Experiences and Results of Parallelisation of Industrial Hard Real-time Applications for the parMERASA Multi-core

Theo Ungerer[1], Christian Bradatsch[1], Martin Frieb[1], Florian Kluge[1], Jörg Mische[1], Alexander Stegmeier[1], Ralf Jahr[1], Mike Gerdes[1⋆], Pavel Zaykov[2], Lucie Matusova[2], Zai Jian Jia Li[2], Zlatko Petrov[3], Bert Böddeker[4], Sebastian Kehr[4], Hans Regler[5], Andreas Hugl[5], Christine Rochange[6], Haluk Ozaktas[6], Hugues Cassé[6], Armelle Bonenfant[6], Pascal Sainrat[6], Nick Lay[7], David George[7], Ian Broster[7], Eduardo Quiñones[8], Milos Panic[8,9], Jaume Abella[8], Carles Hernandez[8], Francisco Cazorla[8,10], Sascha Uhrig[11], Mathias Rohde[11], and Arthur Pyka[11]

[1] University of Augsburg, Augsburg, Germany
[2] Honeywell International s.r.o., Brno, Czech Republic
[3] Honeywell EOOD, Sofia, Bulgaria
[4] DENSO AUTOMOTIVE Deutschland GmbH, Eching, Germany
[5] BAUER Maschinen GmBH, Schrobenhausen, Germany
[6] Université Paul Sabatier, Toulouse, France
[7] Rapita Systems Ltd., York, UK
[8] Barcelona Supercomputing Center, Barcelona, Spain
[9] Technical University of Catalunya, Barcelona, Spain
[10] Spanish National Research Council, Barcelona, Spain
[11] Technical University of Dortmund, Dortmund, Germany

theo.ungerer@informatik.uni-augsburg.de

Abstract. The EC FP-7 project parMERASA[1] (Multi-Core Execution of Parallelised Hard Real-Time Applications Supporting Analysability, Oct. 1, 2011 until Sept. 30, 2014) provides a timing analysable system of parallel hard real-time applications running on a scalable multi-core processor. parMERASA goes one step beyond mixed criticality demands: It targets future complex control algorithms by parallelising hard real-time programs to run on predictable multi-/many-core processors. A software engineering approach was developed to ease sequential to parallel program transformation by developing and supporting suitable parallel design patterns that are analysable. The following sequential hard real-time programs were parallelised by applying the pattern-oriented parallelisation approach: 3D path planning and stereo navigation algorithms (Honeywell International s.r.o.), diesel engine management system (DENSO AUTOMOTIVE Deutschland GmbH), and the control algorithm for a dynamic compaction machine (BAUER Maschinen GmbH). The paper reports on parallelisation approach, experiences made during parallelisation with applications, tools and multi-core architecture, scalability of applications and quantitative results reached.

⋆ Now with Autoliv B.V. & Co. KG

[1] http://www.parmerasa.eu/

1 Introduction

Providing higher performance than state-of-the-art embedded processors can deliver today will increase safety, comfort, number and quality of services, and lower emissions as well as fuel demands for automotive, avionic and automation applications. In the avionic industry, the ever increasing demands for additional aircraft functionality, safety and security drive markets towards the need for greater platform performance. These demands require a significant increase in the computational power hosted on board, together with the necessity for absolute guarantees on the timing performance of applications. In the automotive domain driver assistance systems and 'safety relevant' systems such as automatic emergency-braking triggered by collision avoidance systems can evaluate more sensor signals and master more complex situations if higher performance is provided by future control units. Hybrid car technology and fuel injection can be optimized to reduce gasoline consumption and emissions if better processor performance is available. Finally, on a similar level, automation systems from power plants and large medical equipment to construction machinery can incorporate more sensors and more powerful control algorithms.

The common feature of all the application domains discussed above is that they have hard real-time constraints, requiring absolute timing predictability, i.e. it is essential to guarantee the timing correctness of the application such that an execution deadline will never be missed. Processors in use today for embedded applications with hard real-time requirements are characterized by simpler architectures than desktop processors, encompassing single cores, short pipelines and often in-order execution only. However, the growing performance requirements for hard real-time systems to host more computationally intensive applications with high data throughput rates and low latency demands make it crucial that future processors are able to provide higher performance than today. Multi-core processor technology is considered as an effective solution to cope with the performance requirements of embedded systems.

