INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team caps

Compilation, architectures des processeurs superscalaires et spécialisés

Rennes - Bretagne Atlantique

THEME COM

Activity Report

2007

Table of contents

1. Team 1
2. Overall Objectives 1
3. Scientific Foundations 2
3.1. Panorama 2
3.2. Uniprocess architecture 2
3.3. Exploiting task parallelism on a single chip: multicore and SMT processors 3
3.4. Compiling and optimizing for embedded applications 4
4. Application Domains 5
5. Software 5
5.1. Panorama 5
5.2. ATMI 5
5.3. HAVEGE 6
6. New Results 6
6.1. Processor Architecture 6
6.1.1. Content conscious management of the memory hierarchy 7
6.1.2. Null blocks management on the memory hierarchy 7
6.1.3. Performance guarantee on multicores 7
6.1.4. Scheduling issues on a heterogeneous single ISA multicore 8
6.1.5. Branch prediction 8
6.1.6. Tackling temperature issues 8
6.1.7. Confidence estimation and fetch gating using state-of-the-art branch predictors 9
6.2. Compilers and software environment for high performance embedded or special purpose architectures 9
6.2.1. Speculative thread extraction for SOCs with ASTEX 10
6.2.2. Automatically exploiting GPU with ASTEX 10
6.2.3. Enabling high performance applications on emerging architectures 10
6.2.3.1. APEnext 11
6.2.3.2. Porting highly demanding applications on hardware accelerators 11
6.2.3.3. H.264 SVC decoder for multicore architectures 11
6.3. WCET estimation 11
6.3.1. Structural models for out-of-order pipelined processors 12
6.3.2. WCET-oriented compilation 12
6.3.2.1. Compiler-controlled management of scratchpad memories 12
6.3.2.2. Cache locking, cache analysis and schedulability 13
6.3.2.3. Predictable virtual memory for real-time applications 13
6.4. Applying microarchitecture knowledge to cryptography related problems 13
7. Contracts and Grants with Industry 13
7.1. Research grant from Intel 13
7.2. Start-up 13
7.3. QCDNext 14
7.4. Sceptre project 14
7.5. Scalimages 14
7.6. Mascotte 14
7.7. FAME2 14
7.8. PARA 14
7.9. POPS 15
7.10. Serenitec: SEcurity analysis and Refactoring ENvironment for Internet TEChnology 15
7.11. GaLogic 15
8. Other Grants and Activities 15

8.1. ApeNEXT 15
8.2. NoEs 16
8.3. IP-Fet European project Sarc 16
9. Dissemination **16**
9.1. Scientific community animation 16
9.2. University teaching 16
9.3. Workshops, seminars, invitations, visitors 17
9.4. Miscelleanous 17
10. Bibliography **17**

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, until 15/07/07]

Technical staff Inria
Karine Brifault [until 30/09/07]
Florence Dru
Sylvain Leroy [from 01/10/07]
Sébastien Matz [until 30/09/07]
Christophe Pais [until 30/09/07]
Guillaume Papauré
Robin Schmutz

Junior technical staff Inria
Jerémy Fouriaux [until 31/08/07]

Inria Postdoctoral fellow
Khaled Ibrahim [from 01/06/07]
Jan Staschulat [from 01/05/07 to 31/07/07]
Guntur Surendra [until 30/11/07]

Ph.D. students
Jean-François Deverge [MENRT allocation]
Julien Dusser [Inria Allocation]
Damien Fétis [MENRT allocation]
Robert Guziolowski [Inria Allocation]
Damien Hardy [MENRT Allocation, from 01/10/07]
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], [8], and we are still very active in these areas (cf. 6.1.1, 6.1.5). We pursue researchs on architecture exploiting thread level parallelism on a single chip (cf. [4], 6.1.4). 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. 6.1.6). We are also studying how the compiler and the architecture can interact to optimize the power consumption/performance tradeoff [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. 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. 5.3) and for understanding side chanel attacks (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, ANR funded QCDnext, GaLogic, Para, Mascotte, and “Poles de compétitivités” funded Scalimages, Sceptre, Fame2, POPS and Serenitec). Some of the research prototypes developed by the project during the past few years have been transferred to industry through the CAPS Entreprise start-up (cf. 7.2).

