INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team caps

Compilation, architectures des processeurs superscalaires et spécialisés

Rennes

THEME COM

Table of contents

1. Team

Scientific head
André Seznec [Research Director Inria, HdR]

Administrative Assistant
Evelyne Livache [TR Inria]

Inria staff members
Pierre Michaud [Research scientist]

Academic staff
François Bodin [Professor, University of Rennes 1, HdR]
Jacques Lenfant [Professor, University of Rennes 1, HdR]
Isabelle Puaut [Professor, University of Rennes 1, HdR]
Olivier Rochecouste [Teaching assistant, University of Rennes 1, from 01/09/06]

Technical staff Inria
Karine Brifault [from 24/04/06]
Florence Dru [from 01/01/06]
Sébastien Matz [from 06/03/06]
Christophe Pais [from 01/04/06]
Guillaume Papauré [from 01/12/05]
Olivier Rochecouste [until 31/08/06]
Robin Schmutz [from 15/11/06]

Junior technical staff Inria
Jerémy Fouriaux

Inria Postdoctoral fellow
He Liqiang [till 09/13/06]
Jan Staschulat [from 11/09/06 to 11/11/06]
Guntur Surendra [from 01/12/06]

Ph.D. students
Arnaud Alexis [Inria allocation, till 15/03/06]
Jean-François Deverge [MENRT allocation]
Julien Dusser [Inria Allocation, from 01/09/06]
Damien Fétis [MENRT allocation]
Robert Guziolowski [Inria Allocation, from 15/09/06]
Eric Petit [Inria allocation]
Thomas Piquet [Inria allocation]

2. Overall Objectives

2.1. Overall Objectives

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

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

Our research in computer architecture covers memory hierarchy, branch prediction, superscalar implementation, as well as SMT and multicore processors. In the recent past, we have proposed several new complexity-effective structures for caches and branch predictors[2], [12], and we are still very active in these areas (cf. 6.1.1, 6.1.4). We pursue researchs on architecture exploiting thread level parallelism on a single chip (cf. [5], 6.1.2, 6.1.3). At the same time, power consumption and temperature hot spot management have become major issues for all processors. We have initiated a research activity on temperature management at architectural level (cf. 6.1.5). We are also studying how the compiler and the architecture can interact to optimize the power consumption/performance tradeoff (cf. 6.1.7, [9]).

Performance, but also power consumption or hardware system cost depends on the processor architecture but can also be managed at the compiler/code generation level. For instance, code size is often an issue with embedded systems. We are exploring tradeoffs leveraging code compression and interpretation (cf. [27]). We are exploring thread extraction for the different hardware components for heterogeneous SOCs (System On a Chip) featuring special purpose hardware and one or more execution cores (cf. 6.2.1).

In hard real-time embedded systems the task WCETs must be correctly evaluated, so that it can be proven that task temporal constraints (typically, deadlines) will be met. Our research concerns methods for automatically computing upper bounds of the execution time of applications on a given hardware platform. Embedded platforms may now feature caches, branch predictors, complex pipeline, ... A particular focus is put on hardware-level analysis (static analysis based on timing models) and compiler-directed schemes aimed at improving predictability. Our studies concern WCET-oriented (as opposed to average-performance oriented) compilation and measurement-based WCET estimation (cf 6.3).

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). We also participate in several institutionally funded projects (NoE HIPEAC, IP Fet project SARC, ACI Securité UNIHAVEGE, ANR funded QCDnext, GaLogic, Para, Mascotte, and “Poles de compétitivités” funded Scalimages, Sceptre and Fame2). Some of the research prototypes developed by the project during the past few years have been transferred to industry through the CAPS Entreprise start-up (cf. 7.2).

3. Scientific Foundations

3.1. Panorama

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

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

3.2. Uniprocess architecture

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

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

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

The use of a complex memory hierarchy has been generalized over the past decade. On modern microprocessors, both software and hardware prefetching are now widely used to enable the on-time presence of data and instructions in the memory hierarchy. Highly efficient, but complex data hardware prefetch mechanisms, have been proposed to hide several hundreds of instruction slots [58]. The challenge for computer architects is to reduce the complexity of these hardware mechanisms to enable simpler implementations. Another challenge is to propose new prefetch mechanisms that can hide several thousands of instruction slots.

Over the past decade, efficient branch prediction mechanisms have been proposed and implemented [55][12]. Both branch directions and targets (even indirect jump targets) [45] are predicted. Most of these predictors exploit either local or global branch history. The accuracy of the prediction seems to be reaching a plateau.

The complexity of many components in the processor (in terms of silicon area, power consumption and response time) increases superlinearly (and often quadratically) with the issue width e.g. register renaming, instruction scheduling, bypass network and register file access. These components are becoming the bottlenecks that limit the issue width and the cycle time [62].

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

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

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

Keywords: *multicore processor*.

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

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

General purpose multicore processors are already available and will become mainstream in the next few years. On a multicore, the tasks execute on distinct processing units. Resource sharing concerns only one or several on-chip cache levels, and chip pins. This is to be contrasted with SMT processors, on which all resources are shared apart a few buffers [70]. However, the main difficulty for the design of SMT processors is the design of a very wide issue superscalar processor. Though SMT and multicore approaches both exploit task parallelism, they are orthogonal, as illustrated by the dual-core SMT Pentium 4 and the dual core SMT IBM Power 5.

