2013 16th Euromicro Conference on Digital System Design

# parMERASA – Multi-Core Execution of Parallelised Hard Real-Time Applications Supporting Analysability

T. Ungerer*, C. Bradatsch*, M. Gerdes*, F. Kluge*, R. Jahr*, J. Mische*, J. Fernandes†, P. G. Zaykov†, Z. Petrov†, B. Böddeker‡, S. Kehr‡, H. Regler§, A. Hugl§, C. Rochange¶, H. Ozaktas¶, H. Cassé¶, A. Bonenfant¶, P. Sainrat¶, I. Broster‖, N. Lay‖, D. George‖, E. Quiñones**, M. Panic**‡‡, J. Abella**, F. Cazorla**ˣ, S. Uhrig††, M. Rohde†† and A. Pyka††

*University of Augsburg, Augsburg, Germany
†Honeywell International s.r.o., Brno, Czech Republic
‡DENSO AUTOMOTIVE Deutschland GmbH, Eching, Germany
§BAUER Maschinen GmbH, Schrobenhausen, Germany
¶Université Paul Sabatier, Toulouse, France
‖Rapita Systems Ltd., York, UK
**Barcelona Supercomputing Center, Barcelona, Spain
‡‡Technical University of Catalunya, Barcelona, Spain
ˣSpanish National Research Council, Barcelona, Spain
††Technical University of Dortmund, Dortmund, Germany

http://www.parmerasa.eu
Email: ungerer@informatik.uni-augsburg.de

***Abstract*—Engineers who design hard real-time embedded systems express a need for several times the performance available today while keeping safety as major criterion. A breakthrough in performance is expected by parallelizing hard real-time applications and running them on an embedded multi-core processor, which enables combining the requirements for high-performance with timing-predictable execution.**

**parMERASA will provide a timing analyzable 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 parallelizing hard real-time programs to run on predictable multi-/many-core processors. We aim to achieve a breakthrough in techniques for parallelization of industrial hard real-time programs, provide hard real-time support in system software, WCET analysis and verification tools for multi-cores, and techniques for predictable multi-core designs with up to 64 cores.**

## I. Introduction

The EC FP-7 parMERASA project (Oct. 1, 2011 until Sept. 30, 2014) targets a timing analyzable system of parallel hard real-time applications running on a scalable multi-core processor. Several new scientific and technical challenges arise in the light of timing analyzability: parallelization techniques for industrial applications, timing analyzable parallel design patterns, operating system visualization and efficient synchronization mechanisms, guaranteed worst-case execution times (WCET) of parallelized applications, verification and profiling tools, and scalable memory hierarchies together with I/O systems for multi-core processors.

The objective of the parMERASA project is an at least eightfold performance improvement of the WCET for parallelized legacy applications in avionics, automotive, and construction machinery domains in comparison to the original single-core versions. Application companies selected suitable sequential hard real-time programs that will be parallelized. These applications are: for avionics (Honeywell International s.r.o.) 3D Path Planning algorithm intended for airborne collision avoidance, Stereo Navigation intended for aircraft localization, and Global Navigation Satellite System; for automotive (DENSO AUTOMOTIVE Deutschland GmbH) an diesel engine management system; and for construction machinery (BAUER Maschinen GmbH) the control algorithm for a dynamic compaction machine. A software engineering approach was developed to ease sequential to parallel program transformation by developing and supporting suitable parallel design patterns that are analyzable.

Static WCET analysis techniques for parallel programs are under development and will be included into the OTAWA [1] tool. The envisioned parMERASA multi-core processor and system software provides temporal and spatial isolation between tasks and scale up to 64 cores. Five verification and parallelization support tools are under development by Rapita Systems Ltd. based on the application companies' requirements: (1) Extension of RapiTime[1] to provide on-target timing and WCET analysis tool for parallel programs; (2) Extension of RapiCover to provide on-target code coverage tool for parallel programs; (3) Memory, cache and stack analysis tool

[1] http://www.rapitasystems.com/products/RapiTime


DOI 10.1109/DSD.2013.46

for parallel programs; (4) Tool to assist with the parallelization of existing sequential software; (5) Visualization and profiling tool for parallel programs.

parMERASA will impact new products for transportation systems and industrial applications. It will influence automotive and avionic standards by introducing parallel execution and timing predictability as key features. This will contribute to reinforce the EC position in the field of critical computing systems and yield an advantage for European industry in the highly competitive embedded systems markets.

