Research Statement

Mattan Erez
Stanford University
http://www.stanford.edu/people/mattan.erez mattan.erez@stanford.edu

My research focus in computer architecture is on the critical aspects of locality, parallelism, and bandwidth. This work encompasses improving the cooperation between the hardware, compiler, and programmer. I believe that this direction is the key to enabling new levels of performance, efficiency, and code-portability.

Early in my research career, I worked on low-level architecture and micro-architecture for high-performance super-scalar processors. I am currently the student leader of the Merrimac Streaming Supercomputer project at Stanford. Working on Merrimac allows me to pursue my interests of whole-system research by collaborating with scientists from several departments, and working on the processor architecture, system architecture, compiler, programming languages, and applications for Merrimac. In my future research, I will develop new hardware abstractions that will empower the programmer, compiler, and hardware to cooperate in an effective manner, thus increasing performance and efficiency. I will pay particular attention to the fundamental properties of locality, parallelism, and bandwidth constraints while striving to overcome the limitations of today's architectures.

Past and Current Research

For conventional processors, I developed micro-architectural techniques that address the difficult issues of efficiently supplying a large number of instructions on every processor cycle [1] and scheduling the instructions for execution in the face of dynamic latencies [2, 3, 4, 5]. At a higher architectural level, I suggested modifications to a super-scalar instruction set architecture (ISA) to allow for better cooperation between the compiler and hardware. This simple extension to the ISA along with modification of the GCC compiler enabled the compiler to communicate register-liveness information, which significantly reduced the effects of register spilling and reloading, and improved both performance and power consumption [6].

More recently, I have been working on stream architectures for scientific and engineering computing as part of the Stanford Merrimac Streaming Supercomputer project [7, 8, 9, 10, 11]. Merrimac is a computer system developed with an integrated view of the applications, software system, compiler, and architecture. This integrated view leads to an order of magnitude gain in performance per unit cost and power compared to common scientific computers designed around conventional CPUs. The key contributions of my thesis work on Merrimac are in demonstrating the effectiveness of compute-intensive stream architectures for scientific computing, refining stream hardware, software tools, and programming interfaces for scientific applications, and developing novel fault-tolerance schemes.

Merrimac is a fully-programmable stream architecture based on a deep register hierarchy and a highly parallel execution core [7]. The architecture is designed to take advantage of the strengths of modern VLSI — very high bandwidth over short distances and very high transistor counts — and match them with the characteristics of scientific codes — large amounts of parallelism and data access locality. While developing the Merrimac processor architecture, I led a team of doctoral students that performed low-level area, power, and performance-characteristic studies. We fully specified the architecture and micro-architecture of Merrimac, including the design of specialized mechanisms that answer specific application needs. Examples of such mechanisms are a functional unit that accelerates the calculation of reciprocals and square-roots, a unique scatter-add mechanism that accelerates superposition type calculations [8], and techniques that improve on-chip stream storage management and reduce off-chip bandwidth requirements [9].

Regarding applications, I primarily worked on the execution of unstructured algorithms on Merrimac, which are commonly used in scientific and engineering computations. The data-dependent nature of these algorithms makes taking advantage of parallelism and locality particularly challenging and requires new features in both hardware and software, ranging from language directives to micro-architecture. I enjoyed collaborating with Prof. Eric Darve of the Mechanical Engineering Department and members of Prof. Vijay Pande's group in the Structural Biology Department on an unstructured molecular-dynamics (protein-folding) application [10] (received a best paper award), and with Dr. Timothy J. Barth from the NASA Ames research center and Dr. Frank Ham of Stanford's Center for Integrated Turbulence Research on finite element and finite volume methods.

I have also worked together with Dr. Barth (NASA), Prof. Dally (Stanford), Prof. Hanrahan (Stanford), and Dr. Mark Seager (Lawrence Livermore National Lab) on writing a detailed proposal for developing a Merrimac prototype in response to the NASA H&RT BAA (04-02). The proposal discussed applications relating to radiative transfer, hydrodynamics, and molecular dynamics; processor and system architectures; run-time systems; compilers; and the design, fabrication, and assembly of the prototype. In writing the proposal, we also initiated collaboration with several industrial partners.

While discussing Merrimac with scientists, they stressed a critical requirement of the scientific computing community – the reliability of the computer system and the correctness of the final results. Therefore, we incorporated fault-tolerance in Merrimac's architecture from its inception. I developed the overall reliability strategy, performed the fault-susceptibility estimates, suggested several novel mechanisms for fault-detection and evaluated their effectiveness on a range of applications [11].

A very exciting aspect of my work is developing the software tools and models required for programming compute-intensive architectures. I participated in the successful development of the Brook language, now used to program modern graphics processors, was involved in the effort to build a stream compiler, and am taking part in the design of the novel Sequoia programming model and compiler targeting a wide range of emerging throughput-oriented processors [12].

Future Research

In my future research, I will develop architectural mechanisms and abstractions that express locality, parallelism, and bandwidth. I will also work on compiler technology to use the abstractions to target specific low-level features of emerging and existing processors. I consider exporting the appropriate hardware abstractions and exploiting them in the compiler to be the key to tackling the main problems facing architects today, including: meeting power and energy constraints; providing reliable execution with low hardware costs; achieving performance goals in diverse environments; and managing the ever-growing complexity both in design and in verification. I am very enthusiastic about pursuing these goals in an academic setting where opportunities for collaborations across diverse fields are abundant and where revolutionary thinking is encouraged.

My focus will be on high-performance computing platforms, which consistently push the state of the art in software, compiler technology, architectures, circuits, and devices. Given that most modern features found in today's commodity processors have been originally developed for scientific and high-end applications, my work will impact not only high-performance computing but the wider systems community as well.

