MO901R/MC039A - Seminário de Computação - Seminários em Arquitetura de Computadores e Compiladores

Informações Gerais

Professor: Rodolfo Jardim de Azevedo
Aula: Seg: 10h-12h
Atendimento: Procurar diretamente na minha sala.
Sala: 322

Avaliação

- A avaliação será baseada nas apresentações, presença e participação em sala
- **Qualquer tentativa de fraude implicará em Nmo901/Nmc039 = 0 para todos os envolvidos**
- NotaFinal = 0,5 * NotaApresentação + 0,5 * NotaParticipação
- Conceitos
  - S: presença >= 75%
  - I: presença < 75%

Veja Também

- Alunos matriculados
- Acompanhamento e notas
- Wiki

Avisos

04/03 Verifiquem as datas importantes no calendário de pós-graduação e no calendário de graduação.

Objetivos

Ler, apresentar e discutir artigos recentes da área de Arquitetura de Computadores e Compiladores

Calendário

As referências para os artigos estão no final desta página.

Data	Artigo	Apresentadores	
05/03	Apresentação, seleção dos artigos		
12/03	A case for an SC-preserving compiler	Gabriel (023869)	
19/03	Idempotent processor architecture	Raoni (069297)	Rafael (045840)
26/03	The Case for GPGPU Spatial Multitasking	Leandro (001963)	Rodolfo (134077)
02/04	Releasing efficient beta cores to market early	Jefferson (033452)	Éricles (089237)
09/04	Quasi-Nonvolatile SSD: Trading Flash Memory Nonvolatility to Improve Storage System Performance for Enterprise Applications	André (080681)	Arnaldo

16/04	FabScalar: composing synthesizable RTL designs of arbitrary cores within a canonical superscalar template	Rogério (134078)	Marcelo (085937)
23/04	Statistical Performance Comparisons of Computers	Liana (098364)	Victor (084643)
30/04	Feriado		
07/05	Bundled execution of recurring traces for energy-efficient general purpose processing	Rafael (115153)	Daniel (057612)
14/05	Power Balanced Pipelines	Filomen (134234)	Juan (134063)
21/05	Prefetch-aware shared resource management for multi-core systems	Lucas (134068)	
28/05	On-the-fly detection of precise loop nests across procedures on a dynamic binary translation system	César (115121)	Maria Julia (117964)
04/06	Booster: Reactive Core Acceleration for Mitigating the Effects of Process Variation and Application Imbalance in Low-Voltage Chips	Rene (105641)	Ricardo (106938)
11/06	i-NVMM: a secure non-volatile main memory system with incremental encryption	Alexandre (108411)	Murilo (134072)
18/06	QsCores: trading dark silicon for scalable energy efficiency with quasi-specific cores	Mario (123547)	Emílio (120884)
25/06	Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware	Marcel (071675)	Maxiwell (089057)

Artigos

Todos os alunos devem ler todos os artigos indicados no calendário. Além disto, para cada artigo, uma dupla de alunos fará uma apresentação em sala seguida de tópicos para discussão. Veja a sugestão de artigos para este semestre.

Sebastian Burckhardt, Daan Leijen, Caitlin Sadowski, Jaeheon Yi, and Thomas Ball. 2011. Two for the price of one: a model for parallel and incremental computation. In *Proceedings of the 2011 ACM international conference on Object oriented programming systems languages and applications* (OOPSLA '11). ACM, New York, NY, USA, 427-444. DOI=10.1145/2048066.2048101 http://doi.acm.org/10.1145/2048066.2048101

Parallel or incremental versions of an algorithm can significantly outperform their counterparts, but are often difficult to develop. Programming models that provide appropriate abstractions to decompose data and tasks can simplify parallelization. We show in this work that the same abstractions can enable both parallel and incremental execution. We present a novel algorithm for parallel self-adjusting computation. This algorithm extends a deterministic parallel programming model (concurrent revisions) with support for recording and repeating computations. On record, we construct a dynamic dependence graph of the parallel computation. On repeat, we reexecute only parts whose dependencies have changed. We implement and evaluate our idea by studying five example programs, including a realistic multi-pass CSS layout algorithm. We describe programming techniques that proved particularly useful to improve the performance of self-adjustment in practice. Our final results show significant speedups on all examples (up to 37x on an 8-core machine). These speedups are well beyond what can be achieved by parallelization alone, while requiring a comparable effort by the programmer.