3. Scientific Foundations

3.1. Panorama

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

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

3.2. Uniprocess architecture

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

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

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

The use of a complex memory hierarchy has been generalized over the past decade. On modern microprocessors, both software and hardware prefetching are now widely used to enable the on-time presence of data and instructions in the memory hierarchy. Highly efficient, but complex data hardware prefetch mechanisms, have been proposed to hide several hundreds of instruction slots [46]. 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 [42][8]. Both branch directions and targets (even indirect jump targets) [33] 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 [49].

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, multi-programmed 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 [56]. 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 [50] by redundant execution), or more performance. A few ideas have been recently proposed to speed-up sequential task execution, like for instance speculative threads [38], exception handling [61], helper threads for branch prediction [34], helper threads for memory prefetching [47], 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 [41] 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 [60], [37] 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 [3]. To master code size, code compression techniques [32] 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 [51]. 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 [55]. 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 [58], [57].

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 [36], [44]. 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 [40], [48], [59]. An associated challenge is to preserve the link between aggressively optimized low level code and the source code [54]. As an alternative (or a complement) to performance tuning, automatic iterative compilation techniques [43] address the interactions of optimizations through the use of feedback, to find efficient code transformation sequences.

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

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

4. Application Domains

4.1. Application Domains

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

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

5. Software

5.1. Panorama

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

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

5.2. ATMI

Keywords: *Microarchitecture temperature model*.

Participant: Pierre Michaud.

Contact : Pierre Michaud

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

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

We have developed an ad hoc temperature model, ATMI (Analytical Model of temperature in MIcroprocessors), for studying thermal behaviors over a time scale ranging from microseconds to several minutes. ATMI is based on an explicit solution to the heat equation [6] and on the principle of superposition. ATMI can model any power density map that can be described as a superposition of rectangle sources, which is appropriate for modeling the microarchitectural units of a microprocessor.

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

5.3. HAVEGE

Keywords: *Unpredictable random number generator*.

Participant: André Seznec.

Contact : André Seznec

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

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

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

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

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

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

6. New Results

6.1. Processor Architecture

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

Participants: Julien Dusser, Damien Fétis, Robert Guziolowski, Surendra Guntur, 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], [8]. We are still refining, analyzing and exploring new cache management policies (cf. 6.1.1). New directions in branch prediction have been explored (cf. 6.1.5). We have begun the exploration of new directions of multicore architectures (cf. 6.1.4).

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 the use of branch confidence for fetch gating (cf. 6.1.7)

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 [24], 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. We study two possible uses of such a prediction. First we propose a cache replacement policy that favours the eviction of single-usage blocks on the L2 cache. Second, we propose a bypass scheme where single-usage blocks are directly sent from main memory to the L1 cache. These two strategies increase the L2 cache efficiency and allow hard-to-prefetch memory references to remain into this cache hierarchy level. The bypass scheme achieves a slightly higher performance than the replacement policy. We finally evaluate our bypass scheme on a multi-core architecture for a multi-programmed workload. We observe that reducing the single-usage pollution has two interesting and orthogonal effects. First for some applications, reducing the single-usage pollution increases its own performance. Second for some workloads, reducing single-usage pollution for an application has a significant impact on the performance of the concurrent applications that benefit from the freed cache space.

6.1.2. Null blocks management on the memory hierarchy

Participants: Julien Dusser, André Seznec.

Runtime observations reveal that a significant amount of blocks are totally null in different levels of the memory hierarchy. These blocks, typically 64 bytes long, could be handled separately to improve overall performances.

