CHAMPP: CHalmers Adaptable Multicore Processing Project

Per Larsson-Edefors, Sally A. McKee, Per Stenström, Lars Svensson

Abstract

Performance growth of single processor systems recently stalled because of power and heat problems associated with increasing the clock frequency. In order to continue to deliver a doubling of computational performance every 18 months, the entire computer industry opted for doubling the number of processor cores in each new processor generation. Unfortunately, even if software is rewritten to enjoy the parallelism of multicore, small sequential sections will make it increasingly difficult to exploit multicore performance scaling in the near future.

This research program will investigate architectural issues regarding how to combine adaptation of the processor cores and the memory system to derive a performance scaling methodology that combines multicore scaling with specialization. We begin with a computer architectural framework combining traditional processor cores with adaptable cores and memory structures. The project will study innovative architectural mechanisms to define a multicore architecture whose resources can dynamically adapt to application needs. The four PIs have world-recognized expertise in computer architecture and VLSI design. They will share experimental infrastructures to facilitate cross-pollination of ideas and validation of each other's results at multiple levels of technology. Doing so will guide the development of novel architectural mechanisms whose impacts on performance and energy efficiency will then be carefully evaluated (again, at many levels of technology). The research program's findings can significantly influence how future multicore architectures will be designed, which, in turn, can pervasively affect information and communication infrastructures in any parallel computing platform for years to come.

1 Objective and Potential Impact of Proposed Research

Up to the turn of the century, microprocessor performance doubled in tandem with the doubling of the number of transistors with each new technology generation (roughly every 18 months). This performance scaling was due in part to clock-frequency scaling: over the last several decades a steady flow of innovations in exploiting instruction-level parallelism (ILP) greatly contributed to this impressive performance growth. The impressive performance improvements from the combination of shrinking feature (transistor) sizes and increasing ILP came to an end a few years ago, however, because of prohibitive levels of energy consumption and diminishing return on investment in exploiting more ILP.

Multicore (CMP, or chip multiprocessor) architectures now attempt to meet the aggressive performance-scaling that users have come to expect by doubling the number of *processor cores* on chip with each new technology generation. Assuming that application performance can improve by a factor of N on a N-core architecture, a multicore architecture could double application performance with every decrease in feature size *without increasing clock frequency*. Unfortunately, "embarrassingly parallel" applications that can fully exploit multicores are rare. More commonly, applications consist of parts that are inherently sequential (S) and parts that are inherently parallel (P), and speedups on N cores never exceed $1+TS/TP$, where TS and TP are the execution times for sequential and parallel parts, respectively. Even if as much as 90% of an application can be parallelized, it can run at most 10 times faster than it would on a single core, regardless of the number of cores it attempts to use. It is clear that multicore performance scaling will not meet the performance expectations that have grown out of Moore's Law; alternative performance scaling methodologies must be developed. Despite this well established analytic result that defines limits to performance scaling through sheer parallelism (Amdahl's Law [1]), roadmaps of processor chip manufacturers forecast a doubling of the number of cores with each new technology node. While Borkar [2] acknowledges several technological performance scaling barriers that can likely be circumvented, the challenges of scaling the performance of any single application are acknowledged.

Hill and Marty [3] show through interesting analytical considerations of Amdahl's law that one can significantly impact performance by devoting chip resources to the sequential part of the execution. Specialization (or

customization) is an alternative performance-scaling methodology in which the processor core is specifically adapted to an entire application or to its most significant parts. It is well established that a specialized core can deliver significantly higher performance and energy efficiency than a general-purpose processor core. Assuming that specialization improves performance a factor S, the speedup achieved by combining multicore performance scaling and specialization is multiplicative, and can potentially break through the ceiling established by Amdahl's Law: $S(1+TS/TP)$. The downside of specialization is the lack of flexibility to host a wide spectrum of applications. Ideally, the cores in a multicore architecture should be adaptable to specialize for common computational activities found in a wide range of applications, so that multicore scaling and specialization can be combined to meet the expectations of users who have come to expect performance to track Moore's Law, doubling each time the number of transistors per chip does so. In addition to these heroic growths in application performance, users now expect the performance to be delivered in an energy-efficient manner.

