Recent Advances in In-vehicle embedded systems

Jiao Yu and Bogdan M. Wilamowski,
Department of Electrical and Computer Engineering, Auburn University, Auburn, USA
Email: {jzy0012, wilambm}@auburn.edu

***Abstract*— The number of computer based functions embedded in vehicles has increased significantly in the past two decades. An in-vehicle embedded electronic architecture is a complex distributed system; the development of which is a cooperative work involving different manufacturers and suppliers. There are several key demands in the development process, such as safety requirements, real-time assessment, schedulability, composability, etc. Intensive research is being conducted to address these issues. This paper reviews recent technology advances in relevant aspects and covers a range of topics highlighted above.**

***Index Terms*—In-vehicle embedded electronic architecture, FPGA, real-time assessment, composability**

I. Introduction

An embedded system is typically a micro-computer system with one or few dedicated functions, usually with real-time computation constraints. Different from a general purpose personal computer, it is often embedded as part of a complete device. The usage of embedded systems is so widespread today, e.g. smart phones, programmable systems on chip (SoC), smart sensors, etc. These types of embedded systems include microprocessors, DSPs (digital signal processors), ASICs (application-specific integrated circuits), and FPGAs (field-programmable gate arrays).

Nowadays, considering the high competition in market, cost and time-to-market of the development process of embedded systems must be minimized, while the customers' demand for computation power and speed is ever increasing. Other factors such as the ease of development, power consumption, and sophistication of algorithms also need attention from embedded system developers. Generally, designers have the choice of two main families of digital device technologies [1]: The first family consists of microcontrollers and DSPs, based on a pure software platform. The typical constitution of this family is a performing microprocessor core along with several peripherals. The alternative family is FPGAs, based on configurable hardware elementary cells, and interconnections. End users build specific hardware architecture to meet their requirements.

Compared with software based digital devices, FPGAs gain great advantages in high-speed demanding applications. In safety critical industries such as automotive and aircraft manufacturing, it imposes significant challenges for digital electronics [2]. Thus, paper [3] and [4] propose several state-of-art technologies to reinforce the reliability of FPGA controllers.

Power consumption becomes a key design concern for handheld embedded systems [5], [6]. For the subject of power consumption reduction, several studies have been conducted. In [7], a thorough discussion of the source of power consumption and schemes for its minimization are presented. Paper [8], [9] propose methods to use FPGAs to manage the communication of distributed applications via Ethernet protocol.

It is worth mentioning that an embedded controller based on Artificial Intelligence is becoming more and more popular; intensive research in this domain has evolved. For example, [9] proposes a lightweight method to implement the neuron-by-neuron process [35] on embedded systems to correct the nonlinearities of many sensors and devices. [10], [11] present methods to implement a Fuzzy Logic Controller on reconfigurable FPGA systems. [12] implements Gauss-Newton and particle swarm optimization algorithms to estimate sensor-node physical-position.

The recent two decades has witnessed a trend in the automotive industry---a rapid growth in the percentage of cost of embedded electronic systems, more precisely the software components. As shown from [13] in 2006, the electronic embedded system constituted at least 25% of the total cost of a car and more than 35% for a high-end model. Top-line cars today may contain up to 100 ECUs (Electronic Control Unit). Each controls one or more of the electrical systems or subsystems in a motor vehicle networked over standard communication buses. Local area networks such as LIN, CAN, FlexRay, Most and IDB-1394 are developed as such links. Considering the increase of complexity of embedded electronic architecture, the development of it has to integrate different hardware and software units provided by different vendors, which raises the question of "composability".

This paper will review recent advances in the relevant technologies in the subsequent sections, including temporal isolation of software components, probabilistic approach of latency computation, and upgrading from single core ECUs to multicore ECUs.