In the main memory, a large amount of space is used to store null blocks, so we are studying a new memory organization to prevent storing these blocks in memory, thus artificially increasing the total amount of physical memory, but also avoiding wasting bandwidth on memory bus.

On conventional caches, null blocks are stored together with the other blocks, using precious memory area to store zeros. We are studying a dedicated cache for storing these zero blocks. This zero cache features only tags and no data. This zero content cache can map a very large amount of data with relatively low tag overhead.

6.1.3. Performance guarantee on multicores

Participant: Pierre Michaud.

Multicore processors have become mainstream. However, most applications are sequential and run on a single core. On current multicores, sharing resources such as caches and memory impacts performance of individual processes. Therefore application developpers can not guarantee the performance level that application users will experience on a given multicore platform.

We define the self-performance of a sequential application as the performance measured for one instance of the application on a symmetric run, i.e., when copies of the application are run on all the cores, using the same inputs. Intuitively, the developper would like the application user to experience a performance equal or higher than the self-performance. A multicore will be said to provide performance guarantee for sequential processes if the performance of any sequential application cannot be significantly lower than its self-performance.

Our first study addresses shared caches, as cache sharing is one of the greatest sources of performance variability in existing multicores. Existing multicores usually implement cache replacement policies akin to the least-recently-used policy, which permits exploiting the full cache capacity. However, these replacement policies do not permit to obtain performance guarantee. We have proposed some new hardware replacement policies that allow shared caches to be compatible with performance guarantee.

6.1.4. Scheduling issues on a heterogeneous single ISA multicore

Participants: Robert Guziolowski, André Seznec.

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

A novel intermediate approach consists in combining heterogeneous single ISA cores into one chip. Nevertheless, this architecture needs novel approaches for scheduling the work onto the heterogeneous cores. We propose a scheduling method which subdivides the cores into smaller groups called *core classes*, and then, basing on various core performance counters, automatically migrates tasks between those classes in order to find, in terms of robustness, a proper assignment of the tasks to cores. We show that this method obtains better results than naïve round-robin algorithm [19]. We intend to include in our further study the thermal and power related issues. Furthermore, we want to concentrate on fairness of scheduling algorithm, in order to fully support priorities of the processes.

6.1.5. Branch prediction

Participants: Pierre Michaud, André Seznec.

While most recent researches in branch prediction (including the O-GEHL predictor [10]) 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 [5]. Using GEometric history length as the O-GEHL predictor, the TAGE predictor 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. A version of TAGE [15] won 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. The GTL predictor [14] was defined to explore the limits of branch prediction accuracy. It combines a GEHL predictor, a TAGE predictor and a loop predictor. The GTL predictor won the idealistic track at the 2nd Championship Branch Prediction workshop (http://camino.rutgers.edu/cbp2/).

6.1.6. Tackling temperature issues

Participants: Pierre Michaud, André Seznec, Surendra Guntur, Damien Fétis.

Temperature has become a strong constraint on microarchitecture. This constraint will become stronger with technology feature size reduction. For maximizing performance, the thermal design power (TDP) in current processors is set according to average, instead of worst case, conditions. Hence it is possible to hit the temperature limit depending on applications characteristics and ambient temperature. This is why current processors have several on-chip thermal sensors. Power dissipation is throttled when temperature reaches the fixed limit, which decreases performance. Researchers are currently exploring ways to minimize the performance loss due to thermal throttling in future processors, for example by implementing temperature-aware process scheduling at the operating system level [45], [35], by implementing activity migration at the microarchitecture level [39]. In [13], we study thread migration to limit thermal throttling on multicores.

