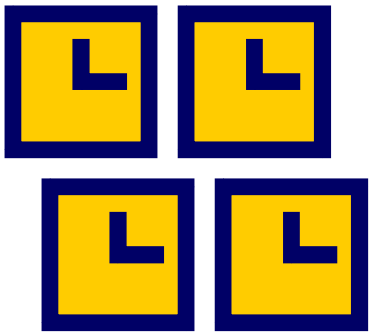

P-SOCRATES

Parallel Software Framework for Time-Critical many-core Systems

PROJECT REPORT - SUMMARY

Grant Agreement number: 611016

Project acronym: P-SOCRATES

Project title: Parallel Software Framework for Time-Critical many-core Systems

Funding Scheme: CP-FP-INFSO

Period: from October 2013 to December 2016

Name of the scientific representative of the project's co-ordinator, Title and Organisation:

Prof. Luís Miguel Pinho
Instituto Superior de Engenharia do Porto (ISEP)

Tel: +351 22 8340502

Fax: +351 22 8321159

E-mail: lmp@isep.ipp.pt

Project website address: http://p-socrates.github.io

Table of Contents

EXECUTIVE SUMMARY **3**

1 PROJECT CONTEXT AND OBJECTIVES **4**

2 MAIN S&T RESULTS **8**

2.1 COMPILER ANALYSIS OF PARALLEL PROGRAMS 8
2.2 PREDICTABLE SCHEDULING OF PARALLEL TASKS ON MANY-CORE SYSTEMS 9
2.3 METHODOLOGY FOR MEASUREMENT-BASED TIMING ANALYSIS 9
2.4 OPTIMIZED OPENMP TASKING RUNTIME SYSTEM 10
2.5 REAL-TIME OPERATING SYSTEMS 11
2.6 ARCHITECTURAL EVOLUTIONS FOR IMPROVED PROGRAMMABILITY AND PREDICTABILITY 12
2.7 THE UPSCALE SDK 12

3 MAIN PUBLICATIONS **14**

Executive summary

Nowadays, the prevalence of electronic and computing systems in our lives is so ubiquitous that it would not be far-fetched to state that we live in a cyber-physical world dominated by computer systems. Examples include pacemakers implanted within the human body, cars and airplanes transporting us, smart grids and traffic management. All these systems demand more and more computational performance to process large amounts of data from multiple sources, some of them with processing times requirements. In other words, systems are required to deliver performance within pre-defined (and sometimes extremely short) time bounds. This timing aspect is vital for systems like planes, cars, business monitoring, e-trading, etc.

As a result of these requirements, the computer electronic devices to which these systems depend on are constantly required to become more and more powerful and reliable, while remaining affordable. To cope with such performance requirements, chip designers have recently started producing chips containing multiple processing units, the so called multi-core processors, effectively integrating multiple computers within a single chip. This radical shift in the chip design paved the way for parallel computing: rather than processing the data sequentially, the cooperation of multiple processing elements within the same chip allowed systems to be executed concurrently, in parallel.

Unfortunately, the parallelization of the computing activities brought upfront many challenges, because it affects the timing behaviour of systems, as well as the entire way people think and the design computers, from the design of the hardware architecture, through the operating system up to the end-user application. Although multi-core processors are promising candidates to improve the responsiveness of these systems, the interactions that the different computing elements may have within the chip, can seriously affect the performance opportunities brought by parallel execution. Moreover, providing timing guarantees becomes harder, because the timing behaviour of the system running within a multi-core processor depends on interactions that are most of the time not know by the system designer. This makes system analysts to be struggled trying to provide timing guarantees for such platforms. Finally, most of the optimization mechanisms buried deep inside the chip are geared only to increase performance and execution speed rather than providing predictable time behaviour.

The aim of P-SOCRATES was thus to allow applications with high-performance and real-time requirements to fully exploit the huge performance opportunities brought by the most advanced many-core processors, whilst ensuring a predictable performance and maintaining (or even reducing) development costs of applications. This was achieved by developing a new methodology, and a set of tools, integrated in the UpScale SDK. Industrial companies benefit from the project outcomes, allowing European technology suppliers to properly exploit the capabilities of next-generation hardware platforms in a predictable way. Impacts are foreseen in the development of technologies for both the high-performance and embedded computing domains.

From an applicative point of view, P-SOCRATES represents a reference point for the implementation of real-time complex event-processing systems, and, more in general, of workload-intensive applications with time-criticality requirements, enabling a more efficient smart society. The computing technology developed in the project was evaluated on real-world use-cases, such as a complex event processing engine, an embedded sensor processing system and an online semantic intelligence tool. The technology also allows a deeper understanding of many-core off-the-shelf systems, enabling new kinds of applications to be developed on top of these platforms.

1 Project context and objectives

High performance computing has been for a long time the realm of a specific community within academia and specialised industries; in particular those targeting demanding analytics and simulations applications that require processing massive amounts of data. In a similar way, embedded computing has also focused mainly on specific systems with specialised and fixed functionalities and for which timing requirements were considered as much more important than performance requirements. However, with the ever-increasing availability of more powerful processing platforms, alongside affordable and scalable software solutions, both high-performance and embedded computing are extending to other sectors and application domains.

