Quirks and Challenges in the Design and Verification of Efficient, High-Load Real-Time Software Systems

Ulrich Margull[1], Michael Niemetz[2], Gerhard Wirrer[2]
[1]1 mal 1 Software GmbH, Maxstraße 31, D-90762 Fürth, ulrich.margull@1mal1.com
[2]Continental Automotive GmbH, P.O. Box 100943, D-93009 Regensburg
michael.niemetz@continental-corporation.com

Abstract: Existing concepts for ensuring the correctness of the timing behavior of real-time systems are often based on schedulability analysis methods using exact proofs. Due to the complexity of the scheduling problem, today typically worst case approximations are used to judge the reliability of the timing behavior in software systems. In industrial practice, however, this leads to large safety margins in the design of products which are commercially unacceptable in many application domains. For such highly-efficient systems, schedulability analysis methods that are too pessimistic are of limited benefit. As a consequence, penetration of real-time analysis is suboptimal in the industrial software development, which possibly leads to insufficient quality of the developed products. Therefore, new approaches are needed to support the design and validation of high-load real-time systems with an average CPU load of 90% or above to improve the situation.

Keywords: Scheduling, real-time systems, highly efficient systems, robustness, automotive power train

I. Introduction

In safety critical real time devices, the design and test of the timing behavior plays a central role. With the increasing complexity of devices during the last years, this fact became more and more a consensus in the embedded systems community. Still, in several application domains for embedded devices the penetration with concepts resulting from research activities is quite low, especially when compared to the desktop and server domain. This is astonishing, as embedded devices are typically facing higher requirements in terms of real-time and reliability.

While a state of the art schedulability analysis based on worst case execution times and maximum interrupt rates is able to provide the proof that the timing behavior of the system is guaranteed under all circumstances for each calculation instance, this leads typically to a normal load of the system which is around 50-60% of CPU calculation power. This safety margin, which will probably never be needed during the lifetime of the system, increases the cost of the system significantly, being especially problematic in case of higher volumes.

In the automotive domain, for example, huge volumes are combined with the need for safe systems plus a high availability and enhanced fault tolerance. This requires to reduce the safety margin as much as possible, while keeping a safe and operational unit.

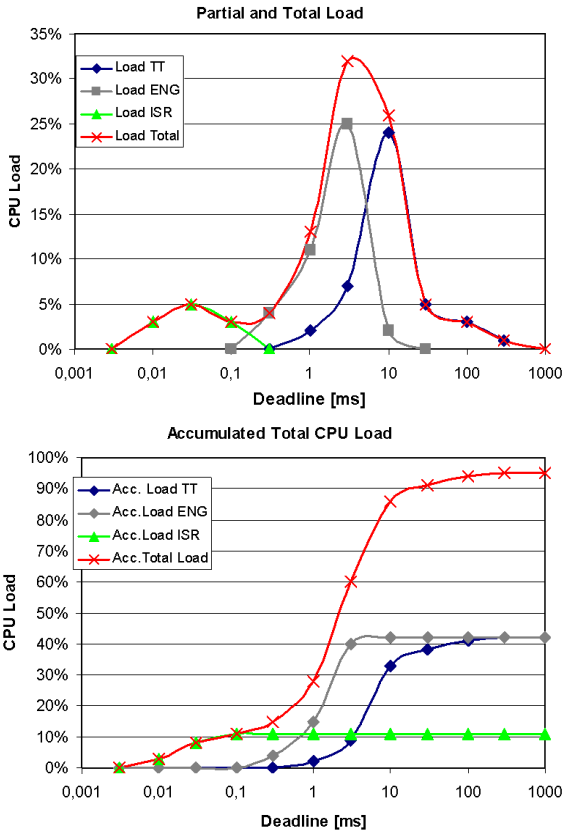

Fig. 1. Partial (top) and accumulated CPU load (bottom) of time-triggered, engine-related and ISR-based calculations

In this paper we will show as an example a non-trivial and significantly event driven embedded system, which has been proven to work reliably with CPU loads above 90%. Afterwards, we will discuss some aspects where an improved theoretical understanding is required in order to design and verify such systems systematically.

