Modeling and Simulation of Embedded Real-Time Multicore Systems

Michael Deubzer[1], Martin Hobelsberger[1], Jürgen Mottok[1], Frank Schiller[2], Reiner Dumke[3], Markus Siegle[4], Ulrich Margull[5], Michael Niemetz[6], Gerhard Wirrer[6]

[1]University of Applied Sciences Regensburg,
LaS³, Faculty Electrical Engineering, P.O. Box 120327, D-93025 Regensburg,
{michael.deubzer,martin.hobelsberger,juergen.mottok}@hs-regensburg.de

[2]Technical University Munich,
itm, Boltzmannstr. 15, D-85748 Garching (Munich), schiller@itm.tum.de

[3]Otto-Von-Güricke University,
IVS, P.O. Box 4120, D-39106 Magdeburg, dumke@ivs.cs.uni-magdeburg.de

[4]University of German Federal Armed Forces,
Dept. of C. Sc., Werner-Heisenberg Weg 39, D-85577 Neubiberg, markus.siegle@unibw.de

[5]1 mal 1 Software GmbH,
Maxstraße 31, D-90762 Fürth, ulrich.margull@1mal1.com

[6]Continental Automotive GmbH,
P.O. Box 100943, D-93009 Regensburg
{michael.niemetz,gerhard.wirrer}@continental-corporation.com

In order to handle the continuously growing functionality in embedded systems, model-based approaches have been established in the development process over the last years. For singlecore architectures, model-based development has increased efficiency of development and software quality. Current trends show that embedded systems will change to multicore architectures within this decade.

In this paper, we argue for a work-efficient migration from single- to multicore architectures through a model-based development process. Concerning the modeling of software and hardware components, we propose relevant properties of an architecture description language. Regarding the design phase, we show how simulation technology can be used for schedulability examination supporting engineers for system design decisions. Afterwards we extend the simulation approach by Monte-Carlo parametrization. Finally we evaluate the enhanced approach in a case study, located in the automotive powertrain domain.

Keywords: Real-Time Systems, Multicore, Model-Driven Development, Architecture Description Language, Simulation

1 Introduction

The embedded systems domain, like the desktop computing domain, is facing the challenge of a rapid increase in software functions and software complexity of their applications. While the desktop computing domain already has changed to multicore architectures to satisfy this demand, the embedded domain is on the way to follow. Since the introduction of multicore processors to embedded systems in the year 2006, the demand in the embedded domain has raised drastically [12].

In embedded systems, multicore processor architectures increase the complexity of embedded software. Furthermore multithreading aggravates the issue of complexity due to the complex interactions among multiple threads [16].

Two of the main properties of embedded systems are temporal requirement satisfaction, namely deadline compliance, and resource efficiency, namely processing and memory capacity. For being able to fulfill both properties for future embedded systems, which will employ multicore technology, the software architecture design decisions are fundamental.

In order to alleviate the problem of high complexity in embedded software systems, which will become even more essential for multicore embedded software design, and have a meaningfull way to express architecture design decisions, we argue for a model-based software development process.

In this work, we discuss the required system model and an appropriate schedulability examination method to achieve these objectives.

In the following, we present a possible software architecture model for future embedded multicore systems. Afterwards we extend necessary properties of architecture description languages. These are used as input for a simulation based schedulability examination, which examines the compliance of temporal requirements. We give a brief overview on theoretical work regarding schedulability tests and why these cannot be applied in the same way for multicore systems as for today's singlecore systems. Then we show how a Monte-Carlo approach can be combined with simulation studies to evaluate software design decisions. Finally, we demonstrate the practical use of our approach by an evaluation of adequate scheduling algorithms for a quadcore system in the automotive powertrain domain.

2 A Model for Multicore Real-Time Systems

The accuracy of the analysis of an embedded system's temporal properties depends on the precision of the system model. This system model includes a task set model τ, a processor model M, and a scheduler model S.

The task set model τ describes the temporal properties of the software application. These properties are the activation behavior, the execution time, and temporal requirements.

Most task set models define the activation behavior in a way that the activation of each task is independent of other tasks. However, many embedded systems have multiple time bases, e.g. Flexray, CAN, or other time bases[1], also referred to as clocks. Task activation patterns are defined concerning this time base. Time base variation results in changing activation patterns of tasks in global time. Additionally, the limitation of time base variation limits the number of possible activation patterns in global time. The proposed multiple time base (MTB) task set model [7] allows to model these task activation dependencies. In the following, we discuss the

[1] E.g. in an automotive powertrain system the position of the crank shaft.

