Aalborg Universitet

AALBORG UNIVERSITY
DENMARK

Schedulability and Energy Efficiency for Multi-core Hierarchical Scheduling Systems

Boudjadar, Abdeldjalil; David, Alexandre; Kim, Jin Hyun; Larsen, Kim Guldstrand; Nyman, Ulrik Mathias; Skou, Arne Joachim

Published in:
Proceedings of ERTS2 2014

Publication date:
2014

Document Version
Early version, also known as pre-print

Link to publication from Aalborg University

Citation for published version (APA):
Boudjadar, J., David, A., Kim, J. H., Larsen, K. G., Nyman, U., & Skou, A. (2014). Schedulability and Energy Efficiency for Multi-core Hierarchical Scheduling Systems. In Proceedings of ERTS2 2014 (pp. 1-4)

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

? Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
? You may not further distribute the material or use it for any profit-making activity or commercial gain
? You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from vbn.aau.dk on: februar 18, 2019

Schedulability and Energy Efficiency for Multi-core Hierarchical Scheduling Systems

A.Jalil Boudjadar, Alexandre David, Jinhyun Kim, Kim. G. Larsen, Ulrik Nyman, Arne Skou
Institute of Computer Science, Aalborg University, Denmark

***Abstract*—We propose a framework for modeling and analyzing the schedulability and energy efficiency of embedded hierarchical scheduling systems running on a multi-core platform. The framework is realized using Hybrid Timed Automata describing the concrete task behavior. The schedulability can be verified in a compositional way using UPPAAL, and the energy profile can be generated using the statistical model checking algorithms of UPPAAL SMC. To our knowledge, our paper is the first one considering hierarchical scheduling, multi-core platforms and energy consumption simultaneously. The framework is being applied to a case-study from the CRAFTERS project.**

I. INTRODUCTION

Embedded systems is an essential part of many modern products including complex safety critical real-time systems. Within the industrial domains of avionics and automotive, the safe composition of several embedded features within the same systems can be achieved through the use of hierarchical scheduling. The separation between features is secured by using time partition scheduling at the system level [7]. A trend within embedded systems is to use multi-core platforms in order to increase performance and to be able to implement more functionality within one embedded system.

In this paper we propose an approach to analyzing both the schedulability and the energy consumption of hierarchical systems on embedded multi-core platforms.

In the literature, a large amount of work has been devoted to the description and analysis of scheduling systems [9] together with energy efficiency [8]. However none of these papers deals with important aspects of modern systems like (1) powerful execution platforms which are based on multi-core technology, (2) hierarchy of system architecture resulting from the component-based design, and (3) concrete task behavior which consists of a set of operations having different energy consumption rates. In all of the previous work systems can be viewed as a set of abstract components competing for CPU and other resources, and having a uniform distribution of energy consumption. In contrast, our framework enables modeling concrete task behavior and differentiated energy consumption rates based on task state.

The rest of the paper is structured in the following way: Related work is discussed in section II. Section III first gives a general overview of our approach followed by a detailed description of how the approach is modeled in terms of Constant Slope Timed Automata (CSTA) and analyzed using Uppaal. Finally the conclusion is given in section V.

II. RELATED WORK

Compositional framework for hierarchical scheduling systems was initially presented in [11] by Shin and Lee as a formal way to elaborate a compositional approach for schedulability analysis of hierarchical scheduling systems [12]. In [10], the authors dealt with a hierarchical scheduling framework for multiprocessors based on cluster-based scheduling. They use analytical methods to perform analysis, however this apporach has difficulty in dealing with complicated behavior of tasks.

In [3], the authors analyzed the schedulability of hierarchical scheduling systems using the TIMES tool [2] and implemented their framework in VxWorks [4]. They constructed an abstract task model as well as scheduling algorithms focusing on the component under analysis. However, their approach requires not only timing attributes of the component under analysis, but also timing attributes of other components that can preempt the execution of the current component. Moreover, they did not consider multi-core execution environments.

