FMTV 2016 Verification Challenge

Arne Hamann, Dirk Ziegenbein, Simon Kramer, Martin Lukasiewycz
Robert Bosch GmbH
Corporate Research, Germany
Email: {arne.hamann|dirk.ziegenbein|simon.kramer2|martin.lukasiewycz}@de.bosch.com

***Abstract*—In [1] we argued that the complex dynamic behavior of automotive software systems, in particular engine management, in combination with emerging multi-core execution platforms, significantly increased the problem space for timing analysis methods. As a result, the risk of divergence between academic research and industrial practice is currently increasing.**

Therefore, we now provide a concrete automotive benchmark, a full blown performance model of a modern engine management system based on [1], with a goal to challenge existing timing analysis approaches with respect to their expressiveness and precision.

I. Challenge

In short, the challenge consists in determining tight end-to-end latency bounds for a set of given cause-effect chains in a full blown engine management software. For this purpose, an Amalthea [2] performance model of the software can be downloaded at the *FMTV challenge website by mid of December*.

As mentioned above, the dynamic behavior of a engine management software is quite complex and contains mechanisms that explore the limits of existing approaches:

- preemptive and cooperative priority based scheduling
- periodic, sporadic, and engine synchronous tasks
- multi-core platform with distributed cause-effect chains including cross-core communication
- label (i.e. data) placement dependent execution times of runnables

The provided Amalthea model contains a hardware model of a simplified microcontroller architecture with four symmetric cores (see Figure 1). The cores are interconnected by a crossbar (full connectivity, FIFO arbitration at memories). The system-wide frequency is 200 MHz. Furthermore, initial mappings (runnable to task, task to core) are specified.

Each core $CORE_x$ is connected to a local memory $LRAM_x$. Additionally, there exist a global memory $GRAM$ that is shared among all cores. The specified runnable execution times assume that code is executed directly from core-exclusive flashes without contention. In contrast, access to labels including memory arbitration effects are not included in the execution times. Initially, all labels are assumed to be stored in the global memory. The following symmetric memory access times are assumed:

- Reading from and writing to the global memory: 9 cycles
- Reading from and writing to the core local memory: 1 cycle
- Reading from and writing to the local memory of a different core: 9 cycles

The memory access model assumes that runnables read all required labels at the beginning of their execution, afterwards the calculation takes place, before all labels are written back. Local RAMs are single-ported, such that concurrent accesses to the same memory lead to contention, and are arbitrated according to the FIFO policy.

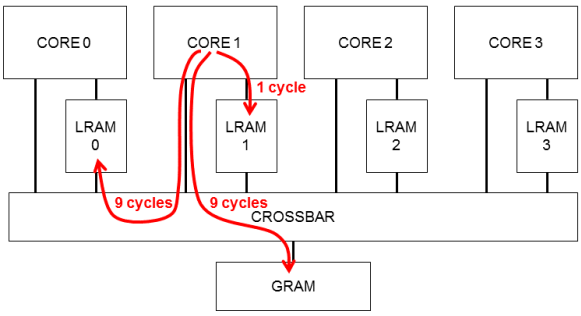

Fig. 1. Microcontroller architecture used in the challenge

Obviously, solving the intertwined problem of scheduling including the effects of memory accesses to the execution times is very hard. Therefore several separate challenges are formulated:

- calculate tight end-to-end latencies ignoring memory accesses and arbitration
- calculate tight end-to-end latencies including memory access and arbitration accesses
- optimize end-to-end latencies by mapping the labels among the local and global memories

II. Approaches

Figure 2 visualizes different approaches that are addressed by the challenge presented in this paper. Two Pareto-fronts show the general trade-off between accuracy and effort of the different approaches.

The first Pareto-front converges towards the the actual worst-case coming from formal and, thus, conservative approximations. Here, compositional timing analysis methods, on the one hand, are very efficient in terms of computational complexity and modeling efforts, but usually lead to overestimated worst-case bounds, especially for distributed systems. Formal method like timed automata, on the other hand, scale badly to large systems due to the underlying exponential nature of model checking.

The second Pareto-front shows simulation-based approach that optimistically underestimate the worst-case. Here, the challenge lies in finding stimuli for the system under simulation, leading to values that are close to the worst-case since

enumerating and simulating all possible situation is infeasible from a practical point-of-view.

In a nutshell, novel methods should improve state-of-the-art and not be dominated by any method on the Pareto-front. That means, either improve in terms of accuracy or reduce effort.

A. Formal Approaches

Classical real-time scheduling [3], [4] considers tasks on a single processor and their schedulability, taking into account execution times, release times, and deadlines. These approaches use problem-specific formalizations to model systems and cannot be applied directly to distributed systems with heterogeneous components, schedulers and protocols. An extension of the classical approaches towards distributed systems is known as *holistic scheduling* [5], [6]. Here, the equations of the specific scheduling approaches are combined by introducing dependency formalizations. Due to the quickly growing complexity of this approach, its applicability is limited by the fast increasing number of equations and dependencies that are introduced with each component in the system.

