Project Description

Processor designers are coming at the cross road of a new era of processor architectures. Undisputably, to improve single processor performance relying on cranking up clock frequency and exploting instruction-level parallelism (ILP) is running out of steam due to several physical limitations including power wall, increasing design complexity and turnaround time, verfication cost, the nature within single-threaded applications, and the scale of economies. At the meantime, there is no sign of slowdown for realizing Moore's law in the near future. Given the trend of feature-size scaling and process technology advancement, it is predicted that integrating 10 to 100 billion transistors on a reasonable die area will become feasible by 2015 [6]. Instead of continuing to enlarge on-die cache capacity, the entire computing industry unanimously envisions that a multi-core or many-core architecture is the de-facto standard in all future processor segments from high performance data centers all the way down to the emerging mobile internet devices. Several integration thrusts are being pursued. Most of the mainstream processor vendors have leveraged off-the-shelf processor economies of scale and provided symmetric, homogeneous multi-core solutions while other designers, e.g., in the embedded domain, integrated heterogeneous processing elements on a chip to accelerate certain classes of applications. Regardless of the target markets, the concensus is to balance the use of transistors for both the computing engines (cores) as well as the supporting structure such as storage or performance enhancement feature (uncores) to synergistically optimize performance and power. This paradigm shift not only alters the way computer hardware is designed, it also substantially impacts the product development in software industries, and profoundly changes the way we teach students who use computers to solve various engineering problems.

1 Motivation — "Parallel" Evolution of Computer Architecture Research

Over the last 20 years, computer architecture researchers have been developing different techniques to address different types of issues. For example, single processor people mostly concentrated on developing microarchitectural techniques to improve the performance of single-threaded programs. These approaches attempt to address the problem of instruction supply and data supply, and eventually to reduce memory latency or exploit ILP. Even though most recent performance research effort has turned the focus to memory level parallelism (MLP) [8, 21, 38, 40] or branch-mispredict level parallelism (BLP) [32], the ultimate goal is still the same — hiding latency or minimizing performance side-effect as much as they can. At the meantime, researchers of parallel architectures mainly focused on the following issues: automatic parallel compiler techniques, high-efficiency interconnection network, low-overhead coherence and memory consistency models, productive programming models, etc. As increasing the number of cores and their heterogenity becomes the trend of future multi-core architectures, providing a more productive programming model has brought architects and programming language designers together more closely to scrutinize the interface in-between hardware and software. While these thrusts taken by ILP processor researchers and parallel architecture researchers may have similar objectives initially, i.e., improving the overall performance with innovative architectural solutions, the outcomes of the directions taken up by them are, however, quite diverged in hindsight.

More than a decade ago, the concept of speculative multithreading was first populated and advocated to exploit long-range ILP based on sequential, single-threaded program semantics. In this seminal paper, *Multiscalar processors* [54], published in ISCA-22, researchers at Univeristy of Wisconsin recognized the thread level parallelism exisiting across tasks and pioneered a new breed of microarchitecture employing an agressive speculation mechanism to exploit distant parallelism. Their framework proposed to break a sequential program into ordered parallel tasks and executing each task in a simple execution engine with multiple program counters. By using dependence speculation, a sequential program running on a multiscalar processor will be able to exploit ILP across split tasks. Later on, this work triggered successive research along the same line. A variety of forms and techniques that exploit thread level parallelism in a speculative manner were widely investigted over the last decade in both academia and industries. **(***HHL: What are commercialized? ***)**

Around the same time, the concept of transactional memory was proposed by Herlihy and Mott in ISCA-20 [25] as an alternate for designing multiprocessor architectures. Transactional memory systems aim at providing a lock-free programming model for highly concurent systems, which has become inreasingly more favorable for several reasons. First, it is evident that parallelization is the only key to unlock the massive amount of performance for applications in future systems. Secondly, a compiler capable of generating parallel codes automatically will not likely be available any time soon. Therefore, programmers are obligatory to parallelize the codes themselves on multicore processors for achieving their desirable level of performance. Thirdly, from what we have learned in programming (massively) parallel processors or supercomputers during the 90s, history will repeat itself that programming a multicore processor using conventional locks is extremely error-prone and very difficult to debug without any hardware debugging support. Fourthly, software developers are expected to maintain exactly the same productivity of writing sequential programs when they write parallel programs for multicore systems. Given all the above, we can conclude that the success of multicore processors highly depends on the following challenging assumptions: (1) the same productivity of programmers, and (2) an anticipated continuous performance improvement achieved for each multicore processor proliferation delivered. To say the least, transactional memory systems appear to address several issues described above and has emerged as one plausible solution for guaranteeing the continuing success of the entire computing industry. More encouragingly, the *Rock processor* to be released by Sun Microsystems has announced the incorporation of a transactional memory implementation, marking the first production processor with the support of hardware transactional memory [11, 34].

If one examines closely the underlying architectural support needed for supporting thread level speculation and transactional memory, it is not difficult to find the principle behind these two techniques are very similar in nature.

To address the divergence and streamline these research thrusts, this research proposes to investigate a unified multi-core architecture to put both thread level speculation and transactional memory into the same box.

The main goal of this research is to

2 Challenges in Many-Core Design

2.1 Energy-Performance Issues in Many-Core Architecture

Issue 1: Power Scalability.