The demand for increased computational performance is currently widespread and is even more challenging when large amounts of data need to be processed, from multiple data sources, with guaranteed processing response times. Although many systems focus on performance and handling large volumes of streaming data (with throughput and latency requirements), many application domains require real-time behaviour and challenge the computing capability of current technology. Some examples are:

- In cyber-physical systems, ranging from automotive and aircrafts, to smart grids and traffic management, computing systems are embedded in a physical environment and their behaviour obeys technical rules dictated by this environment. Typically, they have to cope with the timing requirements imposed by the embedding domain. In the Large Hadron Collider (LHC) in CERN, beam collisions occur every 25ns, which produces up to 40 million events per second. All these events are pipelined with the objective of distinguishing between interesting and non-interesting events to reduce the number of events to be processed to a few hundreds. Similarly, bridges are monitored in real-time with information collected from more than 10,000 sensors processed every 8ms, managing responses to natural disasters, maintaining bridge structure, and estimating the extent of structural fatigue. Another interesting application is in intelligent transportation systems, where systems are developed to allow for fuel consumption reduction of railway systems, managing throttle positions, elaborating big amounts of data and sensor information, such as train horsepower, weight, prevailing wind, weather, traffic, etc.
- In the banking/financial markets computing systems process large amounts of real-time stock information in order to detect time-dependent patterns, automatically triggering operations in a very specific and tight timeframe when some pre-defined patterns occur. Automated algorithmic trading programs now buy and sell millions of dollars of shares time-sliced into orders separated by 1ms. Reducing the latency by 1 ms can be worth up to $100 million a year to a leading trading house. The aim is to cut microseconds off the latency in which these systems can reach to momentary variations in share prices.
- In industry, computing systems monitor business processes based on the capability to understand and process real-time sensor data from the factory-floor and throughout the whole value chain, with Radio Frequency Identification (RFID) components in order to optimise both the production and logistics processes.

The underlying commonality of the systems described above is that they are time-critical (whether business-critical or mission-critical, it is necessary to fulfil specific timing requirements) and with high-performance requirements. In other words, for such systems, the correctness of the result is dependent on both performance and timing requirements, and the failure to meet those is critical to the functioning of the system. In this context, it is essential to guarantee the timing predictability of the performed computations,

meaning that arguments and analysis are needed to be able to make arguments of correctness, e.g. performing the required computations within well-specified bounds.

Until now, trends in high-performance and embedded computing domains have been running in opposite directions. On one side, high-performance systems (HPC) are traditionally designed to make the common case as fast as possible, without concerning themselves for the timing behaviour (in terms of execution time) of the not-so-often-cases. As a result, the techniques developed for HPC are based on complex hardware and software structures that make any reliable timing bound almost impossible to derive. On the other side, real-time embedded systems are typically designed to provide energy-efficient and predictable solutions, without heavy performance requirements. Instead of fast response times, they aim at having deterministically bounded response times, in order to guarantee that deadlines are met. For this reason, these systems are typically based on simple hardware architectures, using fixed-function hardware accelerators that are strongly coupled with the application domain.

The needs for energy-efficiency and for flexibility, coming along with Moore's law greedy demand for performance and the advancements in the semiconductor technology, have progressively paved the way for the introduction of "many-core" systems, i.e., computing chips containing a high number of cores (tens to hundreds) in both domains. The introduction of many-core systems has set up an interesting trend wherein both the HPC and the real-time embedded domain converge towards similar objectives and requirements. Many-core computing fabrics are being integrated together with general-purpose multi-core processors to provide a heterogeneous architectural harness that eases the integration of previously hardwired accelerators into more flexible software solutions.

Many-core processor architectures allow these performance requirements to be achieved, by integrating dozens or hundreds of cores, interconnected with complex networks on chip, paving the way for parallel computing. Unfortunately, parallelization brings many challenges, by drastically affecting the system's timing behavior: providing guarantees becomes harder, because the behavior of the system running on a multi-core processor depends on interactions that are usually not known by the system designer. This causes system analysts to struggle to provide timing guarantees for such platforms.

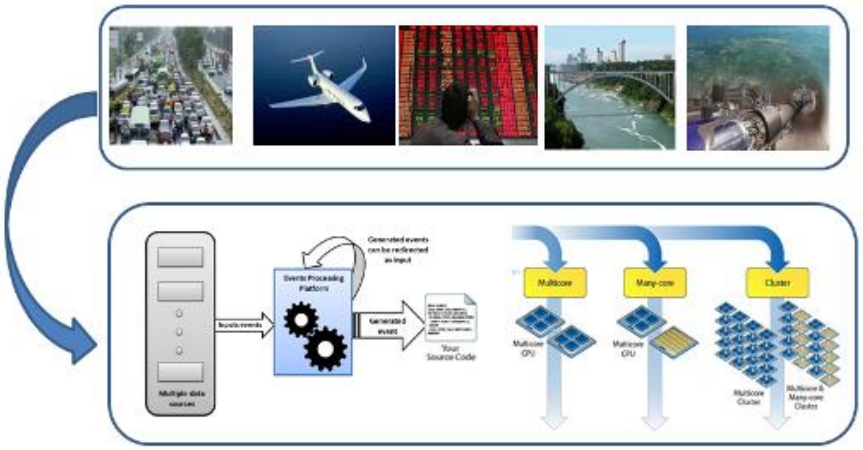

Figure 1 – P-SOCRATES Global perspective

The aim of P-SOCRATES was thus to allow the current and future applications with high-performance and real-time requirements to fully exploit the huge performance opportunities brought by the most advanced Commercial Off-the-Shelf (COTS) many-core embedded processors, whilst ensuring a predictable performance and maintaining (or even reducing) development costs of applications (Figure 1).

To do so, P-SOCRATES focused on combining techniques from different domains: the newest high-performance software techniques for exploiting task parallelism, the most advanced mapping and scheduling methodologies and timing and schedulability analysis techniques used in real-time embedded systems, and the low-energy many-core platforms of the embedded domain. This allowed taking important steps towards the convergence of high-performance computing (HPC) and real-time and embedded domains (Figure 2), providing predictable performance to HPC systems and increasing performance of real-time embedded systems.

