# Domain Controlled Architecture
## *A New Approach for Large Scale Software Integrated Automotive Systems*

Dominik Reinhardt[1] and Markus Kucera[2]
[1]*BMW AG, Munich, Germany*
[2]*University of Applied Science Regensburg, Regensburg, Germany*
*Dominik.Reinhardt@bmw.de, Markus.Kucera@hs-regensburg.de*

Keywords: Large Scale Software Integration, LSSI, Automotive Real Time, Multi-core, Many-core, Embedded Automotive Software Architecture.

Abstract: Electric and electronic functionalities increase exponentially in every mobility domain. The automotive industry is confronted with a rising system complexity and several restricting requirements and standards (like AUTOSAR), in particular to design embedded software for electronic control units. To stand against rampant functionalities software units could be restructured according to their affiliation and should not be attached to a certain place. This can be effected by integration on single controllers. On the one hand the system wide amount of hardware controllers could such be limited. On the other hand the workload for integration CPUs will rise. To support this paradigm, multi-core systems can provide enough processing power in an efficient way. This paper shows a first approach to combine automotive functionality on such a single controller.

## 1 INTRODUCTION

E/E (*Electric/Electronic*) systems represent an important part in premium vehicles and are the major contributor in creating added value for OEMs (*Original Equipment Manufacturer*). The amount of vehicle functions increases exponentially and functionalities get more and more complex. This trend is tightened by additional requirements like lightweight design and construction, energy efficiency, functional safety driven by the new ISO standard for road vehicles, and last but not least by the development standard (AUTOSAR Administration, 2012) for automotive software architecture AUTOSAR (*AUTomotive Open System ARchitecture*).

Facing increasing software workload and rising need for SW/HW robustness, there is a need for more powerful hardware resources. Furthermore, the number of ECU (*Electronic Control Unit*) devices must be reduced (Gut et al., 2012). To save fuel and extend the crusing range of electric cars, the power consumption of embedded systems and in detail of electronic semiconductors shall be minimized (Schöttle, 2012), (Barthels et al., 2012). The actual state-of-the-art in science and technology (Arbeitskreis-Multicore, 2011) shows that multi-core technology can solve the problem in a more efficient way. Ten years ago Gordon Moore's Law (Moore, 1965) was contested in the area of consumer electronics and multi-core processors were published for general public on the market. High performance computing systems or even mobile devices like smart phones cannot get along without this technology to supply the requests of processing power. This paradigm has reached the automotive industry eventually (Schneider et al., 2010) and (Monot et al., 2010).

Dual- or quad-core systems are already available on the market for vehicle manufacturers. In the future this trend will be continued and the amount of cores will rise. Many-core controllers could revolutionize current development standards and strategies. For automotive systems it is not yet completely solved how to obtain more and more speedup with a rising number of cores and in comparison with small but computationally intensive, high interconnected applications. For every single vehicle function built for domains like chassis or powertrain it is to clarify, how to segregate them from each other and how to achieve given constraints in time or space. According to Amdahl's (Amdahl, 1967) and Gustafson's Law (Gustafson, 1988), to optimize the systems speedup, software modules with less dependencies to units located on other cores can be parallelized more efficiently on probably more than 16 cores. This fact will challenge automotive software engineers to review former development techniques and design rules

in order to create loose coupled systems and still fulfill the required functionality at once.

# 2 TODAY'S VEHICLE COMMUNICATION ARCHITECTURES

To improve an automotive E/E system to deal with gaining vehicle functionalities it is modified by a top down approach. At first the vehicle electrical system architecture must be fitted to be flexible enough to get along with future upgrades and modifications. Therefore, the integration of vehicle functions to ECUs has to be reconsidered. In the following the actual decentralized communication network of BMW is compared with a future centralized architecture approach.

