INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team caps

Compilation, architectures des processeurs superscalaires et spécialisés

Rennes

THEME COM

Activity Report

2004

Table of contents

1. Team

Scientific head
André Seznec [Research Director Inria]

Administrative Assistant
Evelyne Livache [TR Inria]

Inria staff members
Pierre Michaud [Research scientist]

Academic staff
Henri-Pierre Charles [Assistant Professor, University of Versailles-Saint-Quentin, on partial secondment at Inria, till 08/31/04]
François Bodin [Professor, University of Rennes 1]
Jacques Lenfant [Professor, University of Rennes 1]
Isabelle Puaut [Professor, University of Rennes 1, from 10/01/04]

Project technical staff Inria
Antoine Colin [till 08/31/04]
Thierry Lafage [till 07/31/04]
Laurent Morin [from 02/01/04]

Inria Postdoctoral fellows
Julio Hernandez [till 02/28/04]

Ph.D. students
Arnaud Alexis [Inria allocation, from 10/01/04]
Amaury Darsch [Inria allocation]
Jean-François Deverge [MENRT allocation, from 10/01/04]
Assia Djabelkhir [French-Algerian allocation]
Romain Dolbeau [Teaching Assistant, till 08/31/04]
Damien Fétis [MENRT allocation, from 11/01/04]
Antony Fraboulet [Inria allocation]
Karine Heydemann [Allocation couplée, till 08/31/04, teaching assistant from 09/01/04]
Gilles Pokam [Inria allocation, till 07/31/04]
Olivier Rochecouste [Inria allocation]
Eric Toullec [MENRT allocation till 09/30/04, teaching assistant from 09/01/04]
Thomas Piquet [Inria allocation, from 10/01/04]
Max Lukyanov [coadvised with DESY, Berlin]

2. 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, and processor simulation techniques. Our objective is to enable the end user to exploit a significant fraction of this theoretical performance while still masking the underlying hardware complexity.

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[3][12], and we are still refining them (cf. 6.1.1, 6.1.3). We have also started research in order to reduce branch misprediction penalties [16][13] (cf. 6.1.3). We also aim at reducing the hardware costs of implementing wide-issue superscalar processors [7][15] (cf. 6.1.4) while we pursue researches on thread level parallelism on a single chip [29] (cf. 6.1.6, 6.1.5).

Instruction Set Architecture (ISA) is where the software meets the hardware. On the one hand the compiler must optimize the codes to take advantage of the micro-architecture. On the other hand, the ISA must be designed in such a way that the compiler can "understand" performance issues. We are exploring the adequation of dynamic execution on embedded applications (cf. 6.2.1). In light of the current knowledge on microarchitecture, we are trying to define the characteristics that a general purpose ISA should feature, to allow efficient and cost-effective implementation (cf. 6.2.2). Researches in architecture and code optimizations require to experiment and to evaluate new architectural ideas. We are defining several simulation frameworks that permits to validate both embedded statically scheduled processor architectures (ABSCISS, cf. 5.2) and out-of-order execution for general-purpose architectures (IATO, cf. 6.2.3). Most scientific applications make use of vector-like computations, while the behavior of the memory hierarchies on such computations is implementation dependent. Most ISAs feature multimedia instructions. These instructions are generally not well handled by compilers. We have designed a C source code optimizer that automatically makes use of these instructions [23]. Power consumption becomes a more and more important issue, in particular with embedded systems. We are studying how the compiler can dynamically reconfigure the architecture to optimize the power consumption/performance tradeoff (cf. 6.2.4). Capitalizing on CAPS know-how in ISAs, compilers and simulators, we participate in the design of a cryptographic processor (PACCMAN, cf. 6.2.5).

Some performance issues must be handled at a higher level than the direct interface that lies between the hardware and the instruction set. For heterogeneous SOCs (System On a Chip) featuring special purpose hardware and one or more execution cores, we are exploring thread extraction for the different hardware components (cf. 6.3.1). Code size is often an issue with embedded systems. We are exploring tradeoffs leveraging code compression and interpretation (cf. 6.3.3).

Finally, we use our knowledge of modern microarchitecture to participate in the definition of an unpredictable random number generator (HAVEGE, cf. 6.5).

Our research is partially supported by industry (Intel, STMicroelectronics, Thomson). We also participate in several institutionally funded projects (OPPIDUM PACMANN, MEDEA+ MESA, ministry of industry EPICEA, ACI Securité UNIHAVEGE). Among our main partners in these institutionally funded projects, let us cite Bull, EADS and STMicroelectonics.

Some of the research prototypes developed by the project during the past few years are currently being transferred to industry through the CAPS Entreprise start-up (cf. 7.5).

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: *Superscalar processor*, *branch prediction*, *memory hierarchy*, *speculative execution*.

The advance of integration technology permits the design of wide-issue superscalar processors with a high clock frequency. The gap between the main memory access time and the clock cycle is an ever increasing bottleneck. Moreover the product (pipeline depth)x(issue width) is also increasing. Effective performance is therefore more and more dependent on the management of control and data dependencies. As it stands, the two main challenges for uniprocessor architects to achieve ultimate performance remain 1) hiding the memory hierarchy latency and 2) hiding (breaking) control and data dependencies. However, computer architects must also address the new challenges associated with the power consumption and the design complexity. Finally, performance predictability will also become a major issue in the near future for both computer architects and software designers.

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 (20 cycles on the Intel Pentium 4) 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 [51]. The challenge for the computer architects will be to reduce the complexity of these hardware mechanisms in order to enable simpler implementation. The challenge is also 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 [48][12]. Both branch directions and targets (even indirect jump targets) [38] are predicted. Most of these predictors exploit either local or global branch history. The accuracy of the prediction seems to be reaching a plateau. New prediction paradigms exploiting other information sources are probably needed to allow new major prediction accuracy gains.

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 [55].

While the 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 is often synonym 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 processor

Keywords: *multicore processor*.

It has now become possible to implement hardware thread parallelism on a single chip. The advantage of single chip hardware parallelism over a conventional parallel machine is that the performance cost of communicating between the different tasks is reduced. Multicores or CMP (chip multiprocessors) as well as SMT (Simultaneous Multithreading) processors are emerging on the server and workstation market. On a multicore, 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 resource sharing concerns most resources. Key issues concerning SMT / multicore processors are the performance on sequential application and the design complexity. This will determine the extent to which they can be used as universal computing components.

