INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team caps

Compilation, architectures des processeurs superscalaires et spécialisés

Rennes

THEME COM

Activity Report

2005

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]
Isabelle Puaut [Professor, University of Rennes 1]

Technical staff Inria
Olivier Rochecouste [from 01/12/05]

Junior technical staff Inria
Jerémy Fouriaux [from 09/01/05]

Inria Postdoctoral fellow
He Liqiang [from 09/13/05]

Ph.D. students
Arnaud Alexis [Inria allocation]
Jean-François Deverge [MENRT allocation]
Assia Djabelkhir [French-Algerian allocation, till 03/31/05]
Damien Fétis [MENRT allocation]
Karine Heydemann [teaching assistant till 08/31/05]
Max Lukyanov [coadvised with DESY, Berlin]
Eric Petit [Inria allocation, from 09/01/05]
Thomas Piquet [Inria allocation]
Olivier Rochecouste [Inria allocation, till 01/31/05]
Eric Toullec [teaching assistant, till 08/31/05]

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, and 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[3], [10], and we are still very active in these areas (cf. 6.1.2, 6.1.3). We also aim at reducing the hardware costs of implementing wide-issue superscalar processors [6], [13] (cf. 6.1.4) while we pursue researchs on thread level parallelism on a single chip (cf. [1], 6.1.5). On embedded systems, performance must be traded against hardware cost, therefore we are exploring the adequation of dynamic execution on embedded applications (cf. 6.1.9). At the same time, power consumption and temperature hot spot management have become major issues for all processors. We have initiated a resarch activity on temperature management at architectural level (cf. 6.1.6). We are also studying how the compiler and the architecture can interact to optimize the power consumption/performance tradeoff (cf. 6.1.8, [7]).

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. 6.2.2). 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. 6.2.1).

In real-time systems, predicting the response time of an embedded software is of prime importance. 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 times of applications on a given hardware platform. Such platform now features 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 augmenting 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, ACI Securité UNIHAVEGE). 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. 7.2).

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 (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 [50]. 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 [47][10]. Both branch directions and targets (even indirect jump targets) [38] are predicted. Most of these predictors exploit either local or global branch history. The accuracy of the prediction seems to be reaching a plateau. New prediction paradigms exploiting other information sources are probably needed to allow new major prediction accuracy gains.

The complexity of many components in the processor (in terms of silicon area, power consumption and response time) increases superlinearly (and often quadratically) with the issue width e.g. register renaming, instruction scheduling, bypass network and register file access. These components are becoming the bottlenecks that limit the issue width and the cycle time [55].

While the complexity of the processors is steadily increasing, predicting, understanding and explaining the effective behavior of the architecture is becoming a major issue, in particular for embedded systems. Unfortunately, high performance 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, 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 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 [62]. 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, 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 [57] by redundant execution), or more performance. A few ideas have been recently proposed to speed-up sequential execution, like for instance speculative threads [43], exception handling [67], helper threads for branch prediction [39], helper threads for memory prefetching [51], etc. Among solutions already proposed, it is not yet clear which are viable and which are not. It will depend on the performance gain / hardware complexity tradeoffs. Ongoing research on this topic will decide the scope of future SMT/multicore processors.

3.4. Compiling and optimizing for embedded applications

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

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

Therefore embedded processors are presenting many new challenges [46] to the hardware and compiler research community.

Code optimization for embedded processors does not directly fit in the traditional "best speed effort at any price” assumption used for supercomputers and workstations. First, the “common case” paradigm using a set of representative benchmarks (e.g., SPEC2000) for general-purpose processor systems is not relevant for the design of compiler optimizations for an embedded processor: one must concentrate on the few optimizations that will bring performance on the few relevant target applications. Second, execution time is not always the only and ultimate criteria. In many cases, execution time may be less important than memory size or power consumption. Third, binary compatibility, while often important, 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 [66], [41] is a major issue in many applications. For small micro-controllers, “the smallest code, the fastest” is an effective rule of thumb. However, for recent embedded processors featuring instruction level parallelism (e.g., VLIW processors), faster code generally means larger code size [5]. To master code size, code compression techniques [36] 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 which behavior is fundamentally difficult to predict [58]. A better synergy between compilers and hardware must be set up and supported by performance debugging tools.