Issue 2: Efficiency of Interconnection Architecture.

2.2 Modern Issues for MPPs

Challenge 1: On-Chip Wire Latency.

Challenge 2: Efficient Interaction with a Host Processor.

Challenge 3: Backward/Forward Binary Compatibility.

Challenge 4: Extensibility.

Implementation	Category	Architecture	Operand Passing Network	Out-of-order Spawn	Lock Parallelization
Multiscalar [55]	TLS	Dedicated	Y	-	-
SVC [22]	TLS	SMP	Y	-	-
Hydra [23]	TLS	CMP	-	Y	-
PolyFlow [1]	TLS	SMT	-	Y	-
TLS4OutOrder [41, 30]	TLS	CMP	-	Y	-
Voltron [71, 28]	TLS	CMP	Y	Y	?
Unified TLS+TM Multicore	TM + TLS	CMP	-	Y	Y
TCC [24]	TM	CMP	-	-	Y
UTM [2]	TM	CMP	-	-	Y
LogTM [35, 36]	TM	CMP	-	-	Y

Table 1: Comparison of Coarse Level Parallelism-Enabling Techniques

3 Proposed Research: A Unified Multicore Architecture

3.1 Research Focus 1:

3.1.1 Task:

3.2 Research Focus 2: Design for Inter-Core Communication

Due to the integration of multiple cores on the same die, the overheads of inter-core communication have been substantially reduce compared to those in the parallel machines in the good old days. Owing to this, the communication needed for enabling thread level speculation is no longer prohibitively expensive. For instance, for fine-grained thread level speculation in many proposed speculative multithreading execution model, copying and transfering register contents or even a snapshop of cache memory was a main parameter in the performance cost function that cannot be ligthly ignored. In contrast, due to the tightly-coupling of cores in a multicore processor, the communication overhead for thread level speculation can be much relieved via either a dedicated channel in-between cores or using the shared cache space. In this research, we will investigate hardware mechanisms and their trade-offs for achieving efficient communication from the perspective of launching speculative threads. We will quantify the performance/cost impact by using either shared memory space or cloning register files to minimize the overheads. When designging dedicated hardware channels for cloning register files, one question to be addressed is the scalability issue. It is quite easy to have a dedicated channel for two cores. As the number of cores is increased, what will be the most area- and power-efficient structure for attaining such purposes? This communication channel, in fact, will also be useful when it comes to security monitoring, logging, and rollback. The same channel can be used for sending instructions or data for security inspection. We will discuss such application in Section 3.4.

(*HHL: Inherent Synchrnization in TM ***)**

3.3 Research Focus 3: Thread Level Speculation with Heterogeneous Resources

Hetergeneous multiprocessing has opened up a new area of exploiting all computational capability provided by such a system. Current type of such architectures integrate a generic multicore processor with a general-purpose graphics processing unit (GPGPU) onto a die. Future systems, such as Intel's Larrabee [43], will likely integrate more such resources onto the same die. A GPGPU or an accelerator either on the same die of a processor or on the same system, can be used to achieve higher energy efficiency for data-parallel workloads. A middleware such as a runtime system or a specialized hardware can be designed to break up and dispatch the workloads to utilize these heterogeneous resources more efficiently. For example, the EXOCHI and CHI were developed by Intel to provide a unified programming framework that supports tightly-coupled integration of heterogeneous computing resources on a system [64]. The Merge framework intends to provide a high-level library system with the assistance of an enahnced map-reduce based programming language to better exploit the richness of the computation resources [29]. Similarly, Industry thrusts such as RapidMinds [33] or OpenCL [37] aim at defining general data parallel APIs to enable processing in the SIMD or SPMD style.

Toward this end, in this research, we would like to further extend this execution model by applying thread-level speculation to take these heterogeneous resources on a multicore system into account. There are several questions to be answered. What are the efficient communication model and recovery mechanisms when launching speculative threads on heterogeneous resources? How does this type of speculation changes the foundation of a hardware-supported transactional memory systems? How does a transactional memory help improve performance for this new thread execution model?

3.4 Research Focus 4: Support for Other Applications

The checkpoint and rollback mechanisms are also often used in other areas for a long time. Two primary areas which may benefit from this research are security and reliability.

3.4.1 Task: Memory Logging for Secure Architectures

Prior works have demonstrated architectural mechanisms using backup logs to enable a revivable system [47, 7, 39]. For example, in PI's INDRA work [47], a delta page based approach was proposed to enable high speed memory state backup and instant rollback when a security violation was detected for a network service transaction. The propossed mechanism requires hardware extension for the TLB and automatic checkpointing for updated memory pages. There are certain additional requirements for guaranteeing the integrity of the checkpointed memory pages due to security considerations. These pages should be allocated in a protected memory space, unexposed to the network. Although it is similar to the architectural support for hardware transactional memory, investigation needs to be done by taking security requirement into account.

