INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team caps

Compilation, architectures des processeurs superscalaires et spécialisés

Rennes

THEME 1A

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

François Bodin [Professor, University of Rennes 1]
Jacques Lenfant [Professor, University of Rennes 1]
Henri-Pierre Charles [Teaching Assistant, university of Versailles-Saint-Quentin, on partial secondment at Inria in 2003]

Project technical staff Inria

Stéphane Bihan [till 11/30/03]
Antoine Colin [from 02/15/03]
Eric Courtois [till 06/30/03]
Thierry Lafage [from 02/15/03]

Junior technical staff Inria

Pierre Villalon [till 10/15/03]

Inria Postdoctoral fellows

Julio Hernandez [from 09/01/03]
Ronan Amicel [with CAPS Entreprise, from 02/01/03]

Ph.D. students

Laurent Bertaux [CIFRE allocation, STMicroelectronics, till 09/30/03]
Amaury Darsch [Inria allocation]
Assia Djabelkhir [French-Algerian allocation]
Romain Dolbeau [Allocation couplée, till 08/31/03, Teaching Assistant, from 09/01/03]
Antony Fraboulet [Inria allocation]
Karine Heydemann [Allocation couplée]
Laurent Morin [CIFRE allocation Thomson MMD, till 09/30/03]
Gilles Pokam [CIFRE allocation STMicroelectronics]
Olivier Rochecouste [Inria allocation]
Pascal Terjan [CIFRE allocation STMicroelectronics]
Eric Toullec [MENRT allocation]

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[9][7], and we are still refining them (cf. 6.1.1, 6.1.2, 6.1.3). We have also started research in order to reduce branch misprediction penalties [11][22] (cf. 6.1.3). We also aim at reducing the hardware costs of implementing wide-issue superscalar processors [3][8] (cf. 6.1.4) while we pursue researches on thread level parallelism on a single chip [20] (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. 6.2.3) and out-of-order execution for general-purpose architectures (IAOO, cf. 6.2.4). Most scientific applications make use of vector-like computations, while the behavior of the memory hierarchies on such computations is implementation dependent. We are developing an original approach for the code generation for vector-like kernels on IA 64 platforms (cf. 6.2.5). 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 [13] (cf. 6.2.7). 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. 6.2.6). Capitalizing on CAPS know-how in ISAs, compilers and simulators, we participate in the design of a cryptographic processor (PACCMAN, cf. 6.2.8).

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. 6.3.2).

Compiler optimizations for embedded systems require advanced software platforms that can support a large family of low end (assembly level) optimizations and that can be easily retargeted with new ISAs as well as new microarchitectures (ALISE, cf. 6.3.3). For the end user, understanding the effective performance of an application on a hardware platform is a major issue, since it is necessary to recognize the limiting component (hardware, compiler or application implementation). We are developing a performance debugging tool for this purpose (ATLLAS, cf. 6.3.4).

Finally, we use our knowledge of modern microarchitecture to participate in the definition of an unpredictable random number generator (HAVEGE, cf. 6.4).

Our research is partially supported by industry (Intel, STMicroelectronics, Thomson MMD). 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. 7.9).

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

Key words: *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 [44]. 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 [41][53]. Both branch directions and targets (even indirect jump targets) [31] 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 [50].

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

Key words: *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 [56]. 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 [66]. Some SMT processors are already available, like the Intel Pentium 4 [47]. 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 [51] by redundant execution), or more performance. A few ideas have been recently proposed to speed-up sequential execution, like for instance speculative threads [35], exception handling [65], helper threads for branch prediction [32], helper threads for memory prefetching [45], 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

Key words: *Embedded processors*, *High Performance*, *Compilation*, *Code Optimization*, *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 [39] 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 [64][34] 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 [17]. To master code size, code compression techniques [30] 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 measurement or by program static analysis. However these techniques are challenged by new hardware which behavior is fundamentally difficult to predict [52]. 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 [58]. 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 [61][60].

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 [33][43]. 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 [38][48][63] (cf. 6.3.4). An associated challenge is to preserve the link between aggressively optimized low level code and the source code [57]. As an alternative (or a complement) to performance tuning, automatic iterative compilation techniques [42] 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 [6] as well as fast processor simulation are key issues to support design exploration. Compiled simulation [12] 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

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

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** is currently being transfered to the CAPS Entreprise start-up and **HAVEGE** is freely distributed for non-commercial use.

