

Automotive Software Integration

Razvan Racu and Arne Hamann and
Rolf Ernst
Institute of Computer and Communication
Network Engineering
Technical University of Braunschweig, Germany
{racu|hamann|ernst}@ida.ing.tu-bs.de

Kai Richter
Symtavision GmbH
Braunschweig, Germany
richter@symtavision.com

ABSTRACT

A growing number of networked applications is 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 modeling and specification. The presentation will give some examples for open issues with respect to performance, timing and interoperability. It will show how recent results in compositional performance analysis can be exploited to analyze such networked systems, and how to apply 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 extend the lifetime of an architecture

Categories and Subject Descriptors

C.3 [Special-Purpose and application-based systems]: Real-time and embedded systems; C.4 [Performance of systems]: Modeling techniques; Performance attributes; Reliability, availability, and serviceability

General Terms

Design, Performance, Reliability, Verification

Keywords

AUTOSAR, Automotive Systems, Timing Model, Software Integration, Formal Analysis, SymTA/S

1. INTRODUCTION

The increasing application complexity, together with a strong time-to-market pressure, requires a massively parallel design of systems, such as in automotive, avionics, multimedia, or telecommunication industries. The supply-chain often contains hundreds

of companies that design their individual components based on requirement definitions from the OEMs or Tier-1 suppliers.

While software modularity is a commonplace in software engineering, it is far from practice in the automotive industry where the electronics system architecture follows company organization and automotive supply chains. There have been good reasons for that approach as it allows clear identification of responsibilities and liabilities. Meanwhile, however, the cost and risk of more complex function integration over a growing software and automotive network complexity have started to change that approach. Automotive networks, today, combine several buses using different protocols (CAN, LIN, FlexRay, MOST) that are combined via gateways [13].

Networked control functions that share sensors and actuators for different functions over a single network with gateways require an integrated view of the automotive electronic systems architecture. Software standards for communication and modularization are urgently needed to support that development.

Even then, systems integration remains a major challenge. Dynamic component interactions result in a variety of non-functional timing and performance dependencies due to scheduling, arbitration, blocking, buffering etc., eventually generating hard-to-find timing problems, including transient overload, buffer under- and over-flows, and missed deadlines. Not having a systematic timing analysis procedure is currently challenging the automotive design process. At the same time, the cost of electronic systems has been rising. This cost increase is mainly due to a lack of understanding and control of integration effects, and a resulting conservative design style.

2. AUTOSAR AND TIMING

The AUTOSAR partnership [7, 8], an alliance of OEM manufacturers and Tier-1 automotive suppliers with many associates, has recognized integration as a major challenge several years ago. Since then, a number of de-facto open industry standards for automotive E/E architectures have been developed: at first confidentially, now for a large part open to the public. The main goal of AUTOSAR is to define a standard modular software infrastructure for application and basic software, which allows exchanging parts of the system's software. This shall enable modularity, scalability, transferability and re-usability of software among projects, variants, suppliers, customers, etc. Figure 1 shows the software layers within a component, as specified in the AUTOSAR standard. Interestingly though, the current AUTOSAR standard still does not contain key aspects of timing and performance. It is important to understand that the primary objective of AUTOSAR is not solving timing problems in particular but supporting integration from a software-engineering perspective.

Timing properties shall be added to the existing component mod-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC 2007, June 4–8, 2007, San Diego, California, USA.

Copyright 2007 ACM ACM 978-1-59593-627-1/07/0006 ...\$5.00.