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 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, multi-programmed systems, scientific computing, ...

The straightforward way to implement task parallelism is to have multiple distinct processors. Current technology is able to put several hundred millions of transistors on a single die. This allows to integrate several high-performance computing cores on the same chip, and presents several advantages, not the least of which are a reduced communication latency between cores, and a potentially higher communication bandwidth.

Multicore processors are already available for some embedded applications, and IBM has introduced the dual-core POWER4 last year for work-stations and servers [59]. Most high-end processor families have a multicore on their roadmap for this decade. The first multicores will feature only two cores and should appear within the next 2–3 years (IBM Power5, HP Mako, Intel Montecito, Sun Ultrasparc IV and Gemini,...). Second generation multicores with a higher number of cores should appear later in the decade.

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 [62]. Some SMT processors are already available, like the Intel Pentium 4 [53]. However, the main difficulty for the design of SMT processors is the design of a very wide issue superscalar processor. Though the SMT and multicore approaches both exploit task parallelism, they are orthogonal, and future multicores may feature SMT cores.

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 [56] by redundant execution), or more performance. A few ideas have been recently proposed to speed-up sequential execution, like for instance speculative threads [43], exception handling [68], helper threads for branch prediction [39], helper threads for memory prefetching [52], 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 are proposing many new challenges to the hardware and compiler research community. Code optimization does not just mean performance optimization, but may mean best performance/code size tradeoff, guaranteeing real time constraints, or getting best power consumption/performance tradeoffs. New software environments, with new demands are needed. For instance, performance/power consumption/code size tuning debugging tools are needed. Since wide spectrum of hardware platforms have to be explored, retargetable compiler infrastructures and retargetable ISA simulators are also needed.

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 a fixed performance level at the smallest possible cost or the smallest power consumption.

Therefore embedded processors are presenting many new challenges [46] 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. First, the “common case” paradigm using a set of representative benchmarks (e.g. SPEC2000) for general-purpose processor systems 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. Second, execution time is not always the only and ultimate criteria. In many cases, execution time may be less important than memory size or power consumption. Third, binary compatibility, while often important, does not overcome the matching between system cost, application functionalities and time-to-market constraints.

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 [67][41] is a major issue in many applications. For small micro-controllers, “the smallest code, the fastest” 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 [5]. To master code size, code compression techniques [37] 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 new hardware which behavior is fundamentally difficult to predict [57]. 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 [61]. 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 tradeoff performance against power consumption [65][64].

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 eachother [40][50]. 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 [45][54][66]. An associated challenge is to preserve the link between aggressively optimized low level code and the source code [60]. As an alternative (or a complement) to performance tuning, automatic iterative compilation techniques [49] 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 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 [11] as well as fast processor simulation are key issues to support design exploration. Compiled simulation [1] 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.

4. Application Domains

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

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, ...). Our research activity implies the development of software prototypes (cf. 5.1, 6.3)

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 present here **ABSCISS** and **HAVEGE**, two softwares developed by the team. **ABSCISS** has been transfered to the CAPS Entreprise start-up and **HAVEGE** is freely distributed for non-commercial use.

5.2. ABSCISS

Keywords: *retargetable processor simulation platform.*

Participant: François Bodin.

ABSCISS (Assembly-Based System for Compiled Instruction-Set Simulation) is a retargetable system for high-speed instruction-set simulation. This tool automatically generates compiled simulators from an assembly program and a description of the target processor instruction set architecture.

As of today, it targets various statically scheduled RISC and VLIW processors. Within this kind of architectures, the simulators generated by ABSCISS are cycle-accurate. Caches can also be simulated by interfacing to an external module. Other architectures can be simulated at a functional level, that is, only the behavior of the program will be simulated.

ABSCISS is optimized both for high speed simulation and fast simulator generation (also providing flexibility through an API for extension "plug ins" written in C++ or Python).

Status : Registered with APP Number IDDN.FR.001.190016.000.S.P.2002.000.10600. ABSCISS is industrialized by CAPS entreprise (cf. 7.5).

5.3. HAVEGE

Keywords: *Unpredictable random number generator.*

Participants: Julio Hernandez, André Seznec.

Contact : André Seznec
Status : Registered with APP Number IDDN.FR.001.500017.001.S.P.2001.000.10000. Available for tests and use in non-commercial software.

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 targets 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.

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*, *simultaneous multi-threading*.

Participants: Damien Fétis, Romain Dolbeau, Antony Fraboulet, Pierre Michaud, Thomas Piquet, Olivier Rochecouste, André Seznec, Eric Toullec.

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 [3][12]. We are still refining, analyzing and exploring new usage of skewed associative caches (cf. 6.1.1). We are also pursuing researches on reducing branch misprediction penalties [16][13] (cf. 6.1.3). We are also exploring new directions in branch predictions. We are also trying to reduce the hardware costs of implementing wide-issue superscalar processors [7][15] (cf. 6.1.4) while continuing ongoing research on thread level parallelism on a single chip [29] (cf. 6.1.6, 6.1.5).

6.1.1. Concurrent support of multiple page sizes on a TLB

Participant: André Seznec.

Some architecture definitions (e.g. Alpha) allow the use of multiple virtual page sizes even for a single process. Unfortunately, on current set-associative TLBs (Translation Lookaside Buffers), pages with different sizes can not coexist. Thus, processors supporting multiple page sizes implement fully-associative TLBs.

We have shown that the skewed-associative TLB can accommodate the concurrent use of multiple page sizes within a single process [24]. This permits to envision either medium size L1 TLBs or very large L2 TLBs supporting multiple page sizes.

6.1.2. Content conscious management of the memory hierarchy

Participants: Thomas Piquet, André Seznec.

