Preface

Over the past few years, research in power-efficient computing has gained momentum globally, widening the scope of technical advances in hardware and software. Technology trends, environmental concerns, and economic factors have enhanced this momentum. Power is now considered to be a formidable obstacle that challenges the status quo in system design.

Doubling the number of transistors on a chip every 18 months (Moore's law) and the corresponding increase in performance may continue for the remainder of the decade. However, there is a growing concern that manufacturing of denser computing chips could be limited by our ability to cool them.

There is also a growing concern about the environmental impact of the energy required to operate and cool computing devices and its potential contribution to global warming. A 1994 study showed that computers consumed up to 10 percent of the electricity budget in North America.

Additionally, there is an economic incentive to reduce the energy required to operate and cool computing systems. This is fueled by the recent trends of server consolidation and outsourcing computing operations to data centers. Some experts estimate that typical data centers attribute up to 25 percent of their operating costs to power consumption and cooling.

This special issue presents 12 technical papers from leading researchers in power-efficient computing from within and outside IBM.

Technology trends

Over the past 25 years, advances in lithography and device manufacturing have contributed to phenomenal growth in the density of computing devices. The quest for higher performance is the main driving factor in computing system design. With increasing device density, however, come related increases in heat densities that are challenging existing cooling technology. These trends have motivated research in hardware–software co-design to solve or mitigate the problem.

Power reduction in today's microprocessors is a global priority. Fundamentally, the power consumed by a device built from switched capacitive elements (e.g., CMOS technologies) can be separated into *dynamic* and *static* components.

The dynamic component can be adjusted by controlling the frequency, the voltage, or the actual parameters of the circuits. It is reduced linearly by reducing the frequency of the processor and hence its performance. However, if the problem involves intensive computing and the processor is not spending a significant fraction of its cycles waiting for memory or switching between threads, there is no change in the work performed. As a result, the energy required for an operation is not reduced. Moreover, there may be a net loss, because the static power component increases owing to an increased running time. On the other hand, for threads that spend a substantial fraction of their time waiting for memory, reducing the frequency can lead to energy savings for each instruction. This is possible because less energy is spent in idle cycles and for cycles needed to switch threads on long-latency memory operations. Since fewer threads compete for the resources (caches), performance and energy efficiency improve.

The second mechanism for saving dynamic power is to reduce the complexity of the circuits in the processor. The return in power efficiency varies greatly for each simplified function. However, for a substantial number of functions, area is proportional to frequency (i.e., the area–delay product is constant). When the switching factor is unaffected, the switched capacitance is proportional to the area of the processor. Therefore, this mechanism is likely to increase the power–efficiency.

The third, and potentially most efficient, way to reduce power is to reduce the operating voltage. As we scale the supply voltage down to about three times the threshold voltage of the transistors, the circuits slow down linearly with the voltage supplied. Hence, the maximum operating frequency must be reduced in equal proportion to the supply voltage. Because power is dependent on supply voltage raised to a factor n greater than 1, the power is reduced at a faster rate than the drop in frequency. For the typical case in which $n = 2$, the power per operation is reduced with the square of the supply voltage. This is an effective mechanism for increasing power efficiency.

The aforementioned techniques are applicable only to dynamic power. Traditionally, static power has been a small component of the power consumed and has received relatively little attention. The best-known technique for dealing with static power is to turn off the device when it is not needed. However, continued scaling to shrink devices and increase device densities will cause leakage currents to increase. It is predicted that the static power component will dominate the power consumed by CMOS devices in the near future.

Software issues

Software plays a key role in power management. The operating system typically provides mechanisms for exploiting the underlying hardware support for power management. It also receives information about the application mix, either through a program interface or direct user input. This information is useful in controlling the underlying hardware to yield the best performance within a given power budget or the minimum power consumption for a given performance requirement.

Industry recognizes the need for standardizing the interface to power management so that system software

and application can communicate information about workloads and control the operation of the hardware. Today, stock hardware exports several standard functions to inquire about the power consumed and to exercise control over the machine resources. This has created several problems as well as opportunities for software to play an active role in power management. Processor scheduling, for example, must now balance the often conflicting requirements of application performance and power management. For desktop applications, the situation is mitigated somewhat by the fairly moderate performance requirements of interactive applications. The problem is more acute for real-time systems, high-end computers, and servers, because performance and timeliness in generating results are important goals.