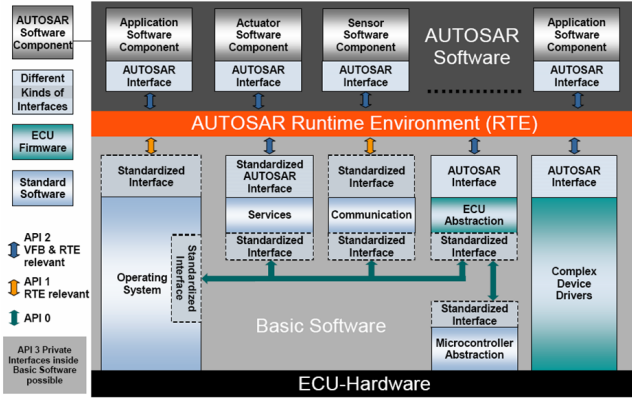


Figure 1: Standardized AUTOSAR software

els in a second step. However, the host of interaction and communication mechanisms defined by AUTOSAR –many of them borrowed from earlier standards such as OSEK/VDX [1, 2]– leads to numerous timing use cases. These, in turn, have many possible interpretations of *timing* with no obvious solution. In the next section we present some examples that show why timing represents a major issue for the correct component integration, and should be considered from the beginning by the AUTOSAR standard. But why is defining a timing model so complex?

The reasons are manifold. Technically, the software engineering view of AUTOSAR lacks clear execution semantics, on which the known approaches to timing analysis could build upon. The simultaneous use of heterogeneous interaction mechanisms complicates timing analysis further and does not match the clear, well-defined models of computation used in real-time systems. Introducing an effective timing model after the software architecture has been defined is a tough technical challenge and requires practical restrictions.

Practicability concerns and economical issues add to that dilemma. A successful technology must support designers in consequently taking decisions; directly and quickly. Therefore, a suitable methodology for using the model and applying the analysis must be in place. The model and methodology must further consider established design and supply-chain processes. IP protection, in particular, complicates modeling as important details are often not available in a particular design stage today.

3. MODEL MISMATCH

In this section we present three key examples of model mismatches that emphasize the complex relations between the timing properties of the system components [12, 11].

3.1 Runnables

At the ECU level, AUTOSAR defines so called *software components (SW-Cs)* as atomic entities from a software development view. However, when it comes to scheduling, each SW-C comprises several so called *runnables*. In the implementation, runnables belonging to different software components are then grouped into *tasks*, which are finally put under operating system control.

Figure 2 shows two software components containing more runnables building three distributed tasks: the first task contains *runnableX* on SW-C2 and *runnableB* on SW-C1; the second task contains *runnableA* on SW-C1 and *runnableY* on SW-C2; the third task contains *runnableZ* on SW-C2. The Gantt diagram shown in Figure 2 presents the execution trace of the three tasks.

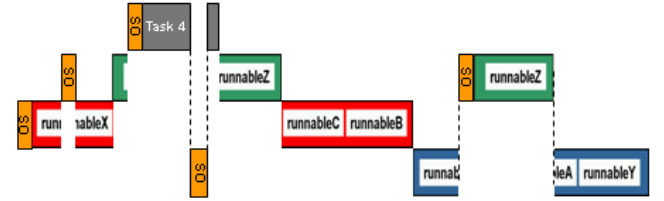
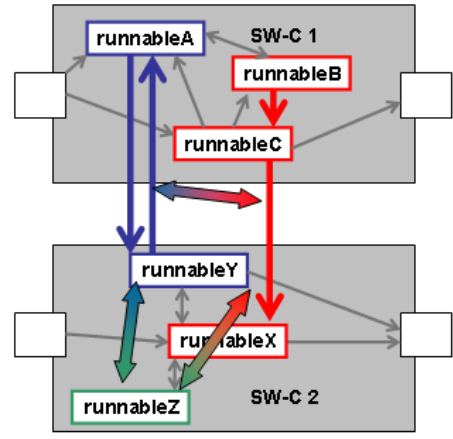


Figure 2: Hidden timing dependencies between SW-Cs

As we can observe, implementation dependent timing behavior results from the low level dependencies crossing the component boundaries. This challenges the real-time behavior of the high level components, by introducing hidden, implementation and state dependent additional jitters and delays at the level of software components.

Clearly, a scheduling analysis can be performed on the low level, but the resulting task timing reveals hardly any direct and intuitive timing-relation with the high-level software components, to which timing information shall finally be attached.

3.2 End-to-End Timing

The second example illustrates another important type of model mismatch at a higher level of communication. With the increasing distribution of functions over several ECUs in a car, the importance of end-to-end timing (and deadlines) is also increasing.

