Towards Verifiable Deeply Embedded Systems *[*]

Terry Tidwell and Christopher Gill
CSE Department, Washington University
St. Louis, MO, USA
{ttidwell,cdgill}@cse.wustl.edu

1. ABSTRACT

An important class of deeply embedded systems (DES) involves interlocking cyber-physical control loops, each with its own dynamics and correctness constraints. For instance, in medical systems which monitor a patient's vital signs and control drug infusion rates, multiple sensing and control systems may be implicitly coupled. These systems present new challenges for control theory, scheduling and verification. What is needed are platforms capable of precise monitoring and enforcement of diverse execution constraints, and techniques to verify system correctness.

2. INTRODUCTION

Deeply embedded systems (DES) have stringent constraints on timeliness and other properties whose assurance is crucial to correct system behavior. These constraints are often imposed by interactions with the real world. To be correct, these systems must maintain these properties invariantly, through all possible system executions. Consider, e.g., an automated system which monitors and administers medication to a patient. Correctness of this system is subject to the real-world constraints of proper dosage and proper administration rates, as well as patient physiology.

To date, reusable platforms and analysis techniques have not addressed the kinds of interactions these systems entail. This results in overly specialized systems built largely from scratch. Traditional priority based scheduling is also inadequate to capture semantics of such inter-related control loops. Group scheduling [2] and other hierarchical scheduling techniques [10] show promise, but it must be possible to verify correctness of the behaviors they induce.

Model checking can formally verify that a system does in fact maintain critical properties. Such verification requires that models of the system have high fidelity to the modeled system and have a state space whose exploration is decidable and tractable. However, model checking systems with DES semantics such as preemption, interval processing times, and tasks that can announce completion to a dispatching thread, while maintaining fidelity, decidability and tractability has been shown to be a formidable challenge [9].

Previous research in real-time systems has extended model checking to a restricted but widely used class of scheduling policies (rate-monotonic scheduling of fixed-period tasks [12]) in which preemption only occurs at well defined points. However, many DES systems may have less constrained scheduling semantics, and indeed in interacting with the real world may require more flexible forms of scheduling outside of the fully vetted scheduling policies currently available.

[*]*Research at Washington University supported in part by NSF awards CCF-0615341 (EHS) and CNS-0716764 (Cybertrust).

3. RELATED WORK

Group scheduling [2] and other hierarchical scheduling techniques [10] have been used to implement scheduling decision functions (SDF) for enforcement of higher level policies [14]. Their composable design allows SDFs in these architectures to address multiple inter-related system constraints. However, for many possible SDF hierarchies, no closed form analysis exists.

The most relevant general purpose modeling techniques (stopwatch automata [4], timed automata [1], and untimed finite automata) have important limitations for modeling preemptively scheduled systems: (1) checking stopwatch automata models may be undecidable, (2) the time representation in traditional timed automata does not model preemption, and (3) the resulting state space for untimed finite automata is intractably large.

The limitations of these general purpose modeling approaches have given rise to a number of other approaches, which capture and leverage additional information about the structure of the system itself. One such approach is to compose automata based on common interfaces and sets of resources [8, 5]. Another relevant approach is to identify quasi-cyclic structures in the system's execution, which can be used to reduce the memory required for complete state exploration [7]. A third approach is to use abstract interpretation in combination with model checking [6] to reason about event interleavings and paths of execution in the system. None of these approaches provides the combined ability to analyze relative resource consumption, timing, and preemption that is needed for verification of the kinds of deeply embedded systems on which this research focuses.

4. PLATFORM

A reusable platform for developing DES systems must provide the following services: (1) flexible and composable SDF design, (2) precise on-line measurement of system behavior, and (3) explicit control over all system behaviors.

The Group Scheduling (GS) [2] and Data Streams [3] facilities in KURT-Linux offer the kinds of platform support needed for DES. They extend Linux with fine grain mechanisms for measurement and control of potentially *all* system services, even kernel services. This approach allows scheduling constraints to be composed hierarchically in a systematic but flexible way.

However, due to the flexibility with which scheduling constraints can be composed, existing scheduling analyses only cover a restricted subset of the possible scheduling policies. The remaining challenge is to develop techniques to verify a wider range of systems built upon these precise yet reusable platforms.

5. QUASI-CYCLIC TIMED STRUCTURES

To improve scalability of model checking for real-world systems, Dwyer, et al., introduced the idea of a quasi-cyclic structure, in which a predicate over system states produces a projection of the state space in which a set of sub-states – with the same values for a subset of the system's state variables – is visited recurrently [7]. This regularity makes the verification of systems with large state-spaces tractable. Our work aims to show that this same regularity, when exploited in models of real-time preemptively scheduled systems, can make verification of a much more general subset of these systems decidable and tractable.

To recognize this structure and exploit it, novel composition techniques are required. The composition technique we use takes as input (1) system processes modeled as timed automata, (2) their preemption semantics and (3) the scheduling policy of the system, and outputs a single automaton we call a time domain automaton (TDA).