Multi-core processors offer solutions to increase the overall aggregated performance of the system beyond the co-hosting of mixed criticality workloads within a single powerful processor. Higher performance levels can be achieved with multi-core processors if applications are parallelised, i.e. applications are split into threads that run in parallel on different cores and synchronize when they need to communicate. The main obstacle for harnessing the potential performance increase offered by multi-core technology is the parallelisation of state-of-the-art single-core applications to timing analysable thread-level parallel programs.

Hard real-time applications require absolute timing predictability in terms of Worst-Case Execution Time (WCET) analysability. There are two families of approaches for performing the WCET analysis of applications [22]: static WCET analysis and measurement-based WCET analysis. Each approach is complementary to the other. Static WCET analysis techniques are based on a timing model of the hardware architecture, while measurement-based approaches derive timings of small blocks of code by direct observation of the execution of the program on the target, avoiding the effort of having to build an exact timing model of the processor, but without the full guarantee to catch the real WCET. When

used together, these methods provide additional assurance of the correctness and precision of the approaches. In addition, a close interaction of both methods enables much tighter results. Path information can be derived by one method easily and used effectively in the other.

Existing academic and commercial WCET tools cannot analyse multi-threaded parallel applications running on multi-core processors. This gap should be filled to match the expected evolution of embedded software. That is, the individual WCETs of parallel threads must be properly combined and, in particular, communication times and waiting times at synchronization points must be determined as accurately as possible.

The next section introduces some related EC projects. Section 3 gives an overview of the parMERASA project and the reached results. Section 4 describes the parallelisation approach and Section 5 the parallelised applications. Section 6 focuses on the achieved results.

2 Related Projects

Several EC FP-7 projects have successfully shown that hard real-time capable multi-core system design is feasible at least for a small number of cores by a combination of hardware techniques, adapted WCET tools, and timing analysable system software. Thus, the PREDATOR project (2007 – 2010) aimed to reduce the uncertainty in the timing behaviour of multi-core processors by providing system design guidelines at hardware and software level. PREDATOR targeted the problem of predictability by developing deterministic designs which enable traditional static and measurement-based WCET analysis approaches. The JEOPARD project (2007 – 2010) aimed to develop a new framework for Java-based real-time applications on modern multi-core processor systems. The strategic objective of the JEOPARD project was to provide the tools for platform independent development of predictable systems that make use of multi-core platforms. The PROARTIS project (2010 – 2013) and its successor project PROXIMA investigate probabilistic timing analysis. The central hypothesis of these projects is that new advanced hardware/software features such as multi-cores enabling truly randomized timing behavior can be defined for use in critical real-time embedded systems. The T-CREST project (2011 – 2014) researched a timing predictable system that simplifies the safety argument with respect to maximum execution time while striving to double performance for four cores and to be four times faster for 16 cores than a standard processor of the same technology (e.g. FPGA). The ultimate goal of the T-CREST system was to lower costs for safety relevant applications, reducing system complexity and at the same time achieving faster timing predictable execution. CERTAINTY project (2011 – 2014) focused on the certification process for mixed-critical embedded systems featuring functions dependent on information of varying confidence levels. Its key objective was to push forward the certification of real-time mixed critical embedded systems, a process currently challenged by the choices made at application design time about reliability and disturbances handling which deals with the management of interferences between different functions of complex control software over the whole system.

The parMERASA project (2011 – 2014) builds upon the results of MERASA project (2007 – 2010) that focused on execution of hard real-time tasks on multi-cores with a relatively small number of cores. The shared-memory and bus-based techniques of the MERASA processor actually limited the scalability to about four to eight cores. Instead, parMERASA focused on parallelisation and parallelisation support tools for hard real-time applications and targeted a much higher performance with a scalable multi-core.