AUTOSAR has already defined models for capturing such timing chains composed of communicating *software components (SW-C)*.

Figure 3 illustrates the software component view of the timing dependencies, mostly determined by the logical flow of data between the software components. Hand-over points (HOPs) shall enable easy composition and decomposition of such chains, thereby providing a framework for system-level timing considerations.

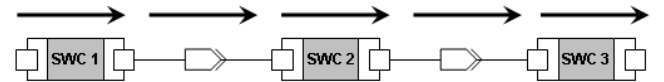


Figure 3: AUTOSAR view on "timing chains"

Similar models are also known from data-flow theory, where clear semantics relate the execution of nodes (here: software components) with timing behavior of the stream. However, AUTOSAR has not yet defined clear rules to describe the activation of the tasks within the software components. Hence, the actual timing

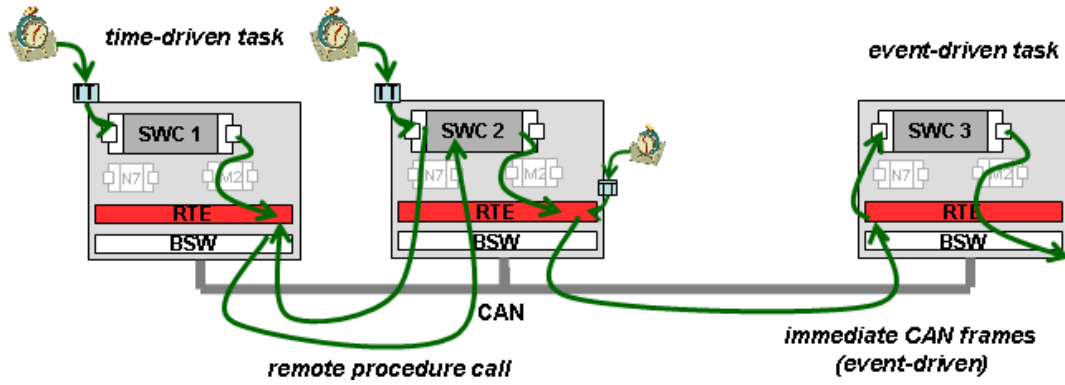


Figure 4: Causality chains in automotive implementations

of software components is undetermined. Moreover, there exist several valid communication semantics including client-server (remote procedure call), periodic sampling including under- and over-sampling, polling, and event-driven. This leads to a variety of indirect *causality chains* in the actual implementation. Figure 4 shows examples for these causality chains through the layered software defined by AUTOSAR.

So, what does end-to-end timing mean in absence of clear execution, communication, and HOP-buffering semantics? The timing is a result of implementation properties and will change with the implementation, uncontrolled by specification and untestable against test component model that lacks the necessary details. There are timing models available that could unambiguously capture and specify such timing properties.

3.3 Bus Communication

A look at bus communication reveals another type of mismatch. AUTOSAR defines a detailed API for the communication stack including several *frame transmission modes* (*direct*, *periodic*, *mixed*, *none*) and *signal transfer properties* (*triggered*, *pending*) with key influences on communication timing. Interestingly, the role of buffers and, in particular, buffer access strategies (FIFO, priority order, etc.) and the over-/underflow mechanisms are mostly left open, despite their enormous influence on signal timing.

Figure 5 illustrates the communication mechanism between the software components of two ECUs connected via a CAN bus. The messages to be transmitted are translated into signals and sent into a queue through *periodic*, *direct* or *mixed frames*. The waiting signals are buffered into the queue according to different buffering strategies (FIFO, priority ordered, hybrid). The signals waiting in the queue are dispatched by driver interrupts and send over the bus as message objects. Obviously, the frame generation modes and the buffering strategy complicate the timing behavior of the transmitted frames and introduce ambiguity and implementation dependency.

Networked functions make guarantees for bus and network communication timing even more important than before. Besides bandwidth and reliability considerations, end-to-end latency becomes crucial as a delay parameter in the control algorithm.