3.4.2 Task: Reliability and Fault Tolerance

4 Evaluation Methodology

4.1 Phase 1: Feasibility and Proof-of-Concept Studies for POD

A cycle-level POD architecture simulator will be developed to carry out our performance study. The simulator will be general enough with configuration knobs to enable a broad range of design space exploration. We will use x86 as the substrate given its popularity and use their SSE instruction set as the SIMD instruction option for acceleration in PAL. We propose a novel methodology to perform the entire POD emulation, that is, running the x86 instructions of the substrate natively on an Intel-based workstation while translating the PAL instructions on-the-fly, checking their dependencies, and executing them in a separate cycle-level simulator via x86-based function calls. At this stage, we plan to develop a parser integrated into our simulator framework to translate the instructions to be executed on the PAL. This simulator needs to model every single feature of the SIMD PEs and memory subsystem including RRQ, external TLB and memory controller. Furthermore, on-chip and off-chip communication bandwidth will be accurately modeled. The simulator will contain well-defined, semantics-independent simulation modules so that it can be integrated into any available architectural simulator later. This new approach will substantially reduce a re-implementation when evaluating different substrates, increasing portability of our framework.

To quantify the performance of the POD architecture, a number of data-parallel benchmark application will be ported to the POD architecture. A few examples are FFT, MPEG encoding/decoding, option pricing (finance), graphics processing, and RMS type of applications [12]. As widely known, automatic code parallelization is a difficult task, many of such techniques are still under research. At this stage, we will rely on hand-optimized acceleration codes rather than developing a full-fledged POD compiler as we will focus on understanding the potential and trade-off of POD architecture as the first priority.

In addition to the architectural performance evaluation, circuits level study will be carried out simultaneously, primarily for estimating the area and power consumption of the PAL layer. Our objective is to justify the use and the size of these heterogeneous, simpler cores on the acceleration layer for delivering the best-in-class power and area efficiency.

Toward this goal, critical paths, power analysis, and area estimation need to be full understood for not only the SIMD processing elements and the general purpose cores but also the point-to-point interconnection network and required buffers. One advantage of using 3D integration is the wire length and its implication on power consumption. We will also evaluate how much wire and clock power can be saved using POD approach. We will establish analytical models based on available technology information to quantify the power and area consumed by POD.

One main challenge of applying 3D integration is thermal dissipation and how it will further impact DRAM stacking. Note that DRAM will leak faster when operating under a high temperature condition and thus requires more frequent refresh to avoid data corruption. We plan to establish a more accurate thermal model for POD using our prior expertise in developing 3D design tools to understand the impact of thermal grading to the PAL and DRAM layers.

4.2 Phase 2: Evaluating Multi-POD Processing

The most challenging yet also the most interesting part of this research is to see what is the maximum potential in performance when multiple POD processors are put into one system. Our goal is to compare such a system against a generic MIMD and other large scale systems such as IBM BlueGene/L from the perspective of performance, energy, area, and cost. First of all, there are several research and technical issues that need to be resolved before we can evaluate a Multi-POD multiprocessor system. Most of them are related to dynamic PAL hardware partitioning issue as well as runtime scheduling as described in Section **??**. We will investigate viable techniques and perform performance simulation on such systems to quantify the benefit in power- and area-efficiency.

4.3 Phase 3: Prototyping

The next level of our evaluation is to perform a more detailed analysis by prototyping the POD architecture to corroborate our proof-of-concepts. Prototyping, mostly a functional attestation, is always considered a more aggressive undertaking for the effort involved. It can also be achieved in many different ways. Today, several commercial FPGAs come with either built-in processor cores or synthesizable soft-cores [68], making them ideal to emulate the POD architecture from functioning standpoint. In other words, we can use the built-in processor as the substrate, while designing a configurable PAL array using the FPGA. Each PE, RRQ, and other minor logic blocks need to be designed, replicated, and synthesized to construct a complete PAL. In reality, such design replication can similarly be done, reducing the design complexity and verification time. Through physical design process, it is our belief that we will gain more insights by prototyping a POD implementation to uncover more corner-case issues which are not to be easily identified during software emulation.

5 Comparisons with Prior Research

6 Education Aspects and Outreach Activities

Given the limits posed by several technology fronts including complexity-effectiveness, verification effort, fundamental physics, and scale of economies, multi-core and future many-core processors have become the universal solution for all computing segments ranging from high-throughput data centers down to mobile internet devices. As this paradigm shift is taking place, it necessitates certain fundamental reconsideration in both our undergraduate and graduate curriculums. In other words, *Think in Parallel!* will become inevitable starting from our very first freshmen computer engineering course.

Along this line, the PI and his colleagues at Georgia Tech (Karsten Schwan, Ada Gavrilovska, and Matt Wolf, Sudhakar Yalamanchili) have been putting a lot of endeavor for a new computer engineering curriculum in both ECE and CS departments to re-align several goals of our computer system education. The ultimate goal is to fulfill the demands of a new kind of computer engineers from this fast-changing industries. The effort started in 2006 with incentive education fund and gifts donated by Intel Corporation. We gradually continue to revamp our architecture, OS, parallel architecture and basic programming classes by developing infusing parallel modules into the lectures and projects. Toward this, the PI has been developing new and value-added course materials. They include how to exploit thread-level parallelism at both the hardware and software levels, how to program multi-core processors and the relevant compilation techniques, how to provide architectural and OS support for multi-threaded execution, and how to do performance analysis and debugging in a multi-threaded execution environment. Projects designed using Intel's Thread Checker and Thread Profiler bundled in Vtune Analyzer are used to enhance students' skill set. All these materials were made open source (Our Georgia Tech CERCS Multi-Core Repository can be found at http://pleuma.cc.gatech.edu/cercs/multicore/index.php/Main_Page) and have been shared by many other institutions. Together with another ECE faulty Aaron Lanterman, the PI designed a new course in multi-core and GPU programming, specifically targeting for 3D games and high-performance computing. In this course, new parallel programming models and languages such as Cg and HLSL, new development platform Direct3D/XNA/CUDA, etc., were taught. The PI designed and provided several project infrastucutres using Direct3D, XNA Game Studio and Cg/HLSL for GPGPUs and IBM Cell/BE to give students challenges, provoke their thoughts, and nurture their experiences in dealing with parallel programming in a more natural manner.