2. Daniel Marino, Abhayendra Singh, Todd Millstein, Madanlal Musuvathi, and Satish Narayanasamy. 2011. A case for an SC-preserving compiler. In Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation (PLDI '11). ACM, New York, NY, USA, 199-210. DOI=10.1145/1993498.1993522 http://doi.acm.org/10.1145/1993498.1993522

The most intuitive memory consistency model for shared-memory multi-threaded programming is sequential consistency (SC). However, current concurrent programming languages support a relaxed model, as such relaxations are deemed necessary for enabling important optimizations. This paper demonstrates that an SC-preserving compiler, one that ensures that every SC behavior of a compiler-generated binary is an SC behavior of the source program, retains most of the performance benefits of an optimizing compiler. The key observation is that a large class of optimizations crucial for performance are either already SC-preserving or can be modified to preserve SC while retaining much of their effectiveness. An SC-preserving compiler, obtained by restricting the optimization phases in LLVM, a state-of-the-art C/C++ compiler,

incurs an average slowdown of 3.8% and a maximum slowdown of 34% on a set of 30 programs from the SPLASH-2, PARSEC, and SPEC CINT2006 benchmark suites. While the performance overhead of preserving SC in the compiler is much less than previously assumed, it might still be unacceptable for certain applications. We believe there are several avenues for improving performance without giving up SC-preservation. In this vein, we observe that the overhead of our SC-preserving compiler arises mainly from its inability to aggressively perform a class of optimizations we identify as eager-load optimizations. This class includes common-subexpression elimination, constant propagation, global value numbering, and common cases of loop-invariant code motion. We propose a notion of interference checks in order to enable eager-load optimizations while preserving SC. Interference checks expose to the compiler a commonly used hardware speculation mechanism that can efficiently detect whether a particular variable has changed its value since last read.

3. Niket K. Choudhary, Salil V. Wadhavkar, Tanmay A. Shah, Hiran Mayukh, Jayneel Gandhi, Brandon H. Dwiel, Sandeep Navada, Hashem H. Najaf-abadi, and Eric Rotenberg. 2011. FabScalar: composing synthesizable RTL designs of arbitrary cores within a canonical superscalar template. In Proceedings of the 38th annual international symposium on Computer architecture (ISCA '11). ACM, New York, NY, USA, 11-22. DOI=10.1145/2000064.2000067 http://doi.acm.org/10.1145/2000064.2000067

A growing body of work has compiled a strong case for the single-ISA heterogeneous multi-core paradigm. A single-ISA heterogeneous multi-core provides multiple, differently-designed superscalar core types that can streamline the execution of diverse programs and program phases. No prior research has addressed the 'Achilles' heel of this paradigm: design and verification effort is multiplied by the number of different core types. This work frames superscalar processors in a canonical form, so that it becomes feasible to quickly design many cores that differ in the three major superscalar dimensions: superscalar width, pipeline depth, and sizes of structures for extracting instruction-level parallelism (ILP). From this idea, we develop a toolset, called FabScalar, for automatically composing the synthesizable register-transfer-level (RTL) designs of arbitrary cores within a canonical superscalar template. The template defines canonical pipeline stages and interfaces among them. A Canonical Pipeline Stage Library (CPSL) provides many implementations of each canonical pipeline stage, that differ in their superscalar width and depth of sub-pipelining. An RTL generation tool uses the template and CPSL to automatically generate an overall core of desired configuration. Validation experiments are performed along three fronts to evaluate the quality of RTL designs generated by FabScalar: functional and performance (instructions-per-cycle (IPC)) validation, timing validation (cycle time), and confirmation of suitability for standard ASIC flows. With FabScalar, a chip with many different superscalar core types is conceivable.

4. Yunsup Lee, Rimas Avizienis, Alex Bishara, Richard Xia, Derek Lockhart, Christopher Batten, and Krste Asanović. 2011. Exploring the tradeoffs between programmability and efficiency in data-parallel accelerators. In Proceedings of the 38th annual international symposium on Computer architecture (ISCA '11). ACM, New York, NY, USA, 129-140. DOI=10.1145/2000064.2000080 http://doi.acm.org/10.1145/2000064.2000080

We present a taxonomy and modular implementation approach for data-parallel accelerators, including the MIMD, vector-SIMD, subword-SIMD, SIMT, and vector-thread (VT) architectural design patterns. We have developed a new VT microarchitecture, Maven, based on the traditional vector-SIMD microarchitecture that is considerably simpler to implement and easier to program than previous VT designs. Using an extensive design-space exploration of full VLSI implementations of many accelerator design points, we evaluate the varying tradeoffs between programmability and implementation efficiency among the MIMD, vector-SIMD, and VT patterns on a workload of microbenchmarks and compiled application kernels. We find the vector cores provide greater efficiency than the MIMD cores, even on fairly irregular kernels. Our results suggest that the Maven VT microarchitecture is superior to the traditional vector-SIMD architecture, providing both greater efficiency and easier programmability.