MTB model by use of an example.

In a typical automotive engine control system, two main sources of task activation exist. The first source is a periodic trigger, which activates tasks with different constant recurrences. The other source is the crank shaft of the engine, which activates tasks depending on the engine rotation speed. Analyzing such systems with a periodic task set model[17] is not possible, because the drifting behavior of the crank shaft activated tasks is not represented. Analyzing such systems with a sporadic task set model [19] will produce too pessimistic results, because the sporadic theorem assumes all tasks to be activated at the same time, which is, in a typical automotive engine control system, prevented by task offsets.

At a MTB task set model, tasks refer to a time base of the system. All tasks concerning the same time base have a defined (but possible variable) phasing in their activation compared to all other tasks referring to this time base.

A task set of tasks, belonging to the MTB task system is defined in the following way:

Definition 2.1 *A task set τ consists of a number of tasks T_i.*

$$\tau = \{T_i\} \quad i = 1,\dots,n; \;\; n \in \mathbb{N}$$

Definition 2.2 *A task T_i is defined by a tuple*

$$T_i = (p, o, e, d, b^v),$$

where the elements of the tuple are the task properties: minimal task recurrence p, first task instance offset o, worst-case execution time e, deadline d, and a reference to a time base b^v.

Definition 2.3 *A time base b^v is defined by the tuple*

$$b^v(t) = (f(t), \varphi(t)) \quad (v = 1,\dots,w; \;\; w \in \mathbb{N}).$$

The properties describe the transformation between the clock of the time base $t_{b^v}(t)$ and the clock of the unique global time $t = [0,\infty)$. The frequency multiplier $b^v.f(t)$ equates the clock ratio $\frac{t_{b^v}}{t}$ and the angular phase shift $b^v.\varphi(t)$ equates an incremental time shift between time base time and global time. For simplification, at time $t = 0$: $t = t_{b^v} = 0$ for all b^v. For time $t > 0$, following transformation has to be applied:

$$t_{b^v}(t) = b^v.f(t) * t + b^v.\varphi(t). \tag{1}$$

Definition 2.4 *For the time base properties, the following restrictions exist:*

$$b^v.f(t) \in \mathbb{R}^{\geq 1} \quad \forall \; t$$

$$b^v.\varphi(t) \in \mathbb{R}^{\geq 0} \quad \forall \; t$$

By definition, the frequency multiplier $b^v.f(t)$ cannot be smaller then 1, therefore $T_i.p$ is the minimal recurrence and can be used for calculation of task utilization. Through a shared reference of two tasks T_a and T_b on the same time base b^x, the activation of both is no longer independent. Assuming a periodic activation in relation to the time base, the inter-arrival time of both tasks changes by the same factor $b^x.f(t)$ or shifts with the same value $b^x.\varphi(t)$.

A further extension of the MTB task set model is a split of the task T_i into task sections T_i^k.

Definition 2.5 *A task T_i is split into a number of task sections T_i^k $(k = 1,\dots,q;\ q \in \mathbb{N})$. According to the task execution time $T_i.e$ and the task section execution time $T_i^k.e$ the following relation exists.*

$$T_i.e = \sum_{k=1}^{q} T_i^k.e$$

Definition 2.6 *All task sections are sequentially dependent. Therefore, task section T_i^b cannot be executed before the task section T_i^a has finished its execution, if $a < b$ and $a, b \in \{1,\dots,q\}$.*

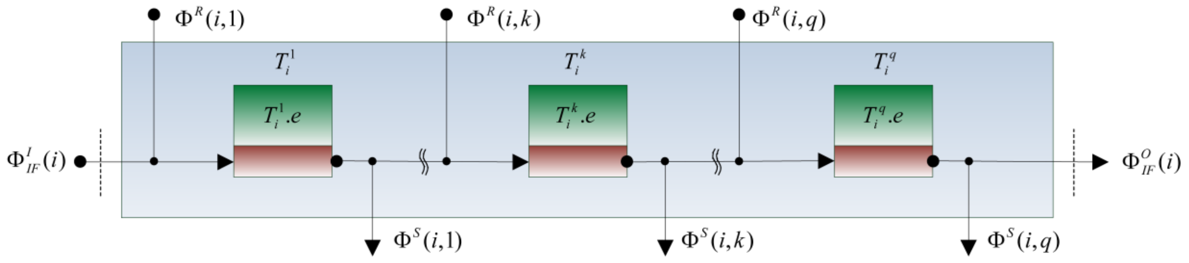


Figure 1: Visualization of a task T_i with its task sections T_i^x and its requested data $\Phi^R(i,x)$ and sent data $\Phi^S(i,x)$ for the task sections $x = \{1,\dots,k,\dots,q\}$.

