Power-aware Computing Systems

Dagstuhl Seminar 07041
January 21st to January 26th 2007

Luca Benini[1], Naehyuck Chang[2], Uli Kremer[3], Christian W. Probst[4]

[1] Università di Bologna, DEIS
Viale Risorgimento 2, 40136 Bologna, Italy
lbenini@deis.unibo.it
[2] Seoul National University, School of Computer Science and Engineering
Silim Dong, Kwanak Gu, Seoul, 151-742, Korea
naehyuck@snu.ac.kr
[3] Rutgers University, Dept.of Computer Science
96 Frelinghuysen Road, NJ 08854 Piscataway, USA
uli@cs.rutgers.edu
[4] Technical University of Denmark, Informatics and Mathematical Modelling
Richard Petersens Plads, 2800 Kongens Lyngby, Denmark
probst@imm.dtu.dk

Abstract. This paper summarizes the objectives and structure of a seminar with the same title, held from January 21st to January 26th at Schloss Dagstuhl, Germany. The seminar started from the results of the preceding Dagstuhl seminar 05141 on the same topic, and tried to identify emerging trends in three areas—*low-power design and reliability*, , and *power estimation and simulation*. The outcome of these discussions is also contained in this article.

1 Introduction

The program of the Dagstuhl seminar 07041 on *Power-aware Computing Systems* featured presentations of about 25 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 continue the work of the predecessor Dagstuhl seminar held in 2005, the results of that seminar [1] were discussed, with the aim of developing a vision of challenges, problems, and research activities in some of the key areas identified in 2005. The first part of the seminar was dedicated to lively discussions that led to the identification of three areas that were considered being most interesting. As a result, three groups were formed to further identify challenges and opportunities. The results of these groups are presented in this report. In addition, abstracts of the presentations as well as work-in-progress papers are published in these proceedings.

Dagstuhl Seminar Proceedings 07041
Power-aware Computing Systems
http://drops.dagstuhl.de/opus/volltexte/2007/1123

The remainder of this section summarizes the starting point of the discussion by giving an overview of the results of the seminar in 2005. It is followed by a presentation of the results of the working groups on *low-power design and reliability* (Section 2), (Section 3), and *power estimation and simulation* (Section 4). Finally, Section 5 concludes this report. More information on the seminar described in this report and its predecessor in 2005 can be found in [2,3].

1.1 Where does the Power go?

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.

The predecessor of the current seminar, held in 2005 [2], developed a classification of the obstacles, 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.

These discussions resulted in the matrix shown in Figure 1 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 remainder of this part gives a (very condensed) description of the discussion results. For a more detailed description please refer to [1].

Logic, Circuit, and Technology. On the hardware level, leakage power was identified as the major problem. 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, 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).

Architectures and Micro-Architectures. The common trend in (micro-) architecture design is to have numerous small, potentially heterogeneous/special-purpose cores, dominated by communication across cores. 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. 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. 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.

Compilers, Virtual-Execution Environment, Operating Systems and Middleware. This layer 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. In general, these components could process implicit and explicit application-level constraints. The challenge, however, 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

		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. 1. 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).

the overall system behavior. The ultimate goal is to develop a holistic 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.

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. General evaluation frameworks would allow to identify the critical consumers in a system design, and target them to reduce their power consumption. 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. However, to enable evaluation frameworks we will need domain-specific power-optimization technologies that cover the system and implementation levels. In addition we need an interdisciplinary engineering approach to co-design hardware, software, and applications of power-aware computing systems.

2 Low-power Design and Reliability

This working group started by identifying the most important failure types, with respect to both permanent and temporary failures (Figure 2). All these of course pose significant reliability issues, with the permanent failures probably being potentially more severe, as the hardware itself is damaged over time. An important point here is, that the effect of many of these errors is accelerated at higher temperatures, meaning that any technique that reduces temperature can potentially help. In contrast, transient errors (usually) do not damage the hardware, but still influence the overall functioning of systems. However, the correlation between low-power techniques and transient errors is less clear, meaning that the impact of these techniques on transient errors is more complicated than for permanent failures.

The main result of this working group was that *low-power techniques* and *reliability-aware design* should be studied together, in order to realize synergy that these techniques can have on reliability.