5. Eiman Ebrahimi, Chang Joo Lee, Onur Mutlu, and Yale N. Patt. 2011. Prefetch-aware shared resource management for multi-core systems. In Proceedings of the 38th annual international symposium on Computer architecture (ISCA '11). ACM, New York, NY, USA, 141-152. DOI=10.1145/2000064.2000081 http://doi.acm.org/10.1145/2000064.2000081

hip multiprocessors (CMPs) share a large portion of the memory subsystem among multiple cores. Recent proposals have addressed high-performance and fair management of these shared resources; however, none of them take into account prefetch requests. Without prefetching, significant performance is lost, which is why existing systems prefetch. By not taking into account prefetch requests, recent shared-resource management proposals often significantly degrade both performance and fairness, rather than improve them in the presence of prefetching. This paper is the first to propose mechanisms that both manage the shared resources of a multi-core chip to obtain high-performance and fairness, and also exploit prefetching. We apply our proposed mechanisms to two resource-based management techniques for memory scheduling and one source-throttling-based management technique for the entire shared memory system. We show that our mechanisms improve the performance of a 4-core system that uses network fair queuing, parallelism-aware batch scheduling, and fairness via source throttling by 11.0%, 10.9%, and 11.3% respectively, while also significantly improving fairness.

6. Siddhartha Chhabra and Yan Solihin. 2011. i-NVMM: a secure non-volatile main memory system with incremental encryption. In Proceedings of the 38th annual international symposium on Computer architecture (ISCA '11). ACM, New York, NY, USA, 177-188. DOI=10.1145/2000064.2000086 http://doi.acm.org/10.1145/2000064.2000086

Emerging technologies for building non-volatile main memory (NVMM) systems suffer from a security vulnerability where information lingers on long after the system is powered down, enabling an attacker with physical access to the system to extract sensitive information off the memory. The goal of this study is to find a solution for such a security vulnerability. We introduce i-NVMM, a data privacy protection scheme for NVMM, where the main memory is encrypted incrementally, i.e. different data in the main memory is encrypted at different times depending on whether the data is predicted to still be useful to the processor. The motivation behind incremental encryption is the observation that the working set of an application is much smaller than its resident set. By identifying the working set and encrypting remaining part of the resident set, i-NVMM can keep the majority of the main memory encrypted at all times without penalizing performance by much. Our experiments demonstrate promising results. i-NVMM keeps 78% of the main memory encrypted across SPEC2006 benchmarks, yet only incurs 3.7% execution time overhead, and has a negligible impact on the write endurance of NVMM, all achieved with a relatively simple hardware support in the memory module.

7. Sangeetha Sudhakrishnan, Rigo Dicochea, and Jose Renau. 2011. Releasing efficient beta cores to market early. In Proceedings of the 38th annual international symposium on Computer architecture (ISCA '11). ACM, New York, NY, USA, 213-222. DOI=10.1145/2000064.2000090 http://doi.acm.org/10.1145/2000064.2000090

Verification of modern processors is an expensive, time consuming, and challenging task. Although it is estimated that over half of total design time is spent on verification, we often find processors with bugs released into the market. This paper proposes an architecture that tolerates, not just the typically infrequent bugs found in current processors, but a significantly larger set of bugs. The objective is to allow for a much quicker time to market. We propose an architecture built around Beta Cores, which are cores partially verified. Our proposal intelligently activates and deactivates a simple single issue in-order Checker Core to verify a buggy superscalar out-oforder Beta Core. Our Beta Core Solution (BCS), which includes the Beta Core, the Checker Core, and the logic to detect potentially buggy situations consumes just 5% more power than the stand-alone Beta Core. We also show that performance is only slightly diminished with an average slowdown of 1.6%. By leveraging program signatures,our BCS only needs a simple in-order Checker Core, at half the frequency, to verify a complex 4 issue out-of-order Beta Core. The BCS architecture allows for a decrease in verification effort and thus a quicker time to market.