Power consumption is becoming a major issue on most processors. For a given processor, power consumption is highly related to performance: in most cases, a compiler optimization reducing execution time also reduces power consumption [61]. A more interesting issue arises with configurable hardware, for instance cache memories that can vary in size or associativity. In that case, the compiler can tradeoff performance against power consumption [64], [63].

While many optimizations and code transformations have been proposed over the past two decades, the interactions between these optimizations are not really understood. The many optimizations used in modern compilers sometimes annihilate eachother [40], [49]. Performance tuning is therefore an important and time consuming task. For embedded systems, developers must perform this tuning while preserving code size or power consumption. New software environments must be designed for this performance tuning [45], [54], [65]. An associated challenge is to preserve the link between aggressively optimized low level code and the source code [60]. As an alternative (or a complement) to performance tuning, automatic iterative compilation techniques [48] 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 [9] as well as fast processor simulation are key issues to support design exploration. Compiled simulation [2] 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

4.1. Application Domains

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

The Caps team is working on the foundation technologies for computer science: processor architecture and performance oriented compilation. The research results have impacts on any application domain that requires high performance executions (telecommunication, multimedia, biology, health, engineering, environment, ...), 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 present here **HAVEGE**, a software developed by the team. **HAVEGE** is freely distributed for non-commercial use.

5.2. 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 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 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.

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

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

6. New Results

6.1. Processor Architecture

Keywords: *Processor, branch prediction, cache, locality, memory hierarchy, multicore, simultaneous multi-threading.*

Participants: François Bodin, Assia Djabelkhir, Damien Fétis, He Liqiang, Pierre Michaud, Thomas Piquet, Olivier Rochecouste, André Seznec, Eric Toullec.

Our research in computer architecture covers memory hierarchy, branch prediction, superscalar implementation, as well as SMT and multicore issues. In the recent past, we have proposed several new complexity-effective cache and branch predictor structures [3], [10]. We are still refining, analyzing and exploring new cache management policies (cf. 6.1.2). New new directions in branch prediction are explored (cf. 6.1.3). We are also trying to reduce the hardware costs of implementing wide-issue superscalar processors [6], [13] (cf. 6.1.4) while continuing ongoing research on thread level parallelism on a single chip (cf. 6.1.5).

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. 6.1.6). We are exploring power consumption reduction through two directions, exploiting the dynamic width of operands (cf. 6.1.8) and using branch confidence for fetch gating (cf. 6.1.7)

6.1.1. Conflict free accesses to strided vectors on a banked cache

Participant: André Seznec.

With the advance of integration technology, it has become feasible to implement a microprocessor, a vector unit and a multimegabyte bank-interleaved L2 cache on a single die.

Parallel access to strided vectors on the L2 cache is a major performance issue on such vector microprocessors. A major difficulty for such a parallel access is that one would like to interleave the cache on a block size basis in order to benefit from spatial locality and to maintain a low tag volume, while strided vector accesses naturally work on a word granularity.

Considering a parallel vector unit with 2^n independent lanes, a 2^n bank interleaved cache and a cache line size of 2^k words, we show that, any slice of 2^{n+k} consecutive elements of any strided vector with stride $2^r R$ with R odd and $r \leq k$ can be accessed in the L2 cache and routed back to the lanes in 2^k subslices of 2^n elements [22].

This work was done in collaboration with Roger Espasa, Intel and UPC Barcelona.

6.1.2. Content conscious management of the memory hierarchy

Participants: Thomas Piquet, André Seznec.