The next section analyses market and application requirements. Section III presents some related EC projects and section IV gives an overview of the project objectives. Section V highlights the results reached so far after 18 months of parMERASA project run-time.

## II. Market and Application Requirements Driving the Project

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. Such a demand for increased computational performance is widespread among key European industries.

In the avionic industry, the ever increasing demands for additional aircraft functionality, safety and security drive markets towards the need for greater platform performance. For instance, next-generation UAVs (unmanned aerial vehicles) are expected to be much more complex in terms of sensing, likely to involve an active electronically scanned area. All these requests will presumably result in a significant increase in the computational power hosted on board, together with the necessity for absolute guarantees on the timing performance of applications. Moreover, it will certainly be desirable to host the workload on as few hardware platforms as possible in order to reduce size, weight, and power (SWaP). Such a trend can already be observed in the latest generation Airbus aircraft A380 with a sixfold increase in the volume of the software control system with respect to the previous generation aircrafts A330/A340 [2].

Weight constraints in the automotive industry are less stringent than in avionics, but cost demands are significantly more severe. The German Federal Ministry of Education and Research states in its "Report on Information and Communication Technology in Year 2020" (IKT 2020) [3] that 80% of all innovations in passenger cars are due to software and electronics, and computation related applications in the automotive sector are increasing from year to year. 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, more powerful control algorithms, degrade performance gracefully saving highly expensive down-times in the event of sensor or machine defects.

Satisfying these demands is of importance not only to maximize long-term economic benefits, but especially to ensure the health and well-being of EC citizens by achieving maximum levels of safety on the road and in the air; and by providing for a high level of environmental protection regarding emissions of atmospheric pollutants [4].

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 in-order execution. However, the growing performance requirements for hard real-time systems to host more computationally intensive applications with increasingly high data throughput rates makes it crucial that future processors are able to provide higher performance than today. Multi-core processor technology is increasingly considered as an effective solution to cope with the performance requirements of embedded systems.

This shift towards multi-core processors can be observed in the avionic domain, which currently investigates multi-core platforms and problems associated with their usage for hosting mixed criticality applications, as well as their impact on verification and certification processes.

Similar trends can be observed in the automotive domain: applications are under constant cost pressure, while demands are made for higher performance because of new standards and quality of service requirements. For this reason, the automotive industry is investigating suitable "compound ECUs" that compose the functionality of several conventional ECUs, and multi-cores are of particular interest to perform the aggregated tasks concurrently on separate cores. The benefit of co-hosting applications on a single multi-core processor is not only a reduction in recurring costs, but also to reduce the volume of communication required among processors, resulting in a simplified architecture which decreases testing and development costs.

Moreover, 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 parallelized, i.e. applications are split into threads that run in parallel on different cores and synchronize when they need to communicate.

The upcoming multi-core processor technologies put high demands on application companies: single-core oriented programs must be parallelized in order to exploit the performance benefits of parallel execution on a multi-core processor. The main obstacle for harnessing the potential performance increase offered by multi-core technology is the parallelization of state-of-the-art single-core applications to timing analyzable thread-level parallel programs. The application industry typically lacks the parallelization know-how and adequate tools.

Hard real-time applications require absolute timing predictability in terms of WCET analyzability. There are two families of approaches for performing the WCET analysis of applications [5]: 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 as evidence or path information can be derived by one method easily and used effectively in the other.

However, when moving towards parallel execution considering medium- or large-scale multi-cores, both approaches raise new issues such as analysis of the communication latencies on a network-on-chip (NoC) or determining the behavior of a multi-level internal memory hierarchy physically distributed over the nodes. Existing academic and commercial WCET tools cannot analyze multi-threaded parallel applications and 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.

Support for on target verification is essential in any certification process. Verification is concerned with the provision of the assurance that requirements are satisfied, i.e. building it right, whereas validation addresses the problem of ensuring that the requirements are correct, i.e. building the right thing. In this project we are focusing on verifying the requirements. A typical verification example is the requirement to demonstrate 100% statement code coverage on target, or even 100% MC/DC (Modified Conditioning/Decision Coverage) coverage for the highest safety levels for avionic applications as required for example by the Do-178B airborne system and equipment standard[2]. In addition, other on target analysis related to debugging and performance profiling of parallel execution are key requirements. Existing tools used for parallelization of server and supercomputer applications do not take timing requirements into account and are not applicable to real-time embedded systems. Thus, providing generic on target verification and timing analysis tools to support the provision of such evidence and understanding of the behavior of the whole system becomes essential.

