INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *Project-Team caps*

# *Compilation, architectures des processeurs superscalaires et spécialisés*

*Rennes - Bretagne-Atlantique*

THEME COM

# Table of contents

# 1. Team

**Research Scientist**

André Seznec [ Research Director Inria, team leader, HdR ]
Pierre Michaud [ Research scientist ]
François Bodin [ External collaborator, HdR ]

**Faculty Member**

Jacques Lenfant [ Professor, University of Rennes 1, HdR ]
Isabelle Puaut [ Professor, University of Rennes 1, HdR ]

**Technical Staff**

Erven Rohou [ from 01/10/08 ]
Florence Dru [ till 31/08/08 ]
Sylvain Leroy
Guillaume Papauré
Robin Schmutz [ till 14/11/08 ]
Khaled Ibrahim [ from 01/06/08 till 30/11/08 ]
Damien Fétis [ from 01/03/08 ]
Thomas Piquet [ from 10/02/08 ]

**PhD Student**

Christophe Levointurier [ Cifre AQL ]
Jean-François Deverge [ MENRT allocation, until 31/08/08 ]
Julien Dusser [ Inria Allocation ]
Robert Guziolowski [ Inria Allocation ]
Damien Hardy [ MENRT Allocation ]
Eric Petit [ Inria allocation till 30/11/08 ]

**Post-Doctoral Fellow**

Khaled Ibrahim [ till 31/05/08 ]

**Administrative Assistant**

Evelyne Livache [ TR Inria ]

# 2. Overall Objectives

## 2.1. Overall Objectives

High performance microprocessors are used in various information technology applications ranging from supercomputers, high-end multiprocessor servers, to PCs and workstations, but also high-end embedded applications (avionics, networks, as well as consumer products such as automotive, set-top boxes or cell phones). The theoretical performance of these processors has been increasing continuously for the past two decades. This trend continues at the cost of a rising hardware complexity (transistor count, power consumption, design cost). At the same time, extracting a significant part of this theoretical performance becomes more and more difficult for the end user, even with the assistance of a compiler.

Research in the CAPS project-team ranges from processor architecture to software platforms for performance tuning, including compiler/architecture interactions, to processor simulation techniques and worst case execution time (WCET) evaluation techniques. Peak performance is one of the objectives, however finding tradeoffs between hardware complexity and performance, performance and power consumption (or code size) is also a major issue, while accurately evaluating (more precisely majoring) the execution time is the challenge for embedded real time systems.

Our research in computer architecture covers memory hierarchy, branch prediction, superscalar implementation, as well as SMT and multicore processors. In the recent past, we have proposed several new complexity-effective structures for caches and branch predictors[2], [11], and we are still very active in these areas (cf. 6.1.1, 6.1.6). We pursue researchs on architecture exploiting thread level parallelism on a single chip (cf. [5], 6.1.5). At the same time, power consumption and temperature hot spot management have become major issues for all processors. We have initiated a research activity on temperature management at architectural level (cf. 6.1.3). We are also studying how the compiler and the architecture can interact to optimize the power consumption/performance tradeoff [10].

Performance, but also power consumption or hardware system cost depends on the processor architecture but can also be managed at the compiler/code generation level. We are exploring thread extraction for the different hardware components for heterogeneous SOCs (System On a Chip) featuring special purpose hardware and one or more execution cores (cf. 6.2.1.1).

In hard real-time embedded systems the task WCETs must be correctly evaluated, so that it can be proven that task temporal constraints (typically, deadlines) will be met. Our research concerns methods for automatically computing upper bounds of the execution time of applications on a given hardware platform. Embedded platforms may now feature caches, branch predictors, complex pipeline, .... A particular focus is put on hardware-level analysis (static analysis based on timing models) and compiler-directed schemes aimed at improving predictability. Our studies concern WCET-oriented (as opposed to average-performance oriented) compilation and measurement-based WCET estimation (cf 6.4).

Our research is partially supported by industry (Intel). We also participate in several institutionally funded projects (NoE HIPEAC, IP Fet project SARC, ANR funded QCDnext, GaLogic, Para, Mascotte, PetaQCD, and “Pôles de compétitivités” funded Scalimages, Sceptre,Terops, POPS and Serenitec). Some of the research prototypes developed by the project during the past few years have been transferred to industry through the CAPS Entreprise start-up (cf. 7.2).

# 3. Scientific Foundations

## 3.1. Panorama

Research activities by the CAPS team range from highly focused studies on specific processor architecture components to software environments for performance tuning on embedded systems. In this context, the compiler/architecture interaction is at the heart of the team research.

In this section, we briefly present the remaining challenges in uniprocess architecture, the new challenges and opportunities for architects created by single-chip hardware thread parallelism, and the challenges for compilers on embedded processors.

## 3.2. Uniprocess architecture

**Keywords:** *branch prediction*, *memory hierarchy*, *speculative execution*, *superscalar processor*.

The gap between processor cycle time and main memory access time is increasing at a tremendous rate and is reaching up to 1000 instruction slots. At the same time, the instruction pipeline depth is also increasing and several instructions can be executed within a single cycle. A branch misprediction will soon lead to a 100-instruction slots penalty.

Over the past 10 years, research results have allowed to limit the performance loss due to these two phenomena. The average effective performance of processors has remained in the range of one instruction per cycle, while these two gaps were increasing by an order of magnitude.

The use of a complex memory hierarchy has been generalized over the past decade. On modern microprocessors, both software and hardware prefetching are now widely used to enable the on-time presence of data and instructions in the memory hierarchy. Highly efficient, but complex data hardware prefetch mechanisms, have been proposed to hide several hundreds of instruction slots [47]. The challenge for computer architects is to reduce the complexity of these hardware mechanisms to enable simpler implementations. Another challenge is to propose new prefetch mechanisms that can hide several thousands of instruction slots.

Over the past decade, efficient branch prediction mechanisms have been proposed and implemented [44][11]. Both branch directions and targets (even indirect jump targets) [36] are predicted. Most of these predictors exploit either local or global branch history. The accuracy of the prediction seems to be reaching a plateau.

The complexity of many components in the processor (in terms of silicon area, power consumption and response time) increases superlinearly (and often quadratically) with the issue width e.g. register renaming, instruction scheduling, bypass network and register file access. These components are becoming the bottlenecks that limit the issue width and the cycle time [50].

It is now possible to integrate several processors on a single chip. One of the main issues in uniprocessor design is to define a processor core architecture that will be able to achieve high performance both on uniprocess workloads and on multiprocess workloads. On uniprocess workloads, the processor must exploit all the resources (memory bandwidth, caches) of the system, while these resources are shared and should not be wasted on a multiprocess workload.

While complexity of the processors is steadily increasing, predicting, understanding and explaining the effective behavior of the architecture is becoming a major issue, in particular for embedded systems. Unfortunately, high performance often comes with high unpredictability and variability in performance. Designing architectures with predictable and high performance will become a major challenge for computer architects as well as compiler designers in the next few years.

## 3.3. Exploiting task parallelism on a single chip: multicore and SMT processors

**Keywords:** *multicore processor*.

It becomes more and more difficult to exploit higher degrees of instruction-level parallelism on superscalar processors. Thus, it has been proposed to exploit task-level parallelism. Two different approaches exist, namely the *multicore* approach and the *simultaneous multi-threading* (SMT) approach. Task parallelism is actually a simple way to increase the execution throughput in certain contexts : embedded applications, servers, multiprogrammed systems, scientific computing, ...

The straightforward way to implement task parallelism is to use multiple distinct processors. Current technology is able to put one billion transistors on a single die. This allows to integrate several high-performance computing cores on the same chip, and provides several advantages.

General purpose multicore processors are already available and will become mainstream in the next few years. On a multicore, the tasks execute on distinct processing units. Resource sharing concerns only one or several on-chip cache levels, and chip pins. This is to be contrasted with SMT processors, on which all resources are shared apart a few buffers [58]. However, the main difficulty for the design of SMT processors is the design of a very wide issue superscalar processor. Though SMT and multicore approaches both exploit task parallelism, they are orthogonal, as illustrated by the dual-core SMT Pentium 4 and the dual core SMT IBM Power 5.