The authors of [5] provided a compositional framework for the verification of hierarchical scheduling systems using a model-based approach. They specified the system behavior in terms of Preemtible Time Petri nets, and only considered a single-core execution platform.

In [1], the authors study the schedulability of real-time embedded systems under energy constraints, such as using solar panels. We extend their approach by considering hierarchy and a multi-core platform while analyzing it in a compositional way.

III. GENERAL APPROACH

In this paper we structure our system model as a set of hierarchical components. Each component, in turn, is the parallel composition of a set of entities with a local scheduler and possible local resources. Moreover, we consider multi-core execution environments where each processor may have different energy consumption rates depending on the current operation it performs. Similarly, the execution of an operation does not consume the same energy on different processors. Furthermore, we elaborate a concrete task model which consists of a set of distinctly different operations, and associate to each operation together with a processor an energy consumption rate. Of course, the energy consumed by an operation depends on its rate and its execution time. Therefore, the energy consumed by a task is obtained by accumulating the energy consumed by the individual task operations.

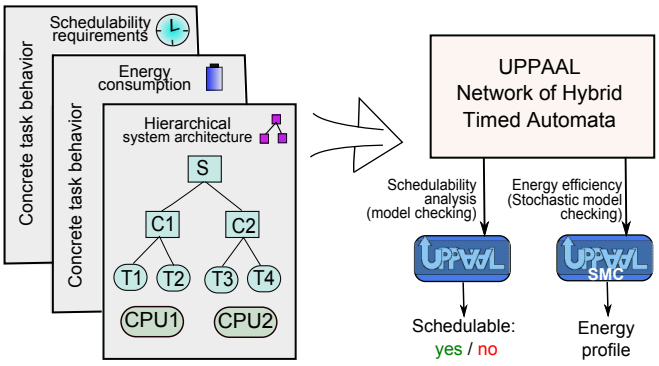

Fig. 1. Overview of the workflow.

Figure 1 summarizes our approach, modeling and verification, where we consider different profiles: schedulability requirements, energy consumption and hierarchical architecture of the system.

1) Concrete behavior and energy consumption of tasks: A task has a concrete behavior perfoming a sequence of timed actions. Each timed action can either be a computation step (`Compute`), acquiring (`Lock`) or releasing (`Unlock`) a resource, or special statements marking the end of the period (`Pend`) or the end of the task execution (`End`).

Definition 1 (Timed action): Given a set of resources $\mathcal{R}$, set of action names $Acts = \{\texttt{Compute, Lock, Unlock, Pend, End}\}$ and a multi-core platform $\mathcal{P}$, a timed action A is a one step computation given by the tuple $\langle Act, Proc, R, BCET, WCET\rangle$ where:

- $Act \in Acts$ is the action name,
- $Proc \subseteq \mathcal{P}$ specifies the identifiers of processors on which the timed action A can be run,
- $R \in \mathcal{R}$ is the resource that action A requires,
- $BCET$ and $WCET$ are respectively best case and worst case execution time,

By $\mathcal{A}$ we denote the set of all timed actions.
We consider a multi-core platform and associate each timed action to a set of processors on which it can execute. Moreover, we associate to each timed action energy consumption rates specifying how much energy is consumed by this action per time unit during its execution on a given processor. To this end, we introduce the rate relation $\Gamma : \mathcal{A} \times \mathcal{P} \rightarrow \mathbb{R}^+$ which associates to the execution of each action on a given processor an energy consumption rate.

Likewise, we define the behavior B of a task as a transition system $\langle L, l^0, \rightarrow\rangle$ specifying the sequence of timed actions perfomred by that task, where L is a set of states, $l^0 \in L$ is the initial state and $\rightarrow \subseteq L \times \mathcal{A} \times L$ is the transition relation. In fact, each transition is guarded by the resource requirement of the corresponding timed action (label). When gathering the whole system, the scheduler decides whether a transition is firable or not according to the availability of required resources. States can be interpreted in the semantic level as valuations of the task variables.