## III. Related Projects

Several EC FP-7 projects, MERASA, PREDATOR, PROARTIS, JEOPARD and T-CREST, 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 analyzable system software.

Thus, the PREDATOR project[3] (2007 - 2010) aimed to reduce the uncertainty in the timing behavior 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[4] (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[5] (2010 - 2013), whose participants also include a subset of the parMERASA partners, aims to facilitate a probabilistic approach to timing analysis. The central hypothesis of PROARTIS 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 (CRTE) systems.

The T-CREST project[6] (2011 - 2014) is developing a timing predictable system that will simplify 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 will be to lower costs for safety relevant applications, reducing system complexity and at the same time achieving faster timing predictable execution.

The parMERASA project builds upon the results of the completed MERASA project (2007 - 2010) that focused on single-threaded 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 focuses on parallelization and parallelization support tools for hard real-time applications and targets a much higher performance with a multi-core that should scale up to 64 cores. The parMERASA multi-core replaces the real-time bus by a scalable timing analyzable (NoC) interconnection. The MERASA core simulator is compatible with the TriCore instruction set architecture from Infineon Technologies. The parMERASA platform is based on the PowerPC instruction set architecture for the parMERASA core. The developed techniques, however, should be applicable to any other high-performance embedded processor with an appropriate hardware structure.

## IV. Project Objectives

The aim of the parMERASA project is to parallelize hard real-time applications in avionic, automotive and construction machinery domains and execute the resulting parallel programs on a timing predictable multi-core platform providing a much higher guaranteed performance than contemporary solutions.

The two overall goals of the project are: (1) an increase of the WCET performance of hard real-time embedded systems

[2] http://www.do178site.com/
[3] http://www.predator-project.eu/
[4] http://www.jeopard.org/
[5] http://www.proartis-project.eu/
[6] http://www.t-crest.org/

by targeting timing predictable multi-cores; (2) an improvement of performance of legacy code by starting from existing industrial code and migrating it to parallel code preserving timing predictability. Additional implicit goals of the project are to provide a significant reduction of the design effort and cost compared to the current state-of-the-art techniques.

Measurable objectives identified to achieve the overall goals:

O1: Develop a software engineering approach targeting WCET-aware parallelization techniques and parallel design patterns that favor WCET analysis. The software engineering approach should define a development path leading from sequential legacy programs to parallel programs and maximize software reuse. The project will develop at least four parallel design patterns that are analyzable. These patterns should be applicable to both commercial off-the-shelf (COTS) multi-cores and the parMERASA platform and will be demonstrated during the case studies.

O2: Achieve parallelization of the industrial case studies by adapting the software engineering approach and applying suitable parallel design patterns. Deliver higher performance than single core processors and achieve at least an eightfold WCET speedup (WCET of the sequential program divided by the WCET of the parallel program). This metric will be demonstrated by applying the parMERASA WCET analysis tools on the avionic, automotive, and construction machinery case studies.

O3: Develop on target verification tools, in particular WCET analysis tools with less than 25% pessimism on WCET estimates of parallel programs. Further tools should be developed concerning code coverage and memory analysis as well as parallel program profiling and visualization. This objective will be demonstrated by review of evidence that can support a certification argument and by applying the tools to the case studies.

O4: Develop a system architecture and system-level SW that supports WCET analyzable parallelization. This will be demonstrated by construction and evaluated during the case studies.

O5: Develop a scalable and timing analyzable multi-core processor with at least a 16 fold average speed-up with 64 cores. This objective will be demonstrated by construction of a simulator prototype and evaluated by running the case studies on the simulator.

O6: Contribute to standards. We aim at providing at least four recommendations to either automotive or avionic standards. In particular we will propose a concept for applying the developed parallel design patterns to AUTOSAR and the IMA standard ARINC 653.

O7: Contribute to open source software. The software developed at the universities, i.e. the static WCET tool OTAWA, the developed system software and the parMERASA simulator, will be made publicly available under an open source license at the end of the parMERASA project.

To reach the objectives the project was structured in three main phases: requirement specification and concept (completed), implementation and parallelization towards maximum parallelism (in progress), and optimization, refinement and evaluation.

