Methods, Tools and Standards for the Analysis, Evaluation and Design of Modern Automotive Architectures

E. Frank
VaST Systems
Munich, Germany

R Wilhelm
Saarland University,
Saarbrücken, Germany

R Ernst
TU Braunschweig,
Braunschweig Germany

A. Sangiovanni-Vincentelli
Univ. of California Berkeley
Berkeley, CA, USA

M. Di Natale
Scuola Superiore S. Anna
Pisa, Italy

Abstract

Automotive systems are increasingly distributed and complex. Reduced time-to-market, cost and safety concerns require advance validation of the integrated systems and its components, from the functional, timing, and reliability standpoints. In particular, function correctness and performance may depend on communication and computation delays imposed by the selected architecture platform. Hence, the need for methods and tools capable of predicting the system-level timing behaviour (latencies and jitter), resulting from the HW platform selection, the synchronization between tasks and messages, and also from the synchronization and queuing policies of the middleware and RTOS levels. In this paper, we review methods and tools for the evaluation of the function performance and its timing correctness by simulation or by worst case static analysis.

1. Introduction

The complexity of car electronic systems is rapidly growing, and the overall size of the embedded software in cars has now reached millions of lines of embedded code. Furthermore, the fundamental paradigm of system development in car electronics is changing. Vehicle architectures are still traditionally component or sub-system focused, where each function is deployed to an autonomous Electronic Control Unit (ECU), but the proliferation in the number of ECUs, subsystems and buses, and the increasing interdependency of functions makes these systems difficult to test and validate. A shift from the single ECU approach toward an increased networking of control modules within application domains (e.g. Powertrain) as well as across domains (e.g., Powertrain and Chassis) is now taking place.

Furthermore, complex automotive functions, including active safety and safety critical systems, are characterized by non-functional requirements, including timing and performance, requirements for safety, and cost, together with reusability, flexibility, scalability and extensibility of the architecture artifacts.

Model-based development allows the description at design time of the fundamental properties (functional and non functional) of the system in a user-friendly high-level modeling language, based on a mathematical formalism. The model is amenable to analysis by verification or simulation methods in order to detect and fix errors and performance issues at design time. When the model includes the architecture platform on which the functions are executed, the term *virtual prototyping* is also used. Later, the automatic derivation of implementation code from a high-level algorithmic representation of the function is typically performed, reducing substantially the time required for the coding and testing stages.

Most of the software functions developed today and possibly even more in the future, considering the upcoming X-by-wire systems are sensitive to time performance and possibly even time-critical. Real-time constraints require a model capable of expressing timing constraints and tools for the analysis of the average case as well as of worst-case behavior.

System-level analysis and new modeling and analysis methods and tools are not only needed for predictability and composability when partitioning end-to-end functions at design time and later at system integration time, but also for providing guidance and support to the designer in the evaluation and selection of the electronics and software architectures.

Architecture evaluation and selection is a vital stage with tremendous impact on the cost, performance and quality of a vehicle architecture, typically performed years in advance

978-3-9810801-3-1/DATE08 © 2008 EDAA

of subsystem development and integration, in which models of the functions and of the possible solutions for the physical architecture need to be defined and matched to evaluate the quality and select the best possible hardware platform with respect to the performance, reliability and cost metrics and constraints.

For timing related metrics (including end-to-end latency, signal jitter and bus/cpu utilization), schedulability analysis theory can provide the formal evaluation of the worst case timing behavior, stochastic analysis for a better insight into the probability of each latency value, and system-level simulation.

The Controller Area Network (CAN) bus [6], used for most types of communication is based on the concept of a deterministic resolution of the contention and on the assignment of priorities to messages. The OSEK standard for real-time operating systems [12] supports predictable priority-based scheduling [9], and bounded worst case blocking time through an implementation of the immediate priority ceiling protocol [14] and the definition of non-preemptive groups [16]. In absence of faults, and assuming that the worst case execution time of a task can be safely estimated, these standards allow predicting the worst-case timing behavior of computations and communications.

A prerequisite of schedulability analysis is however the capability of obtaining an estimate of the worst case execution times of all the application tasks and the operating system services.

Priority-based scheduling of tasks and messages fits well within the traditional design cycle, in which timing properties are largely verified a-posteriori and applications require conformance with respect to worst case latency constraints rather than tight time determinism. Furthermore, control algorithms are designed to be tolerant with respect to small changes in the timing behavior and to the nondeterminism in time that possibly arises because of preemption and scheduling delays [7], or even possibly to overwritten data or skipped task and message instances because of temporary timing faults. Finally, although formally incorrect, there is a common perception that small changes in the timing parameters (decreased periods and/or wrong estimates of the computation times) typically only result in a graceful degradation of the response times of tasks and messages and that such degradation will in any case preserve the high priority computations.