Considering other system components, turning off the bus interface of a memory chip can reduce consumed power by a large percentage (more than 95 percent). The memory chip does not lose its content because it operates in an energy-efficient self-refresh mode. This requires some complexities in the memory-management layer of the operating system, a situation in which some memory banks may be switched off to self-refresh mode. So far, studies have focused on conserving the energy consumed by memory systems.

Moving up the storage hierarchy, changing speeds (spin-up/down) is now a standard staple of computing systems, especially portable ones. Unlike memory chips, power management of the disk has been extremely effective, especially for mobile computers and handheld devices. These use special 1.8- and 2.5-inch disks that are optimized for frequent changes in spin speeds. Server systems, however, have been slow to embrace power management for disks. In fact, early attempts to use power management for disks did not work well because server disks tend to be optimized for continuous operation. Such disks are affected negatively by frequent increases in speed. Recently some disk manufacturers have begun to produce disks optimized for both modes of operation. File system software plays an important role in ensuring good performance with minimum disk power consumption and wear.

This issue

The papers in this issue describe hardware and software approaches to power management in computing systems. In the area of circuits and technology, the issue includes seven papers that involve different areas of processor and microarchitecture design. Nagakome et al. review current issues regarding the design of low-power SRAM and DRAM and investigate techniques for reducing the threshold voltage for such devices without compromising reliability. The paper by Mann et al. follows with a description of an ultralow standby power technology for 0.18-μm- and 0.13-μm-lithography nodes for embedded and standalone SRAM applications, and it reviews the critical leakage components and paths within a six-transistor SRAM cell. In the next paper, Oklobdzija presents a systematic design of a clocked storage element suitable for "time borrowing" and absorption of clock uncertainties. The issues related to power consumption and low-power clock circuit designs are reviewed, and results among representative designs are compared. Zyuban and Strenski relate a metric they call "hardware intensity," which is useful for evaluating issues that affect both circuits and architecture, to an architectural energy efficiency metric. Expressions are derived for evaluating the effect of modifications of processor frequency and power at the microarchitectural level, assuming optimal tuning of a pipeline. In the next paper, Kim et al. examine a new technique for a pipeline design that adapts to the data rate. When the processing rate is slow, several stages of the pipelines are turned off and bypassed, and vice versa.

The use of aggressively scaled silicon-on-insulator (SOI) CMOS to successfully build low-power systems-on-a-chip (SoCs) is next examined by Plouchart et al. Finally, Nowka et al. provide a detailed description of the PowerPC® 405LP implementation, which was successful in producing impressive savings in energy without compromising performance.

The hardware papers are followed by two papers on power-estimation tools. In the first, Shafi et al. describe enhancements to the Mambo simulator to estimate the energy consumed by a PowerPC processor within a 5-percent accuracy. In the second, Brooks et al. describe PowerTimer, a tool for estimating energy at the microarchitecture level. The two papers provide a contrast in the goals they address. The first focuses on the macroarchitecture of the processor, and thus is ideal for system-level tuning. The second, in contrast, focuses more on the microarchitecture, and thus is useful for processor or chip microarchitecture design.

The following section contains three papers that cover system-level algorithms and software for power management. The first paper, by Felter et al., describes a performance study of the Super Dense Server project, which uses low-power processors to build server blades. This project provides an example of how modern servers can be constructed to satisfy the need to reduce power dissipation. The paper shows that opting for low-power processors does not necessarily entail a performance penalty, and in fact performance actually improves for some applications. The second paper, by Rusu et al., describes theoretical scheduling algorithms for real-time systems, in which a study is made of the tradeoff of work completed by the deadline of a task and the amount of energy consumed. The paper shows how the software can

trade the quality of the work performed for lower energy, and vice versa. The last paper, by Bradley et al., describes and evaluates algorithms for system-level power management and unmet demand in parallel computer systems using data obtained from production servers for various applications.

Acknowledgment

We would like to thank Lisa T. Su and David L. Cohn for their contributions to the initial phase of this special issue. We also would like to thank the referees for their help in evaluating the manuscripts.

E. N. (Mootaz) Elnozahy

Rajiv V. Joshi

Guest Editors