System Architecture and Software Design for Electric Vehicles

Martin Lukasiewycz, Sebastian Steinhorst, Sidharta Andalam, Florian Sagstetter,
Peter Waszecki, Wanli Chang, Matthias Kauer, Philipp Mundhenk
TUM CREATE, Singapore
martin.lukasiewycz@tum-create.edu.sg

Shreejith Shanker, Suhaib A. Fahmy
Nanyang Technological University, Singapore
sfahmy@ntu.edu.sg

Samarjit Chakraborty
TU Munich, Germany
samarjit@tum.de

ABSTRACT

This paper gives an overview of the system architecture and software design challenges for Electric Vehicles (EVs). First, we introduce the EV-specific components and their control, considering the battery, electric motor, and electric powertrain. Moreover, technologies that will help to advance safety and energy efficiency of EVs such as drive-by-wire and information systems are discussed. Regarding the system architecture, we present challenges in the domain of communication and computation platforms. A paradigm shift towards time-triggered in-vehicle communication systems becomes inevitable for the sake of determinism, making the introduction of new bus systems and protocols necessary. At the same time, novel computational devices promise high processing power at low cost which will make a reduction in the number of Electronic Control Units (ECUs) possible. As a result, the software design has to be performed in a holistic manner, considering the controlled component while transparently abstracting the underlying hardware architecture. For this purpose, we show how middleware and verification techniques can help to reduce the design and test complexity. At the same time, with the growing connectivity of EVs, security has to become a major design objective, considering possible threats and a security-aware design as discussed in this paper.

Categories and Subject Descriptors

C.0 [**Computer Systems Organization**]: General—*System architectures*

General Terms

Design

Keywords

Electric Vehicle, Software Design, System Architecture

DAC '13, May 29–June 7 2013, Austin, TX, USA.

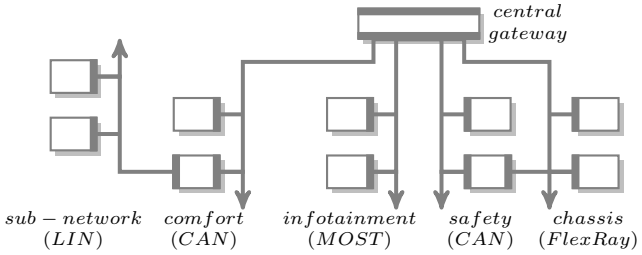

Figure 1: Illustration of a typical in-vehicle network architecture of a modern automobile. Various ECUs (▭) are interconnected via different buses. Gateways (▭) are used to interconnect the buses.

1. INTRODUCTION

EVs are widely accepted as emerging and sustainable solution to environmental and transportation challenges in growing mega-cities. In contrast to Internal Combustion Engine (ICE) cars, EVs comprise several components like the battery, electric motor, or powertrain that bring along new implementation, integration, and control challenges. Moreover, the implementation of drive-by-wire control and novel information systems may increase the safety and energy efficiency of EVs significantly. While these new components and systems bring along several fundamental challenges, EVs at the same time may serve as a platform to drive a paradigm change in the system architecture and software design for road vehicles.

Current top-of-the-range vehicles use embedded system architectures that consist of up to 100 ECUs with several heterogeneous buses that are interconnected by one or more gateways as illustrated in Figure 1. This complex network is a result of an incremental design over the last decades where new functionality is often introduced by adding separate hardware devices. The reasons for this *federated* approach can be found in the structure of the automotive industry where the car manufacturers obtain new functions from several different and competing suppliers.

This publication is made possible by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

While the federated approach has been feasible in the recent years, it is reaching its limits with the growing complexity of in-vehicle networks and the lack of installation space, particularly in EVs. With a growing computational power of novel processing units, the major trend goes towards a consolidation of ECUs and a unification of the in-vehicle network. This *integrated* approach will require an entirely different design methodology where functionality has to be developed for a shared underlying architecture.

In this paper, we give an overview of the challenges and present first results in the system architecture and software design for EVs. Section 2 gives an introduction to EV components, including the battery, electric motor, and powertrain. In Section 3, the trends in communication and computation devices are presented. The software design challenges, comprising the control design as well as the upcoming security in vehicles, are presented in Section 4. Finally, Section 5 makes concluding remarks.