This research program focuses on exploring an interesting architectural design space that combines multicore scaling with specialization to deliver performance scaling for a wide range of software. Our objective is to understand how we can combine specialization with multicore scaling by adapting computational resources to application needs. Our initial design point is a multicore architecture composed of a number of computational tiles communicating via a scalable interconnection network. Each tile consists of processor cores and a portion of the chip's memory resources. Doubling the number of tiles with each new technology generation trivially delivers the desired multicore scaling. Associating an adaptable accelerator core and an adaptable memory structure with each tile creates the specialization (for an entire application, or even just portions) needed to realize the aggressive performance-scaling goals of this research project. The team's research efforts will strive to deliver *insight* into how accelerator-core and memory-structure adaptation can enjoy sufficiently flexible implementations to enable the required performance scaling on a truly broad range of applications of import. These implementations must not only deliver impressive performance scaling, but must do so while consuming less power. Performance scaling and energy conservation are the motivating design goals behind the proposed research program, but achieving these goals is of little value if the resulting architectural mechanisms (and the systems they comprise) are too complex to program. A third design goal is thus ease of use, enabling previously single-threaded applications to enjoy the performance and power benefits of the proposed parallel hardware. To that end, another research focus will develop adaptable and energy-efficient implementations of a chip-wide cache coherence protocol that exposes to software a scalable, coherent, shared memory. As with the other novel components of this proposed architecture, implementations of this protocol and the resulting shared memory abstraction must achieve high performance with extreme energy efficiency.

2 Overview and Background

To establish a common ground for the sub-projects in this research program, we consider an architectural framework for a multicore architecture whose performance can potentially scale up ideally with each technology generation, assuming that we address problems triggered by identified technology barriers. We justify the architectural framework (in Section 2.1) based on previous research, and define the technological issues (in Section 2.2) that the research program will address. Then, in Section 3, we present the approaches we will take to address these technological issues.

2.1 Multicore Architectural Framework

Multicore microprocessors (or chip multiprocessors) were introduced in the Hydra project at Stanford by Hammond *et al.* [4]. In their simplest form, they consist of a number of general-purpose cores with an attached first-level cache connected to a second-level shared cache. When single-processor core performance scaling abruptly ended around 2004, virtually all processor manufacturers followed in the footsteps of Hydra, adopting a similar chip organization. For example, the Sun Microsystems Niagara chip [5] consists of eight processor cores with a memory system organization similar to Hydra. These early multicore chips are symmetric in the sense that the processor cores appear on one side of the interconnect and the shared cache appears on the other. This organization is therefore not scalable.

In order to scale up performance as core count grows, the RAW tiled architecture [6] consists of processor

tiles as nodes in a mesh-like interconnect, and the entire chip memory is distributed across the tiles via a NUMA (non-uniform-memory-access) manner. Shared caches can most efficiently be shared across tiles by using a NUCA (non-uniform cache access) organization [7]. In order to exploit multicore performance scaling, this research program starts from a scalable tiled multicore architectural framework that consists of a number of tiles, each consisting of a general-purpose processor core and a portion of the entire chip memory. The memory is partitioned into a first-level (and optionally a second-level) core-private cache, along with a portion of shared third-level cache. In addition, a portion of the tile-alloted memory resources is devoted to a scratch-pad memory that allows software to manage caching for regular applications (e.g., as used in the IBM Cell [8]). Tiles communicate using a scalable on-chip interconnection network. Many topologies are possible, but we quite agnostically assume a 2D-torus interconnect to make the following discussion of technical issues more concrete.

This basic multicore architecture exposes a shared-memory model to the software in order to facilitate porting of legacy operating systems. The shared-memory model requires a chip-wide cache coherence mechanism to keep data across the tile caches consistent. We initially assume a scalable directory-based protocol that uses presence-flag vectors to keep track of copies across the chip [9]. While the basic architecture can enjoy multicore performance scaling, serial sections in the applications will likely face the limit of Amdahl's Law at a quite modest core count. In this research program we consider adaptation as a primary means to combine multicore scaling with specialization in a methodology to deliver scalable and energy-efficient performance.

A unique aspect considered in the research program is facilitating adaptation by associating adaptable accelerator cores and memory system structures with each tile *in addition to the general-purpose core used for multicore performance scaling.* As a result, a tile consists of three structures: a general-purpose processor core, an adaptable processor core, and an adaptable memory system structure that, based on application needs, can be partitioned dynamically into a private cache, a shared cache, and a non-cacheable scratch-pad memory portion. We next discuss the research questions we will address to realize this system architecture. Creating an appropriate "computing substrate" is essential to our goal of combining multicore scaling with specialization via a unified architectural methodology.

2.2 Problem Statement

Making the multicore architectural approach viable by combining multicore scaling with specialization requires that the following open issues be addressed:

A first critical issue is *how to architect the adaptable accelerator core.* Ideally, the accelerator core should consist of coarse functional blocks that can be reused across a wide scope of applications. Too coarse a functional granularity will, however, lead to limited reuse; too fine a functional granularity will lead to less performance and energy-inefficient designs. What is needed is a methodology to establish what functional granularity should be used, given a set of applications, to meet performance and energy goals. Developing such a methodology is one of the objectives in a project in the research program.