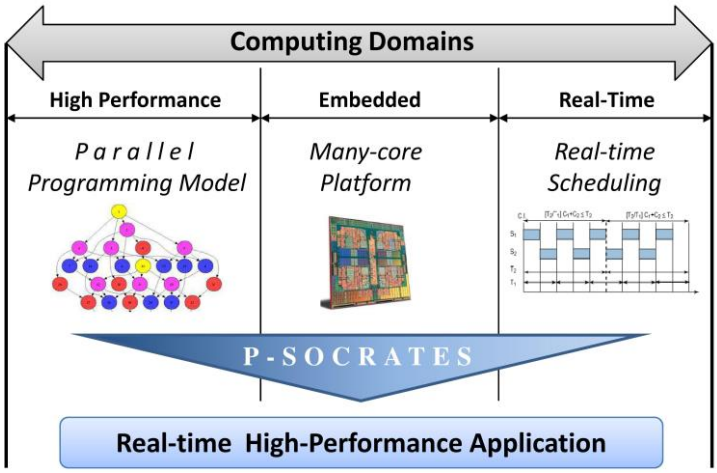

Figure 2 – P-SOCRATES combines high-performance parallel programming models, high-end embedded many-core platforms and real-time systems technology.

The work in the project was guided having in mind a set of strategic goals. First, the *time-criticality* requirements of applications from both high-performance computing and real-time embedded domains are to be guaranteed by ensuring *time predictability* through the whole design stack. Such guarantees allow to provide high-performance capabilities within analytically-derived response-time bounds. A second goal was to provide parallelisation on top of COTS many-core high-end embedded processors, still fulfilling time-critical requirements. By combining *parallel programming models* of high-performance computing with the real-time technology of embedded systems, the *parallel execution* of workload-intensive applications with critical timing requirements is applied on *Off-The-Shelf many-cores embedded processors*. Finally, the whole technology developed within the project was designed towards facilitating the production of efficient high-performance computing and embedded systems with requirements in performance and time predictability. In particular, reducing the complexity of parallel programming in embedded applications was of paramount importance.

Considering these goals, P-SOCRATES had the following detailed objectives:

- **Develop a parallel programming model capable of expressing data dependencies and real-time application properties.**

  Parallel programming models currently used in HPC is extended to contain relevant information about the impact of executing simultaneous parallel task. Such information is contained in an extended task dependency graph automatically generated by the compiler. The extended task dependency graph allows decoupling the application parallelism from the actual mapping of execution in the underlying platform, in order to facilitate the porting of applications to multiple many-core platforms.

- **Develop resource-aware mapping and scheduling algorithms that are able to predictably schedule multiple resources in the system (CPU, interconnect, memory).**

  These algorithms consider the information on parallelism, data-dependencies, and timing properties in the programming model, in order to allocate parallel tasks to cores and for scheduling their execution and access to interconnection networks, memories and other resources.

- **Develop a timing analysis methodology capable of expressing the timing implications of data-dependencies and potential resource-contentions of task-to-core mappings.**

  Analyse the different interfering sources (higher priority workload, network contention, memory bottlenecks, etc.) that might affect the execution of a task, providing upper bounds on the response times of each task. This information influences the selection of the scheduling algorithms to allow for predictable systems, lending themselves to a tighter schedulability analysis and a simplified computation of worst-case timing parameters. This timing analysis considers the extended task dependency graph provided by the compiler and the hardware platform.

- **Identify specific hardware recommendations to improve the predictability of current COTS many-core embedded processor designs.**

  Current many-core embedded processors are designed to provide high throughputs for multiple types of workloads, as well as to accelerate computationally intensive algorithms for various application domains. The project provides a set of recommendations to improve platform predictability and decrease timing bounds without affecting the average-case behaviour of the platform.

- **Integrate the proposed techniques on real world use-case applications, increasing performance, while providing trustworthy real-time bounds, on a COTS many-core embedded platform.**

  One of the main targets of the project was to prove that it is possible to implement real-time HPC systems on next-generation many-core embedded platforms, without sacrificing performance. For this purpose, the implementation of real-work applications characterised by a heavy workload with real-time requirements was provided on a COTS many-core platform, providing analytical bounds on response-time parameters.

- **Contribute to Open Source software.**

  The developed software framework was made publicly available under Open Source licenses at the end of the P-SOCRATES project.

2 Main S&T results

P-SOCRATES developed a complete and coherent software system stack, able to bridge the gap between the application design with both high-performance and real-time requirements, and the hardware platform, in our case a many-core embedded processor. The project provided a *new framework* to combine real-time embedded mapping and scheduling techniques with high-performance parallel programming models and associated tools, able to express parallelisation of applications. The programming model used was based on the state-of-the-art OpenMP specification.

