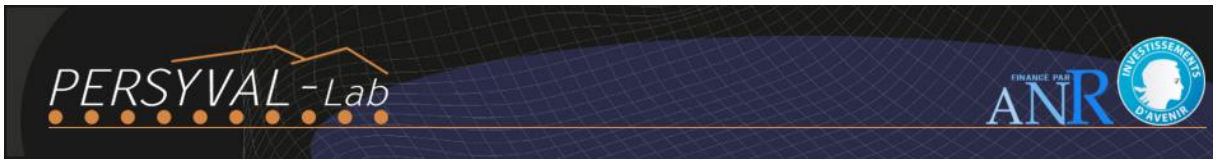

Proposition de projet exploratoire, 31 janvier 2013
Soumis dans le thème PCS (Pervasive Computing Systems)

CESyMPA: Critical Embedded Systems on Multiprocessor Architectures: Towards a Certifiable HW/SW Solution

People involved:

- Florence Maraninchi, VERIMAG (Florence.Maraninchi@imag.fr)
- Pascal Raymond, VERIMAG (Pascal.Raymond@imag.fr)
- Matthieu Moy, VERIMAG (matthieu.Moy@imag.fr)
- Claire Maiza, VERIMAG (Claire.Maiza@imag.fr)
- Stéphane Mancini, TIMA (stephane.mancini@imag.fr)
- Abbas Sheibanyrad, TIMA (abbas.sheibanyrad@imag.fr)

1. Scientific Context

The embedded computing systems found in cars, aircraft and spacecraft, nuclear power plants, etc., are said to be ***critical***, because of the potential consequences of a bug for human lives. In these domains, there exist *norms* on the development process of the computing systems, and even independent ***certification*** *authorities* giving their approval on the computing system. For instance, in the avionics domain, there is a European norm (DO 178B), which imposes strict criteria on how the software projects should be conducted. Before being able to ``fly'' a piece of software, a company has to demonstrate that it has been designed with care. All the tools used in the design flow, including the C compiler at the end, have to be certified. There are similar norms for the hardware components (DO 154). One of the main functional criteria that matter for the certification is the **predictability** of the system behavior. In other words, it should be possible to determine in advance what will happen when the system is deployed. Predictability is usually ensured by careful implementation methods, which guarantee that the behavior is **deterministic**. This also means that the complete implementation process is well understood and mastered, and that the potential sources of non-determinism in the final system have been identified and suppressed. Besides, in critical embedded software, **timing** is part of the function, because the software usually implements some control law, and the execution time is strongly related to the sampling rate of the inputs. Producing a correct value, but too late, is like producing a wrong value. Concerning the timing aspects, predictability means, in particular, that the execution time of a piece of software has to be computable in advance or, at least, that there should be a way to find a guaranteed upper bound, called the *worst-case-execution-time* (WCET).

The first generation (in the 80's) of critical embedded systems, e.g., in aircraft or nuclear power-plants, used the very simple hardware available at that time, and very static software solutions, which was a satisfactory way to guarantee determinism. For instance, in the domain of nuclear power plants, a conservative solution is still in use: the embedded software is an infinite loop in which the processor first reads the sensors, then computes the response and updates its internal memory, then writes to the actuators. This code can run on a machine without any operating system. The hardware architecture is simple, based on a variant of the Motorola 68040 processors which were designed around 1980. This type of processor has no sophisticated mechanisms like branch prediction, caches, pipelines, speculation, etc. The predictability of execution time for a given piece of code is quite good. The absence of an operating system and the simplicity of the processor guarantee determinism.

Even if some of these solutions are still in use in highly critical contexts like nuclear power-plants, they are obviously reaching their limits. In almost all domains of critical systems, it is no longer possible to use simple hardware. First, the old hardware components will soon disappear because no company keeps producing them; second, more and more functions have to be implemented by computing systems and there is a need for **more computing power** than that available with the old hardware components. Unfortunately, the modern individual processors or the multiprocessor architectures are increasingly **non-deterministic** and **unpredictable**. They are designed with the objective of improving the *average performance* of a computing

system, but they may exhibit a very high variation in computation time. The main sources of non-determinism include: the sophisticated features of the processors like pipelining and speculation, the competition of the processors for memory accesses, and the communication elements (buses, networks on chip). The current hardware architectures are, therefore, not suitable for critical embedded systems.