A key issue concerning SMT / multicore processors is whether they can improve sequential execution. Among possible improvements, one may seek to obtain a more reliable execution (for instance [53] by redundant execution), or more performance. A few ideas have been recently proposed to speed-up sequential task execution, like for instance speculative threads [41], exception handling [63], helper threads for branch prediction [37], helper threads for memory prefetching [48], etc. Among solutions already proposed, it is not yet clear which are viable and which are not. It will depend on the performance gain / hardware complexity tradeoffs. Ongoing research on this topic will decide the scope of future SMT/multicore processors.

## 3.4. Compiling and optimizing for embedded applications

**Keywords:** *Code Optimization*, *Compilation*, *Embedded processors*, *High Performance*, *ISA Simulation*.

Embedded processors range from very small, very low-power systems (for instance for telemetry counter sensors which must run on one battery for 10 years) to power hungry high-end processors used in radars or set-top boxes. The spectrum of softwares range from very small code kernels (a few Kinstructions) to millions of code lines including a real time operating system. The constraints on the code quality vary from “just no bugs” to safety critical with hard real time problems, but may also be to achieve a determined performance level at the smallest possible hardware cost or the smallest possible power consumption. Therefore embedded processors are presenting many new challenges [43] to the hardware and compiler research community.

Code optimization for embedded processors does not directly fit in the traditional "best speed effort at any price” assumption used for supercomputers and workstations. The “common case” paradigm is not relevant for the design of compiler optimizations for an embedded processor: one must concentrate on the few optimizations that will bring performance on the few relevant target applications. Execution time is not the only and ultimate criterion. In many cases, execution time may be less important than memory size or power consumption. Binary compatibility, while often important, is not completely mandatory.

Many challenges have to be addressed at the compiler/optimizer level. These include compiling under constraints and mastering the optimization interactions.

Finding a tradeoff between binary code size and execution time [62], [39] is a major issue in many applications. For small micro-controllers, “the smaller the code, the faster” is an effective rule of thumb. However, for recent embedded processors featuring instruction level parallelism (e.g., VLIW processors), faster code generally means larger code size [4]. To master code size, code compression techniques [35] can also be used to reduce memory size of infrequently executed code regions.

In the context of real time systems, average performance is often not a critical issue, but the worst case execution time (WCET) may be critical. WCET estimations can be either obtained by measurements or by static analysis of programs. However these techniques are challenged by recent processors whose behavior is fundamentally difficult to predict [54]. A better synergy between compilers and hardware must be set up and supported by performance debugging tools.

Power consumption is becoming a major issue on most processors. For a given processor, power consumption is highly related to performance: in most cases, a compiler optimization reducing execution time also reduces power consumption [57]. A more interesting issue arises with configurable hardware, for instance cache memories that can vary in size or associativity. In that case, the compiler can trade off performance for power consumption [60], [59].

While many optimizations and code transformations have been proposed over the past two decades, the interactions between these optimizations are not really understood. The many optimizations used in modern compilers sometimes annihilate each other [38], [46]. Performance tuning is therefore an important and time consuming task. For embedded systems, developers must perform this tuning while preserving code size or power consumption. New software environments must be designed for this performance tuning [42], [49], [61]. An associated challenge is to preserve the link between aggressively optimized low level code and the source code [56]. As an alternative (or a complement) to performance tuning, automatic iterative compilation techniques [45] address the interactions of optimizations through the use of feedback, to find efficient code transformation sequences.

Time-to-market is a major challenge for embedded processor designers. Wide spectrum of possible derived hardware platforms (configurations, co-processors, etc.) is also a major issue for embedded system designers. Defining or dimensioning an embedded system (hardware, compiler and application) requires to explore a large solution space for the best cost/performance/application. Retargetable compiler infrastructures key issues to support design exploration. Compiled simulation is one of the promising technique for very fast ISA simulation. These simulators can be used to retarget the compiler very early in the design process.

Finally, many embedded platforms are now built using System-On-a-Chip (SoC) components. A SoC may feature several different processors along with various hardware accelerators. The design of an application for such a SoC must first handle the partitioning of the applications, i.e., one must determine which part of the application is mapped on the different computing units of the SoC. Although the fine grain parallelism is exploited by various automatic optimizations, such as SIMD or loop transformations, the extraction of the coarse-grain parallelism in applications is still performed by the programmer. Automatic or semi-automatic parallelisation for these platforms is one of the software challenges of the next decade.

# 4. Application Domains

## 4.1. Application Domains

**Keywords:** *biology*, *compilers*, *engineering*, *environment*, *health*, *multimedia*, *performance*, *processor architecture*, *telecommunications*, *transportation*.

The Caps team is working on the foundation technologies for computer science: processor architecture and performance oriented compilation. The research results have impacts on any application domain that requires high performance executions (telecommunication, multimedia, biology, health, engineering, environment, ...), but also on many embedded applications that exhibit other constraints such as power consumption, code size and guaranteed response time. Our research activity implies the development of software prototypes (cf. 5.1, 6.2)

# 5. Software

## 5.1. Panorama

The CAPS team is developing several software prototypes for research purposes: compilers, architectural simulators, programming environments, ....

Among the many prototypes developed in the project, we describe here **ATMI**, a microarchitecture temperature model, and **HAVEGE**, an unpredictable random number generator, two softwares developed by the team.

## 5.2. ATMI

**Keywords:** *Microarchitecture temperature model*.

**Participant:** Pierre Michaud.

**Contact :** Pierre Michaud

**Status :** Registered with APP Number IDDN.FR.001.250021.000.S.P.2006.000.10600, Available under GNU General Public License

Research on temperature-aware computer architecture requires a chip temperature model. General purpose models based on classical numerical methods like finite differences or finite elements are not appropriate for such research, because they are generally too slow for modeling the time-varying thermal behavior of a processing chip.

We have developed an ad hoc temperature model, ATMI (Analytical Model of temperature in MIcroprocessors), for studying thermal behaviors over a time scale ranging from microseconds to several minutes. ATMI is based on an explicit solution to the heat equation and on the principle of superposition. ATMI can model any power density map that can be described as a superposition of rectangle sources, which is appropriate for modeling the microarchitectural units of a microprocessor.

Visit http://www.irisa.fr/caps/projects/ATMI or contact Pierre Michaud

## 5.3. HAVEGE

**Keywords:** *Unpredictable random number generator.*

**Participant:** André Seznec.

**Contact :** André Seznec

**Status :** Registered with APP Number IDDN.FR.001.500017.001.S.P.2001.000.10000. Available under the LGPL license.

An unpredictable random number generator is a practical approximation of a truly random number generator. Such unpredictable random number generators are needed for cryptography. Modern superscalar processors feature a large number of hardware mechanisms that target performance improvements: caches, branch predictors, TLBs, long pipelines, instruction level parallelism,.... The state of these components is not architectural (i.e., the result of an ordinary application does not depend on it), it is also volatile and cannot be directly monitored by the user. On the other hand, every invocation of the operating system modifies thousands of these binary volatile states.

HAVEGE (HArdware Volatile Entropy Gathering and Expansion) is a user-level software unpredictable random number generator for general-purpose computers that exploits these modifications of the internal volatile hardware states as a source of uncertainty. HAVEGE combines on-the-fly hardware volatile entropy gathering with pseudo-random number generation.

The internal state of HAVEGE includes thousands of internal volatile hardware states and is merely unmonitorable. HAVEGE can reach an unprecedented throughput for a software unpredictable random number generator: several hundreds of megabits per second on current workstations and PCs.

The throughput of HAVEGE favorably competes with usual pseudo-random number generators such as `rand()` or `random()`. While HAVEGE was initially designed for cryptology-like applications, this high throughput makes HAVEGE usable for all application domains demanding high performance and high quality random number generators, e.g., Monte Carlo simulations.

HAVEGE is currently distributed as user-level library as well as Linux driver.

Last, but not least, more and more modern appliances such as PDAs or cell phones are built around low-power superscalar processors (e.g., StrongARM, Intel Xscale) and feature complex operating systems. HAVEGE can also be implemented on these platforms. A HAVEGE demonstrator for such a PDA featuring PocketPC2002 OS and a Xscale processor is available.

Visit http://www.irisa.fr/caps/projects/hipsor/HAVEGE.html or contact André Seznec.

