Hardware/software codesign for energy-efficient parallel computing

B. Goska[1], J. Postman[1], M. Erez[2], and P. Chiang[1]

[1] Kelley Engineering Center, Oregon State University, Corvallis, OR 97331-5501, USA
[2] Engineering Science Building, University of Texas, Austin, TX 78712-0240, USA

E-mail: pchiang@eecs.oregonstate.edu

Abstract.

Increasing complexity and integration of multicore processors have enabled unprecedented increases in parallel processing throughput. Unfortunately, the power consumed by these massively parallel integrated systems will likely limit the achieved system performance. Many mechanisms have been proposed to minimize the power consumption of this underlying microprocessor hardware (dynamic-voltage scaling; frequency throttling; power/clock gating; I/O bandwidth scaling; sub/near-threshold operation), but these mechanisms all require sacrifices in throughput in order to improve performance/watt. Without system-level cooptimization of the algorithms, compiler, and hardware, the opportunity to improve performance/power will be limited.

In this paper, we describe a holistic software/hardware codesign approach that utilizes information feedback from all levels of the computing system, including the application, operating system, compiler, architecture, and device level to find an energy optimal operating point. We also describe a feedback situation that enables applications and computation systems to adapt to different computational load conditions. Through adaptation of both the software and hardware together, with real-time feedback of both power consumed and throughput achieved, our proposed system will achieve optimal power consumed while minimizing the performance impact, resulting in significantly improved energy efficiency.

1. Introduction

Increasingly high levels of system integration have provided significant opportunities for the scientific computing community. However, while continued technology scaling has improved the speed, energy, and density required to do the same computational work, new limitations have emerged. Thermal heating and power consumption problems have been a primary motivation for the migration away from increasing the sequential performance (through higher clock frequencies and more complex logic) toward multicore systems built around parallel processing cores (using lower frequency, simple processing elements).

Unfortunately, addressing the thermal and power density issues by using slower, energy-efficient many-core processors has introduced new challenges, including increased communications bandwidth demands between both on-chip computing elements and off-chip processor nodes and a plateau in the available sequential performance, limiting many algorithms that cannot be easily parallelized.

Highly tunable, reconfigurable processors will be able to meet the opposing goals of both increased performance and lower power, by intelligently controlling operating characteristics across the entire processor using multiple control knobs. For example, such a system might

expose both spatial and temporal fine-grained control of frequency and voltage scaling of arithmetic units, on-chip networks, and memory bandwidth, as well as power and clock gating mechanisms of on-chip memories and individual pipeline stages.

In this document we lay the groundwork for this hardware/software codesigned, hybrid-adaptive parallel processor by identifying the primary mechanisms, controls, and sensor feedback required for this reconfigurable system to optimize its power/throughput metrics. The goal of our proposed research is to determine the best operating parameters for a particular power/performance goal and then adapt the hardware system, either through auto-tuning or real-time subsampling, to the optimal energy/performance point by controlling parallelism, supply voltages, and clock frequencies to best meet target performance, energy, power, or other weighting function.

The remainder of this paper is organized as follows. Section 2 describes the challenges facing conventional systems, and reviews previous work in software/hardware feedback and energy optimization. Sections 3 and 4 propose optimization mechanisms for intelligent parameter tuning, thereby balancing energy and performance at the software and hardware levels, respectively. Section 5 concludes with a general overview of our proposed work that will enable future intelligent, energy-performance balanced, tunable systems that take advantage of software/hardware interactions for improved energy/performance.

2. Background

While the minimization of power consumed in high-performance computing is a well-explored topic, most previous approaches have largely focused on either hardware or software improvements separately, without considering the interactions between the two. An approach that combines both hardware and software optimizations to reduce power consumption presents many clear advantages [1].

2.1. Software-Based Approaches

Energy-efficient software design can be segmented into two main categories: code and runtime optimizations. Code techniques include hand optimization and compiler improvements, while runtime optimizations involve the operating system or are scheduling based.

Previous work on energy-efficient software includes having the compiler optimize code in order to minimize energy. This optimization requires explicit knowledge about the underlying hardware, such that generalized scheduling at the compiler level for power is difficult. Attempts, such as [2], have been made to produce energy profiles from code sequences for specific architectures, such as the StrongARM processor. Unfortunately, these attempts have focused largely on embedded applications where power minimization has been addressed for many years. One such technique is sleep scheduling, where the processes performs all required computations, then sleeps until more computation needs to be done. This technique has been shown to work well on simple processors, like 8-bit or 16-bit microcontrollers, but may not be useful in multithreaded, cache-based multiprocessors. However, some of these energy optimization techniques developed for embedded systems may still have some relevance to high-performance computing, as shown in [3].

Run-time energy optimizations span from small embedded systems [4] to large cloud-computing scheduling algorithms [5, 6]. Operating system (OS) optimizations have been as drastic as pushing the entire OS into hardware. In [7] a TCP/IP stack was optimized by building thread queues into the hardware. This approach was useful because most of the CPU time in the TCP/IP stack was spent handling threads and thread synchronization; it resulted in a large power savings of nearly 85%.