3 Project Overview

A pattern supported parallelisation approach was developed at University of Augsburg to guide the parallelisation of the legacy software and allow usage of standard tools. The approach eases sequential to parallel program transformation by developing and supporting suitable parallel design patterns and algorithmic skeletons that are analysable with WCET tools. The approach was applied to successfully parallelise four use cases: 3D path planning algorithm and stereo navigation algorithm (Honeywell International s.r.o.), diesel engine management system (DENSO AUTOMOTIVE Deutschland GmbH), and the control algorithm for a dynamic compaction machine (BAUER Maschinen GmbH).

The parallelisation work-flow was supported by several tools. The static WCET analysis tool OTAWA [4] of University of Toulouse was enhanced by modelling the parMERASA multi-core processor and extended with support for the timing analysis of parallel programs. Source code annotations for parallelisation analysis were defined that are based on the parallel design patterns. The new OTAWA tool is available as Open Source software, too.

Tools developed by Rapita Systems Ltd comprise (1) On-target timing and WCET analysis tool RapiTime enhanced for parallel programs; (2) On-target code coverage tool RapiCover with support for code coverage for parallel programs; (3) Constraint verification tool RapiCheck for constraint checking of parallel programs; (4) Dependency Analysis tool to assist with the parallelisation of existing sequential software; (5) Visualization and profiling tool RapiTask for parallel programs. Barcelona Supercomputing Center (BSC) developed a mapping tool to statically allocate tasks to cores of the parMERASA multi-core.

Fig. 1 shows the overall system architecture [6] that was defined to support the execution of parallelised hard real-time applications and their WCET analysability. Based on the requirements of the different, industrial application domains, the parMERASA system software comprises domain specific runtime environment (RTE) services to support the parallelised applications. The AUTOSAR standard for the automotive domain, ARINC 653 specification for the avionic domain, respectively the BIOS for the construction machinery domain serve as basis for these RTEs.

The system software is based on a common Kernel Library [5] developed at University of Augsburg. The Kernel Library also represents the common basis for the domain specific RTE implementations. It acts as a hardware abstraction layer and provides the basic functionalities required by the application domain specific RTE services that is scheduling, protection, communication & synchronization, and I/O. The Kernel Library provides timing-analysable synchronization primitives based on ticket-locks, context management as basis for domain-specific

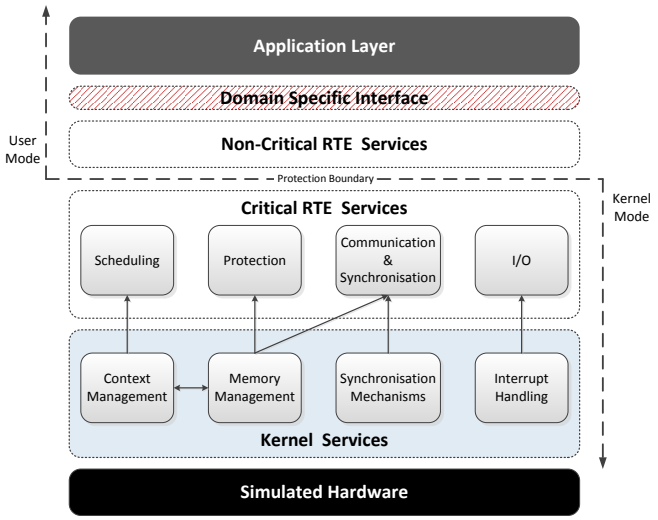

Fig. 1. parMERASA system architecture

scheduling, MMU and interrupt handling. The implementation of RTE services is divided by a protection boundary into non-critical/critical services executed in user/kernel mode respectively. To ensure time and space partitioning, only critical services can influence other partitions. Non-critical services have no access across partition boundaries except if explicitly allowed, for example through memory mapping. In all cases, extensions to support advanced parMERASA mechanisms of synchronization, inter-core communication, and parallel design patterns were implemented. The system software is open source.

parMERASA target applications rely on incremental qualification, that allows each system component to be subject to formal certification (including timing analysis) in isolation and independently of other components, with obvious benefits for cost, time and effort. As a result, they impose the processor architecture to provide mechanisms to guarantee time and space isolation among applications. Moreover, such a property must remain the same even when moving towards parallel execution.