## V. Achieved Results

### A. Parallelization of Industrial Hard Real-time Programs and Real-time Capable Parallel Design Patterns

The *parMERASA pattern-supported parallelization approach* [6], which is an extension of the approach described by Jahr et al. [7] of University of Augsburg, defines a model-based development path from sequential legacy programs to timing analyzable 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 analyzable *parallel design patterns* to introduce parallelism. They are defined, together with platform dependent and timing analyzable *synchronization idioms* [8], in the *Pattern Catalogue*.

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 parallel design patterns is constructed. The aim is to express an extremely high degree of parallelism. Second, this model is refined towards an optimal level of parallelism by agglomeration of its nodes and definition of parameters for scalable patterns. 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. Now the implementation can be started; *algorithmic skeletons* are available in an optimal case for efficient implementation of the parallel design patterns.

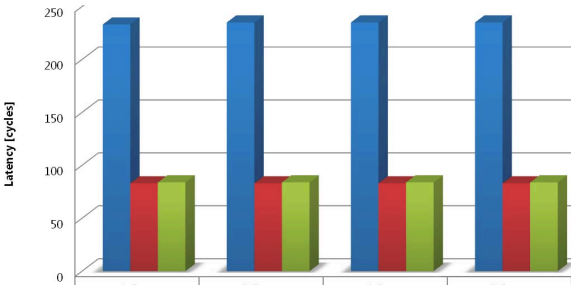
The main advantages of the two well-known parallelization approaches from the HPC domain, on which it is based, are exhibited by the parMERASA approach, too: The clear methodology is derived from the PCAM approach by Foster [9]; the strong use of parallel design patterns is adapted from the approach by Mattson et al. [10], [11], [12].

Low overhead for static WCET analysis will be assured by (a) allowing only analyzable parallel design patterns and (b) an extension of the *Patterns Catalogue* with requirements for static WCET analysis.

Strong tool support is envisioned [6] especially for the adaption of parallelism to the target platform (phase 2); also programming guidelines, decision support for the parallelization and requirements for system software will be developed.

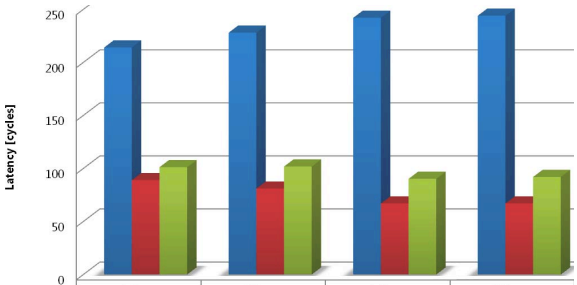
### B. Avionic Applications

In the scope of parMERASA project, the avionic supplier Honeywell International s.r.o. (HON) goal is to parallelize applications running on the following three avionic electronic systems: 1) a 3DPCAS (3D Path Planning/Collision Avoidance System), 2) a SNS (Stereo Navigation System), and 3) a GNSS (Global Navigation Satellite System) receiver. The outcome of HON's efforts are twofold. First, HON's team will be focused in a comparison study planned to validate the parMERASA approach in the avionic domain. Second, HON's team will look at different perspectives of the design cycle for




| | 1 Core | 2 Cores | 4 Cores | 8 Cores |
|---|---|---|---|---|
| Maximum | 233 | 235 | 235 | 235 |
| Minimum | 83 | 83 | 83 | 83 |
| Average | 84 | 84 | 84 | 84 |

Fig. 1: *Read* memory access latencies in Freescale P4080




| | 1 Core | 2 Cores | 4 Cores | 8 Cores |
|---|---|---|---|---|
| Maximum | 214 | 228 | 242 | 244 |
| Minimum | 89 | 81 | 67 | 67 |
| Average | 101 | 102 | 90 | 92 |

Fig. 2: *Write* memory access latencies in Freescale P4080

avionic electronic systems and compare parMERASA IP with COTS products being considered as an alternative.

The first application presents 3DPCAS. The 3DPCAS application algorithm provides a good basis for parallelization, and as such can be easily configured to evaluate the execution time impact of different multi-core architectures. The 3DPCAS 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. An implementation of the 3DPCAS was already analyzed on a Freescale QorIQ P4080 multi-core COTS platform.