The behavior of a component is given by the parallel composition of the transition systems of its nested tasks.

Definition 2 (Task structure): A task T is given by $\langle Prd, BCET, WCET, Prio, B, \Gamma\rangle$ where Prd is the task period, $BCET$ and $WCET$ are respectively best case and worst case execution time of T, $Prio$ is the priority level associated to task T, B is the task behavior defined above and Γ states the energy consumption rates of T actions.
Therefore, the task specification is given by an interface $Prd, BCET, WCET$ stating the time constraints, a behavior B expressed by a sequence of timed actions and a priority $Prio$ that will be applied for each timed action of the task in question.

2) Hierarchical scheduling: We structure our system as a set of concurrent components. Each component, in turn, can also be a parallel composition of either other components or tasks. Accordingly, the leaves of our system are tasks. Roughly speaking, a component is given by an interface stating its time requirements, declaration of possible local resources and a local policy for scheduling its nested entities.

Definition 3 (Component): A component C is a tuple $\langle Prd, Budget, Pri, s, \mathcal{R}, \langle e_1, .., e_n\rangle\rangle$ where:

- Prd and Pri are the same as for tasks,
- $Budget$ is the amount of resource that the component guarantees to provide to its nested entities,
- $s \in \{EDF, FP, RM, ..\}$ is a scheduling policy,
- $\mathcal{R}$ is a set of typed resources,
- $\langle e_1, .., e_n\rangle$ are component entities, either tasks or components (workload).

Similarly, a system is the top level component without timing requirements $(Prd, Budget, Pri)$. We emphasize the fact that our framework can be instantiated for any combination of scheduling algorithms.

IV. Compositional Framework

Our analysis for multi-core hierarchical scheduling systems aims at obtaining verified and specified task designs that satisfy resource constraints, e.g. cpu usage and energy consumption. The analysis framework is compositional in the sense that the analysis is performed on each component individually with respect to its requirements. The schedulability of each component is verified by checking its timing specification against the interface of its sub entities. Using the same framework each component is also analyzed with respect to its energy profile in terms of energy efficiency.

To this end, we present a behavioral model of hierarchical scheduling systems. This model consists of components and task models based on task specifications including energy profiles. We construct a hierarchical scheduling system model using hybrid timed automata. The schedulability is verified by model checking in Uppaal, and the energy efficiency is analyzed by statistical model checking in Uppaal SMC.

The system model in this paper consists of CPU resource models, abstract task models and concrete task models. The CPU resource model represents a component executing its sub entities, such as tasks or sub-components, using a specific scheduling policy, specifying resource allocations that are