Figure 1 visualizes a task, its task sections, and the data a task section consumes and produces. This task section architecture has a benefit in comparison to common task architectures, because the position where data is produced and consumed during task execution can be explicitly modeled on task section level. The idea behind this approach assumes on the one hand an arbitrary amount of local data during the execution of a task section. On the other hand, it assumes a lower amount of inter-task-section data sharing. For the case of cooperative[2] multicore scheduling, e.g. used in the algorithm *P-ERfair-PD²* [7], this is beneficial with respect to context switching overheads in comparison with preemptive[3] scheduling algorithms, e.g. global EDF.

For the processor model, we assume a symmetric processor model, mentioned as multicore M. A multicore $M = \{P_x\}$ $(x = 1,\dots,m;\ m \in \mathbb{N})$ has m identical processing resources P_x. Each processor P_x has the same processing frequency and all tasks of the task set τ are potentially able to execute on all P_x. Further extensions include core-specific processing frequencies, used for dynamic voltage/frequency scaling.

The scheduler model S describes the assignment of all jobs[4], generated by tasks of the task set τ at activation, to the processor M. Multicore real-time scheduling generally distinguish between local and global scheduling approaches. At local scheduling, a heuristic allocates tasks before runtime to a core, subsequently a singlecore scheduling approach can be applied. At global scheduling, there is one single queue of ready tasks and the scheduling approach assigns them during runtime to available cores. Both groups can be subdivided according to priority

[2]Cooperation means that a task is only able to preempt other task at task section end.

[3]Preemptive means that a task is able to preempt other task at arbitrary position.

[4]A job is the instance of a task, generated at task activation.

assignment policies. At task-fix scheduling each job of a task has the same priority. At job-fix scheduling, the jobs of a task can have different priorities, but the priority of one job is static. At job-section-fix scheduling the priority of different task sections can differ, but for one task section the priority is static. At dynamic scheduling, the priority can differ at any time. For further classification see [8].

The next chapter gives a brief overview of appropriate architecture description languages for embedded systems. Then we discuss how these could be extended to have a common format for multicore system models.

3 Extending Architecture Description Languages

A key challenge in the development of embedded systems is that of managing their complexity while ensuring the required quality. As one important means to handle complexity, architecture description languages (ADLs) have emerged as a means to formally describe software and hardware architectures. Furthermore, they provide a basis for the analysis of system properties such as reliability or performance.

Over the recent years a number of solutions for the modeling and description of embedded systems have been developed including

- UML [22], SysML [21] and MARTE [18] developed and standardized by the Object Management Group (OMG)
- EAST-ADL [6] developed in the automotive industry
- the SAE Architecture Analysis and Design Language (AADL) [20] standardized in 2004 by the SAE
- and various domain-specific languages (DSL) and model transformation techniques.

With the introduction of multicore systems to the embedded domain, new requirements for the modeling and description of embedded systems emerge. These requirements include:

- modeling of multicore hardware components including heterogeneous cores with variable processing speed,
- comprehensive execution time models, e.g. expressed by probability distributions,
- support for the annotation of scheduling policies.

While general modeling languages (e.g. UML or SysML) could be extended to support the modeling and description of embedded multicore systems [15], we decided to adapt a domain-specifc modeling language for the use in our approach. The decision is based on the flexibility of the SAE AADL standard to extend the language via specific properties and sublanguage extensions as well as the various analysis capabilities the SAE AADL supports for a single specified model. Furthermore, the SAE AADL is an acknowledged industry-standard for the modeling and analysis of applications and execution platforms in the embedded domain. In the following, a brief introduction to the SAE AADL is provided. Afterwards, we describe extensions to the language, which support the simulation approach in Section 4.