The second application presents SNS. The SNS application was deployed to a variety of ground vehicles for navigation in GPS-denied situations. In the presence of GPS signal, it was used as a surrogate for the IMU (Inertial Measurement Unit) to determine vehicle rotation and translation change (rates and velocities). An implementation of the SNS was recently analyzed on a single core architecture with a maximum update rate of 0.6 Hz. A HON goal is to execute SNS algorithms at 10 Hz (approximately 20 times faster than current single-core implementation) by exploiting computation capabilities of a multi-core platform.

The third application presents GNSS receiver. The GNSS application is an in-development product, which is expected to support all currently available global constellations: GPS, GLONASS, Galileo, and Compass. By parallelizing GNSS receiver software, HON expects to reduce development and production costs, and include computationally demanding features such as: ARAIM (Advanced Receiver Autonomous Integrity Monitoring), multi-path mitigation based on multi-correlators, and vector tracking.

In Fig. 1 and Fig. 2, we present the preliminary results of read and write memory access latencies on multi-core COTS platform (Freescale QorIQ P4080) with disabled caches. In both figures, we compare the maximum, minimum, and average memory access latencies with 1, 2, 4, and 8 cores configured to issue either read-only or write-only operations at the same time. In Fig. 1, as the experiments suggest, there is a minor increase in the read operations latency (as expected) corresponding to the increased number of cores concurrently accessing the memory. In Fig. 2, we observe an increase in the maximum write memory access latency (as expected), but the minimum write memory access latency decreases significantly (not intuitive). Furthermore, a write operation results into a load and a store instruction at the micro-architecture level, i.e., the core first loads cache-line (with fixed transaction size) from the memory to a Data Line Fill Buffer (a mini-cache that is part of the load-store unit in each core), updates its contents, and stores the cache-line back to the memory. With the limited knowledge of the micro-architecture, we speculate that the latency decrease in the write memory accesses results from possible snooping optimizations for the load instructions. We speculate that these snooping optimizations also cause write memory accesses to have lower latency than read memory accesses. Regarding the reduction in the average write memory access latency, we conclude that memory controllers feature an open page policy, i.e., open pages are only closed when necessary. Open page policy is widely employed in COTS architectures, because it exploits data locality. However, similarly to caches, it introduces variations, because memory latency dependent on the history of memory accesses.

Based on the conducted experiments, we conclude that significant increase in the range and variations of the memory access latencies indicate that the COTS memory controllers are sensitive to sharing. Furthermore, write-intensive workloads seem to induce more volatile behavior than read-intensive workloads.

### C. Automotive Applications

The automotive supplier DENSO AUTOMOTIVE Deutschland GmbH parallelises an existing diesel engine management system. 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 parallelizing automotive software is the high number of dependences between runnables. For that reason, an application specific parallelization approach is developed, which uses timing properties and data dependences as constraint. The reconstruction as AUTOSAR-VFB model as well as state-of-the-art WCET analysis and measurement tools, developed in the parMERASA project, are used to extract these constraints from single-core code. Runnables are then distributed over cores/clusters and executed in parallel, see section V-H. To improve speed-up even further, runnables with

a high WCET are itself parallelized to execute on multiple cores of one cluster.

Current work focuses on the extraction of timing properties and data dependences from the sequential implementation. Next steps focus on the development of strategies for the distribution of runnables with the target to achieve a maximum degree of parallelism, while the given constrains are maintained. The pattern-based parallelization approach developed by University of Augsburg will be applied to the engine management system and evaluated on the parMERASA simulator.

### D. Construction Machinery Applications

BAUER Maschinen GmbH is the leading manufacturer of specialist foundation equipment, as e.g. drilling rigs, as well as equipment for bored piles, diaphragm walls, anchors, and sheet pile walls. 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 parallelizes the control algorithm for the dynamic compaction machine of BAUER Maschinen GmbH. With this project the advantage (the increase of performance) of parallelization of existing code should be shown, without changing too much the well-known algorithm. In steps 1 and 2 the control loop will be parallelized such that each supervised sub-task (PWM, CAN connected, I/O) will be potentially mapped on a different core, following a master (control loop)/slave based parallel design pattern. Agglomeration and mappings will be performed to find out which sub-tasks to merge and map to different configurations, down to the dual-cores and quad-cores that are already available.

### E. WCET Analyzability and Verification Tools for Embedded Multi-cores by Rapita Systems Ltd.

Rapita Systems Ltd. based in York, UK, is a specialist in worst-case execution time analysis and simulation of real-time embedded systems for the automotive, avionic, space and telecommunication markets. RapiTime Systems Ltd. brings into the project its RapiTime tool for measurement-based WCET analysis. The company investigates verification tools for safety critical systems using multi-cores.