To that end, BSC developed two novel concepts [16]: parallel Software Partitions (pSWPs) and Guaranteed Resource Partitions (GRPs) that provide incremental qualification for parallel hard real-time applications:

- pSWPs extend the functionality of software partitions (as defined in ARINC653 [3] and ISO26262 [10] standards) to allow parallel execution on multi-core processors. pSWP guarantees that parallel tasks belonging to one application cannot affect the timing (and functional) behaviour of parallel tasks belonging to other applications.
- GRP defines a hardware execution environment composed of a cluster of processor resources, including cores, NoC resources, memory, etc., in which pSWPs run, providing the desirable time isolation properties as defined above. A fundamental property that GRPs must accomplish is time predictability, i.e. the response time of the different processor resources must

be known. Note that the GRP is, in fact, the hardware counterpart of the pSWP: while the pSWP encapsulates parallel applications to provide the desirable time isolation properties imposed by the standards, the GRP encapsulates the pSWP to provide the required time isolation guarantees at the hardware level.

Figure 2(a) shows a block diagram of the envisioned architecture composed of two GRPs, each formed by four cores, a NoC and a memory device. Figures 2(b) and (c) shows two specific implementations of the parMERASA architecture. Figure 2(b) presents a *clusterised two-level hierarchical NoC architecture* composed of a first-level NoC (a tree) to connect cores and a second-level NoC (a bus) to connect clusters among them. Regular NoC designs such as meshes also allow to define GRPs that fulfil the time isolation requirements. To do so, virtual clusters are defined by grouping adjacent cores in rectangular shapes (i.e. organizing cores in groups of 2, 4, 6, 8, 9) with a memory device connecting to one of the cores. In this case, if XY or YX routing policy is used, an isolated communication island is created with properties similar to those of clustered architectures. Figure 2(c) shows a processor implementing a mesh defining four GRPs: GRP1 composed of 6 cores, GRP2 composed of 2 cores and GRP3 and GRP4 composed of 4 cores each.

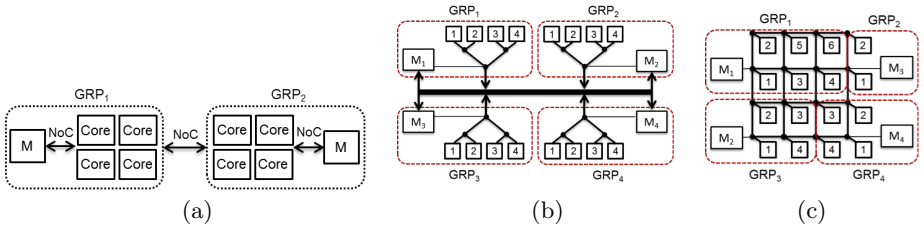

Fig. 2. Different implementations of the parMERASA multi-core processor architecture: (a) general architecture, (b) clusterised implementation and (c) mesh-based implementation.

Moreover, the parMERASA multi-core architecture relies on data caches to reduce latencies and impact of interferences when multiple cores want to access the memory. In this case, because multiple cores can share data, it is mandatory to guarantee coherent data accesses. WCET analysis of standard coherent multi-core caches is particularly hard or even impossible. Therefore a predictable cache coherence mechanism has been developed by Technical University of Dortmund: the On-Demand Coherent Cache (ODC^2) [18–20]. ODC^2 guarantees coherent accesses only to shared data that is accessed inside critical regions and between barriers. Therefore, all local caches are kept free from shared data outside critical regions. Nevertheless, the applications can profit from the advantage of caches inside as well as outside of critical regions. Because the proposed technique is free from cache-to-cache communication, a tight worst-case execution time analysis on the level of a single-core data cache analysis is feasible.