3.1 SAE AADL

The SAE Architecture Analysis and Design Language (AADL) [20] is a textual and graphical language used to model and analyze the software and hardware architecture of embedded systems. It describes the structure of such systems as an assembly of software components mapped onto a hardware platform. Furthermore it is used to describe functional interfaces to components (such as data inputs and outputs) and performance-critical aspects of components (such as temporal requirements), which is a crucial requirement for the analysis of real-time systems.

The standard does not specify how detailed the design of the architecture or the implementation of software and hardware components has to be. It allows different levels of abstraction within one model. Furthermore, AADL may be used in conjunction with existing standard languages in these areas (e.g. via an existing UML profile). The AADL describes interfaces and properties of hardware components including processors, memory, communication channels, and devices interfacing with the external environment. Detailed designs for such hardware components may be specified by associating source text written in a hardware description language such as VHDL. The AADL can describe interfaces and properties of application software components implemented in source text, such as threads, processes, and runtime configurations.

The language includes a standardized XML interchange format based on a Meta model specification of AADL to facilitate model interchange and integration of analytical models and supporting tools.

The purpose of the SAE AADL is to provide a standard and sufficiently precise (machine-processable) way of modeling the architecture of an embedded real-time system, such as an automotive system or avionic system, to permit analysis of its properties and to support the predictable integration of its implementation [13]. It provides a framework for system modeling and analysis, facilitates the automation of code generation and other development activities, and aims to significantly reduce design and implementation errors.

The AADL core language is designed to be extensible to accommodate analyses of the runtime architectures that the core language does not completely support. Extensions can take the form of new properties and analysis specific notations or unique hardware attributes that can be associated with components.

3.2 Property Extensions to SAE AADL

In this section, we present selected properties of the newly defined property set for the modeling of embedded multicore systems.

While SAE AADL provides means for modeling the hardware platform (e.g. processor or memory), it does not provide the possibility for the modeling or mapping of software components, tasks or interrupt service routines, onto a number of cores of a multicore processor.

We extended the properties of the standard execution platform component processor to support the modeling of heterogeneous multicores[5].

In Listing 1, an excerpt of the definition of the new properties is shown. In addition to the *Cores* property that defines a core of a processor, the possibility of adding a quartz to each core

[5]A further extension is the mapping of software components to resource owner which provide means to distribute software components to cores. Due to the space limitations of this paper, this will be considered in future work.

(*Quartz*), as well as defining the quartz frequency and the core instructions per quartz tick are defined. The mapping allows to assign quartzes to specific cores. Additionally, it is possible to configure scheduling policies.

```
property set multicoreEXT is

  Quartz: aadlinteger applies to (device);

  Cores: aadlinteger applies to (processor);

  Core_Instructions_per_Tick: aadlinteger  applies to (processor);

  Quartz_Frequence_Hz: aadlinteger applies to (device);


  -- Scheduling

  Scheduling_Protocol: type enumeration (osek, edf, global_edf, pf_pd2, pf_er_pd2, partly_pf_pd2, p_er_pd2,
       edzl, llref);

  -- Mappings
  -- Mapping Quartz to Core
  Quartz_Core_Binding: inherit reference(processor) applies to (device);

  .
  .
  .

end multicoreEXT;
```
Listing 1: Selected multicore extension of SAE AADL

With these additional properties and the standard scheduling annotations of the SAE AADL, the modeling of a multicore system is possible. In Listing 2 an example modeling of the hardware is shown. The processor *multicore.dualcore* has two cores which each have a mapping to the same quartz. The cores (*Cores.core0/1*) can be annotated with specific properties as defined in Listing 1.

```
— ** marker for the transformation to identify the component as relevant input for the simulation

— multicore processor
 system implementation multicore.dualcore
   subcomponents
     core0: processor Cores.core0;
     core1: processor Cores.core1;
     quartz0: device Quartz.quartz0;
   properties
     — Mapping core to quartz
     multicoreHW::Sim_Processor => 1; — **
     multicoreHW::Quartz_Core_Binding =>  reference core0 applies to quartz0;
     multicoreHW::Quartz_Core_Binding =>  reference core1 applies to quartz0;
 end multicore.dualcore;

— processor−core
  processor implementation Cores.core0
    properties
      multicoreHW::Core => 1; — **
      multicoreHW::Core_Instructions_per_Tick => 10;
  end Cores.core0;

— processor−core
  processor implementation Cores.core1
    properties
      multicoreHW::Core => 1; — **
      multicoreHW::Core_Instructions_per_Tick => 20;
  end Cores.core1;

— quartz
  device implementation Quartz.quartz0
    properties
      multicoreHW::Quartz => 1; — **
      multicoreHW::Quartz_Frequence_Hz => 100000000;
  end Quartz.quartz0;
  .
  .
  .
```