2. ELECTRIC VEHICLE COMPONENTS

In the following, the essential components of EVs such as the battery, electric motor, and electric powertrain are introduced. Moreover, drive-by-wire and information systems are discussed. While it is to some extent possible to convert an ICE vehicle to an EV by replacing the powertrain, a tailored design is considered as more sustainable. Due to the shorter driving ranges and their emission-free operation, it is projected that EVs will play a major role in growing megacities. As a result, a design of the vehicle and its components has to take this scenario into account.

Battery Pack. In EVs, the battery pack is currently the most essential and expensive component while it is also a major bottleneck restricting the driving range of the vehicle. These Electrical Energy Storages (EESs) require both high energy and power density to optimize the utilization of allocated weight and volume of the battery. In this context, Lithium-Ion (Li-Ion) batteries are widely considered to dominate other battery chemistries [1]. However, Li-Ion batteries are sensitive to their operating parameters and exceeding specified bounds such as overcharging or undercharging cells negatively impacts the reliability by causing damage to the battery. In the worst case, this damage leads to a thermal runaway resulting in fire or explosion that poses a serious safety issue.

Therefore, sophisticated Battery Management Systems (BMSs) are applied to maintain the battery in a safe and healthy operating state. Figure 2 illustrates a state-of-the-art BMS architecture in a hierarchical approach. This incorporates monitoring the State of Charge (SoC) of battery cells by measuring voltage levels and the current drawn from the cells, as well as the temperature of cells.

As series-connected battery cells charge and discharge unevenly, Cell Balancing (CB) is required to maintain an equalized overall SoC for the battery. This equalization is achieved by either passive or active CB. Passive approaches that are state-of-the-art discharge cells with a higher SoC over a resistor to the charge level of the cell with the lowest SoC. In contrast, active approaches transfer charges between cells to avoid the waste of energy, increasing the driving range as well as the lifetime of the battery. A recent advancement in the area of active cell balancing system design is presented in [2].

Electric Motor. Synchronous motors are commonly used in EVs due to their high efficiency and light weight [4]. As shown in Figure 3, the motor is driven by sinusoidal waveforms that are controlled by the inverter. The inverter is using six

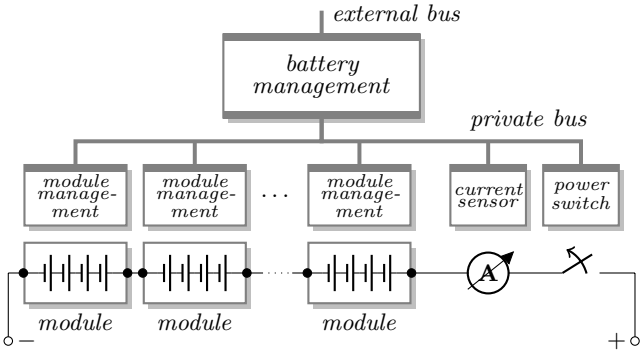

Figure 2: Illustration of the Battery Management System (BMS) consisting of a hierarchical architecture with battery cell modules controlled by module management devices, see [3].

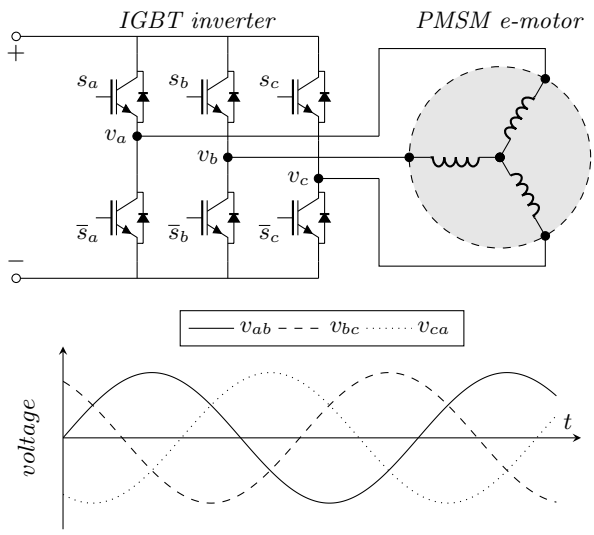

Figure 3: Illustration of the control of a Permanent Magnet Synchronous Motor (PMSM) using an inverter consisting of six Insulated Gate Bipolar Transistors (IGBTs) $s_a, \overline{s}_a, s_b, \overline{s}_b, s_c, \overline{s}_c$. The IGBTs are controlled such that the three voltages are sinusoidal waveforms that are phase-shifted by $\frac{2}{3}\pi$.