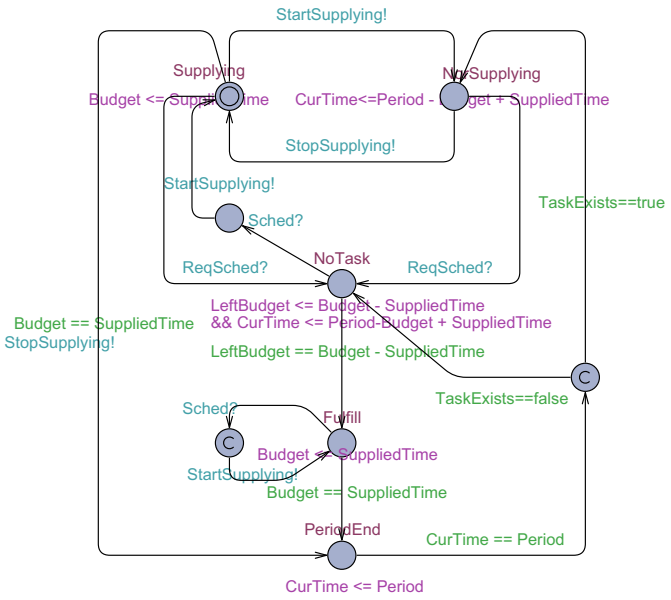

Fig. 2. CPU Resource Model

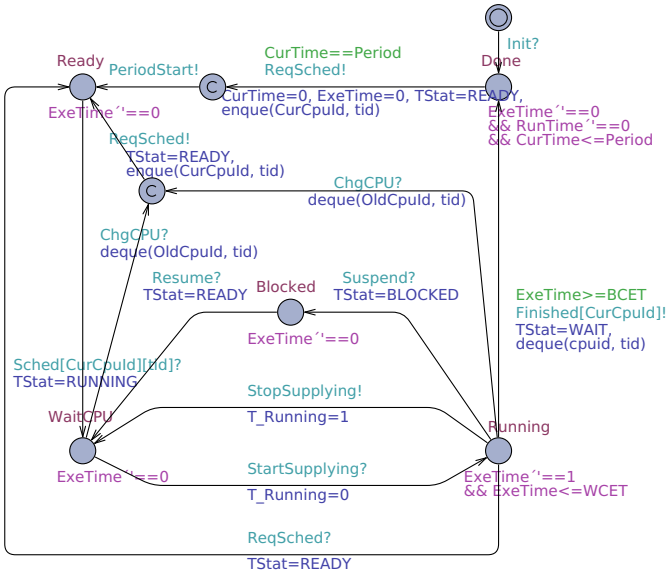

Fig. 3. Abstract Task Model

provided to its sub entities and to verify that they can be scheduled within the component budget. A task specification is described by two task models: an abstract task and a concrete task. The abstract task models the scheduled behavior of tasks with states such as Running, Ready and Blocked. The concrete task models specific processing steps that consume CPU allocation time, energy, and potentially locking system resources.

A. Models of System in Hybrid Timed Automata

The CPU resource, abstract and concrete task models are constructed in terms of hybrid timed automata. The CPU resource model is instantiated with a given component interface, period and budget. The abstract task model is instantiated with individual task timing requirements, period, BCET, WCET and priority. Finally, the concrete task model is instantiated with a sequence of timed actions.

1) Resource Model: In order to achieve compositional analysis, we introduce a non-deterministic CPU resource model. In this way we non-deterministically model all infinitely many ways in which the budget can be supplied by a higher level to a component. The non-deterministic resource model guarantees that it assigns the budgeted amount of resource allocation time to tasks every period. Thus, all components are also guaranteed to receive their budgeted amount of resources, but the supply is non-deterministic within the period. The non-deterministic resource allocation within a period simulates a resource allocation that can always be interrupted by higher priority components that share the same resources. In this way, we facilitate the scheduling analysis of hierarchical scheduling systems in a compositional way.

The non-deterministic CPU resource model is shown in Fig. 2. The transitions with the channels *StartSupplying* and *StopSupplying* are non-deterministic. The clock *SuppliedTime* is the amount of resource allocation that has been provided up to now. The clock is running at the location *Supplying* but stopped at *NotSupplying*.

This resource model guarantees that it provides the budget amount of resource to tasks by handling the two cases: 1) when the left amount of resource allocation budget is equal to the left time up to the end of period 2) when any task has not arrived yet. For the first case, the condition "$CurTime == Period - Budget + SuppliedTime$" on the transition from the location *NoTask* and *Fulfill* and the invariant "$CurTime <= Period - Budget + SuppliedTime$" are given, meaning that the remaining time in the period and the remaining budget are exactly equal. The model continues supplying resource allocation until the end of the period. In the second case, the resource model stays at the location *NoTask* until a task arrives.

In our compositional multi-core framework, each core is individually managed by its corresponding CPU resource model. Each task can be scheduled on different cores over its lifetime.