4 Parallelisation Approach for Industrial Hard Real-time Programs

The parMERASA pattern-supported parallelisation approach developed at University of Augsburg [12, 13] is based on two well-known parallelisation approaches from the HPC domain: The methodology is derived from the PCAM approach by Foster [7]; the use of parallel design patterns is adapted from the approach by Mattson et al. [15]. Both approaches are combined and enhanced to support WCET analysis of parallel programs by predictable parallel design patterns (PDPs). The parMERASA pattern-supported parallelisation approach defines a model-based development path from sequential legacy programs to timing analysable parallel programs. Compared to development of parallel software from scratch, the development and testing effort is strongly reduced because of (a) high reuse of code from the sequential implementation and (b) allowing only best practice, clearly defined, and analysable parallel design patterns to introduce parallelism. They are defined, together with platform dependent and timing analysable synchronization idioms [8], in the Pattern Catalogue [9].

The approach comprises two phases: First, based on the sequential implementation, a model similar to the UML2 Activity Diagram consisting only of sequential code blocks and PDPs is constructed. The goal is to express a high degree of parallelism. The potential (WCET) speed-up is assessed by rating the computation versus synchronization/communication overhead. Second, if a higher (WCET) speed-up can be expected with less parallelism, this model will be refined towards an optimal level of parallelism by agglomeration of its code blocks and collapsing of PDPs to sequential code blocks. The first phase is platform independent whereas in the second phase all the trade-offs and limitations of the target platform have to be taken into account. Next the real coding can be started; algorithmic skeletons [21] are available in an optimal case for efficient implementation of the PDPs.

Fig. 3 shows the principal tools developed to support the parMERASA pattern-supported parallelisation approach. These concern the Dependency Analysis tool of Rapita Systems Ltd to assist parallelisation by the application programmer towards the PDPs, which represent the model-based stage of the approach. Low overhead for static WCET analysis will be assured by (a) allowing only analysable parallel design patterns and (b) an extension of the Patterns Catalogue with requirements for static WCET analysis. In the intermediate stage UML-based verification tools could be used, but also the Speed-up Approximation and Parameter Optimization tool [14] developed by University of Augsburg. Mutator Functions and Algorithmic Skeletons [14] help while coding the parallel program. The Mapping tool [17] developed at BSC assists in mapping tasks to cores. Next the OTAWA tool performs a static WCET analysis of the parallel code. The tools of Rapita Systems Ltd require instrumentation of the parallel code and running the code on the target hardware, in case of parMERASA multicore on its simulator, to collect comprehensive traces. RapiTime then performs a measurement-based WCET analysis, RapiCover shows the code coverage, RapiTask profiling and visualization, and RapiCheck allows constraint checking.

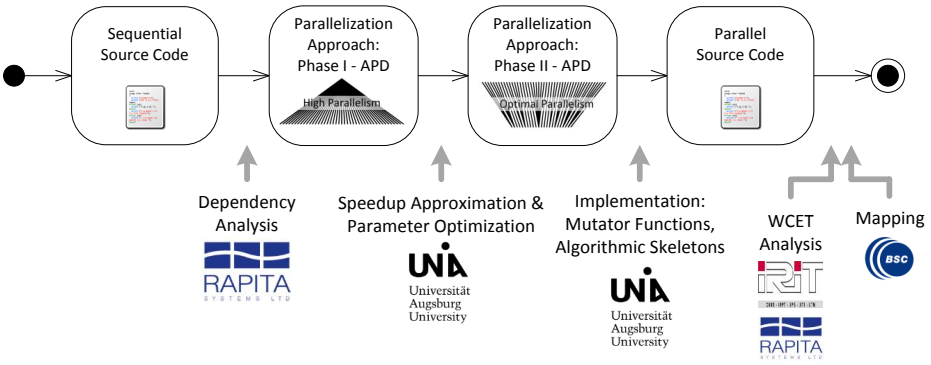

Fig. 3. Overview of the pattern-supported parallelisation process and suggested tools to ease process (analysis, optimization, and implementation)

5 Parallelised Applications

Honeywell International s.r.o. parallelised a 3D path planning (3DPP)/collision avoidance system and a stereo navigation system. The 3DPP algorithm provides a good basis for parallelisation, and as such can be easily configured to evaluate the execution time impact of different multi-core architectures. The 3DPP application is based on the Laplacian multi-grid algorithm that has been extended to take sensor data and use it to set up boundary conditions for the grids. The stereo navigation application comprises several pipelined stages starting from an image of a stereo camera.

