Power-aware Computing Systems

Dagstuhl Seminar 05141
April 3rd to April 8th 2005

Luca Benini[1], Uli Kremer[2], Christian W. Probst[3], and Peter Schelkens[4]

[1] Università di Bologna, DEIS
Viale Risorgimento 2, 40136 Bologna, Italy
lbenini@deis.unibo.it
[2] Rutgers University, Dept.of Computer Science
96 Frelinghuysen Road, NJ 08854 Piscataway, USA
uli@cs.rutgers.edu
[3] Technical University of Denmark, Informatics and Mathematical Modelling
Richard Petersens Plads, 2800 Kongens Lyngby, Denmark
probst@imm.dtu.dk
[4] Vrije Universiteit Brussel, Dept. of Electronics and Information Processing (ETRO)
Pleinlaan 2, 1050 Brussel, Belgium
Peter.Schelkens@vub.ac.be

Abstract. This paper summarizes the objectives and structure of a seminar with the same title, held from April 3rd to April 8th 2005 at Schloss Dagstuhl, Germany.

1 Introduction

Rapidly increasing chip densities and processor speeds have made energy dissipation a leading concern in computer design. The problem raised by energy consumption is especially severe for a whole class of computing devices which has recently become almost ubiquitously available—mobile devices like notebooks, PDAs, or mobile phones. On the one hand, these are only equipped with a very limited power supply, so any computation on such a device should be especially careful about resource usage. Even worse, the battery technology for these devices has not kept pace with advances in processor technology and the growing complexity of software. On the other hand, cooling mechanisms become more and more important. Recent trends suggest that processor power consumption doubles every four years—and cooling costs rise exponentially with heat increases [1]. The future processors will require energy management solutions more cost effective than the cooling fans used today.

It has been the goal of the seminar to bring together researchers from the main communities working on reducing power consumption, namely hardware, operating systems, virtual-execution environments, compilers, and applications. The main seminar result, given in Section 3, is a classification of the obstacles,

Dagstuhl Seminar Proceedings 05141
Power-aware Computing Systems
http://drops.dagstuhl.de/opus/volltexte/2005/321

and therefore research directions, with respect to power consumption seen for different classes of devices, ranging from very low power devices, over handheld devices, to servers and work stations. In a next step the seminar identified the impact different levels of dealing with power concerns can have.

2 Approaches to Reduce Energy Consumption

Figure 1 gives an overview of the different levels that energy saving techniques can be applied at and some of the techniques available. It is noteworthy that leakage is rapidly becoming the dominant source of power consumption.

On the hardware level, decisions influence how software will be able to affect power dissipation. Similarly, decisions on the software level, that is operating system, compiler, virtual-execution environment, or application level, influence the processor's workload. Developing an integrated approach remains an open problem, although a number of research groups have started to explore it. However, currently most research is done in a vacuum, making assumptions about the execution environment that do not take into account the effect of other techniques.

Low-power computing is challenging first of all due to the sheer explosion of total hardware/software decisions [2]. The problem is further complicated by decisions along any dimension having tradeoffs along other dimensions. One tradeoff is between power and performance. Another is that minimizing power might introduce sharp power variations that impair chip reliability.

The third reason low power design is nontrivial is that no decision exists in a vacuum. A compiler can emit code to slow down the processor during memory stalls, but an operating system can override the compiler's decision by speeding up the processor at the same time. Without synergy between the compiler

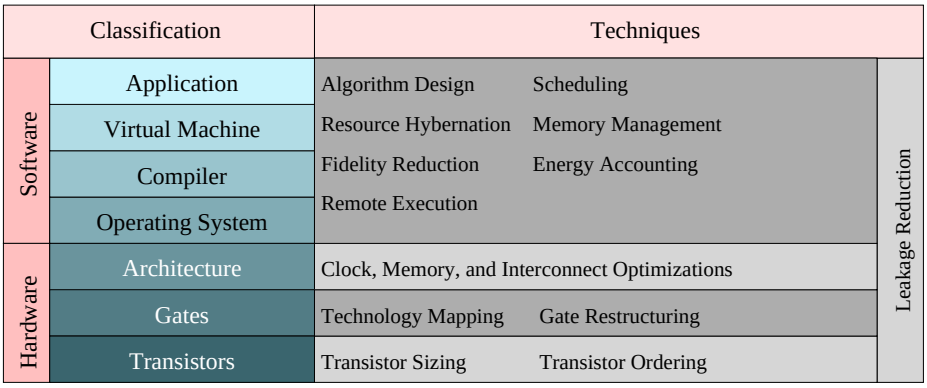

Fig. 1. An overview of techniques to achieve energy savings at different levels. The left hand part lists levels of hard- on software, the right hand part exemplary techniques that are applicable at that level.

and operating system, frequency thrashing can occur, resulting in power and performance degradation. More generally, without synergy among decisions at different levels of design, the effect of any power reduction technique will be short lived.