## 2.1 Present Decentralized Vehicle System Architecture

Actual embedded vehicle functions are shared between up to 70 electronic control units and are connected over several buses (see Figure 1). Evoked by this fact, electrical vehicle functions, which are accounted on the OEM side, are developed by several suppliers. Certainly it is easier to separate these problems to a manageable amount of ECUs, at least to solve the final problem to integrate the system onto an ECU. In present the main advantage not to overload the systems functionality and having a structured software architecture overview causes less prices for software developing and single hardware platforms. According to the shared geometric assembly of ECUs inside of the car transmission paths or rather the cable runs to sensor and actuator components are shortened. In terms of lifetime, it is easier to exchange damaged parts if E/E components are lightweight and do not include most of the vehicles software functionality. In detail, a shared HW/SW architecture gives persons in charge a better control of their encapsulated system and enhances robustness for installing later software updates. At the end, the system is more scalable and could be flexibly adjusted from start until end of production of a vehicle product line.

On the one hand, decentralized systems provide a high amount of flexibility related to loose coupled hardware components networked within the vehicle bus architecture. On the other hand, this flexibility increases the system price due to a lot of overhead. Because of the rising communication demand between software components, some vehicle buses could get overloaded and would not be able to stand the volume of signals and information any more. Due to the system complexity, there are many masked problems like interference with communications between ECUs and their integrated software components. If no common interfaces exist, intra ECU communication will cause a lot of problems.

## 2.2 BMW's Approach for Future Centralized System Architecture Designs

Centralized car functions implicate a single source of information which can be processed internally by ECU and advanced function comprehensive calculations. Functions shall be clustered in a single place. A decreased number of ECUs implies the reduce of internetworking complexity and saves costs for car wiring harness which is probably one of the most determining factors. Due to the joint software integration on an ECU, timing constraints can be achieved more easily in matters of integration time, if no slowing transportation layer must be passed through. Depending of ECU capability and system architecture, perceived events, like video camera signals, can be evaluated much faster if the processing of signal input, decision and output is located in a single system. Nevertheless, the physical connection to most of the actuators and sensors still has to exist. Relocating the software responsibility on single ECUs is associated with additional disadvantages. An obvious problem is the need of a capable hardware controller which can deal with the rising workload. It implies acquiring expensive hardware platforms. In the case of damage, a valuable component must be replaced. This impedes the handling of oversized systems in question of scalability and flexibility and hence poses further disadvantages. If applicable from a safety perspective it means a loss of redundancy which must be solved compliant to automotive standards.

For common hardware and software systems there are many terms and definitions for the method to consciously centralize components (*e.g. High Integration, Highly Integrated Software, Very Large Scale Integration, High Density Integration*). However, each of those do not fit to the presented requirements. Therefore we introduce the term *'**L**arge **S**cale **S**oftware **I**ntegration'* (abbr. LSSI), according to the already common used term Very Large Scale Integration. Enriched by the word *'software'*, it emphasizes our site of operation. A LSSI system centralizes several high integrity vehicle software components onto a single ECU. To avoid interference with others, software components have to be isolated.

