# Model Checking Multitask Applications for OSEK Compliant Real Time Operating Systems

Mark L. McKelvin, Jr. and Gerard Holzmann
Jet Propulsion Laboratory, Laboratory for Reliable Software
California Institute of Technology
Pasadena, California 91109
Email: {mark.mckelvin, gerard.holzmann}@jpl.nasa.gov

***Abstract*—In the verification of multitask software in embedded systems, general purpose model checkers do not inherently consider characteristics of the real time operating system, such as priority-based scheduling, priority inversion, and protocols for protecting shared memory resources. Since explicit-state model checkers generally explore all possible execution paths and task interleaving, this could potentially lead to exploring execution paths that are redundant, unnecessarily increasing verification complexity and hampering tractability. Based on this premise, in this work we investigate how one can improve the performance of explicit-state model checkers, such as SPIN, for the verification of multitask applications that target OSEK compliant real time operating systems.**

## I. Introduction

Increasing system complexity and reliance on the correct operation of embedded systems, motivated the formation of OSEK, an open standard for the specification of real time operating systems, communications, and network protocols as a standard software architecture for embedded systems [1]. System complexity and requirements for reliable operation make it difficult to achieve systems for which designers can justifiably rely on the service it delivers, a characteristic that is referred to in this context as *dependability*. Traditional methods for achieving dependable software, such as testing, debugging, and peer reviews, quickly become costly and inefficient as systems become more complex. Formal software verification methods such as *model checking*[2], [3] has shown promise in addressing the problems associated with the verification of software for complex, dependable embedded systems, for instance as demonstrated in the detection of five previously unknown concurrency errors in National Aeronautics Space Administration (NASA) Deep Space 1 mission [4].

Model checking is a verification technique that utilizes a model of a system under test to exhaustively explore the set of all reachable system states. It has been particularly useful for verifying properties of embedded systems with concurrently executing tasks through an automatic and exhaustive search procedure. But, model checking is prone to the problem of *state explosion* [7]. Techniques, such as symbolic model checking, abstraction, and partial order reduction address the state explosion problem by attempting to reduce the size of the system model, as summarized in, e.g., [8], [5]. Other techniques, such as directed model checking [9] and distributed model checking [10], [6] attempt to improve the performance of the search algorithm of the model checker. In this work, we investigate an approach to model checking multitask applications of an embedded system by modeling invariants of OSEK compliant real time operating systems.

## II. Related Work

Tools such as UPAAL [11] and Kronos [12] are commonly used to verify timing properties of real time systems, that are based on timed automata formalisms [13]. A drawback to these tools and methods is that additional state variables that characterize explicit timing properties of the system are introduced, thus, it increases the size of the search space by an additional multiplicative factor in the size of discrete time values. In the literature, the work that most closely relates to our work includes, Duval and Julliand *et. al.* [14], Parizek *et. al.* [15], and [16] where the authors develop models of the underlying kernel of specific real time operating systems.

Our work differs from the above work since it abstracts discrete time into a temporal ordering of concurrent activities, thereby, focusing on higher levels of abstraction to capture system model and properties related to concurrency, as opposed to capturing discrete timing of system behavior. We focus on modeling the behavior of an OSEK compliant operating system where our implementation targets reuse on different platforms and minimizing the use of data structures that are unnecessary to verify certain properties.

## III. Approach

The approach that is adopted in this work constructs the behavior of an OSEK compliant real time operating system in Promela with the purpose of applying SPIN model checking tool [5] to perform verification of multitask applications. Promela is a language for constructing verification models, and SPIN is a verification tool that executes on a Promela model. The computational model for Promela is based on the concurrent execution of *processes* that communicate via *channels* and global variables. Process executions are controlled by *blocking* or *unblocking* process execution. Thus, in our implementation of OSEK specification, the goal is to utilize the service calls as an interface to signaling the blocking, unblocking of processes and modifying only relevant state variables to represent the task management and synchronization mechanisms per the OSEK specification. This approach allows us to focus only

on relevant state information that is necessary to control the executability of Promela processes in an effort to limit that amount of additional state information that must be captured. Currently, we illustrate our approach by capturing the specification on task and resource management services in the OSEK specification.

Fig. 1. An illustration of the task model for OSEK compliant real time operating systems.

### A. Task Management and Scheduling

Task management and scheduling provides the framework for declaring, defining, and using tasks in a specified order depending on the state of the task as illustrated in the task model Figure 1. In our current implementation in Promela, each OSEK task is defined as a Promela an extension to SPIN that allows for processes to have a priority. Along with knowledge of task state, a new routine in SPIN version 6.2, $\mathrm{highest(pid)}$ allows for the execution of Promela processes in accordance with priority based scheduling rules. This behavior can accurately model the behavior of OSEK compliant schedulers.