2.2. Hardware-Based Approaches

Saving energy through hardware techniques, both circuit and architectural, is common and has been the focus of much VLSI research in the past decade. Typically, this research is in the form of dynamic voltage and frequency scaling [8], and power/clock gating [9]. Other techniques include the use of accelerators for common tasks (such as AES encryption [10]) or reconfigurable hardware (such as memory hierarchy [11]). Voltage scaling has been shown to achieve 10-20x improvements in energy/computation but comes at a decreased throughput of 100–1000x [12]. Frequency scaling alone typically provides linear reductions in power, but also linear reductions in throughput, thereby exhibiting constant energy-efficiency. Power gating, the disabling of functional blocks within a CPU using a transistor switch as a header, can help minimize static leakage power consumption, but the control overhead power as well as latency can be large, thereby relegating this technique to only coarse-grained code segments.

Beyond the increase of computational units on a single die, both memory capacity and bandwidth have been increasing significantly [13] in order to keep cores active, and therefore have been a large component of power. In [1] we see that the memory (including disks) contributes more that 60% of the total system power consumption. As a result, a significant amount of power can be wasted if the full memory bandwidth is not required.

3. Exposing energy control to software

Energy-efficient software has been approached from a number of different directions, ranging from compile-time optimization to dynamic scheduling of control. These different techniques exhibit varying degrees of success based on the type of applications, and how these applications are mapped to the underlying parallel hardware.

What is most important to note is that different applications exhibit different demands on the computing system as a whole. Applications require different utilizations of various components, such that control parameters of units can be adjusted individually, such as computation throughput (supply voltage; clock frequency), memory capacity (number of powered-on memory chips sharing a single DDR bus), memory bandwidth (number of activated memory lanes; memory bus frequency), on-chip interconnect network throughput (on-chip router buffering), memory access rate, or types of computations performed (integer versus double-precision floating point).

All these hardware parameters combine to form the load on the system, resulting in the total power consumed. Previously, we have seen in [8] that memory access rate can have a large impact on energy consumption; by taking into account memory access rate we can get energy savings of 50% to 65% from DVFS compared with 19% to 53% without this information. Different memory access rates can be found in many standard scientific computations. For example, a sparse matrix multiplied by a vector is a common operation in scientific computing, and is typically more memory bandwidth limited than computation limited. This insight provides the opportunity for energy optimization by increasing the memory bandwidth relative to the compute bandwidth. Contrast this situation with another common operation, the general matrix multiply, which is typically more computation limited than memory bandwidth limited. This insight allows the system to be optimized for increased computational throughput while decreasing the memory bandwidth. These two contrasting situations show that for different application loads due to algorithm differences, computation systems can adapt and reconfigure in order to maintain energy optimal operation.

Because the application can determine the effectiveness of the applied power-saving techniques versus the degradation in execution time, providing energy and performance monitoring information back to the software algorithm is vital. This approach has been previously observed for scheduling techniques, such as [5]. In addition, quality-of-service approaches have been attempted [14]. These approaches have exhibited moderate success in

improving energy consumption but could be significantly enhanced from access to finer-grained controls and more accurate modeling of specific hardware component power dissipation values.

4. Circuit tunability for energy/performance-scalable operation

As CMOS technology scaling has continued to increase transistor density, new challenges have emerged in the design of logic circuits, on-chip interconnect, off-chip interconnect, and memory hierarchy, all of which are further complicated by increased device variability in shrinking CMOS process technologies and reduced supply voltages. The limitations of logic were the first to affect the traditional progression of processor development when transistor density caused CMOS designs to hit a thermal density barrier. Energy efficiency can be improved significantly by scaling the supply voltage (nominally around 1V) to sub/near-threshold levels (0.5–0.6 V), at the expense of a significant exponential increase in delay. This energy/throughput scalability exists not only for on-chip computation, but also for on-chip/off-chip interconnect [15]. While projections from the DARPA exascale computing study [16] concluded that ultra-dynamic voltage scaling may be infeasible due to the overhead of the increased communications demands, recent progress in our VLSI group at Oregon State has begun to address and minimize many of these overheads.

The most common solution to the thermal density issues has been to increase computational parallelism and decrease supply voltages and clock frequencies. This increased parallelism, in turn, exhibits larger demands on communications, both within and between these on-chip cores and processor sockets. This is especially true in the high-performance computing environment, where multicore processors require both large on-chip as well as off-chip memory bandwidth [13]. These individual, multicore systems must communicate with each other via high-speed, off-chip interconnect between cores, chips, boards, racks, and rooms. The result is that the energy required for communication, either on-chip or off-chip, can outweigh the energy required for the logic to actually perform a computation [16].

Recent work in our VLSI group at Oregon State has demonstrated that for both on-chip and off-chip interconnect circuits, new signaling techniques can provide scalable performance and energy-efficiency across a wide dynamic range, from subthreshold and superthreshold supply voltages. For example, we demonstrate on-chip signaling techniques operating at gigahertz speeds at nominal 1.0V supply voltage that achieve greater than 6x energy reduction versus standard digital signaling without sacrificing performance. By then decreasing to tens of megahertz at the energy optimal operating point of 0.35 V energy efficiency can be improved by an additional factor of greater than 5x.