# 6. New Results

## 6.1. Processor Architecture

**Keywords:** *Processor*, *branch prediction*, *cache*, *locality*, *memory hierarchy*, *multicore*, *power*, *temperature*.

**Participants:** Julien Dusser, Robert Guziolowski, Pierre Michaud, Thomas Piquet, André Seznec.

Our research in computer architecture covers memory hierarchy, branch prediction, superscalar implementation, as well as SMT and multicore issues. In the recent past, we have proposed several new complexity-effective cache and branch predictor structures [2], [11], [14]. We are still refining, analyzing and exploring new cache management policies (cf. 6.1.1), but also proposing new cache management policies for multicores (cf. 6.1.2). New directions in branch prediction and instruction fetch have been explored (cf. 6.1.6). We are also exploring new directions of heterogeneous multicore architectures (cf. 6.1.5).

Power consumption and temperature management have become a major concern for high performance processor design. We are pursuing research to migitigate temperature issues on single-chip parallel processors (cf. 6.1.3).

### *6.1.1. Null blocks management on the memory hierarchy*

**Participants:** Julien Dusser, André Seznec.

It has been observed that some applications manipulate large amounts of null data. Moreover these zero data often exhibit high spatial locality. On some applications more than 20% of the data accesses concern null data blocks. Representing a null block in a cache on a standard cache line is clearly a waste of resources.

We propose the Zero-Content Augmented cache, the ZCA cache [32]. A ZCA cache consists of a conventional cache augmented with a specialized cache for memorizing null blocks, the Zero-Content cache or ZC cache. In the ZC cache, the data block is represented by its address tag and a validity bit. Moreover, as null blocks generally exhibit high spatial locality, several null blocks can be associated with a single address tag in the ZC cache. For instance, a ZC cache mapping 32MB of zero 64-byte lines uses less than 80KB of storage. Decompression of a null block is very simple, therefore read access time on the ZCA cache is in the same range as on a conventional cache. On applications manipulating large amount of null data blocks, such a ZC cache allows to significantly reduce the miss rate and memory traffic, and therefore to increase performance for a small hardware overhead.

Moreover, we are now studying a new hardware compression scheme exploiting the high rate of null blocks in memory to increase the useful main memory size as well as increasing the useful bandwidth from memory to processors.

### *6.1.2. Quality of service on multicores*

**Participants:** Pierre Michaud, André Seznec.

The presence of shared caches in current multicore processors may generate a lot of performance variability when several applications execute simultaneously. For the programmer of an application with quality-of-service goals, this performance variability may lead to a very pessimistic tuning. To solve this problem, there must be a way for the programmer to define a reasonable performance target and make sure that the actual performance is greater than or close to the target. We propose that the performance target be defined as the performance measured when each core runs a copy of the application, which we call self-performance. We have characterized self-performance and shown how the shared-cache replacement policy can be modified for self-performance to be meaningful [34].

### *6.1.3. Mitigating temperature impact through activity migration for fast sequential execution*

**Participant:** Pierre Michaud.

On each new technology generation, miniaturization permits putting twice as many computing cores on the same silicon area, potentially doubling the processor performance. However, if sequential execution is not accelerated at the same time, Amdahl's law will eventually limit the actual performance. Hence it will be beneficial to have asymmetric multicores where some cores are specialized for fast sequential execution. This specialization may be achieved by architectural means, but it may also be achieved by specializing transistors, voltage, and clock frequency. In the latter case, one of the main constraints is that the power consumption of fast cores is not increased across technology generations. Yet this implies that the instantaneous heat flux in fast cores be potentially doubled on each new generation. High instantaneous heat fluxs can be tolerated by doing periodic activity migration. This requires to double the number of fast cores on each new generation, even though only a single fast core can be used at a given time. To keep the chip temperature below the limit, the migration interval must be divided approximately by four on each new generation. We have shown that this will eventually decrease the apparent level-2 cache size, and we propose to tackle this problem by preparing a certain number of cores before they become active [33].

### *6.1.4. Online compression of cache-filtered address traces*

**Participant:** Pierre Michaud.

Trace-driven simulation is potentially much faster than cycle-accurate simulation. However, one drawback is the large amount of storage that may be necessary to store traces. Trace compression techniques are useful for decreasing the storage space requirement. But the compression ratio of existing trace compressors is limited because they implement lossless compression. We propose two new methods for compressing cache-filtered address traces. The first method, bytesort, is a lossless compression method that achieves high compression ratios on cache-filtered address traces. The second method is a lossy one, based on the concept of phase. We have combined these two methods in a trace compressor called ATC. Our experimental results show that ATC gives high compression ratio while keeping the memory-locality characteristics of the original trace.

### 6.1.5. Scheduling issues on a heterogeneous single ISA multicore

**Participants:** Robert Guziolowski, André Seznec.

Single ISA multicores have become mainstream in general purpose computing. These multicores generally replicate a standard processor. However, this approach struggles with relatively high temperature dissipation and relatively high power consumption. Using multicores featuring different ISAs on distinct processors is often considered to be more power effective, but suffers from difficult software issues.

In 2007, we studied scheduling issues for multiprogrammed workloads on such heterogeneous platforms. This year we continued this work, but targetting multithreaded parallel workloads. As a benchmark suite, the ALPBench (http://www.cs.uiuc.edu/alp/alpbench was adopted. In particular, we are targeting workloads that mix several parallel applications running concurently.

### 6.1.6. Branch prediction and instruction fetch

**Participants:** Pierre Michaud, André Seznec.

Branch prediction feeds a speculative execution processor core with instructions. Branch mispredictions are inevitable and have negative effects on performance and energy consumption. With the advent of highly accurate conditional branch predictors,[14], non-conditional branch instructions are gaining importance.

We have addressed the prediction of procedure returns. On modern processors, procedure returns are predicted through a return address stack (RAS). The overwhelming majority of the return mispredictions are due to RAS overflows and/or overwriting the top entries of the RAS on a mispredicted path. These sources of misprediction were addressed by previously proposed speculative return address stacks. However, the remaining misprediction rate of these RAS designs is still significant when compared to state-of-the-art conditional predictors.

We have presented [21] two low-cost corruption detectors for return address stack (RAS) predictors. They respectectively detect RAS overflows and wrong path corruption with 100 % coverage. As a consequence, when such a corruption is detected, another source can be used for predicting the return. On processors featuring a branch target buffer (BTB), this BTB can be used as a free backup predictor for predicting returns when corruption is detected.

This study was done in collaboration with Hans Vandierendonck from University of Ghent.

### 6.1.7. Exploiting confidence in SMT processors

**Participants:** Pierre Michaud, André Seznec.

Balancing resource usage among the threads is one of the main issues on SMT processor [58]. Resource usage objectives may vary from delivering the maximum total performance to achieving maximum performance on a high priority thread while still delivering some performance on the other threads. Even when maximum throughput is the main objective, fairness is also often desired, i.e., one would like that each thread gets a reasonable part of the computing power.

In a previous study [16], we introduced the use of Speculative Instruction Window Weighting (SIWW) for optimizing power consumption on superscalar processor. SIWW can be also used for managing SMT fetch policy. SIWW estimates for each thread the amount of outstanding work in the processor pipeline. Fetch proceeds for the thread with the least amount of work left. With SIWW, each instruction is assigned a weight at fetch time. SIWW can then use and combine virtually any of the indicators that were previously proposed for guiding the instruction fetch policy (number of inflight instruction instructions, number of low confidence branchs, number of predicted cache misses, ..). Therefore, SIWW is an *approach to designing SMT fetch policies*, rather than a particular fetch policy.

Moreover, SIWW can be adapted to achieve different objectives such as maximizing the overall performance but also ensuring high quality of service for a high priority thread while maximizing performance on the other thread.

This study was done in collaboration with Hans Vandierendonck from University of Ghent.

## 6.2. Compilers and software environment for high performance embedded or special purpose architectures

**Keywords:** *compilation*, *optimization platform*, *performance debugging*, *thread extraction*.

**Participants:** François Bodin, Florence Dru, Khaled Ibrahim, Christophe Levointurier, Sylvain Leroy, Guillaume Papauré, Eric Petit, Erven Rohou, Robin Schmutz, André Seznec.