8. Mark Gebhart, Daniel R. Johnson, David Tarjan, Stephen W. Keckler, William J. Dally, Erik Lindholm, and Kevin Skadron. 2011. Energy-efficient mechanisms for managing thread context in throughput processors. SIGARCH Comput. Archit. News 39, 3 (June 2011), 235-246. DOI=10.1145/2024723.2000093 http://doi.acm.org/10.1145/2024723.2000093

Modern graphics processing units (GPUs) use a large number of hardware threads to hide both function unit and memory access latency. Extreme multithreading requires a complicated thread scheduler as well as a large register file, which is expensive to access both in terms of energy and latency. We present two complementary techniques for reducing energy on massively-threaded processors such as GPUs. First, we examine register file caching to replace accesses to the large main register file with accesses to a smaller structure containing the immediate register working set of active threads. Second, we investigate a two-level thread scheduler that maintains a small set of active threads to hide ALU and local memory access latency and a larger set of pending threads to hide main memory latency. Combined with register file caching, a two-level thread scheduler provides a further reduction in energy by limiting the allocation of temporary register cache resources to only the currently active subset of threads. We show that on average, across a variety of real world graphics and compute workloads, a 6-entry per-thread register file cache reduces the number of reads and writes to the main register file by 50% and 59% respectively. We further show that the active thread count can be reduced by a factor of 4 with minimal impact on performance, resulting in a 36% reduction of register file energy.

9. Charles R. Lefurgy, Alan J. Drake, Michael S. Floyd, Malcolm S. Allen-Ware, Bishop Brock, Jose A. Tierno, and John B. Carter. 2011. Active management of timing guardband to save energy in POWER7. In Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44 '11). ACM, New York, NY, USA, 1-11. DOI=10.1145/2155620.2155622 http://doi.acm.org/10.1145/2155620.2155622

Microprocessor voltage levels include substantial margin to deal with process variation, system power supply variation, workload induced thermal and voltage variation, aging, random uncertainty, and test inaccuracy. This margin allows the microprocessor to operate correctly during worst-case conditions, but during typical conditions it is larger than necessary and wastes energy. We present a mechanism that reduces excess voltage margin by (1) introducing a critical path monitor (CPM) circuit that measures available timing margin in real-time, (2) coupling the CPM output to the clock generation circuit to adjust clock frequency within cycles in response to excess or inadequate timing margin, and (3) adjusting the processor voltage level periodically in firmware to achieve a specified average clock frequency target. We implemented this mechanism in a prototype IBM POWER7 server. During better-than-worst case conditions our guardband management mechanism reduces the average voltage setting 137-152 mV below nominal, resulting in average processor power reduction of 24% with no performance loss while running industry-standard benchmarks.

10. Shantanu Gupta, Shuguang Feng, Amin Ansari, Scott Mahlke, and David August. 2011. Bundled execution of recurring traces for energy-efficient general purpose processing. In Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44 '11). ACM, New York, NY, USA, 12-23. DOI=10.1145/2155620.2155623 http://doi.acm.org/10.1145/2155620.2155623

Technology scaling has delivered on its promises of increasing device density on a single chip. However, the voltage scaling trend has failed to keep up, introducing tight power constraints on manufactured parts. In such a scenario, there is a need to incorporate energy-efficient processing resources that can enable more computation within the same power budget. Energy

efficiency solutions in the past have typically relied on application specific hardware and accelerators. Unfortunately, these approaches do not extend to general purpose applications due to their irregular and diverse code base. Towards this end, we propose BERET, an energy-efficient co-processor that can be configured to benefit a wide range of applications. Our approach identifies recurring instruction sequences as phases of "temporal regularity" in a program's execution, and maps suitable ones to the BERET hardware, a three-stage pipeline with a bundled execution model. This judicious off-loading of program execution to a reduced-complexity hardware demonstrates significant savings on instruction fetch, decode and register file accesses energy. On average, BERET reduces energy consumption by a factor of 3-4X for the program regions selected across a range of general-purpose and media applications. The average energy savings for the entire application run was 35% over a single-issue in-order processor.