Performance on modern processor architecture highly depends on the memory hierarchy behavior. Complex memory hierarchy involves up to three levels of caches. The usual way of managing memory blocks is to fetch a block from memory on a miss and to store it in all the intermediate levels (L3 cache, L2 cache, L1 cache). Prefetching, i.e. predicting in advances the blocks that will be used, is also used to avoid misses or to decrease miss penalty whenever possible. However, the management of cache contents is far from optimal and apart set replacement policies, poor management of cache content is currently used. While there have been a few studies targeting temporal and spatial locality on L1 caches, the conscious management of L2 and L3 caches contents has not been yet addressed.

We are initiating a study on such a conscious management of L2 and L3 caches. We want to focus on maintaining the "useful" blocks in the whole memory hierarchy as a complementary approach to prefetching.

6.1.3. Branch prediction and instruction sequencing

Participants: Anthony Fraboulet, Pierre Michaud, André Seznec.

6.1.3.1. Ahead pipelining instruction address generator

On a N-way issue superscalar processor, the front end instruction fetch engine must deliver instructions to the execution core at a sustained rate higher than N instructions per cycle. The instruction address generator/predictor (IAG) has to predict the instruction flow at an even higher rate.

Very complex IAGs featuring different predictors for jumps, returns, conditional and unconditional branches and complex logic are used. Usually, the IAG uses information (branch histories, fetch addresses, ...) available at a cycle to predict the next fetch address(es). Unfortunately, a complex IAG cannot deliver a prediction within a short cycle. Therefore, processors rely on a hierarchy of IAGs with increasing accuracies but also increasing latencies: the accurate but slow IAG is used to correct the fast, but less accurate IAG.

As an alternative to the use of a hierarchy of IAGs, it is possible to initiate the instruction address generation several cycles ahead of its use [16]. We have explored in details such an ahead pipelined IAG and shown that, even when the instruction address generation is (partially) initiated five cycles ahead of its use, it is possible to reach approximately the same prediction accuracy as the one of a conventional one-block ahead complex IAG [13]. Moreover when the instruction address generation and instruction fetch are decoupled, one can implement an even more efficient IAG. This allows either to further enhance the instruction fetch bandwidth or to run the IAG at half of the frequency of the rest of the processor core, thus saving power [18].

6.1.3.2. Global history branch predictors

Research on global history predictors has been continued in two directions.

First we have continued the researches initiated in 2003 on improving the promising concept of perceptron predictors [47]. This research has generated two new branch predictor schemes, the MAC-RHSP (for Multiply-ACcumulate Redundant History Skewed Perceptron Predictor) [36] and the O-GEHL (for Optimized GEometric History Length) predictor [34].The advantages of these predictors over the initial perceptron predictor proposal are first a substantially improved accuracy and, second a more realistic hardware implementation cost.

Second we have investigated a top-down approach for inventing new predictor schemes. The method most often used for inventing branch predictors is to start from a known predictor and try to improve it, based on some new intuition. However, when this method fails to decrease the mispredict rate, it is difficult to analyze the reasons for this failure, precisely because the mispredict rate does not change. We have proposed a new approach, which consists in first defining a model of ideal predictor, and then introducing successive degradations corresponding to hardware constraints, until we obtain a realistic predictor. On each degradation, it is possible to quantify the loss, analyse the reasons for it, and sometimes propose remedies. We applied the method on the family of tag-based predictors derived from PPM predictor (partial path matching) and, with the help of the method, we were able to obtain new insights and propose improvements to a tag-based predictor [35][28].

6.1.4. Mastering hardware complexity on wide-issue supercalar processors

Participants: André Seznec, Olivier Rochecouste, Eric Toullec.

With the continuous shrinking of transistor size, processor designers are facing new difficulties to achieve high clock frequency. In wide issue superscalar processors, register file read time, wake up and selection logic traversal delay, bypass network transit delay, and their respective power consumption constitute such major difficulties.

The general-purpose ISAs currently in use feature a single logical register file. This central view has also been adopted for the hardware implementation of dynamically scheduled superscalar processors. Until now, the following unwritten rule has always been applied: *every general-purpose physical register can be the source or the result of any instruction executed on any integer functional unit.*

In [15], we showed that transgressing this rule can be advantageous. Indeed, the set of physical registers can be divided into distinct subsets that are only read-connected (resp. write-connected) with a subset of the entries (resp. a subset of the exits) of the functional units. Therefore, the number of write and read ports on each *individual physical register* and the overall complexity of the physical register file, the bypass network and the wake-up logic is decreased. This proposed hybrid approach is referred to as WSRS (for Write Specialization Read Specialization).

We are currently exploring instruction allocation policies on functional unit clusters. We also explore the benefits of integrating the simultaneous multithreading paradigm into our WSRS architecture.

6.1.5. Resource sharing on single chip

Participants: Romain Dolbeau, André Seznec.

As the increase of issue on superscalar processors bring diminishing returns, thread parallelism with a single chip is becoming a reality. In the past few years, both SMT (Simultaneous MultiThreading) [62] and CMP (Chip MultiProcessor) [43] approaches were first investigated by academics and are now implemented by the industry. In some sense, CMP and SMT represent two extreme design points. We are exploring possible intermediate design points for on-chip thread parallelism in terms of design complexity and hardware sharing. The CASH parallel processor (for CMP And SMT Hybrid) retains resource sharing à la SMT when such a sharing can be made non-critical for implementation, but resource splitting a la CMP whenever resource sharing leads to a superlinear increase of the implementation hardware complexity. CASH does not exploit the complete dynamic sharing of resources enabled on SMT. But it outperforms a similar CMP on a multiprogrammed workload, as well as on a uniprocess workload. Our CASH architecture [21] shows that there exists intermediate design points between CMP and SMT.

6.1.6. Exploiting multicore processors with single process workload

Participant: Pierre Michaud.

It has now become possible to put several hundreds of millions of transistors on a single chip. This is large enough to put several high-performance computing cores on the same chip.

Multicore or CMPs (Chip MultiProcessors) should provide large performance gains on multi-programmed workloads and parallel applications. However, CMPs are not intended to speed-up sequential applications, and the first CMPs will be confined to the server market and certain embedded applications. We are searching ways to speed-up sequential execution on a CMP, in order to make CMPs "universal" computing devices. Indeed, integration on a single-chip decreases communication latency and permits a high communication bandwidth. This offers new possibilities.