4. THE SYMTA/S APPROACH

With SymTA/S [9], the designer follows an analysis based approach. SymTA/S is based on a modular mathematical model [4] which scales to include software architectures from different sources and suppliers. It capitalizes on the host of work in scheduling theory (e.g. [14, 5, 6]). For any new architecture, a new model can be

easily developed or adapted, such that the tool can follow the development of automotive architectures over time. As an important example, SymTA/S is also capable of considering frame offsets and task-dependencies that will play a dominant role in the experiments later in this paper.

Then, we embedded the powerful analyses into a flexible design framework that poses very few restrictions on the systems and design processes. This framework supports incomplete requirements, missing data, and heterogeneous system configurations. Thanks to this flexibility, SymTA/S can be applied in non-ideal situations that were not amenable to effective and safe analysis, so far.

In practice very often only part of data is available to the OEM, usually in the form of a so called communication matrix, covering the period, length, and priority (CANid). The important dynamic influences are often not available in detail. These include so called *mixed* and *direct* frames that can dynamically appear at virtually any time within the periodic frames (names according to AUTOSAR and OSEK-VDX definitions of *frame transmission mode*). Similarly, the queuing strategies influence frame ordering in the COM layers and drivers, and can "undermine" the priority-driven nature of the CAN protocol. This data is part of the ECU implementation and typically not disclosed to the OEM.

The same holds for frame offset values. Such offsets define local phase relations between frames sent by the same ECU. The rationale behind using offsets is to "balance" the production of frames. Roughly speaking, offsets produce gaps (idle times) in the schedule that other frames can exploit. This balances the ECU interruptions by COM, and it also balances the bus load. Hence, offsets have a positive effect on bus load and, subsequently, response times. But again, such local offsets are typically defined by the ECU supplier and not disclosed to the OEM, nor asked for by the OEM.

From our experience, it seems that such non-ideal situations with lots of "unknowns" are the typical ones, while precise timing requirements and design from scratch are exceptions.

So, can analysis technology really help when data availability is a major concern? In fact, it can, if it is only flexible enough, as we will see by the end of the next section.

5. CASE STUDY

We have applied the SymTA/S technology in a number of studies for automotive OEMs [10], where several ECUs are connected to a bus, sending and receiving a total number of 100 frames and more. In all studies, we imported the length, CAN id (priority), and the period of each frame from a (customer-specific) *communication matrix* or *dictionary*. We knew the frame offsets of only few ECUs,

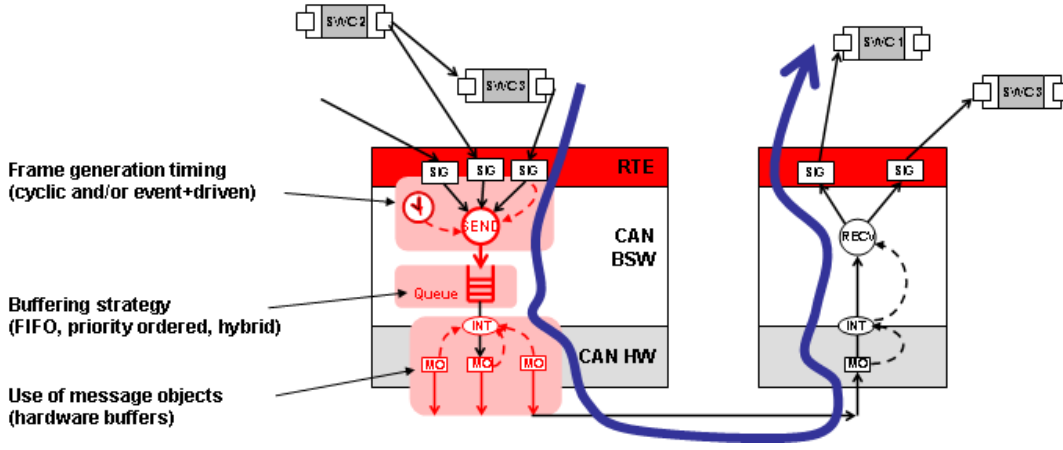


Figure 5: Communication mechanism defined in AUTOSAR

typically the gateways that are under control of the OEM. For other ECUs, we had no offset information. Information about dynamic patterns of mixed and direct frames was mostly missing.