Insulated Gate Bipolar Transistors (IGBTs) to convert the DC source from the battery pack to AC current with the desired frequency and magnitude to drive the motor. For this purpose, the IGBTs work as electronic switches that are controlled by Pulse Width Modulation (PWM) signals using the space-vector modulation technique [5]. Here, proper switching sequences for all six IGBTs guarantee that the output is the desired three-phase alternating current with specific phase shifts.

The efficient and reliable control of electric motors in EVs can become a challenging task. For instance, due to harsh operating environments like high voltages, switching frequency, and changing temperatures, IGBTs might fail. One or more faults in the IGBT package make the output current no longer sinusoidal which drives the motor into unpredicted

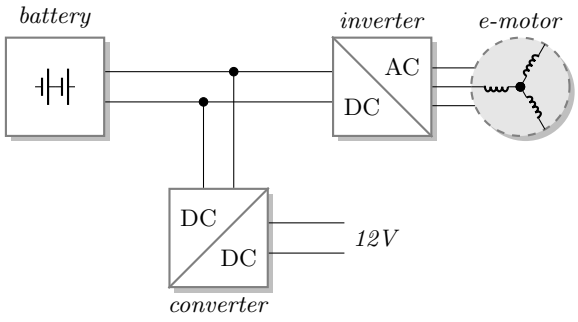

Figure 4: Illustration of a basic electric powertrain of a full electric vehicle.

operating modes and thus may jeopardize passenger lives. Therefore, a fault-tolerant control strategy is desirable to ensure normal operation of the motor under one fault or a combination of different faults. The solution should be optimized to require minimal redundant hardware while covering a high amount of faulty situations. The controller needs to calculate post-fault PWM sequences and react in real-time, requiring efficient algorithms to ensure that the motor returns to the best possible operation quickly enough such that safety requirements of the vehicle are not compromised.

Electric Powertrain. The electric powertrain is a distinctive feature of EVs. It is understood that the transition from ICE cars to EVs cannot be accomplished by simply replacing the ICE with an electric motor and the fuel tank with a battery. As illustrated in Figure 4, an architecture for the powertrain of an EV has to be developed for the motor, battery, converters, and inverters which needs a sophisticated control to operate in an effective and efficient way. Energy flow control is of particular importance in this context [6]. This incorporates the management of the energy storage as well as the energy consumers and providers.

As both the electric motor and the battery can operate as energy sources and consumers, depending on the driving state such as acceleration or regenerative braking, the control and the timing of energy flows are crucial design tasks that significantly affect the energy efficiency of the overall system. To increase the efficiency, a distributed real-time control becomes inevitable. Moreover, a conversion between different voltage levels may be required in the powertrain. Therefore, an optimization of the different conversion levels using DC-DC converters is equally important for the energy efficiency of the architecture.

Information Systems. One major challenge in EVs is the limited driving range due to the relatively small battery capacity. To cope with this drawback, an information system becomes necessary that ensures that driving ranges are never exceeded. At the same time, it is projected that information systems will be tightly coupled with next-generation entertainment systems.

Information systems in EVs need to be more sustainable than systems using fixed mounted screens that require a high amount of maintenance and are outdated within a short period of time in comparison to the fast progress in consumer electronics [7]. One way to achieve this goal is by establishing a native connection between the user devices such as smartphones or tablets and the vehicle. These user devices might interface the vehicle to access information, entertainment, and control related functionality. Implementations of this approach might consist of one central server component in the vehicle, offering wireless open access for user devices. Via the central vehicle server, the devices can be integrated with Vehicle-to-Vehicle (V2V) and Vehicle-to-Grid (V2G) infrastructures that are particularly important for EVs. Providing information on available charging stations to drivers can be further qualified by taking into account the locations, energy-consumption and destinations of all vehicles, as well as the number and location of charging stations. On the other hand, such an open system brings along several challenges in the domain of security that have to be addressed appropriately to guarantee that safety-critical applications are under no circumstances compromised.