Once the set of functional blocks has been established, a second issue is *how to implement them in a performance- and energy-efficient manner.* In a second project in the research program, a hardware-software codesign methodology will be developed. This aids in the design of functional blocks for adaptable accelerator cores so that they can be powered down dynamically to achieve high energy efficiency.

Efficient management of memory resources is a key to performance scaling, as the memory bandwidth into future multicore chips will be a scarce resource. In addition, the latency experienced by a memory request for data that is resident in the memory of another tile will relatively increase as we move to new technology nodes. The miss rate experienced comes from either a too large working set (capacity misses) or from communication (coherence misses). A *flexible and adaptable memory system structure in each tile* will address efficiently exploiting available bandwidth at acceptable latencies and power budgets. The design principles of such an adaptable memory system structure are investigated in a third project.

Finally, while scalable directory-based protocols successfully reduce the bandwidth needed to keep caches consistent, they cause long latencies due to the necessary indirection through the directory. Moreover, they waste bandwidth due to interactions, making them energy inefficient. In a fourth project, the low-latency nature of snooping protocols will be exploited to derive *a new class of coherence protocols* that makes use of snoop filtering to lower bandwidth usage and increase energy-efficiency over previously proposed protocols.

3 Project descriptions

In this section, we describe the four project components by first establishing the current state of the art. We then elaborate the approaches we will take to advance that state of the art.

3.1 Accelerator Core Architecture

Many embedded applications are multimedia intensive, memory intensive, or both. They must deliver good performance at low power budgets. For instance, some automotive software that now controls what we take for granted in our vehicles must operate in real time. Furthermore, embedded software increasingly requires the same hardware support as supercomputer applications (indeed, several of the most powerful supercomputers, including Roadrunner [10] and the Blue Gene [11] series, include tens of thousands of embedded processors). Many embedded applications require efficient memory use for good performance, just like their high-end computing counterparts. In addition, multimedia applications rely heavily on digital signal processing (DSP) and/or graphics processing. Even search engines have entered the embedded domain [12]. Our increasing reliance on such tools combined with rapid growth of the digital data they must index [13] places additional demands on the architectures we design for embedded software.

Given the diversity of applications increasingly populating what we traditionally consider "the embedded domain", it makes sense to design flexible hardware that meets the formidable demands of current software *and* can adapt to address the needs of emerging applications. We therefore propose to add lightweight, composable hardware accelerators to tiles in a scalable chip multiprocessor.

This research will identify a small set of composable computation accelerators to include in given tiles. This set will include a memory access component that exploits user/application/compiler/OS knowledge of reference patterns to balance tradeoffs between power and performance when accessing data. The unit must interface to other accelerators (e.g., graphics units, crypto microengines, or regular expression matchers) as well as to caches and main memory controllers. The choice of which accelerators to include in which tiles central to the research agenda. Specific functionality will depend on application needs. We will initially study embedded suites such as MiBench [14] and CSiBE [15], together with kernels from the popular GNU *flex* regular expression scanner generator. We choose the latter because it is a common, computationally intensive, real-world workload. Search engines, compilers, and even intrusion detection applications exploit tokenization and pattern matching. We will construct other kernels with Lucene [16], the software at the heart of *apache*. Finally, we will study PARSEC [17], a suite intended to represent emerging applications from large-scale multithreaded commercial software, including data recognition, mining, and synthesis (RMS); image processing; video encoding; physical simulation for image animation; and financial portfolio management.

The programmable memory accelerator could prefetch structured data (e.g., strided data, sparse data described by indirection, or "pointer chasing" data structures). It could initiate DMAs to restructure data for better DRAM access latencies and on-chip cache utilization [18]. Using such capabilities intelligently will both improve performance and reduce power consumption. Multimedia applications require DSP and graphics accelerators (perhaps fed by a memory accelerator). Accelerator interfaces may consist of memory-mapped registers, ISA additions (as in Intel's SSE streaming SIMD extensions), or novel uses of certain instruction encodings (e.g., an unimplemented instruction or NOP could signal that the following load represents the start of accelerator code). The memory accelerator could even fetch such code efficiently. Communication between main cores and accelerators takes place through memory, and requires no core or bus modifications. This research includes choosing among interface options that balance efficiency and ease of use.

Our accelerators must interact with the memory hierarchy to improve efficiency, both in terms of power and performance. For instance, given knowledge of future reference patterns (provided by the memory accelerator), the memory controller can employ memory access scheduling [19] to reduce energy consumption and minimize access latencies. Cache optimizations range from data restructuring to eliminating memory references known to load the value zero.