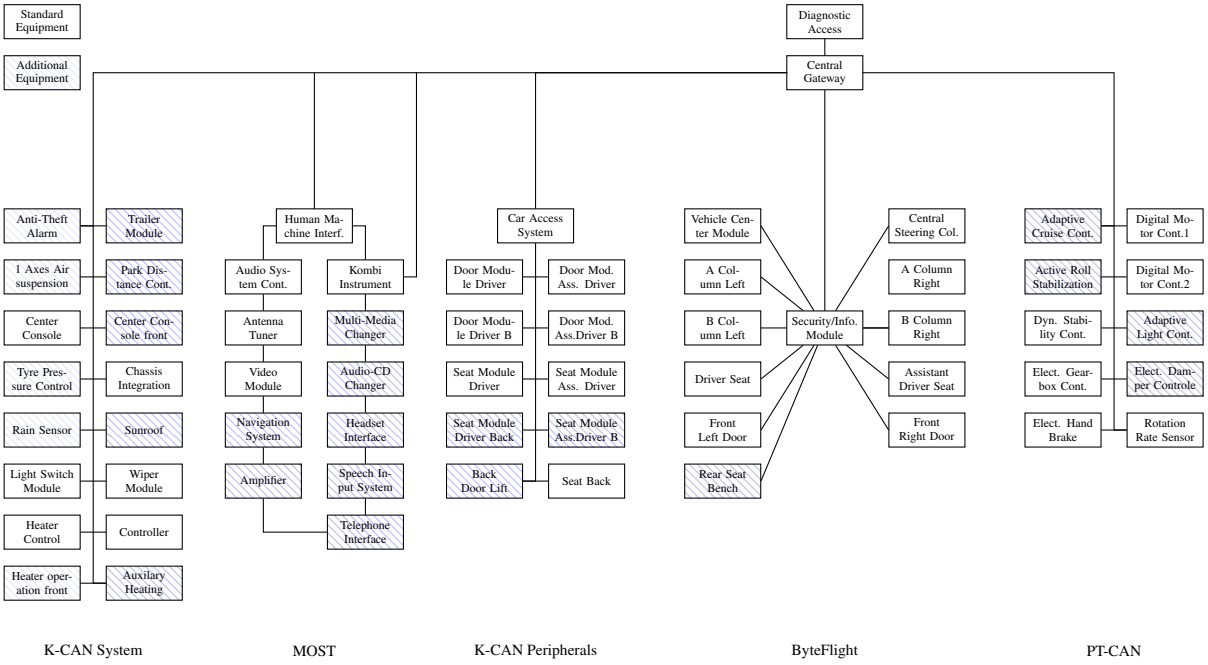


Figure 1: Decentralized ECUs of a example vehicle electrical system architecture (Barthels et al., 2012), (Michel et al., 2012).

# 3 POSSIBLE REALIZATION OF A DOMAIN CONTROLLED AUTOMOTIVE SYSTEM USING MULTI-CORE CPUS

Facing the consistently growing amount of car features, which are attended by rising software complexity, vehicle electrical systems need to be restructured and more flexible architecture patterns must be established. Until now vehicle functions have been dedicated to a certain controller.

In an *domain controlled architecture*, ECUs and in special corresponding E/E vehicle functionality shall be reordered in a domain specific manner. Typical domains are Infotainment, Chassis, Powertrain and Body and Comfort (Michel et al., 2012). To reach a higher level of system scalability, an additional abstraction layer, in the form of capable server ECUs called *domain controller* (compare with Figure 2), is established for each special domain. Every domain controller can control several field bus systems. To save energy (Schöttle, 2012), connected network clusters can be deactivated individually to allow a partial bus operation (Fuchs et al., 2010). Furthermore, domain controllers are used as integration platform for system software components. Applications, which must always be available during car usage are integrated on these ECUs. In general domain controllers contain OEMs specific software components. Beside others, these servers act like routers and are responsible to regulate the activity for their affiliated sub domain ECUs. These slave controllers in contrast are light and flexible for operation. Typically commodity functions like the airbag system will be integrated on them, which imply a more flexible partitioning in different locations.

## 3.1 Domain Driven Partitioning of Software Components within the Vehicle Communication Network

The software components unity is analyzed from a logical side of view to reorder E/E functions within the vehicle electrical system (see Figure 2). Applications which were sorted according to their specific domain are sliced and integrated on a domain or sub controller. Thereby located units, which are correlated to their neighbors, form a closed composition. The path of information between software components shall be as condensed as short as possible. This paradigm brings new requirements for software architectures and interface designs to dissolve strong coupled applications and system functionality. Measured by their lines of code, automotive E/E functions are typically small applications. Therefore, it is hard to split associated software components and share them

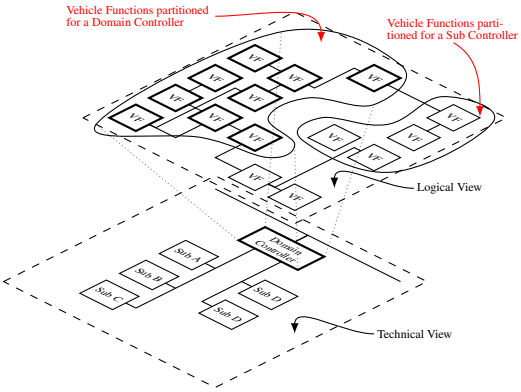


Figure 2: Logical segmentation of an vehicle electrical system structure.