A key issue concerning SMT / multicore processors is whether they can improve sequential execution. Among possible improvements, one may seek to obtain a more reliable execution (for instance [64] by redundant execution), or more performance. A few ideas have been recently proposed to speed-up sequential task execution, like for instance speculative threads [50], exception handling [75], helper threads for branch prediction [46], helper threads for memory prefetching [59], etc. Among solutions already proposed, it is not yet clear which are viable and which are not. It will depend on the performance gain / hardware complexity tradeoffs. Ongoing research on this topic will decide the scope of future SMT/multicore processors.

3.4. Compiling and optimizing for embedded applications

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

Embedded processors range from very small, very low-power systems (for instance for telemetry counter sensors which must run on one battery for 10 years) to power hungry high-end processors used in radars or set-top boxes. The spectrum of softwares range from very small code kernels (a few Kinstructions) to millions of code lines including a real time operating system. The constraints on the code quality vary from “just no bugs” to safety critical with hard real time problems, but may also to achieve a determined performance level at the smallest possible hardware cost or the smallest possible power consumption. Therefore embedded processors are presenting many new challenges [54] to the hardware and compiler research community.

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

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

Finding a tradeoff between binary code size and execution time [74], [48] is a major issue in many applications. For small micro-controllers, “the smaller the code, the faster” is an effective rule of thumb. However, for recent embedded processors featuring instruction level parallelism (e.g., VLIW processors), faster code generally means larger code size [4]. To master code size, code compression techniques [44] can also be used to reduce memory size of infrequently executed code regions.

In the context of real time systems, average performance is often not a critical issue, but the worst case execution time (WCET) may be critical. WCET estimations can be either obtained by measurements or by static analysis of programs. However these techniques are challenged by recent processors whose behavior is fundamentally difficult to predict [65]. 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 [69]. A more interesting issue arises with configurable hardware, for instance cache memories that can vary in size or associativity. In that case, the compiler can trade off performance for power consumption [72], [71].

While many optimizations and code transformations have been proposed over the past two decades, the interactions between these optimizations are not really understood. The many optimizations used in modern compilers sometimes annihilate each other [47], [57]. 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 [53], [61], [73]. An associated challenge is to preserve the link between aggressively optimized low level code and the source code [68]. As an alternative (or a complement) to performance tuning, automatic iterative compilation techniques [56] address the interactions of optimizations through the use of feedback, to find efficient code transformation sequences.

Time-to-market is a major challenge for embedded processor designers. Wide spectrum of possible derived hardware platforms (configurations, co-processors, etc.) is also a major issue for embedded system designers. Defining or dimensioning an embedded system (hardware, compiler and application) requires to explore a large solution space for the best cost/performance/application. Retargetable compiler infrastructures [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.

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

4. Application Domains

4.1. Application Domains

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

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

5. Software

5.1. Panorama

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

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

5.2. ATMI

Keywords: *Microarchitecture temperature model*.

Participant: Pierre Michaud.

Contact : Pierre Michaud

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

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

We have developed an ad hoc temperature model, ATMI (Analytical Model of temperature in MIcroprocessors), for studying thermal behaviors over a time scale ranging from microseconds to several minutes. ATMI is based on an explicit solution to the heat equation [7] and on the principle of superposition. ATMI can model any power-density map that can be described as a superposition of rectangle sources, which is appropriate for modeling the microarchitectural units of a microprocessor.

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

5.3. HAVEGE

Keywords: *Unpredictable random number generator*.

Participants: Olivier Rochecouste, André Seznec.

Contact : André Seznec

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

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

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

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

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

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

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

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

6. New Results

6.1. Processor Architecture

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

Participants: François Bodin, Damien Fétis, Julien Dusser, Robert Guziolowski, He Liqiang, Pierre Michaud, Thomas Piquet, Olivier Rochecouste, André Seznec.

Our research in computer architecture covers memory hierarchy, branch prediction, superscalar implementation, as well as SMT and multicore issues. In the recent past, we have proposed several new complexity-effective cache and branch predictor structures [2], [12]. We are still refining, analyzing and exploring new cache management policies (cf. 6.1.1). New directions in branch prediction are explored (cf. 6.1.4). We are continuing research on thread level parallelism on a single chip (cf. 6.1.2) and have begun the exploration of new directions of multicore architectures (cf. 6.1.3).

Power consumption and temperature management have become a major concern for high performance processor design. We have initiated a new research direction in temperature issues on single-chip parallel processors (cf. 6.1.5). We are exploring power consumption reduction through two directions, exploiting the dynamic width of operands (cf. 6.1.7) and using branch confidence for fetch gating (cf. 6.1.6)

6.1.1. Content conscious management of the memory hierarchy

Participants: Thomas Piquet, Olivier Rochecouste, André Seznec.

Efficient cache hierarchy management is of a paramount importance when designing high-performance processors. Upon a miss, the conventional operation mode of a cache hierarchy is to retrieve back the missing block from higher levels and to store the block into all hierarchy levels. It is however difficult to assert that storing the block into intermediate levels will be really useful.

In [39], we address this phenomenon in the highest level of cache hierarchy. Our observations reveal that cache blocks that are only accessed once - single-usage blocks - are quite significant at runtime and especially in the highest level of cache hierarchy. Employing a simple prediction mechanism is sufficient to uncover most of the single-usage blocks. For a two-level cache hierarchy, these blocks are directly sent from main memory to level-1 cache. Performing data bypassing on level-2 cache maximizes its efficiency and allows hard-to-prefetch memory references to remain into this cache hierarchy level. Our experimental results show that minimizing single-usage cache pollution in the level-2 cache leads to a significant miss rate decrease; resulting therefore in noticeable performance gains.