Due to the lack of this important information, we typically conducted a set of experiments with SymTA/S, each based on different assumptions on the missing information. Simulators or prototypes were not required, as SymTA/S is based on mathematical techniques known from real-time scheduling theory. However, the SymTA/S analysis libraries are tailored to the specific real-world mechanisms of protocols (and OSes), and thus, the results have a quite high degree of accuracy.

In the case study presented here, we first ignored all offsets. We considered all frames as being asynchronous and determined their response times. Obviously, such simplifications (no offsets) lead to overestimations with limited practical relevance. However, the fact that we were able to carry out such "what-if" observations within minutes, without any simulation, prototype or test equipment is very important.

Next we repeated the experiment with typical offset values and found out that frame response times dropped by almost 50% for the lower-priority frames (see "utilization gain" in Figure 6). This improvement results from the load balancing that the offsets introduce. We determined these typical values based on experience from other projects in which we were able to compare experimental assumptions with real CAN traces. It turned out that the SymTA/S typical values represent a reasonable approximation to the real world. In other words, the results resemble realistic timing profiles of the studied CAN buses, even though detailed design data was not available.

6. OFFSET OPTIMIZATION

In another set of experiments, we used the SymTA/S automatic exploration plug-in [3, 15] to optimize the offsets in order to reduce the response times further by more than 35%. The results of these experiments are summarized in Figure 6. Each curve represents one experiment. The response times (y-axis) are shown for a representative subset of frames in the order of their priority (x-axis). The frame number 23 (by the bold arrows) resembles a response time improvement from initially 90ms to 45ms (with typical offsets) down to 30ms (with optimized offsets).

It might surprise that –without changing the number of frames, their length and CANId, the bus speed or the average bus load– such an enormous improvement in bus utilization was achieved.

These experiments show that offsets in particular make a huge difference in bus utilization and responsiveness of frames. These are the important dynamic properties of "performance".

We conclude that having an analysis that allows comparing and reasoning about offsets systematically, provides new possibilities for OEMs to optimize their networks without the need for changing the design process.

7. ROBUSTNESS TO ACCEPT ADDITIONAL FRAMES

In the last set of experiments, we were interested in the *robustness* of each bus configuration against additional frames. We wanted to know how many and what frames can be added to a given bus without violating constraints. This is a frequent question when automotive product lines are planned. Room for additional frames allows for additional functions in variants and "face-lifted" later versions of a car.

To analyze this robustness, we gradually added more and more asynchronous high-priority frames to each configuration. We analyzed the new response times and determined which and how many frames miss their deadlines. These additional frames could be mixed or direct frames, or they could model frame retransmission that result from bus errors. In principle, the experiment provides a general measure that could be particularized further.

The results are shown in Figure 7. Again, each curve resembles one of the three known configurations. On the x-axis, the number of additional frames is provided. The y-axis captures the number of frames for which the deadline (10% of frame period, at least 40ms) is violated. The optimized configuration can accept significantly more additional frames before a deadline violation occurs. We call such a configuration *robust*, because the network can safely carry more frames without violating any performance requirements. The number of deadline violations in the typical configuration increases much faster. In that case the bus is much more *sensitive* to additional frames, and therefore less extensible.

The curves indicate a quality increase similar to that in Figure 6, but this time in a more sophisticated context. Not only response times are considered, but they are also related to performance requirements/constraints such as deadlines. Scheduling analysis using SymTA/S offers a wide range of such views that can be customized and extended.