II. Temporal interoperability

The development of embedded automotive software plays a key role in pushing the improvement of the automotive industry in terms of safety, cost, performance, and comfort. Several new functions are made possible with reasonable costs thanks to the development of software technology, such as multimedia and telematics in vehicles. Electronic control units (ECUs) are the specialized programmable hardware platforms which automotive software runs on. ECUs have a real-time operating system and domain specific basic software, e.g. for engine control. Different software components are potentially developed by different OEMs (Original Equipment Manufacturer, or carmaker) and several Tier 1 suppliers. This raises the question of how to integrate all the software components in an efficient and secure way, in particular for safety-critical functions.

Modeling languages are one way to reach this goal, e.g. AIL_transport, EAST-ADL, EAST-ADL2. These modeling languages are capable of representing the system at all its design steps and common to all the actors involved in the design process [13]. EAST-ADL is a domain specific language using meta-modeling constructs such as classes, attributes, and relationships [14]. EAST-ADL can be used to model the structural aspects of automotive elements and describe the dependencies between them. The EAST-ADL scope includes early analysis via functional design to the implementation perspective and back to integration and acceptance testing on vehicle level.

The international program AUTOSAR[15] also addresses the same target. Automobile manufacturers, suppliers, and tool developers jointly develop an open and standardized automotive software architecture--AUTOSAR(AUTomotive Open System ARchitecture), with the objective of creating and establishing open standards for automotive E/E (Electrics/Electronics) architectures that will provide a basic infrastructure to assist with developing vehicular software, user interfaces, and management for all application domains. This includes the standardization of basic systems functions, scalability to different vehicle and platform variants, transferability throughout the network, integration from multiple suppliers, maintainability throughout the entire product life-cycle, and software updates and upgrades over the vehicle's lifetime as some of the key goals[16].

Specification of *components* that implement complicated and distributed functions in an in-vehicle embedded system is allowed in AUTOSAR. Interfaces between different components are formally defined, and thus facilitate functional integration of components. *Software components* are atomic (i.e., not distributed) pieces of functionality. Components are composed of software components and can be deployed to the physical nodes in the vehicle system. Thanks to components and software components, vehicle manufacturers are allowed to structure the functions of a vehicle and to get the efforts required to implement the functions partitioned.

However, vehicle functions often have stringent real-time requirements, which means that only functional correctness cannot guarantee their correct behavior, but also their temporal correctness needs to be taken into account. That is, the right values (or right actions) have to be delivered (taken) at the right time [17]. Unexpected problems may occur when integrating different individually developed functions on a single ECU. Due to interference from other software components, functions that work nicely when running on an ECU exclusively may exhibit incorrect behavior when they have to share an ECU with other functions. The integration processes are in the late phases of the whole development process for a vehicle; any incorrectness in this phase may be very costly.

To tackle this late integration problem, [17] proposes the use of server-based scheduling techniques in AUTOSAR. As stated by [17], the *server technology* is provided by a Hierarchical Scheduling Framework (HSF), implemented as a layer between the AUTOSAR OS and the AUTOSAR Runtime Environment (RTE). CPU allocates a part of its total capacity to a server, which is an operating-system level object. By mapping a software component or set of components to one server, access to its allocated share of computing resources can be guaranteed without worries about the need of resources of any other functions in the system. Via servers, functions' computational resource will remain the same even if other components or functions are added or removed from the system. Thus, "temporal firewalls" can be implemented by servers between functions of components. In this way the temporal correctness of a software function can be preserved no matter in what context it is integrated [18].

III. Stochastic Analysis of End to End Latency

Designers need to define, evaluate, and choose car electronic architectures years in advance, but at that time the functions they will support are not completely known. This stage will have significant influence on the future cost, performance, and extensibility of vehicles. Many automotive applications, including most of those developed for active safety and chassis systems, must comply with hard real-time deadlines and are also sensitive to the average latency of the end-to-end computations from sensors to actuators. A characterization of the timing behavior of functions is used to estimate the quality of an architecture configuration in the early stages of architecture selection [19].