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

We are initiating a study on such a conscious management of L2 and L3 caches. We want to focus on maintaining the "useful" blocks in the whole memory hierarchy as a complementary approach to prefetching. From the performance perspective, it is more important to maintain blocks that are difficult to predict in the memory hierarchy rather than blocks that can be prefetched in time. Therefore if a memory block can always be prefetched in time, it can be assigned a low priority for the replacement policy. Our focus is then on defining new replacement policies for caches that try to maintain “hard-to-prefetch” blocks in the memory hierarchy.

As a first step, we have defined a new replacement policy to take into account the usage of a memory block instead of the classical LRU or pseudo LRU policies. We focus now on combining this policy with prefetching.

6.1.3. Branch prediction

Participants: Pierre Michaud, André Seznec.

During the two past years, we have explored global history predictors in two directions.

First, we have introduced and analyzed [23], [30] the Optimized GEometric History Length (O-GEHL) branch predictor that efficiently exploits very long global histories in the 100-200 bits range. The O-GEHL predictor features several predictor tables $T(i)$ (e.g. 8) indexed through independent functions of the global branch history and branch address. The set of used global history lengths forms a geometric series, i.e., $L(j) = \alpha^{j-1}L(1)$. This allows the O-GEHL predictor to efficiently capture correlation on recent branch outcomes as well as on very old branches. As on perceptron predictors, the prediction is computed through the addition of the predictions read on the predictor tables.

We have also investigated a top-down approach for inventing new predictor schemes. The method most often used for inventing branch predictors is to start from a known predictor and try to improve it, based on some new intuition. However, when this method fails to decrease the mispredict rate, it is difficult to analyze the reasons for this failure, precisely because the mispredict rate does not change. We have proposed a new approach, which consists in first defining a model of ideal predictor, and then introducing successive degradations corresponding to hardware constraints, until we obtain a realistic predictor. On each degradation, it is possible to quantify the loss, analyse the reasons for it, and sometimes propose remedies. We applied the method on the family of tag-based predictors derived from PPM predictor (predicting by partial matching) and, with the help of the method, we were able to obtain new insights and propose improvements to a tag-based predictor [21].

The O-GEHL predictor and the tagged PPM predictors were recognized as very efficient branch predictors, since they respectively won the 2nd and the 5th places at 1st ChampionShip Branch Prediction workshop held in Portland in december 2004 (http://www.jilp.org/cbp). They were also the only two presented competitors with a reasonable hardware implementation cost.

We are currently pursuing further studies on branch predictors in order to mix the use of geometric history lengths and tagged partial matching approach.

6.1.4. Mastering hardware complexity on wide-issue supercalar processors

Participants: 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 [13], 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 have further explored instruction allocation policies on functional unit clusters, the various cluster configurations and the benefits of integrating the simultaneous multithreading paradigm into our WSRS architecture [17].

6.1.5. Resource sharing on 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) [62] and CMP (Chip MultiProcessor) [43] approaches were first investigated by academics and are now implemented by the industry. In some sense, CMP and SMT represent two extreme design points. In [4], 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 resource sharing à la SMT 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 intend to further study the possible resource sharing on single chip parallel processors.

6.1.6. 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 [42], [59]. 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.

The temperature problem in multi-cores can be alleviated by migrating threads from core to core [44], [56]. This is particularly interesting when there are less threads than cores. Preliminary studies indicate that the performance penalty for a single thread running on a multi-core can be tolerated provided the migration interval is at least several tens of thousands of cycles [26]. However, modeling such temperature-constrained multi-core is difficult, as it requires modeling a realistic multi-core microarchitecture, the chip layout, power consumption, temperature, and multiple threads running concurrently.