5.2. ABSCISS

Participants: Ronan Amicel, François Bodin.

Key words: *retargetable processor simulation platform*.

ABSCISS (Assembly-Based System for Compiled Instruction-Set Simulation) is a retargetable system for high-speed instruction-set simulation (cf. 6.2.3). 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. 7.9).

5.3. HAVEGE

Participants: Julio Hernandez, André Seznec.

Key words: *Unpredictable random number generator*.

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

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

Key words: *Processor*, *cache*, *locality*, *memory hierarchy*, *branch prediction*, *simultaneous multithreading*, *multicore*.

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 [9][7]. We are still refining, analyzing (cf. 6.1.1) and exploring new usage of skewed associative caches (cf. 6.1.2). We are also pursuing researches on reducing branch misprediction penalties [11][22] (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 [3][8] (cf. 6.1.4) while continuing ongoing research on thread level parallelism on a single chip [20] (cf. 6.1.6, 6.1.5).

6.1.1. Modeling skewed-associativity

Participants: Pierre Michaud, André Seznec.

Past studies on skewed-associativity [9], based on cache simulations, have demonstrated the efficiency of this method of cache organization. However, the reasons of the observed efficiency are not intuitive. We have provided a theoretical framework for better understanding skewed-associativity [19]. This study has shown that the efficiency of skewed-associativity is inherently statistical, and does not require spatial locality. The model also provides a better understanding of the potential and limit of the self-data reorganization effect. Thanks to self-data reorganization, 2-way skewed-associativity emulates full associativity for working-sets up to 50% the cache size, and 3-way skewed-associativity emulates full associativity for working-sets up to 90% the cache size.

6.1.2. 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 [15]. This permits to envision either medium size L1 TLBs or very large L2 TLBs supporting multiple page sizes.

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 [11]. 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. We have also shown that, provided some extra storage in the checkpoint mechanism, the execution can be resumed without any extra penalty on a branch misprediction. The proposed solution allows to deliver a sustained address generation rate close to one instruction block per cycle with state-of-the-art accuracy [22].

6.1.3.2. Global history branch predictors

In the past few years, the CAPS team has been studying branch predictors built around the skewed predictor model [10][7]. In 2003, we have made available to the research community *2bcgskew* predictor simulators (http://www.irisa.fr/caps/project/Architecture/2bcgskew.html).

We have also initiated research on the promising concept of perceptron predictors [40]. Initial results show that the first studies by Jiménez and Lin were underestimating the potential of the perceptron predictor [27].

6.1.4. Mastering hardware complexity on wide-issue supercalar processors

Participants: Olivier Rochecouste, André Seznec, 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 [8], 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) [66] and CMP (Chip MultiProcessor) [35] 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 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" [20], 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.

6.2. Interaction on architecture and compilers

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

Participants: Ronan Amicel, Laurent Bertaux, Stéphane Bihan, François Bodin, Antoine Colin, Eric Courtois, Amaury Darsch, Assia Djabelkhir, Thierry Lafage, Gilles Pokam, André Seznec, Pierre Villalon.

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. 6.2.3) and out-of-order execution general-purpose architectures (IAOO, cf. 6.2.4). Most scientific applications make use of vector-like computations, while the behavior of the memory hierarchies on such computations is very implementation dependent. We are developing an original approach for the code generation for vector-like kernels on IA 64 platforms (cf. 6.2.5). 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 [13] (cf. 6.2.7). 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. 6.2.6). Capitalizing on CAPS know-how in ISAs, compilers and simulators, we participate in the design of a cryptographic processor (PACCMAN, cf. 6.2.8).

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 [16].

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 [59], 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. High Speed Instruction-Set Simulation

Participants: Ronan Amicel, François Bodin, André Seznec.

Instruction-set simulation can be used to evaluate different instruction-set architectures (ISAs) in the context of architecture exploration, or to validate a compiler back-end, to test, tune and debug programs, on a user-friendly PC or workstation rather than on actual silicon which might not even exist yet.