More recently, the PI was awarded an NSF CCLI Phase I Explorary grant to creat a multithreaded programming course that targets general-purpose multi-core processors at the senior undergraduate level. The PI proposed a problem-based learning (PBL) approach with real-life multi-core programming problems as an experiment for educating the engineering majors about parallel programming based on current off-the-shelf tools provided by the industries. This education plan was strongly supported and endorsed by Intel's VP of Research Dr. Andrew Chien. Working together with Prof. Wei Zhang from Southern Illinois University at Carbondale, the PI expects to create course materials, CDs, multithreaded programming mini-projects within two semesters (starting January, 2009) and will make their results available to other education institutes at the end of the project.

In general, multi-cores exist in several different forms among computing platforms. They can be classified into: (1) general purpose homogeneous multi-core processors offered by Intel and AMD, (2) high throughput systems such as Ultra Sparc T1/T2, IBM BlueGene/L, (3) heterogeneous multi-cores such as STI Cell processor, (4) system-on-chip implementation with discrete cores from several vendors in many embedded applications, (5) specialized acceleration engines such as Nvidia G80, Tesla or AMD/ATI's Radeon, or Aegia's Physics accelerator. Taking any of this system as a component, one can potentially construct an even larger scale parallel computing systems for special purposes. The PI will integrate these materials into their parallel computer architecture as well as his new multi-core programming class to enrich students with industry experiences.

The PI is also in contact with Sun Microsystems to discuss the use of OpenSparc in their computer architecture courses. OpenSparc is a license-free T1 core design, consisting of complete tool chains for students to learn a reasonably simple processor design in one semester. The PI will use this as a jumpstart point to enable homogeneous multi-core designs using OpenSparc and its tools as a foundation in his architecture classes.

On the other hand, the PI has been working and sponsored by Intel Corporation on using multicore processors for accelerating 3D medical image reconstruction. The PI's research team is closely working with researchers and engineers from Intel's Embedded Medical Division and Radisys, a system provider specialized in multicore solutions for medical imaing. The PI has published their work using Intel's dual socket quad-core processors [14] and is currently porting their OpenMP codes onto Intel's latest Nehalem processor and Nvidia's Tesla C870 board using CUDA.

Georgia Tech has two NSF-sponsored programs — Facilitating Academic Careers in Engineering and Science for African American (FACES) and Summer Undergraduate Research Experience for minorities (SURE), which aim to improve engineering education for under-represented students. The PI has supervised 2 African American graduate students (Apeworkin, Hammond currently at MIT) and 1 female graduate student (Viswanathan, current full-time employee at Intel.) The PI will continue his passion and endeavor in recruiting and encouraging more under-represented students to participate in his research projects. In addition, the PI had participated in a parallel curriculum workshop as a panelist joined by visitors from schools of Historically-Black Colleges and Universities (HBCU) in the southeast region this summer. The parallel course modules mentioned earlier have been adopted by several faculty from these colleges.

7 Results from Prior NSF Support

The PI's prior research supported by NSF includes the following activities.

ITR CCF-0326396: Collaborative Research: Morphable Software Services: Self-Modifying Programs for Distributed Embedded Systems, 10/2003-06/2007 (finished). (Lee was listed as a senior personnel.) The project investigated low-power processing techniques for sustaining collaborative morphable services under extreme, inaccessible conditions. The PI and his team have developed several microarchitectural techniques based on compression, access properties, and semantic region partitioning that reduce energy consumption in memory hierarchy including BTB, TLB, caches, shared caches in multi-core processors, and DRAMs. The outcomes of this work were published in [3, 4, 5, 17, 18, 10, 15, 13, 19, 20, 26, 27, 66].

ITR CNS-0325536: Toward Autonomous Computing Platforms: System-Wide Hardware/ Software Performance Monitoring and Adaptation, 10/2003-09/2008. This project focuses on developing a flexible, FPGA-assisted infrastructure for non-intrusive hardware monitoring across the entire system. The goal of the project is to construct a self-adapting, self-aware system with the assistance of microarchitectural support and the FPGA. The project is a collaboration between Lee and Sally McKee from Cornell University. At Georgia Tech, the PI and his students investigated the types and mechanisms of security monitoring required for trustworthiness and the design of monitor capsule using Xilinx Virtex-II boards. Currently, they have been implementing monitoring capsule in the FPGA using AVnet development board (Virtex-2 Pro based) to perform architectural co-simulation and coherence traffic analysis for multiprocessors. The PI and his team used FPGAs to explore the opportunities of accelerating architectural simulations for single-core and multi-core processors. The outcomes of this work have been published in [9, 16, 31, 42, 46, 52, 45, 44, 47, 48, 49, 50, 51, 53, 56, 57, 58, 59, 60, 61, 62, 72].