Drive-by-wire. While drive-by-wire is a technology emerging also for ICE cars, it is of particular importance for EVs. In EVs, the acceleration is already not performed mechanically and as current battery technology only allows a limited range, energy recuperation during braking is essential to extend the driving range [8]. The energy savings using this mechanism as well as using prudent acceleration could be optimized via sophisticated and supportive control, enhancing the desired driver inputs. Therefore, a mechanical decoupling between the braking pedal and the brakes becomes necessary as provided by a brake-by-wire implementation [9]. Besides that, EVs are more straightforward to actuate by electronics than ICE cars.

However, since drive-by-wire is highly safety-critical, it needs to be designed in a fault-tolerant fashion, introducing a certain amount of redundancy in the control system. With most software errors being of systematic nature, straightforward component duplication may not be sufficient to reduce the associated risk to a sufficiently small level [10]. Functions may have to be implemented by different programmers or at least run on non-identical hardware. Additional care needs to be taken to fail-proof actuating and sensing devices. As a result, the introduction of this technology in the automotive industry where cost sensitivity prevails is facing certain obstacles. In recent years, the advent of more sensing devices and cheaper as well as more efficient ECUs have sparked the interest again in the context of EVs.

3. SYSTEM ARCHITECTURE

In the following, challenges and trends in the system architecture in EVs are presented. We give an overview of the in-vehicle communication as well as next-generation computational devices in EVs. A major trend goes towards the consolidation of ECUs, relying on a homogeneous network. This will require significantly more powerful computational devices as well as deterministic bus systems with a high bandwidth. In the automotive domain, this would require a paradigm change where EVs might serve as a platform for an entire redesign of the system architecture. This transition from *federated* systems with one function per single-core ECU to *integrated* systems, comprising several applications on one powerful ECU can impede the growing complexity of automotive architectures by reducing the overall number of controllers and thus increase the flexibility and reliability [11].

3.1 Communication

Protocols. Automotive architectures today make use of many different heterogeneous bus systems such as Controller Area Network (CAN), Local Interconnect Network (LIN), FlexRay, and Media Oriented Systems Transport (MOST), see Figure 1. As a result of the complexity of this in-vehicle network, integration and configuration became time and

cost intensive tasks that may even prohibit distributed functions that rely on predictable communication with high bandwidths. Therefore, one of the key challenges for next-generation architectures is the design of a unified in-vehicle network that allows to integrate and configure components and subnetworks transparently and independently of each other. The basis for such architectures are deterministic communication protocols. One suitable solution introduced in recent years is FlexRay which is based on a hybrid protocol supporting both time-triggered and event-triggered communication with a bandwidth of 10MBit/s [12]. For next-generation vehicles and particularly EVs, Ethernet with a bandwidth of 100MBit/s and more may become the first choice as a cost-efficient and light-weight implementation, using a physical layer implementation that allows twisted-unshielded pair cables [13]. As standard Ethernet is non-deterministic and therefore unsuitable for time-critical applications, extensions to support stringent real-time requirements become necessary. In this context, the Audio/Video Bridging (AVB) protocol is currently used in the automotive industry [14] for the implementation of entertainment systems, relying on a synchronization with Precision Time Protocol (PTP) [15] which might also serve as basis for other time-triggered Ethernet protocols.

Besides the communication between the components in the in-vehicle network, EVs also benefit from V2G and to some extend V2V communication. For this purpose, protocols like Bluetooth or IEEE 802.11 (WiFi) may become a common feature in the communication infrastructure. In this context, Service Oriented Architectures (SOAs) have to be employed to enable a provision of information and control services [16].

Time-triggered Scheduling. EVs implement many functions that require a deterministic communication such as battery management, electric motor control, and upcoming drive-by-wire implementations. Additional safety-critical functions will even increase the need for real-time communication with very strict end-to-end timing, such that time-triggered protocols will further gain importance. As schedules are defined at design time, time-triggered protocols ensure predictability and provide a deterministic communication. In particular, the time-synchronization between ECUs allows to obtain a global schedule, considering all tasks and messages.

While synchronous time-triggered scheduling allows to significantly reduce the end-to-end timing delays of applications, obtaining schedules for all tasks and messages concurrently is highly complex and existing approaches only provide limited scalability [17]. A remedy might be an integration approach where the configuration for each component or subsystem might be defined independently and integrated into a global schedule in the integration phase [18]. This methodology is also in accordance with the design approach in the automotive industry where individual components are designed and tested with a valid configuration before being integrated in a later stage.