The increasing size and complexity of embedded software require extremely fast instruction-set simulation. Compiled instruction-set simulation is an approach that is potentially much faster than interpretation, but it has a startup cost due to the generation and compilation of the simulator. This startup cost is often seen as a major drawback and has limited the adoption of compiled instruction set simulation.

We have designed a new approach to compiled instruction-set simulation, that aims at reconciling flexibility, retargetability, high simulation speed, and small startup costs. This approach was implemented in ABSCISS [1][12], a generator of compiled instruction-set simulators that works at the assembler level (cf. 5.2).

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

Participants: François Bodin, Amaury Darsch, André Seznec, Pierre Villalon.

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 IAOO, 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.

Additionally, we have begun to investigate 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 [23].

6.2.5. Optimizing scientific applications on IA64 platforms

Participants: François Bodin, Eric Courtois, Amaury Darsch, André Seznec, Pierre Villalon.

In the framework of the EPICEA project (cf. 7.2), CAPS collaborates with PRiSM (University of Versailles). PRiSM research group is in charge of benchmarking and analyzing the specific behavior of each implementation of the IA64 architecture (Itanium 1, Itanium 2, Madison...). A significant effort is devoted to the analysis of the memory subsystem, which exhibits many pathological behaviors. Using this knowledge of the implementations of IA64 architecture, CAPS team develop specific technics to use the specific features of the implementation to optimize codes. The main achievement of this collaboration is the development by CAPS research group, of a highly optimized code generator for computation intensive loops. Such portions of code (called computation intensive kernels) are very common in scientific applications, and require very specific optimizations. These kernels usually represent vector computations. The generator can exploit all the IA64 architecture features like software pipeline (implemented in hardware with the rotating registers mechanism) and prefetch instructions. We have shown that the current implementations of the IA64 architecture can essentially be used like a vector machine on vector-like codes.

A methodology for optimizing softwares was developed around this generator. This method includes the detection of portions of code to optimize, the generation of the corresponding optimized kernels and the kernels integration in a library, the source code modification to make use of the generated kernels, and the linking of the modified source code and the library containing the generated kernels.

6.2.6. Low Power and Architecture Configurability

Participants: Gilles Pokam, François Bodin.

Managing power consumption/performance tradeoff on high-performance embedded processors has become a major challenge. The cache hierarchy is a typical example of such a power/performance tradeoff design point. On the one hand, a cache allows to maintain an important fraction of the embedded code and data workload on-chip, thus reducing the amount of memory traffic and thereby improving the performance and the power consumption. On the other hand, however, on some embedded processors, the cache memory accounts for up to 50% of the total chip area and for about 80% of the total transistor count [37], making the cache memory subsystem a critical source of power dissipation.

The processor design community reacted to this problem by making the cache subsystem reconfigurable [28][62]. This may allow to adapt the cache size requirement of a running program to a desired power/performance tradeoff. Former studies on reconfigurable cache [62][36][46] for embedded systems have only considered reconfiguration on a per-application basis.

It is well known that programs usually execute as a series of phases [54]. Our research has focused on the study of reconfiguring caches on a per-phase basis rather than on an application level. An important aspect of this research has focused on devising a fine-grain cache size adaptation scheme, driven by the compiler, for automatically characterizing the different cache size requirements of a program phase. Our model considers a great degree of reconfiguration flexibility, enabling a cache to be resized along its size and/or degree of associativity.

6.2.7. Exploitation of special purpose instruction sets in C programs

Participants: Stéphane Bihan, Gilles Pokam, François Bodin.

Many of today modern processors provide extensions to their instruction set specifically designed for computation-intensive multimedia applications (PowerPc AltiVec, TriMedia, Pentium MMX , ...). These extensions are based on SIMD instructions that operate on data subwords packed into registers. They are usually provided as intrinsics that can be inserted in the C code but they can also be inserted in the assembly code.

Direct insertion in assembly code requires good knowledge of both processor architecture and compilation techniques. Moreover such an approach does not lead to portable codes. Using intrinsics in C source code still requires code transformations (such as vectorization) for highlighting code regions where data parallelism can be exploited.