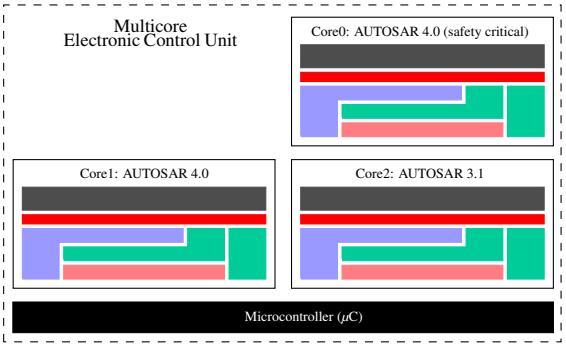


Figure 3: Possible core partitioning adapted for independent ECU projects.

on several ECUs. To gain the best SpeedUp in spite of achieving all applications requirements, the challenge therein is to develop the optimal allocation for software components. It has to be balanced if vehicle E/E software functions are to be entirely integrated on a domain controller, or on a sub domain ECU, or are split up to work partly on each of them.

The architectural design has to be flexible in such a way that E/E feature's expansion stage is alterable to the related car. To guarantee a flexible operation, software components are developed in a loose coupled way to be mixable and individually integrated on a single ECU. The isolation of these components from each other will pose another challenge if they are classified as safety critical. At all points, it must be secured within large scale software integrated systems, that no installed application interferes with others during runtime and manipulates not related data. Given that a lot of software components will be executed on a domain controller, the workload is high, and ECUs with sufficient performance are necessary. Besides, to supply the required computing capacity by using a power efficient hardware we propose to use multi-core technology. To waste no additional energy, efficient algorithms and protocols must be developed. The calculation capacity of on-chip integrated cores shall not be utilized of inter core communication or task synchronization overhead.

## 3.2 Proposed Static ECU Allocation Integrated on Closed Cores

To deal with a multi core system, nearly the whole integrated software must be completely restructured and optimized for parallel processing (Akhter and Roberts, 2006). In a large scale software integrated ECU, composed of multiple self-contained software projects, there will be typically loose coupled software units. Beside components with high dependency to related functionalities (like former master and slave architecture constructs), lots of car software components exist with less or even no connection to their neighbors. Rearranging and integrating these units on an ECU is not too challenging from an architectural perspective. Because of their independence to each other, such an architecture pattern represents an embarrassingly parallel problem (Foster, 1995). If heterogeneous software components run on an embedded multi-core system, shared on several cores, they probably offer the best benefit because there are no sequential parts to execute. To integrate non related systems on different cores dissolves the former bin packaging problem to efficiently migrate as much software components as possible on a single core machine and to deal with hundreds of synchronous and asynchronous tasks concurrently. To realize decoupled system behavior, hardware resources are split up between any system and core (Hilbrich, 2012), (Brewerton et al., 2012).

Efficient ECU partitioning could give an impulse for future research projects. Advancing further migration strategies on the one hand and support for functional safety for orthogonal automotive software systems on the other hand. As pictured in Figure 3, even software projects using different AUTOSAR versions or Automotive Safety Integrity Levels (International Organization for Standardization, 2011) could be integrated. To realize an independent operation, hardware resources must be separated. The Memory Management Unit (abbr. MMU) or Memory Protection Unit (abbr. MPU) for instance must be configured correctly to control the assigned Random Access Memory of any core. Access problems will occur if interfaces, like communication controllers, are used simultaneously and therefore must be reserved for usage exclusively.

Actually there are already isolated ECUs which comprise a massive amount of car functionalities. An

example is the engine control unit which includes up to 300 software components. It deals with lots of sensor signals and is highly interconnected with other controllers. However, on such a centralized software system, some integrated functionalities are highly involved with each other and are not suitable for further relocation. Inter core communication will cause a lot of overhead between cores or even between ECUs if higher synchronization mechanisms have to handle data exchange between every component. Because applications shall be shared individually between ECUs and cores this scenario could be an origin to fulfill the postulated requirements of a future domain controller.

### 3.3 A Standardized Middleware for Domain Controllers

Based on a technical side of view, every migrated single core software project and their software components run encapsulated on each core. Thus, the appropriate Run Time Environment Layer (abbr. RTE) must be located independently on every core, as well. In practice, the upcoming bottlenecks are hardware resources (e.g. memory and communication controllers in all kinds), which have to be allocated simultaneously. These resources shall be under control of centralized basic software modules, like the operating system, which handles each access. According to the ISO Standard 26262 (International Organization for Standardization, 2011), for safety critical vehicle applications, which are specified with an ASIL, there must be arrangements made to avoid any interference caused by other components during runtime. The approach to run single core applications shared individually on multi-core core systems is covered from the AUTOSAR 4.0 standard and is already included in its basic software.