This is only partly true and can lead to timing faults because of discontinuities in the function that defines the dependency of the worst-case response times of tasks and messages from its computation or transmission times and periods.

In face of the development of larger and more complex applications, which are deployed with a significant amount of parallelism on each ECU and consist of a densely connected graph of distributed computations, and of new safety-critical functions, that require tight deadlines and guaranteed absence of timing faults, the previous assumptions can no more be trusted. Hence, a new rigorous science needs to be established.

A number of issues needs to be considered.

- Priority-based scheduling can lead to discontinuous behavior in time and timing anomalies. The dependency of the response time of a lower priority task or message with respect to the computation time (or period) of a higher priority task is not linear and not even continuous. A small additional high priority load can lead to a sudden increase in the response time on some computation paths [5]. Furthermore, especially in distributed systems, it may even be possible that shorter computation times result in larger latencies [13]. A recently developed branch of worst case timing analysis is focusing on sensitivity analysis [5][13] as a means for understanding which computation and communication loads are critical for the preservation of deadlines on selected paths.

- Variability of the response times between the worst case and the best case scenario, together with the possible preemptions, can lead to the violation of time-deterministic model semantics in the implementation of software models by priority scheduled tasks and messages [3].

- Extensibility and (to some degree) tolerance with respect to unexpectedly large resource requirements from tasks and messages that is allowed by priority-based scheduling comes at the price of additional jitter and latency and lack of timing isolation.

- Future applications, including safety critical (x-by-wire) and active safety need shorter latencies and time determinism (reduced jitter) because of increased performance. The current model for the propagation of information, based on *communication by periodic sampling*, among non-synchronized nodes [4] has very high latency in the worst case and a large amount of jitter between the best case and the worst case delays. Even if communication-by-sampling can be formally studied and platform implementations can be defined to guarantee at least some fundamental properties of the communication flow (such as data preservation [4]), time determinism is typically disrupted and the application must be able to tolerate the large latencies caused by random sampling delays. In a time-triggered system, the scheduling of the tasks and messages can be arranged in such a way that latencies and jitter are controlled and are possibly much shorter.

- The deployment of reliable systems requires timing isolation in the execution of the software components, and protection from timing faults. Timing protection is even more important in light of AUTOSAR, when components from Tier1 suppliers are integrated into the same ECU and faulty behaviors (functional and temporal) need to be contained and isolated. The development of future applications will also require the enforcement of composability and compositionality not only in the functional domain but also for para-functional properties of the system, including the timing behavior of the components and their reliability.

One of the major down-sides of priority-based scheduling of resources is that faulty high priority computation or communication flows can easily obtain the control of the ECU or the bus, subtracting time from lower priority tasks or messages.

In the future scenario, in which application tasks from multiple Tier-1 suppliers are integrated into the same ECU, leveraging the standardization of interfaces allowed by AUTOSAR, protecting the tasks of each IP from the timing errors of other IPs is of fundamental importance. Timing isolation is therefore required to provide for additional separation of concerns and protection.

Time-based schedulers, including those supported by the FlexRay and OSEKTime [11] standards force context switches on the ECUs and the assignment of the communication bus at predefined points in time, regardless of the outstanding requests from the tasks for computation and communication bandwidth. Therefore, they are better suited to provide temporal protection, except that the enforcement of a strict time window for the execution and communication requires a much better capability of the designer in predicting the worst case execution times ([8]) of tasks so that the execution window can be appropriately sized, and guardians are needed to ensure that an out-of-time transmission will not disrupt the communication flow on the bus.

Moving from an event-triggered system scheduled by priority to a time-triggered system in which resources are allocated by time slots (or time windows) requires a much better a-priori understanding of the timing properties (worst case computation time) of the software and of the communication messages to allocate communication slots and define the scheduling tables. Development processes must be updated to include design-time specification and early verification of timing properties, including determination of the worst case computation time of tasks.

The following sections will provide insight on the major challenges in the development of methods and tools for performance and worst case timing analysis. Section 2 presents a software technology that allows to explore a number of architectural alternatives in automotive subsystems by virtual prototyping. Section 3 provides an overview of the problem of computing the worst case computation time of tasks starting from a code implementation. Section 4 describes the use of timing analysis tools for the verification of end-to-end latencies in complex distributed systems and the evaluation of architecture solutions.