Published at ERTS 2010, Toulouse www.erts2010.org

II. System Description

Engine controls units (ECU) have strong real-time requirements. More than 80% of all timing requirements are faster than 10ms. In specific system scenarios, e.g. car going downhill with continuous engine overspeed, an extreme high average CPU load is achieved which can reach even more than 90%. Figure 1 shows the load caused by calculations having been assigned to a certain deadline[1] in relation to the deadlines in a typical application.

The upper part of Figure 1 shows the typical load associated with time-triggered (Load TT), with engine-synchronous (Load ENG) and with interrupt associated calculations. Only about half of all calculations are time-triggered, the rest is event-based, either dependent on the engine position or on other event sources.

On the bottom part of Figure 1 the same load is shown, this time accumulated over the deadline. On the right side the total load of approximately 95% is shown. Please note that this is the average load, measured over many seconds, not only a short peak!

From an overall system point of view many of these deadlines are really *hard deadlines*. For example, missing an injection or ignition point in time can lead to engine misfire, missed combustion cycles or even engine damage and is therefore *not* acceptable. Being able to reach such high loads under those requirements shows to what elaborate performance the industry is driven to in the quest for efficient, affordable systems.

III. Challenges for System Design and Verification

When designing such a system, one faces a set of contradicting requirements: the system has to fulfill safety requirements, it must be reliable, and highly efficient all at the same time. There are several strategies to implement these requirements, and they must be well-balanced against each other.

The real-time behavior of the system is strongly influenced by the scheduling approach and the distribution of the calculation loads. In the automotive industry, operating systems compliant to the OSEK standard are used in most control units. It provides a simple priority-based scheduling with priority-ceiling protocol. However, fast calculations with deadlines of $100\mu s$ or below are often implemented as interrupts. The main challenges in system design is the partitioning of the calculations and how to distribute them on different priority levels.

One critical point is to determine whether a given system will work correctly with the chosen loads and priority levels. Using a classical response time analysis [1], one would determine the worst-case execution time (WCET) for all calculations, model the maximum event stimuli and event rates, for calculating the feasibility. However, this approach is by far too pessimistic. For example, for a typical ECU system, such an analysis easily yields a requirement for 130% to 200% of the available CPU resources! In contrast, the industrial practice shows that the systems are working correctly and are robust and fail-safe in operation. The consequence is that a WCET schedulability analysis has no real meaning for productive systems: if it is feasible, then it is not efficient enough, and if it is efficient, the analysis yields a ”not feasible” result!

A rather pragmatic proceeding to overcome this dilemma is what we call the *optimistic approach*. Here, several aspects of pessimism are replaced by more optimistic approaches. For example, Figure 2 (top) shows an allegory of the system state of an electronic control unit. The bubble in the middle represents all the state space that is explored in operation with the ”center of mass” denoting an average CPU utilization of about 60%.

In order to guarantee schedulability, a classic analysis of this system has to contain some amount of pessimism which is indicated by the box around the bubble.

Using an optimistic approach, we remove the pessimism where possible, thus concentrating on the core of the state space while neglecting some of the rarely used corners (Figure 2 middle), which in the end allows higher average system utilizations (Figure 2 bottom), e.g. up to 90%. However, since we no longer guarantee schedulability in all possible circumstances, the system has to be designed quite differently. It must be robust in such a way that sporadic overload situations are handled gracefully, as well as encapsulating safety relevant or other important functions in areas that can be guaranteed to be scheduled correctly (or are completely independent of the scheduling, e.g. by implementation in hardware).

A classic feasibility analysis contains different kinds of pessimism that can be questioned. One way to reduce this pessimism is to use the *average* execution times (AET) instead of the WCET normally used in classic feasibility analysis.

Our experience shows that when using AET we are able to model the system realistically even at very high CPU loads. On the other hand there is the risk that an AET based analysis might give a too optimistic picture of the system, so the analysis results need to be used carefully and require backup of measurements from the real system. In order to improve the analysis, we add different levels of stochastic behavior to the system: variation of stimuli or variation of runtime (e.g. like [2]).