We are currently developing such a simulation infrastructure in collaboration with Pr Sazeides' team from University of Cyprus. This development requires updating and connecting existing tools, as well as developing new tools. In particular, we have developed two temperature models. One model is based on analytical methods [33]. We use this model for exploration and analysis. With this analytical model, we have shown that restricting the thread migration frequency to at most one per millisecond still allows to exploit most of the potential thermal benefit of thread migrations. The second temperature model we have developed is based on finite differences and takes into account more details concerning the physical system. We have connected this second temperature model with an existing power consumption model, and have started to run simulations.

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

Participants: Pierre Michaud, André Seznec.

Modern microprocessors feature moderate issue width (4 to 6) associated with very deep pipeline to enable high performance. They rely heavily on the usage of accurate branch predictors. However, the ratio of instructions executed on the wrong path is still very high and leads to useless power consumption on a single processor and/or suboptimal resource usage on SMT processors.

Fetch gating based on confidence estimation of the branch prediction allows to reduce the number of instructions executed on the wrong path without significantly impairing the performance, thus leading to more power effective architecture [52]. However only gshare-like branch predictors [53] have been considered.

Till this initial proposal, there has been only very few studies on the use of confidence estimation for reducing wrong path fetching and execution, despite substantial advances in branch prediction. Compared with gshare predictors, misprediction rates have been almost halved. New confidence estimators must be defined for these state-of-the-art branch predictors.

In collaboration with Hans Vandierendonk (post-doc at University of Ghent), we have begun a study on confidence estimation for fetch gating in conjunction of state-of-the-art branch prediction. This collaboration begun while Hans Vandierendonck was visiting the CAPS project for 3 months (april to june 2005).

6.1.8. *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. Moreover, these operations are well distributed across a program run. 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. Simple and efficient heuristics to steer the narrow-width operations towards clusters were defined. Using a WPM model saves power and area with a minimal impact on performance [16], [34].

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

6.1.9. *Embedded applications and 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 shown the adequation of a more limited form of dynamic execution, namely decoupled architecture, to embedded applications. Decoupled architecture is known to work very efficiently whenever the execution does not suffer from inter-processor dependencies causing some loss of decoupling, called LOD events. Regularity of codes are addressed in terms of the LOD events that may occur. We address three aspects of regularity: control regularity, control/memory dependency, and patterns of referencing memory data. We showed that most of kernels of MiBench suite of embedded benchmarks are amenable to efficient performance on a decoupled architecture.

We have proposed the implementation of a decoupled access architecture for standard RISC instruction set architecture, having single register file. For this implementation we define the dynamic code partitioning mechanism, inter-processor communication scenarios, and the validation mechanism. These mechanisms were validated through simulations [15].

6.2. Compilers and software environment for high performance embedded processors

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

Participants: François Bodin, Eric Petit, Jérémy Fouriaux, Karine Heydemann.

Some performance issues must be handled at higher level than the direct interface between the hardware and the instruction set. For heterogeneous SOCs featuring special purpose hardware and one or more execution cores, we are exploring the speculative thread extraction for the different hardware components (cf 6.2.1). Code size is often an issue on embedded systems. We are exploring tradeoffs based on code compression and interpretation (cf. 6.2.2). Domain specific processors require also 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

Participants: François Bodin, Eric Petit.

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

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

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

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

6.2.2. Performance Code Size TradeOff

Participants: Karine Heydemann, François Bodin.

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

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

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

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

We have investigated strategies for optimization under code size constraint. We have also defined a software compiler driven compression scheme. We have proposed a selective compression strategy under performance constraint to reduce code size with control over the performance decrease. We have also formalized the trade-off notion based on curves expressing the relation between code size and performance. The trade-off point is the point where the trend reverses: less and less performance benefit for more and more code size increase. The combination of trade-off computation and effective optimization strategies under constraints provides good trade-offs [29].

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

6.2.3. Code generation for the APEnext project

Participants: François Bodin, Jérémy Fouriaux, Max Lukyanov.