3.2 Computation

Multi-core. The ever growing demand for innovation and new functionality in modern cars necessitates a paradigm shift in automotive system architectures in terms of hardware and software. In particular, the introduction of drive-by-wire applications and high-performance BMSs for EVs as well as new information systems clearly redefines the requirements of current hardware platforms in terms of computation, communication, and overall topology. This change, in turn, calls for the implementation of multi-core ECUs for which three main reasons can be pointed out. First of all, gaining higher computational performance without increasing the power consumption and heat dissipation can only be achieved by adding new cores rather than increasing clock frequencies of single-core processors [19]. Moreover, the enhanced performance per watt ratio supports the strict energy requirements in EVs and the higher level of parallelism provided by multi-core systems allows for the compliance with automotive safety standards like ISO 26262 [20]. Finally, a major advantage is that software from single-core ECUs can be ported to multi-core ECUs more easily than to architectural different computational platforms, like FPGAs.

Besides a reliable high-performance hardware, safety-critical applications as used for next-generation driver assistance systems would strongly benefit from real-time capable multi-core Operating Systems (OSs). By providing comprehensive software partitioning and resource sharing mechanisms an OS can guarantee the desired functionality. The challenge is to design a specialized multi-core OS which provides not only hard real-time guarantees for safety-critical tasks but also mechanisms for segregating trusted and non-trusted code and strategies for intelligent cache utilization. Furthermore, to ensure a fully predictable real-time behavior for both the multi-core ECU and the corresponding communication within the system architecture, only time-triggered execution models come into consideration. Although current multi-core OSs, like *PharOS* [21] or *Barrelfish* [22], are not able to fulfill the aforementioned requirements, they can highlight general design trends for the development of a reliable and deterministic multi-core OSs for system architectures of EVs.

GPU. Modern vehicles comprise a significant amount of image and sensor processing to increase the safety of all road users. In this context, there are specific challenges for EVs which for instance have almost soundless engines, making them extremely dangerous in situations where pedestrians come into close proximity with the vehicle. A remedy might be a radar or camera-based pedestrian recognition that is coupled with a warning system that generates audible alerts to pedestrians.

One possible implementation of the warning system can be achieved using cameras and image processing techniques. Cameras capture a snapshot of the environment and then the image is processed in real-time to detect any pedestrians. This image processing demands a lot of computational resources which can be provided by Graphical Processing Units (GPUs) [23]. When compared to a CPU, due to more hardware-level parallelism, a GPU is significantly faster at processing an image [23]. Thus, GPUs would make a good choice for implementing safety-critical functions that require a high amount of parallel processing.

GPU programming frameworks such as OpenCL and CUDA are generally used for programming GPUs [24]. However, due to the inherent parallelism, programming a GPU is much more complicated than programming a CPU. The programmer must be aware of the underlying hardware architecture and structure the program carefully to avoid multiple threads accessing the same memory locations at the same time, resulting in either inconsistent data or deadlock during memory access.

FPGA. Field Programmable Gate Arrays (FPGAs) have found favor in a number of application domains as a platform for accelerating complex algorithms. They offer the benefits of a custom hardware implementation at a fraction of the fixed cost of designing a custom Integrated Circuit (IC). Hence, they find use in medium-volume markets or those where evolving standards may require regular changes to the hardware.

While FPGAs are currently not often used in the automotive domain, they bring along several advantages for EVs. For computationally intensive tasks, within an embedded systems power budget, FPGAs are often the only sensible choice. Furthermore, since computation is implemented spatially, it is possible to completely isolate distinct tasks while maintaining their individual determinism. This could help to reduce the number of ECUs and at the same time guarantee the isolation between various safety-critical applications. Note that Application-Specific Integrated Circuits (ASICs) are cheaper and even more energy-efficient, but they might not provide the necessary flexibility that is required in case many functions are implemented on one ECUs.

An additional capability that is unique to FPGAs is that, as volatile devices, they can be reconfigured. This has so far primarily been used to allow for design iteration, or incorporation of improvements in subsequent revisions. However, this capability can also be leveraged at runtime, in the form of dynamic reconfiguration, allowing different applications to be implemented at different times. All these capabilities make the incorporation of FPGAs in EVs a promising prospect [25].

