Special Session: Future Automotive Systems Design: Research Challenges and Opportunities

Selma Saidi
Hamburg University of Technology
Hamburg, Germany
selma.saidi@tuhh.de

Sebastian Steinhorst
Technical University of Munich
Munich, Germany
sebastian.steinhorst@tum.de

Arne Hamann, Dirk Ziegenbein
Robert Bosch GmbH
Renningen, Germany
arne.hamann | dirk.ziegenbein@de.bosch.com

Marko Wolf
ESCRYPT GmbH
Munich, Germany
marko.wolf@escrypt.com

***Abstract*—Automotive systems are currently undergoing a radical shift in the way they are designed, implemented and deployed. Such changes impose an increased complexity and a high number of requirements in order to be more automated, connected, dependable and cost-effective. This raises several challenges that the embedded systems community has to face, encountered at different stages of systems development from modeling and design to software/hardware deployment and validation. This paper discusses current industrial trends and open problems in the hardware and software design of automotive systems, dictating rigorous constraints in terms of safety, real-time capabilities and security which have to be considered in future automotive systems design.**

I. Introduction

The landscape in the automotive industry is rapidly changing, driven by new automotive functionalities and applications such as autonomous driving and fueled by the entry of new players from the IT industry. Therefore, there is a need to leverage new technologies and integrate advanced autonomous systems for future mobility solutions that will be electrified, automated and connected.

Consequently, we witness the emergence of new trends in the automotive field imposing new challenges at the hardware and software level and at the way future automotive systems will be designed and developed. This includes the integration of formerly separated function domains onto centralized computing platforms, leading to a heterogeneous mix of applications with different models of computation, and the diversification and specialization of hardware platforms (comprising e.g., application cores, safety cores, deep learning accelerators, issued across several control units and connected by networks). Such platforms are required to meet the tremendous increase of computing power enforced by functionalities such as automated driving or on-line optimization of operating strategies for electrified powertrains. Furthermore, there is an increase of the connectivity beyond vehicle boundaries, such as vehicle-to-vehicle and vehicle-to-infrastructure communication (V2X) leading to systems-of-systems integration problems including function deployment from control unit over edge to cloud infrastructure as well as questions of how to guarantee sufficient availability of communication links and security on these connections.

These new trends constitute a real paradigm shift in the way classical automotive systems are viewed, thereby heavily challenging established methods and methodologies in automotive systems design. This provides a great opportunity for practitioners from both academia and industry to rethink the design of future automotive systems in order to accommodate high efficiency, safety and security requirements. However, the scientific effort provided by academic institutions often does not match the needs of the industry as the proposed solutions usually fail to consider the automotive boundary conditions. This often emerges from a lack of inter-disciplinary interaction and the absence of clear and reported industrial challenges provided to the research community.

In this paper, we first discuss the main trends that are currently driving innovation in the automotive industry. We then present an overview of the emerging challenges and application constraints necessary to comply with the increased requirements with respect to cost, variability, support of legacy software and dependability. The goal is to raise awareness about emerging needs and application constraints in automotive, required to devise highly efficient automotive systems and systems-of-systems.

II. Current Trends in Automotive Systems Design

A. The Move to Centralized E/E Architectures

Traditionally, automotive Electric/Electronic (E/E) architectures consist of around 100 mainly mono-functional electronic control units (ECUs), which are ordered into domains like chassis, powertrain, comfort and infotainment and connected via a mix of in-vehicle networks featuring CAN, LIN, Flexray and Ethernet. For over two decades a move from mainly mono-functional ECUs to domain- or vehicle-centralized architectures has been advocated but did not materialize beyond occasional small-scale integrations, such as a combined engine and transmission control ECU. While it had been shown in studies or publicly funded projects (e.g. RACE [4]) that such centralized E/E architectures can be advantageous in terms of

