Timing and Schedulability Analysis for Distributed Automotive Control Applications

Samarjit Chakraborty
TU Munich
Munich, Germany
samarjit@tum.de

Marco Di Natale
Scuola Superiore Sant'Anna
Pisa, Italy
marco.dinatale@sssup.it

Heiko Falk
TU Dortmund
Dortmund, Germany
heiko.falk@udo.edu

Martin Lukasiewycz
TUM CREATE
Singapore
martin.lukasiewycz@tum-create.edu.sg

Frank Slomka
University of Ulm & INCHRON GmbH
Ulm, Germany
frank.slomka@uni-ulm.de

ABSTRACT

High-end cars today consist of more than 100 electronic control units (ECUs) that are connected to a set of sensors and actuators and run multiple distributed control applications. The design flow of such architectures consists of specifying control applications as Simulink/Stateflow models, followed by generating code from them and finally mapping such code onto multiple ECUs. In addition, the scheduling policies and parameters on both the ECUs and the communication buses over which they communicate also need to be specified. These policies and parameters are computed from high-level timing and control performance constraints. The proposed tutorial will cover different aspects of this design flow, with a focus on timing and schedulability problems. After reviewing the basic concepts of worst-case execution time analysis and schedulability analysis, we will discuss the differences between meeting timing constraints (as in classical real-time systems) and meeting control performance constraints (e.g., stability, steady and transient state performance). We will then describe various control performance related schedulability analysis techniques and how they may be tied to model-based software development. Finally, we will discuss various schedule synthesis techniques, both for ECUs as well as for communication protocols like FlexRay, so that control performance constraints specified at the model-level may be satisfied. Throughout the tutorial different commercial as well as academic tools will be discussed and demonstrated.

Categories and Subject Descriptors

C.3 [**Special-Purpose And Application-Based Systems**]: Real-time and embedded systems

General Terms

Algorithms, Design, Performance

Keywords

Timing Analysis, Schedulability Analysis, Distributed Automotive Systems, Control Applications

EMSOFT'11, October 9–14, 2011, Taipei, Taiwan.
ACM 978-1-4503-0714-7/11/10.

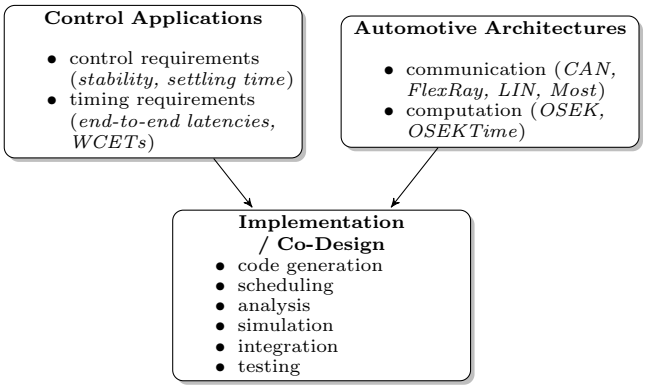

Figure 1: Illustration of a concurrent design of distributed automotive control systems.

1. OUTLINE

In the following, different aspects of a design flow for timing and schedulability analysis of distributed automotive control application are discussed. These aspects break down to state-of-the-art *automotive architectures*, *control applications*, and a design methodology that considers a concurrent *co-design*. An overview is illustrated in Figure 1.

1.1 Automotive Architectures

Modern cars consist of a large number of electronic components that carry out highly diverse applications with the goal to increase the safety and comfort of the driver. Recent innovations in the automotive domain like *adaptive cruise control* or *lane*, *parking*, and, *traffic assistant* systems are put into practice by integrated hardware/software solutions. As a result, modern high-end cars consist of more than 100 electronic control units (ECUs) that are connected to a set of sensors and actuators, running a multitude of distributed control applications. These automotive architectures are highly heterogeneous, consisting of many different computational and communication components. This growing complexity issues enormous challenges to designers and developers of automotive systems.

Computing devices in the automotive domain have to fulfill stringent safety and reliability requirements. Moreover,

these devices and their corresponding operating systems have to offer a predictable behavior to enable a sound timing and schedulability analysis. In order to cope with complexity of novel control applications, the introduction of novel hardware architectures such as multi-core processors becomes necessary, making the analysis even more challenging.