### B. Resource Management

The OSEK specification manages access to shared resources in a way such priority inversion and deadlocks are guaranteed to not occur by specifying a priority ceiling protocol. An implementation of resource management is supported by two new additional routines to the SPIN verification and simulation engine, $\mathrm{get\_priority(pid)}$, which returns the current priority of the process with process instantiation number $pid$ and $\mathrm{set\_priority(pid, N)}$, which sets the priority of the process with $pid$ to $N$, where $N \in \{1, 2, \ldots, 255\}$ The additional routines provide the capability to implement the OSEK priority ceiling protocol and provide resource protection.

## IV. Conclusions and Future Work

A real time operating system plays an important role in software intensive computing systems, in particular, embedded systems that must operate correctly. We have outlined our work in progress on using model checking for verifying multitask applications on software intensive embedded systems that conform to the OSEK specification. We plan to implement the remaining service calls that are necessary to perform exhaustie verification of a multitask application for OSEK compliant systems, and evaluate its application.

## Acknowledgment

The authors would like to thank the Ed Gamble, Micah Clark, Michel D. Ingham, and Mihai Florian for thoughtful discussions on this work.

## References

[1] (2005) Osek/vdx operating system specification 2.2.3. [Online]. Available: http://www.osek-vdx.org

[2] E. A. Emerson and E. M. Clarke, "Characterizing correctness properties of parallel programs using fixpoints," in *International Congress of Mathematicans*, 1980, pp. 169–181.

[3] J.-P. Queille and J. Sifakis, "Specification and verification of concurrent systems in cesar," in *Proceedings of the 5th Colloquium on International Symposium on Programming*. London, UK: Springer-Verlag, 1982, pp. 337–351.

[4] K. Havelund, M. Lowry, and J. Penix, "Formal analysis of a space-craft controller using spin," *IEEE Trans. Softw. Eng.*, vol. 27, pp. 749–765, August 2001.

[5] G.J. Holzmann, "The Spin Model Checker: primer and reference manual," Addison-Wesley, Reading, MA, USA, 2004.

[6] G.J. Holzmann and D. Bosnacki, "The Design of a multi-core extension of the Spin Model Checker," *IEEE Transactions on Software Engineering*, vol. 33, No. 10, pp. 659–674, Oct. 2009.

[7] A. Valmari, "The state explosion problem," in *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets*. London, UK: Springer-Verlag, 1998, pp. 429–528.

[8] E. M. C. Jr., O. Grumberg, and D. A. Peled, *Model Checking*. The MIT Press, 1999.

[9] S. Edelkamp, V. Schuppan, D. Bonaki, A. Wijs, A. Fehnker, and H. Aljazzar, "Survey on directed model checking," in *Model Checking and Artificial Intelligence*, ser. Lecture Notes in Computer Science, D. Peled and M. Wooldridge, Eds. Springer Berlin / Heidelberg, 2009, vol. 5348, pp. 65–89.

[10] R. Kumar and E. G. Mercer, "Load balancing parallel explicit state model checking," *Electronic Notes in Theoretical Computer Science*, vol. 128, no. 3, pp. 19 – 34, 2005.

[11] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi, "UPPAAL — a Tool Suite for Automatic Verification of Real–Time Systems," in *Proc. of Workshop on Verification and Control of Hybrid Systems III*, ser. Lecture Notes in Computer Science, no. 1066. Springer–Verlag, Oct 1995, pp. 232–243.

[12] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine, "Kronos: A model-checking tool for real-time systems," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, A. Hu and M. Vardi, Eds. Springer Berlin / Heidelberg, 1998, vol. 1427, pp. 546–550.

[13] J. Bengtsson and W. Yi, "Timed automata: Semantics, algorithms and tools," in *Lectures on Concurrency and Petri Nets*, ser. Lecture Notes in Computer Science, J. Desel, W. Reisig, and G. Rozenberg, Eds. Springer Berlin / Heidelberg, 2004, vol. 3098, pp. 87–124.

[14] G. Duval and J. Julliand, "Modeling and verification of the rubis -kernel with spin," in *In SPIN95 Workshop Proceedings*, 1995.

[15] P. Parizek, T. Kalibera, and J. Vitek, "Model checking real-time java," Department of Distributed and Dependable System, Charles University, Tech. Rep. Technical Report 1, 2010.

[16] T. Aoki, "Model checking multi-task software on real-time operating systems," in *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, may 2008, pp. 551 –555.