### *6.2.1. Compilers and software environment for high performance embedded or special purpose architectures*

**Participants:** François Bodin, Eric Petit, Guillaume Papauré, Robin Schmutz.

The new trend in computer architecture is to design multi-core heterogeneous platforms using very specific hardware accelerators. Mapping an application on such a platform is a challenge. For these heterogeneous platforms, we are defining and developing the ASTEX approach (Automatic Speculative Thread EXtractor). ASTEX aims at the extraction of speculative threads for the different hardware components of a SoC. We apply ASTEX for automatically exploiting a GPU.

Highly demanding scientific applications are using specific supercomputing facilities. We are studying optimization strategies for specific architectures as well as new parallelization strategies for porting high end scientific applications on new non-standard platforms such as GPUs.

#### *6.2.1.1. Speculative thread extraction for SOCs with Astex*

**Participants:** François Bodin, Eric Petit, Robin Schmutz, Guillaume Papauré.

Systems on a chip (SoCs) are highly integrated architectures which combine multiple heterogeneous computing units. A main processor runs the application. Coprocessors which may feature their own memory, are used to speedup the processing of some parts of the application. A SoC implementing such configuration aims at exploiting each processing unit. Typically, coprocessors run computation intensive sections of an application. The design of an application for such a SoC begins with the partitioning of the applications in threads that are then mapped onto the computing units of the SoC. In ASTEX, Automatic Speculative Thread EXtractor we address the problem of partitioning C code into threads for heterogeneous SoCs. The same approach is used for out of core accelerator usage, like FPGA or GPU.

ASTEX first performs a profile measurement to identify hotpaths on the application. Based on the identified hotpaths, sets of possible speculative threads are then evaluated; the objective is to maximize the intersection between the detected threads code coverage and the capabilities of each co-processor. Since the threads are computed from a profile, speculation arises at two levels. At control flow level: the thread is a subset of the possible paths in the execution of the application program. At data dependency level: the data dependency between the threads and the main program must be preserved. After the hotpath detection, ASTEX builds an executable C version of the code with the speculative threads. At this step, the instrumented program is exercised against many different input data and ASTEX determines a third level of speculation: a speculative

memory usage model. The last step, according to execution results, evaluates the characteristics of the potential threads and refines the selection if needed. Due to speculative nature of the threads, the data structures used by the thread must be copied in the local memory of the coprocessor. The speculative thread must test its validity and returns an error on any misspeculation: control flow, data dependency and memory usage.

This year we manage the technological transfer of ASTEX to the industry. The program is now fully automatic for a very large set of benchmarks and the effective results in output of Astex match the awaited one. Other developments concerned the use of ASTEX, mostly with GPUs.

*6.2.1.2. Automatically exploiting GPU with ASTEX*

**Participants:** Eric Petit, François Bodin.

Because of their high potential computing power, Graphical Processing Units (GPU) look very attractive to speed up programs, even if they are difficult to program. Porting code to the GPU is generally done manually. Our objective is to define and develop programming tools able to exploit GPUs in the context of general programming. GPU usage as a coprocessor is a convenient paradigm of automatically exploiting hardware accelerator.

ASTEX is used to implement a dynamic analysis that detects speculative threads which contains computing kernels with a potential speedup on the GPU. However recognizing these kernels is not sufficient. The effective performance of GPUs as hardware accelerators on general-purpose applications is highly dependent on the communication overhead between the main memory and the GPU memory. Optimizing this communication is an error prone process. The programmer must decide when to prefetch, update or download the remote data while ensuring that the GPU data are coherent when the remote procedure call is performed on the GPU. We are studying an automatic technique for inserting software data prefetching and upload for compute intensive kernels remotely executed on a GPU. The first technique we propose [8] relies on an hybrid approach that mixes speculative data collected via the thread analysis environment ASTEX and usual static program analysis.

In 2008, we designed a more accurate and efficient solution, also based on hybrid analysis. Our implementation works on C source codes and deals with pointer aliasing issues.

### *6.2.2. Compiler optimizations*

**Participants:** François Bodin, Guillaume Papauré, Damien Fétis.

High performance necessitates the compiler to be efficient on managing parallelism at every level, i.e., source code as well as assembly level.

*6.2.2.1. Data Locality Analysis of Parallel C Programs*

**Participants:** François Bodin, Guillaume Papauré.

OpenMP programs are composed of sequential and parallel code sections. The performance of each section can be impacted by the data layout in memory. This impact is particularly important on multicores. Cache block sharing or invalidations may severely degrade performance.

In order to tackle this issue, we have defined and developed a software prototype profiling tool to detect at runtime implicit data redistributions among threads in OpenMP programs. Instrumentation is added on source code. For each thread, our prototype computes at runtime the sections of the arrays accessed by the thread. The resulting analysis is a directed acyclic graph (DAG) representing the data accessed by the threads at execution, and the dynamic redistribution of data from thread to thread are the edges of this DAG (figure 1). The DAGs can be used to detect data sharing patterns. The knowledge of these patterns can then be used by the application developers to identify performance bottlenecks associated with implicit data redistribution.

*6.2.2.2. Backend optimizations: Sofan for the Terops project*

**Participants:** François Bodin, Damien Fétis.

In the context of the ApeNEXT project [1], CAPS has developed over the last few years SOFAN (Software Optimizer for ApeNEXT). This software optimizer attempts to explore different back end optimization strategies for ApeNEXT applications. In 2007, a production version of SOFAN was delivered to the users of APEnext.

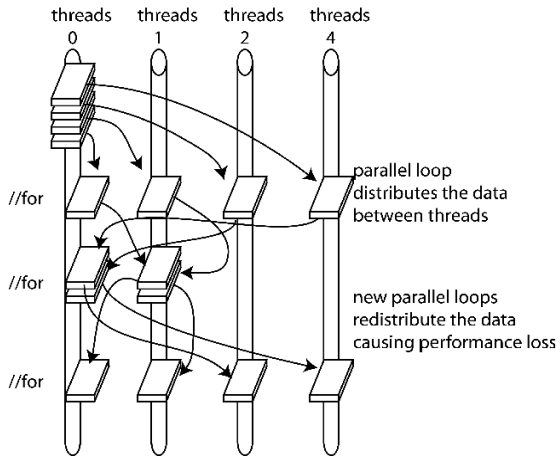


*Figure 1. DAG representation of data locality and movements.*

In the context of the Ter@ops project, SOFAN has been retargeted towards one of the accelerators of the Ter@ops machine, the Thomson FIRE EVO coprocessor. FIRE EVO is SIMD coprocessor with an array of 16 VLIW processing units. A gcc code generator was developed by the Alchemy EPI for this accelerator. SOFAN was adapted to realise the control flow and data flow analysis of SIMD assembly code. It also optimizes the VLIW Processing Units resources utilisation.

### *6.2.3. Enabling high performance applications on emerging architectures*

**Participants:** François Bodin, Florence Dru, Khaled Ibrahim, André Seznec.

#### *6.2.3.1. Porting highly demanding applications on hardware accelerators*

**Participants:** François Bodin, Khaled Ibrahim, André Seznec.

Simulation of Lattice QCD is a challenging computational problem that conventionally requires building special supercomputing machines. One of the objectives of the ANR QCDNext project is to explore architectural alternatives, including hardware accelerators, to build cost effective machine for simulating the Lattice QCD. We explored the use of GPUs [20], the Cell broadband engine [25], [19], as well as other general-purpose processors [26] to implement the most time consuming kernel of the computation.

Studying Lattice QCD computation on GPUs [20], we found that the granularity of the thread of execution has a great impact on performance. For the Lattice QCD, threads of finer granularity usually perform better because more resources are allocated per thread of execution. On the other hand, fine-granularity thread assignment may lead to different computations to perform. We proposed multiple code transformations to make the computations more homogeneous for all the fine-grained threads, thus improving the performance on GPUs.

In our study of Lattice QCD on the Cell broadband engine [25], we devised a novel technique to efficiently vectorize the code on the Cell synergetic processing elements. In this technique, we fuse the data accessed from different sites of the lattice to form data units that can be treated efficiently with SIMD instructions. We also explored multiple implementations with different memory access patterns to find the most suitable method for direct memory access (DMA) management. We were able to reduce the pressure on the memory bandwidth by removing redundancy and we suggested memory access technique that improves the contiguity of data access.