6.1.2. Resource sharing between cores on a single chip

Participants: He Liqiang, André Seznec.

As the increase of issue width on superscalar processors brings diminishing returns, thread parallelism with a single chip is becoming a reality. In the past few years, both SMT (Simultaneous MultiThreading) [70] and CMP (Chip MultiProcessor) [50] approaches were first investigated by academics and are now implemented by the industry. In some sense, CMP and SMT represent two extreme design points. In [3], we showed that there exists 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 SMT-like resource sharing when such a sharing can be made non-critical for implementation, but resource splitting à la CMP whenever resource sharing leads to a superlinear increase of the implementation hardware complexity.

We have further evaluated the performance trade-off associated with the CASH approach [37].

6.1.3. Exploring new directions in multicore architectures

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

Chip multiprocessors (CMP) is now considered as the most cost-effective solution for getting high performance from a single component by most manufacturers. Due to this trend, it can be expected that in the next few years, parallelism will be taken into account during application development.

Through the Ph.Ds of R. Guziolowski and J. Dusser both started in september 2006, we want to explore the design space of multicores In particular, we want to explore intermediate design points between a few complex cores CMP design and a large number of simple cores CMP design. We want also to explore the new tradeoffs on memory hierarchy implied by multicores, in particular the traditional solutions "use the remaining silicon area for caches" and "saturate bandwidth with prefetching" can not be used directly.

This study is supported by the European project SARC

6.1.4. Branch prediction

Participants: Pierre Michaud, André Seznec.

While most recent researches in branch prediction (including the O-GEHL predictor [15]) were acknowledging that using an adder tree as the final prediction computation function was the best approach, we identified that partial tag match is an alternate candidate [6]. Using GEometric history length as the O-GEHL predictor, the TAGE predictor [25] uses (partially) tagged components as the PPM-like predictor. TAGE relies on (partial) hit-miss detection as the prediction computation function. For realistic hardware budgets in the 64Kbits-256Kbits range, TAGE provides state-of-the-art prediction accuracy on conditional branches [25], [33]. A version of TAGE [33] was presented to the realistic track at the 2nd Championship Branch Prediction workshop held in december 2006 (http://camino.rutgers.edu/cbp2/).

However, for very large (and irrealistic) storage budgets and very large (and irrealistic) number of predictor components, the GEHL predictor is more accurate than the TAGE predictor on most benchmarks. Moreover a loop predictor can capture part of the remaining mispredictions. To explore the limits of branch prediction accuracy, the GTL predictor [34], combining a GEHL predictor, a TAGE predictor and a loop predictor, was presented to the idealistic track at the 2nd Championship Branch Prediction workshop (http://camino.rutgers.edu/cbp2/).

The accuracy of the prediction of the targets of indirect branches is a major issue on some applications. We have shown that the principles of the TAGE predictor can be directly applied to the prediction of indirect branches. The ITTAGE predictor (Indirect Target TAgged GEometric history length) significantly outperforms previous state-of-the-art indirect target branch predictors [25].

6.1.5. Tackling temperature issues

Participants: Pierre Michaud, André Seznec, Damien Fétis.

Now power density has reached levels that make temperature a constraint that affects the microarchitecture [49], [66]. Temperature must be limited because of its detrimental effect on circuit timing, mean time to failure, and leakage currents. The advent of multi-core processors exacerbates this problem, as the electric power dissipated in a processing core increases the temperature in other cores.

Our research on temperature-aware architecture is mainly divided in two parts : (1) studying temperature models and (2) using the models to study the implications of the temperature constraint on the architecture and the operating system.

Our research on temperature modeling led to an analytical model of chip temperature [7]. The ATMI software (cf 5.2) implements this analytical model. Most studies on temperature-aware architectures blindly use the HotSpot temperature model [42]. This model is based on a coarse-grain space discretization. Our experiments showed that the coarse grain space discretization used in HotSpot is not sufficient to model correctly transient temperature [26]: previous studies blindly using HotSpot underestimate the slope of temperature transients. The ATMI model is based on an exact solution to the heat equation. ATMI models transient temperature more accurate than Hotspot.

The ATMI model was used to study the implications of the temperature constraints on the architecture and the operating system. Previous studies, focusing on steady-state behavior, have shown that thread migration is an effective technique for decreasing temperature in multi-cores [52], [63]. Hence thread migration allows to increase the computing throughput when performance is constrained by temperature. However, the benefit of thread migration is underestimated if one considers only the steady state. We have shown that thread migration brings most of its benefit during the first tens of seconds following a decrease of the number of running threads [22].

Understanding temperature oscillations is critical for understanding the temperature hot-spot issues and the associated trade-offs [38]. For instance, we have identified the need for the operating system scheduler to work with smaller time slices as the performance and power density of microprocessors increase due to technology miniaturization. We have proposed a new scheduling method for thermally-constrained multi-core chips. This method takes advantage of thread migration to decrease temperature by spreading heat evenly on all cores. We

have shown that, for respecting fairness while taking into account different thread priorities, it is sometimes necessary to run fewer threads than cores. Our scheduling method achieves fairness while keeping the multi-core close to thermal saturation and maximum computing throughput.

6.1.6. Confidence estimation and fetch gating using state-of-the-art branch predictors

Participants: Pierre Michaud, André Seznec.