For a complex function distributed onto several ECUs communicating through a network, to compute a deterministic upper bound for its end-to-end response time can be difficult and even not possible. Therefore [20] proposes a probabilistic approach of getting valuable estimations of such information in the early stage of the architecture design. Task response times, message response times, and communication delays are a few key factors affecting end-to-end latencies for computations that are distributed over several ECUs and communicating via CAN buses. Worst case analysis based on schedulability theory allows computing the contribution of tasks and messages [21] to end-to-end latencies and provides the architecture designer with a set of values (one for each end-to-end path) on which he/she can check correctness of an architecture solution. However, a worst case analysis alone is not sufficient for designers; it should be complemented by probabilistic analysis in order to avoid wastefully conservative design.

As summarized by [19], probabilistic analysis of priority-scheduled systems has been addressed in the past. In [22] for each job released in the busy interval, the probability of deadline misses is computed, and the maximum is chosen as an upper bound on the probability of deadline misses for the corresponding task. [20], [23] present simulation-based analysis and stochastic methods to compute the probability density functions (pdfs) or probability mass functions (pmfs) in the discrete case of the response times of tasks scheduled on single or multiprocessor platforms. In [24], for messages scheduled on a CAN bus, the concept of worst case deadline failure probability (WCDFP) because of transmission errors is introduced. WCDFPs are computed with respect to the *critical instant*, i.e., the worst case response time scenario. The worst case response time analysis gets extended in [25] using random message transmission times that take into account the probability of a given number of stuff bits but still in the worst case scenario. Worst case analysis of task and message response times in priority-based systems addressed independent periodic tasks [26], tasks with offsets and jitter [27] and OSEK systems [28]. In [29], Davis *et al.* discuss scheduling and response time analysis of CAN messages.

IV. Multicore ECUs

As the demand for computing power is becoming higher and higher, multicore ECUs are getting popular in car electronic architecture and promise to be a solution for the current situation of over-numbered single core ECUs[30]. What's more, several new attracting features such as higher levels of parallelism are brought to the designers by multicore ECUs, which ease the respect of the safety requirements such as the ISO 26262 and the implementation of other automotive use-cases. Of course the drawbacks of these new features including more complexity in the design, development, and verification of the software applications also come along. Hence, OEMs and suppliers will require new tools and methodologies for deployment and validation. [31]

With the growing popularity of multicore ECUs used in automotive real-time environments, scheduling and synchronization analysis of these platforms receive increasing attention. With multicore ECUs, it becomes possible to integrate previously separated functionality for body electronics or sensor fusion onto a single unit and to execute complex computations over multiple cores in parallel. With multiple CPUs, an ECU is turned into a highly integrated "networked system" microcosm, in which there exist complex interdependencies among those CPUs due to the use of shared resources even in partitioned scheduling [32]. To achieve predictable performance, resource arbitration protocols need to be developed and have been studied in literature.

A high percentage of current major ECU software has been developed for single-core ECU systems. Real-time operating systems based on the OSEK/VDX specification are developed to arbitrate multiple functions executed on a single-core ECU. In these OS, priority based scheduling is performed and resource synchronization is achieved via locks administered according to the priority ceiling protocol (PCP). The more recent AUTOSAR OS specifications also adopt this lock based method. In this trend of upgrading to multicore ECUs, how to reuse the previous software generations and configurations becomes a major concern of automotive suppliers and manufacturers, as property changes can be costly involving many different departments and companies. In the past, this has often implied that nonoptimal design choices are accepted for the sake of compatibility, for example, by respecting legacy priority assignments and mapping decisions [33].

Multiprocessor priority ceiling protocol (MPCP) provides a solution for the problem of compatibility with previous single core systems. MPCP can be thought of as an extension of PCP because it reduces to PCP when used on a single processor [34].

V. Conclusion

Embedded systems, especially in-vehicle embedded systems, are ubiquitously related to our everyday life. The development of embedded systems greatly facilitates the comfort of people's life, changes our view of things, and has a significant impact on society. On the other hand, even though embedded systems technologies are becoming more and more mature, currently there still exist many challenges in technique and will be more with the ever growing speed and reliability demand from market. We expect more enlightening researches in this area.