Advanced techniques like Partial Reconfiguration (PR) extend the capabilities of FPGAs for use in safety-critical applications. PR allows us to define fault-tolerant embedded computing units that can recover from faults by selectively reconfiguring the faulty module alone, while a redundant mode with lower specifications takes over control during the recovery process [26].

4. SOFTWARE DESIGN

A consolidated system architecture will require an entirely different function and software design approach in EVs. A central component of this software design approach might be a middleware that enables a more flexible development and operation. In the following, we discuss challenges and approaches in the area of software design for EVs. First, we present techniques to enable an efficient implementation of control functions using several different techniques. Finally, the issue of security in an increasingly connected vehicle is discussed.

4.1 Control

Middleware Approach. Current premium-cars already require millions lines of code [27] and it is projected that EVs will further increase the amount of software in vehicles. A major reason is the requirement to create additional sources of revenue for car manufacturers to cope with the high costs of batteries as well as the loss of the profitable service of the ICE. Such a source of revenue for EVs could be made possible by allowing additional purchasing of functionalities while the vehicle is already in operation. In turn, this will require a significantly more flexible system and software design to cope with the growing complexity of distributed control.

For the sake of higher flexibility and shorter time-to-market, a lean middleware approach is a potential solution. This middleware may abstract the underlying hardware and operating systems and enable a unified platform for the design of software while providing support for virtualization such that various tasks can be executed in parallel on the same hardware in an isolated fashion. As a result, software tasks may be distributed in a more flexible way, supporting the desired reduction of ECUs.

While flexibility is a major goal in software design for vehicles, the determinism of functions and the entire system has to be ensured. This might be achieved by incorporating the mentioned time-triggered mechanisms into the middleware as well as verification techniques.

Verification of distributed control systems. Many components in EVs require a precise and responsive control. Battery packs and upcoming drive-by-wire applications have a spatially distributed control where sensors, controllers, and actuators cannot be implemented on a single device or ECU, respectively. This leads to significant communication delays in the control loops that further complicate the correct design of the control functions. As a remedy, verification approaches might be used to formally guarantee the correct functionality of safety-critical control functions. This approach represents a significant improvement in current design flows where, although controller models are formally verified, their implementation on a distributed architecture is validated in an ad-hoc fashion with extensive testing and integration efforts.

The classic control-theoretic approach relies on idealized assumptions. Typically, computation and even communication are assumed to function perfectly and without delay. This is to a certain extent reasonable for single ECU systems where no communication takes place. Recently, verification techniques have been applied to tackle this problem. In [28], it was shown that an ω-regular language can be used as interface between the performance requirements for the control system and the transmission timings on a communication network. It is more expressive than an interface that only relies on combinations of periods and deadlines. In [29], this model is further extended such that an automaton framework for streaming systems is adapted to describe the transmission of control messages. It is then verified that the communication remains within the allowed patterns using model checking. In turn, this guarantees the initial performance requirement. A major challenge of these verification approaches that are very versatile remains the scalability.

Precise Timing Analysis. Safety-critical systems require guarantees on the functionality as well as the timing characteristics of programs. This requires modelling timing behaviour of micro-architectural features such as memory hierarchies, pipelines and buses to compute the Worst Case Execution Time (WCET) of a program. The ability to compute precise timing analysis is important and is dependent on the architectural features. For example, caches with replacement policies like LRU provide the best predictability, while PLRU and FIFO are much harder to analyse [30].

Static analysis of caches with various allocation and replacement algorithms have been studied in great detail [30, 31]. In general, there is a trade-off between the precision (tightness of the estimate) and the scalability (analysis time). For example, [31] captures the precise behavior of caches. However, this technique does not scale for large programs. In contrast, the approach in [30] avoids the state-explosion using an abstraction that scales for very large programs at the expense of reduced precision.

An alternative to caches are ScratchPad Memories (SPMs) that are fully software controlled caches. In SPMs, the allocation and replacement decisions are made in software, guided by compile time decisions. Recent work on SPMs focuses on developing software allocation algorithms and/or designing tailored architectures with SPMs [32]. SPMs are allocated statically and are easier to analyze, but they provide comparatively lower average and sometimes worst case performance when compared to caches. However, for successful implementation of safety-critical systems, it is important to emphasize on *predictability* rather than *performance*. Thus, SPMs are

an ideal design choice for implementations of safety-critical systems.