DOI: 10.1109/CODESISSS.2018.8525873

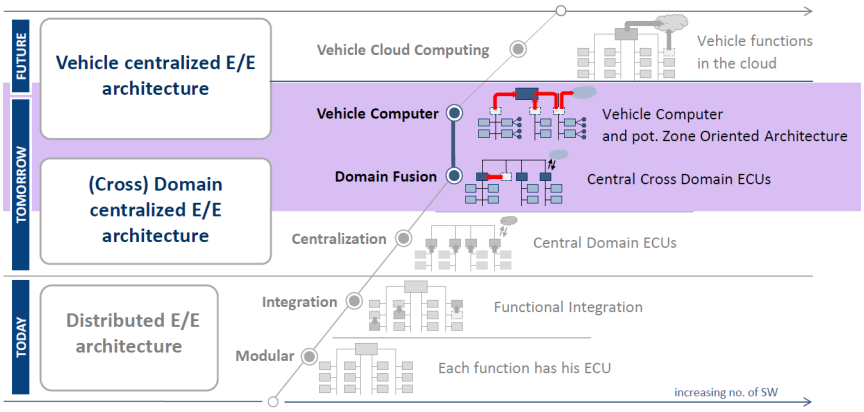

Fig. 1. Roadmap showing the evolution from mono-functional ECUs to centralized E/E architectures (source: [8])

cost, space and weight, they remained, until recently, roadmap items (see e.g. Figure 1) in industrial practice.

One factor in this delayed appearance can probably be best explained by Conway's law [5] which in brief states that the system architecture follows the structure of the developing organization, which for traditional vehicle manufacturers and suppliers typically developed along domains and functionalities of a vehicle. So the market entry of players starting on a clean sheet without such legacy certainly helped the introduction of centralized E/E achitectures.

An even more dominant influence had the advent of new functionalities like automated driving or connected vehicle functions such as predictive cruise control as well as app-like business models which introduced additional reasons to change the vehicle's E/E architecture:

- The computational power and communication bandwidth required by these new functionalities exceeds the capabilities of the current compute nodes (mainly microcontroller SoCs) and triggers the use of a new class of computing nodes featuring more powerful microprocessors and GPUs. The central driver assistance control unit zFAS [2] by Audi integrates several functionalities from the driver assistance domain and features two powerful MPSoCs (Nvidia Tegra K1 and a Mobileye EyeQ3). The presence of such compute nodes in the vehicle eases the off-loading of specific functionalities from other ECUs. An important side effect of the dramatic increase in computing power is that the electrical power consumption of these compute nodes now becomes a major factor in the overall energy balance of the vehicle.
- While currently the control systems in the vehicles are developed to be fail-safe, the nature of automated driving functions (for levels L4/L5) requires the vehicle control to become fail-operational, because the driver as fall-back solution is no longer present. This commands a degree of redundancy which can be provided in centralized architectures at a lower cost.
- Optimization functions of a vehicle's range efficiency based on an electronic horizon, fleet information or predictive maintenance can be implemented only or most efficiently when having a connectivity between the vehicle and a cloud back-end. The opportunities and risks of this communication link can be best managed using a centralized powerful communication control unit.

B. Impact on Automotive Systems Design

While this trend towards centralized E/E architectures certainly changes the hardware as well as the software architectures in vehicles, it has even more significant impact on the development process of future automotive systems:

- Now there are heterogeneous applications co-existing on the same HW platform, heterogeneous not only in their model of computation (ranging from classical periodic control [7] over event-based planning to stream-based perception applications) but also in their criticality in terms of real-time and safety requirements. Isolating these applications from each other causes new integration challenges, in contrast to the often stated argument that a centralized architecture reduces the complexity of integration.
- The burden of this integration is shifted from the network to the ECU level and in this regard typically from the vehicle manufacturer to the supplier of the control unit. This requires for new competences on the side of this integrator as well as on the side of other suppliers who now deliver software functions to the integrator instead of ECUs to the vehicle manufacturer. In this context, software verification gains importance over the system test. And as the verification of the control functions typically still needs to be performed in closed loop, there is a significant shift from hardware-in-the-loop systems to software-in-the-loop frameworks.
- The connectivity beyond vehicle boundaries enables updating the vehicle software without a trip to the dealer or repair shop. This gives vehicle manufacturers not only the possibility to offer (and charge for) functional updates and services but also paves the way for field-based validation where, e.g., an assistance function is being validated in

a deactivated silent mode while the vehicle is driven by the customer. For the automotive industry this prolongs the development process until after the deployment which creates new challenges in terms of release and homologation. Further, this software over the air update creates a chicken and egg problem w.r.t. security: On the one hand, the cloud connectivity significantly widens the security attack surface, on the other hand, the cloud connectivity is mandatory to install security updates in a cost-efficient manner.