2. Simulation-Based Analysis of Automotive Electronic Control Units

Automobiles are increasingly differentiated by features enabled by electronic control units (ECUs). Consequently electronics content grows as a percentage of automobile development costs-electronics are typically 40% of luxury vehicles and over 70% of the cost of hybrids. As electronics complexity and content escalates, methods for precisely optimizing system performance and cost become essential in the automotive electronic design process. The use of virtual prototypes-high-speed, cycle-accurate software representations (models) of ECUs and their surrounding components, such as plant models-allow precise empirical analysis of system performance. Furthermore, the relative ease of automated system reconfiguration and analysis provide a practical method for supporting directed exploration of the solution space where architectural tradeoffs can be measured to determine precisely the impact on system performance.

The central question is: what role does each member of the automotive electronics design supply chain play in architectural analysis? Generally, the Semiconductor suppliers have the device hardware expertise, and the Tier 1 and OEMs have the end-systems and system software expertise. New higher performance multi-core architectures require deeper understanding of the tradeoffs between hardware and software, and therefore require more precise methods for semiconductor suppliers and control systems engineers to interact. Ultimately, what is needed is an executable specification medium, that can be passed to the OEM and Tier 1, on which they can run their full (and often confidential) software suites and both pass back to the suppliers actual performance results as well as recommended, alternative architectural changes. The Virtual Prototype is being employed at leading OEMs and Tier 1s as such an executable specification.

This presentation discusses the underlying modeling technology behind virtual prototypes, and presents methods for performing automated simulation-based exploration and analysis of proposed solution spaces for automotive electronic control units.

3. Code-Level Worst-Case Execution Time Analysis

Hard real-time systems are subject to stringent timing constraints which are dictated by the surrounding physical environment.

Schedulability analysis has to be performed in order to guarantee that all timing constraints will be met. This stage is also called *timing validation*. Schedulability analysis requires a preceding determination of upper bounds for the execution times of all the system's tasks. These upper bounds are commonly called worst-case execution times (WCETs). The determination of the WCET of software is an increasingly complex problem due to the use of processor components such as caches, pipelines, and all kinds of speculation, which make the execution time of an individual instruction locally unpredictable. Such execution times may vary between a few cycles and several hundred cycles.

A Working Solution A combination of static program analysis with integer linear programming (ILP) has been successfully used to determine precise upper bounds on the execution times of real-time programs with uninterrupted execution.

The method and tools based on it are in routine use in the aeronautics and automotive industries. AbsInt's tool, aiT, has been used in the certification of several time-critical subsystems of the Airbus A380 [2] and is therefore accepted as validated tools for these applications by the European Airworthiness Authorities.

Integration Timing Analysis profits from information available at the model level in model-based design and often lost during code synthesis and compilation. aiT has been integrated with Scade and ASCET-SD to increase precision. It has also been integrated with Evidence's and Symtavision's schedulability analysis, as schedulability analysis is the prime customer of upper bounds on execution times [1].

Extensions Possible extensions to the basic WCET analysis of sequential task code are the following.

1. Timing analysis of tasks for preemptive scheduling needs the determination of context-switch costs. A clever memory-allocation strategy may reduce the interference of the tasks on the processor caches and thus reduce these costs [18].

2. Many automotive systems are executed in different operating modi. These modi have different associated real-time constraints. A modus-specific timing analysis is able to produce more precise upper bounds for the different modi compared to an overall upper bound for the whole system.

Alternative Approaches In [17] a survey of most existing approaches and tools for the determination of the WCET can be found. Alternative approaches based on measurement of execution times both end-to-end and piecewise have problems with soundness, i.e. cannot derive guarantees, or precision or both.

Design for Predictability The architecture of the underlying execution platform, characteristics of the software, e.g. coding rules, code synthesis, abstraction layers, component-based design, the used scheduling strategy influence the analysability of the system and the precision of the timing-analysis results. Recent results on the predictibility of different cache architectures will be presented [10].

A new discipline, Design for Predictability is advocated encompassing all system layers [15].

4. Automotive system level performance analysis and optimization

A growing number of networked applications are implemented on increasingly complex automotive platforms with several bus standards and gateways. Together, they challenge the automotive design process. Recent automotive software standards, in particular AUTOSAR that defines a network runtime environment on top of the existing automotive standards, are intended to improve portability and interoperability. AUTOSAR shall replace or extend earlier proprietary software architecture solutions, but it does not yet sufficiently address time and platform modelling and specification. This is a serious concern as AUTOSAR has increased portability, reuse, scalability, and interoperability as main targets. Only with a generally applicable timing model, AUTOSAR can develop its full capacity.