We explored multiple code optimizations on general-purpose architectures [26]; e.g., Intel x86 and Intel Itanium. The performance achieved on the Cell broadband engine is at least an order of magnitude better than the performance on the studied general-purpose processors.

*6.2.3.2. H.264 SVC decoder for multicore architectures*

**Participants:** François Bodin, Damien Fétis.

The goal of the Scalim@ges project is to study and implement scalable video coding/decoding chain. This new encoding technology aims at providing a solution to the exponential growth in the volume of broadcasted content, the multiplication of reception platforms, and the diversity of the means of transport and broadcasting.

Video decoding is a computationnally demanding domain. In this project, CAPS is in charge of analyzing the potential parallelism in the H.264 SVC decoder and then of proposing and implementing a solution for parallel decoding on multicore architectures. The analysis is performed through profiling an available implementation, the JSVM (Joint Scalable Video Model) software.

## 6.3. Around processor virtualization

**Participants:** François Bodin, Christophe Levointurier, Sylvain Leroy, Erven Rohou, André Seznec.

The usage of the Java language has been generalized in the past few years. Applications are now very large and are deployed on many different platforms, since they are highly portable. However ensuring code quality maintenance and code security on those applications is challenging. To address these issues, we are defining a refactoring platform for Java. Java has popularized the distribution of software through bytecodes. Functional portability is the main argument for such a usage of bytecodes. With the new diversity of multicore platforms, functional, but also performance portability will become the major issue in the next 10 years. We have initiated a research effort to efficiently compile towards bytecodes.

### *6.3.1. Analysis and transformation of Java codes*

**Participants:** François Bodin, Sylvain Leroy, Christophe Levointurier.

All along its lifetime, an application software evolves. Development rules that were initially defined are often progressively ignored or forgotten. Development rules may even evolve. Therefore one generally observes deterioration of the quality of the code. In particular, ignoring some design rules may affect the code security or its overall performance. To avoid this deterioration, automatic refactoring has been proposed [40]. Code is automatically transformed through enforcing development rules.

Within the Serenitec project, we are developing a framework for automatic refactoring of Java codes. Our framework has been first directed towards the audit of code rules. It can analyze large applications featuring a million java code lines or more. Automatic java to java transformations will be addressed later.

One of our objectives is to analyse Java web applications security through an automatic process. Nowadays static code analysis methods exist to detect invalid pointers, code injections or possible data buffer overflows. Those failures are commonly used in successful attacks of web applications. However in many cases, the tests produce lots of false positive alerts. This renders the use of these tests impractical in large application development. Automatic discrimination between real security breaches and false positive alerts is a major issue. We are addressing this issue through a new system built on top of our automatic refactoring framework. This system is based on complex pattern detection derived from the SWARP approach [9] coupled with a case database. The result accuracy is improved through the use of case-based reasoning to discriminate against false positives.

### *6.3.2. Performance portability through virtualization*

**Participants:** Erven Rohou, André Seznec.

Applications have a much longer lifetime than hardware. It will become increasingly important to be able to run applications on architectures with a higher degree of parallelism than they were designed for, and with different kinds of processors.

Virtualization can be of a great help to provide functional and performance portability on the new multicore systems. The final code generation occurs at run time, mapping and scheduling of computations can be performed across all available processing nodes, independently from their underlying architectures.

In the fall 2008, we have initiated a research action on combine processor virtualization and split compilation to address the problem of application portability, in terms of both functionality and performance.

## 6.4. WCET estimation

**Participants:** Jean-François Deverge, Damien Hardy, Thomas Piquet, Isabelle Puaut.

Predicting the amount of resources required by embedded software is of prime importance for verifying that the system will fulfill its real-time and resource constraints. A particularly important point in hard real-time embedded systems is to predict the Worst-Case Execution Times (WCETs) of tasks, so that it can be proven that task temporal constraints (typically, deadlines) will be met. Our research concerns methods for obtaining automatically upper bounds of the execution times of applications on a given hardware. A particular focus is put on hardware-level analysis (static analysis based on timing models) and compiler-directed schemes aimed at augmenting software predictability.

In 2008, our new results concern the analysis and/or software control of the memory hierarchy (instruction and data caches, cache hierarchies, static on-chip memory, virtual memory).

### *6.4.1. WCET analysis of data accesses*

**Participants:** Jean-François Deverge, Isabelle Puaut.

*6.4.1.1. Static timing analysis of data caches.*

Comparatively to the worst-case timing behavior for instruction caches, which has been extensively studied, the worst-case timing behavior for data and unified caches has been the subject of a lower number of studies. In particular, most previous approaches have focused on predictable memory access patterns. In [17] we have developed a new data cache analysis method that supports arbitrary memory access patterns, supports multiple replacement policies (LRU, PLRU, MRU) and applies to data caches, instruction caches and unified caches. The tightness of the method has been demonstrated for a StrongARM-110 processor for different cache architectures (see [17] for details).

*6.4.1.2. Software-control of static on-chip memories (scratchpad memories).*

An alternative to caches for on-chip storage is scratchpad memory. Scratchpad memories are small on-chip static RAMs that are mapped onto the address space of the processor at a predefined address range. Their inherent predictability have made them popular in real-time systems. Significant effort has been invested in developing efficient allocation techniques for scratchpad memories. However, except [55], all these techniques aim to reduce the average execution time (ACET) of programs using memory access profiles. In [17] we have proposed a solution for WCET-oriented allocation of data in scratchpad memories. The allocation problem (selection of the variables to be loaded into scratchpad memory and selection of load points) is formulated using Integer Linear Programming (ILP). One property of our allocation strategy is that the granularity of placement of memory transfers can be parametrized (function or basic block boundaries). This flexibility allows to trade-off between the complexity of allocation and its quality.

### *6.4.2. Predictable virtual memory for real-time applications*

**Participants:** Damien Hardy, Isabelle Puaut.

There is a need for using virtual memory in real-time applications: using virtual addressing provides isolation between concurrent processes; in addition, paging allows the execution of applications whose size is larger than main memory capacity, which is useful in embedded systems where main memory is expensive and thus scarce. However, virtual memory is generally avoided when developing real-time and embedded applications due to predictability issues.

We have previously proposed in [51] a compiler approach to introduce a predictable form of paging of *code* pages, in which page-in and page-out points are selected at compile-time. The problem under study was formulated as a graph coloring problem, as in register allocation within compilers. The work presented in [51] was improved in two directions: support for paging of *data* ; improvement of the quality of page allocation, using an Integer Linear Programming (ILP) formulation instead of graph coloring. We have demonstrated in [23] that the ILP formulation outperforms our previous approach with regard to page allocation quality, with a still reasonable page allocation time even for the biggest applications.

#### *6.4.3. Timing analysis of multi-level caches*

**Participants:** Damien Hardy, Thomas Piquet, Isabelle Puaut.

With the advent of increasingly complex hardware in realtime embedded systems (processors with performance enhancing features such as pipelines, cache hierarchy, multiple cores), many processors now have a set-associative L2 cache. Thus, there is a need for considering cache hierarchies when validating the temporal behavior of real-time systems, in particular when estimating tasks WCETs. In [24], we propose a safe static instruction cache analysis method for multi-level non-inclusive caches. The proposed method is experimented on medium-size and large programs. We show that the method is reasonably tight. We further show that in all cases WCET estimations are much tighter when considering the cache hierarchy than when considering only the L1 cache. An evaluation of the analysis time is conducted, demonstrating that analyzing the cache hierarchy has a reasonable computation time.

#### *6.4.4. Impact of cache replacement policy on the tightness of WCET estimation*

**Participants:** Damien Hardy, Thomas Piquet, Isabelle Puaut.

Recently Reineke et. al have proposed in [52] new theoretical results about the predictability of cache replacement policies, through the introduction of metrics (*evict* - eviction distance, *mls* - minimal life span) for different replacement policies. In [27], we have integrated these results in a static WCET estimation tool in order to quantify the impact of cache replacement policy (LRU/PLRU/Random) on the tightness of WCET estimation, for medium-size benchmarks. As expected, the LRU replacement policy can be analyzed more tightly than the PRLU replacement policy, which itself can be analyzed more tightly than the Random replacement policy. However, when the loops in the applications are small enough, there are no difference between the tightness of the analyses. One can also notice than the higher the cache associativity, the larger the difference between the tightness of the analyses.