The AUTOSAR Operating System (abbr. OS) is enriched by the new IOC (*Inter-OS-Application Communication*), which allows information exchange between OS-Applications (AUTOSAR, 2011). Therefore all information is passed through by the IOC (see fig. 4). For software components, running on application layer level, this module is not directly accessible and is decoupled by the RTE (AUTOSAR Administration, 2011). If information transfer between software components shared on several cores is needed, this feature is a possible approach for it. As proposed in (Scheidemann et al., 2010a), (Scheidemann et al., 2010b), the IOC enables the communication channel between separated memory partitions, surrounding software component groups which assure freedom from interference related on memory faults. Thus, if the MPU or MMU is configured correctly, software components cannot modify memory of another partition. Considering a multi-core scenario, the IOC facilitates communication over core borders. However these mechanisms have an influence on the whole system performance and availability. Especially for safety critical functions, mixed integrated on a multicore controller with hard timing constraints, provisions like appropriate scheduling algorithms have to be arranged at software integration phase (Schmidt et al., 2012), in order to guarantee the predicted execution time of each software component.

## 4 CONCLUSIONS AND FUTURE WORK

In the future, Tier-2 and also Tier-1 suppliers will mainly supply and develop multi-core platforms for OEMs. Car manufactures are confronted with the actual hardware controller designs and have to deal with the problem to parallelize their software and to reduce complexity. More and more software components must be integrated on less automotive ECUs. If that assumption applies, large scale software integration techniques could be the enabler for upcoming automotive E/E software architectures. As mentioned before, these approaches will come along with multi-core hardware platforms, derived from consumer electronics.

This paper proposes a domain controlled architecture in order to exhibit a possible approach for future vehicle electrical system. Furthermore, it is demonstrated that there is a need for further research, to transfer multi-core technologies to the automotive industry, by using the examples of consumer electronics. Nevertheless, considering the different system and safety requirements inflicted on mobility domains, typical software parallelization scenarios, as used for personal computer software, are not entirely suitable. To make embedded multi-core systems ready for operation, the ARAMiS (*Automotive, Railway and Avionics Multicore Systems*[1]) project has been started in beginning of 2012. It constitutes a national research project which aims at enabling multi-core systems for automotive, avionic and railway transportations domains, to smooth the way for *Cyber Phyical Systems* (Geisberger and Broy, 2012) in the future.

[1]Project website: http://www.projekt-aramis.de

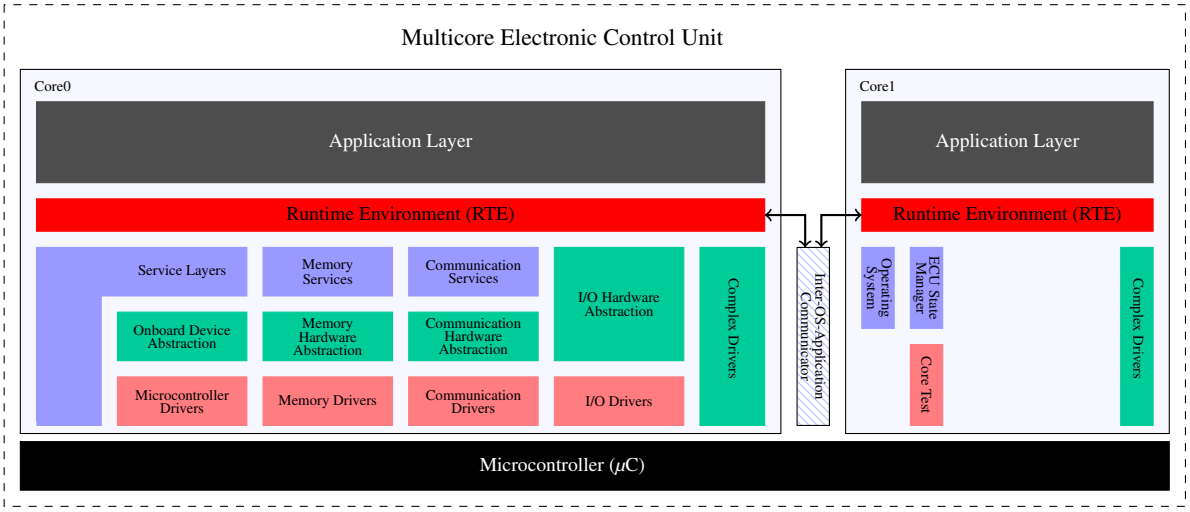


Figure 4: Inter core communication, using Inter-OS-Application Communicator.