Another useful analysis strategy which we have successfully applied are stress-tests, where the system is driven artificially into a crisis by adding a dummy load according to a certain pattern. Stress tests help to

[1] We are using relative deadline, i.e. the time duration from activation of one job instance until its calculation is finished.

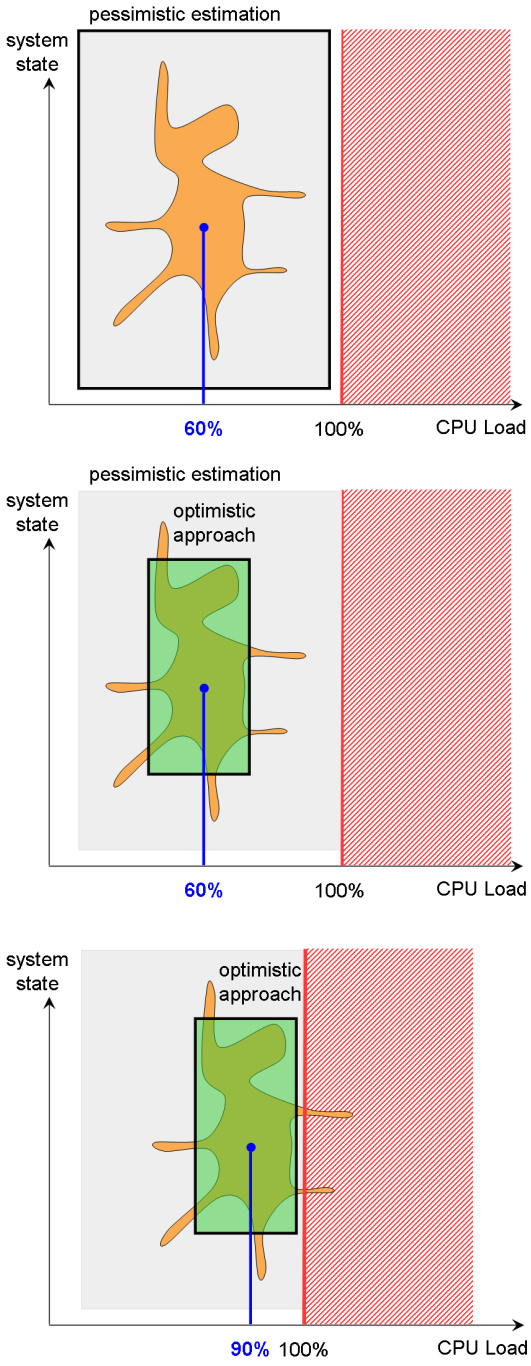

Fig. 2. The overall system load typically has a complex dependency on the current system state. The details of this dependency are often not known in advance due to the complexity of the system. In the figures above, the space of valid system states is represented by the ragged blob-like shape assuming for further simplification that every position within the shape has the same probability of occurrence. Ignoring small areas (with a consequently small probability) during the design of the system allows a more efficient hardware usage. However, this leads to timing violations in some dedicated operation states which need to be known and handled safely.

understand the system dynamics in critical conditions: At which point does my system break down? How does it break-down, what are the weak points leading to a break-down?

The analysis of our systems is further complicated by the fact that our systems are too complex to be fully understood in theory. For example, they are not rate-monotonic, not even strictly deadline-monotonic, they are partly time-triggered, partly event-based, the scheduling is to large parts cooperative, not preemptive, data dependencies cannot be modeled completely due to the complexity of the software (more than 150.000 dependencies), etc. No analysis method seems to be available that takes all restrictions into account, or is able to handle such complexity. As a pragmatic approach we use simulation methods: a model is created which is abstract enough to allow easy simulation, but still detailed enough to reflect the relevant timing aspects of the system (see [3]–[9]). The advantage is that nearly all constraints can be modeled in short time, enabling us to evaluate a system in a fast, flexible way. The main disadvantage of a simulation approach lies in the fact that one carefully has to prepare the simulation, otherwise the worst-case scenarios are not triggered, and the result might then be too optimistic.