- Due to the increased complexity of functionality, interference paths, supplier relationships and software updates, compositionality becomes a key property of any development step or design paradigm. In this context, compositionality means that local changes may only have a bounded local effect such that incremental verification and validation are possible. Design paradigms with a "change anything, change everything" characteristic will be extremely costly and not be sustainable in the long run.

Finally, it is important to stress that the traditional boundary conditions of automotive systems design still hold. Electronic control units and vehicle networks are extremely cost-sensitive, need to provide dependability guarantees (most prominently w.r.t. safety, security and real-time) and cope with a high degree of variability to cater the individualization possibilities customers are used to. Furthermore, the established incremental development model still mandates the support of legacy software and its porting to new HW/SW platforms.

III. Emerging Challenges and Opportunities

A. The Shift to High Performance Computing Platforms

The integration trend and increased required computing power is driving the adoption of common embedded consumer devices as new building blocks for automotive centralized E/E architectures. These platforms include heterogeneous Multi-processor Systems-on-Chip (MPSoCs) [2] integrating general-purpose ARM-based microprocessors and accelerators such as GPUs, previously restricted to the high performance computing domain.

Compared to micro-controllers, general-purpose MPSoCs are highly parallel and feature an increased I/O interaction and a more complex memory system composed of multiple levels of on-chip shared SRAMs memories (caches or scratch-pads) and off-chip DRAMs, in addition to dedicated DMA engines providing hardware support for moving data between memory locations. However, with the increased complexity in the memory system also appears an increased correlation between the execution of a program and the access to the data it manipulates. This data is transparently available to the program through virtual address space, however it is in practice stored in different remote memory locations with different access latencies and accompanied with complex mechanisms for ensuring data coherency and consistency. The adoption of this class of systems therefore poses the following main challenges.

Ensuring timing prediction: Many existing approaches in embedded systems design rely on the worst-case execution time (WCET) abstraction [9], [15] as an interface between the software and hardware where the timing effect of shared hardware components is captured during code-level analysis and concealed in the WCET of a task. This abstraction was valid for many years where micro-controllers platforms were predominant. However, with the advent of complex heterogeneous MPSoCs architectures, the performance effects due to micro-architecture choices, shared resources, interconnects, and memory hierarchy are becoming a dominant factor in systems design that cannot be solely captured by the WCET considering simple hardware architectural models and a holistic analysis approach. In heterogeneous MPSoCs, the timing effect of memory contentions is by orders of magnitude more severe compared to mircocontrollers. This effect is further worsened in the presence of data intensive applications which are becoming an important class of automotive systems to ensure advanced automated driving assistance functions. Therefore, bounding the timing effect of memory contentions in architectures with a multi-level shared memory hierarchy and complex mechanisms for transferring data becomes extremely challenging. Novel analytic approaches as well as mechanisms are needed to constructively control contention effects in the memory hierarchy and ensure timing predictability.

Safe migration of automotive legacy code: Standard techniques such as multithreading and prefetching are commonly used in order to reduce or hide the high memory latency in microprocessor architectures. Coping with these capabilities when migrating legacy automotive software is very challenging to guarantee that the legacy code which initially developed for a single threaded environment remains safe when executed in a multi-threaded environment. The main problem emerges with global variables and shared structures used by the program which can be accessed by other threads executed concurrently. For this, the code has to be re-factored and re-architected in order to preserve data consistency. Appropriate formal verification approaches are needed to guarantee that the functional as well as the timing behavior of the automotive software still complies with the required specifications.