In a dynamic reordering superscalar processor, the front-end fetches instructions and places them in the issue queue. Instructions are then issued by the back-end execution core. The front-end fetches instructions as fast as it can until it is stalled by a filled issue queue or some other blocking structure. This approach wastes energy: (i) speculative execution causes many wrong-path instructions to be fetched and executed, and (ii) back-end execution rate is usually less than its peak rate, but front-end structures are dimensioned to sustained peak performance. Dynamically reducing the front-end instruction rate and the active size of front-end structure (e.g. issue queue) is a required performance-energy trade-off.

In [35], we propose Speculative Instruction Window Weighting (SIWW), a fetch gating technique that allows to address both fetch gating and instruction issue queue dynamic sizing. A global weight is computed on the set of inflight instructions. This weight depends on the number and types of inflight instructions (non-branches, high confidence or low confidence branches, ...). The front-end instruction rate can be continuously adapted based on this weight. SIWW is shown to perform better than previously proposed fetch gating techniques. SIWW is also shown to allow to dynamically adapt the size of the active instruction queue.

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

6.1.7. Width partitioned microarchitectures

Participants: Olivier Rochecouste, François Bodin, André Seznec.

Current superscalar processors feature 64-bit datapaths to execute the program instructions, regardless of their operands sizes. Analysis indicates, however, that most executions comprise a large amount (40%) of narrow-width operations, i.e. instructions which exclusively process narrow-width operands and results. These properties can be exploited to master the hardware complexity of superscalar processors. We have proposed a width-partitioned microarchitecture (WPM) to decouple the treatment of narrow-width operations from that of the other program instructions. E.g. a 4-way issue processor is split into two clusters: one executing 64-bit operations, load/store and complex operations and the other treating the 16-bit operations. Revealing the narrow-width operations to the hardware appears to be sufficient to keep the workload balanced and the communications minimized between clusters. Using a WPM reduces the complexity of several critical processor components, in particular the register file and the bypass network. A WPM also lowers the complexity of the interconnection fabric since the 16-bit cluster is only able to propagate narrow-width data. Using a WPM model saves power and area with a minimal impact on performance [24].

This work was done in collaboration with Gilles Pokam, while he was with University of San Diego.

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

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

Participants: François Bodin, Eric Petit, Jérémy Fouriaux, Florence Dru, Sébastien Matz, Guillaume Papauré.

High performance embedded platforms are now buil with System On a Chip (SoC) components. Mapping an application on such an heterogeneous platform is a challenge. For these heterogeneous SoCs,, we are defining and developing the ASTEX approach. ASTEX aims at the extraction of speculative threads for the different hardware components of a SoC (cf 6.2.1). We apply ASTEX for automatically exploiting a GPU (cf 6.2.2).

Domain specific processors also require optimized code generation. In the framework of the ApeNEXT project, we are exploring optimization strategies on a high performance VLIW processor.

6.2.1. Speculative thread extraction for SOCs with ASTEX

Participants: François Bodin, Eric Petit, Florence Dru, Guillaume Papaure.

Systems on a chip (SoCs) are highly integrated architectures which combine multiple heterogeneous computing units. A main processor runs the application. Coprocessors which may feature their own memory, are used to speedup the processing of some parts of the application. A SoC implementing such configuration aims at exploiting each processing unit. Typically, coprocessors run computation intensive sections of an application. The design of an application for such a SoC begins with the partitionning of the applications in threads that are then mapped onto the computing units of the SoC. In ASTEX, Automatic Speculative Thread EXtractor,[30], [28], we address the problem of partitionning C code into threads for heterogeneous SoCs.

ASTEX first performs a profile measurement to identify hotpaths on the application. Based on the identified hotpaths, sets of possible speculative threads are then evaluated; as a criteria, we try to minimize the amount of memory transfers. Since the threads are computed from a profile, speculation arizes at two levels. At control flow level: the thread is a subset of the possible paths in the execution of the application program. At data dependency level: the data dependency between the threads and the main program must be preserved.

After the hotpath detection, ASTEX builds an executable C version of the code with the speculative threads. At this step, the instrumented program is exercised against many different input data and ASTEX determines a third level of speculation: a speculative memory usage model. The last step, according to execution results, evaluates the characteristics of the potential threads and refines the selection if needed.

Due to speculative nature of the threads, the data structures used by the thread must be copied in the local memory of the coprocessor. The speculative thread must test its validity and returns an error on any misspeculation: control flow, data dependency and memory usage.

In the first version of ASTEX, we assume that there is no overlap between the execution of the main program and the speculative thread. This constraint is currently being removed.

6.2.2. Automatically exploiting GPU with ASTEX

Participants: Eric Petit, Sébastien Matz, François Bodin.

Because of their high potential computing power, Graphical Processing Units (GPU) looks very attractive to speed up programs, even if they are difficult to program. However, as data transfers from the main memory to the GPU may strongly impact the resulting performance, porting code to the GPU is generally done manually.

The goal of this study [29] is to find better programming tools for GPU, able to automatically exploit the GPU in the context of general programming. We focus the effort on pieces of code that fit GPU constraints, while looking at the locality of data transfers.

ASTEX, 6.2.1, is used to implement a dynamic analysis that detects speculative threads which contains computing kernels with a potential speedup on the GPU. We have developed a tool that automatically looks for parallelisable code fragments and replaces them with code for a graphics co-processor through an underlying parallel model. This tool is able to compute the static and dynamic properties of code sequence to evaluate their potential s for GPU usage.