2) Abstract Task Model: The abstract task model, Fig. 3, is responsible for the periodic execution of the concrete task. It asks for a CPU scheduling using the channel *ReqSched* with CPU Id and task Id at the beginning of the period. It stops if its execution time, *ExeTime*, is fulfilled. It can also be preempted by a higher priority task when other tasks request a CPU scheduling. The abstract task can wait for the next scheduling at the location *Ready*. When its requested resource, such as a semaphore, is delayed, it can be blocked at the location *Blocked* until the resource is available. The clock *RunTime* is used to check whether the task misses the deadline.

3) Concrete Task Model: A timed action is carried out by the concrete task model, Fig. 4. When a task starts a new period and is *Running*, the concrete task starts to execute its sequence of timed actions.

The timed action *Compute* executes using one of the CPUs specified in *Proc*. The execution time is modeled using lower and upper bounds in form of *BCET* and *WCET*. If the abstract

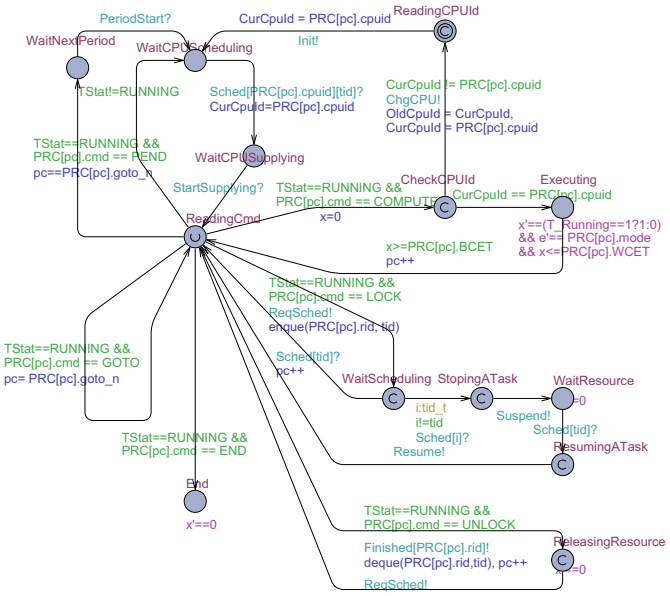

Fig. 4. Concrete Task Model

task is preempted or suspended, the corresponding concrete task should stop running. The progress of clock x depends on the variable *T_Running* representing the running status of its associated abstract task. x is running if *T_Running* is 1, otherwise, it is stopped. The progress of clock e, representing the consumed energy, depends on the current timed action and the CPU on which that action is executing. In addition to CPU, the task can use other types of resource by executing a *LOCK* action. If the task acquires a resource, it moves on to the next timed action, otherwise, it stops at the location *WaitResource*. Whenever this concrete task executes a *COMPUTE* timed action, it checks the CPU Id at *CheckCPUId*. If the current CPU Id (CurCPUId) differs from the CPU Id of the new timed action, the concrete task asks the abstract task to request to be scheduled on the corresponding CPU using *ChgCPU*.

B. Analysis: Schedulability and Energy Efficiency

The schedulability is checked using the UPPAAL model checking engine while the energy efficiency is analyzed using the UPPAAL SMC engine. Namely, UPPAAL SMC is a statistical model checker which monitors a number of simulations of a system based on a stochastic semantics of the hybrid timed automata.

From the viewpoint of the supplier and sub entities, the utilization of resources, such as CPU and energy, can be analyzed as follows: Let C be a component, $W = (T_1, T_2, ...T_n)$ the tasks, s a scheduling algorithm, and R any resource model. I denotes the collective requirements, (Prd and Bud given in *Definition 3*). For any resource model R, a scheduling unit $\Psi(W, R, s)$ is said to be schedulable if and only if:

$$\forall t > 0 \quad \mathtt{dbf}_{\mathtt{A}}(W, t) \leq \mathtt{sbf}_{\mathtt{R}}(t)$$

where $\mathtt{dbf}_{\mathtt{A}}(W, t)$ is the demand-bound function and $\mathtt{sbf}_{\mathtt{R}}(t)$ is the supply-bound function [10].