Our first proposition is based on "execution migration" [29], i.e., the possibility for a sequential task to migrate quickly from one core to another. The goal of execution migration is to take advantage of the overall on-chip level-2 cache capacity. We introduced the affinity algorithm, a method for controlling migrations that can be implemented in hardware. We have shown that it is possible for an application to take advantage of the overall level-2 cache capacity if the application exhibits a property which we call "splittability". Our study shows that "splittability" is a rather frequent property of programs.

6.1.7. Tackling temperature issues

Participants: Pierre Michaud, Damien Fetis.

Now power density has reached levels that make temperature a constraint that affects the microarchitecture [42][58]. Microarchitectural solutions to the temperature problem can be divided in two approaches: one tries to decrease the electric power, the other tackles temperature hot spots directly. Whereas the former approach has been explored for several years, direct solutions to temperature hot spots have been explored only recently [58][44].

Execution migration is also a possible way to control temperature by spreading the electric power on a larger area. We are currently exploring this possibility, in a collaboration with the team of professor Yiannakis Sazeides from the University of Cyprus. One of the goals of this research is to determine the extent to which execution migration can solve the temperature problem. Another goal is to propose architectural supports permitting efficient migrations. For this research, we are currently developing a temperature model derived from first principles. We plan to use this model to study several questions concerning temperature on multicore chips, among which execution migration.

6.2. Interaction on architecture and compilers

Keywords: *Architecture*, *ISA (Instruction Set Architecture)*, *code optrimization*, *compilation*, *multimedia instruction set*, *power consumption*, *processor simulation*.

Participants: François Bodin, Antoine Colin, Amaury Darsch, Assia Djabelkhir, Thierry Lafage, Max Lukyanov, Gilles Pokam, Olivier Rochecouste, André Seznec.

The characteristics of the applications, the instruction set architecture and the compiler/architecture interaction have a significant impact on the effective performance that can be achieved by an application and/or the complexity of the hardware needed to achieve a predetermined performance level. We are exploring the adequation of dynamic execution on embedded applications (cf. 6.2.1). In the light of current microarchitecture knowledge, we are trying to define the characteristics that a general purpose ISA should feature to allow efficient and cost-effective implementation (cf. 6.2.2). Researches in architecture and code optimizations require to be able to experiment and to evaluate new architectural ideas. We are defining simulation frameworks to validate both embedded statically scheduled processor architectures (ABSCISS, cf. 5.2) and out-of-order execution general-purpose architectures (IATO, cf. 6.2.3). Most scientific applications make use of vector-like computations, while the behavior of the memory hierarchies on such computations is very implementation dependent. Most ISAs feature multimedia instructions. These instructions are generally not well handled by compilers. We have designed a C source code optimizer that automatically makes use of these instructions [23]. Power consumption becomes a more and more important issue, in particular on embedded systems. We are studying how the compiler can dynamically reconfigure the architecture to optimize the power consumption/performance tradeoff (cf. 6.2.4). Capitalizing on CAPS know-how in ISAs, compilers and simulators, we participate in the design of a cryptographic processor (PACCMAN, cf. 6.2.5).

6.2.1. Characterization of embedded application behaviors for decoupled architectures

Participants: Assia Djabelkhir, André Seznec.

Needs for performance on embedded applications will lead to the use of dynamic execution on embedded processors in the next few years. However, complete out-of-order superscalar cores are still expensive in terms of silicon area and power dissipation. We have studied the adequation of a more limited form of dynamic execution, namely decoupled architecture, to embedded applications.

Decoupled architecture is known to work very efficiently as long as the execution does not suffer from inter-processor dependencies causing some loss of decoupling, called LOD events. We have studied regularity of codes in terms of the LOD events that may occur. We have addressed three aspects of regularity: control regularity, control/memory dependency, and patterns of referencing memory data. Most of the kernels in MiBench, a set of benchmarks for embedded systems, will be amenable to efficient performance on a decoupled architecture [4].

6.2.2. Balancing instruction sets for modern microarchitectures

Participants: Olivier Rochecouste, André Seznec.

Considering the advancements achieved in the micro-architectural and technical fields, a number of criteria highlight that current general-purpose ISAs (Instruction Set Architecture) are no longer suited to modern microprocessors. For instance, on RISC ISAs, instruction source operands are read from the register file and results are written into it. The pressure exerted on this structure implies that the register file must provide enough access ports so that performance is not impaired. Therefore, constraining instructions to use a small fixed number of read and write operands, may lead to a design simplification of the register file. Furthermore, in pipelined architectures, some operations need less than one processor cycle to be executed. Thus, combining these operations in single instructions [63], may increase both the processor cycle utilization and the ILP (Instruction Level Parallelism). This work aims at defining a new general-purpose ISA that exploits these proposals in order to be in harmony with the state-of-the-art micro-architectures.

6.2.3. Simulation platform for an out-of-order execution IA64 microarchitecture

Participants: François Bodin, Amaury Darsch, André Seznec.

The ISA (Instruction Set Architecture) has an important impact on the effective implementation of processors. Recently HP and Intel have introduced a new 64-bit ISA for general-purpose systems, called the IA64 or IPF.

Unlike previous generation general purpose ISAs, the IA64 makes extensive use of predication and provides support for speculative execution (advance loads for instance). Unlike traditional ISAs, this new instruction set is very resistant to an out-of-order architecture, because of the resource size as well as the complexity of executing predicated instructions. Although the research community has started to study the execution of predicated instruction within an out-of-order core, no definitive solution has been adopted yet.

We are developing a software simulation platform called IATO, that is designed to provide the research community with a framework that permits to investigate the out-of-order execution of the IA64-like instruction sets. The framework is built around a set of libraries and applications programs that permit to analyze in details the structure of binary programs as well as emulating or simulating full binary executables.

IATO is currently distributed under GPL license (see http://www.irisa.fr/caps/projects/ArchiCompil/iato/).

Additionally, we have investigated a novel register management policy that is designed to operate smoothly with a fully predicated ISA. This new system is based on an intermediate representation called *Translation Register Buffer* or TRB. The TRB mechanism that translates a logical register into a physical register is shown to be effective when an instruction is canceled by a predicate [25].

6.2.4. Speculative Software Management of Datapath-width for Energy Optimization

Participants: Gilles Pokam, Olivier Rochecouste, François Bodin, André Seznec.