The automotive supplier DENSO AUTOMOTIVE Deutschland GmbH parallelised an existing diesel engine management system (EMS). This is a typical application for controlling a combustion engine. Its software structure comprises many cyclic or event-driven functions called runnables. A key issue of parallelising automotive software is the high number of dependences between runnables. For that reason, an application specific parallelisation approach is developed, which uses timing properties and data dependences as constraint. Parallelisation is done on three levels:

1. Inter-task level: parallel execution of AUTOSAR tasks that comprise several runnable.
2. Intra-task level: parallel execution of runnables within a task by distribution runnables over cores.
3. Intra-runnable: runnables with a high WCET are itself parallelised to execute on multiple cores of one cluster.

Inter-task level was optimized by combining runnables scheduled to the same point in time into single tasks. Inter-task and intra-task level activities were statically mapped to cores using the BSC mapping tool [17]. For Intra-runnable level several large runnables were analysed for potential parallelism. The parallelisation is based on PDPs and Timing Analysable Algorithmic Skeletons (TAS) and demonstrated on a single selected runnable.

Current control algorithms of BAUER Maschinen are sequential, but in future, the control applications for construction machines will be more complicated (more automatic function, more safety and security functions are expected). So limits in performance, time, and code structure of the sequential code will be reached. BAUER Maschinen GmbH parallelised the control algorithm for a large compaction machine. The main control loop was parallelised such that each supervised sub-task (PWM, CAN connected, I/O) could be mapped on a different core. Code was divided into periodic and non-periodic tasks. Periodic tasks derived from the main loop were executed on two dedicated cores. Timing-analysable algorithmic skeletons were applied to the non-periodic tasks. Detailed information on algorithmic structures of all case studies is provided in [1, 2].

The parMERASA pattern-based parallelisation approach was applied to parallelise all applications based on the respective PDPs of the Pattern Catalog and its Algorithmic Skeletons. Table 1 correlates the applied PDPs to the applications.

Table 1. Applied PDPs to applications

Application		**Task Parallelism**	**Periodic Task Parallelism**	**Data Parallel**	**Parallel Pipeline**
Honeywell: 3DPP				X	X
Honeywell: Stereo Nav.				X	X
Bauer:	Compaction M.	X	X		
DENSO:	Diesel EMS	X	X		

6 Experiences and Results of Parallelisation of Industrial Hard Real-time Applications

To find out, whether the parMERASA approach is feasible, we have to compare the performance of the original sequential programs with their parallelised versions. Performance of parallel programs is typically measured by running the sequential and the parallelised programs on scalable hardware – COTS processors or in our case the parMERASA multi-core simulator. Dividing the execution time of the sequential program by the execution time of the parallelised version delivers the general speed-up, and dividing the speed-up through the number of threads/cores gives the efficiency.

For our hard real-time applications, however, the general speed-up is of less importance compared to the performance improvement in the WCET. We therefore define the WCET speed-up as the WCET of the sequential program divided by the WCET of the parallelised program, and WCET efficiency again by dividing the WCET speed-up by the number of threads/cores. To ease analysability we run each thread on another core. For static WCET speed-up we use OTAWA tool with its architectural model of the predictable parMERASA multi-core architecture; for measurement-based WCET speed-up we use RapiTime tool based on measurements on the parMERASA simulator.

For speed-up calculations we require a sequential program, which is the case for the two Honeywell applications. The BAUER and DENSO applications, however, are comprised of a set of interrupt-driven tasks running on a single-core processor, together with a main loop in the BAUER case study. For the diesel EMS we evaluate inter-task parallelism by the overall runtime of the scheduled tasks on a single core versus the maximum of the runtimes of the tasks distributed over several cores. Intra-task parallelism is evaluated by the runtime of a task versus the runtime of the parallel version where all runnables of the task are distributed over different cores. Intra-runnable parallelism is derived from parallelising the sequential runnable. For the BAUER application we compare the runtime of the main loop with parallelised versions of the main loop with additionally spreading the interrupt-driven tasks onto further cores.