In the context of the Medea+ MESA project, CAPS has developed a C-to-C retargetable preprocessor prototype called SWARP [13]. SWARP searches for portions of code suitable to the use of multimedia instructions, and automatically inserts their intrinsics equivalent. SWARP is based on modern code analysis and code transformation (dependence analysis, alias analysis, loop transformation, vectorization, ...) and on a pattern matcher to recognize and replace suitable code patterns.

6.2.8. The Paccman project

Participants: Thierry Lafage, François Bodin, Antoine Colin, Ronan Amicel.

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 [6]), and (ii) a high speed compiled simulator generator (ABSCISS [12]).

The industrialization and maintenance of the PACCMAN development suite will be transfered to Caps Entreprise (cf. 7.9).

6.3. Compilers and software environment for high performance embedded processors

Key words: *compilation*, *code compression*, *optimization platform*, *performance debugging*.

Participants: Laurent Bertaux, François Bodin, Henri-Pierre Charles, Eric Courtois, Karine Heydemann, Laurent Morin, Pascal Terjan.

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 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. 6.3.2). Compiler optimizations for embedded systems necessitate software platforms which can support a large family of low end (assembly level) optimizations and can be easily retargeted with new ISAs as well as new microarchitectures (ALISE, cf. 6.3.3). For the end user, understanding the effective performance of a code on a hardware platform is a major issue, since one wants to recognize the faulty component (hardware, compiler or application implementation). We are developing a performance debugging tool for this purpose (ATLLAS, cf. 6.3.4).

6.3.1. Thread extraction for SOCs

Participants: Pascal Terjan, François Bodin.

Systems On Chip, aka SoC, are highly integrated architectures combining on the same chip at least a programmable processor, some memory and include other computing units. They suit very well embedded applications with some intensive computations.

Synthesizing for an application implies chosing the hardware components fitting the application and then mapping the application on the chosen architecture. Some code sections may require a dedicated component for some specific types of computations while others can be parallelized.

To achieve thread extraction, our approach focuses on two criteria : computational density and potential parallelism granularity. Compute intensive areas are determined through profiling. Dependencies between parts of the application are first statically computed from the call-graph. We then refine this analysis through an instrumented execution, particularly on the accesses to shared data.

6.3.2. Code Compression for Performance Code Size TradeOff

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

Designing an embedded system is essentially a tradeoff between hardware and software. Developers must achieve fast design taking into account various constraints such as memory space, power consumption and application speed. Memory space is a strong constraint as it directly impacts the cost and the functionalities of the system.

One would like to minimize the amount of memory space allocated to programs 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 good performance [17].

Two major techniques impact code size. On the one hand, optimizations improve performance (especially on architecture 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 large code size on critical sections where such a use 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 code may allow to reach a near optimal code size versus performance tradeoffs.

We have already investigated strategies for optimization under code size constraint [17]. We are currently designing a flexible infrastructure (target, constraint ...) offering both optimization and compression to explore this tradeoff.

We have defined a software compiler driven compression scheme for this infrastructure [18]. Our compression scheme can be used in multi-task systems and is machine independent. It has been designed to compressi small code shunks, since unfrequently executed code regions are spread all over the application.

Our scheme achieves a high compression ratio [1]. The compressed code is typically half the size of the original code. As we only compress the sparsely executed sections, total code size of the application is reduced from 13% up to 33% [24].

Our recent work deals with finding tradeoff strategy for tradeoff using this compression scheme.

6.3.3. ALISÉ : Assembly Level Infrastructure for Software Enhancement

Participants: Laurent Bertaux, François Bodin.

Optimization infrastructures are a major asset to produce efficient code for embedded systems. In particular, optimization infrastructures are needed for testing and validating alternative code optimizations [5]. ALISÉ is a new retargetable infrastructure for low level code optimizations. ALISÉ relies on a target description machine that provides structural information (instruction timing, resource usage etc.) and instruction semantic information. ALISÉ provides many functionalities, such as code region cloning, that allows alternative code optimizations.

ALISÉ is especially adapted to complex DSP processors featuring non-orthogonal data paths and multiple register files. Using ALISÉ we have defined a new combined scheduling/code generation technique. Thanks to semantic equivalence, code sequences can be replaced by semantically equivalent ones for shorter schedulings. This has been experimented on the MMDSP architecture from STMicroelectronics.