Specifications for verification and profiling tools were developed based on the tool requirements of the project partners. Five verification and parallelization support tools were selected by Rapita Systems Ltd. based on the application companies' requirements. The identified tools are: (1) extension of RapiTime to provide on-target timing and WCET analysis tool for parallel programs; (2) extension of RapiCover to provide on-target code coverage tool for parallel programs; (3) memory, cache and stack analysis tool for parallel programs; (4) tool to assist with the parallelization of existing sequential software; (5) visualization and profiling tool for parallel programs. It was decided based on the requirements of the application companies that tools (4) and (5) are the most urgently required ones in phase 2 of the project. Tools (1), (4) and (5) are already being implemented. The remaining tools are currently in the detailed specification phase.

### F. Static WCET analysis

Université Paul Sabatier, located in Toulouse, France, hosts in its Computer Science Institute (IRIT) the TRACES (Toulouse Research group on Architecture & Compilation for Embedded Systems) research team, which has significant experience in static WCET analysis. The TRACES team investigates automatic static WCET analysis of parallel applications. It will adapt its static WCET analysis tool OTAWA[7] by an automatic static WCET analysis of parallel applications and support features added in the parMERASA multi-core. The current OTAWA tool supports several instruction sets and micro-architectures and includes the TriCore instruction set and the MERASA multi-core architectural description.

Timing analyzability of parallel design patterns was investigated and synchronization primitives to implement the patterns were devised [1]. A method to compute worst-case waiting times in combination with an annotation scheme of the parallel source code was proposed.

New techniques will be developed to support hardware schemes involved in the design of timing predictable multi-cores with 16 to 64 cores (multi-level and distributed memory hierarchy as well as interconnection network).

The standard process of WCET analysis, designed to support sequential tasks, will also be extended to support parallel applications. Main issues are the identification of the application structure and the determination of waiting times on synchronizations and communications. Foreseen solutions will combine code analysis techniques and user-/compiler-defined annotations. These solutions will be completed with guidelines to timing predictable parallel applications.

Up to now WCET analysis tool OTAWA was enhanced by the PowerPC ISA, which was selected as core ISA of the parMERASA multi-core processor, the specification of the annotation format was refined, and a further analysis of synchronization primitives is in progress.

### G. System Architecture

One of the main responsibilities of the parMERASA system architecture is to ensure that the timing assumptions done at the application level are maintained. This is achieved by the provision of time and space partitioning between application partitions among each other. Fig. 3 shows the general system architecture approach proposed for the parMERASA project by University of Augsburg. The RTOS Kernel Library represents the common basis for domain specific runtime environment (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 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. This guarantees that malfunctioning, non-critical services cannot affect the whole system. A domain

[7] http://www.otawa.fr/

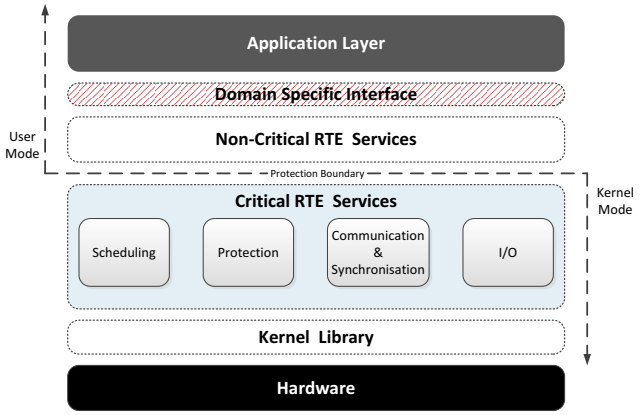



Fig. 3: parMERASA System Architecture

specific interface connects the application layer with the RTE services.

The overall system architecture [13] was defined to support the execution of parallelized hard real-time applications and their WCET analyzability. Based on the requirements of the different, industrial application domains, the parMERASA system software will introduce specific RTEs called TinyAUTOSAR and TinyIMA. The AUTOSAR standard[8] respectively ARINC 653 specification[9] serve as basis for these RTEs. Additionally, a construction machinery RTE will be provided. Each RTE implementation will incorporate the RTOS Kernel Library as a common basis. In all cases, extensions to support advanced parMERASA mechanisms of synchronization, inter-core communication, and parallel design patterns will be implemented.