4.2 Security

Threats. A recent analysis of automotive architectures has shown that current series vehicles are often insufficiently protected against attacks aiming to temper the system. For instance, researchers were able to gain access to the in-vehicle network via Bluetooth and implemented a virus [33, 34]. While EVs have many security vulnerabilities in common with ICE vehicles, additional security threats are arising: Latest generation charging plugs implement a communication protocol to allow information exchange between the BMS and the charging station, e.g., for billing or for future V2G applications. This might allow man-in-the-middle attacks where the attacker attaches a connector between the charging plug of the car and the charging station. For a more detailed discussion of the security threats for electric vehicles see [35].

Security-aware Design. Due to a significantly higher connectivity of EVs, security has to become a major design objective for automotive architectures. Besides protecting potential access points, the charging plug or V2G communication interfaces with authentication approaches such as challenge response [36], additional security measures for the in-vehicle network become necessary. For instance, a secure in-vehicle communication is required that encrypts messages transmitted between ECUs, including an authentication of the sender. While existing buses like CAN are unsuitable for a secure communication due to the limited message size, upcoming protocols in the automotive domains such as Ethernet with IPSec [37] offer an established security solution. Additionally, as security leaks are commonly introduced by careless integration of secure components, a holistic design approach is essential to obtain a secure automotive architecture. The basis for such an architecture might be the discussed middleware which allows isolating the components and software functions from each other such that one compromised component does not affect the whole system.

5. CONCLUDING REMARKS

Already today, a vast majority of innovations in vehicles is driven by electronics and software. However, the current architectures grew incrementally over the past decades and the integration approach to implement new functions by adding new hardware devices is reaching its limits. Here, EVs may serve as a platform to implement a consolidated hardware architecture using middleware approaches to drastically simplify the integration, providing a solution to cope with the growing pressure to innovate in the automotive industry. At the same time, there exist several challenges such as the limited driving range of EVs that need to be addressed and solved properly in order not to become obstacles in this potential paradigm change in the automotive industry.

6. REFERENCES

[1] M. Brandl et al. Batteries and battery management systems for electric vehicles. In *Proc. of DATE*, pages 971 –976, 2012.

[2] M. Kauer, S. Narayanaswami, S. Steinhorst, M. Lukasiewycz, S. Chakraborty, and L. Hedrich. Modular system-level architecture for concurrent cell balancing. In *Proc. of DAC 2013*, 2013.

[3] M. Brandl et al. Batteries and battery management systems for electric vehicles. In *Proc. of DATE*, pages 971 –976, 2012.

[4] I. Boldea. Control issues in adjustable speed drives. *IEEE Industrial Electronics Magazine*, 2(3):32–50, Sep 2008.

[5] K. Zhou and D. Wang. Relationship between space-vector modulation and three-phase carrier-based PWM: A comprehensive analysis. *IEEE Transactions on Industrial Electronics*, 49(1):186–196, Feb 2002.

[6] H. Yoo, S. Sul, Y. Park, and J. Jeong. System integration and power-flow management for a series hybrid electric vehicle using supercapacitors and batteries. *IEEE Transactions on Industry Applications*, 44(1):108 –114, 2008.

[7] A. Schmidt, A.K. Dey, A. L. Kun, and W. Spiessl. Automotive user interfaces: human computer interaction in the car. In *Ext. Abstracts CHI*, pages 3177–3180, 2010.

[8] C. C. Chan. The state of the art of electric, hybrid, and fuel cell vehicles. *Proceedings of the IEEE*, 95(4):704–718, 2007.

[9] E.a. Bretz. By-wire cars turn the corner. *IEEE Spectrum*, 38(4):68–73, 2001.

[10] R. Isermann, R. Schwarz, and S. Stolzl. Fault-tolerant drive-by-wire systems. *IEEE Control Systems*, 22(5):64 – 81, October 2002.

[11] P. Peti, R. Obermaisser, F. Tagliabo, a. Marino, and S. Cerchio. An integrated architecture for future car generations. In *Proc. of ISORC*, pages 2–13, 2005.

[12] FlexRay Consortium. FlexRay communications systems - protocol specification. http://www.flexray.com.

[13] Broadcom. BroadR-Reach Ethernet hardware, 2011.