6.2.3. Code generation for the ApeNEXT project

Participants: François Bodin, Jérémy Fouriaux.

ApeNEXT is the latest generation of massively parallel supercomputers dedicated to particles physics simulations [19]. It delivers Teraflops performances. This machine is an array of dedicated VLIW processors called J&T. Those processors fulfill all requirements needed to run single program multiple data (SPMD) applications.

CAPS is in charge of the building of SOFAN (Software Optimizer for ApeNEXT). This software optimizer attempts to explore different optimization strategies for ApeNEXT applications. Indeed, applications of physics phenomenons, coded with a maximum of parallelism, are hard to optimize with traditional methods: Some optimizations become useless, and others cause side effects which modify the performance of other optimizations. As ApeNEXT is designed for these applications, we must also take full advantage of its original hardware.

Due to the atypical architecture of ApeNEXT and of its dedicated applications, several original optimizations are developed in combination with well-known techniques. This mix will lead to a set of optimizations particularly valuable on an ApeNEXT-like machine. Another goal of SOFAN is to provide information to users on more aggressive, but possibly not semantically safe optimizations.

6.3. WCET estimation

Participants: Karine Brifault, Isabelle Puaut, François Bodin, Christophe Païs, Jean-François Deverge, Robin Schmutz, Jan Staschulat.

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

In 2006, our results concern the definition of hardware models for out-of-order pipelines and WCET-oriented (as opposed to average-performance oriented) compilation.

6.3.1. Structural models for out-of-order pipelined processors

Participants: François Bodin, Karine Brifault, Robin Schmutz.

High performance microprocessors combine caches, pipelines and control speculation. The execution time of any instruction sequence cannot be analyzed in isolation since it depends on the execution history. There is no complete formal description of an existing superscalar processor that would allow to prove that a WCET estimation is safe. Therefore using complex processors is problematic when dealing with real time constraints.

In order to take into account the timing effects of the various components in the processors, recent researches have developed static methods to try to obtain cycle accurate models. However, these methods do not reveal the causes of the timing effects. As a consequence, to be safe, the obtained WCET on pipelined processors with out-of-order execution is often overestimated.

Our approach [41] is to evaluate the execution time of each basic block through a cycle-accurate emulator of the processor. The emulator is derived from the processor documentation. The model is then checked against the real hardware. This allows to detect timing effects, and their causes. This also enables to detect timing anomalies (which might be due to an uncomplete documentation). This may allow to define guidelines to generate codes with more predictable execution times.

The design, this model has been implemented and validated for two different platforms, a PowerPc 7450 and an Intel Xscale PX255.

6.3.2. WCET-oriented compilation

Participants: Isabelle Puaut, Jean-François Deverge, Christophe Païs, Jan Stachulat.

Hard real-time tasks must meet their deadline in all situations, including in the worst-case ones. Moreover many hard real-time applications need to be fast as well. As a consequence, architectures with caches and/or on-chip static RAM (scratchpad memories) are of interest for such applications. Our work on WCET-oriented compilation has focused on the definition of schemes for software-control of the memory hierarchy (cache, scratchpad memories). Our goal was to find the best trade-off between performance and predictability.

6.3.2.1. Instruction cache locking

Cache memories are sources of predictability problems because of their dynamic and adaptive behavior, and thus require special attention to be used in hard real-time systems. However, cache-aware WCET analysis techniques are not always applicable due to the lack of documentation of hardware manuals concerning the cache replacement policies. Moreover, they tend to be pessimistic with some cache replacement policies (e.g. random replacement policies) [51] and may even be sources of *timing anomalies* in dynamically scheduled processors [60].

To reconciliate performance and predictability of caches, we have designed algorithms for software control of instruction caches. The proposed algorithms statically divide the tasks code into regions, for which the cache contents is statically selected. At run-time, at every transition between regions, the cache contents computed off-line is loaded into the cache and the cache replacement policy is disabled (the cache is *locked*). Two algorithms have been designed. First, a greedy algorithm selecting cache contents according to knowledge of references along the program worst-case execution path [18], [32]. It selects cache contents by exploiting execution frequencies of basic blocks along the worst-case execution path. The second algorithm is a genetic algorithm[32] exploring the search space (reload points + cache contents) in a blind manner.

We then proposed an efficient hybrid algorithm. The greedy algorithm is used to select the initial population for a genetic search. Experimental results showed that the WCET estimates of applications with locked instruction caches is close to the WCET estimates considering caches without software control (and in some cases even better), as far as applications exhibit temporal locality.

6.3.2.2. Compiler-controlled management of scratchpad memories

An alternative to caches for on-chip storage is scratchpad memory. Scratchpad memories are small on-chip static RAMs that are mapped onto the address space of the processor at a predefined address range. Their inherent predictability have made them popular in real-time systems. The allocation of code/data memory to the scratchpad memory is under software control.

Significant effort has been invested in developing efficient allocation techniques for scratchpad memories. However, except [67], all these techniques aim to reduce the average execution time (ACET) of programs, through memory access profiles. To the best of our knowledge, no WCET-oriented dynamic scratchpad allocation method has been proposed so far. In [31] we propose a generic off-line algorithm for allocating *code* portions in on-chip memory. This algorithm supports both sratchpad memories and locked caches. It introduces multiple load points in the code of a single task and selects the values to be loaded at run-time into the on-chip memory. The algorithm is WCET oriented in the sense that it aims at minimizing the task WCET estimate. Experimental results show that the worst-case performance of applications using the two types of memory are very close to each other in most cases.