CAREER CNS-0644096: Introspective Computing: A Multicore Approach to Availability, Reliability, and Security, 06/2007-05/2012. This project investigates an introspective multi-core processor architecture that can perform fine-grained security introspection, instant low-overhead checkpoint, and fast, on-demand rollback recovery. The goal is to provide a synergistic and holistic solution toward the challenges of achieving high availability, reliability, and security for a computing system. The outcome was published in [63, 69, 70].

CPA CCF-0811738: Parallel-On-Demand — A Broad Purpose 3D-Integrated Performance Acceleration Layer for General Purpose Processors, 07/2008-07/2011. This project investigates a new many-core architecture which aims at providing the optimal performance-power and performance-area ratios. It studies a SIMD PE array integrated on top of general processors using 3D die-stacking technology. The outcome of this work was published in [65, 67].

References

[1] Mayank Agarwal, Kshitiz Malik, Kevin M. Woley, Sam S. Stone, and Matthew I. Frank. Exploiting postdominance for speculative parallelization. In *Proceedings of the 2007 International Symposium on High Performance Computer Architecture*, February 2007.

[2] C. Scott Ananian, Krste Asanovic, Bradley C. Kuszmaul, Charles E. Leiserson, and Sean Lie. Unbounded transactional memory. In *Proceedings of the Eleventh International Symposium on High-Performance Computer Architecture*, pages 316 – 327, February 2005.

[3] Chinnakrishnan S. Ballapuram, Hsien-Hsin S. Lee, and Milos Prvulovic. Synonymous Address Compaction for Energy Reduction in Data TLB. In *ISPLED'05: Proceedings of the 2005 ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 357–362, San Diego, CA, 2005.

[4] Chinnakrishnan S. Ballapuram, Kiran Puttaswamy, Gabriel H. Loh, and Hsien-Hsin S. Lee. Entropy-based low power data tlb design. In *Proceedings of the ACM/IEEE International Conference on Compilers Architecture and Synthesis for Embedded Systems*, 2006.

[5] Chinnakrishnan S. Ballapuram, Ahmad Sharif, and Hsien-Hsin S. Lee. Exploiting access semantics and program behavior to reduce snoop power in chip multiprocessors. In *Proceedings of the 13th ACM Internatioanl Conference on Architectural Support for Programming Languages and Operating Systems*, 2008.

[6] Shekhar Borkar. Thousand Core Chips — A Technology Perspective. In *Proceedings of the 44th Design Automation Conference*, 2007.

[7] Shimin Chen, Babak Falsafi, Phillip B. Gibbons, Michael Kozuch, Todd C. Mowry, Radu Teodorescu, Anastassia Ailamaki, Limor Fix, Gregory R. Ganger, Bin Lin, and Steven W. Schlosser. Log-based architectures for general-purpose monitoring of deployed code. In *ASID '06: Proceedings of the 1st workshop on Architectural and system support for improving software dependability*, pages 63–65, 2006.

[8] Yuan Chou, Brian Fahs, and Santosh Abraham. Microarchitecture optimizations for exploiting memory level parallelism. In *ISCA '04: Proceedings of the 31st International Symposium on Computer Architecture*, pages 76–87, 2004.

[9] Christopher R. Clark, Ripal Nathuji, and Hsien-Hsin S. Lee. Using an FPGA as a Prototyping Platform for Multi-core Processor Applications. In *Workshop on Architecture Research using FPGA Platforms in conjunction with International Symposium on High-Performance Computer Architecture (WARFP-05)*, February 2005.

[10] Yuvraj Singh Dhillon, Abdulkadir Utku Diril, Abhijit Chatterjee, and Hsien-Hsin Sean Lee. Algorithm for Achieving Minimum Energy Consumption in CMOS Circuits Using Multiple Supply and Threshold Voltages at the Module Level. In *ICCAD '03: Proceedings of the 2003 IEEE/ACM International Conference on Computer-Aided Design*, pages 693–700, 2003.

[11] Dave Dice, Maurice Herlihy, Doug Lea, Yossi Lev, Victor Luchangco, Wayne Mesard, Mark Moir, Kevin Moore, and Dan Nussbaum. Applications of the Adaptive Transactional Memory Test Platform. In *Proceedings of the 3rd ACM SIGPLAN Workshop on Transactional Computing*, 2008.

[12] Pradeep Dubey. Recognition, Mining and Synthesis Moves Computers to the Era of Tera. In *Technology@Intel Magazine*, February 2005.

[13] Mongkol Ekpanyapong, Pinar Korkmaz, and Hsien-Hsin S. Lee. Choice Predictor for Free. In *Proceedings of the 9th Asia-Pacific Computer System Architecture Conference*, pages 399–413, September 2004.

[14] Eric Fontaine and Hsien-Hsin S. Lee. Optimizing Katsevich Image Reconstruction Algorithm on Multicore Processors. In *Proceedings of the 13th IEEE International Conference on Parallel and Distributed Systems*, 2007.

[15] Joshua B. Fryman, Chad Huneycutt, Hsien-Hsin S. Lee, Kenneth M. Mackenzie, and David E. Schimmel. Energy-Efficient Network Memory for Ubiquitous Devices. *IEEE Micro special issue on Power Complexity Aware Design*, 23(5):60–70, September/October 2003.