To explore these techniques, researchers use simulators. Existing cycle-accurate microarchitectural simulators, coupled with energy models, provide estimates of performance and power consumption. However, cycle accurate simulators are generally too slow to simulate more than a few seconds of execution. Hence researchers use sampling to obtain an accurate estimation of the long term performance and power consumption. Up to now, thermal throttling have been studied through integrating thermal models like HotSpot [29] or our ATMI model [22] (cf. 5.2) in these micro-architecturural simulators. However, sampling is not a valid method for thermal simulation because temperature at a given time depends on all past energy events.

We are currently exploring a new simulation method for taking into account thermal throttling. This method is based on decoupling cycle accurate and thermal simulations [28]. A cycle accurate simulator is used to generate energy traces representing the complete execution of applications. We use program phase classification [52] to speed-up trace generation and obtain compact traces. The thermal simulator takes energy traces as input and simulates thermal throttling and its impact on performance.

6.1.7. *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 [27], [16], 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.

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

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

Participants: François Bodin, Florence Dru, Jérémy Fouriaux, Khaled Ibrahim, Sylvain Leroy, Sébastien Matz, Guillaume Papauré, Eric Petit, Robin Schmutz, André Seznec.

High performance embedded platforms are now build 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 (Automatic Speculative Thread EXtractor). 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).

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

6.2.1. Speculative thread extraction for SOCs with ASTEX

Participants: François Bodin, Eric Petit, Robin Schmutz, 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 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.

This year ASTEX have been upgraded for robustness in order to facilitate the technological transfer to the industry. Other developments concerned the use of ASTEX, mostly with GPUs.

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) look very attractive to speed up programs, even if they are difficult to program. Porting code to the GPU is generally done manually. Our objective is to define and develop programming tools able to exploit GPUs in the context of general programming.

ASTEX is used to implement a dynamic analysis that detects speculative threads which contains computing kernels with a potential speedup on the GPU. However recognizing these kernels is not sufficient. The effective performance of GPUs as hardware accelerators on general-purpose applications is highly dependent on the communication overhead between the main memory and the GPU memory. Optimizing this communication is an error prone process. The programmer must decide when to prefetch, update or download the remote data while ensuring that the GPU data are coherent when the remote procedure call is performed on the GPU. We are studying an automatic technique for inserting software data prefetching and upload for compute intensive kernels remotely executed on a GPU. The proposed technique [23] relies on an hybrid approach that mixes speculative data collected via the thread analysis environment ASTEX and usual static program analysis. Our implementation works on C source codes and deals with pointer aliasing issues.

6.2.3. Enabling high performance applications on emerging architectures

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

6.2.3.1. APEnext

Lattice Quantum Chromodynamics (Lattice QCD) is a grand challenge problem that requires peta-flop capable machines. CAPS project-team has been involved in the APEnext project with physicists from INFN (Italy), DESY (Germany) and University of Paris-Sud (France) for several years. ApeNEXT is the latest generation of massively parallel supercomputers dedicated to lattice QCD simulations [1]. ApeNEXT 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 has been 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.

In 2007, a production version of SOFAN has been delivered to the users of APEnext.

6.2.3.2. Porting highly demanding applications on hardware accelerators

The ANR QCDNext project aims at proposing architecture of a low cost next generation computer systems for highly demanding scientific applications. Using Lattice QCD as our main benchmark, we explore the use of hardware accelerators for supercomputing.

Using GPUs as accelerators faces several challenges. GPUs are essentially SIMD machines. However when mapping an application on such a GPU, one must take in account parallelism degree, host/GPU communication overhead, thread ressources and control flow dependent computations. Our study on mapping Lattice QCD on GPUs [21] explores the impact of parallelization granularity on performance. Fine-grained threads are shown to achieve better performance. We describe code transformations enabling the SIMDization and present optimized communication pattern between the host and the GPU. An 8.3x performance speedup was observed on the code using both the GPU and the host over an optimized host code using the SSE2 ISA.

Currently, we are investigating the IBM Cell SPUs (Synergetic Processing Units) as accelerators. Different facilities are provided by the processor to supply data, as well as multiple programmatic choices can be explored.