11. Marc de Kruijf and Karthikeyan Sankaralingam. 2011. Idempotent processor architecture. InProceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture(MICRO-44 '11). ACM, New York, NY, USA, 140-151. DOI=10.1145/2155620.2155637 http://doi.acm.org/10.1145/2155620.2155637

Improving architectural energy efficiency is important to address diminishing energy efficiency gains from technology scaling. At the same time, limiting hardware complexity is also important. This paper presents a new processor architecture, the idempotent processor architecture, that advances both of these directions by presenting a new execution paradigm that allows speculative execution without the need for hardware checkpoints to recover from mis-speculation, instead using only re-execution to recover. Idempotent processors execute programs as a sequence of compiler-constructed idempotent (re-executable) regions. The nature of these regions allows precise state to be reproduced by re-execution, obviating the need for hardware recovery support. We build upon the insight that programs naturally decompose into a series of idempotent regions and that these regions can be large. The paradigm of executing idempotent regions, which we call idempotent processing, can be used to support various types of speculation, including branch prediction, dependence prediction, or execution in the presence of hardware faults or exceptions. In this paper, we demonstrate how idempotent processing simplifies the design of in-order processors. Conventional in-order processors suffer from significant complexities to achieve high performance while supporting the execution of variable latency instructions and enforcing precise exceptions. Idempotent processing eliminates much of these complexities and the resulting inefficiencies by allowing instructions to retire out of order with support for re-execution when necessary to recover precise state. Across a diverse set of benchmark suites, our quantitative results show that we obtain a geometric mean performance increase of 4.4% (up to 25% and beyond) while maintaining an overall reduction in power and hardware complexity.

12. Ganesh Venkatesh, Jack Sampson, Nathan Goulding-Hotta, Sravanthi Kota Venkata, Michael Bedford Taylor, and Steven Swanson. 2011. QsCores: trading dark silicon for scalable energy efficiency with quasi-specific cores. In Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44 '11). ACM, New York, NY, USA, 163-174. DOI=10.1145/2155620.2155640 http://doi.acm.org/10.1145/2155620.2155640

Transistor density continues to increase exponentially, but power dissipation per transistor is improving only slightly with each generation of Moore's law. Given the constant chip-level power budgets, this exponentially decreases the percentage of transistors that can switch at full frequency with each technology generation. Hence, while the transistor budget continues to increase exponentially, the power budget has become the dominant limiting factor in processor design. In this regime, utilizing transistors to design specialized cores that optimize energy-per-computation becomes an effective approach to improve system performance. To trade transistors for energy efficiency in a scalable manner, we propose Quasi-specific Cores, or QsCores, specialized processors capable of executing multiple general-purpose computations while providing an order of magnitude more energy efficiency than a general-purpose processor. The QsCores design flow is based on the insight that similar code patterns exist within and across applications. Our approach exploits these similar code patterns to ensure that a small set of specialized cores support a large number of commonly used computations. We evaluate QsCores's ability to target both a single application library (e.g., data structures) as well as a diverse workload consisting of applications selected from different domains (e.g., SPECINT, EEMBC, and Vision). Our results show that QsCores can provide 18.4 x better energy efficiency than general-purpose processors while reducing the amount of specialized logic required to support the workload by up to 66%.

13. Andrew Hay, Karin Strauss, Timothy Sherwood, Gabriel H. Loh, and Doug Burger. 2011. Preventing PCM banks from seizing too much power. In Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44 '11). ACM, New York, NY, USA, 186-195. DOI=10.1145/2155620.2155642 http://doi.acm.org/10.1145/2155620.2155642

Widespread adoption of Phase Change Memory (PCM) requires solutions to several problems recently addressed in the literature, including limited endurance, increased write latencies, and system-level changes required to exploit non-volatility. One important difference between PCM and DRAM that has received less attention is the increased need for write power management. Writing to a PCM cell requires high current density over hundreds of nanoseconds, and hard limits on the number of simultaneous writes must be enforced to ensure correct operation, limiting write throughput and therefore overall performance. Because several wear reduction schemes only write those bits that need to be written, the amount of power required to write a cache line back to memory under such a system is now variable, which creates opportunity to reduce write power. This paper proposes policies that monitor the bits that have actually been changed over time, as opposed to simply those lines that are dirty. These polices can more effectively allocate power across the system to improve write concurrency. This method for allocating power across the memory subsystem is built on the idea of "power tokens," a transferable, but time-specific, allocation of power. The results show that with a storage overhead of 4.3% in the last-level cache, a power-aware memory system can improve the performance of multiprogrammed workloads by up to 84%.