Table 2 presents the achieved WCET speed-ups of the WCET-aware parallelisation of the 3DPP application. The algorithm of the application allows a theoretical maximum speed-up of half of the number of threads. A comparison of several different implementations showed the superiority of a barrier-implementation versus a lock-/condition variable based implementation of the Data Parallel pattern for WCET analysability and static WCET speed-up [11]. The stereo navigation algorithm proved as hard to parallelise and achieved only partly speed-ups for algorithmic stages. For feature extraction, one of the computationally intensive computing stages, we achieved a measurement based WCET speed-up of 1.5 times for a 12 thread configuration.

Table 2. Theoretical maximum and WCET speed-ups of the parallelised 3DPP application derived by OTAWA and RapiTime

# Threads	8	16
OTAWA	2.10	2.81
RapiTime	2.60	4.58
Theoretical	4.00	8.00

For the diesel EMS three levels of parallelism was investigated. Intra-runnable parallelism was exploited with the pattern-based approach. A WCET speed-up of up to 2.3 was reached by using RapiTime on 4 cores for one runnable. Intra-task parallelism was exploited with the parMERASA mapping tool. A static WCET speed-up of up to 3.3 was estimated on 4 cores for a single task. The concept of supertasks (i.e., tasks scheduled to the same point in time are treated as one task respecting data dependencies) further improved the speed-up. Inter-task parallelism was exploited with timed implicit communication. A static WCET speed-up of up to 4.46 was estimated by mapping of all tasks on 8 cores. Eventually, intra- and inter-task parallelism were combined (one task was distributed over 2 cores) and the static WCET speed-up estimate increased to 5.97 on 8 cores. Hence, the efficiency of the combined approach is 0.75.

The WCET-aware parallelisation of dynamic compaction machine application showed that multi-core platforms are applicable and that a static WCET

speed-up of 2.38 can be reached with 4 cores. A higher number of cores decreases the static WCET speed-up.

The overall experience from parallelising the four industrial algorithms suggests that parallelising real-world hard real-time applications that run successful on single-core reach moderate speed-ups on multi-cores. The scalability of such algorithms is limited. In particular static WCET speed-ups suffer from high pessimism caused by global memory accesses in a multi-core.

7 Conclusion

The EC FP-7 project parMERASA (Oct. 1, 2011 until Sept. 30, 2014) investigated a timing analysable system of parallel hard real-time applications running on a scalable multi-core processor. parMERASA goes one step beyond mixed criticality demands: It targets future complex control algorithms by parallelising hard real-time programs to run on predictable multi-/many-core processors. A major breakthrough in the project was certainly achieved by having shown the ability to increase performance by parallelising hard real-time applications onto multi-core hardware.

parMERASA project paved the way for future high-performance embedded systems applications. The hard real-time control algorithms of such future systems will not be limited in their performance by single-core processors. More complex control algorithms than today can be applied. Such newly developed control algorithms should be scalable and able to utilize the performance increase offered by multi- and many-core processors.

Acknowledgement

The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement no. 287519.

References

1. Deliverable 2.4 – "sequential to parallel program development path and parallel execution patterns" (Sep 2014), `http://www.parmerasa.eu/`
2. Deliverable 2.6 – "final evaluation results on parallel industrial applications" (Sep 2014), `http://www.parmerasa.eu/`
3. ARINC Inc.: ARINC Specification 653: Avionics Application Software Standard Standard Interface, Part 1 and 4, Subset Services (June 2012)
4. Ballabriga, C., Cass, H., Rochange, C., Sainrat, P.: Otawa: An open toolbox for adaptive wcet analysis. In: Min, S., Pettit, R., Puschner, P., Ungerer, T. (eds.) Software Technologies for Embedded and Ubiquitous Systems, Lecture Notes in Computer Science, vol. 6399, pp. 35–46. Springer Berlin Heidelberg (2011)
5. Bradatsch, C., Kluge, F.: parmerasa multi-core rtos kernel. Tech. Rep. no. 2013-02, University of Augsburg (Feb 2013)
6. Bradatsch, C., Kluge, F., Ungerer, T.: A cross-domain system architecture for embedded hard real-time many-core systems. In: 11th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC-13). IEEE (Nov 2013)