Listing 2: Modeling of a dualcore processor in SAE AADL

After the system is modeled in SAE AADL and annotated with the multicore properties, it can be used as an input for the scheduling simulation. In addition to the scheduling analysis, this specified single SAE AADL model can be analyzed for multiple qualities e.g. availability and reliability, security or resource consumption.

4 Simulation-based Schedulability Examination at Software Design Phase

Schedulability tests concern the problem of testing, whether a scheduling algorithm S is able to assign a task set τ on a certain processor architecture M in a way that all deadlines are met.

For singlecore or multiple processor systems, a multitude of schedulability tests are available. In addition, these tests are also available for complex event task activation patterns which are expressed by arrival curves [5] or hierarchical inter-task activation [1]. These tests are designed based on two assumptions: The critical instant (CI) theorem, originally introduced by [2], roughly speaking says that the worst-case response time occurs when all tasks are activated simultaneously. The worst-case execution time (WCET) assumption says, if a task set fulfills all deadlines with its worst-case execution time, it also fulfills all deadlines when the execution times are smaller.

These assumptions allow to reduce the validation space to determine the worst case response time. They are valid for singlecore systems with task-fix and job-fix priority assignment.

For multicore systems, no finite collection of worst-case job arrival sequences has been identified for global scheduling of sporadic task systems [4]. For dynamically and globally scheduled multicore systems, it is not guaranteed that the worst case response time happens under WCET and CI assumption [14].

Therefore, analytical schedulability approaches, which are available for singlecore systems, are not applicable for multicore systems with global dynamic scheduling algorithms. For task systems with small integer values for task periods and task execution times, Baker et al. [3] introduced a brute-force approach, which performs an exhaustive search of a very large state space to determine whether a sporadic task set is schedulable by a global fixed task-priority or global EDF scheduling. Unfortunately, this approach results in a state explosion when analyzing practical systems due to a higher granularity of task execution times and task periods.

In [9] an alternative approach based on a discrete event simulation was introduced. During the simulated time, a task T_i generates a number of jobs (i.e. instances) $T_{i,j}$, which are assigned to cores by a scheduler model. To examine the schedulability of a task set τ, job release times and job execution times are modified in their valid range to approximate the worst case response times.

The valid range of the release time depends on the task arrival model and the time base of the task. Since the arrival model[6] defines a task recurrence $T_i.p$ in the time of the time base, the transformation of equation (1) has to be applied to activate a task at the correct global time.

The valid range of the execution time depends on the execution time model. At the classical WCET model, the job execution time $T_{i,j}.e = T_i.e$ for all jobs $T_{i,j}$, whereas $T_{i,j}$ is the j-th instance of the i-th task. Because the maximal execution time does not necessarily represent the worst case at global and dynamic scheduled multicore systems[7], the complete range of execution times has to be considered. In the simulation, the execution time variation occurs with respect to a probability function for each task section execution time $T_i^k.e$, which has at least an upper and lower bound of execution times.

As result of the simulation, a trace contains all state transitions of the simulation. This trace is required for the schedulability examination. We consider the transitions when the j-th job of the i-th task has its finishing time $F_{i,j}$ and when the job has its deadline $D_{i,j}$. With these time stamps, the compliance of all temporal requirements can be analyzed by determining the maximum normed lateness mNL of a task set τ.

First of all the lateness $l(T_{i,j})$ of the j^{th} job of task T_i is calculated by

$$l(T_{i,j}) = F_{i,j} - D_{i,j} \ < 0 \quad when\ deadline\ is\ met.$$

$l(T_{i,j})$ is equivalent to the time left until reaching the deadline. The lateness is negative when the finishing time $F_{i,j}$ is smaller than the absolute deadline $D_{i,j}$, i.e. if the task has finished in time.