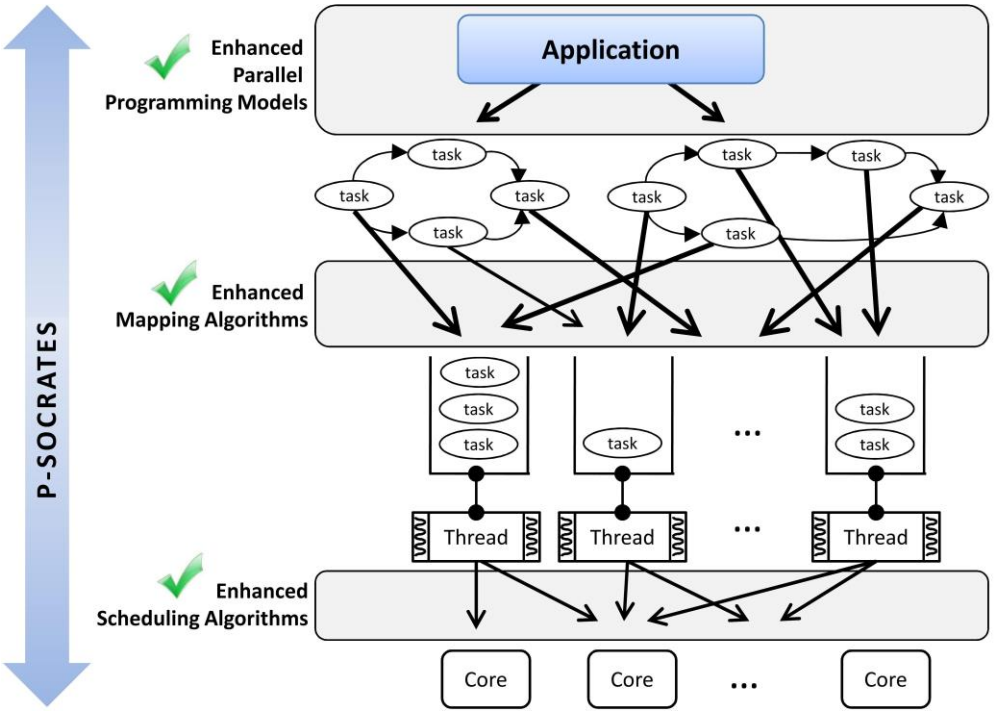

Figure 3 – Vertical stack of application decomposition

The software stack (shown in Figure 3) is able to extract a task dependency graph from the application, statically or dynamically mapping these tasks to the threads of the operating system, which then dynamically schedules them on the many-core platform. The main scientific and technology advances provided are described in the following subsections.

2.1 Compiler Analysis of Parallel Programs

P-SOCRATES developed a compilation framework with a twofold objective: (1) To extract all information needed to derive timing and schedulability analysis of OpenMP programs, and (2) facilitate the programmability of the many-core architecture, by supporting the OpenMP acceleration model. Next we briefly summarise both achievements.

The P-SOCRATES compilation framework, based on the source-to-source compiler Mercurium developed at BSC, incorporates a novel compiler analysis phase capable of extracting all control-flow and data-flow information needed to analyse the timing behaviour of OpenMP programs. This analysis phase is composed of two stages: In the *first stage,* the *parallel control flow graph* (PFCG) is extracted, the induction variables involved in loops and conditionals containing OpenMP tasks are identified, and a complete range analysis is

performed to determine the value of each program variable impacting on the parallel structure of the program. In the *second stage,* the control flow structures are resolved and expanded, identifying those OpenMP tasks that are instantiated and the data dependencies existing among them. This analysis generates a *direct acyclic graph* (DAG) or *task dependency graph* (TDG), which incorporates all possible OpenMP tasks and run after dependencies existing among them, generated at deployment time.

The P-SOCRATES compiler also incorporates a *source-code lowering phase,* that transforms a subset of OpenMP directives coded into the application (those implementing the tasking and acceleration model), to the corresponding API implemented in P-SOCRATES run-time libraries, i.e. *libpsocoffload*, which implements the offloading mechanisms between IO cores and computing clusters, and *libpsocomp,* which implements the OpenMP tasking model.

Overall, the P-SOCRATES compilation process is completely automatic and transparent to the developer, so the compilation framework invokes the corresponding compiler to generate the binaries executing on the IO cores and computing clusters, linking all together in a unique binary.

Finally, P-SOCRATES investigated the compiler support needed to guarantee functional safety of OpenMP programs, identifying the techniques needed to guarantee that the parallel execution is free of errors such as race-conditions or dead-locks. This is a fundamental step to incorporate parallel programming into safety-critical real-time systems.

2.2 Predictable Scheduling of Parallel Tasks on Many-core Systems

The P-SOCRATES project provided substantial advances in the real-time scheduling and schedulability analysis of parallel graphs. In a first phase, the work focused on providing an approach based on mapping parallel tasks to threads and these to cores, with no migration support. Subsequently, P-SOCRATES analysed possible strategies to schedule parallel tasks considering thread migration among processors.

As a result, and as set forth in the project goals, P-SOCRATES developed an overall schedulability analysis of parallel real-time systems. The analysis is based on the computation of the worst-case response time of real-time tasks concurrently executing on a given cluster of cores. The results of the project provided novel techniques and analysis for two different approaches, depending on the mapping/scheduling mechanisms supported by the framework: (i) a dynamic solution based on a global scheduler allowing a work-conserving behaviour, and (ii) a fully static solution based on a partitioned scheduler and a fixed task-to-thread mapping. The analysis is complemented with a software module to be used by the mapping algorithms to verify the response times of particular mappings, integrated in the UpScale SDK.

An important aspect to note is the research performed by the project, where for the first time a parallel task model used for real-time schedulability analysis has been enriched with control-flow information, thus effectively taking into consideration the problem of conditional branches potentially creating different sub-graphs. This led to the new real-time model of conditional parallel graphs, advancing the state of the art in the domain. In parallel, extensive work was performed on the schedulability analysis of the OpenMP tasking execution models (tied and untied), showing how to provide predictability guarantees for applications parallelised with these models.

2.3 Methodology for measurement-based timing analysis

In the first phase pf the project, the partners have deeply investigated and summarized the currently-available methods to timing analysis. After weighing the pros and cons of each technique, the project

concluded that a new technique based on runtime measurements would suit best the project objectives in terms of flexibility of the proposed solution, its development effort and cost and its adaptation to new hardware platforms.

The project thus focused on the design and development of that new measurement-based technique. The methodology includes the following main steps: configuration and compilation of the target application on the platform; configuration of the execution conditions in which the target application is to be timed; execution and timing of the application; collection and post-processing of the timed traces; extraction of the worst-case timing estimates; report generation and feedback to the designer. The methodology is based on executing all those steps several times using different execution conditions. It then reports on the variability in the timing behaviour of the application under these conditions and thus on its sensitivity to the external events that may influence those conditions.