Many applications manipulate operands that appear at execution to be have narrow width. Unfortunately, due to high level programming, the compiler is often unable to detect this property. We proposed to augment the ISA with instructions directly exposing the datapath and the register widths to the compiler. Simple exception management allows this exposition to be only speculative[33].

In this way, we permit the software to speculatively accommodate the execution of a program on a narrower datapath-width in order to save energy. For this purpose, we introduce a novel register file organization, the *byte-slice* register file, which allows the width of the register file to be dynamically reconfigured, providing both static and dynamic energy savings.

By combining the advantages of the *byte-slice* register file with the advantages provided by clock-gating the datapath on a per-region basis, we showed that on our benchmark set, up to 17% of the datapath dynamic energy could be saved, while a 22% reduction of the datapath static energy was achieved.

6.2.5. The Paccman project

Participants: Thierry Lafage, François Bodin, Antoine Colin.

The PACCMAN project (*PlAteforme de Composants Cryptographiques pour Multi-Applications Nationales* cf. 7.3) aims at designing and producing a "French" cryptographic platform. This platform will be composed of a cryptography ASIP (*Application Specific Integrated Processor*) and a complete compiler toolchain (including debugger and profiler). The goal of PACCMAN platform is to improve the technological independence and the competitiveness of the French cryptography industry. The two main industrial application fields of this project are the implementation of secured PMR (Professional Mobile Radio) communications and high bandwidth VPNs (Virtual Private Networks).

The CAPS research group has in charge Inria's contribution to PACCMAN. This contribution is four-fold:

1. CAPS collaborates to the **specification of the instruction set architecture** (ISA) of the PACCMAN processor.
2. CAPS designs, implements and validates the **compiler toolchain** for the PACCMAN processor. This compiler toolchain is made of:

   – an ANSI-C compiler including both state-of-the-art and dedicated optimizing techniques,

- a software debugger,
- a software profiler.

3. CAPS designs, implements and validates a **cycle-accurate simulator** of the PACCMAN processor for:

   - functional validation of the compiler toolchain itself,
   - fine tuning dedicated optimizations implemented in the compiler,
   - pre-silicon validations of software applications.

4. CAPS produces an ISA reference manual and technical documentation for the compiler toolchain and the cycle-accurate simulator.

The PACCMAN compiler toolchain and the PACCMAN simulator integrate several technologies developed by CAPS: (i) a retargetable system for assembly language transformation and optimization (SALTO [11]), and (ii) a high speed compiled simulator generator (ABSCISS [1]).

The industrialization and maintenance of the PACCMAN development suite will be transfered to the start-up Caps Entreprise.

6.3. Compilers and software environment for high performance embedded processors

Keywords: *code compression*, *compilation*, *optimization platform*, *performance debugging*.

Participants: François Bodin, Henri-Pierre Charles, Karine Heydemann, Laurent Morin, Gilles Pokam.

Some performance issues must be handled at higher level than the direct interface between the hardware and the instruction set. For heterogeneous SOCs featuring special purpose hardware and one or more execution cores, we are exploring the speculative thread extraction for the different hardware components 6.3.1. Code size is often an issue on embedded systems. We are exploring tradeoffs based on code compression and interpretation (cf. 6.3.3).

6.3.1. Speculative thread extraction for SOCs

Participants: François Bodin, Laurent Morin.

Optimization of high performance applications is mainly based on parallelism extraction. Though the fine-grain parallelism is exploited by various optimizations (SIMD instructions, loop transformations, ...), the extraction of the coarse-grain parallelism is still performed by the programmer.

Systems On Chip (SoC), are highly integrated architectures that combine a programmable processor, memory and other specialized computational units on the same chip. For optimizing their design, one has to consider the mapping of the application on the architecture. Some code sections may require a dedicated component for some specific computation sections while other sections can be parallelized. A coarse grain parallelism analysis may enable an effective extraction of some execution traces - threads - for specialized units.

To achieve thread extraction, our approach focuses on two criteria: the computational and the memory transfer densities. Intensive computation areas are determined through an adaptive profiling of the control flow graph. The analysis is then refined by both static and dynamic computations of the memory accesses. The static analysis computes data dependencies and performs a selective memory access instrumentation. The thread is then defined by the trace execution and the memory mapping. The thread is then simulated by a speculative posix implementation of the thread extraction. During a thread execution, a speculative code handles the effective path taken by the program and all possible memory access misspredictions. This method provides a realistic global view of the optimizing potential of the thread.

This year has been dedicated to the design and the implementation of an infrastructure managing all the steps of the thread extraction, from the trace generation to the posix implementation

6.3.2. Hot paths computation

Participants: Gilles Pokam, François Bodin.

The effectivity of software optimization techniques is highly reliant on the runtime behavior of the program. A key issue is to identify program paths that are likely to achieve the greatest performance benefits at runtime. Several approaches have been proposed to address this problem. However, most of these methods are restricted to loops or procedures. This year, we have introduced a novel approach for representing and analyzing complete program paths. Unlike the whole-program path approach that relies on a DAG to represent program paths, our program trace is processed using *suffix-array*[]. This work serves as a basis for extracting threads for SOC (cf. 6.3.1).

6.3.3. Performance Code Size TradeOff

Participants: Karine Heydemann, François Bodin, Henri-Pierre Charles.

The design of an embedded system results from a tradeoff between hardware and software. Developers must achieve fast design while taking into account various constraints such as memory space, power consumption and application response time. Memory footprint is a strong constraint as it may directly impact the cost and the functionalities of the system.

For some designs, one would like to minimize the amount of memory space allocated to each program to allow more applications to fit in the device or to reduce the number of memory chips. One may also want to optimize the amount of memory for an embedded system design, i.e to find the exact quantity needed to allow the desired level of performance [22].

Two major techniques impact code size. On the one hand, optimizations improve performance (especially on architectures featuring instruction level parallelism) while increasing code size. On the other hand, code compression reduces code size while degrading performance.