14. Soontae Kim, Jongmin Lee, Jesung Kim, and Seokin Hong. 2011. Residue cache: a low-energy low-area L2 cache architecture via compression and partial hits. In Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44 '11). ACM, New York, NY, USA, 420-429. DOI=10.1145/2155620.2155670 http://doi.acm.org/10.1145/2155620.2155670

L2 cache memories are being adopted in the embedded systems for high performance, which, however, increases energy consumption due to their large sizes. We propose a low-energy low-area L2 cache architecture, which performs as well as the conventional L2 cache architecture with 53% less area and around 40% less energy consumption. This architecture consists of an L2 cache and a small cache called residue cache. L2 and residue cache lines are half sized of the conventional L2 cache lines. Well compressed conventional L2 cache lines are stored only in the L2 cache while other poorly compressed lines are stored in both the L2 and residue caches. Although many conventional L2 cache lines are not fully captured by the residue cache, most accesses to them do not incur misses because not all their words are needed immediately, which are termed as partial hits in this paper. The residue cache architecture consumes much lower energy and area than conventional L2 cache architectures, and can be combined synergistically with other schemes such as the line distillation and ZCA. The residue cache architecture is also shown to perform well on a 4-way superscalar processor typically used in high performance systems.

15. Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware. ASPLOS 2012

Emerging scale-out workloads require extensive amounts of computational resources. However, data centers using modern server hardware face physical constraints in space and power, limiting further expansion and calling for improvements in the computational density per server and in the per-operation energy. Continuing to improve the computational resources of the cloud while staying within physical constraints mandates optimizing server efficiency to ensure that server hardware closely matches the needs of scale-out workloads. In this work, we introduce CloudSuite, a benchmark suite of emerging scale-out workloads. We use performance counters on modern servers to study scale-out workloads, finding that today's predominant processor micro-architecture is inefficient for running these workloads. We find that inefficiency comes from the mismatch between the workload needs and modern processors, particularly in the organization of instruction and data memory systems and the processor core micro-architecture. Moreover, while today's predominant micro-architecture is inefficient when executing scale-out workloads, we find that continuing the current trends will further exacerbate the inefficiency in the future. In this work, we identify the key micro-architectural needs of scale-out workloads, calling for a change in the trajectory of server processors that would lead to improved computational density and power efficiency in data centers.

16. Manu Awasthi, Manjunath Shevgoor, Kshitij Sudan, Bipin Rajendran, Rajeev Balasubramonian, Viji Srinivasan. Efficient Scrub Mechanisms for Error-Prone Emerging Memories. HPCA 2012.

Many memory cell technologies are being considered as possible replacements for DRAM and Flash technologies, both of which are nearing their scaling limits. While these new cells (PCM, STTRAM, FeRAM, etc.) promise high density, better scaling, and nonvolatility, they introduce new challenges. Solutions at the architecture level can help address some of these problems; e.g., prior research has proposed wear-leveling and hard error tolerance mechanisms to overcome the limited write endurance of PCM cells. In this paper, we focus on the soft error problem in PCM, a topic that has received little attention in the architecture community. Soft errors in DRAM memories are typically addressed by having SECDED support and a scrub mechanism. The scrub mechanism scans the memory looking for a single-bit error and corrects it before the line experiences a second uncorrectable error. However, PCM (and other emerging memories) are prone to new sources of soft errors. In particular, multi-level cell (MLC) PCM devices will suffer from resistance drift, that increases the soft error rate and incurs high overheads for the scrub mechanism. This paper is the first to study the design of architectural scrub mechanisms, especially when tailored to the drift phenomenon in MLC PCM. Many of our solutions will also apply to other soft-error prone emerging memories. We first show that scrub overheads can be reduced with support for strong ECC codes and a lightweight error detection operation. We then design different scrub algorithms that can adaptively trade-off soft and hard errors. Using an approach that combines all proposed solutions, our scrub mechanism yields a 96.5% reduction in uncorrectable errors, a 24.4 × decrease in scrub-related writes, and a 37.8% reduction in scrub energy, relative to a basic scrub algorithm used in modern DRAM systems.

17. Timothy N. Miller, Xiang Pan, Renji Thomas, Naser Sedaghati, Radu Teodorescu. Booster: Reactive Core Acceleration for Mitigating the Effects of Process Variation and Application Imbalance in Low-Voltage Chips. HPCA 2012

Lowering supply voltage is one of the most effective techniques for reducing microprocessor power consumption. Unfortunately, at low voltages, chips are very sensitive to process variation, which can lead to large differences in the maximum frequency achieved by individual cores. This paper presents Booster, a simple, low-overhead framework for dynamically rebalancing performance heterogeneity caused by process variation and application imbalance. The Booster CMP includes two power supply rails set at two very low but different voltages. Each core can be dynamically assigned to either of the two rails using a gating circuit. This allows cores to quickly switch between two different frequencies. An on-chip governor controls the timing of the switching and the time spent on each rail. The governor manages a "boost budget" that dictates how many cores can be sped up depending on the power constraints) at any given time. We present two implementations of Booster: Booster VAR, which virtually eliminates the effects of core-to-core frequency variation in near-threshold CMPs, and Booster SYNC, which additionally reduces the effects of imbalance in multithreaded applications. Evaluation using PARSEC an SPLASH2 benchmarks running on a simulated 32-core system shows an average performance improvement of 11% for Booster VAR and 23% for Booster SYNC.