The timing tool has been fully integrated with the compiler, the mapper, and the schedulability analysis tool designed in the project to create the UpScale SDK. This single higher-level and fully-automated process is not only able to analyse the timing behaviour of the target application using the SDK, but also it is able to setup the scheduling algorithm and mapping tool so that the system is automatically configured to satisfy all the application timing requirements.

One of the project outcome is a thus fully operational, implemented, and tested methodology and tool to capture the timing properties of the target application and derive estimations of its maximum execution requirement.

2.4 Optimized OpenMP Tasking Runtime System

The P-SOCRATES methodology entirely relies on the parallel computing abstraction provided by the OpenMP tasking model, and its conceptual similarities to the DAG model, to achieve predictable task scheduling.

Task-based parallelism has historically provided a powerful conceptual framework to exploit massive and irregular parallelism in target applications from the high-performance computing (HPC) domain. However, a space- and performance-efficient design of a tasking run-time environment (RTE) targeting a many-core system-on-chip is a challenging task, as embedded parallel applications typically exhibit very fine-grained parallelism. The applicability of the tasking approach to embedded applications and embedded many-core accelerators is often limited to coarse-grained parallel tasks, capable of tolerating the high overheads typically implied in a tasking RTE.

Before P-SOCRATES, existing tasking RTEs for embedded many-cores could only accelerate fine-grained parallelism for very simple patterns (all the parallel tasks are created from the same root node), as they all support only tied tasks. The consequences of this limitation are numerous.

First, as tied tasks cannot migrate between threads, it is impossible to implement work-conserving schedulers, which severely constrains the schedulability analysis. Second, tied tasks imply significant speedup reduction compared to untied tasks when more realistic parallel execution patterns are considered (e.g., recursion).

Third, tied tasks limit the available scheduling policies. In particular, work-first scheduling (WFS) – which is usually preferred for a better cache behaviour than others – leads to fully sequential execution in presence of tied tasks.

P-SOCRATES succeeded in developing an OpenMP tasking RTE that fully supports untied tasks with very low time and space overheads. This has been achieved by means of tightly integrated operation with the operating system APIs. While the official project SDK (which has been made available through the UpScale initiative) targets the Kalray MPPA 256 SoC, we have demonstrated that its adoption onto a different heterogeneous many-core only requires small additional effort.

2.5 Real-time Operating Systems

P-SOCRATES has successfully tackled the challenge of creating an efficient Real-Time Operating System (RTOS) for many-core architectures. The ERIKA Enterprise RTOS provided by partner Evidence has gone under a complete re-design and re-development of the internal data structures and run-time mechanisms aiming at an efficient execution on this kind of platforms. The new version of the RTOS (informally called "ERIKA3") allows to share a single binary kernel image across several cores of the platform, reducing the overall memory consumption. It is worth to underline that, unlike most existing efforts for creating a RTOS for many-core architectures born in academia or research centres (e.g., Akaros, Barrelfish, Corey, etc.), P-SOCRATES has chosen to build an industrial-grade product using a RTOS already certified and used by well-known companies operating in the automotive (e.g., Magneti Marelli, Vodafone automotive, Aprilia) and household appliances (e.g., Honeywell) markets.

The validity of the approach as well as the correctness of the implementation has been proved in the course of the project through the porting on the reference Kalray MPPA platform and the execution of very different use-cases. At the same time, some basic benchmarks have allowed to prove a performance boost against the state-of-the-art of the operating system on this platform (i.e., NodeOS). The huge work and the excellent results achieved in the P-SOCRATES project have naturally triggered the interest of the many-core chip manufacturer (i.e., Kalray).

The preliminary release of the new RTOS version has been made freely available within the UpScale SDK. Given the interest of the industry (e.g., Magneti Marelli) for such a multi-core industrial-grade RTOS, the new version of the RTOS will be ported on several other platforms (e.g., Infineon Tricore, ARM Cortex-A and Cortex-R) in the course of 2017.

The P-SOCRATES project has also investigated the concurrent execution of different operating systems (i.e., "Multi-OS" approach) by mixing the Linux operating system and the ERIKA Enterprise RTOS on the same reference MPPA platform. This activity has allowed the consortium to investigate issues, challenges and techniques, especially related to CPU/memory scheduling, real-time performance and communication between different operating systems. This activity is in turn paving the way to a paradigm shift where multi-criticality and security issues get solved at the low operating system level. A part of the consortium is therefore already investigating potential approaches and techniques to mix these operating systems on ARM-based architectures. Some of these techniques will be shown at the Embedded World exhibition in Nuremberg on March 2017, where the new version of ERIKA Enterprise will be officially announced as well.

We can conclude that, concerning the operating system, the P-SOCRATES consortium has successfully met, and even overtaken, its initial goals. The work started by the project will continue to evolve in the next years through the new version of the ERIKA Enterprise RTOS and the exploration of further approaches and techniques for mixing different operating systems. Last, but not least, the project has contributed to strengthening the European position on research and development of embedded real-time operating systems.

2.6 Architectural evolutions for Improved Programmability and Predictability

During the development of the P-SOCRATES framework we were able to identify a number of features of the Kalray MPPA hardware that potentially constitute a bottleneck to the programmability, the performance or the predictability of the deployed applications. Among such features, we especially focused on a few major ones.

First, the lack of shared-memory-based communication between host and accelerator. As it is widely acknowledged that memory management is one of the key difficulties in heterogeneous systems programming, virtually all heterogeneous systems vendors are increasingly tackling this issue by implementing unified virtual memory (UVM): main DRAM is physically shared between host and accelerator processor, and memory sharing is simplified at the program level by abstracting virtual-to-physical address translation through a IOMMU.