This work is supported by STMicroelectronics (Central R&D - Crolles).

6.3.4. Atllas: A Performance Debugging Tool

Participants: Laurent Morin, François Bodin.

Multimedia processors, and mainly VLIW processors, heavily rely on compilers for the production of high performance code. Our objective, supported by THOMSON R&D, is the design of a retargetable tuning tool called ATLLAS for Analysis and Tuning tool for Low Level Assembly and Source code. ATLLAS performs the static extraction of code quality informations from the optimized assembly level (number of types of instructions, functional unit occupation, WCET, critical path) and reports it at the source code level. It also allows the user to cross-check C source code and optimized assembly code through a graphical and interactive interface.

Support of multiple target architectures is enabled through the use of an abstraction of the code-checking coupled with a generic control flow analysis. Retargetability is allowed through the use of two software components. First the processor hardware description is handled through the SALTO [6] library. Second, we have implemented a target independent control flow graph builder used for the analysis of the optimized assembly code. Key of the code cross-checking, this component is based on an abstract Boolean model [29] used for predicates analysis as in [55] and for a specific data flow analysis.

ATTLAS has been fully implemented and tested. We have validated the principles of iterative interactive performance debugging with ATLLAS. We currently fully support the PHILIPS TRIMEDIA TM1000, and the EQUATOR MAP1000 and to a less extent the SPARC and the MIPS architectures.

ATTLAS also features an in-depth analysis of the C source code (satisfying the C ISO/IEC 9899 standard), an analysis of the relative performance by a worst case execution time analysis [49] and some automatic source code transformations.

6.4. HAVEGE: generating empirically strong random numbers

Key words: *unpredictable random number*, *cryptography*, *security*.

Participants: Julio Hernandez, André Seznec.

[1] We define the compression ratio as $\frac{total\ \ final\ \ code\ \ size}{origin\ \ code\ \ size}$ and the compression rate as $100 - compression\ \ ratio$

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.

More and more modern appliances such as PDAs, cell phones,... are built around low-power superscalar processors (e.g. StrongARM, Intel Xscale) and features complex operating systems. This year, a demonstrator of HAVEGE for such a PDA featuring PocketPC2002 and a Xscale processor has been developed and is now distributed.

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. 8.1).

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: Stéphane Bihan, François Bodin, Eric Courtois.

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 6.2.7 and 6.3.2.

7.2. Project EPICEA (2001-2004)

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

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 6.2.4 and 6.2.5.

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 6.2.8.

7.4. Research grant from Intel

Participants: Eric Toullec, Antony Fraboulet, André Seznec.

The researches on instruction fetch front ends (cf. 6.1.3) and register file structures (cf. 6.1.4) 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. Code analysis and performance evaluation for embedded systems (2000-2003)

Participants: François Bodin, Laurent Morin.

The doctoral studies of Laurent Morin is supported by a convention CIFRE with the Thomson MMD society (cf. 6.3.4).

7.6. Flexible infrastructure for code scheduling and optimisation (2000-2003)

Participants: François Bodin, Laurent Bertaux.

The doctoral studies of Laurent Bertaux are supported by a convention CIFRE with the STMicroelectronics society (cf. 6.3.3).

7.7. Thread extraction for heterogeneous embedded systems (2002-2005)

Participants: François Bodin, Pascal Terjan.

The doctoral studies of Pascal Terjan are supported by a convention CIFRE with the STMicroelectronics society (cf. 6.3.1).

7.8. Compilation and power consumption (2000-2003)

Participants: François Bodin, Gilles Pokam.

The doctoral studies of Gilles Pokam are supported by a convention CIFRE with the STMicroelectronics society (cf. 6.2.6).

7.9. Start-up company project

Participants: Ronan Amicel, Stéphane Bihan, François Bodin, Eric Courtois, Romain Dolbeau, Karine Heydeman, Laurent Bertaux, Antoine Monsifrot, Laurent Morin, André Seznec.

In 2003 a large part of the research team was involved in the set up of the start-up company CAPS entreprise (http://www.caps-entreprise.com/). CAPS entreprise aims at bringing innovative software tools, solutions and services to the market of high performance embedded systems. The company aims at becoming a reliable partner for system builders, platform designers and developers seeking the best system performance, by helping them match their software to the specificities of the underlying hardware platform.