B. Integrating Heterogeneous Software Applications

The move to the so-called domain- and vehicle-centralized E/E architectures leads to the major challenge of integrating heterogeneous applications from different domains (e.g., driver assistance, active safety, connected functions) and with different QoS requirements following different *models of computation* onto the same hardware platform, while ensuring that individual real-time and safety requirements are met.

Furthermore, there is an emergence of new business models in the automotive domain where pure software functions are sold and must be integrated into many different execution contexts (nomadic software). Such business models can only be successful if the software functions and the underlying platforms enable efficient *portability* and *composability*. Achieving this for embedded software sensing the environment and

controlling physical processes is by far no trivial task. Please note that many technical aspects for realizing SW-Download over the Air (SotA) are variations of the challenge to achieve software portability and composability.

As a consequence, there is a need for advanced and expressive *Execution Management Mechanisms* that enable shared resources (especially the processing units) to be provisioned in a "correct-by-construction" manner. Constructive approaches are especially important given the emerging distributed development style, where one integrator is in charge of integrating software functionalities from many different vendors onto a single platform. Clearly, traditional integration approaches where one vendor is in control of most design parameters to optimize the final system are not feasible anymore.

There are several challenging key requirements imposed on upcoming execution management mechanisms to support complex integration scenarios, which we summarize in the following.

1) Temporal isolation: One of the essential features of future execution mechanisms is that temporal isolation between applications must be guaranteed. This property ensures that the execution behavior and temporal properties of a given application is independent of the behavior of other co-existing applications. With this, applications can be verified in isolation and easily integrated on the target platform, thereby reducing qualification efforts.

2) Support for heterogeneous applications: Another key requirement of a future execution mechanism is that it should be capable of addressing the real-time requirements of heterogeneous applications with different *Models of Computations*. This is in terms of varying workloads (massive streaming data, short scalar data), activations (time triggered, event triggered), data flow semantics (synchronous, actor based, register semantics, independent activations, task chains), real-time characteristics (soft, hard, firm, weakly-hard) and fixed and dynamic workloads. Application heterogeneity also encompasses integrating tasks with different criticalities and safety requirements on the same platform.

3) Composability with precise QoS control: The execution mechanism must enable composability, allowing to add new applications seamlessly into an existing system without affecting the behavior of existing applications, and this without the need for reconfiguration or reparametrization. Additionally, it should be possible to configure the minimum resources that each application will receive. The mechanism should inherently be able to enforce these minimum guarantees so as to control the QoS of different applications.

4) Efficiency: Typically, most execution mechanisms ensure temporal isolation by "fixed allocations" and by dimensioning systems according to their worst-case requirements. However this comes at the cost of wasted system resources. Hence the need for efficient execution mechanisms that utilize the system resources meaningfully.

System designers have traditionally employed scheduling mechanisms like Fixed Priority Preemptive scheduling (FPP) or Time Division Multiple Access (TDMA) to integrate multiple applications onto a common HW platform. Fixed priority preemptive scheduling ensures that at any given time, the processor executes the highest priority application among all applications that are currently ready to execute, see Figure 2. Alternatively, TDMA applications are executed in exclusively assigned and periodically repeating time slices, see Figure 3. Thereby, if an application does not fully use its assigned time slice(s) the system idles until the time slice of the next ready application is reached.