6.2.3.3. H.264 SVC decoder for multicore architectures

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

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

6.3. WCET estimation

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

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

In 2007, 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, Guillaume Papauré, Isabelle Puaut.

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 the timing effects of the various components in the processors into account, 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 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.

This model was ported on the ST200 architecture.

6.3.2. WCET-oriented compilation

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

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. 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 [53], 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 [26] we propose a generic off-line algorithm for allocating *code* portions in on-chip memory. This algorithm supports both scratchpad 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.

In [18] we present a solution for WCET-oriented allocation of *data* in scratchpad memories. A static inter-procedural address analysis allows to compute a safe approximation of the referenced data (static data and stack-allocated data). The allocation problem (selection of the variables to be loaded into scratchpad memory and selection of load points) is formulated using Integer Linear Programming (ILP). One property of our allocation strategy is that the granularity of placement of memory transfers can be parametrized (function or basic block boundaries). This flexibility allows to trade-off between the complexity of allocation and its quality.

6.3.2.2. 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 into 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) is currently used for the comparison.

6.3.2.3. Predictable virtual memory for real-time applications

Use of virtual memory has been traditionally avoided in real-time systems, since it induces additional difficulties for timing analysis. However, many systems implement different functions through concurrent processes. On systems with a limited amount of available physical memory, demand-paging is then required. So far, attempts to provide real-time address spaces have focused on the predictability of virtual to physical address translation and do not implement demand-paging.

In [25] we propose a compiler approach to introduce a predictable form of paging, in which page-in and page-out points are selected at compile-time. This problem can be formulated as a graph coloring problem, analogous to register allocation in compilers. Since graph coloring is NP-complete, we have defined a new heuristic aiming at minimizing worst-case performance instead of average-case performance. Our experimental results show that such a compile time approach provides predictability without impairing performance. Furthermore, in [20] we reformulated the page allocation problem using Integer Linear Programming. This yielded to a better page allocation, but at a slightly higher allocation complexity.

6.4. Applying microarchitecture knowledge to cryptography related problems

Keywords: *cryptography*, *security*, *side channel attacks*.

Participant: André Seznec.

Recent studies [31], [30] have shown that on some PC platforms part of the key can be recovered for the AES or RSA encryption softwares through side-channel attacks. We have studied and analyzed these attacks. In particular, we have analyzed the Simple Branch Prediction Analysis attack [30]. This attack uses the Simultaneous Multithreading mode of the Pentium 4 HT. A branch instruction outcome (taken or not-taken) of a given process may be spied by an attacker process, even in user mode. We have been able to reproduce and analyze such an an attack on toy code. This comprehensive study was presented in the Interstice journal (http://interstices.info) and on http://www.irisa.fr/activity/new/007/branchpredictionattack004.

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. 6.1.5) 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 2007 with the start-up company CAPS Entreprise that was created in 2003 by members of the research team. This collaboration addresses topics such as very high performance code generation for complex processors (IA64 for instance) and compilation for ASIP.

7.3. QCDNext

Participants: François Bodin, Khaled Ibrahim, André Seznec.

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

7.4. Sceptre project

Participants: François Bodin, Robin Schmutz.

The goal of the project is to develop a toolkit for helping the implementation of multimedia algorithms on a reconfigurable multiprocessor network. In particular, this toolkit should help to explore and analyze hardware / software tradeoffs. The goal is to drastically reduce application port time and to obtain flexible realizations over a family of algorithms, thus increasing the lifespan of each development. This year has been focus on code partitioning to define co-processors to speedup multimedia applications.

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

7.5. Scalimages

Participants: François Bodin, Florence Dru.

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

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

7.6. 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.

7.7. 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.

7.8. PARA

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). This year we have focus the work to exploiting graphics processor capabilities and study H264 application code partitioning for hybrid multicore architecture.

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

7.9. POPS

Participants: François Bodin, Guillaume Papauré.