Finally, we propose to exploit lightweight per-core power prediction (via analytic models that use performance monitoring counter and temperature sensor values as parameters [20]). Initial studies of this novel approach appear promising: predictions are simple to compute in real time, and such models have been applied to three

different microarchitectures with high prediction accuracy. Furthermore, the models are architecture-specific and application-independent, thus they will be equally useful on emerging and future workloads. Preliminary work uses the models to schedule threads on real multicore machines so that a given power envelope is not exceeded. Studying new ways to harness such simple and powerful models for the proposed scalable architecture is another integral part of the proposed research.

Architectural design experiments will rely on simulation, with FPGA implementations of portions of the functionality later in the design cycle. This research shares a common simulation infrastructure with the larger project. Some ideas discussed here are not new, but since their initial introduction, much has changed with respect to both basic technology and system design goals. The synthesis and adaptation of these ideas and techniques to a high-performance, low-power, scalable embedded CMP represents a significant research effort and a valuable contribution to the state of the art in embedded system design.

3.2 Reconfigurable Accelerator Design

This part of the project is concerned with new hardware/software codesign principles for a low-power accelerator.

The combination of a host processor and a reconfigurable coprocessor can help achieve a good combination of flexibility and specialization. The instruction set of the host processor can be augmented, so that those (typically more complex) operations needed by a particular application are really provided. Among the early attempts to implement instruction augmentation, we find the PRISM architecture (1992) that coupled a single Xilinx XC3090 with a Motorola node processor to perform fine-grained, bitwise functions 20 faster than the host processor, and the Garp architecture (1997) that incorporated a MIPS-II processor and an FPGA core. The host+coprocessor arrangement is beneficial for applications that often utilize the coprocessor. For other applications, however, the coprocessor will incur a significant overhead in terms of timing, area and power.

A general reconfigurable circuit can be (statically) configured to compute some functions and later, either at compile-time or at run-time, it can be (dynamically) reconfigured to perform some other tasks. Partial dynamic reconfiguration or run-time reconfiguration (RTR) has become increasingly more important, as it enables circuits to be time-shared across multiple tasks to provide on-demand computing.

Traditionally, reconfigurable circuits presuppose FPGA technology, which is a neither performance- nor energy-efficient technology. In this subproject, we target the design of a run-time reconfigurable accelerator that can reconcile the performance and energy efficiency of custom-designed ASIC circuits with the flexibility of an FPGA reconfigurable platform. In the following, the key constituents of the design methodology are outlined:

Design-time datapath configuration: A domain of applications are scheduled on an initial configuration of the accelerator. After making an incremental accelerator reconfiguration, we revaluate the accelerator by rescheduling the application domain software. The figure of merit for the new configuration would be a combination of the resulting execution time and the hardware parameters, such as power per cycle, area and timing.

Run-time datapath configuration: A powerful accelerator is complex and design-time considerations alone will not yield minimal energy: A fine-grain accelerator control is needed to e.g. shut down idle parts to save energy. Thus, the accelerator should support run-time adaptivity, such as partial or full power on/off of units. Assuming different operating modes of the accelerator units, we will attempt to use speculative scheduling to systematically identify where adaptivity pays off.

Control scheme: If accelerator instructions need to traverse a significant part of the CMP hierarchy, instruction power dissipation will become high. Thus, a certain degree of tile, core and/or accelerator autonomy is desired. Different accelerator control strategies will be evaluated; ranging from fine-grain, static VLIW to more autonomous schemes, such as one where the host processor initializes a computation, but the accelerator issues local instructions via local state machines.

Most hardware/software codesign techniques for reconfigurable hardware do not target dynamic reconfiguration, however, in recent years design techniques that combine performance and flexibility have emerged. The LISA-based rASIP scheme [21] uses one pre-fabrication and one post-fabrication design phase, but, again, FPGA technology is assumed for the reconfigurable part. The PARLGRAN scheduling approach [22] increases application performance on partial RTR architectures in the form of a matrix of configurable logic blocks, by considering parallelism granularity for the data-parallel tasks.

A unique feature of this subproject is that reconfigurability is enabled by dedicated circuits. In comparison to FPGA technology, such circuits generally offer a lower degree of reconfigurability, but in exchange they yield high performance and low energy. We will leverage previous work in the FlexSoC project; particularly that of flexible circuits [23] and design-time scheduling [24].