The parMERASA system software, i.e. RTOS Kernel Library as common platform basis, and individual "tiny" RTEs for each application domain are under development to assist the applications. Currently, the library is widely implemented and the TinyAUTOSAR implementation is under construction. It will follow the distributed approach of fos OS [14] respectively MOSSCA [15] proposal for parallel RTOSs.

### *H. Embedded Multi-core Processors, System Architecture and System Software*

Safety critical real-time systems 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. Currently, incremental qualification is guaranteed by providing robust space and time partitioning to applications, i.e. the functional and timing behavior of each application is not affected by other applications, so formal certification can be provided. Therefore, the parMERASA target applications 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 in which

[8]http://www.autosar.org/
[9]http://www.arinc.com/

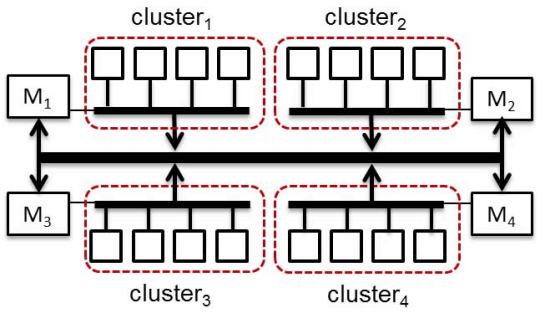



Fig. 4: A clustered architecture deploying a two-level hierarchical bus

applications are spawned in multiple parallel threads, i.e. parallel threads belonging to one application cannot affect the functional and timing behavior of parallel threads belonging to other applications running simultaneously within the parMERASA architecture.

Therefore, the parMERASA architecture relies on a clustered processor architecture, in which cores are organized in *clusters* connected through a dedicated NoC. At the same time, each cluster is connected with another NoC that allows cores from different clusters to communicate among them. Such type of processor architectures are nowadays a reality in the embedded system domain; examples include the STM P2012 [16] and the Kalray MPPA 256 [17].

The use of clustered architectures provides to applications isolated islands of communications, in which they can run. Clusters prevent communication requests from different applications to interfere among them as only the NoC that connect cores is used (intra-cluster communication). Only in case applications want to communicate among each other (inter-cluster communication), a potential interference may occur in the NoC that connects the different clusters. However, in most safety-critical real-time systems, such communication is known statically at system integration (e.g. inter-partition communication in the avionic domain and port communication in the automotive domain), which allows scheduling the inter-cluster communication in a predictable way.

It is important to remark that functional isolation is also guaranteed by controlling inter-cluster communication: requests from an application cannot access data from other applications unless inter-cluster communication is used. To that end, the processor architecture must provide a complete memory management unit that will allow the operating system to define different memory regions within each cluster: data local to each core, data global to each cluster and data global to all clusters.

Fig. 4 shows an example of a clustered architecture deploying a two-level hierarchical bus. The application that executes within cluster 1 will reside in memory 1 and so intra-cluster requests will only use its corresponding bus, without interference with other clusters. In case an application wants to communicate with other applications, the requests will traverse the second bus level without affecting other clusters, and the message will be directly routed to the corresponding cluster in which the destination application runs.

Moreover, the parMERASA architecture relies on caches to reduce latencies and impact of interferences when multiple cores from the same cluster want to access the memory. In this case, because multiple cores can share data, it is mandatory to guarantee coherent data accesses. To that end, a predictable cache coherence mechanism has been proposed: the On-demand Coherent Cache ($ODC^2$) [18]. $ODC^2$ guarantees coherent accesses only to data that is accessed inside critical regions. 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.

## VI. Conclusion

The parMERASA project targets parallelizing hard real-time programs to run on predictable multi-/many-core processors. The first project phase (Oct. 2011 - June 2012) "Requirement Specification and Concept" has been successfully completed. The project is currently in the middle of the second project phase (July 2012 - September 2013) "Full Specification and Implementation".

From application side, five use cases were selected for parallelization: for avionics (Honeywell International s.r.o.) 3D Path Planning, Stereo Navigation, and Global Navigation Satellite System; for automotive (DENSO AUTOMOTIVE Deutschland GmbH) a diesel engine management system; and for construction machinery (BAUER Maschinen GmbH) the control algorithm for a dynamic compaction machine.