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

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

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

Participants: François Bodin, Sylvain Leroy.

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

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

7.11. GaLogic

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

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

Current work is the development a WCET analysis software for the ST200 processor.

8. Other Grants and Activities

8.1. ApeNEXT

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 [1].

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 has been a member of the committee of Smart 07, the 1st Workshop on Statistical and Machine learning approaches to ARchitectures and compilation. He was the publicity chair for the PACT 2007 symposium. He has been the local chair for Topic 4, High performance architectures and compilers at Europar 2007. He is a member of the editorial board of the scientific programming journal.

  F. Bodin has organized the INRIA booth at Supercomputing '07 in colaboration with Julie Paul, INRIA Rocquencourt.
- Pierre Michaud is a member of the organization committee of the 2008 CEA-EDF-INRIA Computing Summer school.
- I. Puaut is a member of program committee of ECRTS07 (19th Euromicro Conference on Real-Time Systems), RTSS 2007 (28th IEEE Real-Time Systems Symposium), ISORC 2007 (10th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing), DATE 2008 (Design Automation and Test Europe) and CFSE6 (6th French conference on operating systems, 2008), embedded systems track of HPCC07. 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 committees of PACT 2007, HiPEAC'2008, HPCA-2008, MULTIPROG 2008 and MICRO 2008 top picks. 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. She teaches real-time systems in the BSc degree *Embedded automotive systems*.

9.3. Workshops, seminars, invitations, visitors

- F. Bodin has given invited presentations at the 8th The International Workshop on Performance Analysis and Optimization of High-End Computing Systems in November 2007, STIC' 2007 in November 2007, forum ORAP 07 in june 2007, and CTHPC 2007 in Taiwan in March 2007.
- A. Seznec has presented a seminar on branch prediction at University of Wisconsin at Madison in october 2007 entitled "All you will never have wanted to know on branch prediction".
- A. Seznec has been a keynote speaker at Europar 2007, August 31, Rennes. His talk was entitled "15 mm x 15 mm: the new frontier for parallelism"
- A. Seznec has presented a seminar entitled "Vulnerabilies on high -end processors" in the context of the GIS Diwall in June 2007. in june 2007,
- P. Michaud has given a course entitled "Architecture des processeurs généralistes haute-performance" at the thematic school ARCHI07.
- I. Puaut has given a tutorial on worst-case execution time estimation at the Universidad Complutense de Madrid, March 2007
- D. Hardy has presented his work on predictable paging at the PhD session of french summer school on real-time systems, Nantes, september 2007

9.4. Miscelleanous

- F. Bodin is 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 is an elected member of the scientific comittee of INRIA.
- A. Seznec is a member of the « conseil scientifique du programme calcul intensif et grilles de calcul à l'ANR »
- 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] F. Belletti, S. F. Schifano, R. Tripiccione, F. Bodin, P. Boucaud, J. Micheli, O. Pene, N. Cabibbo, S. de Luca, A. Lonardo, D. Rossetti, P. Vicini, M. Lukyanov, L. Morin, N. Paschedag, H. Simma, V. Morenas, D. Pleiter, F. Rapuano. *Computing for LQCD: ApeNEXT*, in "Computing in Science and Engineering", vol. 8, n^{o} 1, 2006, p. 18–29.

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

[3] 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.

[4] 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.

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

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

[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] 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.

[9] 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.

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

[11] 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.

Year Publications

Articles in refereed journals and book chapters

[12] K. D. BOSSCHERE, W. LUK, X. MARTORELL, N. NAVARRO, M. O'BOYLE, D. PNEVMATIKATOS, A. RAMIREZ, P. SAINRAT, A. SEZNEC, P. STENSTRÖM, O. TEMAM. *High-Performance Embedded Architecture and Compilation Roadmap*, in "Transactions on High-Performance Embedded Architectures and Compilers", January 2007, p. 5-29.