References

[1] E. Monmasson, L. Idkhajine, M.N. Cirstea, I. Bahri, A. Tisan, M.W. Naouar, "FPGAs in Industrial Control Applications ," *IEEE Trans. on Industrial Informatics,* vol. 7, no. 2, pp. 224 - 243 , May 2011.

[2] J. Munoz-Castaner, R. Asorey-Cacheda, F. J. Gil-Castineira, F. J. Gonzalez-Castano, and P. S. Rodriguez-Hernandez, "A review of aeronautical electronics and its parallelism with automotive electronics," *IEEE Trans. Ind. Electron.*, vol. 54, no. 99, p. 1, Apr. 2010.

[3] P. Conmy and I. Bate, "Component-based safety analysis of FPGAs," *IEEE Trans. Ind. Informat.*, vol. 6, no. 2, p. 195, May 2010.

[4] F. Salewski and S. Kowalewski, "Hardware/software design considerations for automotive embedded systems," *IEEE Trans. Ind. Informat.*, vol. 4, no. 3, p. 56, Aug. 2008.

[5] J.-J. Chen, X. S. Hu, D. Mossé, and L. Thiele, "Guest editorial special section on power-aware computing," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 253–254, Jul. 2010.

[6] J. Choi and H. Cha, "A processor power management scheme for handheld systems considering off-chip contributions," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 255–264, Jul. 2010.

[7] F. Sun, H.Wang, F. Fu, and X. Li, "Survey of FPGA lowpower design," in *Proc. ICICIP'2010 Conf.*, 2010, pp. 547–550.

[8] P. Ferrari, A. Flammini, D. Marioli, and A. Taroni, "A distributed instrument for performance analysis of real-time Ethernet networks," *IEEE Trans. Ind. Informat.*, vol. 4, no. 1, pp. 16–25, Feb. 2008.

[9] N.J. Cotton, B.M. Wilamowski, "Compensation of Nonlinearities Using Neural Networks Implemented on Inexpensive Microcontrollers ," *IEEE Trans. on Industrial Electronics*, vol. 58, no. 3, pp. 733 - 740 , March 2011.

[10] S. Poorani, T. V. S. Urmila Priya, K. Udaya Kumar, and S. Renganarayanan, "FPGA based fuzzy logic controller for electric vehicle," *J. Inst. Elect. Eng., Singapore*, vol. 45, no. 5, pp. 1–14, 2005.

[11] D. Kim, "An implementation of fuzzy logic controller on the reconfigurable FPGA system," *IEEE Trans. Ind. Electron.*, vol. 47, no. 3, pp. 703–715, Jun. 2000.

[12] Hao Guo, Kay-Soon Low, Hong-Anh Nguyen, "Optimizing the Localization of a Wireless Sensor Network in Real Time Based on a Low-Cost Microcontroller," *IEEE Trans. on Industrial Electronics*, vol. 58, no. 3, pp. 741 - 749 , March 2011.

[13] F. Simonot-Lion, "Guest Editorial Special Section on In-Vehicle Embedded Systems ," *IEEE Trans. on Industrial Informatics*, vol. 5, no. 4, pp. , Nov 2009.

[14] J. Kolbusz, S. Paszczyński and B.M. Wilamowski, "Network traffic model for industrial environment", *IEEE Transaction on Industrial Informatics*, vol. 2, No. 4, pp. 213-220, 2006

[15]AUTOSAR GbR, Technical Overview, V3.0. Available at: http://www.autos ar.org, 2008.

[16] http://en.wikipedia.org/wiki/AUTOSAR

[17] T. Nolte, Insik Shin, M. Behnam, M. Sjodin, "A Synchronization Protocol for Temporal Isolation of Software Components in Vehicular Systems ," *IEEE Trans. on Industrial Informatics*, vol. 5, no. 4, pp 375-387. , Nov 2009.

