Challenges in Future Automobile Control Systems with Multicore Processors

Dakai Zhu and Chunjiang Qian
University of Texas at San Antonio

1 Introduction

Modern vehicles have become increasingly computerized to satisfy the more strict safety requirements and to provide better driving experiences. Therefore, the number of *electronic control units (ECUs)* in modern vehicles has continuously increased in last few decades (e.g., some luxury cars can have up to 100 ECUs). In addition, advanced functionalities (e.g., collision avoidance and adaptive cruise control) put higher computational demand on ECUs, which further increases the design complexity of automotive control systems. Multicore processors, where multiple processing units are integrated on a single chip, have emerged to be the main computing engine not only for high-end servers but also for embedded control systems. For instance, some multicore processors have been recently developed for automotive ECUs [3, 11].

With multicore processors, more centralized architecture designs can be adopted for automotive control systems. That is, instead of having many ECUs following the traditional approach of *"one function per ECU"*, we can have a few powerful multicore ECUs and each of them integrate the functionalities of several single-core ECUs from the same or similar domains (e.g., powertrain and body). The recent initiative on *AUTOmotive Open System ARchitecture (AUTOSAR)* has established several standards for automotive software and hardware designs, which include guidelines for designing centralized architecture with multicore ECUs for automotive control systems [2, 5]. With AUTOSAR, it is expected that computational control tasks of different functions can share one ECU or run on any ECU connected with in-vehicle network (e.g., CAN and FlexRay).

Although multicore processors provide great opportunities to mitigate the design complexity of automotive control systems with reduced number of ECUs, the integration and scheduling of different control tasks on multicore ECUs also bring various challenges. In what follows, we first discuss the challenges for designing the hardware architecture with multicore ECUs. Then, we will address the benefits and scheduling issues within multicore ECUs and across in-vehicle networks.

2 Multicore Domain Control Units (MDCUs)

Following the *"one function per ECU"* design paradigm, the number of ECUs in modern vehicles is typically around 50 to 70, ranging from basic safety-related functions (e.g., ABS and engine and transmission control) to auxiliary and advanced functions (e.g., navigation and collision avoidance systems) [1]. With increased computation power in modern processors, the control tasks from

several ECUs can be integrated and processed on a single powerful ECU. For instance, engine and transmission control units have been merged to be the powertrain control module [1]. With the rich computation power of multicore processors, it can be expected that more control tasks from the same or similar domains will be integrated and run on a *multicore domain control unit (MDCU)*. Therefore, instead of having nearly 100 ECUs, depending on the integration levels, future vehicles may have around 10 to 20 MDCUs assuming that each MDCU has 4 to 8 processing cores, which are connected through CAN or FlexRay network, as illustrated in Figure 1.

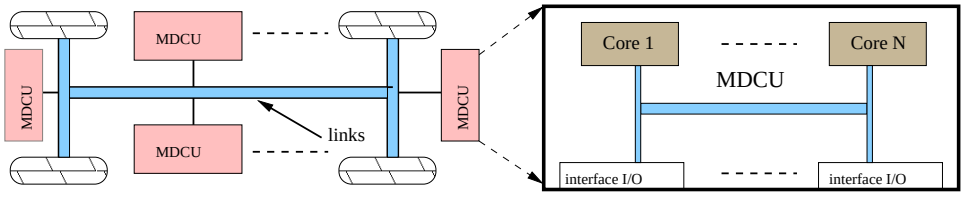

Figure 1: Automotive control system architecture with MDCUs

However, considering the diverified control functions for various vehicle components, such integration brings many challenges and requires careful design and verification. For instance, in addition to the computational requirements of control tasks from different ECUs, the physical locations of the corresponding sensors and actuators should also be considered. Moreover, critical control tasks may need to be integrated with non-critical auxiliary tasks to increase management flexibility and fault tolerance as discussed in the next section.

3 Scheduling Challenges for Automotive MDCUs

The performance interference between tasks running currently on multicore processors due to implicitly shared resources (such as caches) makes runtime analysis very difficult [9, 10], which has been the main obstacle to adopt multicore processors in real-time control systems. Following the AUTOSAR specifications, several software development and scheduling schemes have been studied very recently [6, 7, 8], where the state-of-the-art solutions are based on the static partitioned approach to assign control tasks to individual processing cores offline. Such resulting static cyclic schedules greatly limit the flexibility and system utilization efficiency.