Finding a global tradeoff between code size and performance consists in allowing code size increase on critical sections where it provides important performance returns, while saving code space on seldom executed code sections. This tradeoff concept is crucial in the compilation of embedded applications and is dependent on the target system and its applications. Enabling both optimizations and compressions on a single compilation scheme may allow to cover various needs. Furthermore being able to compute a good trade-off is of crucial important to avoid the hard and long manual search for parameters meeting the system design constraints.

We have investigated strategies for optimization under code size constraint [22]. We have also defined a software compiler driven compression scheme. We have proposed a selective compression strategy under performance constraint to reduce code size with control over the performance decrease [19]. We have also formalized the trade-off notion based on curves expressing the relation between code size and performance [19]. The trade-off point is the point where the trend reverses: less and less performance benefit for more and more code size increase. The combination of trade-off computation and effective optimization strategies under constraints provides good trade-offs [22].

We are currently working on applying this approach to optimizations where some involved parameters are antagonistic.

6.3.4. APEnext project

Participants: François Bodin, Max Lukyanov.

The apeNEXT is the latest generation of massively parallel supercomputers with a multi-TFlops performance dedicated for particle physics simulations. It is developed in the framework of the APE project which is carried out by the collaboration of INFN (Italy), DESY (Germany) and University of Paris-Sud (France). Following the single program multiple data (SPMD) programming model, where the nodes of the machine run in a slightly asyncronous mode, it represents an array of processing nodes, where each node is an independent, VLIW controlled ASIC implementing all functionalities including network.

Phase coupling in code generation is a well-known problem. It has been noted that the separation of the code generation phases typically does not lead to optimum performance. Though a number of different techniques was developed and used for many years, the interaction and ordering of the phases is still not well understood.

The software optimizer (SOFAN) has been implemented to serve as a test-bed for the exploration of different optimization strategies for the apeNEXT architecture, in particular the study of the phase coupling in the code generation. It also addresses the high irregularity of the apeNEXT architecture by realizing target dependent and state-of-the-art optimization techniques.

6.4. WCET analysis

Participants: Isabelle Puaut, Alexis Arnaud, Jean-François Deverge.

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 the framework of 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, compiler-directed schemes aimed at augmenting predictability).

This activity started in CAPS in october 2004 with the move of Isabelle Puaut from the ACES project-team to the CAPS project-team. A complete description of this research activity can be found in the ACES 2004 activity report.

6.5. HAVEGE: generating empirically strong random numbers

Keywords: *cryptography*, *security*, *unpredictable random number*.

Participants: Julio Hernandez, André Seznec.

HAVEGE (HArdware Volatile Entropy Gathering and Expansion) is a user-level software unpredictable random number generator for general-purpose computers that exploits the modifications of the internal volatile hardware states of a processor as a source of uncertainty.

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 which aim at improving performance: 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) [14] (cf. 5.3) 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.

Showing that HAVEGE-like softwares can be a source of unpredictable random numbers on most modern computing appliances is the objective of the UNIHAVEGE project (cf. 8.4).

This year, we have concentrated our research on the cryptanalysis of HAVEGE, i.e. finding weaknesses in the HAVEGE concept. No weakness was discovered during this study, but during this study, we developped a new test for random number generator [26] as well as a new efficient pseudo random number generator [27].

This research is done in cooperation with the Inria Rocquencourt CODES team (Nicolas Sendrier).

7. Contracts and Grants with Industry

7.1. MEDEA+ A-502 MESA (2000-2004)

Participants: François Bodin, Laurent Morin.

To meet the huge computational capabilities of the 100nm IC generation together with the exploding market needs, efficient design methods for multi-processor embedded systems architectures are needed. The MESA project aims at conceiving a flexible multi-processor design platform that offers tools for application domain analysis, reconfigurable IP blocks usage, communication protocol design and validation techniques. MESA involves industrial partners (Philips, Bull, Eads Telecom, STMicroelectronics and Alcatel), SMEs (ARM, CoWare, MetaSymbiose, PolySpace, CAPS entreprise) and public/academic institutions (EPUN/MCSE, IMEC, Inria, KU-Leuven, LIP6, TIMA).

CAPS contributions are described in 6.3.1 and 6.3.3.

7.2. Project EPICEA (2001-2004)

Participants: François Bodin, Henri-Pierre Charles, Amaury Darsch, André Seznec.

The EPICEA (Explicit Parallelism Instruction Computer Compiler Environment and Architecture) project is a collaboration between Inria team CAPS, University of Versailles Saint-Quentin, CEA and Bull funded by the ministry of Industry. The overall goal is to develop a software environment for scientific applications for architectures using the IA 64 ISA. Contribution from CAPS is described in 6.2.3.

7.3. The OPPIDUM Paccman project

Participants: Thierry Lafage, François Bodin, Antoine Colin.

The PACCMAN project (*PlAteforme de Composants Cryptographiques pour Multi-Applications Nationales*) aims at designing and producing a “French” cryptographic platform. This platform will be composed of a cryptography ASIP (*Application Specific Integrated Processor*) and a complete compiler toolchain (including debugger and profiler). The goal of PACCMAN platform is to improve the technological independence and the competitiveness of the French cryptography industry. The two main industrial application fields of this project are the implementation of secured PMR (Professional Mobile Radio) communications and high bandwidth VPNs (Virtual Private Networks).

The PACCMAN project is supported by the OPPIDUM research network and involves several industrial partners. These partners are: EADS-DSN (the world leader in digital PMR), MATRAnet (software for network security), Bull TrustWay (software and hardware components for network security), and Inria.

The contribution of CAPS project is described in 6.2.5.

7.4. Research grant from Intel

Participants: Eric Toullec, Antony Fraboulet, Thomas Piquet, André Seznec.

The researches on instruction fetch front ends (cf. 6.1.3), on register file structures (cf. 6.1.4) and on optimized cache management are supported by the Intel company through a research grant (Convention 4 01 C 0677 00 31308 06 1) and a material donation.

7.5. Start-up

Participants: François Bodin, Laurent Morin, Romain Dolbeau, Karine Heydemann, André Seznec.

The collaboration has been pursued in 2004 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.

8. Other Grants and Activities

8.1. Zenon cooperation

Participants: Pierre Michaud, André Seznec.