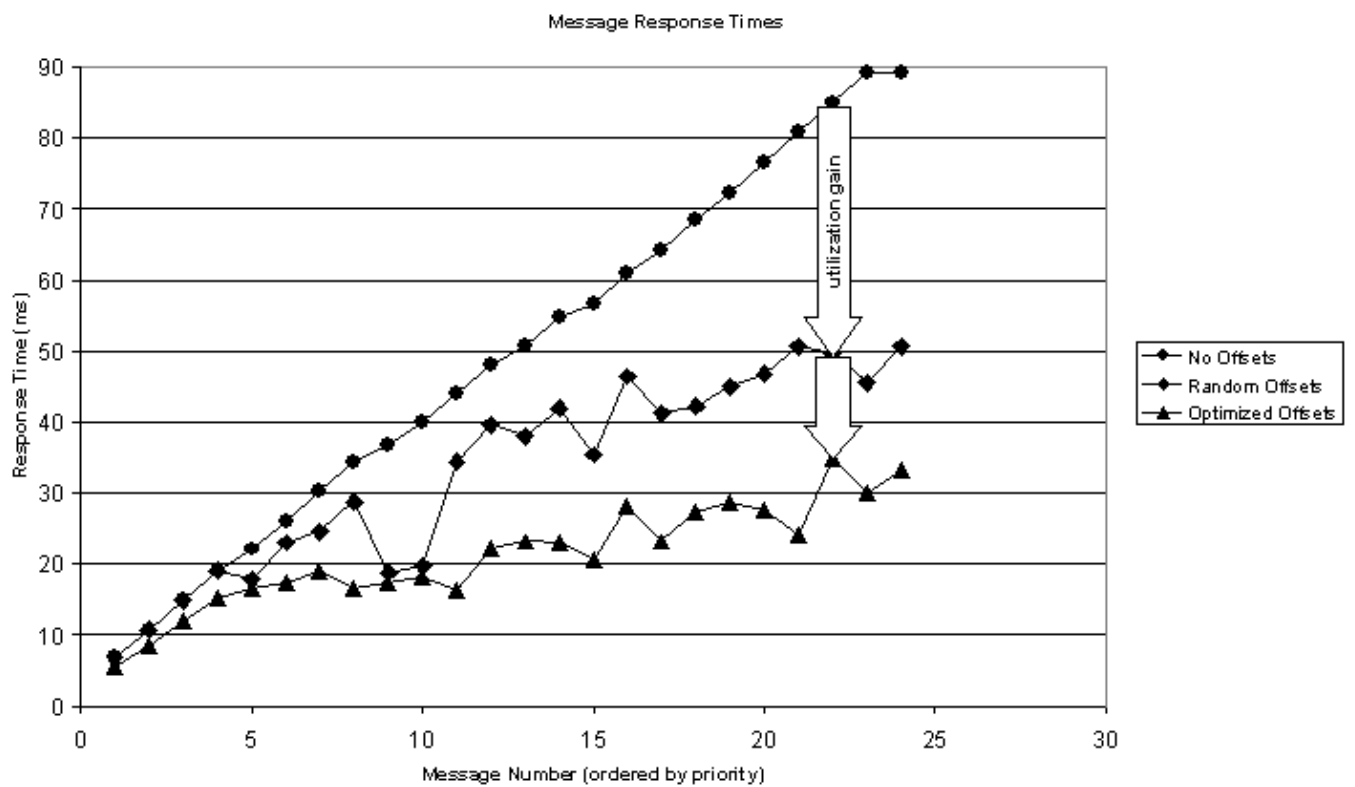


Figure 6: Reducing frame response times through offset optimization

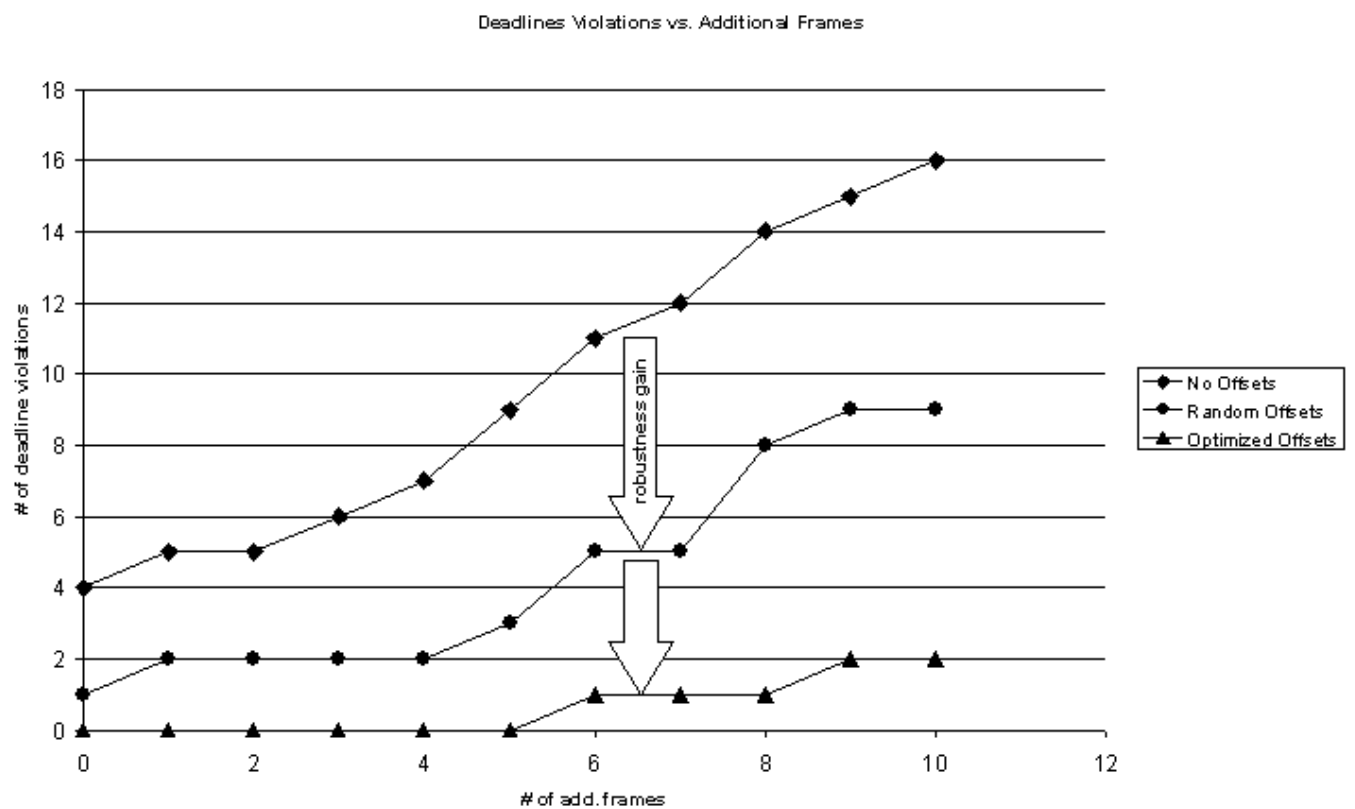


Figure 7: Reducing deadline violations through offset optimization

8. PRACTICABILITY CONCERNS

In addition to a clear and intuitive timing model, designers also need a methodology to determine and to utilize the timing model in the established design flow. Based on the feedback we have been receiving from a variety of designers, this in particular requires:

- Generating or obtaining the data needed for analysis (be it by definition, measurement, test, or simply asking the right people).
- Having a specific strategy when and how to apply the technology.
- Interpreting the results and consequently taking decisions.
- Being able to do all this in a reasonable amount of time, after a reasonable amount of training on that technology.

If a technology appears too complex, designers will avoid it. If input data is not readily available, they cannot use it, and if using the technology takes longer than finding a sub-optimal but acceptable manual solution, it will be considered equally useless. We highlight this, as researches (rightfully) tend to do work that is *elegant* or *systematic* in itself without paying too much attention to practical issues.