# 7. Contracts and Grants with Industry

## 7.1. Research grant from Intel

**Participants:** Julien Dusser, André Seznec.

The researches on content conscious cache management (cf. 6.1.1), and on branch prediction (cf. 6.1.6) are partially supported by the Intel company through a research grant.

## 7.2. Start-up

**Participants:** François Bodin, André Seznec.

The collaboration has been pursued in 2008 with the start-up company CAPS Entreprise that was created in 2003 by members of the research team. This collaboration addresses topics such as very high performance code generation for complex processors (IA64 for instance) and compilation for ASIP.

## 7.3. QCDNext

**Participants:** François Bodin, Khaled Ibrahim, André Seznec.

The QCDNext ("programme blanc of ANR") combines the efforts of physicists, computer scientists and electrical engineers to propose the architecture of a low cost next generation computer system for highly demanding scientific applications. This ANR funded project is strongly related with the ApeNEXT collaboration.

## 7.4. Sceptre project

**Participants:** François Bodin, Robin Schmutz.

The goal of the project is to develop a toolkit for helping the implementation of multimedia algorithms on a reconfigurable multiprocessor network. In particular, this toolkit should help to explore and analyze hardware / software tradeoffs. The goal is to drastically reduce application port time and to obtain flexible realizations over a family of algorithms, thus increasing the lifespan of each development.

This project is funded by the "Pôle de compétitivité Minalogic".

## 7.5. Ter@ops project

**Participants:** François Bodin, Damien Fétis.

Ter@ops aims at defining and developing a large-scale embedded multi-core architectures. In this project, CAPS adapts the backend optimizer SOFAN to a SIMD architecture part of the Ter@ops project.

This project is funded by the "Pôle de compétitivité SYSTEMATIC".

## 7.6. Scalimages

**Participants:** François Bodin, Florence Dru.

The goal of this project is to study and implement scalable video encoding. This new encoding technology aims at providing a solution to the exponential growth in the volume of broadcasted content, the multiplication of reception platforms, and the diversity of the means of transport and broadcasting. In this project, CAPS is working on code optimization techniques.

This project is funded by the "Pôle de compétitivité Images et réseaux".

## 7.7. Mascotte

**Participants:** Isabelle Puaut, Thomas Piquet.

MasCotTE (http://www.projet-mascotte.org/) is an acronym for "MAîtriSe et COnTrôle des Temps d'Exécution" (Estimation and control of execution times). MasCotTE is funded by the Predit program of the ANR.

The aim of MasCotTE is to design the methods, techniques and tools required for controlling the execution times of automotive embedded real-time software (through static analysis and/or testing). Emphasis is put on the study of the impact of performance enhancing features on the predictability of embedded software. The project defines some guidelines on how to use such performance enhancing features in a predictable manner.

## 7.8. PARA

**Participants:** François Bodin, Khaled Ibrahim.

The objective of the PARA project is to study and develop optimization techniques in order to fully exploit all kind of parallelism found in modern computer architectures. This is done by associating different public and private research communities (application developers, compilation and operating system experts and system conceptors).

PARA is funded by the ANR program "Calcul Intensif et Grilles de Calcul".

## 7.9. POPS

**Participants:** François Bodin, Khaled Ibrahim, Guillaume Papauré.

POPS “Pour une nouvelle génération de serveurs et d’applications intensives á l’échelle du PetaFlops” is a project of the Pôle de compétitivité Systematics. The partners of the project are BULL, Caps enterprise, CS Systèmes d’information, EDF, ESI, Eurodecision, Medit, NewPhenix, Resonate, CEA DAM, CEA LIST, Ecole centrale de Paris, IFP, INRIA, INT, Université d’Evry, Université de Paris Sud, Université de Versailles St Quentin. The project aims at building supercomputers achieving Petaflop effective performance range.

In the project, we study programming environment to exploit the hybrid multicore architecture.

## 7.10. Serenitec: SEcurity analysis and Refactoring ENvironment for Internet TEChnology

**Participants:** François Bodin, Christophe Levointurier, Sylvain Leroy.

Serenitec aims at analyzing and improving security of Java Web applications. To achieve its goals, the project mixes a set of techniques from static program analysis, case based reasoning and refactoring techniques. Security analysis are based on the work of the Open Web Application Security Project. To validate the techniques, large web analysis will be used (500 kloc to 1 Mloc).

In this project, CAPS studies basic analysis and refactoring techniques for Java codes. Serenitec is a project of the Pôle de compétitivité Images et Réseaux. It is funded by the Region Bretagne and Rennes Métropole. Partners of this project are Silicom-AQL, Caps Entreprise and Irisa/INRIA (prime).

## 7.11. GaLogic

**Participants:** François Bodin, Guillaume Papauré, Isabelle Puaut.

GaLogiC is a RTNL contract between STMicroelectronics, VERIMAG and IRISA. It proposes to integrate three technologies: the worst-case execution time analysis for individual software components, the management of real-time in a context of application control, and the programming of basic components. The part of our team consists in the characterization of the worst case execution-time in each software component in order to generate predictable code.

# 8. Other Grants and Activities

## 8.1. NoEs

**Participants:** François Bodin, Pierre Michaud, Erven Rohou, André Seznec.

- F. Bodin, P. Michaud, A. Seznec and E. Rohou are members of European Network of Excellence HiPEAC2. HiPEAC2 addresses the design and implementation of high-performance commodity computing devices in the 10+ year horizon, covering both the processor design, the optimising compiler infrastructure, and the evaluation of upcoming applications made possible by the increased computing power of future devices.

## 8.2. IP-Fet European project Sarc

**Participants:** Julien Dusser, Robert Guziolowski, Pierre Michaud, Eric Petit, André Seznec.

SARC is an integrated IP-FET project concerned with long term research in advanced computer architecture http://www.sarc-ip.org/. It focuses on a systematic scalable approach to systems design ranging from small energy critical embedded systems right up to large scale networked data servers.

The CAPS team is involved in the microarchitecture research, including temperature management and memory hierarchy management, and the compiler research.

# 9. Dissemination

## 9.1. Scientific community animation

- Pierre Michaud was a member of the organization committee of the 2008 CEA-EDF-INRIA Computing Summer school.
- Pierre Michaud has been a member of the program committees of MuCoCoS 2008 and CMP-MSI 2008.
- Isabelle Puaut has been a member of program committee of ECRTS 2008 (20th Euromicro Conference on Real-Time Systems), WCET08 (8th Workshop on WCET analysis, held in conjunction with ECRTS08), RTNS 2008 (16th International Conference on Real-Time and Network Systems), RTCSA 2008 (14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications), the embedded systems track of HPCC07 (International Conference on High Performance Computing and Communications), 2008 and 2009 editions of the Real-Time and (Networked) Embedded Systems track of ETFA (13th and 14th IEE international conferences on Emerging Technologies and Factory Automation), and CARI 2008 (9e Colloque Africain sur la Recherche en Informatique et en Mathématiques Appliquées). I. Puaut is member of the editorial board of Interstices (french on-line resources dedicated to the discovery of research in computer science, http://interstices.info/).
- Isabelle Puaut is the program committee chair of ECRTS 2009 (21th Euromicro Conference on Real-Time Systems), to be held in Dublin, July 2009.
- Isabelle Puaut was the chair of the work-in-progress session of the 29th IEEE Real-Time Systems Symposium, held in Barcelona, Spain, december 2002.
- Isabelle Puaut was the general chair of RTNS 2008 (16th International Conference on Real-Time and Network systems), Rennes, France, october 2008. Jean-François Deverge was the chair of the 2nd Junior Researcher Workshop on Real-Time Computing, in conjunction with the 16th International Conference on Real-Time and Network Systems (RTNS'08)
- André Seznec is a member of ISPASS'09, MULTIPROG'09 and MEDEA' 08 program comittees. He is a member of the editorial board of the HiPEAC Transactions (Transactions on High-Performance Embedded Architectures and Compilers).
- André Seznec is the general co-chair of HiPEAC 2009 conference (Paphos, Cyprus, january 2009)

## 9.2. University teaching and responsabilities

- F. Bodin, A. Seznec, I. Puaut and E. Rohou are teaching computer architecture and compilation in the master of research in computer sciences at University of Rennes I.
- I. Puaut teaches operating systems, real-time systems and real-time programming in the master degree of computer science of the University of Rennes I. She teaches real-time systems in the BSc degree *Embedded automotive systems*.
- I. Puaut is responsible of the 1st year of Master in computer science at University of Rennes I.
- Pierre Michaud is teaching computer architecture at the engineering degree in computer science IFSIC, university of Rennes 1.