In order to achieve the large improvements required to overcome the challenges outlined above, we must break with tradition and create new principles for designers and programmers. I will apply a whole-system view, and develop relevant and consistent abstractions at both the micro-architecture/compiler and compiler/programmer interfaces. In particular, I will concentrate on the need to maximize locality and parallelism while restricting bandwidth usage.

Software today is written in sequential style languages and much effort is spent on algorithms that minimize the number of operations at the cost of memory bandwidth and locality of reference. However, the trends in VLSI technology are the opposite, with parallel operations becoming cheap when locality is maintained and bandwidth conserved. I plan to incorporate these ideas into the programming model and compiler and to collaborate with algorithm developers to account for the new tradeoffs afforded by technology. Two examples are re-materializing data as opposed to re-reading it from memory, and applying more accurate and computationally-intensive algorithms to reduce the overall time to solution by requiring fewer iterations.

At the hardware level, I will start my research by focusing on new opportunities for hardware–software cooperation in heterogeneous, many-core processors. Traditionally, all coordination between operations has been done in software only, using a coarse-grained "driver model", or in hardware, with a fine-grained "instruction model". Yet, when multiple specialized cores will interact in tomorrow's processors we are likely to see operations that are not well suited to either approach and will require a new methodology for medium-grained scheduling and interaction. My longer-term plans include re-examining the task partitioning between software and hardware. Current programmable architectures tend to place all the burden on either software or hardware and display extremes of very coarse- or fine-grained control. For example, FPGAs use very fine-grained software control where the hardware is programmable at a circuit level, while most other architectures suggested today are a collection of entire micro-processors. I will work on a systematic evaluation of the strengths and weaknesses of circuits and software for common tasks and develop a new methodology and architecture, which includes medium-grained structures. In addition to the efficiency gains due to better task partitioning, the programmable architecture will stress locality, parallelism, and bandwidth for high performance and energy conservation. Some simple examples of medium-grained structures are auto-increment and similar primitive operations required by state-machines. Such building blocks are at a larger granularity than an FPGA block, yet do not resort to using a generic micro-processor pipeline, which is wasteful in many applications.

As a final thought, I am also very excited about utilizing my background in physics, scientific computing, and system and processor architectures to form collaborations with other faculty and industry, in order to explore and develop architectures and system-methodologies for computing beyond silicon.

References

[1] Stephan Jourdan, Lihu Rappoport, Yoav Almog, Mattan Erez, Adi Yoaz, and Ronny Ronen. eXtendedBlock Cache. In *Proceedings of the Sixth International Symposium on High-Performance Computer Architecture (HPCA-6)*, Toulouse, France, January 2000.

[2] Adi Yoaz, Mattan Erez, Ronny Ronen, and Stephan Jourdan. Speculation Techniques for Improving Load Related Instruction Scheduling. In *Proceedings of the 26th International Symposium on Computer Architecture (ISCA-26)*, Atlanta, Georgia, USA, May 1999.

[3] Adi Yoaz, Mattan Erez, and Ronny Ronen. US Patent #6,697,932: System and Method for Early Resolution of Low Confidence Branches and Safe Data Cache Accesses, February 2004.

[4] Adi Yoaz, Gregory Pribush, Freddy Gabbay, Mattan Erez, and Ronny Ronen. US Patent #6,757,816: Fast Branch Misprediction Recovery Method and System, June 2004.

[5] Adi Yoaz, Ronny Ronen, Lihu Rappoport, Mattan Erez, Stephan Jourdan, and Robert Valentine. US Patent #6,694,421: Cache Memory Bank Access Prediction, February 2004.

[6] Mattan Erez, Brian Towles, and William J. Dally. Spills, Fills, and Kills - An Architecture for Reducing Register-Memory Traffic. Technical report, Stanford University, 2000.

[7] William J. Dally, Patrick Hanrahan, Mattan Erez, Timothy J. Knight, Francois Labonte, Jung-Ho Ahn, Nuwan Jayasena, Ujval J. Kapasi, Abhishek Das, Jayanth Gummaraju, and Ian Buck. Merrimac: Supercomputing with Streams. In *Proceedings of the 2003 International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'03)*, November 2003.

[8] Jung Ho Ahn, Mattan Erez, and William J. Dally. Scatter-Add in Data Parallel Architectures. In *Proceedings of the Eleventh International Symposium on High-Performance Computer Architecture (HPCA-11)*, February 2005.

[9] Nuwan Jayasena, Mattan Erez, Jung Ho Ahn, and William J. Dally. Stream Register Files with Indexed Access. In *Proceedings of the Tenth International Symposium on High-Performance Computer Architecture (HPCA-10)*, Madrid, Spain, February 2004.

[10] Mattan Erez, Jung Ho Ahn, Ankit Garg, William J. Dally, and Eric Darve. Analysis and Performance Results of a Molecular Modeling Application on Merrimac. In *Proceeding of the 2004 International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'04)*, Pittsburgh, Pennsylvaniva, November 2004.

[11] Mattan Erez, Nuwan Jayasena, Timothy J. Knight, and William J. Dally. Fault Tolerance Techniques for the Merrimac Streaming Supercomputer. In *Proceeding of the 2005 International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'05)*, Seattle, Washington, USA, November 2005.

[12] Kayvon Fatahalian, Timothy J. Knight, Mike Houston, Ji Young Park, Mattan Erez, Alex Aiken, Pat Hanrahan, and William J. Dally. Programming the Memory Hierarchy. In *preparation for the 2006 International Conference on Programming Languages Design and Inplementation (PLDI-06)*, Ontario, Canada, 2006.