8.1 Supply-Chain Issues

Specifically car manufacturers nowadays have to cope with an increasing number of unprecedented real-time problems that are caused by the integration of networked applications. Even though OEMs do not develop large parts of the software, they are responsible for the network that is the main basis of integrations. The network timing, however, depends not only on the protocol but also on driver hardware and software (SW-Cs and COM stack), which is mostly out of the OEM's scope of responsibility and control. The supply-chain communication between OEMs and suppliers will have to evolve, most likely by establishing timing contracts between OEMs and suppliers. In order to be accepted

- Responsibilities and scope must be clearly defined, and must match the established roles of suppliers and OEMs.
- IP protection must be ensured, in particular on the supplier's side. Together with already existing standards like AUTOSAR, this will have a dominant impact on the abstraction of a timing model.
- A comprehensive and reliable timing verification methodology must be in place, since there is no point in modeling something that cannot be analyzed.
- It must be clarified what kind of analysis results and what level of accuracy can be obtained at a particular design stage, and the required effort.

9. EXPERIENCE WITH FORMAL TIMING MODELS

It is clear that automotive platform design and planning can be much more systematic, if supported by a suitable global timing view, the enables reasoning about timing across company borders. On the one hand, academic expertise and support in defining such a model is extremely welcome but it must carefully consider established technological and business processes. On the other hand, industry design practice must also evolve to make designs much more transparent and analyzable, possibly imposing new roles and

responsibilities (and liabilities?) for both OEMs and suppliers. As we have shown in the examples above, such a trend is not well supported by the current AUTOSAR standard. Rules and best practice examples are needed to avoid non-transparent and inflexible designs that counter some of the key goals of AUTOSAR, platform and supplier independence.

10. REFERENCES

- [1] OSEK/VDX Consortium. *OSEK/VDX Communication*. v.3.0.3, July 2004.
- [2] OSEK/VDX Consortium. *OSEK/VDX Operating System*. V.2.2.3, February 2005.
- [3] Arne Hamann, Marek Jersak, Kai Richter, and Rolf Ernst. A Framework for Modular Analysis and Exploration of Heterogeneous Embedded Systems. *Real-Time Systems Journal*, 33(1-3):101–137, July 2006.
- [4] Rafik Henia, Arne Hamann, Marek Jersak, Razvan Racu, Kai Richter, and Rolf Ernst. System Level Performance Analysis - the SymTA/S Approach. *IEE Proceedings Computers and Digital Techniques*, 152(2):148–166, March 2005.
- [5] M. Joseph and P. Pandya. Finding Response Times in a Real-Time System. *The Computer Journal*, 29(5):390–395, 1986.
- [6] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.
- [7] AUTOSAR Partnership. www.autosar.org.
- [8] AUTOSAR Partnership. AUTOSAR—Current Results and Preparations for Exploitation. In *7th EUROFORUM Conference "Software in the vehicle"*, Stuttgart, Germany, May 2006.
- [9] SymTA/S Project. <http://www.symta.org>. Institute of Computer and Communication Network Engineering, Technical University of Braunschweig, Germany.
- [10] Razvan Racu, Rolf Ernst, Kai Richter, and Marek Jersak. A Virtual Platform for Architecture Integration and Optimization in Automotive Communication Networks. In *SAE Congress, System Level Architecture Design Tools and Methods*, volume SP-2129, Detroit, Michigan, April 2007. SAE International.
- [11] Razvan Racu, Kai Richter, and Rolf Ernst. The Need of a Timing Model for the AUTOSAR Software Standard. In *Workshop on Models and Analysis Methods for Automotive Systems (RTSS Conference)*, Rio de Janeiro, Brazil, December 2006.
- [12] Kai Richter. The AUTOSAR Timing Model – Status and Challenges. In *ARTIST2 Workshop Beyond AUTOSAR*, Innsbruck, Austria, 2006.
- [13] Kai Richter and Michael Bartels. Scheduling Analysis of ECUs and Controller Networks. *ATZ Elektronik*, 2, April 2007.
- [14] K. Tindell, A. Burns, and A. Wellings. An Extendible Approach for Analysing Fixed Priority Hard Real-Time Systems. *Journal of Real-Time Systems*, 6(2):133–152, Mar 1994.
- [15] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In *Proc. Evolutionary Methods for Design, Optimisation, and Control*, pages 95–100, Barcelona, Spain, 2002.