Exploring the interaction between functional performance and scheduling abstractions

Paolo Pazzaglia, Alessandro Biondi, Marco Di Natale, Giorgio Buttazzo, Matteo Secchiari
Scuola Superiore Sant'Anna, Pisa, Italy
E-mail: {paolo.pazzaglia, alessandro.biondi, marco.dinatale, giorgio.buttazzo}@sssup.it

***Abstract*—Several schedulability analysis models and results are based on the assumption that the functional performance of the system can be represented (abstracted) by a very simple task model with release and completion time constraints. The hard deadline model is probably the best known example. However, several real applications, such as engine control challenge both the task activation and execution time model, and also the deadline abstraction and the hard schedulability assumptions. Our project on the development of a Simulink based co-simulation environment aims at a more detailed exploration of the impact of (possibly late) response time on the performance of complex control functions. We describe not only our cosimulation framework, but also additional process steps and challenges that relate to its use.**

I. Description

Many results in real-time modeling, analysis, and scheduling are based on the assumption that a timing task model can abstract information on the functional performance of the application or even its correctness by using a very simple set of attributes and constraints. A typical example is the hard deadline model as it applies to periodic or sporadic tasks. However, both the task model and the deadline criticality assumptions need to be often extended to cope with the requirements of actual applications.

An example is the multiframe task model [1] that was developed to cope with the requirements of multimedia encoding or decoding tasks, where a simple worst case execution time assumption was too pessimistic to allow for an accurate analysis. In addition, these applications usually cope with temporary overloads by changing the functional behavior in a way that could be represented as a mode change or by an imprecise computation model. Other task models try to cope with conditional executions or other types of inter-job dependencies. Similarly, the hard deadline assumption was challenged for most constrol applications. Deadline misses could be easily tolerated in many cases and jitter plays a significant role [7].

The analysis of fuel injection applications revealed the needs for a further extension of the task model. The task activation times are now associated with a physical process (the rotation of the engine shaft) with a known dynamic, and the task execution time is adaptive with respect to the activation rate. These characteristics have been included in the Adaptive Variable Rate task model [2]. However, what is also missing from the current analysis is a refinement of the deadline assumption. Even if all analysis papers on this model assume hard deadlines, in reality, fuel injection controls can miss deadlines without critical consequences. The only effect is that the fuel injection is performed at an angle and with a duration as computed in the previous cycle.

To better understand the impact of scheduling decisions on the performance of engine control applications (and possibly others), we developed a co-simulation framework in which a model of the engine and all the components that impact the internal combustion process is represented together with a model of the engine controls and their execution in time under the control of a scheduler. The models are integrated in Simulink and allow to evaluate how performance and cost metrics (output power, pollutants and noise) are affected by scheduling policies and scheduling attributes.

Last year we presented the framework structure [3], and the subsystems in it, including the engine model and the extensions to the T-Res framework [4] for simulating the real-time scheduler and the task execution times, including the adaptive execution time behavior.

Since then, the framework has been enhanced to cope with several other details of interest. The engine model has been improved and the control models have been extended by allowing for multiple fuel injections in a single cycle. The multiple injection model provides the first justification for an adaptive task execution behavior. However, in reality, the control applications are quite complex and both the task model and the adaptive behavior are more complex than what we assumed in our early release.

In our talk, we discuss the most recent model improvements, but also current efforts for including a more realistic model of the controls in our cosimulation environment. The source information for our improved model are the Amalthea model in the FMTV challenge and an AUTOSAR model of an engine control application, as discussed in [6].

The first possible outcome of our improved models is the analysis of more accurate models for the description of the impact of deadline misses on the application performance. The m-k model [5] is a first attempt in this direction but has two fundamental limitations. It is still a binary model, in which the application is either safe or faulty, but has no indication of a performance value. the second limitation is that the m-k model does not account for the pattern in time of possible deadline misses.

Finally, the simulation framework cannot live in isolation. We highlight the many ways in which it needs to (or can) be integrated in the development process. Some of the fundamental interactions are the following

- The simulation of the task (segment) executions require WCET estimates. The WCET estimates can in turn be obtained by analyzing code that could be obtained using automatic generation techniques from the model itself. Also, for most recent multicore automotive architectures, the WCET assumptions can not be independent from assumptions on the task placement and even the allocation of scratchpad memory.

- The function and task placement in multicore platforms affects in a significant way the scheduling and the overall performance of the application. We discuss possible options for integrating the placement definition (optimization) and its outcome in the simulation model, possibly in interative loops.

References

[1] Baruah S, Chen D, Gorinsky S, Mok A (1999) Generalized multiframe tasks. Real-Time Syst 17(1):522

[2] . A. Biondi, M. D. Natale, and G. Buttazzo. Response-time analysis for real-time tasks in engine control applications. In Proceedings of the 6th International Conference on Cyber-Physical Systems (ICCPS 2015), Seattle, Washington, USA, April 14-16, 2015

[3] Paolo Pazzaglia, Alessandro Biondi, Giorgio Buttazzo and Marco Di Natale A Simulation Framework to Analyze the Scheduling of AVR tasks with respect to Engine Performance Proceedings of the WATERS Workshop, Toulouse, June 2016.

[4] F Cremona, M Morelli, M Di Natale, TRES: A Modular Representation of Schedulers, Tasks, and Messages to Control Simulations in Simulink in Proc. of the ACM SAC Conference, 2015, Salamanca, Spain.

[5] Guillem Bernat, Alan Burns, and Albert Liamosi. Weakly hard real-time systems. Computers, IEEE Transactions on, 50(4):308321, 2001.

[6] A Biondi, M Di Natale, Y Sun, S Botta, *Moving from single-core to multicore: initial findings on a fuel injection case study* SAE Technical Paper, SAE Conference, DEtroit, USA, April 2016

[7] Cervin, A., Henriksson, D., Lincoln, B., Eker, J., Arzen, K.E. How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime, in IEEE control systems 23(3), 1630 (June 2003)

[8] T. Henzinger, B. Horowitz, and C. Kirsch, Giotto: a time-triggered language for embedded programming, Proceedings of the IEEE, vol. 91, no. 1, pp. 8499, Jan 2003