3.3 Adaptable Energy-Efficient Cache Coherence Protocols

A shared-memory model simplifies the design of parallel applications and porting of complex legacy software systems such as operating systems. A prerequisite for supporting a shared-memory model is to maintain cache coherence. Scalable implementations of a cache coherence assume that a directory associated with each memory block keeps track of which nodes have a cached copy of that block [9]. In a tiled multicore architecture, a directory is associated with each memory block in the shared third-level cache. A tile that requests a copy of that block first sends a request to the tile containing the directory. If the memory block is in a modified state in another tile, the directory redirects the request to that tile which then responds to the requester in addition to sending a message back to the directory notifying it that it has given up the ownership of the block. While directory-based protocols conserve precious interconnect bandwidth by sending requests to only the nodes involved in a coherence transaction, the redirection through the directory adds latency that may have a significant effect on performance. Moreover, in order to keep the directory information up-to-date, additional messages are needed that add to the network traffic and may result in severe energy inefficiencies.

Snooping protocols, on the other hand, do not cause directory indirection but rely on broadcasting to locate the tiles that have a copy of the block. In comparison with directory-based protocols, snooping protocols yield shorter latency but waste more bandwidth [25]. Another drawback of snooping protocols is that they either rely on interconnection networks that establish an order between consecutive requests issued by different nodes/tiles, such as buses and trees, or on elaborate time-stamp mechanisms that enforce ordering [26]. Recently, Agarwal *et al.* proposed an ordering scheme applicable to unordered scalable interconnection networks to enforce ordering of requests [27]. Yet, however, snooping protocols waste enormous interconnection network bandwidth for broadcasts that leads to inferior performance and energy efficiency.

The objective of this project is to take advantage of the low-latency property of snooping protocols and at the same time advance state-of-the-art by techniques to drastically reduce their bandwidth demands. In the past, we contributed with snoop filtering techniques to reduce the number of snooping actions needed in bus-based interconnects [28] by keeping track of memory pages that are not shared across nodes. Other researchers followed on in our footsteps to contribute with improved snoop filtering techniques [29, 30]. The Regionscout approach [29] uses snoop filters that keep track of memory regions that are not shared. Salapura *et al.* contributed with improved snoop filters to achieve a higher filtering efficiency [30]. Unfortunately, these proposals require broadcasts and cannot be applied to unordered general scalable networks.

This project aims at investigating a new class of energy-efficient adaptive coherence protocols for unordered general scalable interconnection networks. As a concrete example let us assume the architectural framework presented in Section 2 in which tiles are connected using a 2D mesh network. When a tile requests a copy of a block, then assuming a snooping protocol, it would inject a request in both horizontal directions (west and east). At each tile, the request would have to be routed also in the vertical direction (north and south) assuming dimension-order routing to avoid deadlock. This would obviously consume enormous amounts of bandwidth for each coherence transaction. We would like to extend the concept of using snoop filters to avoid injecting requests along a horizontal or vertical segment if tiles on these segments are known to not have any copies of the requested block. Assuming perfect snoop filters, a block request would travel the shortest path to the destination and back again thus experiencing a minimum latency and causing a minimum of bandwidth and energy consumption. To envision such protocols, however, several important issues have to be addressed:

- **Snoop-filter architecture design space exploration.** An open question is how to extend the notion of snoop filters to unordered interconnects to keep track of whether a block or a region of blocks exist in a horizontal or vertical interconnect segment. There is an interesting and unexplored architectural design space both at the overall chip-level architecture as well as design considerations of the snoop filters to make them fast, accurate, and energy-efficient.

- **Ordering.** Multiple coherence transactions for the same block must be allowed to be in progress. An interesting issue is how to orchestrate ordering with as low an impact on concurrency as possible.
- **Deadlock, livelock, and starvation freedom.** A critical issue for any coherence protocol is to address these potential correctness/performance issues. The design of this new class of snooping protocols opens up new challenges in solving them at a reasonable complexity.
- **Impact of interconnection network topology.** While we initially will target a 2D-torus network, we want to extend the applicability of the new class of protocols to any general scalable interconnection network to support future multicore multicore processors.

We will use architectural simulation to evaluate the design space. In the research group, we have over 20 years of experience of architectural simulation and will leverage on a multicore architectural modular simulation tool we have developed in which cache and interconnection modules are available and can be composed in an object-oriented fashion. The simulator is driven by Simics, a functional model of a processor, on which we run parallel applications (e.g. PARSEC, SPLASH-2 and Parallel Bio) to experimentally derive the impact of the protocol design on performance and energy consumption. When it comes to detailed energy-efficiency and performance issues for the snoop-filer designs, we will leverage on Per Larsson-Edefors's group's experimental infrastructure to analyze detailed designs.