The apeNEXT is the latest generation of massively parallel supercomputers with a multi-TFlops performance dedicated to 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 asynchronous mode, it represents an array of processing nodes, where each node is an independent, VLIW controlled ASIC implementing all functionalities including network.

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

The software optimizer (SOFAN) has been implemented to serve as a test-bed for the exploration of different optimization strategies for the apeNEXT architecture, in particular the study of the phase coupling in the code generation. It also addresses the high irregularity of the apeNEXT architecture by realizing target dependent and state-of-the-art optimization techniques [19].

6.3. WCET analysis

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

Predicting the amount of resources required by embedded software is of prime importance for verifying that the system will fulfill its real-time and resource constraints. A particularly important point in the framework of hard real-time embedded systems is to predict the Worst-Case Execution Times (WCETs) of tasks, so that it can be proven that task temporal constraints (typically, deadlines) will be met. Our research concerns methods for obtaining automatically upper bounds of the execution times of applications on a given hardware. A particular focus is put on hardware-level analysis (static analysis based on timing models) and compiler-directed schemes aimed at augmenting predictability. In 2005, our studies concern WCET-oriented (as opposed to average-performance oriented) compilation and measurement-based WCET estimation.

6.3.1. WCET-oriented compilation

6.3.1.1. WCET-oriented static branch prediction

Participants: François Bodin, Isabelle Puaut, André Seznec.

Branch prediction mechanisms are becoming commonplace within current generation processors. Dynamic branch predictors, albeit able to predict branches quite accurately in average, are becoming increasingly complex (e.g. [30], [10]. Thus, determining their worst-case behavior is getting increasingly difficult and error-prone, and may even be soon impossible for the most complex branch predictors. In contrast, static branch predictors are *inherently predictable*, to the detriment of a lower prediction accuracy.

In [24] we have proposed a WCET-oriented static branch prediction scheme. Unlike related work on compiler-directed static branch prediction, our scheme does not address program average-case performance (i.e. average-case branch misprediction rate) but addresses *worst-case program performance* instead (i.e. branch mispredictions which impact programs WCET estimates). The static branch prediction scheme is implemented using an iterative algorithm working on the program control flow graph. Conditional branches along the program worst-case execution path are systematically statically predicted, avoiding mispredictions on this path. By iterating this process until convergence, we reduce the program WCET estimate. Experimental results on a PowerPC 7451 architecture have shown that the estimated WCET can be decreased by a factor up to 21% (with an average improvement of 15%) compared with the method where all branches are conservatively considered as mispredicted.

6.3.1.2. WCET-oriented control of the memory hierarchy through cache locking

Participants: Isabelle Puaut, Alexis Arnaud.

Cache are used to bridge the gap between processor cycle time and main memory response time. Unfortunately, caches are sources of response time unpredictability , because of their dynamic and adaptive behavior. Thus they require special attention when they are in hard real-time systems. Cache-aware WCET analysis techniques are not always applicable or may be too pessimistic. An alternative approach for using caches in real-time systems is to lock their contents (i.e. disable cache replacement) such that memory access times and cache-related preemption times become predictable. In 2005, we have studied both *static* and *dynamic* instruction cache locking.

The study on static instruction cache locking was conducted in cooperation with the *Universitat Politèctica de València*, Spain. We have compared in [25] the performance of two algorithms for static instruction cache locking: one using a *genetic algorithm* for cache contents selection [37] and a former heuristic algorithm, called *reference-based* algorithm that we defined in [8]. *Reference-based* uses the string of memory references issued by a task on its worst-case execution path as the input of the cache contents selection algorithm. Experimental results show that (i) both algorithms behave identically with respect to the system worst-case utilization; (ii) the genetic algorithm behaves slightly better than the reference-based algorithm with respect to the average slack of tasks; (iii) our reference-based algorithm is much faster than the genetic algorithm.