In contrast to holistic scheduling, compositional approaches promise a better extensibility by relying on components that exchange information via event streams [7], [8], [9]. These event streams capture properties like periodic behavior or jitter while end-to-end latencies can be determined by adding delays along the entire data flow. These compositional approaches can deliver relatively tight latencies, but further reduction of the inherent over-approximations of the determined latencies are obliviously limited by the information in the event streams.

Of course a system might be modeled using timed automata such that the resulting end-to-end latencies are exact when applying model checking. However, as the approaches scale exponentially, they are generally integrated as components into compositional approaches [10].

B. Simulative Approaches

In contrast to the formal approaches, simulative approaches do not require an abstraction of the model and can therefore be easily extended with new components, schedulers, and protocols. Without the need to abstract the model, simulative approaches are theoretically capable of determining the exact latency values without any over-approximation. However, determining the right stimuli for the simulation that will result in the actual worst-case latency is extremely difficult, and therefore usually randomized inputs are chosen. It is obvious that with a longer runtime of simulative approaches, the undesirable under-approximation can be reduced.

Due to their good usability and simple integration, simulation tools like TIMING ARCHITECTS [11], CHRONSIM [12], or TRACEANALYZER [13] enjoy great popularity in industry. Nevertheless, for safety critical application it can be dangerous to rely on simulative approaches as they do not guarantee to capture the actual worst-case even if a certain safety margin is added to the observed worst-case after extensive simulations.

C. Hybrid Approaches

Finally, hybrid approaches that combine simulations and formal approaches might be considered. These approaches might use traces of simulations for the input of formal methods

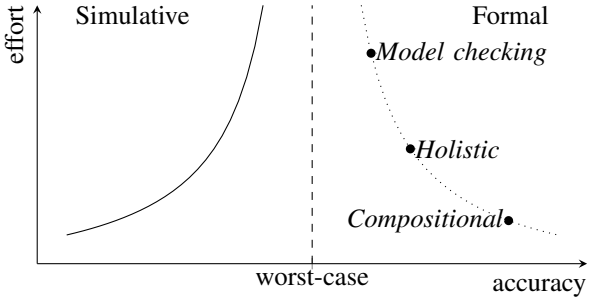

Fig. 2. Illustration of different approaches with respect to accuracy and effort.

to deliver a worst-case that is larger than the simulation result and at the same time lower than a purely analytically determined value. Another approach to mix the two views of simulation and formal approach is the consideration of typical worst-case analysis [14]. Here, a so-called typical worst-case is determined that is only violated by a strictly bounded number of occurrences in a given time window. This approach is very useful in specific scenarios where occasional deadline violations do not affect the correct behavior of the system. As a result, this approach is applied in combination with the domain knowledge of the underlying system, e.g., a control application.

REFERENCES

[1] S. Kramer, D. Ziegenbein, and A. Hamann, "Real world automotive benchmark for free," in *Sixth International Workshop on Analysis Tools and Methodologies for Embedded Real-time Systems (WATERS)*, 2015.

[2] "Amalthea 4 Public Project," http://www.amalthea-project.org/.

[3] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[4] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real time scheduling theory: A historical perspective," *Real-time systems*, vol. 28, no. 2-3, pp. 101–155, 2004.

[5] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocessing and microprogramming*, vol. 40, no. 2, pp. 117–134, 1994.

[6] J. P. Gutiérrez, J. G. García, and M. G. Harbour, "On the schedulability analysis for distributed hard real-time systems," in *Proceedings of the Ninth Euromicro Workshop on Real-Time Systems*. IEEE, 1997, pp. 136–143.

[7] K. Gresser, "An event model for deadline verification of hard real-time systems," in *Real-Time Systems, 1993. Proceedings., Fifth Euromicro Workshop on*. IEEE, 1993, pp. 118–123.

[8] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4. IEEE, 2000, pp. 101–104.

[9] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis–the symta/s approach," *IEE Proceedings-Computers and Digital Techniques*, vol. 152, no. 2, pp. 148–166, 2005.

[10] K. Lampka, S. Perathoner, and L. Thiele, "Analytic real-time analysis and timed automata: a hybrid method for analyzing embedded real-time systems," in *Proceedings of the seventh ACM international conference on Embedded software*. ACM, 2009, pp. 107–116.

[11] "Timing Architects," http://www.timing-architects.com/.

[12] "ChronSIM," http://www.inchron.com/tool-suite/chronsim.html.

[13] "TraceAnalyzer," https://www.symtavision.com/products/symtas-traceanalyzer/.

[14] S. Quinton, T. T. Bone, J. Hennig, M. Neukirchner, M. Negrean, and R. Ernst, "Typical worst case response-time analysis and its use in automotive network design," in *Proceedings of the 51st Annual Design Automation Conference*, ser. DAC '14. New York, NY, USA: ACM, 2014, pp. 44:1–44:6. [Online]. Available: http://doi.acm.org/10.1145/2593069.2602977