To determine the task-deadline compliance for a complete task set, we normalize the lateness $l(T_{i,j})$ with the relative deadline d_i of the task T_i and identify the job which yields the largest normalized lateness for each task. We denote the maximum of that value of all tasks in a task set τ as maximal normed lateness $mNL(\tau)$.

$$mNL(\tau) = \max_{T_{i,j}} \left(\frac{l(T_{i,j})}{d_i} \right) \tag{2}$$

[6]The recurrence p can be used to represent a periodic offset based arrival model from Section 2, or to represent the inter arrival time between two successive activation of any other arrival model.

[7]This is also mentioned as non-predictability, see [14] for details.

5 Statistical Analysis of Randomized Parameter Sets

The introduced method for simulation-based schedulability examination analyzes a system model with one task set τ, one multicore processor M, and one scheduler S. Design decisions on one of these three components often have to fit for multiple variants of the other components. For example, the task set τ differs from project to project, but the selection of processor or scheduler should be the same for all projects.

In [9], a statistical method for sensitivity analysis of system characteristic was introduced. In this method, based on stochastical description of system components and Monte-Carlo parametrization, randomized parameter sets are generated and simulated.

In this work, we use this approach to compare two scheduling approaches for multicore systems. We apply a stochastic task set description which generalizes todays automotive powertrain applications, scheduled on a quad-core processor ($m = 4$). The case study compares two scheduling algorithms with regard to the maximal deadline violation for each task set, expressed through $mNL(\tau)$.

For the sensitivity analysis, we apply the following process:

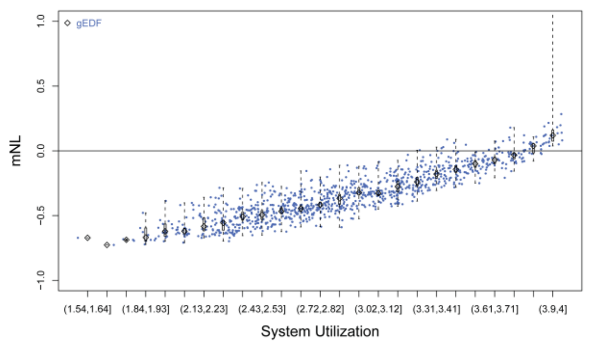
- Pseudo random generation of task sets $\{\tau\}$ according to stochastic task set description.
- Simulation and determination of quality metric (e.g. $\{mNL^S(\tau)\}$ for deadline compliance evaluation) for all generated task sets $\{\tau\}$ and for all scheduling algorithms S.
- Clustering of task sets according to a system characteristic (e.g. utilization $U(\tau) = \sum_i \frac{e_i}{p_i}$ for efficiency evaluation) in equally sized clusters over the task set characteristic.
- Determination of statistical estimators for clustered system characteristic (e.g. upper 1% quantil, median, lower 99% quantil) and calculation of confidence intervals with bootstrapping approach [11]
- Visualization of the results in a diagram

6 Case Study

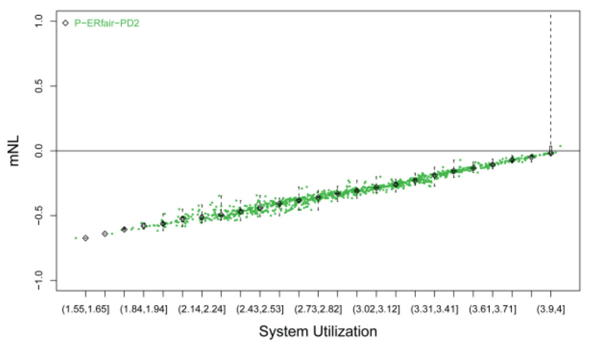
The objective of this case study is a comparison of the two global scheduling algorithms *global EDF* (gEDF) and *P-ERfair-PD²* for multiple task sets, originating from a stochastic task set description.

A comparable study on a dual-core processor [9] showed that *P-ERfair-PD²* a global multicore scheduling algorithm with job fix priorities, allows a system utilization of nearly 1.96, whereas *WFD-EDF*, a local scheduling algorithm with Bin-Packing heuristic and job fixed priority scheduling, has a multitude of deadline violations, when system utilization exceeds 1.65.

In this work, we use the stochastic task set definition from [7], with an extension of scheduling execution times of 2 μs for each scheduler call. We compare gEDF with *P-ERfair-PD²* multicore scheduling on a quad-core processor. gEDF stores ready tasks in a single queue and assigns priorities by Earliest Deadline First manner. EDF is known to be non-optimal for the global case. Reasons are certain task sets that fail at very low processor utilizations, essentially leaving all but one processor idle nearly all of the time, also known as Dhall's effect [10].