We are collaborating with professor Yiannakis Sazeides from the university of Cyprus in the study of execution migration as a means to control temperature on multicore processors. Travels and expenses are funded by the french ministery of foreign affairs in the context of Zenon, a bilateral action between France and Cyprus.

8.2. APEnext project

Participants: François Bodin, Max Lukyanov.

The apeNEXT is the latest generation of massively parallel supercomputers with a multi-TFlops performance dedicated for particle physics simulations. It is developed in the framework of the APE project which is carried out by the collaboration of INFN (Italy), DESY (Germany) and University of Paris-Sud (France). Following the single program multiple data (SPMD) programming model, where the nodes of the machine run in a slightly asyncronous mode, it represents an array of processing nodes, where each node is an independent, VLIW controlled ASIC implementing all functionalities including network.

8.3. NoE HiPeac

Participants: François Bodin, Pierre Michaud, André Seznec.

F. Bodin, P. Michaud and A. Seznec are members of European Network of Excellence HiPeac. HiPEAC 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.

8.4. ACI Sécurité UNIHAVEGE (2003-2006)

Participants: Julio Hernandez, André Seznec.

Researches on unpredictable random number generation are funded through the *ACI sécurité* project UNIHAVEGE. Main partners are CAPS team and CODES team from Inria Rocquencourt.

8.5. RTP CNRS Architecture et Compilation

Participants: Francois Bodin, Pierre Michaud, André Seznec.

CAPS members participate to RTP CNRS *Architecture et Compilation*:

- André Seznec is member of the steering committee.
- Pierre Michaud participates to specific action “Nouvelles technologies et nouveaux paradigmes d’architecture”.
- François Bodin participates to specific action “ Compilation pour les systèmes embarqués”.

9. Dissemination

9.1. Scientific community animation

- A. Seznec has been a member of the program committee of ISCA’31, Performance Computing). He is a member of the program committees of ISCA’31, HPCA’11, CARI’2004 and MASCOTS 2004. He was vice-chair of the program committee of Europar workshop on parallel architecture and was the program co-chair of ICS 2004 (International Conference on Supercomputing).
- F. Bodin is a member of the editorial board of TSI. He has been a member of the program committee of MEMOCODE’2004. He is also co-chair of the ACM SAC’05 conference track EMBS (track on embedded systems). F. Bodin participate to the program committee of the workshop Sympa’05. of TSI.

9.2. University teaching

F. Bodin and A. Seznec are teaching computer architecture and compilation at research master in computer sciences, at DIIC and DESS ISA at IFSIC, University of Rennes I.

9.3. Workshop, seminar, invitations

- A. Seznec has presented a seminar entitled “Revisiting the perceptron predictor” at Intel, Shrewsbury Massachussets in February 2004.
- P. Michaud has presented a seminar at the University of Cyprus entitled "Exploiting the cache capacity of a single-chip multicore processor with execution migration"
- F. Bodin made an invited presentation entitled “Compiler improvement: a computer-scientist’s view” in the workshop “ Block Courses on Lattice QCD”, at Orsay, France.

9.4. Miscelleanous

- F. Bodin is an elected member of the IFSIC Committee.
- F. Bodin is responsible of the Research Master in computer sciences at University of Rennes I and chairman for doctoral studies at Irisa.
- F. Bodin is vice-chairman of Ecole doctorale Matisse (http://www.irisa.fr/matisse).
- F. Bodin is member of the board of the fundation M. Métivier (http://www.fondation-metivier.org).
- F. Bodin is a member of the "Commission de spécialiste 27e" at Universite de Bretagne Sud, Université de Rennes 1 et Université de Versailles.
- F. Bodin is a scientific advisor for the company CAPS entreprise.
- Jacques Lenfant is a member of “académie des sciences et des technologies”.
- A. Seznec is an elected member of the evaluation committee of Inria.

10. Bibliography

Major publications by the team in recent years

[1] R. AMICEL. *Simulation de jeux d'instructions à hautes performances*, Ph. D. Thesis, University of Rennes 1, January 2003.

[2] R. AMICEL, F. BODIN. *A New System for High-Performance Cycle-Accurate Compiled Simulation*, in "5th International Workshop on Software and Compilers for Embedded Systems", may 2001.

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

[4] A. DJABELKHIR, A. SEZNEC. *Characterization of embedded applications for decoupled processor architecture*, in "Proceedings of the IEEE 6th Annual Workshop on Workload Characterization, Austin, TX", oct. 2003.

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

[6] T. LAFAGE. *Étude, réalisation et application d'une plate-forme de collecte de traces d'exécution de programmes*, PhD. Thesis, université de Rennes I, December 2000.

[7] P. MICHAUD, A. SEZNEC. *Data-flow prescheduling for large instruction windows in out-of-order processors*, in "7th International Conference on High Performance Computer Architecture", January 2001.

[8] P. MICHAUD, A. SEZNEC, R. UHLIG. *Trading conflict and capacity aliasing in conditional branch predictors*, in "Proceedings of the 24th International Symposium on Computer Architecture, Denver", IEEE-ACM (editor)., June 1997.

[9] A. MONSIFROT. *Utilisation du raisonnement à partir de cas et de l'apprentissage pour l'optimisation de code*, Ph. D. Thesis, Dec 2002.

[10] E. ROHOU, F. BODIN, C. EISENBEIS, A. SEZNEC. *Handling Global Constraints in Compiler Strategy*, in "International Journal of Parallel Programming", August 2000.

[11] E. ROHOU. *Infrastructures et stratégies de compilation pour parallélisme à grain fin*, PhD, University of Rennes I, November 1998.

[12] 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.

[13] A. SEZNEC, A. FRABOULET. *Effective ahead pipelining of instruction block address generation*, in "Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA-03)", June 9–11 2003, p. 241–252.

[14] 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.

[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] A. SEZNEC, S. JOURDAN, P. SAINRAT, P. MICHAUD. *Multiple-block ahead branch predictors*, in "Proceedings of the 7th conference on Architectural Support for Programming Languages and Operating Systems", October 1996.

Doctoral dissertations and Habilitation theses

[17] A. DARSCH. *L'exécution dans le désordre des jeux d'instructions prédiquées*, Ph. D. Thesis, University of Rennes I, December 2004.