## 9.3. Workshops, seminars, invitations, visitors

- Khaled Ibrahim presented an invited seminar at University of Toronto in April 2008 entitled “Parallel computing from specialty to ubiquity,”.
- A. Seznec has presented a seminar on branch prediction at the ARM company in Cambridge in december 2008 entitled “All you will never have wanted to know on branch prediction”.
- Pierre Michaud has given a seminar entitled "The implications of energetic and thermal constraints on current and future processors" at the 2008 CEA-EDF-INRIA Computing Summer school.

## 9.4. Miscelleanous

- I. Puaut is member of the advisory board of the fundation M. Métivier (http://www.fondation-metivier.org).
- J. Lenfant is a member of “académie des sciences et des technologies”.
- A. Seznec is an elected member of the scientific comittee of INRIA.
- CAPS is a member of the “pôle de compétitivité System@tic”, the “pôle de compétitivité réseau image”, and the “pôle de compétitivité Minalogic”.

# 10. Bibliography

## Major publications by the team in recent years

[1] F. BELLETTI, S. F. SCHIFANO, R. TRIPICCIONE, F. BODIN, P. BOUCAUD, J. MICHELI, O. PENE, N. CABIBBO, S. DE LUCA, A. LONARDO, D. ROSSETTI, P. VICINI, M. LUKYANOV, L. MORIN, N. PASCHEDAG, H. SIMMA, V. MORENAS, D. PLEITER, F. RAPUANO. *Computing for LQCD: ApeNEXT*, in "Computing in Science and Engineering", vol. 8, n$^{o}$ 1, 2006, p. 18–29.

[2] F. BODIN, A. SEZNEC. *Skewed associativity improves performance and enhances predictability*, in "IEEE Transactions on Computers", May 1997.

[3] M. FOWLER, K. BECK, J. BRANT, W. OPDYKE, D. ROBERTS. *Refactoring: improving the design of existing code*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[4] K. HEYDEMANN, F. BODIN, P. KNIJNENBURG, L. MORIN. *UFC : a Global Tradeoff Strategy for Loop Unrolling for VLIW Architectures*, in "CPC’2003", February 2003, p. 59-70.

[5] P. MICHAUD. *Exploiting the Cache Capacity of a Single-chip Multi-core Processor with Execution Migration*, in "Proceedings of the 10th International Conference on High-Performance Computer Architecture (HPCA-10 2004)", IEEE Computer Society, January 2004.

[6] P. MICHAUD. *A PPM-like, Tag-based Predictor*, in "Journal of Instruction Level Parallelism", April 2005, http://www.jilp.org/vol7.

[7] P. MICHAUD, Y. SAZEIDES, A. SEZNEC, T. CONSTANTINOU, D. FETIS. *A study of thread migration in temperature-constrained multi-cores*, in "ACM Transactions on Architecture and Code Optimization", vol. 4, n$^{o}$ 2, 2007, 9.

[8] E. PETIT, F. BODIN, R. DOLBEAU. *An Hybrid Data Transfer Optimization for GPU*, in "Compilers for Parallel Computers (CPC2007)", July 2007.

[9] G. POKAM, S. BIHAN, J. SIMONNET, F. BODIN. *SWARP: a retargetable preprocessor for multimedia instructions*, in "Concurrency and Computation: Practice and Experience", vol. 16, n$^{o}$ 2, 2004, p. 303–318.

[10] G. POKAM, O. ROCHECOUSTE, A. SEZNEC, F. BODIN. *Speculative Software Management of Datapath-width for Energy Optimization*, in "proceedings of the Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'04)", June 2004.

[11] A. SEZNEC, S. FELIX, V. KRISHNAN, Y. SAZEIDES. *Design trade-offs on the EV8 branch predictor*, in "Proceedings of the 29th International Symposium on Computer Architecture (IEEE-ACM), Anchorage", May 2002.

[12] A. SEZNEC, N. SENDRIER. *HAVEGE: a user-level software heuristic for generating empirically strong random numbers*, in "ACM Transactions on Modeling and Computer Systems", October 2003.

[13] A. SEZNEC. *Analysis of the O-GEHL branch predictor*, in "Proceedings of the 32nd Annual International Symposium on Computer Architecture", June 2005.

[14] A. SEZNEC. *The L-TAGE Branch Predictor*, in "Journal of Instruction Level Parallelism", May 2007, http://www.jilp.org/vol9.

[15] A. SEZNEC, E. TOULLEC, O. ROCHECOUSTE. *Register Write Specialization Register Read Specialization: A Path to Complexity Effective of Wide Issue Superscalar Processors*, in "Proceedings of the 35th International Symposium on Microarchitecture (IEEE-ACM), Istanbul", November 2002.

[16] H. VANDIERENDONCK, A. SEZNEC. *Fetch gating control through speculative instruction window weighting*, in "proceedings of the 2nd HIPEAC conference, LNCS 4327", January 2007, p. 120-135.

# Year Publications

## Doctoral Dissertations and Habilitation Theses

[17] J.-F. DEVERGE. *Contributions à l'analyse du comportement temporel de la hiérarchie mémoire pour l'estimation de pire temps d'exécution*, Ph. D. Thesis, Université de Rennes I, July 2008.

[18] T. PIQUET. *Gestion consciente du contenu de la hiérarchie mémoire.*, Ph. D. Thesis, Université de Rennes I, May 2008.

## Articles in International Peer-Reviewed Journal

[19] K. Z. IBRAHIM, F. BODIN. *Efficient SIMDization and Data Management of the Lattice QCD Computation on the Cell Broadband Engine*, in "Journal of Scientific computing: special issue on high performance computing on Cell B.E. processors", 2008.

[20] K. Z. IBRAHIM, F. BODIN, O. PENE. *Fine-grained Parallelization of Lattice QCD Kernel Routine on GPUs*, in "Journal of Parallel and Distributed Computing", vol. 68, n$^{o}$ 10, 2008, p. 1350-1359.

[21] H. VANDIERENDONCK, A. SEZNEC. *Speculative Return Address Stack Management Revisited*, in "ACM Transactions on Architecture and Code Optimization", to appear, 2008.

[22] R. WILHELM, J. ENGBLOM, A. ERMEDAHL, N. HOLSTI, S. THESING, D. WHALLEY, G. BERNAT, C. FERDINAND, R. HECKMANN, F. MUELLER, I. PUAUT, P. PUSCHNER, J. STASCHULAT, P. STENSTRÖM. *The Determination of Worst-Case Execution Times—Overview of the Methods and Survey of Tools*, in "ACM Transactions on Embedded Computing Systems (TECS)", vol. 7, n° 3, April 2008.

### International Peer-Reviewed Conference/Proceedings

[23] D. HARDY, I. PUAUT. *Predictable Code and Data Paging for Real Time Systems*, in "Proc. of the 20th Euromicro Conference on Real-Time Systems, Prague, Czech Republic", July 2008, p. 266–275.

[24] D. HARDY, I. PUAUT. *WCET analysis of multi-level non-inclusive set-associative instruction caches*, in "Proc. of the 29th IEEE Real-Time Systems Symposium, Barcelona, Spain", December 2008.

[25] K. Z. IBRAHIM, F. BODIN. *Implementing Wilson-Dirac Operator on the Cell Broadband Engine*, in "The 22nd ACM/SIGARCH International Conference on Supercomputing", Jun. 2008, p. 4–14.

[26] K. Z. IBRAHIM, J. JAEGER, Z. LIU, L. POUCHET, P. LESNICKI, L. DJOUDI, D. BARTHOU, F. BODIN, C. EISENBEIS, G. GROSDIDIER, O. PENE, P. ROUDEAU. *Simulation of the Lattice QCD and Technological Trends in Computation*, in "arXiv:0808.0391, 2008, also appears in the 14th International Workshop on Compilers for Parallel Computers (CPC'09)".