Note that, the inherent hardware redundancy provides excellent opportunities for fault tolerance and one recent research also addressed the safety issues in multicore powered ECUs [4]. For critical control tasks (e.g., engine control and ABS), we may process them on more than one cores to increase their reliabilities. Moreover, instead of statically assign tasks to fixed cores, we can explore *adaptive* and *global scheduling* schemes to disable the non-critical tasks while ensuring the computational requirements of critical ones in case some cores within a MDCU fail. If the MDCU fails, the coordinated scheduling schemes can also dynamically re-map and migrate critical control tasks to other healthy MDCUs appropriately. Such flexibility can greatly increase the dependability of the automotive control system. In addition, by adopting the optimal global scheduling schemes (such as Pfair-like algorithms), better system utilization can also be achieved.

Considering the increased delay for sensor input and control signals in MDCUs architecture, we should consider the control system *stability* when determining the scheduling policy and timing

parameters for control tasks. With more control signals being delivered through the in-vehicle network, the co-scheduling of the signals on the communication links and tasks on processing cores in MDCUs becomes more important and thus demands more research efforts.

Bio for the Authors

Dakai Zhu is currently an Assistant Professor in the Department of Computer Science at the University of Texas at San Antonio. He received the Ph.D. degree from the University of Pittsburgh in 2004. His research interests include real-time systems, power aware computing and fault-tolerant systems. Dr. Zhu was a recipient of the NSF CAREER Award in 2010. Email: `dzhu@cs.utsa.edu`; Phone: 210-458-7453.

Chunjian Qian is currently an Associate Professor in the Department of Electrical and Computer Engineering at the University of Texas at San Antonio. He received the Ph.D. degree from the Case Western Reserve University in 2001. His research interests include control theory (e.g., nonlinear, robust, adaptive and real-time optimal control), hybrid control systems and cyber physical systems. Dr. Qian was a recipient of the NSF CAREER Award in 2003. Email: `chunjiang.qian@utsa.edu`; Phone: 210-458-5587.

References

[1] Automotive electronic systems, available at http://www.cvel.clemson.edu/auto/, 2010.

[2] Autosar (automotive open system architecture), http://www.autosar.org/., 2010.

[3] J. Brodt (NEC Electronics America). Revving up with automotive multicore; http://www.edn.com/, 2008.

[4] C. Aussagues, 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 The Embedded Real-Time Software and Systems (ERTS^2)*, May 2010.

[5] F. Kluge, C. Yu, J. Mische, S. Uhrig, and T. Ungerer. Implementing autosar scheduling and resource management on an embedded smt processor. In *Proc. of th 12th Int'l Workshop on Software and Compilers for Embedded Systems (SCOPES)*, pages 33–42, 2009.

[6] H. Kopetz, R. Obermaisser, C.E. Salloum, and B. Huber. Automotive software development for a multi-core system-on-a-chip. In *Proc. of 4th Int'l Workshop on Software Engineering for Automotive Systems*, 2007.

[7] A. Monot, N. Navet, F. Simomot, and B. Bavoux. Multicore scheduling in automotive ecus. In *Proc. of The Embedded Real-Time Software and Systems (ERTS^2)*, May 2010.

[8] N. Navet, A. Monot, B. Bavoux, and F. Simonot-Lion. Multi-source and multicore automotive ecus: Os protection mechanisms and scheduling. In *Proc. of IEEE Int'l Symposium on Industrial Electronics*, Jul. 2010.

[9] S. Schliecker, M. Negrean, and R. Ernst. Response time analysis on multicore ecus with shared resources. *IEEE Trans. on Industrial Informatics*, 5(4):402 – 413, 2009.

[10] S. Schliecker, J. Rox, M. Negrean, K. Richter, M. Jersak, and R. Ernst. System level performance analysis for real-time automotive multicore and network architectures. *IEEE Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 28(7):979–992, 2009.

[11] P. Leteinturier (Infineon Technologies). Multi-core processors: Driving the evolution of automotive electronics architectures; http://www.embedded.com/design/multicore/, 2007.