A model-based parallelization approach for timing analyzable parallel software was developed making strong use of parallel design patterns, which were selected together with the application companies. For the static WCET analysis tool OTAWA, source code annotations for parallelization analysis were defined that are based on the parallel design patterns. Requirements on verification tools were defined and five verification and profiling tools were selected, which will be implemented based on Rapita Systems' RVS tools.

The overall system architecture, a common RTOS Kernel Library and the required functionalities for TinyAUTOSAR, TinyIMA, and construction machinery RTEs are defined. The RTOS Kernel Library and TinyAUTOSAR functions are partly implemented. Multi-core architecture design space exploration was done based on the requirements of the applications, system architecture and WCET analysis.

The experiences of parMERASA will be provided as recommendations to the execution models of AUTOSAR and IMA.

## Acknowledgment

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

## References

[1] C. Ballabriga, H. Cassé, C. Rochange, and P. Sainrat, "OTAWA: An Open Toolbox for Adaptive WCET Analysis," in *Software Technologies for Embedded and Ubiquitous Systems*, ser. Lecture Notes in Computer Science, S. Min, R. Pettit, P. Puschner, and T. Ungerer, Eds. Springer Berlin Heidelberg, 2011, vol. 6399, pp. 35–46.

[2] G. Edelin, "Embedded Systems at THALES: the Artemis challenges for an industrial group," Presentation at ARTIST Summer School in Europe, Sep. 2009.

[3] "IKT 2020. Forschung für Innovationen," Mar. 2007. [Online]. Available: www.bmbf.de/pub/ikt2020.pdf

[4] "Euro 5 and Euro 6 standards: Reduction of pollutant emissions from light vehicles," Sep. 2011. [Online]. Available: europa.eu/legislation_summaries/environment/air_pollution/l28186_en.htm

[5] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, "The Worst-Case Execution-Time Problem—Overview of Methods and Survey of Tools," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, pp. 36:1–36:53, May 2008.

[6] R. Jahr, M. Gerdes, and T. Ungerer, "On efficient and effective model-based parallelization of hard real-time applications," in *Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme IX*. Schloss Dagstuhl: fortiss GmbH, München, April 2013, pp. 50–59.

[7] ——, "A Pattern-supported Parallelization Approach," in *Proceedings of the 2013 International Workshop on Programming Models and Applications for Multicores and Manycores*, ser. PMAM '13. New York, NY, USA: ACM, Feb. 2013, pp. 53–62.

[8] M. Gerdes, F. Kluge, T. Ungerer, C. Rochange, and P. Sainrat, "Time Analysable Synchronisation Techniques for Parallelised Hard Real-Time Applications," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, March 2012, pp. 671–676.

[9] I. Foster, *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[10] B. L. Massingill, T. G. Mattson, and B. A. Sanders, "Patterns for Parallel Application Programs," 1999.

[11] ——, "More Patterns for Parallel Application Programs," in *Proceedings of the eighth Pattern Languages of Programs Workshop (PLoP 2001)*, September 2001.

[12] T. Mattson, B. Sanders, and B. Massingill, *Patterns for parallel programming*, 1st ed. Addison-Wesley Professional, 2004.

[13] C. Bradatsch and F. Kluge, "parMERASA Multi-core RTOS Kernel," University of Augsburg, Tech. Rep. no. 2013-02, Feb. 2013.

[14] D. Wentzlaff and A. Agarwal, "Factored operating systems (fos): the case for a scalable operating system for multicores," *SIGOPS Oper. Syst. Rev.*, vol. 43, no. 2, pp. 76–85, Apr. 2009.

[15] F. Kluge, B. Triquet, C. Rochange, and T. Ungerer, "Operating Systems for Manycore Processors from the Perspective of Safety-Critical Systems," in *Proceedings of 8th annual workshop on Operating Systems for Embedded Real-Time applications (OSPERT 2012)*, Pisa, Italy, Jul. 2012.

[16] L. Benini, E. Flamand, D. Fuin, and D. Melpignano, "P2012: Building an ecosystem for a scalable, modular and high-efficiency embedded computing accelerator," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, march 2012, pp. 983 –987.

[17] "Kalray MPPA 256 processor," 2013. [Online]. Available: http://www.kalray.eu/products/mppa-manycore/mppa-256/

[18] A. Pyka, M. Rohde, and S. Uhrig, "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, Berlin, Germany*, Jan. 2013.