[16] Lan Gao, Jun Yang, Marek Crobak, Youtao Zhang, San Nguyen, and Hsien-Hsin S. Lee. A Low-Cost Memory Remapping Scheme for Address Bus Protection. In *PACT'06: To appear in Proceedings of International Conference on Parallel Architectures and Compilation Techniques*, September 2006.

[17] Mrinmoy Ghosh and Hsien-Hsin S. Lee. Smart refresh: An enhanced memory controller design for reducing energy in conventional and 3d die-stacked drams. In *Proceedings of the 40th International Symposium on Microarchitecture*, 2007.

[18] Mrinmoy Ghosh and Hsien-Hsin S. Lee. Virtual exclusion: An architectural approach to reducing leakage energy in caches for multiprocessor systems. In *Proceedings of the 13th IEEE International Conference on Parallel and Distributed Systems*, 2007.

[19] Mrinmoy Ghosh, Emre Ozer, Stuart Biles, and Hsien-Hsin S. Lee. Efficient System-on-Chip energy Management with a Segmented Bloom Filter. In *ARCS '06: Proceedings of the 19th International Conference on Architecture of Computing Systems*, pages 283–297, March 2006.

[20] Mrinmoy Ghosh, Weidong Shi, and Hsien-Hsin S. Lee. CoolPression — A Hybrid Significance Compression Technique for Reducing Energy in Caches. In *Proceedings of the IEEE International System-on-Chip Conference*, pages 399–402, September 2004.

[21] Andy Glew. MLP Yes! ILP No! In *Wild and Crazy Idea Session held in conjunction with the ASPLOS XIII*, 1998.

[22] Sridhar Gopal, T. N. Vijaykumar, James E. Smith, and Gurindar S. Sohi. Speculative versioning cache. In *Proceedings of the 4th International Symposium on High-Performance Computer Architecture*, pages 195–205, 1998.

[23] Lance Hammond, Mark Willey, and Kunle Olukotun. Data speculation support for a chip multiprocessor. In *ASPLOS-VIII: Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 58–69, 1998.

[24] Lance Hammond, Vicky Wong, Mike Chen, Brian D. Carlstrom, John D. Davis, Ben Hertzberg, Manohar K. Prabhu, Honggo Wijaya, Christos Kozyrakis, and Kunle Olukotun. Transactional memory coherence and consistency. In *Proceedings of the 31st Annual International Symposium on Computer Architecture*, pages 102 – 113, June 2004.

[25] M. Herlihy and J. E. B. Moss. Transactional memory: Architectural support for lock-free data structures. In *Proceedings of the Twentieth Annual International Symposium on Computer Architecture*, 1993.

[26] Hsien-Hsin S. Lee and Chinnakrishnan S. Ballapuram. Energy Efficient D-TLB and Data Cache using Semantic-Aware Multilateral Partitioning. In *ISPLED'03: Proceedings of the 2003 ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 306–311, Seoul, Korea, August 2003.

[27] Hsien-Hsin S. Lee, Joshua B. Fryman, A. Utku Diril, and Yuvraj S. Dhillon. The Elusive Metric for Low-Power Architecture Research. In *Workshop on Complexity-Effective Design in conjunction with the 30th ACM/IEEE International Symposium on Computer Architecture (WCED-03)*, San Diego, California, June 2003.

[28] Steven A. Lieberman, Hongtao Zhong, and Scott Mahlke. *Extracting Statistical Loop-Level Parallelism using Hardware-Assisted Recovery*. Technical Report CSE-TR-528-07, University of Michigan, February 2007.

[29] Michael D. Linderman, Jamison D. Collins, Hong Wang, and Teresa H. Meng. Merge: A Programming Model for Heterogeneous Multi-core Systems. In *ASPLOS XIII: Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*, pages 287–296, 2008.

[30] Wei Liu, James Tuck, Luis Ceze, Wonsun Ahn, Karin Strauss, Jose Renau, and Josep Torrellas. POSH: A TLS compiler that exploits program structure. In *Proceedings of the 11th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 158–167, 2006.

[31] Chenghuai Lu, Tao Zhang, Weidong Shi, and Hsien-Hsin S. Lee. ”M-TREE: A High Efficiency Security Architecture for Protecting Integrity and Privacy of Software. *Journal of Parallel and Distributed Computing for a special issue on Security in Grid and Distributed Systems*, 66(9), 2006.

[32] Kshitiz Malik, Mayank Agarwal, Sam S. Stone, Kevin M. Woley, and Matthew I. Frank. Branch-mispredict Level Parallelism (BLP) for Control Independence. In *Proceedings of the 14th Annual Symposium on High Performance Computer Architecture*, 2008.

[33] Michael D. McCool, Kevin Wadleigh, Brent Henderson, and Hsin-Ying Lin. Performance evaluation of gpus using the rapidmind development platform. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 181, 2006.

[34] Mark Moir, Kevin Moore, and Dan Nussbaum. The Adaptive Transactional Memory Test Platform: A Tool for Experimenting with Transactional Code for Rock. In *Proceedings of the 3rd ACM SIGPLAN Workshop on Transactional Computing*, 2008.

[35] Kevin E. Moore, Jayaram Bobba, Michelle J. Moravan, Mark D. Hill, and David A. Wood. LogTM: Log-based transactional memory. In *Proceedings of the 12th International Conference on High Performance Computer Architecture*, pages 254 – 265, February 2006.