3.4 Tile Memory Design

Memory in our adaptable multicore machine will be distributed across the processor tiles, providing private caches for each processor as well as a distributed shared cache for the entire design. We wish to investigate the following problem: How can the principles of reconfigurable cache architectures and non-uniform cache access be applied in an adaptable multicore architecture?

It has long been established for uniprocessors that a single, traditional multiple-level cache hierarchy design point is unlikely to work well across an application range, or even across the computational phases within a single application. Reconfigurable cache architectures [31] have been proposed, where cache memory hardware blocks can be logically rearranged into configurations of varying sizes and associativity, in response to changes in load characteristics; recent work [32] confirms possible performance gains also in multicore machines.

Furthermore, maintaining constant access times to all parts of a large cache memory is increasingly difficult in the face of larger chips, longer signal wires, and steeper edge rates. Non-Uniform Cache Architectures [7] allow access times within a single cache level to vary according to the physical distance the signal has to cover. In a large cache, such as may be distribued across a multicore processor of the type considered in this proposal, access times may vary greatly as a processor retrieves data from its own portion of the cache and from progressively farther-away portions.

In the research proposed here, we will additionally evaluate the use of some of the tile memory for explicitly-addressable scratchpad memory, as may be especially beneficial for the reconfigurable accelerators. There are two performance issues we want to target. The first is that future multicores will likely be memory-bandwidth constrained. For applications that demand a high memory bandwidth, the size of the shared cache is critical. A second issue is that the latency for communication inside the chip will be longer in future multicores. Artifactual communication caused by limited capacity in private caches can be reduced if the private caches are larger. This speaks in favor of considering the entire memory resource inside a tile as a flexible medium that can be partitioned into shared and private caches and scratchpad memory. We want to explore how to orchestrate adaptive partitioning by considering aspects ranging from monitoring application needs via algorithms for adaptation to the architectural structures and circuits needed to make adaptation as energy efficient as possible.

Memory-configuration performance estimation and selection is complicated by technology and tool issues. Standard power-prediction methods for digital circuits are not applicable, since memories differ from other logic in important ways. First, signal swing in standard logic is typically rail-to-rail; in memories, bitline swing is kept low for speed and power reasons, and sense amplifiers are used to reestablish the logic levels. Second, memory cells use ratio logic when values are written, and thus the correct function of the memory depends on device strengths; in contrast, in typical CMOS logic, only performance suffers when device strength varies. Third,

fan-in values (for a memory cell driving a bitline) and activity factor (for the wordline drivers) are very different from values commonly seen in logic. To overcome these problems, designers may fall back on detailed circuit simulation, but are then faced with overwhelming simulation times due to the very large number of devices involved in a typical memory.

For these reasons, microprocessor cache memory configurations are often determined using tools such as CACTI [33]. CACTI embodies "best-practice" design principles and aims to extend these to future processes, but gives less guidance when the design situation motivates a different solution. Second, CACTI is focused on a "pure" cache design problem, whereas the present problem covers a wider range, including not only dynamically-reconfigurable cache partitions but also explicitly-managed scratchpad memories.

An alternative method [34], developed in the Chalmers VLSI group, uses detailed circuit simulations to calibrate highly parameterized power and timing models. The method is highly accurate since it is calibrated with the circuits and process parameters actually used, and may easily be adapted to new memory organizations. We will use and extend this method as required for tile memory power and performance prediction.

4 Project Plan

4.1 Scientific Method

The project PIs bring together world-renowned expertise in computer architecture and VLSI design, making it possible to explore architectural structures in a new design space from two important angles: application and technology impact on performance, and energy efficiency.

Starting from the application impact on architectural structures, Stenström and McKee have decades of experience in experimental architecture research based on state-of-the-art simulation methodologies. Typically, architectural simulators of an entire multicore architecture are modeled at some level of abstraction, e.g., at the clock-cycle level, to estimate the impact of various structures on the execution times and energy consumption of applications. Technology parameters such as circuit delays are often estimated. The PIs will use Simics, one of the most detailed and complete architectural simulators available outside of industry, to study complex interactions among hardware components and all levels of the software stack, including the operating system and the proposed architectural designs. In order to exercise our proposed designs as thoroughly as possible (e.g., to understand how all components interact with software and with each other), we choose application software exhibiting significantly varied behaviors and representing real applications from important areas of commerce, technology, and science.

Continuing with the technology impact on architectural structures, Larsson-Edefors and Svensson have decades of experience in experimental VLSI design research using state-of-the-art CAD tools; in CHAMPP, we will use the Chalmers Cadence installation augmented with Synopsys and experimental point tools to obtain accurate estimates of delays and energy consumption for circuits in architectural structures.