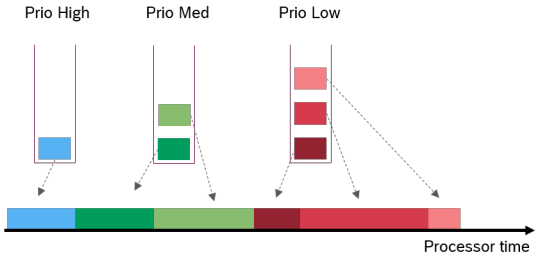

Fig. 2. Fixed Priority Scheduling

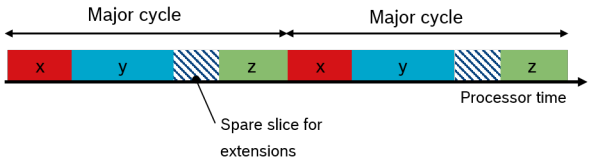

Fig. 3. TDMA Scheduling

These mechanisms were adequate in dealing with traditional automotive applications that were simpler as they had well-known activation patterns (usually periodic) and execution requirements (worst-case execution times) that were easy to obtain for the used simpler micro-controllers. Hence it was easy to reasonably dimension a system for these static and periodically occurring workloads. However, emerging applications have more complex activation and data flow semantics and it is difficult to precisely determine their execution requirements on heterogeneous HW platforms.

Table I summarizes the shortcomings of FPP and TDMA with respect to the aforementioned key requirements for future execution mechanisms. The main disadvantages of FPP is that it is not composable and does not provide temporal isolation. For instance, a newly added functionality affects the temporal behavior of all lower priority functionalities. In other words, temporal guarantees must be reassessed, and usually a system reconfiguration is necessary. Due to its static nature, TDMA also suffers from non-composability. It is built for ensuring temporal isolation but at the cost of heavy inefficiency due to its *non work-conserving* nature where unused time slices cannot be utilized opportunistically by other applications. This makes TDMA in particular unsuitable for heterogeneous applications with varying execution requirements. Even if the system integrator "reserves" spare slices for future extensions, it is not guaranteed that the size and position of those slices

Characteristics	FPP	TDMA
Temporal Isolation	**No** (no protection against overruns)	**Yes** (via exclusive time slices)
Efficiency	**limited** (for predictable periodic workloads only)	**No** (non work-conserving)
Heterogeneous Application support	**No** (limited support for dynamic workloads)	**No** (dynamic workloads not supported as per design)
Composability w/ QoS Control	**No** (interference on lower priority levels)	**No** (static time slice configuration)

TABLE I
SHORTCOMINGS OF ESTABLISHED EXECUTION MANAGEMENT APPROACHES

in the major cycle fit the execution requirements. Therefore, adding an application into an existing TDMA-based system typically also results in a system reconfiguration and revalidation. This calls for novel scheduling or execution management mechanisms.

C. Reconciling Predictability and Performance

Automotive systems are increasingly using Commercial-Off-The-Shelf (COTS) components in an effort to raise performance and lower production costs. While their computational capabilities are well understood, there is an increased need to comprehend these platforms from the perspective of extra-functional requirements like predictability, determinism, and freedom-from-interference. This is essential given that, e.g., automated driving applications need to simultaneously satisfy multiple criteria like performance, timeliness, and safety.

COTS platforms are designed with the primary objective of providing maximum throughput. They are optimized for the average-case performance and extensively make use of shared resources, hence introducing a high degree of interference and imposing a strong timing correlation between concurrently running components on the same chip. Controlling interference on shared resources constitutes one of the main challenges for the integration of COTS components in automotive systems, since they may cause common-cause failures and induce unpredictable overloads violating the timing constraints of the system [12].

Several existing approaches rely on a strict temporal and spatial partitioning of shared resources in order to comply with the freedom-from-interference principle required by the safety standard ISO 26262. This class of solutions includes virtualization techniques based on hypervisors (e.g. PikeOS, Lynx) preserving safety by wrapping safety-critical functions in separate OS components which are given exclusive access to hardware resources shared using a static time partitioning scheme (TDMA). Other solutions rely on a physical partitioning of shared (memory and network) resources. One example is cache partitioning [1] where every task is assigned a private cache partition. Inter-task cache evictions are therefore avoided, leading to a reduction of contention effects and thus improved predictability.

However, these approaches suffer from a trade-off between predictability and performance. Constraining COTS platforms to guarantee a spatial and temporal isolation or enforcing the system to operate in the worst-case indeed improves predictability but this comes at the cost of decreased performance. Novel HW/SW mechanisms for controlling interference on COTS are needed to *statically* mitigate the effect of contention (e.g PREM [11]) and simplify the timing analysis of the system where standard techniques can still be applied, but which are also *dynamic* in order to guarantee a flexible (software and hardware) execution which currently emerges as a major requirement.