[36] Michelle J. Moravan, Jayaram Bobba, Kevin E. Moore, Luke Yen, Mark D. Hill, Ben Liblit, Michael M. Swift, and David A. Wood. Supporting nested transactional memory in LogTM. In *ASPLOS-XII: Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 359–370, 2006.

[37] Aagtab Munshi. OpenCL: Parallel Computing on the GPU and CPU. Courses: Beyond Programmable Shading: Fundamentals, SIGGRAPH 2008.

[38] Onur Mutlu and Thomas Moscibroda. Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, pages 63–74, 2008.

[39] Edmund B. Nightingale, Daniel Peek, Peter M. Chen, and Jason Flinn. Parallelizing security checks on commodity hardware. In *ASPLOS XIII: Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*, pages 308–318, 2008.

[40] Moinuddin K. Qureshi, Daniel N. Lynch, Onur Mutlu, and Yale N. Patt. A case for mlp-aware cache replacement. In *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*, pages 167–178, 2006.

[41] Jose Renau, James Tuck, Wei Liu, Luis Ceze, Karin Strauss, and Josep Torrellas. Tasking with out-of-order spawn in TLS chip multiprocessors: Microarchitecture and compilation. In *ICS '05: Proceedings of the 19th Annual International Conference on Supercomputing*, pages 179–188, 2005.

[42] Martin Schulz, Brian S. White, Sally A. McKee, Hsien-Hsin S. Lee, and Jurgen Jeitner. Owl: Next Generation System Monitoring. In *Proceedings of the ACM Computing Frontier 2005*, pages 116–124, 2005.

[43] Larry Seiler, Doug Carmean, Eric Sprangle, Tom Forsyth, Michael Abrash, Pradeep Dubey, Stephen Junkins, Adam Lake, Jeremy Sugerman, Robert Cavin, Roger Espasa, Ed Grochowski, Toni Juan, and Pat Hanrahan. Larrabee: a Many-core x86 Architecture for Visual Computing. *ACM Transactions on Graphics*, 27(3):1–15, 2008.

[44] Weidong Shi, Joshua B. Fryman, Guofei Gu, Hsien-Hsin S. Lee, Youtao Zhang, and Jun Yang. InfoShield: A Security Architecture for Protecting Information Usage in Memory. In *HPCA-12: Proceedings of the 12th International Symposium on High-Performance Computer Architecture*, pages 225–234, 2006.

[45] Weidong Shi and Hsien-Hsin S. Lee. Authentication control point and its implications for secure processor design. In *Proceedings of the 39th International Symposium on Microarchitecture*, pages 103–112, 2006.

[46] Weidong Shi and Hsien-Hsin S. Lee. Accelerating memory decryption and authentication with frequent value prediction. In *Proceedings of the ACM International Conference on Computing Frontiers*, pages 35–46, 2007.

[47] Weidong Shi, Hsien-Hsin S. Lee, Laura Falk, and Mrinmoy Ghosh. An Integrated Framework for Dependable and Revivable Architecture Using Multicore Processors. In *ISCA '06: Proceedings of the 32nd Annual International Symposium on Computer Architecture*, pages 102–113, 2006.

[48] Weidong Shi, Hsien-Hsin S. Lee, Mrinmoy Ghosh, and Chenghuai Lu. Architectural Support for High Speed Protection of Memory Integrity and Confidentiality in Symmetric Multiprocessor Systems. In *PACT'04: Proceedings of International Conference on Parallel Architectures and Compilation Techniques*, pages 123–134, September 2004.

[49] Weidong Shi, Hsien-Hsin S. Lee, Mrinmoy Ghosh, Chenghuai Lu, and Alexandra Boldyreva. High Efficiency Counter Mode Security Architecture via Prediction and Precomputation. In *ISCA '05:*

Proceedings of the 32nd Annual International Symposium on Computer Architecture, pages 14–24, 2005.

[50] Weidong Shi, Hsien-Hsin S. Lee, Chenghuai Lu, and Mrinmoy Ghosh. Towards the issues in architectural support for protection of software execution. *SIGARCH Computer Architecture News*, 33(1):6–15, 2005.

[51] Weidong Shi, Hsien-Hsin S. Lee, Chenghuai Lu, and Tao Zhang. Attacks and Risk Analysis for Hardware Supported Software Copy Protection Systems. In *DRM '04: Proceedings of the 4th ACM workshop on Digital Rights Management*, pages 54–62, 2004.

[52] Weidong Shi, Hsien-Hsin S. Lee, Richard M. Yoo, and Alexandra Boldyreva. A digital right enabled graphics processing system. In *Proceedings of the ACM SIGGRAPH/Eurographics Workshop of Graphics Hardware*, pages 17–26, 2006.

[53] Weidong Shi, Chenghuai Lu, and Hsien-Hsin S. Lee. Memory-centric Security Architecture. In *Proceedings of the 2005 International Conference on High Performance Embedded Architectures and Compilers*, pages 153–168, 2005.

[54] Gurindar S. Sohi, Scott E. Breach, and T. N. Vijaykumar. Multiscalar processors. In *ISCA '95: Proceedings of the 22nd annual international symposium on Computer architecture*, pages 414–425, New York, NY, USA, 1995. ACM.