We are currently working on a WCET-oriented approach for dynamic *data* allocation in programs.

6.3.2.3. Cache locking, cache analysis and schedulability

When preemptive scheduling is used, task preemptions incur a cache related preemption delay to reload the cache after the preemption point. In schedulability analyses this delay must be taken in account in WCET estimates. The maximum additional time for reloading replaced cache blocks is a function of the maximum size of the interference zone of the two tasks

Our objective here is to compare different analysis approaches including cache partitioning, cache locking and cache-aware schedulability analysis approaches. A common evaluation framework including tools from IRISA (Heptane, algorithms for locked caches and scratchpad allocation) and from Technical University Braunschweig (Sympta/P, static cache analysis) was developed in october 2006 and is currently used for the comparison.

6.4. Applying microarchitecture knowledge to cryptography related problems

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

Participants: André Seznec, Olivier Rochecouste.

The research described in this section is done in cooperation with the Inria Rocquencourt CODES project (Nicolas Sendrier and Cédric Lauradoux) in the framework of ACI sécurité project UNIHAVEGE (cf. 7.4).

6.4.1. HAVEGE

Unpredictable random number generators, i..e., a practical approximation of a truly random number generator, are 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.

HAVEGE is now available as a driver name /dev/hrandom for the Linux operating system. It is also provided as a static or dynamic for Linux and Windows. A prototype has been developed for PalmOS (cf. 5.3).

6.4.2. Cache side channel attacks on AES

Recent studies [43] have shown that on some PC platforms part of the key of the AES encryption software can be recovered through precise measure of the encryption time. In [36], we have shown that both the micro-architecture of the processor and the cache initial state impact the amount of side-channel information that can be recovered.

Our experiments also showed that these side-channel attacks are not effective when the look-up tables are periodically and randomly permuted (e.g. every 256 encryptions). HAVEGE can be used to provide the random numbers needed. Computation time is only increased by a marginal 5%.

7. Contracts and Grants with Industry

7.1. Research grant from Intel

Participants: Thomas Piquet, André Seznec.

The researches on content conscious cache management (cf. 6.1.1), and on branch prediction (cf. 6.1.4) are partially supported by the Intel company through a research grant.

7.2. Start-up

Participants: François Bodin, André Seznec.

The collaboration has been pursued in 2006 with the start-up company CAPS Entreprise that was created in 2003 by members of the research team. This collaboration addresses topics such as very high performance code generation for complex processors (IA64 for instance) and compilation for ASIP.

7.3. QCDNext

Participants: François Bodin, André Seznec.

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

7.4. ACI Sécurité UNIHAVEGE (2003-2006)

Participants: Olivier Rochecouste, 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.

7.5. Sceptre project

Participants: François Bodin, Robin Schmutz.

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

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

7.6. Scalimages project

Participants: François Bodin, Karine Brifault.

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

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

7.7. Mascotte

Participants: Isabelle Puaut, Christophe Païs.

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

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

7.8. FAME2

Participants: François Bodin, Florence Dru, Guillaume Papaure.

The FAME2 project is part of Bull FAMEx program. The objective is to conceive the new multiprocessor servers generation for 2008. Shared memory multiprocessors will be associated as clusters to create high power servers that meet the requirements of the high performance computing (HPC) applications.

This project is funded by the "Pôle de compétitivité" Syst@matic.

7.9. PARA project

Participants: François Bodin, Sébastien Matz.

The objective of the PARA project is to study and develop optimization techniques in order to fully exploit all kind of parallelism found in modern computer architectures. This is done by associating different public and private research communities (application developers, compilation and operating system experts and system conceptors). The final objective is to globally increase the efficiency of selected codes found in different application fields/domains on next generation platforms as for instance high performance GPU 6.2.2.

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

7.10. GaLogic

Participants: François Bodin, Karine Brifault, Isabelle Puaut, Christophe Pais.

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

8. Other Grants and Activities

8.1. ApeNEXT project

Participants: François Bodin, Jérémy Fouriaux.

The ApeNEXT is the latest generation of massively parallel supercomputers with a multi-teraflops 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 [19].

8.2. NoEs

Participants: François Bodin, Pierre Michaud, Isabelle Puaut, 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.
- I. Puaut is an affiliated member of the Artist2 Network of excellence (Network of Excellence on Embedded Systems Design, http://www.artist-embedded.org/FP6/) in the cluster *Compilers and Timing Analysis*.