Second, the use of non-coherent L1 caches in the accelerator fabric, which not only puts on the programmer the burden of managing data consistency but also implies significant performance overheads. The use of a cache, in general, might also hamper predictability;

The P-SOCRATES project has investigated the exploration of extensions or evolutions of the Kalray hardware and current software platform to improve performance, scalability, timing predictability and application deployment, beyond what is possible today.

To this aim, a new emulation platform has been developed, based on the integration of a commodity Xilinx Zynq platform, providing an ARM host system plus (limited) FPGA resources, and a large Kintex 7 UltraScale FPGA platform. The FPGA has been used to explore architectural alternatives to what is provided by the target platform of the PSOCRATES project, the Kalray MPPA 256 and to develop associated SW strategies in a manner which is very close to real HW.

The practical outcome of this exploratory task has been a set of HW/SW techniques (generalized as "recommendations") to improve the performance and predictability of the baseline platform.

2.7 The Upscale SDK

P-SOCRATES thus provides a complete and coherent software framework for applications with high-performance and real-time requirements in COTS many-cores embedded processors. This software framework is publicly released and a community is being created under the brand of the UpScale SDK (Software Development Kit). The UpScale SDK includes the tools to manage the application compilation process, its timing analysis and its execution (Figure 4):

- *Compiler flow*. This flow has a twofold objective: (i) to guide the process to generate the binary that will execute on the many-core architecture and (ii) to generate the application Direct Acyclic Graph (DAG) used for the timing analysis and run-time components.
- *Analysis flow*. This flow is in charge of deriving timing guarantees of the parallel execution considering execution time traces of the application running on the many-core platform and incorporated in the DAG. Timing guarantees are derived by means of execution time bounds and static scheduler or dynamic scheduler supported with response time analysis.
- *Execution stack*. These two components are in charge of orchestrating the parallel execution of the application in a time predictable manner, based on the DAG.

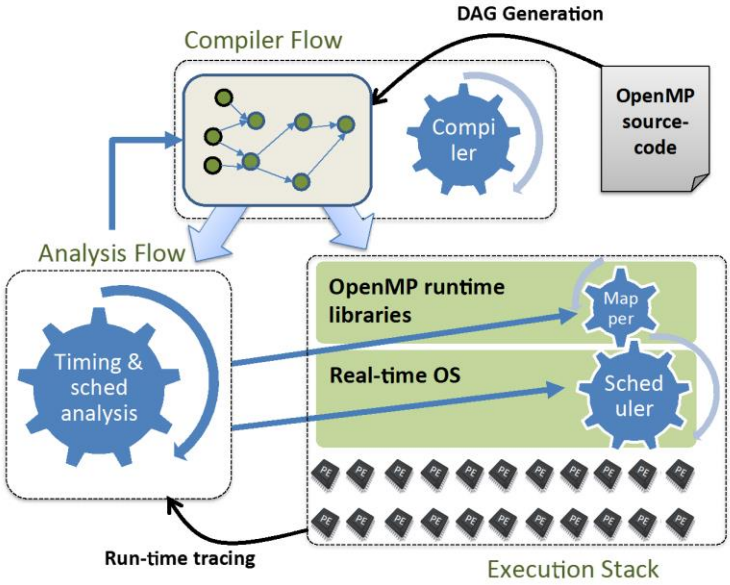

Figure 4 – The UpScale SDK

3 Main Publications

"Architecture Support for Tightly-Coupled Multi-Core Clusters with Shared-Memory HW Accelerators", Dehyadegari, M.; Marongiu, A.; Reza K, M.; Mohammadi, S.; Yazdani, N.; Benini, L., IEEE Transactions on Computer, vol.64, no.8, IEEE, 2015

"He-P2012: Performance and Energy Exploration of Architecturally Heterogeneous Many-Cores.", Conti, F.; Marongiu, A.; Pilkington, C.; Benini, L., Journal of Signal Processing Systems, vol.85, no.3, Springer, 2015

"Simplifying Many-Core-Based Heterogeneous SoC Programming With Offload Directives.", Marongiu, A.; Capotondi, A.; Tagliavini, G.; Benini, L., IEEE Transactions on Industrial Informatics, vol.11, no.4 , IEEE, 2015

"Optimizing memory bandwidth exploitation for OpenVX applications on embedded many-core accelerators.", Tagliavini, G.; Haugou, G.; Marongiu, A.; Benini, L., Journal of Real-Time Image Processing, Special Issue Paper, Springer, 2015

"P-SOCRATES: a Parallel Software Framework for Time-Critical Many-Core Systems.", Pinho, L. M.; Nélis, V.; Meumeu Y., P.; Quiñones, E.; Bertogna, M.; Burgio, P.; Marongiu, A.; Scordino, C.; Gai, P.; Ramponi, M.; Mardiak, M., Journal of Microprocessors and Microsystems, vol.39, no.8, Elsevier, 2015

"A framework for memory contention analysis in multi-core platforms.", Dasari, D.; Nélis, V.; Åkesson, B., Journal of Real-Time Systems, vol.52, no.3, Springer, 2015

"Controlling NUMA effects in embedded manycore applications with lightweight nested parallelism support.", Marongiu, A.; Capotondi, A.; Benini, L., Journal of Parallel Computing, vol.59, Elsevier, 2016

"VirtualSoC: A Research Tool for Modern MPSoCs.", Bortolotti, D.; Marongiu, A.; Benini, L., ACM Transactions on Embedded Computing Systems, vol.16, no.1, ACM, 2016

"He-P2012: Performance and Energy Exploration of Architecturally Heterogeneous Many-Cores.", Conti, F.; Marongiu, A.; Pilkington, C.; Benini, L., Journal of Signal Processing Systems, vol.85, no.3, Kluwer Academic Publishers, 2016