[18] F. Baronti, E. Petri, S. Saponara, L. Fanucci, R. Roncella, R. Saletti, P. D'Abramo, "Design and Verification of Hardware Building Blocks for High-Speed and Fault-Tolerant In-Vehicle Networks ," *IEEE Trans. on Industrial Electronics*, vol. 58, no. 3, pp. , March 2011.

[19] Haibo Zeng, M. Di Natale, P. Giusto, A. Sangiovanni-Vincentelli, "Stochastic Analysis of CAN-Based Real-Time Automotive Systems ," *IEEE Trans. on Industrial Informatics,* vol. 5, no. 4, pp 388-401. , Nov 2009.

[20] J. L. Diaz, D. F. Garcia, K. Kim, C.-G. Lee, L. Lo Bello, J. M. Lopez, S. L. Min, O. Mirabella, "Stochastic Analysis of Periodic Real-Time Systems", *23rd IEEE Real-Time Systems Symposium (RTSS'02), 2002.*

[21] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (can) schedulability analysis: Refuted, revisited and revised,"*Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007

[22] M. K. Gardner, "Probabilistic analysis and scheduling of critical soft real-time systems," Ph.D. dissertation, Dept. Comput. Sci., Univ. Illinois at Urbana-Champaign, Urbana, IL, 1999.

[23]J. M. López, J. L. Díaz, J. Entrialgo, and D. García, "Stochastic analysi sof real-time systems under preemptive priority-driven scheduling," *Real-Time Syst.*, vol. 40, no. 2, pp. 180–207, 2008.

[24] N. Navet, Y.-Q. Song, and F. Simonot, "Worst-case deadline failure probability in real-time applications distributed over controller area network," *J. Syst. Architecture*, vol. 46, no. 7, pp. 607–617, 2000.

[25]T. Nolte, H. Hansson, and C. Norström, "Probabilistic worst-case response-time analysis for the controller area network," in *Proc. 9th IEEE Real-Time and Embedded Technol. Appl. Symp. (RTAS)*, May 2003, pp. 200–207.

[26]J. P. Lehoczky, L. Sha, and Y. Ding, "The rate-monotonic scheduling algorithm: Exact characterization and average case behavior," in *Proc. 10th IEEE Real-Time Syst. Symp.*, Dec. 1989, pp. 166–171.

[27]J. C. Palencia and M. Gonzáles Harbour, "Schedulability analysis for tasks with static and dynamic offsets," in *Proc. 19th IEEE Real-Time Syst. Symp. (RTSS'98)*, Dec. 1998, pp. 26–37.

[28]W. Lei, Z. Wu, and M. Zhao, "Worst-case response time analysis for osek/vdx compliant real-time distributed control systems," in *Proc. 28th Annu. Int. Comput. Softw. Appl. Conf. (COMPSAC'04)*, 2004, pp. 148–153.

[29]R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (can) schedulability analysis: Refuted, revisited and revised,"*Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[30]S. Schliecker, M. Negrean, R. Ernst, "Response Time Analysis on Multicore ECUs With Shared Resources ," *IEEE Trans. on Industrial Informatics,* vol. 5, no. 4, pp. , Nov 2009.

[31] N. Navet, A. Monot, B. Bavoux, F. S. Lion, "Multi-source and multicore automotive ECUs-OS protection mechanisms and scheduling", *IEEE International Symposium on Industrial Electronics,* (ISIE'10), 2010.

[32] C.-M. Chen and S. K. Tripathi, "Multiprocessor priority ceiling based protocols," Univ. Maryland, College Park, Tech. Rep., 1994.

[34] M. Rahmani, K. Tappayuthpijarn, B. Krebs, E. Steinbach, R. Bogenberger, "Traffic Shaping for Resource-Efficient In-Vehicle Communication ," *IEEE Trans. on Industrial Informatics,* vol. 5, no. 4, pp. , Nov 2009.

[35] B. M. Wilamowski, H. Yu, "Improved Computation for Levenberg Marquardt Training," *IEEE Trans. on Neural Networks,* vol. 21, no. 6, pp. 930-937, June 2010.