[55] Gurindar S. Sohi, Scott E. Breach, and T. N. Vijaykumar. Multiscalar processors. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 414–425, 1995.

[56] Taeweon Suh, Douglas M. Blough, and Hsien-Hsin S. Lee. Supporting cache coherence in heterogeneous multiprocessor systems. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition Volume II (DATE'04)*, page 21150. IEEE Computer Society, 2004.

[57] Taeweon Suh, Daehyun Kim, and Hsien-Hsin S. Lee. Cache Coherence Support for Non-Shared Bus Architecture on Heterogeneous MP SoCs. In *Proceedings of the 42nd Design Automation Conference (DAC-42)*, pages 553–558, June 2005.

[58] Taeweon Suh, Hsien-Hsin S. Lee, and Douglas M. Blough. Integrating Cache Coherence Protocols for Heterogeneous Multiprocessor Systems, Part I. *IEEE Micro special issue on Embedded Systems: Architecture, Design and Tools*, pages 33–41, July/August 2004.

[59] Taeweon Suh, Hsien-Hsin S. Lee, and Douglas M. Blough. Integrating Cache Coherence Protocols for Heterogeneous Multiprocessor Systems, Part II. *IEEE Micro*, pages 70–78, September/October 2004.

[60] Taeweon Suh, Hsien-Hsin S. Lee, Shih-Lien Lu, and John Shen. Initial Observations of Hardware/Software Co-Simulation using FPGA in Architecture Research. In *Workshop on Architecture Research using FPGA Platforms in conjunction with International Symposium on High-Performance Computer Architecture (WARFP-06)*, February 2006.

[61] Taeweon Suh, Hsien-Hsin S. Lee, Sally A. McKee, and Martin Schulz. Evaluating System-wide Monitoring Capsule Design Using Xilinx Virtex-II Pro FPGA. In *Workshop on Architecture Research using FPGA Platforms in conjunction with International Symposium on High-Performance Computer Architecture (WARFP-05)*, February 2005.

[62] Taeweon Suh, Shih-Lien L. Lu, and Hsien-Hsin S. Lee. An fpga approach to quantifying coherence traffic efficiency on multiprocessor systems. In *Proceedings of the 17th International Conference on Field Programmable Logic and Applications*, 2007.

[63] Vikas R. Vasisht and Hsien-Hsin S. Lee. SHARK: Architectural Support for Autonomic Protection Against Stealth by Rootkit Exploits. In *Proceedings of IEEE/ACM 41th International Symposium on Microarchitecture*, 2008.

[64] Perry H. Wang, Jamison D. Collins, Gautham N. Chinya, Hong Jiang, Xinmin Tian, Milind Girkar, Nick Y. Yang, Guei-Yuan Lueh, and Hong Wang. Exochi: architecture and programming environment for a heterogeneous multi-core multithreaded system. In *PLDI '07: Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation*, pages 156–166, 2007.

[65] Dong Hyuk Woo, Joshua B. Fryman, Allan D. Knies, Marsha Eng, and Hsien-Hsin S. Lee. POD: A 3D-Integrated Broad-Purpose Acceleration Layer. *IEEE Micro Special Issue on Accelerator Architectures*, pages 28–40, July/August 2008.

[66] Dong Hyuk Woo, Mrinmoy Ghosh, Emre Ozer, Stuart Biles, and Hsien-Hsin S. Lee. Reducing energy of virtual cache synonym lookup using bloom filters. In *Proceedings of the ACM/IEEE International Conference on Compilers Architecture and Synthesis for Embedded Systems*, 2006.

[67] Dong Hyuk Woo and Hsien-Hsin S. Lee. Extending Amdahl's Law for Energy-Efficient Computing in Many-Core Era. *To appaer in IEEE Computer*.

[68] Xilinx. Virtex-II Pro FPGA, http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/ virtex_ii_pro_fpgas/index.htm .

[69] Richard M. Yoo and Hsien-Hsin S. Lee. Adaptive Transaction Scheduling for Transactional Memory Systems. In *Proceedings of the 20th ACM Symposium on Parallelism in Algorithms and Architectures in the Special Track on Hardware and Software Techniques to Improve the Programmability of Multicore Machines*, pages 169–178, 2008.

[70] Richard M. Yoo, Yang Ni, Adam Welc, Bratin Saha, Ali-Reza Adl-Tabatabai, and Hsien-Hsin S. Lee. Kicking the Tires of Software Transactional Memory: Why the Going Gets Tough. In *Proceedings of the 20th ACM Symposium on Parallelism in Algorithms and Architectures in the Special Track on Hardware and Software Techniques to Improve the Programmability of Multicore Machines*, pages 265–274, 2008.

[71] Hongtao Zhong, Steven Lieberman, and Scott Mahlke. Extending multicore architectures to exploit hybrid parallelism in single-thread applications. In *Proceedings of the 2007 International Symposium on High Performance Computer Architecture*, February 2007.

[72] Xiaotong Zhuang, Tao Zhang, Hsien-Hsin S. Lee, and Santosh Pande. Hardware Assisted Control Flow Obfuscation for Embedded Processors. In *Proceedings of the 2004 International Conference on Compilers, Architectures, Synthesis on Embedded Systems (CASES-04)*, Washington D.C., 2004.