One of the key features for these new interconnect circuit techniques is the ability to digitally calibrate their operating characteristics for the target supply voltage and operating frequency, in order to maximize performance while minimizing the energy required per bit transmitted. Unfortunately, while we currently have the ability to digitally tune the performance and energy characteristics of the circuits, no control information is intrinsically available at the circuit level to select the proper operating point. Hence, our next research goal is to expose these controls and operating characteristics so that control decisions can be made intelligently to choose an optimally energy-efficient voltage and frequency combination based on information available to the architecture, application and programmer levels.

5. Future Work

We have shown that software can benefit from dynamic performance tuning, while the hardware is capable of trading energy for delay. Future work will include how to best expose these energy trade-offs from the hardware to the software. If performed correctly, such as autotuning using a small code subblock, we hypothesize running any large algorithm close to its energy-optimal conditions, approaching the energy per computation limits. One of the biggest challenges to

this vision is determining the energy optimal point for this complex multi-core system. This could be attempted by the compiler in an open loop fashion, or control provided through a real time feedback loop. Using feedback loops for energy consumption control has been previous explored sparsely. One common method is thermal throttling of CPUs but is coarse grained and simply a preservation method. Other attempts have been made at the application level through the use of APIs, such as application heartbeats [14].

Many open questions remain regarding this feedback mechanism. For example, what methods of feedback are optimal with regards to energy, as well as the associated hardware costs? How should we control the computation hardware and its level of control granularity? Where and how should the feedback be applied? These problems will not be solved on any single level of the design hierarchy, but rather requires a complete, holistic, approach to device, architecture, system, application, and operating system design and development.

Acknowledgments

This work was funded in part by NSF-CCF0811820, a NSF doctoral fellowship, a Department of Energy Early CAREER award, and gift donations from Intel Corporation.

References

[1] A. Sivasubramaniam, M. Kandemir, N. Vijaykrishnan, and M. Irwin, "Designing energy-efficient software," *Proceedings of the 16th International Parallel and Distributed Processing Symposium*, pp. 176–183, 2002.

[2] T. Simunic, L. Benini, and G. De Micheli, "Energy-efficient design of battery-powered embedded systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 1, pp. 15–28, 2001.

[3] D. Jensen and A. Rodrigues, "Embedded systems and exascale computing," *Computing in Science & Engineering*, vol. 12, no. 6, pp. 20–29, 2010.

[4] D. P. Volpato, A. K. Mendonca, L. C. dos Santos, and J. L. Güntzel, "A post-compiling approach that exploits code granularity in scratchpads to improve energy efficiency," *2010 IEEE Computer Society Annual Symposium on VLSI*, pp. 127–132, Jul. 2010.

[5] L. M. Zhang, K. Li, and Y.-Q. Zhang, "Green task scheduling algorithms with speeds optimization on heterogeneous cloud servers," *2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, pp. 76–80, Dec. 2010.

[6] Q. Tang, S. S. Gupta, D. Stanzione, and P. Cayton, "Thermal-aware task scheduling to minimize energy usage of blade server based datacenters," *2006 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pp. 195–202, 2006.

[7] N. Maruyama and T. Ishihara, "An RTOS in hardware for energy efficient software-based TCP/IP processing," *Application Specific Processors*, pp. 58–63, 2010.

[8] W.-Y. Liang, S.-C. Chen, Y.-L. Chang, and J.-P. Fang, "Memory-aware dynamic voltage and frequency prediction for portable devices," *2008 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 229–236, Aug. 2008.

[9] S. Hong and H. Kim, "An integrated GPU power and performance model," *Proceedings of the 37th annual International Symposium on Computer Architecture - ISCA '10*, p. 280, 2010.

[10] A. Elbirt, "Fast and efficient implementation of AES via instruction set extensions," *21st International Conference on Advanced Information Networking and Applications Workshops*, vol. 1, 2007. [Online]. Available: http://www.computer.org/portal/web/csdl/doi/10.1109/AINAW.2007.182

[11] Z. Ge, H. Lim, and W. Wong, "A reconfigurable instruction memory hierarchy for embedded systems," in *International Conference on Field Programmable Logic and Applications*. IEEE, 2005, pp. 7–12. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1515691

[12] S. Hanson *et al.*, "Ultralow-voltage, minimum-energy CMOS," *IBM Journal of Research and Development*, vol. 50, no. 4.5, pp. 469 –490, july 2006.

[13] F. O'Mahony *et al.*, "The future of electrical I/O for microprocessors," in *International Symposium on VLSI Design, Automation and Test*. IEEE, 2009, pp. 31–34. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5158087

[14] H. Hoffmann *et al.*, "Application heartbeats for software performance and health," *Proceedings of the 15th ACM SIGPLAN symposium on Principles and practice of parallel programming - PPoPP '10*, p. 347, 2010.

[15] G. Balamurugan *et al.*, "A scalable 5–15 Gbps, 14–75 mW low-power I/O transceiver in 65 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 1010 –1019, april 2008.

[16] P. Kogge *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," 2008.