For dynamic instruction cache locking, the set of instructions that are locked in the cache must be managed at run-time. This set of instructions must be determined in order to improve the WCET estimation. We have proposed a low-complexity algorithm to determine a set of locked cache contents and an associated set of cache reloading points. With regard to performance evaluation against a system without any instruction cache, a sharp improvement is observed on the execution time in the worst case, and in the average case as well. Moreover, for many cache parametrizations, the worst-case performance is comparable with results from instruction cache analysis. In some cases, cache locking even outperforms cache analysis.

6.3.2. Measurement-based WCET estimation

Participants: Jean-François Deverge, Isabelle Puaut.

Static WCET analysis techniques require a reliable timing model of the processor. Unfortunately, this model is rarely available. An alternative approach is to use measurements of program executions on real hardware (or a cycle accurate simulator) to obtain (dynamic) WCET estimates. In order to guarantee a robust WCET estimate, test generation methods have to cover all paths, which may be untractable in practice. In [28] we propose to employ structural testing methods on program fragments (clusters) to reduce the complexity of test-case generation. Second, we suggest to use program transformations and compiler techniques to reduce (ideally eliminate) the timing variability of program mesurements through the control of hardware mechanisms (cache, pipeline, branch predictors). We are currently studying compiler methods for controlling the memory hierarchy (data cache, scratchpad memory).

6.4. HAVEGE: generating empirically strong random numbers

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

Participants: André Seznec, Olivier Rochecouste.

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) [12] (cf. 5.2) is a user-level software unpredictable random number generator for general-purpose computers that exploits these modifications of the internal volatile hardware states as a source of uncertainty.

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

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

7. Contracts and Grants with Industry

7.1. Research grant from Intel

Participants: Eric Toullec, Thomas Piquet, André Seznec.

The researches on content conscious cache management (cf. 6.1.2), on branch prediction (cf. 6.1.3) and on register file structures (cf. 6.1.4) are partially supported by the Intel company through a research grant (Convention 4 01 C 0677 00 31308 06 1).

7.2. Start-up

Participants: François Bodin, Karine Heydemann, André Seznec.

The collaboration has been pursued in 2005 with the start-up company CAPS Entreprise that was created in 2003 by members of the research team. This collaboration addresses topics such as very high performance code generation for complex processors (IA64 for instance) and compilation for ASIP.

8. Other Grants and Activities

8.1. Zenon bilateral cooperation

Participants: Pierre Michaud, André Seznec.

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

8.2. APEnext project

Participants: François Bodin, Jérémy Fouriaux, Max Lukyanov.

The apeNEXT is the latest generation of massively parallel supercomputers with a multi-TFlops performance dedicated for particle physics simulations. It is developed in the framework of the APE project which is carried out by the collaboration of INFN (Italy), DESY (Germany) and University of Paris-Sud (France). Following the single program multiple data (SPMD) programming model, where the nodes of the machine run in a slightly asyncronous mode, it represents an array of processing nodes, where each node is an independent, VLIW controlled ASIC implementing all functionalities including network [19].

8.3. 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*.

8.4. 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.

9. Dissemination

9.1. Scientific community animation

- I. Puaut is a member of program committee of ECRTS05 (17th Euromicro Conference on Real-Time Systems, Palma de Mallorca, Spain, July 2005), WCET05 (5th Workshop on WCET analysis, held in conjunction with ECRTS05), RTS'05 (13th French Conference on Real-Time Systems, Paris, France, April 2005), CFSE4 (4th French Conference on Operating Systems, Le Croisic, France, April 2005),and EUC'2005 (2005 IFIP International Conference on Embedded And Ubiquitous Computing, Nagasaki, Japan, december 2005). She served as program chair of CFSE4 (4th French Conference on Operating Systems, Le Croisic, France, April 2005) and chaired the Work-In-Progress session of ECRTS05 [31]. Since july 2005, 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 ISCA'32.
- F. Bodin was co-chair of the ACM SAC'05 conference track EMBS (track on embedded systems). F. Bodin participated to the program committee of the workshop Sympa'05.