[18] A. FRABOULET. *Génération des adresses des instructions pour les processeurs superscalaires fortement pipelinés*, Ph. D. Thesis, Université de Rennes I, Décembre 2004.

[19] K. HEYDEMANN. *Schéma de compilation global sous contraintes pour la recherche de compromis entre la taille d'un code et sa performance*, Ph. D. Thesis, University of Rennes I, December 2004.

[20] G. POKAM. *Techniques de compilation pour la gestion et l'optimisation de la consommation d'énergie des architectures VLIW*, Ph. D. Thesis, University of Rennes I, July 2004.

Articles in referred journals and book chapters

[21] R. DOLBEAU, A. SEZNEC. *CASH: Revisiting hardware sharing in single-chip parallel processor*, in "Journal of Instruction Level Parallelism", April 2004.

[22] K. HEYDEMANN, F. BODIN, P. KNIJNENBURG, L. MORIN. *UFS: a Global Trade-off Strategy for Loop Unrolling for VLIW Architectures*, in "To appear in Concurrency and Computation: Practice and Experience Journal", 2004.

[23] G. POKAM, S. BIHAN, S. SIMONNET, F. BODIN. *SWARP: a retargetable preprocessor for multimedia instructions*, in "Concurrency and Computation: Practice and Experience", vol. 16, n^{o} 2–3, February/March 2004, p. 303–318.

[24] A. SEZNEC. *Concurrent support of multiple page sizes on a skewed associative TLB*, in "IEEE Transactions on Computers", july 2004.

Publications in Conferences and Workshops

[25] A. DARSCH, A. SEZNEC. *IATO: A Flexible EPIC Simulation Environment*, in "Proceedings of the 16th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2004)", October 2004.

[26] J. C. HERNANDEZ, P. ISASI, A. SEZNEC. *On the design of state-of-the-art pseudorandom number generators*

by means of genetic programming, in "Proceedings of the 2004 IEEE Congress on Evolutionary Computation, Portland, Oregon", IEEE Press, 20-23 June 2004, p. 1510–1516.

[27] J. C. HERNÁNDEZ, J. M. SIERRA, A. SEZNEC. *The SAC Test: A New Randomness Test, with Some Applications to PRNG Analysis*, in "Proceedings of the International Conference Computational Science and Its Applications - ICCSA 2004, LNCS vol. 3043", April 2004, p. 960-967.

[28] P. MICHAUD. *A PPM-like, tag-based branch predictor*, in "Proceedings of the First Workshop ChampionShip Branch Prediction in conjunction with MICRO-37", December 2004.

[29] P. MICHAUD. *Exploiting the cache capacity of a single-chip multi-core processor with execution migration*, in "Tenth International Symposium on High-Performance Computer Architecture", Feb 2004.

[30] G. POKAM, F. BODIN. *An Offline Approach for Whole-Program Paths Analysis using Suffix Arrays*, in "Proceedings of the 17th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2004)", September 2004.

[31] G. POKAM, F. BODIN. *Exploring the energy-efficiency potential of a phase-based cache resizing scheme for embedded systems*, in "Proceedings of the 8th Annual Worskhop on Interaction between Compilers and Computer Architectures (INTERACT-8)", February 2004.

[32] G. POKAM, F. BODIN. *Understanding the Energy-Delay Tradeoff of ILP-based Compilation Techniques on a VLIW Architecture.*, in "Proceedings of the 11th Workshop on Compilers for Parallel Computers (CPC 2004)", July 2004.

[33] 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.

[34] A. SEZNEC. *The O-GEHL branch predictor*, in "Proceedings of the First Workshop ChampionShip Branch Prediction in conjunction with MICRO-37", December 2004.

Internal Reports

[35] P. MICHAUD. *Analysis of a tag-based branch predictor*, Technical report, n^o PI-1660, IRISA, November 2004.

[36] A. SEZNEC. *Revisiting the perceptron predictor*, Research report, IRISA, May 2004.

Bibliography in notes

[37] 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.

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

[39] R. S. CHAPPELL, J. STARK, S. P. KIM, S. K. REINHARDT, Y. N. PATT. *Simultaneous Subordinate Mi-*

crothreading (SSMT), in "Proceedings of the 26th Annual International Symposium on Computer Architecture", May 1999.

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

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

[42] S. GUNTHER, F. BINNS, D. CARMEAN, J. HALL. *Managing the impact of increasing microprocessor power consumption*, in "Intel Technology Journal", 1st quarter 2001.

[43] L. HAMMOND, E. AL.. *The Stanford Hydra CMP*, in "IEEE Micro", vol. 20, n^o 2, March 2000.

[44] S. HEO, K. BARR, K. ASANOVIĆ. *Reducing power density through activity migration*, in "Proceedings of the International Symposium on Low Power Electronics and Design", 2003.

[45] 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, http://citeseer.nj.nec.com/hsu98iperf.html.

[46] 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.

[47] D. JIMÉNEZ, C. LIN. *Neural Methods for Dynamic Branch Prediction*, in "ACM Transactions on Computer Systems", November 2002.

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

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

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

[51] 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.

[52] 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.

[53] D. MARR, E. AL.. *Hyper-threading technology : architecture and microarchitecture*, in "Intel Technology Journal", vol. 6, n^o 1, February 2002.

[54] 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.

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

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

[57] 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.

[58] K. SKADRON, M. STAN, W. HUANG, S. VELUSAMY, K. SANKARANARAYANAN. *Temperature-aware microarchitecture*, in "Proceedings of the 30th Annual International Symposium on Computer Architecture", 2003.

[59] J. TENDLER, E. AL.. *POWER4 system microarchitecture*, in "IBM Journal of Research and Development", vol. 46, n^o 1, January 2002.

[60] C. TICE, S. GRAHAM. *Key Instructions: Solving the Code Location Problem for Optimized Code*, 2000, http://citeseer.nj.nec.com/tice00key.html.

[61] 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.

[62] 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.

[63] L. WU, C. WEAVER, T. AUSTIN. *CryptoManiac: A Fast Flexible Architecture for Secure Communication*, in "Proceedings of the 28th International Symposium on Computer Architecture, Göteborg", June 2001.

[64] 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.

[65] 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.

[66] 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.

[67] 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.

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