Algorithm gEDF



Algorithm P-ERfair-PD²

Figure 2: Sensitivity analysis of schedulability of 1000 randomly generated task sets. The x-axis shows system utilization and the y-axis shows $mNL(\tau)$. As long $mNL(\tau) \leq 0$, all deadlines are met.

The experiment results in Figure 2 show the $mNL(\tau)$ of 1000 randomly generated task sets. We generated task sets according to a stochastic task set description, but didn't discard task sets fulfilling Dhall's effect. The utilization $U(\tau) = \sum_{T_i \in \tau} \frac{T_i.e}{T_i.p}$ lies in the range $(1.54,4]$, whereby 4 is equal a quad core system utilization of 100%. For statistical analysis, we divided the range of derived utilization values in 25 equally sized clusters.

At gEDF, the first deadline violations enter at a utilization of 3.21, whereby P-ERfair-PD² successfully schedules all task sets up to a utilization of 3.9. Additionally, the range of $mNL(\tau)$ variations is much lower at P-ERfair-PD², which indicates a higher predictability of schedulability at task set variation.

P-ERfair-PD² has in average 2.25 times more scheduler calls[8] than gEDF. For the considered stochastic task set, P-ERfair-PD² has a schedulability bound which is 21,5% higher than gEDF's schedulability bound. Especially at high utilization, P-ERfair-PD² has a benefit, because it has more information of the taskset (namely minimal task activation and task execution time), and therefore is able to schedule jobs better than gEDF (which knows only task deadlines).

7 Conclusion

In this paper, we presented a possible software architecture model for embedded multicore systems, based on a multiple time base task set model, which allows the modeling of task activation dependencies. To model this software architecture with an industry-standard language notation, we extended the SAE AADL with properties to support the modeling and specification of embedded multicore systems. This AADL model can be used as an input for an automated simulation-based schedulability examination and supports, without further adaption, a number of additional architectural analyses. We showed how a Monte-Carlo approach can be combined with scheduling simulation to evaluate software design decisions for multicore systems. In a case study, we verified the practical application of our approach by a comparison of global multicore scheduling algorithms.

[8]We chose for both scheduling algorithms an equal execution time, because the number of schedulable jobs is equal at both algorithms and both algorithms work with fixed job priorities (At P-ERfair-PD ^{2}, scheduling policies can simply be mapped statically to task sections).

Acknowledgment

This work is supported by BMBF grant DynaS³ ("Dynamische SW-Architekturen in Steuergeräten in Fahrzeugsystemen unter Berücksichtigung von Anforderungen zur Funktionalen Sicherheit") FKZ1752X07. More information at www.LaS3.de.

References

[1] Albers, K. and Bodmann, F. and Slomka, F. Hierarchical event streams and event dependency graphs: A new computational model for embedded real-time systems. *18th Euromicro Conference on Real-Time Systems*, pages 10, 2006.

[2] Audsley, N. C. and Burns, A. and Richardson, M. and Tindell, K. and Wellings, A.J. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284, 1993.

[3] Baker, T. P. and Cirinei, M. Brute-force determination of multiprocessor schedulability for sets of sporadic hard-deadline tasks. *10th International Conference on Principles of Distributed Systems*, (0509131):62-75, 2007.

[4] Baruah, S. K. Techniques for Multiprocessor Global Schedulability Analysis. *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, :119-128, 2007.

[5] Chakraborty, S. and Kuenzli, S. and Thiele, L. A general framework for analysing system properties in platform-based embedded system designs. *Proc. 6th Design, Automation and Test*, :190-195, 2003.

[6] Cuenot, P. and Chen, D.J. and Gérard, S. and Lönn, H. and Reiser, M.O. and Servat, D. and Sjöstedt, C.J. and Kolagari, R.T. and Törngren, M. and Weber, M. Managing complexity of automotive electronics using the East-ADL.

[7] Deubzer, M. and Mottok, J. and Margull, U. and Niemetz, M. Efficient Scheduling of Reliable Automotive Multi-core Systems with PD² by Weakening ERfair Task System Requirements. *Proceedings of the Automotive Safety & Security 2010*, pages 5--67, 2010.