IV. Need for Theoretical Understanding

As it was shown in the previous chapters, it is obviously possible to design systems where the state of the art methods of schedulability analysis can not deliver a valid result while the system is behaving well. Today, several state of the art techniques like e.g. EDF scheduling are available to be introduced in the automotive powertrain domain [10] for coping with the the challenges described above. Besides the application of existing concepts also new approaches are required.

In order to manage the development and quality control for such systems, several additional research fields would be of interest.

First of all, design principles need to be developed which allow to design robust systems that are able to work up to very high processor loads without failure due to execution time problems. This may for example involve special scheduling strategies, design patterns for algorithms, a more flexible approach for classifying and handling of deadlines, e.g. by allowing sporadic, i.e. non successive, deadline misses, or concepts how to distribute processes to tasks or CPU cores. Also research to evaluate requirements regarding their consequences for the robustness of the timing behavior would be extremely helpful, as this may help to influence the engineering of the system in a very early phase. Depending on certain properties, some functional requirements will not only consume the resources needed for their implementation, but will also increase the required resources (typically

calculation power) that have to be kept as a reservoir to ensure a reliable and safe operation. Finally, new concepts for validating systems are needed, e.g. by introducing probabilistic testing approaches, or by defining metrics that show the degree of dynamic robustness of the system. Such research would also allow to compare different project configurations and thus enable the definition of quality thresholds and successive improvement of the software as described in [11].

In general, for domains like automotive powertrain new strategies are required for creation of efficient, high load systems above 90% average CPU load that are at the same time safe, reliable *and* efficient. The focus needs to be on robust systems that are capable of handling small disturbances and a certain amount of “glitches” in the system gracefully, rather then on proving 100% error-free and “feasibility under all possible conditions”. Here, maybe, the embedded systems need to learn from biological systems which are far from being perfect — but are often astonishingly reliable, robust and efficient.

Acknowledgment

We thank Denis Claraz and Annette Kempf for inspiring discussions and helpful comments.

References

[1] N. Audsley, “Optimal priority assignment and feasibility of static priority tasks with arbitrary start times,” *Real-Time Systems*, 1991.

[2] S. Manolache, *Schedulability analysis of real-time systems with stochastic task execution times*. Department of Computer and Information Science, Linköpings universitet, 2002.

[3] INCHRON GmbH, “chronSim,” http://www.inchron.de/. [Online]. Available: http://www.inchron.de/

[4] R. Münzenberger, M. Dörfel, U. Margull, and G. Wirrer, “Entwurf echtzeitfähiger Steuergerätesoftware in FlexRay-Netzwerken,” in *KFZ-Entwicklerform Design&Elektronik*, 2007.

[5] M. Deubzer, U. Margull, J. Mottok, M. Niemetz, and G. Wirrer, “Partitionierungs-Scheduling von Automotive Restricted Tasksystemen auf Multiprozessorplattformen,” in *Proceedings of the 2nd Embedded Software Engineering Congress*, ser. ISBN 978-3-8343-2402-3, December 2009, pp. 536–542.

[6] M. Deubzer, J. Mottok, U. Margull, and M. Niemetz, “Efficient Scheduling of Reliable Automotive Multi-core Systems with PD^2 by Weakening PFAIR Tasksystem Requirements,” in *Proceedings of the Automotive Safety & Security 2010 (Accepted for Publishing)*, June 2010.

[7] M. Deubzer, F. Schiller, J. Mottok, M. Niemetz, and U. Margull, “Effizientes Multicore-Scheduling in Embedded Systemen: Teil 1 - Multicore-Scheduling für zuverlässige Echtzeitsysteme,” 2010.

[8] ——, “Effizientes Multicore-Scheduling in Embedded Systemen: Teil 2 - Ein simulationsbasierter Ansatz zum Vergleich von Scheduling-Algorithmen,” 2010.

[9] R. Davis and A. Burns, “A survey of hard real-time scheduling algorithms and schedulability analysis techniques for multiprocessor systems,” University of York, Department of Computer Science, techreport YCS-2009-443, 2009.

[10] U. Margull, M. Niemetz, and G. Wirrer, “Improved Schedulability using EDF in Automotive Powertrain Software,” to be published.

[11] 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 *Date 2009*, 2009.