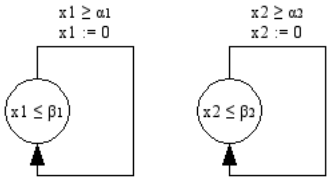

Figure 1: Process Automata P1 and P2

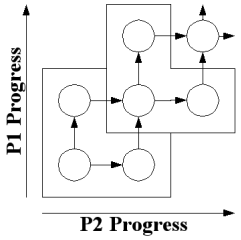

Figure 2: Time Domain Automaton with Quasi-Cyclic Structure

In the process models (shown in Figure 1) the clocks are independent. However, in systems with preemption this assumption does not hold. In order to model the processes' preemption semantics a subset of these clocks must be composed into a common time domain - representing when the processes are competing for a common resource.

The transitions in a TDA (shown in Figure 2) represent distinct events in the system, the composed process models' states record execution times, and the guards on each transition and the invariants in each state represent minimum and maximum bounds on the demand function [11] of each process.

The system's scheduling semantics is specified as a set of constraints over the composed state space (i.e. precedence or other event ordering constraints, etc.). These constraints can capture domain specific knowledge and allow us to encode this additional information into a TDA model.

Once we have defined (1) the processes, (2) the subset of dependent clocks, and (3) the scheduling constraints, we can generate the time domain automaton, which has infinitely many diverging states that represent all possible interleavings of system events as well as bounds on the time at which these events could occur.

We then search the model for quasi-cyclic structures, as Figure 2 illustrates. If found, each quasi-cyclic repetition is indexed with an integer value. We can then construct equations for scheduling induced bounds on system execution as a function of the original process models and this discovered index variable [13].

Our research to date has shown that for scheduling functions that restrict event interleavings (as encoded by an finite untimed automata) scheduling induced bounds can be calculated that show fidelity with empirical system executions and bound system states in such a way that decidablity is ensured in each bounded region.

6. REFERENCES

[1] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[2] T. Aswathanarayana, V. Subramonian, D. Niehaus, and C. Gill. Design and performance of configurable endsystem scheduling mechanisms. In *Proceedings of 11th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS)*, 2005.

[3] B. Buchanan, D. Niehaus, D. Dhandapani, R. Menon, S. Sheth, Y. Wijata, and S. House. The data stream kernel interface. Technical Report ITTC-FY98-TR11510-04, Information and Telecommunication Technology Center, University of Kansas, 1998.

[4] F. Cassez and K. G. Larsen. The impressive power of stopwatches. In *11th International Conference on Concurrency Theory (CONCUR 2000)*, pages 138–152, Aug. 2000.

[5] A. Chakrabarti, L. de Alfaro, T. Henzinger, and M. Stoelinga. Resource interfaces. In *Third ACM International Conference on Embedded Software (EMSOFT 2003)*, pages 117–133, Oct. 2003.

[6] X. Deng, J. Lee, and Robby. Bogor/kiasan: A k-bounded symbolic execution for checking strong heap properties of open systems. Technical Report SAnToS-TR2006-1, Laboratory for Specification, Analysis, and Transformation of Software (SAnToS), Kansas State University, 2006.

[7] M. B. Dwyer, Robby, X. Deng, and J. Hatcliff. Space reductions for model checking quasi-cyclic systems. In *Third ACM International Conference on Embedded Software (EMSOFT 2003)*, pages 173–189, Oct. 2003.

[8] T. A. Henzinger and S. Matic. An interface algebra for real-time components. In *Proceedings of 12th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS)*, Apr. 2006.

[9] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Decidable integration graphs. *Information and Computation*, 150(2):209–243, 1999.

[10] Regehr, Reid, Webb, Parker, and Lepreau. Evolving Real-time Systems Using Hierarchical Scheduling and Concurrency Analysis. In 24^{th} *IEEE Real-time Systems Symposium*, Cancun, Mexico, Dec. 2003.

[11] I. Shin and I. Lee. Compositional Real-Time Scheduling Framework. In *The 25th IEEE Real-time Systems Symposium (RTSS)*, Lisbon, Portugal, Dec. 2004.

[12] V. Subramonian, C. Gill, C. Sánchez, and H. B. Sipma. Reusable models for timing and liveness analysis of middleware for distributed real-time and embedded systems. In *Sixth ACM/IEEE International Conference on Embedded Software (EMSOFT 2006)*, pages 252–261, Oct. 2006.

[13] T. Tidwell, C. Gill, and V. Subramonian. Scheduling induced bounds and the verification of preemptive real-time systems. Technical Report WUCSE-2007-34, Department of Computer Science and Engineering, Washington University in St.Louis, 2007.

[14] X. Wang, H.-M. Huang, V. Subramonian, C. Lu, and C. Gill. CAMRIT: Control-based Adaptive Middleware for Real-time Image Transmission. In *Proc. of the 10th IEEE Real-time and Embedded Tech. and Applications Symp. (RTAS)*, Toronto, Canada, May 2004.