[8] Deubzer, M. and Schiller, F. and Mottok, J. and Niemetz, M. and Margull, U. Effizientes Multicore-Scheduling in Eingebetteten Systemen - Teil 1: Algorithmen für zuverlässige Echtzeitsysteme. *atp, Automatisierungstechnische Praxis*, 2010.

[9] Deubzer, M. and Schiller, F. and Mottok, J. and Niemetz, M. and Margull, U. Effizientes Multicore-Scheduling in Eingebetteten Systemen - Teil 2: Ein simulationsbasierter Ansatz zum Vergleich von Scheduling-Algorithmen. *atp, Automatisierungstechnische Praxis,* 2010.

[10] Dhall, SK. Scheduling periodic-time-critical jobs on single processor and multiprocessor computing systems. *PhD thesis*, University of Illinois at Urbana-Champaign Champaign, 1977.

[11] Efron, B. and Tibshirani, R. *An introduction to the bootstrap*. Chapman & Hall/CRC, 1993.

[12] Heikkila, E. and Gulliksen, J. E. Multi-Core Computing in Embedded Applications: Global Market Opportunity and Requirements Analysis. *Venture Development Corporation Embedded Hardware and Systems Practice*, 2007.

[13] Feiler, P.H. and Lewis, B.A. and Vestal, S. The SAE architecture analysis & design language (aadl) a standard for engineering performance critical systems. *2006 IEEE Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, pages 1206-1211, 2006.

[14] Ha, R. and Liu, J.W.S. Validating timing constraints in multiprocessor and distributed real-time systems. *International Conference on Distributed Computing Systems*, pages 162-162, 1994. Citeseer.
[15] Hsiung, P.A. and Lin, S.W. and Tseng, C.H. and Lee, T.Y. and Fu, J.M. and See, W.B. VERTAF: An application framework for the design and verification of embedded real-time software. *Software Engineering, IEEE Transactions on*, 30(10):656-674, 2004.
[16] Lee, E.A. The problem with threads. *Computer*, 39(5):33-42, 2006.
[17] Liu, C. L. and Layland, J. W. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20(1):46-61, 1973.
[18] MARTE OMG. Specification. A UML Profile for MARTE, Beta 1. 2007.
[19] Mok, A.K.L. *Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment*. PhD thesis, Massachusetts Institute of Technology, 1983.
[20] SAE. Architecture Analysis & Design Language. *SAE standard AS5506*, 2004.
[21] SysML OMG. Specification 1.0. 2008.
[22] UML OMG. specification 2.0., *http://www.Omg.org/technology/documents/formal/uml.htm*, 2005.

Authors

Dipl.-Ing. (FH) Michael Deubzer is research assistent at the Laboratory for Safe and Secure Systems (www.LaS3.de) working in the research project DynaS³. He is a PhD student at Technical University Munich.

Dipl.-Inf. (FH) Martin Hobelsberger is research assistent at the Laboratory for Safe and Secure Systems (www.LaS3.de) working in the research project DynaS³. He is a PhD student at Otto-von-Guericke-University Magdeburg.

Prof. Dr. rer. nat. Jürgen Mottok is professor of software engineering, computer languages, operating systems and safety at the University of Applied Sciences Regensburg. He is the head of the Laboratory for Safe and Secure Systems (www.LaS3.de).

Prof. Dr.-Ing. Frank Schiller is professor of automation engineering at the chair of information technology in mechanical engineering at the Technical University Munich. His reaseach interests are in the field of modeling and simulation of mechatronic systems as well as reliability and safety analysis of mechatronic systems and automotive software systems.

Prof. Dr.-Ing. Markus Siegle is professor of computer science at the Universität der Bundeswehr München. He is working on methods and tools for model-based performance and dependability evaluation of computer and communication systems.

Prof. Dr.-Ing. habil. Reiner Dumke is professor of software engineering and head of the Institute for Distributed Systems at the Otto-von-Guericke-University Magdeburg.

Dr. rer. nat. Michael Niemetz is expert for software architecture at the Continental Automotive GmbH in Regensburg at the Powertrain Engine Systems Engineering division.

Dipl.-Ing. (TU) Gerhard Wirrer is the head of the function- and software architecture department at the Continental Automotive GmbH at the Powertrain Engine Systems Engineering division.

Dr. rer. nat. Ulrich Margull is the director of the 1 mal 1 Software GmbH. He counsels concerns in software architecture and real-time system related questions.