The question of how to design an embedded system for the critical domains, using modern hardware architectures, therefore raises a huge interest, both in companies, and in academia**.** There is no satisfactory solution yet. A related question is that of **mixed-criticality** systems, i.e., systems in which several pieces of software, some critical, and some which are not, are executed on the same architecture. In this case, there are additional problems of time and space isolation between these pieces of software. This type of problem is very common in the automotive industry, for instance, where there is a trend towards fewer processors in a car, for more functions.

2. Scientific Objectives, Related and Non-Related Work

We think that these topics deserve a new and fresh look, "forgetting" about the constraints of existing components or software solutions. As already mentioned, the existing multi-processor architectures are designed for *average performance* and the solutions at all levels are meant to be *transparent* for the application programmer (from the prefetching mechanisms in memory controllers to the routing algorithms in networks on a chip, including the very principle of a cache, load balancing techniques, etc.). Several levels of such transparent mechanisms that try and improve the average performance constitute the main obstacle to predictability.

In this project, we aim at exploring ways to implement critical systems as software running on multiprocessor architectures, in such a way that the complete solution be simple and provably deterministic, therefore acceptable by certification authorities. The main directions are:

- In contrast with other projects involving industrial partners and existing hardware components, we will consider here that we are free to propose **new solutions at all levels**.
- We will keep in mind that the solutions have to be **sufficiently simple** for the certification constraint. The objective is not to define new sophisticated architectures and optimal software-to-hardware mapping algorithms. A solution that under-exploits the hardware, but in a simple and deterministic way, will be preferred.
- We will adopt a **holistic** view of the problem (hardware and all software layers together) which is necessary for the certification objective anyway, and also explore **cross-layer** solutions (e.g., allow the software to control the running modes of the hardware) if they improve performance without compromising predictability. In other words, we aim at *programmable and predictable performance*, instead *of transparent average performance*.

We would like to come up with a clear idea of what could be an ideal hardware architecture and design flow for "predictable-by-construction" critical embedded systems. Even if it is not feasible for a number of reasons, ranging from hardware fabrication problems to economic viability, this is scientifically worth trying because it would give an estimation of the distance between such an ideal solution and what exists now, and help identifying the tricky problems with the current hardware.

The related work covers hardware design and modeling, software specification, concurrent programming models, WCET analysis, real-time scheduling, and software-to-hardware mapping (see more details below).
The **non-related work** includes: (i) *operating systems*, because in the domain of critical systems, the operating system services are usually compiled together with the application software, and there is nothing like a runtime system; (ii) *automatic partitioning of the software* for a multi-processor architecture, because this partitioning is usually imposed by the application itself (a task corresponding to one control law, for instance) and we may expect hints given by the application designer on how to partition it; (iii*) automatic and dynamic load balancing* because of our main objective of predictability; we might exploit well-defined running modes of the applications, in order to reconfigure the hardware for better performances, but this should be computable in advance.
Some of the points that deserve attention are the following:

- Among the set of existing solutions for mapping real-time applications to hardware, we will have to identify those that make **realistic assumptions** about the sources of non-determinism in the hardware (some tricky behaviors like *timing anomalies* in the processors may have a huge impact on the correctness of some scheduling algorithms, for instance).

- We will study specification formalisms for application software, in which there is a way to **capture all the characteristics** that help getting a predictable implementation. For instance, a lot of critical applications are divided into dataflow blocks for which we know a "clock", i.e., points in time when it needs activations.
- Similarly, among the new hardware components, we will have to select those for which there is a way to get **a guaranteed behavior and timing**. For instance we may prefer networks on chip that offer programmable routers, in such a way that the whole traffic can be studied offline, and scheduled once and for all to avoid collisions or deadlocks.
- The types of **memories** to be used (DRAM, SRAM) are important, because the latency of accesses may vary a lot depending on the type of memory, and their controllers may contain mechanisms like pre-fetching that degrade predictability.
- The **granularity of data** also matters a lot, and it might be necessary to distinguish between control information that could travel between processors using dedicated links, and data like images, which need accesses to the memory via the network on chip.
- We will look at solutions in which the **software and the hardware exchange information**, for instance when the application changes modes of operation, and the hardware has to be reconfigured (by changing the arbitration policy on a bus, for instance).