For all of these reasons, the problem of developing an integrated approach to power management remains unsolved, though researchers are addressing power issues at every level of design.

3 Where does the Power go?

In a series of discussion sessions the seminar participants tried to identify where power is consumed in devices. This resulted in a matrix (Figure 2) that ranks how much different areas (Computation, Communication and I/O, Storage, Other) of different classes of systems (Very Low Power, Systems on a Chip, General Purpose Computing) contribute to the total system power consumption.

This matrix was then used to identify the impact that we expect different levels to have, as well as techniques provided by and problems to be solved in each of the levels. The levels considered are

- Logic, Circuit, and Technology,
- Architectures and Micro-Architectures,

		Storage	Communication I/O	Computation	Other
General Purpose Computing	Server	2 DRAM/DISK	3	1	3 Power Supply
	Work Station	2	1 WLAN/LCD	1 GPU/CPU	3
SoC for Digital Convergence	Receive	2	1	1	-
	Duplex	3	1 RF/LCD	2 CPU	-
	no Communication	1	1	2 specialized HW	-
Very Low Power		2 may be 1 for non-RF hardware	1	3	-

Fig. 2. Contribution of different areas to the power consumption of classes of devices, ranging from very low power devices like sensors, over hand-held devices, to work stations and servers. The areas are sorted according to their importance from 1 (most important) to 3 (least important).

- Compilers, Virtual-Execution Environments, Operating Systems, and Middleware, and
- Applications and Algorithms.

The following sections give an overview of the results identified in working groups for the respective levels.

3.1 Logic, Circuit, and Technology

On the hardware level leakage power has been identified as the major problem. In scaling for performance, the main goal is to reduce both supply voltage as well as gate threshold voltage to reduce dynamic power, while keeping enough drive current. Unfortunately, that creates an exponential rise in leakage power that even can exceed dynamic power—that is, the energy used without the system doing anything is higher than the energy used for the actual operation.

One solution would be to increase the gate threshold voltage, which results in lower leakage power, but also shrinks the acceptable range of supply voltage. To compensate for this, one would need ever thinner gate oxides—which results in increased gate leakage.

So using current technologies, scaling as usual will not do the trick. At the same time new device architectures are needed. However, since Moore's law still seems to be valid, choices will have to be made rather quickly, identifying the right technology to create cost effective yielding processes that will be usable in giga-scale architectures.

The ITRS roadmap [3] requests that the increase in overall power consumption should be almost zero in this decade, while dynamic power alone currently increases 1.4 times per 3 years. If the leakage component is added to these numbers, the increase becomes much steeper. Thus the power increase might become a big stumbling block to Moores law. At the same time, frequency, supply voltage, and threshold-voltage selection heavily depend on the workload and can thus require interaction with the system level. Hardware, however, severely limits the degrees of freedom available, and beyond 65nm technology both leaking as well as gate delay get out of hand, unless the size of devices is increased, counteracting the idea of scaling. Clearly a fundamental change in CMOS device architecture is needed to keep up the pace of Moore's law.

From the perspective of the logic, circuit, and technology level it is questionable whether more parallelism is going to reduce the overall energy consumption. While extra parallelism allows to reduce the supply voltage for the same performance and in this way to reduce power/energy, it also introduces extra transistors (more leakage) and extra and longer wires (more capacitive coupling, noise issues). On top of these issues a lower supply voltage also stresses the variability issues. To be able to resolve these problems we need to better understand the underlying physical phenomena. This will require more interaction between the system/architecture, circuits, and technology communities.

3.2 Architectures and Micro-Architectures

The common trend in (micro-) architecture design is to have numerous small, potentially heterogeneous/special-purpose cores. These architectures are dominated by communication across cores. In the general-purpose domain the main problem is resource contention that requires intelligent coordination to be resolved. In the application-specific domain, architectures can be optimized with respect to power consumption, e.g. by using a simplified voltage-gating approach.

In order to allow an energy-efficient usage of a given architecture, it should expose any non-uniformity to allow its exploitation by the software system running on top. Other needs include support for high-bandwidth communication between devices, more explicit support for concurrency, as well as mechanisms to exploit locality. Additionally there is an urgent need to develop APIs and instruction set architectures that allow systems to express and control variability in the architecture and application. This would require a holistic approach that encompasses I/O, storage, and compute resources.