D. Programming Heterogeneous HW platforms

One important topic when it comes to heterogeneous HW platforms is programmability, i.e. a common framework and programming models that enable software engineers to seamlessly exploit the parallel capabilities of SoCs. Established (cross-platform) abstraction layers such as CUDA, OpenMP, OpenCL, and SyCL are crucial to enable portability and increase development efficiency. The problem, however, is that different HW platforms from different vendors support a different subset of those frameworks. For instance, the GPUs in the Nvidia Jetson series are programmed with CUDA, and OpenCL is not officially supported. The Xilinx Zynq UltraScale+, on the other hand, is programmable using OpenCL, and neither OpenMP nor CUDA are supported. The lack of a commonly supported programming model challenges the efficient use of heterogeneous platforms from an engineering point-of-view.

Another important issue is the lack of clear semantics from a performance point-of-view. Event though code written, for instance, in OpenCL might be ”code-portable”, it might not exhibit the same execution semantics on a different platform, or, in other words, it is not ”performance-portable”. Recent studies reveal performance pitfalls when using CUDA for programming real-time applications on NVIDIA GPUs [16]. That work showed by means of microbenchmarks that the documented (or intended) behavior of CUDA program code often diverges from the actual behavior on the platform. One prominent reason for that are blocking effects due to implicit synchronization that are not transparent to the programmer. Moreover, depending on the implementation, the behavior of the same piece of code may change between different HW platform generations. Therefore, there is a need for a clear formally defined programming model for heterogeneous HW platforms that encompasses real-time predictability, i.e. allowing to derive bounds on response times and blocking effects.

One approach consists in restricting some features of standard parallel programming languages like dynamic memory allocation in order to enforce predictability [14]. Other approaches could rely on developing middlewares capable of intercepting and reordering or delaying GPU operations.

E. Safety and Dependability

Automotive systems are undergoing a fundamental change regarding their safety and dependability requirements. While, until recently, fail-safety was the design goal of conventional system architectures, future systems need to be fail-operational. Consider a scenario where an autonomous (driverless) vehicle with only children as passengers is supposed to drop them off at a birthday party.

In a traditional fail-safe automotive scenario, any failure to a system component of the vehicle is supposed to ensure that the vehicle remains controllable and gets to a controlled stop. In the above autonomous scenario, however, it would be unacceptable that the vehicle just safely stops somewhere on the way to the destination and unlocks the doors in case of a system failure, as this might put the passengers at high risk. Therefore, fail-operational systems are not only required to maintain safety aspects, but also have to maintain higher-level functionality aspects of the system. As a consequence, the requirements for such a fail-operational system architecture mandate a complete change in the design paradigms such that a system remains operational in case of failure [3]. However, as automotive systems are highly price sensitive, established fail-operational approaches such as triple modular redundancy (on hardware and software levels) from the avionics domain cannot be implemented. Thus, software-controlled redundancy will be a core requirement of emerging consolidated automotive computation platforms, which, in turn, completely changes the value chain of automotive system components and their architecture integration.

Compared to other application domains such as avionics, the automotive domain is currently embracing a new class of technologies such as machine learning which cannot be considered within the criteria of functional safety introduced by ISO 26262 [6]. However, in order to enable autonomous driving, there seems to be no alternative to the application of such black-box methods which violate any predictability assumptions of the system behavior. Methods to enable verification and eventually certification of machine learning within an automotive safety context are urgently required for a mass roll-out of autonomous driving features. At the same time, a new class of safety standards has to be developed which is considering system architectures combining predictable and unpredictable computational methods as well as functional insufficiencies of the system components.