The well-established bus protocols in the automotive domain are CAN, FlexRay, and LIN. In particular, the novel FlexRay bus, see [5], is a promising candidate for upcoming automotive architectures and control applications which have much higher demands on bandwidth and real-time behavior. The FlexRay protocol has been developed to meet these requirements and provides a robust, scalable, deterministic, and fault-tolerant communication system for advanced automotive control applications. However, the scheduling and timing analysis of the bus is a challenging task. The configuration requires 70 network parameters as well as a schedule that fulfills the real-time requirements of the control applications.

1.2 Control Applications

In a modern car, the complexity of control applications is highly diverse, putting different demands on the safety and real-time behavior of the system. These demands have to be derived from typical control requirements such as the *stability* or *settling time* of the control model, see [3]. While some control applications are implemented isolated on single ECUs, others are distributed throughout the entire in-vehicle network. Here, the amount of data that has to be exchanged might be very high with strict end-to-end timing delays.This will require novel scheduling approaches and analytical techniques to verify the correct implementation of the control algorithms on the distributed architecture.

The software implementation of distributed control applications running on automotive architectures is typically realized by model-based software development. This involves automatic code generation from high-level control models in description languages such as Simulink/Stateflow, along with the communication driver stack and the operating system. To cope with the complexity of the software development in the automotive domain, AUTomotive Open System ARchitecture (AUTOSAR), see [6], was introduced by several car manufacturers and suppliers. The AUTOSAR initiative enables a standardized and platform-independent development of distributed and concurrent control applications.

A model-based design flow requires an appropriate set of compilers. Traditionally, compilers are unable to use precise estimates of execution times for optimization, and timing properties of code are derived after compilation. A specific number of design iterations is required if timing constraints are not met. A reconciliation of compilers and timing analysis becomes necessary to create a worst-case execution time (WCET) aware compiler in this way [2]. Such WCET-aware compilers can exploit precise WCET information during compilation.

1.3 Co-Design

The model-based development of complex distributed embedded systems requires a much tighter integration of competencies in controls, formal methods and software architectures and platforms, which will require substantial innovation in the education programs and immediate help from the research community. Today, the number of distributed control applications in vehicles is rapidly evolving while further complexity is added with the integration of the upcoming AUTOSAR standard for the definition of application-level components and automotive hardware/software architectures.

A model-based flow and continuous tool support is a promising approach to cope with existing challenges and provide possible solutions, see [7]. Schedules for the bus systems and operating systems have to be synthesized in compliance with the requirements of the distributed applications. In case of control applications, a methodology becomes necessary that also takes the specific characteristic and requirements of the control models into account [3].

Predicting real-time behavior of distributed automotive control systems is an enormous challenge [4]. For the analysis of real-time behavior of distributed control systems there exists a gap between analytical approaches like SymTA/S [8] and simulation environments like chronSIM [1]. On the one hand, simulation as well as emulation is very time-consuming such that results of the experiments are available late in the design process. On the other hand, an analytical approach might lead to overly pessimistic results. Therefore, simulation and analysis have to be combined to make a system understandable in its real-time behavior for the designer and system architect in early design stages of the development.

2. REFERENCES

[1] ChronSIM. `http://www.inchron.de/chronsim.html`.

[2] H. Falk and P. Lokuciejewski. A Compiler Framework for the Reduction of Worst-case Execution Times. *Real-Time Systems*, 46(2):251–300, 2010.

[3] D. Goswami, R. Schneider, and S. Chakraborty. Co-design of cyber-physical systems via controllers with flexible delay constraints. In *Proceedings of the 16th Asia and South Pacific Design Automation Conference (ASPDAC)*, pages 225–230, 2011.

[4] F. König, D. Boers, F. Slomka, U. Margull, M. Niemetz, and G. Wirrer. Application Specific Performance Indicators for Quantitative Evaluation of the Timing Behavior for Embedded Real-time Systems. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, pages 519–523, 2009.

[5] M. Lukasiewycz, M. Glaß, P. Milbredt, and J. Teich. FlexRay Schedule Optimization of the Static Segment. In *Proceedings of the 7th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 363–372, 2009.

[6] R. Racu, A. Hamann, R. Ernst, and K. Richter. Automotive software integration. In *Proceedings of the 44th annual Design Automation Conference (DAC)*, pages 545–550, 2007.

[7] A. Sangiovanni-Vincentelli and M. Di Natale. Embedded System Design for Automotive Applications. *Computer*, 40(10):42–51, 2007.

[8] SymTA/S. `http://www.symtavision.com/symtas.html`.