CAPS entreprise offers standalone tools that are specialized for a given task (code transformation, simulation, worst-case execution time analysis...). These tools can act as building blocks in a software tool chain and are designed for seamless integration into common development environments.

The company proposes global compilation solutions, tailored to the customer's needs. After a detailed exploration of the user requirements against the existing process, specific additions and enhancements to the previous code generation infrastructure are proposed and implemented.

CAPS entreprise finally offers custom consulting services, such as performance analyses or instruction-set evaluations. Through these services, customers benefit from the company's in-house expertise and tools, helping them make strategic decisions on complex technology issues.

The company was awarded in 2003 by the french ministry of Industry as an innovative company. CAPS entreprise is incubated by Emergys and Inria-Transfert.

Industrial tranfer of ABSCISS (cf. 6.2.3) to CAPS entreprise is supported by Inria through the postdoc fellowship of Ronan Amicel.

8. Other Grants and Activities

8.1. 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.

8.2. 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 HiPC'2003 (High Performance Computing). He is a member of the program committees of 31th International Symposium on Computer Architecture (ISCA'31) and of CARI'2004. He is the program co-chair of ICS 2004 (International Conference on Supercomputing).
- F. Bodin is a member of the editorial board of TSI and has been a member of the program committee of MEMOCODE'2003.

9.2. University teaching

F. Bodin and A. Seznec are teaching computer architecture and compilation at DEA of computer science, at DIIC and DESS ISA at IFSIC, University of Rennes I.

9.3. Workshop, seminar, invitations

- A. Seznec has presented a seminar entitled " HAVEGE: Hardware Volatile Entropy Gathering and Expansion, generating unpredictable random number at user-level" at UPC, Barcelona in April 2003.
- A. Seznec has presented a seminar entitled " Ahead Pipelining the Instruction Address Generator" at UPC, Barcelona in April 2003 and at Intel, Hillsboro, Oregon in June 2003.
- Yanos Sazeides, University of Cyprus, has been visiting the project for a month in May 2003.

9.4. Misceleanous

- A. Seznec is an elected member of the evaluation committee of Inria.
- F. Bodin is an elected member of the IFSIC Committee.
- F. Bodin is responsible of the DEA of computer science 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 with V. Verdon and the help of Y. Sost at the origin of the creation of the fundation M. Métivier (http://www.fondation-metivier.org).
- Jacques Lenfant is the chairman of the professor hiring committee of University of Rennes I.
- Jacques Lenfant is a member of “académie des sciences et des technologies”.

10. Bibliography

Major publications by the team in recent years

[1] 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.

[2] 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.

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

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

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

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

[7] 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.

[8] 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.

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

[10] 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 », IEEE-ACM, editor, Denver, June, 1997.

[11] 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

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

Articles in referred journals and book chapters

[13] G. POKAM, S. BIHAN, J. SIMONNET, F. BODIN. *SWARP: A Retargetable Preprocessor for Multimedia Instructions.* in « Concurrency and Computation: Practice and Experience », number 2-3, volume 16, February - March, 2004, pages 303 - 318.

[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. *Concurrent support of multiple page sizes on a skewed associative TLB.* in « IEEE Transactions on Computers », to appear.

Publications in Conferences and Workshops

[16] 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.

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

[18] K. HEYDEMANN, H.-P. CHARLES, F. BODIN. *Schéma de compression reconfigurable avec un émulateur logiciel pour la recherche de compromis entre la taille du code et sa performance.* in « RenPar'15/CFSE'3/SympAAA'2003 », pages 417-424, La Colle sur Loup, France, 2003.

[19] P. MICHAUD. *A statistical model of skewed-associativity.* in « Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software », pages 204-213, March, 2003.

[20] 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, to appear.

[21] 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.

[22] 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) », pages 241–252, June 9–11, 2003.

Internal Reports

[23] A. DARSCH, A. SEZNEC. *Out-of-order execution of predicated instruction sets through translation register buffers.* Technical report, number 1573, IRISA, November, 2003, http://www.inria.fr/rrrt/rr-5011.html.