# REFERENCES


Akhter, S. and Roberts, J. (2006). *Multi-Core Programming: Increasing Performance Through Software Multi-threading*. Books by engineers for engineers. Intel Press.

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA. ACM.

Arbeitskreis-Multicore (2011). Relevanz eines Multicore-Ökosystems für künftige Embedded Systems. Technical report, BICCnet.

AUTOSAR (2011). Specification of Operating System. Version 4.0.3 - Final.

AUTOSAR Administration (2011). Layered Software Architecture - R4.0. Version 4.0.3 - Final.

AUTOSAR Administration (2012). AUTomotive Open System ARchitecture. http://www.autosar.org.

Barthels, A., Michel, H.-U., and Walla, G. (2012). Jedes Watt zählt - Intelligentes Energie- und Leistungs-Management für die Autos von morgen. *Elektronik automotive*, 05:24–28.

Brewerton, S. P., Willey, N., Gandhi, S., Rosenthal, T., Stellwag, C., and Lemerre, M. (2012). Demonstration of Automotive Steering Column Lock using Multicore AutoSAR® Operating System. In *SAE International*.

Foster, I. T. (1995). *Designing and building parallel programs - concepts and tools for parallel software engineering*. Addison-Wesley.

Fuchs, M., Scheer, P., and Grzemba, A. (2010). Selektiver Teilnetzbetrieb im Fahrzeug. In *AmE 2010 - Automotive meets Electronics*. VDE-Verlag.

Geisberger, E. and Broy, M. (2012). *agendaCPS: Integrierte Forschungsagenda Cyber-Physical Systems*. acatech Studie. Springer, Berlin.

Gustafson, J. L. (1988). Reevaluating amdahl's law. *Commun. ACM*, 31(5):532–533.

Gut, G., Allmann, C., Schurius, M., and Schmidt, K. (2012). Reduction of Electronic Control Units in Electric Vehicles Using Multicore Technology. In *Multicore Software Engineering, Performance, and Tools*, volume 7303, pages 90–93. Springer.

Hilbrich, R. (2012). How to safely integrate multiple applications on embedded many-core systems by applying the "correctness by construction" principle. *Adv. Soft. Eng.*, 2012:3:1–3:14.

International Organization for Standardization (2011). ISO 26262 Road vehicles - Functional safety - Part 1-10.

Michel, H.-U., Kaule, D., and Salfer, M. (2012). Vision einer intelligenten Vernetzung. *Elektronik automotive*, 4:28–32.

Monot, A., Navet, N., Bavoux, B., and Simonot-Lion, F. (2010). Multicore scheduling in automotive ECUs. In *Embedded Real Time Software and Systems - ERTSS 2010*, Toulouse, France.

Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117.

Scheidemann, K., Knapp, M., and Stellwag, C. (2010a). Load Balancing in AUTOSAR-Multicore-Systemen Teil 1. *elektroniknet*, 1/2:22–25.

Scheidemann, K., Knapp, M., and Stellwag, C. (2010b). Load Balancing in AUTOSAR-Multicore-Systemen Teil 2. *elektroniknet*, 3:21–25.

Schmidt, K., Buhlmann, M., Ficek, C., and Richter, K. (2012). Design patterns for highly integrated ECUs with various ASIL levels. *ATZ elektronik*, 01/2012:22–26.

Schneider, J., Bohn, M., and Rößger, R. (2010). Migration of Automotive Real-Time Software to Multicore Systems. In *Proceedings Work-In-Progress Session of the 22th Euromicro Conference on Real-Time Systems*, ECRTS'10, pages 37–40.

Schöttle, M. (2012). Wir wollen den elektrischen Stromverbrauch halbieren. *ATZ elektronik*, 01/2012:16–19. Springer Automotive Media GmbH (2012).