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 seminars on the HAVEGE random number generator to Thomson, Rennes, april 2005 and Oberthur Card Systems, october 2005. He gave a talk entitled “Thread-level parallelism: it's time now” at the colloquium for the 30 years of IRISA.

- I. Puaut was external examiner of the PhD thesis of Li Xianfeng, entitled “Microarchitecture modeling for timing analysis of embedded software”, National University of Singapore, october 2005.
- I. Puaut presented a tutorial on Worst-Case Execution Time Analysis at the 4th french summer school on real-time systems, Nancy, september 2005 [35].
- J.-F. Deverge gave a talk entitled ”Safe measurement-based WCET estimation” at RJCITR05 (Rencontres des Jeunes Chercheurs en Informatique Temps Réel 2005), Nancy, in September 2005.
- H. Vandierendonk, University of Ghent, spent 3 months with the project (april-june 2005). He worked on applying confidence estimation for fetch gating in the context of state-of-the-art branch predictors in collaboration with A. Seznec.
- T. Constantinou, Ph. D student at University of Cyprus, visited the group for two weeks in July 2005. He worked on processor temperature modeling in collaboration with P. Michaud.
- F. Bodin has been an invited speaker at the ScalPerf’05 workshop.

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 an elected member of the IFSIC Committee.
- F. Bodin is responsible of the Research Master in computer sciences at University of Rennes I and chairman for doctoral studies at Irisa.
- F. Bodin is vice-chairman of Ecole doctorale Matisse (http://www.irisa.fr/matisse).
- F. Bodin is member of the board of the fundation M. Métivier (http://www.fondation-metivier.org).
- F. Bodin is a member of the "Commission de 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 was an elected member of the evaluation committee of Inria till june 2005.
- J.-F. Deverge is an elected member of the Scientific Council of University of Rennes 1.
- CAPS is a member of the “pole de compétitivité System@tic”.

10. Bibliography

Major publications by the team in recent years

[1] *Exploiting the Cache Capacity of a Single-chip Multi-core Processor with Execution Migration*, IEEE Computer Society, 2004.

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

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

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

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

[6] 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.

[7] 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.

[8] 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 (RTSS02), Austin, Texas", December 2002, p. 114-123.

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

[10] 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.

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

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

[13] A. SEZNEC, 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.

[14] 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

[15] A. DJABELKHIR. *Etude de l'exécution dynamique et/ou spéculative et des processeurs enfouis: un cas d'étude de l'architecture découplée*, Ph. D. Thesis, University of Rennes I, March 2005.

[16] O. ROCHECOUSTE. *Architecture et bits significatifs*, Ph. D. Thesis, University of Rennes I, October 2005.

[17] E. TOULLEC. *Vers une nouvelle architecture des processeurs superscalaires à exécution dynamique*, Ph. D. Thesis, University of Rennes I, October 2005.

Articles in refereed journals and book chapters

[18] G. ANTONIU, J.-F. DEVERGE, S. MONNET. *How to bring together fault tolerance and data consistency to enable grid data sharing*, in "Concurrency and Computation: Practice and Experience", To appear, 2005.

[19] F. BODIN, ET AL.. *Computing for LQCD: apeNEXT*, in "IEEE Computing in Science and Engineering", to appear.

[20] J. C. HERNANDEZ, J. M. SIERRA, A. SEZNEC, A. IZQUERDO, A. RIBAGORBA. *The strict avalanche criterion randomness test to PRNG Analysis*, in "Mathematics and Computation in Simulation", Feb 2005.

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

[22] A. SEZNEC, R. ESPASA. *Conflict free accesses to strided vectors on a banked cache*, in "IEEE Transactions on Computers", july 2005.