On the other hand, there are topics that can be better dealt with by other layers. The level of logic, circuits, and technology should support an energy-aware design and provide power-management knobs that support not only DVS, threshold-power control, supply-power control, and leakage, but also new, yet undiscovered approaches. The compiler, operating system, virtual-execution environment, and middleware layer should leverage the API provided by the architecture. Finally, at the application layer one would need new programming models to allow the overall system to efficiently use the resources provided by the architecture.

The main open issue on the architecture level is parallelism and how to extract it efficiently. This might require re-architecting the CPU, storage hierarchies, and more. Increases in parallelism might also call for rethinking the hardware support and hardware/software coordination regarding task scheduling, placement, and migration.

3.3 Compilers, Virtual-Execution Environment, Operating Systems and Middleware

The layer of compilers, virtual-execution environments, operating systems, and middleware is especially well suited to predict and determine the current and future behavior of programs and tasks by using just-in-time compilation to reshape program behavior at run time. Components on this level can pass information up and down to lower and higher levels, e.g. to disable memory banks in computation intensive areas of an application.

In general, these components could process implicit and explicit application-level constraints, e.g. related to security, reliability, and real time. However, this is a mechanism that is mostly unsupported by current programming models. To allow systems and programmers to benefit from constraint processing we would need new power-aware programming models.

As a rule of thumb one can say that the higher the overhead of an optimization or analysis, the coarser granularity is needed to make it work, that is to justify the need to enable it. This leads to the question which optimizations are most profitable for the reduction of energy consumption. The working group identified hibernation of resources to reduce leakage, DVS of whole processors and individual cores, and quality of result tradeoffs, where the last one would require semantic models to allow the quantification of tradeoffs. The most profitable optimizations and analyses to enable decisions on how to adjust system parameters are parallelization, phase detection (e.g. flag change in communication requirements), and pattern detection (e.g. producer/consumer). Especially the detection of phases and patterns would allow to adjust the overall system parameters according to the current system role.

The challenge in the area of compilers, virtual-execution environments, operating systems, and middleware is to define and enable interactions across different layers. By systematically designing all layers of complete systems, each layer can be designed such that it can make assumptions on the behavior of other layers and will be able to influence the overall system behavior. The ultimate goal is to develop a whole-system solution that can deal with the variability of the resource requirements and execution constraints of the application as well as of the features and resources of the target system.

3.4 Applications and Algorithms

Generally speaking the optimization potential increases with the abstraction level, making it advantageous to optimize at the system and algorithmic level. There are two possible approaches to enable those optimizations—either by acquiring application knowledge or by developing domain-specific systems and algorithmic-evaluation frameworks.

Evaluation frameworks need to support scenarios that reflect the huge system space in a representative way to reduce the time needed for simulation of the system behavior. The main property required in these frameworks is that they offer domain-specific architectural templates that include and model all system components to allow substantiated simulation results. In addition, each system component needs to be equipped with a power model that is as accurate as possible. Currently these models are often hard to obtain from industry or even unavailable, e.g. for some analogue components whose behavior is hard to describe.

Having these general evaluation frameworks available would allow to identify the critical consumers in a system design. These would then be targeted to reduce their power consumption. To support this process, metrics for the tradeoff are needed. For example in multimedia applications a technique to measure or judge the picture quality per consumed Watt would be needed.

In the area of applications and algorithms this working group identified two major challenges. To enable evaluation frameworks we will need domain-specific power-optimization technologies that cover the system and implementation lev-

els. In addition we need an interdisciplinary engineering approach to co-design the hardware, software, and applications of power-aware computing systems.

4 The Seminar

The program of the seminar featured presentations of about 35 participating researchers from academia and industry. They were chosen to represent major areas in targeting the energy consumption of a computing system—Applications, Compilers, Virtual-execution Environments, Operating Systems, and Hardware.

In order to identify problem areas and future research areas, discussion groups were formed that resulted in four working groups whose results are presented in this report. In addition, abstracts of the presentations as well as work-in-progress papers are published in these proceedings. Some of the work presented at the seminar will be published in a special issue of the International Journal of Embedded Systems on power, energy, and thermal topics.

References

1. Gunther, S.H., Binns, F., Carmean, D.M., Hall, J.C.: Managing the Impact of Increasing Microprocessor Power Consumption. Intel Technology Journal (2001) Q1 issue.
2. Venkatachalam, V., Franz, M.: Power Reduction Techniques for Microprocessor Systems. Accepted for publication in ACM Computing Surveys (2005)
3. ITRS: The International Technology Roadmap for Semiconductors, available at `http://public.itrs.net` (2004)