8.3. IP-Fet European project Sarc

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

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

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

9. Dissemination

9.1. Scientific community animation

- F. Bodin was has been a track chairman of ACM SAC'06 (the 21st ACM Symposium on Applied Computing 2006 conference, Dijon, France, April 2006), on the special track Embedded Systems: Applications, Solutions, and Techniques). He was a member of the program committee of SEDS06 workshop (Perspectives on Science and Engineering Driven Supercomputing, Bertinoro, Italy, May 2006, http://events.lal.in2p3.fr/conferences/seds06/). He has been a program vice-chairman of the Embedded Systems track in HPCC'06 (the International Conference on High Performance Computing and Communications, Munich, Germany, September 2006). F. Bodin was a co-editor of a special issue on HiPEAC Transactions (Transactions on High-Performance Embedded Architectures and Compilers).
- P. Michaud has been a member of the program comittee of HiPC 2006 conference.
- I. Puaut is a member of program committee of ECRTS06 and ECRTS07 (18th and 19th Euromicro Conference on Real-Time Systems, Dresden, Germany, July 2006, Pisa, Italy, July 2007), WCET06 (6th Workshop on WCET analysis, held in conjunction with ECRTS05), RTSS 2006 (27th IEEE Real-Time Systems Symposium, Rio de Janeiro, Brazil, december 2006), ISORC 2007 (10th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing, Santorini Island, Greece, May 2007) and CFSE5 (5th French Conference on Operating Systems, Perpignan, France, october 2006) She serves as program chair of RTNS 2007 (15th International Conference on Real-Time and Network systems, Nancy, France, March 2007). I. Puaut is member of the editorial board of Interstices (french on-line resources dedicated to the discovery of research in computer science, http://interstices.info/).
- A. Seznec has been a member of the program committee of ISPASS'07 and CBP-2. He was in charge of the architecture and hardware session at SESD'06 workshop (Bertinoro, may 2006). He is a member of the editorial board of the HiPEAC Transactions (Transactions on High-Performance Embedded Architectures and Compilers).

9.2. University teaching

F. Bodin and A. Seznec are teaching computer architecture and compilation at research master in computer sciences, at DIIC at IFSIC, University of Rennes I. I. Puaut teaches operating systems, real-time systems and real-time programming in the master degree of computer science of the University of Rennes I.

9.3. Workshops, seminars, invitations, visitors

- A. Seznec has presented a seminar on branch prediction at the university of Ghent in march 2006 entitled "Just a new case for Geometric History Length branch predictors".
- A. Seznec has been an invited speaker at the ScalPerf'06 workshop (Bertinoro, september 2006).
- A. Seznec has been an invited speaker at CEA-INRIA-EDF summer school in June 2006.
- F. Bodin has been an invited speaker at the ScalPerf06 workshop, in September 2006.
- F. Bodin has been an invited speaker at CEA-INRIA-EDF summer school in June 2006.
- F. Bodin has been an invited speaker at the HPC Bull convention in September 2006.
- E. Petit, F. Bodin, G. Papaure and F. Dru have presented a poster entitled *ASTEX: A Hot Path Based Thread Extractor for Distributed Memory System on a Chip* on SC'06 (the 19th International Conference for High Performance Computing, Networking, Storage and Analysis, Tampa, Florida, November 2006).

- E. Petit, S. Matz and F. Bodin have presented a poster entitled *Partitioning Program for automatically Exploiting GPU* on SC'06 (the 19th International Conference for High Performance Computing, Networking, Storage and Analysis, Tampa, Florida, November 2006).
- E. Petit and F. Dru and F. Bodin have hold a booth on SC'06 (the 19th International Conference for High Performance Computing, Networking, Storage and Analysis, Tampa, Florida, November 2006).

9.4. Miscelleanous

- F. Bodin is a member of the « commission des rapporteurs du RNTL à l'ANR »
- F. Bodin is a member of the « conseil scientifique du programme calcul intensif et grilles de calcul à l'ANR »
- 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 spécialistes" in computer science at Université de Bretagne Sud, Université de Rennes 1 and Université de Versailles.
- F. Bodin is a scientific advisor for the company CAPS entreprise.
- J. Lenfant is a member of "académie des sciences et des technologies".
- I. Puaut is responsible of 1st year Master in computer sciences at University of Rennes I.
- A. Seznec has been elected at the scientific comittee of INRIA in october 2006.
- J.-F. Deverge is an elected member of the Scientific Council of University of Rennes 1.
- CAPS is a member of the "pôle de compétitivité System@tic", the "pôle de compétitivité réseau image", and the "pôle de compétitivité Minalogic".
- J.F. Deverge and C. Païs are in the organizing committee of RTNS 2007, 15th International Conference on Real-Time and Network systems, Nancy, France, March 2007.

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

[3] R. DOLBEAU, A. SEZNEC. *CASH: Revisiting Hardware Sharing in Single-Chip Parallel Processors*, in "Journal of Instruction-Level Parallelism", vol. 6, April 2004.

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

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

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

[7] P. MICHAUD, Y. SAZEIDES, A. SEZNEC, T. CONSTANTINOU, D. FETIS. *An analytical model of temperature in microprocessors*, Technical report, IRISA, 2005.

[8] 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.

[9] 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.

[10] I. PUAUT, D. DECOTIGNY. *Low-complexity algorithms for static cache locking in multitasking hard real-time systems*, in "Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS' 02), Austin, Texas", December 2002.

[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 2003.

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

[16] 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.

[17] 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.

Year Publications

Doctoral dissertations and Habilitation theses

[18] A. ARNAUD. *Utilisation prévisible de caches d'instructions dans les sytèmes temps-réel strict*, Ph. D. Thesis, Université de Rennes I, April 2006.

Articles in refereed journals and book chapters

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

[20] T. CONSTANTINOU, Y. SAZEIDES, P. MICHAUD, D. FETIS, A. SEZNEC. *Performance Implications of Single Thread Migration on a Chip Multi-Core*, in "ACM SIGARCH Computer Architecture News", Nov 2005.

[21] K. HEYDEMANN, F. BODIN, P. KNIJNENBURG, L. MORIN. *UFS: a Global Trade-off Strategy for Loop Unrolling for VLIW Architectures*, in "Concurrency and Computation: Practice and Experience Journal", sept 2006.

[22] P. MICHAUD, Y. SAZEIDES, A. SEZNEC, T. CONSTANTINOU, D. FETIS. *A study of thread migration in temperature-constrained multi-cores*, in "ACM Transactions on Architecture and Code Optimization", To appear, 2006.