7. Foster, I.: Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1995)
8. Gerdes, M., Kluge, F., Ungerer, T., Rochange, C., Sainrat, P.: Time analysable synchronisation techniques for parallelised hard real-time applications. In: Design, Automation Test in Europe Conference Exhibition (DATE). pp. 671–676 (Mar 2012)
9. Gerdes, M., Jahr, R., Ungerer, T.: parMERASA Pattern Catalogue: Timing predictable parallel design patterns. Tech. Rep. no. 2013-11, University of Augsburg (Nov 2013)
10. ISO: Road vehicles – Functional safety – Part 6: Product development at the software level, Ref. num. ISO 26262-6:2011(E) (2011)
11. Jahr, R., Gerdes, M., Ungerer, T., Ozaktas, H., Rochange, C., Zaykov, P.: Effects of structured parallelism by parallel design patterns on embedded hard real-time systems. In: Embedded and Real-Time Computing Systems and Applications (RTCSA), IEEE 20th International Conference on. pp. 1–10 (Aug 2014)
12. Jahr, R., Gerdes, M., Ungerer, T.: On efficient and effective model-based parallelization of hard real-time applications. In: Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme IX. pp. 50–59 (Apr 2013)
13. Jahr, R., Gerdes, M., Ungerer, T.: A pattern-supported parallelization approach. In: Proceedings of the 2013 International Workshop on Programming Models and Applications for Multicores and Manycores (PMAM). pp. 53–62 (Feb 2013)
14. Jahr, R., Stegmeier, A., Kiefhaber, R., Frieb, M., Ungerer, T.: User manual for the optimization and wcet analysis of software with timing analyzable algorithmic skeletons. Tech. Rep. no. 2014-05, University of Augsburg (May 2014)
15. Mattson, T., Sanders, B., Massingill, B.: Patterns for parallel programming. Addison-Wesley Professional, first edn. (2004)
16. Panic, M., Quinones, E., Zaykov, P., Hernandez, C., Abella, J., Cazorla, F.: Parallel many-core avionics systems. In: Proceedings of the 14th International Conference on Embedded Software (EMSOFT). pp. 26:1–26:10 (Oct 2014)
17. Panic, M., Kehr, S., Quiñones, E., Böddeker, B., Abella, J., Cazorla, F.J.: Runpar: An allocation algorithm for automotive applications exploiting runnable parallelism in multicores. In: 12th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) (2014)
18. Pyka, A., Tadros, L., Uhrig, S., Cass, H., Ozaktas, H., Rochange, C.: WCET analysis of parallel benchmarks using on-demand coherent cache. In: 3rd Workshop on High-performance and Real-time Embedded Systems (HiRES) (2015)
19. Pyka, A., Rohde, M., Uhrig, S.: A real-time capable first-level cache for multi-cores. In: Workshop on High Performance and Real-time Embedded Systems (HiRES) in conjunction with HiPEAC'13 (Jan 2013)
20. Pyka, A., Rohde, M., Uhrig, S.: A real-time capable coherent data cache for multicores. Concurrency and Computation: Practice and Experience 26(6), 1342–1354 (2014)
21. Stegmeier, A., Frieb, M., Jahr, R., Ungerer, T.: Algorithmic skeletons for parallelization of embedded real-time systems. In: 3rd Workshop on High-performance and Real-time Embedded Systems (HiRES) (2015)
22. Wilhelm, R., Engblom, J., Ermedahl, A., Holsti, N., Thesing, S., Whalley, D., Bernat, G., Ferdinand, C., Heckmann, R., Mitra, T., Mueller, F., Puaut, I., Puschner, P., Staschulat, J., Stenström, P.: The worst-case execution-time problem—overview of methods and survey of tools. ACM Trans. Embed. Comput. Syst. (TECS) 7(3), 36:1–36:53 (May 2008)