18. Jacob T. Adriaens, Katherine Compton, Nam Sung Kim, Michael J. Schulte. The Case for GPGPU Spatial Multitasking. HPCA 2012.

The set-top and portable device market continues to grow, as does the demand for more performance under increasing cost, power, and thermal constraints. The integration of Graphics Processing Units (GPUs) into these devices and the emergence of general-purpose computations on graphics hardware enable a new set of highly parallel applications. In this paper, we propose and make the case for a GPU multitasking technique called spatial multitasking. Traditional GPU multitasking techniques, such as cooperative and preemptive multitasking, partition

GPU time among applications, while spatial multitasking allows GPU resources to be partitioned among multiple applications simultaneously. We demonstrate the potential benefits of spatial multitasking with an analysis and characterization of General-Purpose GPU (GPGPU) applications. We find that many GPGPU applications fail to utilize available GPU resources fully, which suggests the potential for significant performance benefits using spatial multitasking instead of, or in combination with, preemptive or cooperative multitasking. We then implement spatial multitasking and compare it to cooperative multitasking using simulation. We evaluate several heuristics for partitioning GPU stream multiprocessors (SMs) among applications and find spatial multitasking shows an average speedup of up to 1.19 over cooperative multitasking when two applications are sharing the GPU. Speedups are even higher when more than two applications are sharing the GPU.

19. Xuehai Qian, Benjamin Sahelices, Josep Torrellas. BulkSMT: Designing SMT Processors for Atomic-Block Execution. HPCA 2012.

Multiprocessor architectures that continuously execute atomic blocks (or chunks) of instructions can improve performance and software productivity. However, all of the prior proposals for such architectures assume single-context cores as building blocks — rather than the widely-used Simultaneous Multithreading (SMT) cores. As a result, they are underutilizing hardware resources. This paper presents the first SMT design that supports continuous chunked (or transactional) execution of its contexts. Our design, called BulkSMT, can be used either in a single-core processor or in a multicore of SMTs. We present a set of BulkSMT configurations with different cost and performance. We also describe the architectural primitives that enable chunked execution in an SMT core and in a multicore of SMTs. Our results, based on simulations of SPLASH-2 and PARSEC codes, show that BulkSMT supports chunked execution cost-effectively. In a 4-core multicore with eager chunked execution, BulkSMT reduces the execution time of the applications by an average of 26% compared to running on single context cores. In a single core, the average reduction is 32%.

20. Lei Jiang, Bo Zhao, Youtao Zhang, Jun Yang, Bruce R. Childers. Improving Write Operations in MLC Phase Change Memory. HPCA 2012.

Phase change memory (PCM) recently has emerged as a promising technology to meet the fast growing demand for large capacity memory in modern computer systems. In particular, multi-level cell (MLC) PCM that stores multiple bits in a single cell, offers high density with low per-byte fabrication cost. However, despite many advantages, such as good scalability and low leakage, PCM suffers from exceptionally slow write operations, which makes it challenging to be integrated in the memory hiearchy. In this paper, we propose architectural innovations to improve the access time of MLC PCM. Due to cell process variation, composition fluctuation and the relatively small differences among resistance levels, MLC PCM typically employs an iterative write scheme to achieve precise control, which suffers from large write access latency. To address this issue, we propose write truncation (WT) to reduce the number of write iterations with the assistance of an extra error correction code (ECC). We also propose form switch (FS) to reduce the storage overhead of the ECC. By storing highly compressible lines in SLC form, FS improves read latency as well. Our experimental results show that WT and FS improve the effective write/read latency by 57%/28% respectively, and achieve 26% performance improvement over the state of the art.

21. Yangyang Pan, Guiqiang Dong, Qi Wu, Tong Zhang. Quasi-Nonvolatile SSD: Trading Flash Memory Nonvolatility to Improve Storage System Performance for Enterprise Applications. HPCA 2012.