F. Security and Privacy

New disruptive automotive technologies such as connected cars, e-mobility, and especially autonomous driving dramatically increase the importance and complexity of automotive cybersecurity. Automotive cybersecurity, in turn, is essential to fulfill increasing financial, legal, and especially safety protection requirements against malicious infringements of any kind. However, these new disruptive challenges can hardly be handled with most existing automotive security engineering approaches (if any) that are often based on classical automotive engineering, for instance, focusing on single functions, focusing on technology only, and usually ending with start of production (SOP). In order to handle these upcoming challenges adequately, it is mandatory to develop a holistic cybersecurity orchestration approach that covers:

- The complete technology chain from the ECUs silicon layer up to the connected IT services in the cloud,
- the complete product lifecycle from the first idea until final phase-out, and
- the complete organization from internal security processes, roles, functions, and policies up to security orchestration of external suppliers and third parties.

Applying this holistic cybersecurity orchestration approach, automotive manufactures will be able to manage also upcoming disruptive automotive cybersecurity challenges efficiently and effectively.

There is an inherent challenge to implement security on constrained embedded devices widely used in ECUs. In case of sufficiently powerful computational resources, conventional established security measures from the domain of Internet communication can be applied. While in the domain of vehicle-to-X communication dedicated gateways can be integrated into vehicles which enable both symmetric as well as asymmetric cryptography satisfying the real-time requirements, most existing automotive in-vehicle system architectures are not capable to implement conventional approaches for device authentication and message stream authorization down to the level of, e.g., ECUs for electrical seat adjustment. Thus, there is a requirement to develop and implement efficient lightweight security approaches such that the vehicle is secured both against attacks from the outside as well as within its in-vehicle network [10]. Introducing such security aspects includes considering, e.g., how to cope with the replacement of vehicle components such that a reauthentication can be performed efficiently but only by trusted parties.

Security has an immediate impact on the safety of a vehicle, as a security breach might violate safety functionality. At the same time, even without a successful attack on the security integrity of a vehicle, privacy aspects are gaining importance. The increasing connectedness of vehicles with the sharing of driving parameters brings questions of data ownership to the table. Once autonomous driving has emerged to a level where mobility is considered as a service, all route information, number of passengers, etc. will be shared with the service providers. From a conventional perspective where driving a vehicle used to be synonymous with being in a private space which is only controlled by the driver, this requires introduction of consequent privacy policies and a regulatory positioning towards data ownership questions as a trade-off between the interests of service providers and users.

G. Energy Management

Until recently, the in-vehicle electronics had a negligible contribution to the power requirements of cars, satisfied by the generator of internal combustion engine vehicles or directly provided by the Electric Vehicle (EV) battery. With the introduction of further ADAS systems towards full autonomous

driving capabilities, the energy aspects of computational platforms in cars have to be revisited. High-performance computing for computer vision and machine learning applications can increase the power requirements by orders of magnitude. Especially with EVs still being limited regarding their driving range due to the relatively low energy density of Lithium-Ion batteries compared to gasoline[1], the increasing power requirements for advanced computation become relevant and demand for low-power computing, as every kWh saved for computation results in a couple of kilometers of additional driving range.

In the domain of energy management for EVs, battery management has gained importance. Managing the battery to maximize its usable capacity and lifetime are relevant due to the aforementioned constraints regarding the available maximum capacity that can be economically installed in vehicles. At the same time, information from acquired driving profiles or connected mobility applications can be utilized to optimize the state of charge of batteries when charging and for route optimization.

Similar to the computational platforms for the general vehicle functionality changing, battery system architectures and management techniques offer ample research opportunities, as the battery is the single most expensive component of a vehicle. Changing conventional centralized battery architectures and their management electronics towards a decentralized setup to increase robustness, scalability and modularity could introduce further opportunities for sustainable mobility concepts [13].

IV. Conclusion

Automotive systems constitute a class of embedded systems with a specific set of constraints that must be met during design and deployment. With the presented different challenges and opportunities outlined in this paper, it becomes clear that an interaction between different disciplines combining resource-constrained embedded, cyber-physical and real-time aspects is mandatory. This provides a great opportunity for different research communities to *collaboratively* work on innovative solutions, capable of meeting the needs of future automotive systems design while being compliant with automotive systems constraints.