[27] A. JUNIER, D. HARDY, I. PUAUT. *Impact of instruction cache replacement policy on the tightness of WCET estimation*, in "Proc. of the 2nd Junior Researcher Workshop on Real-Time Computing, in conjunction to RTNS 2008, Rennes, France", October 2008, p. 5–8.

[28] M. TAWK, K. Z. IBRAHIM, S. NIAR. *Multi-granularity Sampling for Simulating Concurrent Heterogeneous Applications*, in "The ACM/SIGBED international Conference on Compilers Architecture and Synthesis for Embedded Systems (CASES'08)", Oct. 2008.

### Books or Proceedings Editing

[29] G. BUTTAZZO, P. MINET, I. PUAUT (editors). *Proceedings of the 16th International Conference on Real-Time and Network Systems (RTNS'08)*, October 2008, http://hal.inria.fr/RTNS2008/en/.

[30] J.-F. DEVERGE (editor). *Proceedings of the 2nd Junior Researcher Workshop on Real-Time Computing, in conjunction with the 16th International Conference on Real-Time and Network Systems (RTNS'08)*, October 2008.

[31] I. PUAUT (editor). *Proceedings of the Work in Progress session of the 29th IEEE Real-Time Systems Symposium*, December 2008.

### Research Reports

[32] J. DUSSER, T. PIQUET, A. SEZNEC. *Zero-Content Augmented Caches*, Research Report, n° RR-6705, INRIA, 2008, http://hal.inria.fr/inria-00337742/en/.

[33] P. MICHAUD. *Periodic activity migration for fast sequential execution in future heterogeneous multicore processors*, Technical report, n° PI-1909, IRISA, November 2008.

[34] P. MICHAUD. *Replacement policies for shared caches on symmetric multicores : a programmer-centric point of view*, Technical report, n$^{o}$ PI-1908, IRISA, November 2008.

## References in notes

[35] A. BESZÉDES, R. FERENC, T. GYIMÓTHY, A. DOLEN, K. KARSISTO. *Survey of Code-size reduction methods*, in "ACM Computing Survey", vol. 35, n$^{o}$ 3, September 2003, p. 223-267.

[36] P.-Y. CHANG, E. HAO, Y. N. PATT. *Target Prediction for Indirect Jumps*, in "Proceedings of the 24th Annual International Symposium on Computer Architecture", may 1997.

[37] R. S. CHAPPELL, J. STARK, S. P. KIM, S. K. REINHARDT, Y. N. PATT. *Simultaneous Subordinate Microthreading (SSMT)*, in "Proceedings of the 26th Annual International Symposium on Computer Architecture", May 1999.

[38] K. CHOW, Y. WU. *Feedback-Directed Selection and Characterization of Compiler Optimizations*, in "Proc.2nd workshop on Feedback-Directed Optimization", November 1999.

[39] S. DEBRAY, W. EVANS. *Profile-Guided Code Compression*, in "ACM PLDI'02", vol. 37, n$^{o}$ 5, 2002.

[40] M. FOWLER, K. BECK, J. BRANT, W. OPDYKE, D. ROBERTS. *Refactoring: improving the design of existing code*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[41] L. HAMMOND, ET AL.. *The Stanford Hydra CMP*, in "IEEE Micro", vol. 20, n$^{o}$ 2, March 2000.

[42] C.-H. HSU, U. KREMER. *IPERF: A Framework for Automatic Construction of Performance Prediction Models*, in "In Workshop on Profile and Feedback-Directed Compilation (PFDC)", October 1998.

[43] M. JACOME, G. DE VECIANA. *Design challenges for new application specific processors*, in "IEEE Design and Test of Computers", vol. 17, n$^{o}$ 2, 2000, p. 40–50.

[44] R. E. KESSLER. *The Alpha 21264 microprocessor*, in "IEEE Micro", vol. 19, n$^{o}$ 2, 1999.

[45] T. KISUKI, P. KNIJNENBURG, M. O'BOYLE, H. WIJSHOFF. *Iterative compilation in program optimization*, in "Compilers for Parallel Computers 2000", 2000, p. 35–44.

[46] P. KULKARNI, W. ZHAO, H. MOON, K. CHO, D. WHALLEY, J. DAVIDSON, M. W. BAILEY, Y. PAEK, K. GALLIVAN. *Finding Effective Optimization Phase Sequences*, in "LCTES'03", 2003, p. 12-23.

[47] A.-C. LAI, C. FIDE, B. FALSAFI. *Dead-Block Prediction & Dead-Block Correlating Prefetchers*, in "Proceedings of the 28th Annual International Symposium on Computer Architecture Computer Architecture News", June 2001.

[48] C.-K. LUK. *Tolerating memory latency through software-controlled pre-execution in simultaneous multithreading processors*, in "Proceedings of the 28th annual international symposium on Computer architecture", june 2001.

[49] J. MELLOR-CRUMMEY, R. FOWLER, D. WHALLEY. *Tools for application-oriented performance tuning*, in "Proceedings of the 15th international conference on Supercomputing", ACM Press, 2001, p. 154–165, http://doi.acm.org/10.1145/377792.377826.

[50] S. PALACHARLA, N. P. JOUPPI, J. E. SMITH. *Complexity-Effective Superscalar Processors*, in "Proceedings of the 24th Annual International Symposium on Computer Architecture", 1997.

[51] I. PUAUT, D. HARDY. *Predictable paging in real-time systems: a compiler approach*, in "Proc. of the 19th Euromicro Conference on Real-Time Systems, Pisa, Italy", July 2007, p. 169–178.

[52] J. REINEKE, D. GRUND, C. BERG, R. WILHELM. *Timing predictability of cache replacement policies*, in "Real-Time Systems Journal", vol. 37, n$^{o}$ 2, 2007, p. 99–122.

[53] S. K. REINHARDT, S. MUKHERJEE. *Transient fault detection via simultaneous multithreading*, in "Proceedings of the International Symposium on Computer Architecture", 2000.

[54] C. ROCHANGE, P. SAINRAT. *Difficulties in Computing the WCET for Processors with Speculative Execution*, in "2nd Intl. Workshop on Worst Case Execution Time Analysis", June 2002.

[55] V. SUHENDRA, T. MITRA, A. ROYCHOUDHURY, T. CHEN. *WCET Centric Data Allocation to Scratchpad Memory*, in "Real Time Systems Symposium 05", December 2005.

[56] C. TICE, S. GRAHAM. *Key Instructions: Solving the Code Location Problem for Optimized Code*, 2000, Tech. Report 164, Compaq Systems Research Center, Palo Alto, CA.

[57] V. TIWARI, S. MALIK, A. WOLFE. *Compilation techniques for low energy: An overview*, in "Proceedings of the IEEE Symposium on Low Power Electronics", October 1994.

[58] D. TULLSEN, S. EGGERS, H. LEVY. *Simultaneous multithreading : maximising on-chip parallelism*, in "22nd Annual International Symposium on Computer Architecture", June 1995, p. 392-403.

[59] S.-H. YANG, M. POWELL, B. FALSAFI, K. ROY, T. VIJAYKUMAR. *An Integrated Circuit/Architecture Approach to Reducing Leakage in Deep-Submicron High Performance I-caches*, in "Proceedings of the International Symposium on High Performance Computer Architecture", January 2001.

[60] C. ZHANG, F. VAHID, W. NAJJAR. *A Highly Configurable Cache Architecture for Embedded Systems*, in "Proceedings of the 30th International Symposium on Computer Architecture", June 2003.

[61] W. ZHAO, B. CAI, D. WHALLEY, M. W. BAILEY, R. VAN ENGELEN, X. YUAN, J. D. HISER, J. W. DAVIDSON, K. GALLIVAN, D. L. JONES. *VISTA: a system for interactive code improvement*, in "Proceedings of the joint conference on Languages, compilers and tools for embedded systems", ACM Press, 2002, p. 155–164, http://doi.acm.org/10.1145/513829.513857.

[62] H. ZHOU, T. M. CONTE. *Code Size Efficiency in Global Scheduling for VLIW/EPIC Style Embedded Processors*, in "The 6th Annual Workshop on Interaction between Compilers and Computer Architectures (INTERACT-6) held in conjunction with HPCA-8", Feburary 2002.

[63] C. ZILLES, J. EMER, G. SOHI. *The use of multithreading for exception handling*, in "Proceedings of the International Symposium on Microarchitecture", 1999.