This paper advocates a quasi-nonvolatile solid-state drive (SSD) design strategy for enterprise applications. The basic idea is to trade data retention time of NAND flash memory for other system performance metrics including program/erase (P/E) cycling endurance and memory programming speed, and meanwhile use explicit internal data refresh to accommodate very short data retention time (e.g., few weeks or even days). We also propose SSD scheduling schemes to minimize the impact of internal data refresh on normal I/O requests. Based upon detailed memory cell device modeling and SSD system modeling, we carried out simulations that clearly show the potential of using this simple quasi-nonvolatile SSD design strategy to improve system cycling endurance and speed performance. We also performed detailed energy consumption estimation, which shows the energy consumption overhead induced by data refresh is negligible.

22. Arun Raghavan, Yixin Luo, Anuj Chandawalla, Marios Papaefthymiou, Kevin P. Pipe, Thomas F. Wenisch, Milo M. K. Martin. Computational Sprinting. HPCA 2012.

Although transistor density continues to increase, voltage scaling has stalled and thus power density is increasing each technology generation. Particularly in mobile devices, which have limited cooling options, these trends lead to a utilization wall in which sustained chip performance is limited primarily by power rather than area. However, many mobile applications do not demand sustained performance; rather they comprise short bursts of computation in response to sporadic user activity. To improve responsiveness for such applications, this paper explores activating otherwise powered-down cores for sub-second bursts of intense parallel computation. The approach exploits the concept of computational sprinting, in which a chip temporarily exceeds its sustainable thermal power budget to provide instantaneous throughput, after which the chip must return to nominal operation to cool down. To demonstrate the feasibility of this approach, we analyze the thermal and

electrical characteristics of a smart-phone-like system that nominally operates a single core (_1W peak), but can sprint with up to 16 cores for hundreds of milliseconds. We describe a thermal design that incorporates phase-change materials to provide thermal capacitance to enable such sprints. We analyze image recognition kernels to show that parallel sprinting has the potential to achieve the task response time of a 16W chip within the thermal constraints of a 1W mobile platform.

23. John Sartori, Ben Ahrens, Rakesh Kumar. Power Balanced Pipelines. HPCA 2012.

Since the onset of pipelined processors, balancing the delay of the microarchitectural pipeline stages such that each microarchitectural pipeline stage has an equal delay has been a primary design objective, as it maximizes instruction throughput. Unfortunately, this causes significant energy inefficiency in processors, as each microarchitectural pipeline stage gets the same amount of time to complete, irrespective of its size or complexity. For poweroptimized processors, the inefficiency manifests itself as a significant imbalance in power consumption of different microarchitectural pipestages. In this paper, rather than balancing processor pipelines for delay, we propose the concept of power balanced pipelines – i.e., processor pipelines in which different delays are assigned to different microarchitectural pipestages to reduce the power disparity between the stages while guaranteeing the same processor frequency/performance. A specific implementation of the concept uses cycle time stealing [19] to deliberately redistribute cycle time from low-power pipeline stages to power-hungry stages, relaxing their timing constraints and allowing them to operate at reduced voltages or use smaller, less leaky cells. We present several
static and dynamic techniques for power balancing and demonstrate that balancing pipeline power rather than delay can result in 46% processor power reduction with no loss in processor throughput for a full FabScalar processor over a power-optimized baseline. Benefits are comparable over a Fabscalar baseline where static cycle time stealing is used to optimize achieved frequency. Power savings increase at lower operating frequencies. To the best of our knowledge, this is the first such work on microarchitecturelevel power reduction that guarantees the same performance.

24. Tianshi Chen, Yunji Chen, Qi Guo, Olivier Temam, Yue Wu, Weiwu Hu. Statistical Performance Comparisons of Computers. HPCA 2012.

As a fundamental task in computer architecture research, performance comparison has been continuously hampered by the variability of computer performance. In traditional performance comparisons, the impact of performance variability is usually ignored (i.e., the means of performance measurements are compared regardless of the variability), or in the few cases where it is factored in using parametric confidence techniques, the confidence is either erroneously computed based on the distribution of performance measurements (with the implicit assumption that it obeys the normal law), instead of the distribution of sample mean of performance measurements, or too few measurements are considered for the distribution of sample mean to be normal. We first illustrate how such erroneous practices can lead to incorrect comparisons. Then, we propose a non-parametric Hierarchical Performance Testing (HPT) framework for performance comparison, which is significantly more practical than standard parametric techniques because it does not require to collect a large number of measurements in order to achieve a normal distribution of the sample mean. This HPT framework has been implemented as an open-source software.