[14] H.T. Lim, L. Volker, and D. Herrscher. Challenges in a future IP/Ethernet-based in-car network for real-time applications. In *Proc. of DAC*, pages 7–12, 2011.

[15] IEEE standard for a precision clock synchronization protocol for networked measurement and control systems, 2008.

[16] T.J. Giuli, D. Watson, and K.V. Prasad. The last inch at 70 miles per hour. *IEEE Pervasive Computing*, 5(4):20–27, October 2006.

[17] M. Lukasiewycz, R. Schneider, D. Goswami, and S. Chakraborty. Modular scheduling of distributed heterogeneous time-triggered automotive systems. In *Proc. of ASP-DAC*, pages 665–670, 2012.

[18] F. Sagstetter, M. Lukasiewycz, and S. Chakraborty. Schedule integration for time-triggered systems. In *Proc. of ASP-DAC*, 2013.

[19] J. Wolf, M. Gerdes, F. Kluge, S. Uhrig, J. Mische, S. Metzlaff, C. Rochange, H. Casset', P. Sainrat, and T. Ungerer. RTOS support for parallel execution of hard real-time applications on the MERASA multi-core processor. In *Proc. of ISORC*, pages 193–201, 2010.

[20] N. Navet, A. Monot, B. Bavoux, and F. Simonot-Lion. Multi-source and multicore automotive ecus-os protection mechanisms and scheduling. In *Proc. of ISIE*, pages 3734–3741, 2010.

[21] C. Aussaguès, D. Chabrol, V. David, D. Roux, N. Willey, A. Tournadre, and M. Graniou. PharOS, a multicore os ready for safety-related automotive systems: Results and future prospects. In *Proc. of ERTS*, 2010.

[22] A. Baumann, P. Barham, P.E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schüpbach, and A. Singhania. The multikernel: a new OS architecture for scalable multicore systems. In *Proc. of SOSP*, pages 29–44, 2009.

[23] B. Bilgic, B.K.P. Horn, and I. Masaki. Fast human detection with cascaded ensembles on the GPU. In *Proc. of IV*, pages 325 –332, June 2010.

[24] F. Jianbin Fang, A.L. Varbanescu, and H. Sips. A comprehensive performance comparison of cuda and opencl. In *Proc. of ICPP*, pages 216 –225, 2011.

[25] S. Shreejith, S. A. Fahmy, and M. Lukasiewycz. Reconfigurable computing in next-generation automotive networks. *IEEE Embedded Systems Letters*, 5(1):12–15, 2013.

[26] S. Shreejith, K. Vipin, S. A. Fahmy, and M. Lukasiewycz. An approach for redundancy in FlexRay networks using FPGA partial reconfiguration. In *Proc. of DATE*, 2013.

[27] R. Charette. This car runs on code. *IEEE Spectrum*, 46(3):3, 2009.

[28] R. Alur and G. Weiss. Regular specifications of resource requirements for embedded control software. In *Proc. of RTAS*, 2008.

[29] M. Kauer, S. Steinhorst, D. Goswami, R. Schneider, M. Lukasiewycz, and S. Chakraborty. Formal verification of distributed controllers using time-stamped event count automata. In *Proc. of ASP-DAC*, 2013.

[30] H. Theiling, C. Ferdinand, and R. Wilhelm. Fast and precise WCET prediction by separated cache and path analyses. *Journal of Real-Time Systems*, 18:157–179, 1999.

[31] N. Singh, T. Mitra, and A. Roychoudhury. Accurate estimation of cache-related preemption delay. In *Proc. of CODES+ISSS*, pages 201–206, 2003.

[32] I. Liu, J. Reineke, D. Broman, M. Zimmer, and E. Lee. A PRET microarchitecture implementation with repeatable timing and competitive performance. In *Proc. of ICCD*, 2012.

[33] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *Proc. of Usenix Security*, 2011.

[34] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. Mccoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. Experimental security analysis of a modern automobile. In *IEEE Symposium on Security and Privacy*, pages 447–462, 2010.

[35] F. Sagstetter et al. Security challenges in automotive hardware/software architecture design. In *Proc. of DATE*, 2013.

[36] R. Falk and S. Fries. Electric vehicle charging infrastructure - security considerations and approaches. In *Proc. of INTERNET*, pages 58–64, 2012.

[37] Internet Engineering Task Force. RFC 4301 security architecture for the internet protocol, 2005.