By combining the strengths of the two groups experimental infrastructures, it will be possible to make high-level design exploration using the experimental tool chains of the computer architecture research team, and then perform detailed design exploration at the circuit level, in some cases verifying modeling and analysis by fabricating candidate circuits. This multi-level methodology promises to deliver solid research findings.

4.2 Time Plan

This is a four year project with the following concrete goals:

Year 1: Definition of architectural framework. The goal of the first year is to define the architectural framework using a high-level architectural model. We will use this model to establish the gains that can be achieved through adaptation using the set of application benchmarks.

Year 2-3: Concept exploration. During the second and third years, we will focus on concept exploration in the areas detailed in Section 3. The concepts will take form as architectural structures, circuits, or even methods, such as how to analyze application characteristics to identify what functional blocks should be supported in the

adaptable architectures designed to support that software.

Year 4: Generalization. In the fourth and final year, we will generalize our findings to broaden the impact our research results on the computer industry. Along with disseminating our results through key conferences and journals, we will leverage our participation in the European HiPEAC network (see Section 4.3).

4.3 International Collaborations

The research groups are members of the EU FP7 funded Network of Excellence on High-Performance and Embedded Architecture and Compilers (HiPEAC), and Per Stenström is a co-founding partner of the network. Under his leadership, Chalmers heads the multicore architecture research cluster within the network, which consists of around 70 researchers across Europe. Moreover, the computer architecture group participates in the EU FET-funded Scalable Architecture (SARC) integrated project, where Chalmers leads the memory system design work package. The SARC project concludes this year. Its use of accelerators integrated into each tile and its successful demonstration that specialization significantly boosts performance have provided inspiration and motivation for extending those strategies with the launch of this project. CHAMPP significantly advances SARC's findings and approaches by adding adaptation as a key dimension of the architectures to be studied.

Finally, all PIs participate in extensive service to the computer architecture community. For instance, in this year alone, Stenström serves as the program chair for the IEEE flagship conference on parallel processing (IEEE IPDPS), and McKee serves as general co-chair for the flagship conference on parallel architectures and compilers (IEEE PACT).

5 Strategic Relevance

The paradigm shift to multicore architectures is one of the most dramatic and fundamental changes that the computer industry has yet experienced. Whether the forecasted doubling of the number of cores every eighteen months will double computing performance, in spite of much positive "hype" within the industry, remains an open question. Overwhelming software challenges prevent most applications from taking advantage of the raw compute power of these chip multiprocessors. The alternative performance scaling methodology that we propose here can potentially help break down this "software wall". If successful, the project will have a dramatic impact on the evolution of microprocessor technology well into the future.

References

[1] G.M. Amdahl. Validity of the single-processor approach to achieving large-scale computing capabilities. In *Proc. American Federation of Information Processing Societies Conference*, pages 483–485, 1967.

[2] S. Borkar. Thousand core chips: A technology perspective. In *Proc. ACM/IEEE Design Automation Conference*, pages 746–749, June 2007.

[3] M. Hill and M. Marty. Amdahl's law in the multicore era. *IEEE Computer*, 41(7):33–38, July 2008.

[4] Lance Hammond, Ben Hubbert, Michael Siu, Manohar Prabhu, Mike Chen, and Kunle Olukotun. The Stanford Hydra CMP. *IEEE MICRO Magazine*, 20:71–84, March-April 2000.

[5] Poonacha Kongetira, Kathirgamar Aingaran, and Kunle Olukotun. Niagara: A 32-way multithreaded SPARC processor. *IEEE MICRO Magazine*, 25(2):21–29, March-April 2005.

[6] Michael Taylor et al. Evaluation of the Raw microprocessor: An exposed-wire-delay architecture for ILP and streams. In *Proc. ACM/IEEE ISCA*, pages 2–13, June 2004.

[7] Haakon Dybdahl and Per Stenstrom. An adaptive shared/private NUCA cache partitioning scheme for chip multiprocessors. In *Proc. IEEE Int. Symp. HPCA*, pages 2–12, February 2007.

[8] H. Peter Hofstee. Power efficient processor architecture and the cell processor. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture*, 2005.

[9] Daniel Leonski et al. The Stanford DASH multiprocessor. *Computer*, pages 63–79, March 1992.

[10] K.J. Barker et al. Entering the petaflop era: The architecture and performance of roadrunner. In *Proc. ACM/IEEE Conference on Supercomputing*, pages 1–11, November 2008.