3. Local, National and International Contexts

The work on how to implement critical systems on multiprocessor architectures follows several paths:

a) The study of existing processors, in order to identify subclasses having better predictability (see, e.g., the work by R. Wilhelm, U. Saarbrucken)
b) The design of predictable-by-construction new hardware architectures (e.g., the PRET machine in the US, the European projects MERASA, PAR MERASA and T-CREST)
c) New methods for the evaluation of the worst-case-execution-time (WCET) on modern processors, including models of the most non-deterministic hardware features (e.g., U. Saarbrucken, the Absint company, Inria-Popart, IRIT, INRIA-Rennes)
d) The extension of traditional WCET analyses to cope with the influence of parallel computing units and communication elements in modern architectures (e.g., work by A. Burns, U. York, European action TACLE)
e) The design of dedicated implementation methods for multi- or many-core architectures, starting from high-level models or programming languages (e.g., D. Potop at Inria-Aoste, LIP6, PRET-C associated with the PRET machine)
f) The design of controllable hardware elements for better predictability (IRIT Toulouse, LIP6 Paris, P. Marwedel at U. Dormund, ...)

Our objectives are close to those of T-CREST (point **b**) which focuses on hardware optimized for the WCET analysis. We go a step further by considering cross-layer solutions, i.e., cases in which controlling the hardware based on application-level information may improve determinism. The work in (**a**) helps selecting a simple processor; the work in (**d**) helps identifying the communication elements that have no chance to satisfy our needs; the work in (**f**) is a starting point and a source of solutions for our new architecture; the work in (**e**) gives directions for our objective of building a complete chain.

4. Expected Results and Methodology

The **results** of the project will be complete proof-of-concept solutions covering: (i) a formalism for the description of the application, capturing all the constraints that may be exploited for better predictability of the resulting implementation; (ii) a multiprocessor hardware architecture, offering hooks for a precise control by the software; (iii) a set of algorithms for the implementation of applications expressed in the formalism, onto the architecture; (iv) a set of proofs for the phases of the implementation, guaranteeing determinism and time-predictability; (v) a discussion on the "distance" between this ideal solution and what can be done on the existing hardware.

As usual, the project requires a first phase to share expertise, build a common knowledge, and review existing work. However, this will overlap with the construction of the first "solution". Indeed, we will adopt an agile-like **methodology**, starting from something very simple yet representative of the problems that occur at all levels, and then enrich this first example in several steps. We will choose a simple existing processor and specify new hardware elements for a multiprocessor architecture, using simulators and virtual-prototyping tools (e.g.,

SystemC with instruction-set-simulators). We will select a few classes of critical applications for which there is a need for more computing power and available information like “clocks” for parts of the software. We will reuse and extend the dataflow formalisms and programming models popular in both critical software and multi-core implementations. We will not look (yet) at modular solutions, or at dynamic and reconfigurable systems.

5. Value Added by the Collaboration, and External Partners

Verimag-Synchrone brings the expertise in critical applications, formally-defined programming languages, and predictable implementations on mono-processor architectures with real-time operating systems. TIMA-SLS brings the expertise in hardware design, implementations on multi-processor architectures, networks-on-chip, and virtual prototyping. The partners therefore cover all the needed aspects. Moreover, both Verimag and TIMA have been involved in a lot of projects with industrial partners. Among them, some can be consulted for representative critical applications and the kind of solution that has some chance to be accepted by certification authorities: Peugeot, Airbus, Astrium, Thalès avionics, Continental. Others are references on modern multiprocessor architectures: STMicroelectronics, Kalray. Similarly, both laboratories are involved in national or international projects on some of the points mentioned in section 3, with the main actors of these research activities.