"Tightly-Coupled Hardware Support to Dynamic Parallelism Acceleration in Embedded Shared Memory Clusters.", Burgio, P.; Tagliavini, G.; Marongiu, A.; Benini, L., Design, Automation and Test in Europe, , IEEE, 2014

"He-P2012: Architectural Heterogeneity Exploration on a Scalable Many-Core Platform.", Conti, F.; Pilkington, C.; Marongiu, A.; Benini, L., IEEE 25th International Conference on Application-specific Systems, Architectures and Processors, , IEEE, 2014

"Hard Constant Bandwidth Server: Comprehensive Formulation and Critical Scenarios.", Biondi, A.; Melani, A.; Bertogna, M., 9th IEEE International Symposium on Industrial Embedded Systems, , IEEE, 2014

"Parallelism in Ada: Status and Prospects.", Pinho, L. M.; Moore, B.; Michell, S., 19th International Conference on Reliable Software Technologies - Ada-Europe, vol.8454, Springer, 2014

"Optimal Design for Reservation Servers under Shared Resources.", Biondi, A.; Melani, A.; Bertogna, M.; Buttazzo, G., 26th Euromicro Conference on Real-Time Systems, , IEEE, 2014

"Explicit Preemption Placement for Real-Time Conditional Code.", Peng, B.; Fisher, N.; Bertogna, M., 26th Euromicro Conference on Real-Time Systems, , IEEE, 2014

"P-SOCRATES: a Parallel Software Framework for Time-Critical Many-Core Systems.", Pinho, L. M.; Quinones, E.; Bertogna, M.; Marongiu, A.; Carlos, J. P.; Scordino, C.; Ramponi, M., 17th Euromicro Conference on Digital System Design, , IEEE, 2014

"On the Effectiveness of Energy-Aware Real-Time Scheduling Algorithms on Single-Core Platforms.", Bambagini, M.; Bertogna, M., 19th IEEE International Conference on Emerging Technologies and Factory Automation, , IEEE, 2014

"Response-Time Analysis of Synchronous Parallel Tasks in Multiprocessor Systems.", Maia, C.; Nogueira, L.; Pinho, L. M.; Bertogna, M., 22nd International Conference on Real-Time and Network Systems, , ACM, 2014

"Safe Parallel Programming in Ada with Language Extensions.", Tucker Taft, T.; Moore, B.; Pinho, L. M.; Michell, S., ACM SIGAda Annual Conference on High Integrity Language Technology, , ACM, 2014

"OpenMP and Timing Predictability: A Possible Union?", Vargas, R.; Quinones, E.; Marongiu, A., Design, Automation, and Test in Europe conference, , IEEE, 2015

"A Multi-DAG Model for Real-Time Parallel Applications with Conditional Execution.", Fonseca, J.; Nélis, V.; Raravi, G.; Pinho, L. M., 30th ACM/SIGAPP Symposium On Applied Computing, , ACM, 2015

"An Execution Model for Fine-Grained Parallelism in Ada.", Pinho, L. M.; Moore, B.; Michell, S.; Taft, S., 20th International Conference on Reliable Software Technologies - Ada-Europe, Vol.9111, Springer, 2015

"Response-Time Analysis of Conditional DAG Tasks in Multiprocessor Systems.", Melani, A.; Bertogna, M.; Bonifaci, V.; Marchetti, A.; Buttazzo, G., 27th Euromicro Conference on Real-Time Systems, , IEEE, 2015

"Supporting Component-based Development in Partitioned Multiprocessor Real-Time Systems.", Biondi, A.; Buttazzo, G.; Bertogna, M., 27th Euromicro Conference on Real-Time Systems, , IEEE, 2015

"Timing Analysis of Fixed Priority Self-Suspending Sporadic Tasks.", Nelissen, G.; Fonseca, J.; Raravi, G.; Nélis, V., 27th Euromicro Conference on Real-Time Systems, , IEEE, 2015

"Methodologies for the WCET Analysis of Parallel Applications on Many-core Architectures.", Nélis, V.; Yomsi, P.; Pinho, L., Euromicro Conference on Digital System Design, , ACM, 2015

"Timing Characterization of OpenMP4 Tasking Model.", Serrano, M.; Melani, A.; Vargas, R.; Marongiu, A.; Bertogna, M.; Quiñones, E., ACM/IEEE International Conference on Compilers, Architectures and Synthesis of Embedded Systems, , IEEE, 2015

"Task scheduling strategies to mitigate hardware variability in embedded shared memory clusters.", Rahimi, A.; Cesarini, D.; Marongiu, A.; Gupta, R.; Benini, L., 52nd Annual Design Automation Conference, , ACM, 2015

“Synergistic Architecture and Programming Model Support for Approximate Micropower Computing.”, Tagliavini, G.; Rossi, D.; Marongiu, A.; Benini, L., IEEE Computer Society Annual Symposium on VLSI, , IEEE, 2015

“ADRENALINE: An OpenVX Environment to Optimize Embedded Vision Applications on Many-core Accelerators.”, Tagliavini, G.; Haugou, G.; Marongiu, A.; Benini, L., IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip, , IEEE, 2015

“Enabling Scalable and Fine-Grained Nested Parallelism on Embedded Many-cores.”, Capotondi, A.; Marongiu, A.; Benini, L., IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip, , IEEE, 2015

“Lightweight virtual memory support for many-core accelerators in heterogeneous embedded SoCs.”, Vogel, P.; Marongiu, A.; Benini, L., International Conference on Hardware/Software Codesign and System Synthesis, , IEEE, 2015

“A memory-centric approach to enable timing-predictability within embedded many-core accelerators.”, Burgio, P.; Marongiu, A.; Valente, P.; Bertogna, M., Real-Time and Embedded Systems and Technologies, , IEEE, 2015

