10. Kohn, A., Schneider, R., Vilela, A., Roger, A., Dannebaum, U. (2016): Architectural concepts for fail-operational automotive systems. In *SAE 2016 world congress and exhibition*. United States: SAE International. <https://doi.org/10.4271/2016-01-0131>.
11. Leitner, A., Ochs, T., Bulwahn, L., Watzenig, D. (2017): Open dependable power computing platform for automated driving (pp. 3–16). Cham: Springer. https://doi.org/10.1007/978-3-319-31895-0_14.
12. Lelarsmee, E. (1982): The waveform relaxation method for time domain analysis of large scale integrated circuits: Theory and applications. Ph.D. thesis, EECS Department, University of California, Berkeley. URL: <http://www2.eecs.berkeley.edu/Pubs/TechRpts/1982/9614.html>.
13. Monot, A., Navet, N., Bavoux, B., Simonot-Lion, F. (2012): Multisource software on multicore automotive ecus – combining runnable sequencing with task scheduling. *IEEE Trans. Ind. Electron.*, 59(10), 3934–3942. <https://doi.org/10.1109/TIE.2012.2185913>.
14. Nešić, D., Teel, A. R. (2004): Input–output stability properties of networked control systems. *IEEE Trans. Autom. Control*, 49(10), 1650–1667. <https://doi.org/10.1109/TAC.2004.835360>.
15. Sinha, P. (2011): Architectural design and reliability analysis of a fail-operational brake-by-wire system from ISO 26262 perspectives. *Reliab. Eng. Syst. Saf.*, 96(10), 1349–1359. <https://doi.org/10.1016/j.res.2011.03.013>. <http://www.sciencedirect.com/science/article/pii/S095183201100041X>.
16. Teixeira, A. (2014): Toward cyber-secure and resilient networked control systems. Ph.D. thesis, KTH Royal Institute of Technology, School of Electrical Engineering, Department of Automatic Control, Stockholm, Sweden.
17. Ulbrich, P. M. (2014): Ganzheitliche Fehlertoleranz in eingebetteten Softwaresystemen. Ph.D. thesis, Technische Fakultät der Friedrich-Alexander-Universität Erlangen-Nürnberg.
18. Walsh, G. C., Ye, H., Bushnell, L. G. (2002): Stability analysis of networked control systems. *IEEE Trans. Control Syst. Technol.*, 10(3), 438–446. <https://doi.org/10.1109/87.998034>.
19. Watzenig, D., Horn, M. (2017): Introduction to automated driving (pp. 3–16). Cham: Springer. https://doi.org/10.1007/978-3-319-31895-0_1.

Authors

Stephanie Grubmüller

is Senior Researcher at Virtual Vehicle Research Center. She received her Master's degree in Telematics from Graz University of Technology in 2014. Her main research areas are model predictive control, automated driving, parallel computing and networked control systems in automotive systems.

Georg Stettinger

is Senior Researcher at Virtual Vehicle Research Center. His main research areas include automated driving, state estimation and filtering, system identification and FDI.

Dragan Nešić

received the B.E. degree in mechanical engineering from the University of Belgrade, Belgrade, Serbia, in 1990, and the Ph.D. degree in systems engineering from the Research School of Information Sciences, Australian National University, Canberra, ACT, Australia, in 1997.

He is currently a Professor with the Department of Electrical and Electronic Engineering,

University of Melbourne, Parkville, VIC, Australia. His research interests include networked control systems, discrete-time, sampled-data and continuous-time nonlinear control systems, input-to-state stability, extremum seeking control, applications of symbolic computation in control theory, hybrid control systems, etc.

Dr. Nešić was awarded a Humboldt Research Fellowship in 2003 by the Alexander von Humboldt Foundation, an Australian Professorial Fellowship in 2004–2009, and a Future Fellowship in 2010–2014 by the Australian Research Council. He was a Distinguished Lecturer of Control Systems Society, IEEE. He is a fellow of the Institution of Engineers Australia. He served as an Associate Editor for *Automatica*, the *IEEE Transactions on Automatic Control*, the *IEEE Transactions on Control of Networked Systems*, *Systems and Control Letters*, and the *European Journal of Control*.

Daniel Watzenig

was born in Austria. He received his Master's degree in electrical engineering and the doctoral degree in technical science from Graz University of Technology, Graz, Austria, in 2002 and 2006, respectively. In 2009 he received the *venia docendi* (habilitation) for Electrical Measurement and Signal Processing.

Since 2017 he has been a Full Professor of Automated Driving at the Institute of Automation and Control, Graz University of Technology, Austria.

In addition, he is Divisional Director and Scientific Head of the Automotive Electronics and Software Department of the Virtual Vehicle Research Center in Graz. Since 2015, he has been Vice President of the ARTEMIS Industry Association (European Research Association on Embedded Control Systems).

His research interests focus on sense & control of automated vehicles, signal processing, sensor fusion, uncertainty estimation and quantification, robust optimization, non-invasive measurement techniques, and sampling methods. He is IEEE Senior Member of the IEEE Control Systems, Signal Processing and Instrumentation & Measurement Societies.

He is author or co-author of over 150 peer-reviewed papers, book chapters, patents, and articles. In 2016, he published the book "Automated Driving – safer and more efficient future driving", describing the state-of-the-art of this emerging technology (ISBN 978-3-319-31893-6). He is the Austrian representative (appointed by the Austrian Federal Ministry BMVIT) of Electrified, Connected, and Automated Vehicles within the International Energy Agency (IEA). Since 2016 he has been the Chair of the Task Force on "Methods and Tools for Virtual Development and Validation" of the European Automotive Research Partners Association (EARPA).