6. Organization and Activities

The specification work will be done by the permanent researchers involved in the project, together with master students. The necessary developments will be done by students doing their engineer internship. We plan to invite people who are specialists in one or several topics needed for the project. This includes:

- **Christine Rochange or Pascal Sainrat**, IRIT, Toulouse, for their expertise on predictable architectures, and their participation in the European projects MERASA and PAR-MERASA
- **Dumitru Potop-Butucaru**, INRIA Rocquencourt, for his work on static schedules.
- **Jan Reinecke**, Saarbrucken, for his work on a predictable DRAM memory, and the PRET project
- **Reinhart Wilhelm**, Saarbrucken, for his expertise on the analysis of existing processors, and his participation in the projects PROMPT and Predator
- **Joel Goossens**, Brussels, for his work on multi-core implementations
- **Christian Ferdinand**, Saarbrucken and Absint, for his participation in the projects T-Crest and Predator, and his industrial point of view on predictability
- **Rob Davis,** York, for his work on scheduling techniques that take into account realistic assumptions on the hardware.
- **Kees Goossens,** TuE Eindhoven, for his expertise in mixed soft and hard real-time systems, and his work on real-time networks on chip.

7. Relevant Publications of the Two Groups

- **Stéphane Mancini and Frédéric Rousseau.** Enhancing non-linear kernels by an optimized memory hierarchy in a High Level Synthesis flow. Design, Automation & Test in Europe Conference, 2012.
- **Stéphane Mancini, Zahir Larabi, Yves Mathieu, Tomasz Toczek, and Lionel Pierrefeu.** Exploration of 3D grid caching strategies for ray-shooting. Journal of Real Time Image Processing; 2012.
- **Tomasz Toczek and Stéphane Mancini.** Algorithm-Architecture Matching for Signal and Image Processing, chapter Efficient Memory Management for Uniform and Recursive Grid Traversal. Springer, 2011.
- **Sahar Foroutan, Abbas Sheibanyrad, and Frédéric Pétrot**. Cost-efficient buffer sizing in shared-memory 3D-MPSoCs using wide I/O interfaces, DAC, ACM, San Francisco, 2012
- **Christoph Cullmann, Christian Ferdinand, Gernot Gebhard, Daniel Grund, Claire Maiza, Jan Reineke, Benoit Triquet and Reinhard Wilhelm.** Predictability Considerations in the Design of Multi-Core Embedded Systems. Embedded Real Time Software and Systems, 2010.
- **Reinhard Wilhelm , Sebastian Altmeyer, Claire Burguière-Maiza, Daniel Grund, Jörg Herter, Jan Reineke, Björn Wachter, Stephan Wilhelm.** Static Timing Analysis for Hard Real-Time Systems. VMCAI, 2010.
- **Sebastian Altmeyer, Claire Maiza and Jan Reineke.** Resilience Analysis: Tightening the CRPD Bound for Set-Associative Caches. Languages, compilers, and tools for embedded systems (ACM LCTES), 2010.

- **Sebastian Altmeyer, Robert I. Davis, Claire Maiza.** Improved cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems. Real-Time Systems, 2013
- **Albert Benveniste, Anne Bouillard, Paul Caspi**. A Unifying View of Loosely Time-Triggered Architectures. ACM International Conference on Embedded Software (EMSOFT), 2010
- **Mouaiad Alras, Paul Caspi, Alain Girault, Pascal Raymond**. Model-Based Design of Embedded Control Systems with a Synchronous Intermediate Model. 6th IEEE International Conference on Embedded Systems and Software (ICESS), 2009
- **Marc Pouzet, Pascal Raymond** Modular Static Scheduling of Synchronous Data-flow Networks, An efficient symbolic representation. International Conference on Embedded Software (EMSOFT), 2009
- **Paul Caspi, Norman Scaife, Christos Sofronis, Stavros Tripakis**. Semantic-preserving multitask implementation of synchronous programs. ACM Trans. on Embedded Computing Systems, 2008
- **Matthieu Moy, Florence Maraninchi, Laurent Maillet-Contoz.** LusSy: an open Tool for the Analysis of Systems-on-a-Chip at the Transaction Level. Design Automation for Embedded Systems, 2006
- **N. Berthier, F. Maraninchi, L. Mounier**. Synchronous Programming of Device Drivers for Global Resource Control in Embedded Operating Systems. ACM Transactions on Embedded Computing Systems (TECS), 2012

8. Costs

Invitations of researchers	**3000**
Conferences for the members of the project	**2300**
Master students (5 months x 470 euros) x 2	**4700**
Total	**10000**