[24] K. HEYDEMANN, H.-P. CHARLES, F. BODIN. *A compression scheme for code size versus performance trade-off.* Research Report, number 1574, IRISA, 2003.

[25] G. POKAM, F. BODIN. *Energy reduction potential of a phase-based cache resizing scheme for embedded systems.* Technical report, number 1582, IRISA, December, 2003, http://www.inria.fr/rrrt/rr-5036.html.

[26] G. POKAM, F. BODIN. *Energy-Delay Tradeoff Analysis of ILP-based Compilation Techniques on a VLIW Architecture.* Technical report, number 1572, IRISA, November, 2003, http://www.inria.fr/rrrt/rr-5026.html.

[27] A. SEZNEC. *Redundant History Skewed Perceptron Predictors: pushing limits on global history branch predictors.* Research report, number 1554, IRISA, September, 2003.

Bibliography in notes

[28] D. ALBONESI. *Selective Cache Ways: On-demand Cache Resource Allocation.* in « Proceedings of the 32nd International Symposium on Microarchitecture », November, 1999.

[29] ANDERSEN, HULGAARD. *Boolean Expression Diagrams.* in « LICS: IEEE Symposium on Logic in Computer Science », 1997.

[30] A. BESZÉDES, R. FERENC, T. GYIMÓTHY, A. DOLEN, K. KARSISTO. *Survey of Code-size reduction methods.* in « ACM Computing Survey », number 3, volume 35, September, 2003, pages 223-267.

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

[32] 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.

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

[34] S. DEBRAY, W. EVANS. *Profile-Guided Code Compression.* in « ACM PLDI'02 », number 5, volume 37, 2002.

[35] L. HAMMOND, E. AL.. *The Stanford Hydra CMP.* in « IEEE Micro », number 2, volume 20, March, 2000.

[36] F. HAYAKAWA, H. OKANO, A. SUGA. *An Eight-Way VLIW Embedded Multimedia Processor with Advanced Cache Mechanism.* in « Proceedings of the Third IEEE Asia-Pacific Conference on ASICs (AP-ASIC2002) », August, 2002.

[37] J. HENNESSY. *The Future of Systems Research.* in « IEEE Computer », August, 1999, pages 27-33.

[38] 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.

[39] M. JACOME, G. DE VECIANA. *Design challenges for new application specific processors.* in « IEEE Design and Test of Computers », number 2, volume 17, 2000, pages 40–50.

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

[41] R. E. KESSLER. *The Alpha 21264 microprocessor.* in « IEEE Micro », number 2, volume 19, 1999.

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

[43] 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 », pages 12-23, 2003.

[44] 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.

[45] 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.

[46] A. MALIK, B. MOYER, D. CERMAK. *A Low Power Unified Cache Architecture Providing Power and Performance Flexibility.* in « International Symposium on Low Power Electronics and Design », 2000.

[47] D. MARR, E. AL.. *Hyper-threading technology : architecture and microarchitecture.* in « Intel Technology Journal », number 1, volume 6, February, 2002.

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

[49] G. OTTOSSON, M. SJÖDIN. *Worst-Case Execution Time Analysis for Modern Hardware Architectures.* in « ACM SIGPLAN 1997 Workshop on Languages, Compilers, and Tools for Real-Time Systems (LCT-RTS'97) », 1997.

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

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

[52] 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.

[53] 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.

[54] T. SHERWOOD, B. CALDER. *Time Varying Behavior of Programs.* Technical report, University of California, San Diego, August, 1999.

[55] J. W. SIAS, W. MEI W. HWU, D. I. AUGUST. *Accurate and efficient predicate analysis with binary decision diagrams.* in « Proceedings of the 33rd annual IEEE/ACM international symposium on Microarchitecture December 2000 », ACM Press, 2000.

[56] J. TENDLER, E. AL.. *POWER4 system microarchitecture.* in « IBM Journal of Research and Development », number 1, volume 46, January, 2002.

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

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] C. ZHANG, F. VAHID, W. NAJJAR. *A Highly Configurable Cache Architecture for Embedded Systems.* in « 30th International Symposium on Computer Architecture », June, 2003.

[63] 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, pages 155–164, 2002, http://doi.acm.org/10.1145/513829.513857.

[64] 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.

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

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