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

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

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

[16] H. VANDIERENDONCK, A. SEZNEC. *Fetch gating control through speculative instruction window weighting*, in "Transactions on High-Performance Embedded Architectures and Compilers", july 2007, p. 19-39.

[17] R. WILHELM, J. ENGBLOM, A. ERMEDAHL, N. HOLSTI, S. THESING, D. WHALLEY, G. BERNAT, C. FERDINAND, R. HECKMANN, F. MUELLER, I. PUAUT, P. PUSCHNER, J. STASCHULAT, P. STENSTRÖM. *The Determination of Worst-Case Execution Times—Overview of the Methods and Survey of Tools*, in "ACM Transactions on Embedded Computing Systems (TECS)", 2007.

Publications in Conferences and Workshops

[18] J. DEVERGE, I. PUAUT. *WCET-directed dynamic scratchpad memory allocation of data*, in "Proc. of the 19th Euromicro Conference on Real-Time Systems, Pisa, Italy", July 2007, p. 179–190.

[19] R. GUZIOLOWSKI, A. SEZNEC. *Scheduling Issues on a Heterogeneous Single ISA Multicore*, in "Poster Session at ACACES Summer School 2007", July 2007.

[20] D. HARDY, I. PUAUT. *Predictable paging in real-time systems: an ILP formulation*, in "Ecole d'été Temps Réel, session doctorants, Nantes, France", September 2007.

[21] K. Z. IBRAHIM, F. BODIN, O. PENE. *Fine-grained Parallelization of Lattice QCD Kernel Routine on GPU*, in "First Workshop on General Purpose Processing on Graphics Processing Units", October 2007, http://www.ece.neu.edu/GPGPU/.

[22] P. MICHAUD, Y. SAZEIDES. *ATMI: analytical model of temperature in microprocessors*, in "Third Annual Workshop on Modeling, Benchmarking and Simulation", june 2007.

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

[24] T. PIQUET, O. ROCHECOUSTE, A. SEZNEC. *Exploiting Single-Usage for Effective Memory Management*, in "Proceedings of the Asia-Pacific Computer Systems Architecture Conference 2007", august 2007, p. 90-101.

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

[26] I. PUAUT, C. PAIS. *Scratchpad memories vs locked caches in hard real-time systems, a quantitative comparison*, in "Design Automation and Test Europe (DATE), Nice, France", April 2007, p. 1484–1489.

[27] H. VANDIERENDONCK, A. SEZNEC. *Fetch gating control through speculative instruction window weighting*, in "proceedings of the 2nd HIPEAC conference, LNCS 4327", January 2007, p. 120-15.

Internal Reports

[28] S. GUNTUR, P. MICHAUD. *Decoupled Thermal Simulation*, Technical report, n^{o} 1871, IRISA, november 2007.

References in notes

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

[30] O. ACIIÇMEZ, Ç. K. KOÇ, J.-P. SEIFERT. *On the power of simple branch prediction analysis*, in "ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security, New York, NY, USA", ACM, 2007, p. 312–320.

[31] 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.

[32] 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.

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

[34] 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.

[35] J. CHOI, C.-Y. CHER, H. FRANKE, H. HAMANN, A. WEGER, P. BOSE. *Thermal-aware task scheduling at the system software level*, in "Proceedings of the International Symposium on Low Power Electronics and Design", 2007.

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

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

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

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

[40] 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.

[41] 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.

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

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

[44] 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.

[45] A. KUMAR, L. SHANG, L.-S. PEH, N. JHA. *HybDTM : a coordinated hardware-software approach for dynamic thermal management*, in "Proceedings of the 43rd Design Automation Conference", 2006.

[46] 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.

[47] 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.

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

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

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

[51] 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.

[52] T. SHERWOOD, E. PERELMAN, G. HAMERLY, B. CALDER. *Automatically characterizing large scale program behavior*, in "Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems", 2002.

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

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

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

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