“An Evaluation of Memory Sharing Performance for Heterogeneous Embedded SoCs with Many-Core Accelerators.”, Vogel, P.; Marongiu, A.; Benini, L., International Workshop on Code Optimisation for Multi and Many Cores, , ACM, 2015

“Runtime Support for Multiple Offload-Based Programming Models on Embedded Manycore Accelerators.”, Capotondi, A.; Haugou, G.; Marongiu, A.; Benini, L., International Workshop on Code Optimisation for Multi and Many Cores, , ACM, 2015

“A framework for optimizing OpenVX applications performance on embedded manycore accelerators.”, Tagliavini, G.; Haugou, G.; Marongiu, A.; Benini, L., 18th International Workshop on Software and Compilers for Embedded Systems, , ACM, 2015

“Semi-Partitioned Scheduling of Fork-Join Tasks using Work-Stealing.”, Maia, C.; Yomsi, P.; Nogueira, L. ; Pinho, L. M., 13th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, , IEEE, 2015

“A Lightweight OpenMP4 Run-time for Embedded Systems.”, Vargas, R.; Royuela, S.; Serrano, M.; Martorell, X.; Quiñones, E., 21st Asia and South Pacific Design Automation Conference, , IEEE, 2016

“Response-Time Analysis of DAG Tasks under Fixed Priority Scheduling with Limited Preemptions.”, Serrano, M.; Melani, A.; Bertogna, M.; Quiñones, E., Design, Automation, and Test in Europe conference, , IEEE, 2016

“On the effectiveness of OpenMP teams for cluster-based many-core accelerators.”, Capotondi, A.; Marongiu, A., International Conference on High Performance Computing & Simulation, , IEEE, 2016

“Response Time Analysis of Sporadic DAG Tasks under Partitioned Scheduling.”, Fonseca, J.; Nelissen, G.; Nélis, V.; Pinho, L. M., 11th IEEE International Symposium on Industrial Embedded Systems, , IEEE, 2016

“An optimized task-based runtime system for resource-constrained parallel accelerators.”, Cesarini, D.; Marongiu, A.; Benini, L., Design, Automation & Test in Europe Conference & Exhibition, , IEEE, 2016

"A static scheduling approach to enable safety-critical OpenMP applications.", Melani, A.; Serrano, M.; Bertogna, M.; Cerutti, I.; Quiñones, E.; Buttazzo, G., 22nd Asia and South Pacific Design Automation Conference, , IEEE, 2017

"Time Criticality Challenge in the Presence of Parallelised Execution.", Pinho, L. M.; Quiñones, E.; Bertogna, M.; Benini, L.; Carlos, J. P.; Scordino, C.; Ramponi, M., 2nd Workshop on High-performance and Real-time Embedded Systems, , , 2014

"A Virtualization Framework for IOMMU-less Many-Core Accelerators.", Pinto, C.; Marongiu, A.; Benini, L., International Workshop on Manycore Embedded Systems, , ACM, 2014

"On the Relevance of Architectural Awareness for Efficient Fork/Join Support on Cluster-Based Manycores.", Al-Khalissi, H.; Berekovic, M.; Marongiu, A., International Workshop on Manycore Embedded Systems, , ACM, 2014

"How to deal with control-flow information in parallel real-time applications?", Fonseca, J.; Nélis, V.; Raravi, G.; Pinho, L. M., 5th Real-Time Scheduling Open Problems Seminar, , , 2014

"The challenge of time-predictability in modern many-core architectures.", Nélis, V.; Yomsi, P. M.; Pinho, L. M.; Fonseca, J.; Bertogna, M.; Quiñones, E.; Vargas, R.; Marongiu, A., 14th International Workshop on Worst-Case Execution Time Analysis, , Schloss Dagstuhl--Leibniz-Zentrum fuer Informatik, 2014

"Another look at the pWCET estimation problem.", Nélis, V.; Yomsi, P. M.; Pinho, L. M.; Bernat, G., Work in Progress Session, IEEE Real-Time Systems Symposium, , IEEE, 2014

"A system model and stack for the parallelization of time-critical applications on many-core architectures.", Nelis, V.; Yomsi, P. M.; Pinho, L. M.; Quiñones, E.; Bertogna, M.; Marongiu, A.; Gai, P.; Scordino, C., 3rd Workshop on High-performance and Real-time Embedded Systems, , , 2015

"Real-Time Fine-Grained Parallelism in Ada.", Pinho, L. M.; Moore, B.; Michell, S.; Taft, S., International Real-Time Ada Workshop, , ACM , 2015

"Analysis of self-interference within DAG tasks.", Fonseca, J.; Nélis, V.; Nelissen, G.; Pinho, L. M., 6th Real-Time Scheduling Open Problems Seminar, , , 2015

"Online Admission of Parallel Real-Time Tasks.", Maia, C.; Nogueira, L.; Pinho, L. M., 6th Real-Time Scheduling Open Problems Seminar, , , 2015

"Real-Time Support in the Proposal for Fine-Grained Parallelism in Ada.", Pinho, L. M.; Moore, B., Work in Progress Session, IEEE Real-Time Systems Symposium, , IEEE, 2015

"The variability of application execution times on a multi-core platform.", Nélis, V.; Yomsi, P. M.; Pinho, L. M., 16th International Workshop on Worst-Case Execution Time Analysis, Vol.55, Schloss Dagstuhl--Leibniz-Zentrum fuer Informatik, 2016

"Reduction of Parallel Computation in the Parallel Model for Ada.", Taft, S.; Moore, B.; Pinho, L. M.; Michell, S., 18th International Real-Time Ada Workshop, , ACM, 2016

“Constraints on the Use of Executors in Real-time Systems.”, Michell, S.; Moore, B.; Pinho, L. M.; Taft, S., 18th International Real-Time Ada Workshop, , ACM, 2016