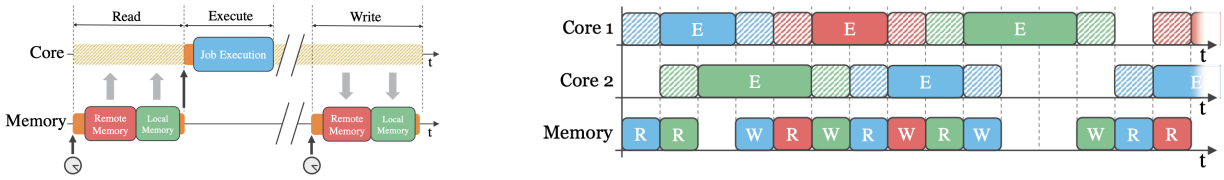
Code Generation and Compilation for Multi-Phase Tasks

Contact: Matthias Becker (mabecker@kth.se), KTH.
Location: Kista, Sweden.
Duration: 20 weeks, full time.

The advent of multi-core platforms brings many possibilities for embedded system designers. However, these architectures often implement complicated memory hierarchies that aggravate their timing analysis [1].
One way to tackle this problem is based on memory-centric approaches [2], where access to shared memory is co-scheduled with computation tasks. Such a treatment allows to account and plan for possible memory-contention scenarios which can significantly increase the predictability of the system.
Many industrial domains already today employ similar execution models. Typically, the execution is divided into three distinct phases, *read-execute-write* (see Fig. 1). While the read and write phase access shared memory, the execute phase only operates on local copies of variables and thus accesses no shared data. This model can for example be found in the automotive domain [3, 4] or in the avionics domain [5]. Approaches for dynamic [6, 7, 8, 9]. as well as static [4, 10] scheduling have been proposed for this model.
While this model provides predictability, applications are often not designed to follow the phased model. Manual transformation and management of memory operations is further cumbersome and source for errors. To alleviate those challenges, code generation and compilation approaches exist that produce PREM-compliant code [11, 12, 13, 14].



(a) (b)

Figure 1: (a) Memory and execution phases for one task instance (with read and write phase split into access to different memory areas). (b) Example schedule for the read-execute-write model.

Scope and Outline of the Master Thesis Project

The aim of the thesis is to investigate automated methods to transform and/or compile programs to follow the phased execution model and be compliant with an existing execution framework for mullti-phase tasks. The performance of the approach will be compared against unmodified applications on a real multi-core platform.

1. A state-of-the-art analysis of the thesis domain.
2. Conceptual design of the proposed code-transformation or compilation methods to generate tasks compliant to the phased execution model.
3. Implementation of the proposed method.
4. Extensive evaluations of the method.

References

[1] D. Dasari, V. Nelis, and B. Akesson, "A framework for memory contention analysis in multi-core platforms," *Real-Time Systems*, vol. 52, no. 3, pp. 272–322, May 2016.

[2] R. Pellizzoni, E. Betti, S. Bak, G. Yao, J. Criswell, M. Caccamo, and R. Kegley, "A predictable execution model for cots-based embedded systems," in *17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2011, pp. 269–279.

[3] S. Kramer, D. Ziegenbein, and A. Hamann, "Real World Automotive Benchmarks for Free," in *6th Int. Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems (WATERS)*, 2015.

[4] M. Becker, D. Dasari, B. Nicolic, B. Åkesson, V. Nélis, and T. Nolte, "Contention-free execution of automotive applications on a clustered many-core platform," in *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, 2016, pp. 14–24.

[5] C. Maia, L. Nogueira, L. M. Pinho, and D. G. Pérez, "A closer look into the aer model," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2016, pp. 1–8.

[6] A. Alhammad and R. Pellizzoni, "Schedulability analysis of global memory-predictable scheduling," in *International Conference on Embedded Software (EMSOFT)*, 2014, pp. 1–10.

[7] C. Maia, G. Nelissen, L. Nogueira, L. M. Pinho, and D. G. Pérez, "Schedulability analysis for global fixed-priority scheduling of the 3-phase task model," in *2017 IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2017, pp. 1–10.

[8] J. Arora, C. Maia, S. A. Rashid, G. Nelissen, and E. Tovar, "Schedulability analysis for 3-phase tasks with partitioned fixed-priority scheduling," *Journal of Systems Architecture*, vol. 131, p. 102706, 2022.

[9] T. Thilakasiri and M. Becker, "An exact schedulability analysis for global fixed-priority scheduling of the AER task model," in *28th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2023.

[10] M. Becker, S. Mubeen, D. Dasari, M. Behnam, and T. Nolte, "Scheduling multi-rate real-time applications on clustered many-core architectures with memory constraints," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018, pp. 560–567.

[11] R. Mancuso, R. Dudko, and M. Caccamo, "Light-prem: Automated software refactoring for predictable execution on cots embedded systems," in *2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*, 2014, pp. 1–10.

[12] F. Fort and J. Forget, "Code generation for multi-phase tasks on a multi-core distributed memory platform," in *2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2019, pp. 1–6.

[13] B. Forsberg, M. Solieri, M. Bertogna, L. Benini, and A. Marongiu, "The predictable execution model in practice: Compiling real applications for cots hardware," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 5, jul 2021. [Online]. Available: https://doi.org/10.1145/3465370

[14] Z. Gu and R. Pellizzoni, "Optimizing parallel prem compilation over nested loop structures," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1249–1254.