[23] I. PUAUT. *Chapitre 5, intitulé Méthodes d'obtention de majorants sur les temps d'exécution : état de l'art et perspectives*, in "Systèmes temps-réel, volume ordonnancement, réseaux et qualité de service", 2006.

[24] O. ROCHECOUSTE, G. POKAM, A. SEZNEC. *A Case for a Complexity-Effective, Width-partitioned Microarchitecture*, in "ACM Transactions on Architecture and Code Optimization", September 2006.

[25] A. SEZNEC, P. MICHAUD. *A case for (partially) TAgged GEometric history length predictor*, in "Journal of Instruction Level Parallelism", Feb 2006, http://www.jilp.org/vol8.

Publications in Conferences and Workshops

[26] D. FETIS, P. MICHAUD. *An evaluation of HotSpot-3.0 block-based temperature model*, in "5th Annual Workshop on Duplicating, Deconstructing, and Debunking", June 2006.

[27] K. HEYDEMANN, F. BODIN. *Iterative Compilation for two Antagonistic Criteria : Application to Code Size and Performance.*, in "4th Workshop on Optimizations for DSP and Embedded Systems", march 2006.

[28] E. PETIT, F. BODIN. *Extracting Threads Using Traces for System on a Chip*, in "Compilers for Parallel Computers (CPC2006), Coruna, Spain", January 2006.

[29] E. PETIT, S. MATZ, F. BODIN. *GPGPU with ASTEX: Hot Path Based Thread Extractor for Distributed Memory System*, in "General-Purpose GPU Computing: Practice and Experience Workshop, Tampa, Florida, USA", November 2006.

[30] E. PETIT, G. PAPAURE, F. DRU, F. BODIN. *ASTEX: a Hot Path Based Thread Extractor for Distributed Memory System on a Chip*, in "High-Performance Embedded Architecture and Compilation Industrial Workshop (HiPEAC), Grenoble, France", May 2006.

[31] I. PUAUT, C. PAÏS. *Scratchpad memories vs locked caches in hard real-time systems: a quantitative comparison*, in "DATE 2007 (Design, Automation and Test, Europe), Nice, France", To appear, April 2007.

[32] I. PUAUT. *WCET-Centric Software-controlled Instruction Caches for Hard Real-Time Systems*, in "18th European Conference on Real-Time Systems, Dresden, Germany", July 2006.

[33] A. SEZNEC. *A 256 Kbits L-TAGE branch predictor*, in "The 2nd JILP Championship Branch Prediction Competition (CBP-2)", december 2006.

[34] A. SEZNEC. *Looking for limits in branch prediction with the GTL predictor*, in "The 2nd JILP Championship Branch Prediction Competition (CBP-2)", december 2006.

[35] H. VANDIERENDONCK, A. SEZNEC. *Fetch gating control through speculative instruction window weighting*, in "2nd HIPEAC conference, january 2007", To appear, 2006.

Internal Reports

[36] A. CANTEAUT, C. LAURADOUX, A. SEZNEC. *Understanding cache attacks*, Technical report, n^{o} 5881, INRIA, May 2006.

[37] L. HE, R. DOLBEAU, A. SEZNEC. *CASH Design Space Exploration*, Technical report, n^{o} 5994, INRIA, October 2006.

[38] P. MICHAUD, Y. SAZEIDES. *Scheduling issues on thermally-constrained processors*, Technical report, n^{o} PI-1822, IRISA, October 2006.

[39] T. PIQUET, O. ROCHECOUSTE, A. SEZNEC. *Minimizing Single-Usage Cache Pollution for Effective Cache Hierarchy Management*, Technical report, n^{o} PI-1826, IRISA, November 2006.

[40] I. PUAUT, C. PAIS. *Etat de l'art des méthodes d'estimation statiques de WCET pour les logiciels embarqués automobiles*, Technical report, n^{o} PI-1818, IRISA, October 2006.

[41] R. SCHMUTZ, K. BRIFAULT, F. BODIN. *A structural model for WCET estimation of Simple Out-of-Order Superscalar Processor*, Technical report, IRISA/INRIA Rennes, September 2006.

References in notes

[42] *HotSpot*, http://lava.cs.virginia.edu/HotSpot/.

[43] D. J. BERNSTEIN. *Cache-timing attacks on AES*, Revised version of earlier 2004-11 version, April 2005, http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

[44] 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.

[45] 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.

[46] 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.

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

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

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

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

[51] R. HECKMANN, M. LANGENBACH, S. THESING, R. WILHELM. *The Influence of Processor Architecture on the Design and the Results of WCET Tools*, in "Proceedings of the IEEE", vol. 91, n^{o} 7, July 2003.

[52] 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.

[53] 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.

[54] 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.

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

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

[57] 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.

[58] 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.

[59] 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.

[60] T. LUNDQVIST, P. STENSTRÖM. *Timing Anomalies in Dynamically Scheduled Microprocessors*, in "IEEE Real-Time Systems Symposium", 1999, p. 12-21.

[61] 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.

[62] 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.

[63] M. POWELL, M. GOMAA, T. VIJAYKUMAR. *Heat-and-run: leveraging SMT and CMP to manage power density through the operating system*, in "Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems", 2004.

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

[65] 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.

[66] 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.

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

[68] C. TICE, S. GRAHAM. *Key Instructions: Solving the Code Location Problem for Optimized Code*, 2000.

[69] 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.

[70] 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.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

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