On the other hand, there are recent results in compositional performance analysis which can be exploited to analyze such automotive networked systems, and, due to their low computational cost compared to simulation, can be applied to design space exploration in a complex automotive supply chain. The resulting tools and methods can even be used to optimize the robustness of an architecture which is important to handle updates and to extend the lifetime of an architecture.

These tools and methods can be used to improve the automotive system quality. This has been shown in automotive component designs, even for cars that are already in production today. On the other hand, a shift in the verification process towards using more formal models will be a major effort for industry. For good reasons the automotive industry has developed a conservative approach to introducing innovations in the design process since changes affect

the whole supply chain. Yet, the challenges of new platform architectures including multiple gateways and processors with caches, an increased dynamic in the design process due to software updates, and variants, networked control functions, such as adaptive cruise control have created new challenges that do not fit the current prototyping and test based verification processes any more. In effect, AUTOSAR only made that trend towards complexity more visible. The expected advent of multi-core architectures might have an even higher impact.

It is, therefore, very important to enable a smooth evolution when introducing the new techniques, starting as a complement to the existing techniques, checking the results of prototyping and test for completeness. A very good place to start are the early design phases where analytical approaches based on estimated values have always been used. There, the necessary data of process timing and communication can be captured which can be refined throughout the development process and compared against the estimation data. This requires new measurement and profiling techniques, or even a combination with formal process timing analysis where highly reliable data are needed such as in safety critical applications.

In is important to fully understand these issues, together with the status of formal performance models and tools and how to apply them in the automotive design process.

References

[1] Interest ist-2005-2.5.3 project. website, http://www.ist-world.org/, 2006.

[2] AbsInt. Absint enhances the safety of the a380. In *available at http://www.absint.com/releases/050427.htm*.

[3] M. Baleani, A. Ferrari, L. Mangeruca, and A. Sangiovanni-Vincentelli. Efficient embedded software design with synchronous models. In *EMSOFT '05: Proceedings of the 5th ACM international conference on Embedded software*, pages 187–190, New York, NY, USA, 2005. ACM Press.

[4] A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. L. Guernic, and R. de Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91, January 2003.

[5] E. Bini, M. D. Natale, and G. Buttazzo. Sensitivity analysis for fixed-priority real-time systems. In *Euromicro Conference on Real-Time Systems*, Dresden, Germany, June 2006.

[6] R. Bosch. Can specification, version 2.0. Stuttgart, 1991.

[7] P. Caspi and A. Benveniste. Toward an approximation theory for computerised control. In *Proceedings of the EMSOFT 2002 conference*, pages 294–304, 2002.

[8] C. Ferdinand et al. Reliable and precise WCET determination for a real-life processor. In *EMSOFT*, pages 469–485, 2001.

[9] M. G. Harbour, M. Klein, and J. Lehoczky. Timing analysis for fixed-priority scheduling of hard real-time systems. *IEEE Transactions on Software Engineering*, 20(1), January 1994.

[10] J. Reineke, D. Grund, C. Berg and R. Wilhelm. Timing predictability of cache replacement policiess. In *Real-Time Systems, 37(2):99-122*, November 2007.

[11] OSEK. *OSEK/VDX Steering Committee: Time-Triggered Operating System*. vailable online at http://www.osek-vdx.org.

[12] OSEK. Osek os version 2.2.3 specification. available at http://www.osek-vdx.org, 2006.

[13] R. Racu and R. Ernst. Scheduling anomaly detection and optimization for distributed systems with preemptive task-sets. In *12th IEEE Real-Time and Embedded Technology and Applications Symposium*, San Jose, April 2006.

[14] L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE transaction on computers*, 39(9):1175–1185, September 1990.

[15] L. Thiele and R. Wilhelm. Design for timing predictability. In *Real-Time Systems, 28:157 - 177*, 2004.

[16] Y. Wang and M. Saksena. Scheduling fixed priority tasks with preemption threshold. In *Proceedings, IEEE International Conference on Real-Time Computing Systems and Applications*, December 1999.

[17] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, F. Mueller, I. Puuat, P. Puschner, J. Staschulat, and P. Stenström. The determination of worst-case execution times-overview of the methods and survey of tools. In *ACM Transactions on Embedded Computing Systems (TECS)*, 2008.

[18] G. Gebhard and S. Altmeyer Optimal task placement to improve cache performance EMSOFT 2007, Pages: 259 - 268