The interface I of a component $C(W, s)$ is said to be schedulable if the scheduling unit $\Psi(W, R, s)$ is schedulable with $R = I$, i.e. the resource model R satisfies the interface I of the component timing requirements.

For an processor resource $CPU \in \mathcal{P}$, the meaning of the schedulability in [10] is adopted for this framework. To verify the schedulability, we check the following property:

$$\forall t > 0 \quad \mathtt{dbf}_{\mathtt{A_{CPU}}}(W, t) \leq \mathtt{sbf}_{\mathtt{R_{CPU}}}(t)$$

For an energy resource $E \in \mathcal{R}$, an energy profile I_E of a component is said to be efficient if and only if the following condition is satisfied for all time intervals t:

$$\forall t > 0 \; \mathtt{dbf}_{\mathtt{A_E}}(W, t) \leq \mathtt{sbf}_{\mathtt{R_E}}(t)$$

V. CONCLUSIONS

We have presented a compositional framework for the analysis of schedulability and energy efficiency of hierarchical embedded multi-core real-time systems. The framework has been instantiated as reusable models given in terms of hybrid timed automata which we analyzed using UPPAAL and UPPAAL SMC. The reusable models ensure that when modeling a hierarchical scheduling application only the concrete task behavior and the hierarchical structure need to be specified by the system engineer. The framework also allows for instant changes of the scheduling policy at each given level in the hierarchy. We are currently applying the framework to an avionics UAV case-study from the CRAFTERS [6] project.

REFERENCES

[1] Y. Abdeddaïm and D. Masson. Real-time scheduling of energy harvesting embedded systems with timed automata. In *RTCSA*, pages 31–40. IEEE Computer Society, 2012.

[2] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi. Times: A tool for schedulability analysis and code generation of real-time systems. In K. G. Larsen and P. Niebert, editors, *FORMATS*, volume 2791 of *Lecture Notes in Computer Science*, pages 60–72. Springer, 2003.

[3] M. Behnam, T. Nolte, I. Shin, M. Åsberg, and R. Bril. Towards hierarchical scheduling in vxworks. In *OSPERT 2008, Proceedings of the Fourth International Workshop on Operating Systems Platforms for Embedded Real-Time Applications*, pages 63–72, 2008.

[4] M. Behnam, T. Nolte, I. Shin, M. Åsberg, and R. J. Bril. Towards hierarchical scheduling on top of VxWorks. In *Proceedings of the 4th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT'08)*, pages 63–72, July 2008.

[5] L. Carnevali, A. Pinzuti, and E. Vicario. Compositional verification for hierarchical scheduling of real-time systems. *IEEE Transactions on Software Engineering*, 39(5):638–657, 2013.

[6] CRAFTERS. project website. http://www.crafters-project.org/.

[7] B. Leiner, M. Schlager, R. Obermaisser, and B. Huber. A comparison of partitioning operating systems for integrated systems. In F. Saglietti and N. Oster, editors, *SAFECOMP*, volume 4680 of *Lecture Notes in Computer Science*, pages 342–355. Springer, 2007.

[8] L. Niu and J. Xu. Improving schedulability and energy performance for weakly hard real-time systems. In *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*, pages 41–50, 2012.

[9] J. Rufino, J. Craveiro, and P. Veríssimo. Building a time- and space-partitioned architecture for the next generation of space vehicle avionics. In *SEUS*, volume 6399 of *Lecture Notes in Computer Science*, pages 179–190. Springer, 2010.

[10] I. Shin, A. Easwaran, and I. Lee. Hierarchical scheduling framework for virtual clustering of multiprocessors. In *ECRTS*, pages 181–190. IEEE Computer Society, 2008.

[11] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *RTSS*, pages 2–13. IEEE Computer Society, 2003.

[12] I. Shin and I. Lee. Compositional real-time scheduling framework with periodic model. *ACM Trans. Embedded Comput. Syst.*, 7(3), 2008.