[23] A. SEZNEC. *Genesis of the O-GEHL branch predictor*, in "Journal of Instruction Level Parallelism", April 2005, http://www.jilp.org/vol7.

Publications in Conferences and Workshops

[24] F. BODIN, I. PUAUT. *A WCET-oriented static branch prediction scheme for real-time systems*, in "Proc. of the 17th Euromicro Conference on Real-Time Systems", July 2005, p. 33-40.

[25] A. M. CAMPOY, I. PUAUT, A. P. IVARS, J. B. MATAIX. *Cache contents selection for statically-locked instruction caches: an algorithm comparison*, in "Proc. of the 17th Euromicro Conference on Real-Time Systems", July 2005, p. 49-56.

[26] T. CONSTANTINOU, Y. SAZEIDES, P. MICHAUD, D. FETIS, A. SEZNEC. *Performance Implications of Single Thread Migration on a Chip Multi-Core*, in "Workshop on Design, Architecture and Simulation of Chip Multi-

Processors (DasCMP)", 2005.

[27] J.-F. DEVERGE, S. MONNET. *Cohérence et volatilité dans un service de partage de données dans les grilles de calcul*, in "Actes des 16èmes Rencontres Francophones en Parallélisme, Architecture, Système et Composant, Le Croisic, France", April 2005, p. 47-55.

[28] J. DEVERGE, I. PUAUT. *Safe Measurement-based WCET Estimation*, in "Proc. of the 5th Workshop on Worst-Case Execution Time Analysis, held in conjunction with the 17th Euromicro Conference on Real-Time Systems", July 2005, p. 7-10.

[29] K. HEYDEMANN, F. BODIN, H.-P. CHARLES. *A Software-only Compression System for Trading-offs between Performance and Code Size*, in "Proceedings of SCOPES 2005, Dallas, USA", October 2005.

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

Internal Reports

[31] *Proceedings of the Work-In-progress Session of the 17th Euromicro Conference on Real-Time Systems*, Technical report, n° PI-1723, IRISA, June 2005, http://www.irisa.fr/bibli/publi/pi/2005/1723/1723.html.

[32] F. B. ERIC PETIT. *Extracting Speculative Threads Using Traces for System on a Chip*, Technical report, n° 1789, 2005.

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

[34] O. ROCHECOUSTE, G. POKAM, A. SEZNEC. *A Case for a Complexity-Effective, Width-partitioned Microarchitecture*, Research report, n° 1732, IRISA, August 2005.

Miscellaneous

[35] I. PUAUT. *Méthodes de calcul de WCET (Worst-Case Execution Times) Etat de l'art*, September 2005, http://etr05.loria.fr/.

Bibliography in notes

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

[37] A. M. CAMPOY, A. P. IVARS, J. V. BUSQUETS-MATAIX. *Using genetic algorithms in content selection for locking-caches*, in "Proceedings of the IASTED International Symposium on Applied Informatics, Innsbruck, Austria", February 2001, p. 271–276.

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

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

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

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

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

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

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

[45] C.-H. HSU, U. KREMER. *IPERF: A Framework for Automatic Construction of Performance Prediction Models*, in "In Workshop on Profile and Feedback-Directed Compilation (PFDC)", October 1998, http://citeseer.ifi.unizh.ch/650733.html.

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

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

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

[49] 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.

[50] 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.

[51] 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.

[52] S. MANNE, A. KLAUSER, D. GRUNWALD. *Pipeline Gating: Speculation Control for Energy Reduction.*, in "ISCA", 1998, p. 132-141.

[53] S. MCFARLING. *Combining Branch Predictors*, Technical report, n^o WRL-TN-36, Hewlett Packard Laboratories, June 14 1993, http://www.hpl.hp.com/techreports/Compaq-DEC/WRL-TN-36.html.

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

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

[56] 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.

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

[58] 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.

[59] 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.

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

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

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

[63] 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.

[64] 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.

[65] 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.

[66] 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.

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