[11] M. Ohmacht et al. Blue Gene/L compute chip: Memory and ethernet subsystem. *IBM Journal of Research and Development*, 49(2-3):255–264, May 2005.

[12] J.M. Gitlin. Ars exclusive: Review of Papers for iPhone. http://arstechnica.com/apple/reviews/2009/02/ars-exclusive-review-papers-for-iphone.ars, February 2009.

[13] J.F. Gantz, D. Reinsel, C. Chute, W. Schlichting, J. Mcarthur, S. Minton, I. Xheneti, A. Toncheva, and A. Manfrediz. The expanding digital universe a forecast of worldwide information growth through 2010. http://www.emc.com/collateral/analyst-reports/expanding-digital-idc-white-paper.pdf, March 2007.

[14] M.R. Guthaus et al. MiBench: A free, commercially representative embedded benchmark suite. In *Proc. IEEE Workshop on Workload Characterization*, pages 3–14, December 2001.

[15] University of Szeged Department of Software Engineering. GCC code-size benchmark environment (csibe). http://www.csibe.org/.

[16] Apache Software Foundation. Lucene. http://lucene.apache.org, April 2009.

[17] C. Bienia, S. Kumar, J.P. Singh, and K. Li. The PARSEC benchmark suite: Characterization and architectural implications. In *Proc.IEEE/ACM PACT 2008*, pages 72–81, October 2008.

[18] B. Chandramouli, J.B. Carter, W.C. Hsieh, and S.A. McKee. A cost framework for evaluating integrated restructuring optimizations. In *Proc.IEEE/ACM International Conference on Parallel Architectures and Compilation Techniques*, pages 98–107, September 2001.

[19] S.A. McKee. *Maximizing Memory Bandwidth for Streamed Computations*. PhD thesis, School of Engineering and Applied Science, Univ. of Virginia, May 1995.

[20] K. Singh, M. Bhadauria, and S.A. McKee. Real time power estimation of multi-cores via performance counters. *Proc. Workshop on Design, Architecture and Simulation of Chip Multi-Processors*, November 2008.

[21] A. Chattopadhyay et al. Design space exploration of partially re-configurable embedded processors. In *Proc. Design, Automation and Testing in Europe (DATE)*, pages 1–6, April 2007.

[22] S. Banerjee et al. Exploiting application data-parallelism on dynamically reconfigurable architectures: Placement and architectural considerations. *IEEE Transactions on VLSI Systems*, 17(2):234–247, 2009.

[23] M. Själander and P. Larsson-Edefors. Multiplication acceleration through twin precision. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2009. in print.

[24] T. Schilling, M. Själander, and P. Larsson-Edefors. Scheduling for an embedded architecture with a flexible datapath. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, May 2009. to be presented.

[25] E. E. Bilir et al. Multicast snooping: A new coherence method using a multicast address network. In *Proceedings of International Symposium on Computer Architecture*, May 1999.

[26] M. M. K. Martin et al. Timestamp snooping: An approach for extending smps. In *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems*, Nov 2000.

[27] Niket Agarwal et al. In-network snoop ordering: Snoopy coherence on unordered interconnects. In *Proceedings of 15th International Symposium on High-Performance Computer Architecture (HPCA)*, Feb 2009.

[28] M. Ekman, F. Dahlgren, and P. Stenstrom. TLB and snoop energy-reduction using virtual caches in low-power chip-multiprocessors. In *Proc. Int'l Symp. Low Power Electronics and Design*, 2002.

[29] Jason Cantin et al. Coarse-grain coherence tracking: Regionscout and region coherence arrays. *IEEE Micro*, pages 70 – 79, 2006.

[30] V Salapura et al. Design and implementation of the blue gene/p snoop filter. In *Proc. of High Performance Computer Architecture*, pages 5 – 14, 2008.

[31] Rajeev Balasubramonian et al. Memory hierarchy reconfiguration for energy and performance in general-purpose processor architectures. In *Proc. ACM/IEEE Int. Symp. on Microarchitecture*, pages 245 – 257, 2000.

[32] Jie Tao, Marcel Kunze, Fabian Nowak, Rainer Buchty, and Wolfgang Karl. Performance advantage of reconfigurable cache design on multicore processor systems. *Int J Parallel Prog*, (36):347–360, 2008.

[33] Shyamkumar Thoziyoor, Naveen Muralimanohar, Jung Ho Ahn, and Norman P. Jouppi. Cacti 5.1. Technical Report HPL-2008-20, HP Labs, 2008.

[34] Minh Quang Do. *Accurate Leakage-Conscious Architecture-Level Power Estimation for SRAM-based Memory Structures*. PhD thesis, Chalmers University of Technology, June 2007.