V. Acknowledgement

This work was partially funded within the project ARAMiS II by the German Federal Ministry for Education and Research with the funding ID 01IS16025. The responsibility for the content remains with the authors. With the support of the Technische Universität München - Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement n°291763. Additionally, we would like to thank Gurulingesh Raravi from NVIDIA for his participation to the Special Session organized around this topic.

[1] 1 liter of gasoline contains about 10kWh of energy, which can be used with an efficiency of 25-40%, while a battery storing the equivalent usable energy at least weighs about 20-30 times more

References

[1] S. Altmeyer, R. Douma, W. Lunniss, and R. I. Davis. Outstanding paper: Evaluation of cache partitioning for hard real-time systems. In *2014 26th Euromicro Conference on Real-Time Systems*, pages 15–26, July 2014.

[2] AUDI. Audi a8 - central driver assistance controller (zfas).

[3] K. Becker and S. Voss. Analyzing graceful degradation for mixed critical fault-tolerant real-time systems. In *2015 IEEE 18th International Symposium on Real-Time Distributed Computing*, pages 110–118, April 2015.

[4] M. Buechel, J. Frtunikj, K. Becker, S. Sommer, C. Buckl, M. Armbruster, C. Klein, A. Marek, A. Zirkler, and A. Knoll. An automated electric vehicle prototype showing new trends in automotive architectures. In *Proceedings of the IEEE 18th International Conference on Intelligent Transportation Systems (ITSC 2015)*, 2015.

[5] M. E. Conway. How do committees invent? *Datamation*, April 1968.

[6] F. Falcini, G. Lami, and A. M. Costanza. Deep learning in automotive software. *IEEE Software*, 34(3):56–63, May-Jun. 2017.

[7] S. Lampke, S. Schliecker, D. Ziegenbein, and A. Hamann. Resource-aware control - model-based co-engineering of control algorithms and real-time systems. *SAE International Journal on Passenger Cars - Electronic And Electrical Systems*, 2015.

[8] M. Lunt. E/e architecture in a connected world, 2017.

[9] M. Lv, N. Guan, J. Reineke, R. Wilhelm, and W. Yi. A survey on static cache analysis for real-time systems. *LITES*, 3(1):05:1–05:48, 2016.

[10] P. Mundhenk, A. Paverd, A. Mrowca, S. Steinhorst, M. Lukasiewycz, S. A. Fahmy, and S. Chakraborty. Security in automotive networks: Lightweight authentication and authorization. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 22(2):25, 2017.

[11] R. Pellizzoni, E. Betti, S. Bak, G. Yao, J. Criswell, M. Caccamo, and R. Kegley. A predictable execution model for cots-based embedded systems. In *2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 269–279, April 2011.

[12] S. Saidi, R. Ernst, S. Uhrig, H. Theiling, and B. D. de Dinechin. The shift to multicores in real-time and safety-critical systems. In G. Nicolescu and A. Gerstlauer, editors, *2015 International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2015, Amsterdam, Netherlands, October 4-9, 2015*, pages 220–229. IEEE, 2015.

[13] S. Steinhorst, M. Lukasiewycz, S. Narayanaswamy, M. Kauer, and S. Chakraborty. Smart cells for embedded battery management. In *Proceedings of the 2nd International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA 2014)*, pages 59–64, 9 2014.

[14] M. M. Trompouki and L. Kosmidis. Brook auto: high-level certification-friendly programming for gpu-powered automotive systems. In *Proceedings of the 55th Annual Design Automation Conference, DAC 2018, San Francisco, CA, USA, June 24-29, 2018*, pages 100:1–100:6. ACM, 2018.

[15] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström. The worst-case execution-time problem - overview of methods and survey of tools. *ACM Trans. Embed. Comput. Syst.*, 7(3):36:1–36:53, May 2008.

[16] M. Yang, N. Otterness, T. Amert, J. Bakita, J. H. Anderson, and F. D. Smith. Avoiding Pitfalls when Using NVIDIA GPUs for Real-Time Tasks in Autonomous Systems. In S. Altmeyer, editor, *30th Euromicro Conference on Real-Time Systems (ECRTS 2018)*, volume 106 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:21, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.