Beside hardware aspects, this working group also discussed reliability-aware design at the software level. Here, some algorithms may tolerate "faulty" data,

Permanent Failures		Temporary Failures
TDDB	Time-dependent Dielectric Breakdown	SEU Single Event Upset
NBTI	Negative Bias Temperature Instability	Power Supply Noise
HCI	Hot Carrier Degradation	Crosstalk
EM	Electro Migration	Substrate Noise

Fig. 2. Permanent and temporary failures

Low Power Technique	Permanent Errors				Transient Errors			
	TDDB	NBTI	HCI	EM	SEU	PN	CT	SN
VDD scaling	++	++	++	++	–	–	–	?
Multiple Vt	++	++	++	++	+/-	N/A	++	?
Clock gating	?	?	++	?	–	?	?	?
Power gating	?	++	?	++	?	?	?	?
MLV	++	?	?	?	?	?	?	?
RBB	?	?	?	?	?	?	?	?
Transistor scaling	?	?	?	?	?	?	?	?

Fig. 3. The impact of low-power techniques on permanent and transient errors. "++" stands for a reduction of an error type by a certain technique, "–" for an increase.

and still produce correct results, although at lower "quality". The discussion here centered around how to develop systems that allow to influence how much quality reduction an end user is willing to tolerate, and how to specify both user requirements and system capabilities. Interestingly, similar issues re-appeared in the two other working groups.

3 Parallelism

The current trend towards multi-core architectures is expected to continue for the foreseeable future, and seems a promising way of overcoming the leakage power bottleneck. The working group on parallelism identified three main challenges. With respect to *chip-level parallelism*, the trade-off between (the reduction of) power consumption and chip reliability is of high importance (and related to the first working group). This is especially the case for chip-parallel systems in nano-scale technology.

The second challenge, identified in the first seminar on this topic, is how to *specify application requirements and system capabilities*, such that software can be best mapped onto the system. The main question with respect to parallel systems is, how much power resource management we are able (or want) to hide from the programmer, especially as programming parallel systems is already difficult. This is especially important for developing scalable *and* power-efficient programs in heterogeneous environments.

Heterogeneous environments also where at the center of the working group's third focus, namely how to map processes with respect to processing and communication onto highly specialized, heterogeneous components. Due to process variation, this mapping problem has both static and dynamic aspects. If applications are able to specify requirements, then systems need to map the application and system resources such that these requirements are fulfilled. The working group also discussed how this assignment could be performed by a virtualization layer.

4 Power Estimation and Simulation

One issue that repeatedly came up in the initial discussion round was predictability of techniques and their evaluation. While many tools exist on various levels, it is unclear how to get a unified simulation and estimation framework that allows for an integrated approach with the possibility to use different granularity for different system parts. At the same time it is close to impossible to integrate all different levels—from gates and circuits to software architectures—in a single tool. Instead, this working group investigated how to design a tool chain by standardizing the quantities and information of relevance to power and performance analysis, that can be extracted from a design (hardware or executable benchmark) by a tool, along with a measure of accuracy of these tool-reported estimates. For example, in addition to, or including its standard parameters, a tool might take as input a standardized tuple of (`ambient_temperature`, `cycle_time`), and might report as its output the tuple ($50E^{-3}$, 1%), corresponding, for example, to a power estimation report (averaged over some time window) of 50 mW and an associated accuracy of that value to be within 1% of hardware. This value could correspond to either, say, the reported leakage power of a temperature-aware leakage estimation tool, which ignores the supplied cycle time parameter, or to the reported average power consumption for some window of time on an instruction-set simulator, with the simulated processor's cycle time as given. These are simple illustrative examples of the potential uses of the interface. Such

Level	Relevant Optimizations	Properties
Transaction Level	Communication optimization	dynamic power, leakage power
Behavioral Level	Architectural optimization	dynamic power, leakage power
Register Transfer Level	Structural optimization	dynamic power, leakage power
Gate Level	Logic optimization	dynamic power, leakage (sub-threshold, gate I_{on}, gate I_{off})
Transistor Level	Transistor sizing	dynamic power, leakage (sub-threshold, gate I_{on}, gate I_{off}, short circuit)
Layout Level	Interaction with lithographic process, and techniques such as OPC and RET; Technology parameters provided by fab	dynamic power, leakage (sub-threshold, gate Ion, gate Ioff, short circuit), losses due to device/ layout structure effects

Fig. 4. Abstraction layers, and examples of the associated possible optimizations and relevant system properties.

an interface, alongside the standardization of configuration parameters that can be provided to a tool (e.g., operating voltage, ambient temperature), enables the composition of tools conforming to the interface, into a system-level framework.

Figure 4 shows an overview over the contra-variant measures of abstraction and accuracy of simulation and estimation aspects, along with examples for properties of interest as well as relevant optimizations at the different levels. Some initial results of this working group can be found in an article in this proceedings.

5 Conclusion

The second Dagstuhl seminar on *Power-aware Computing Systems* picked up the discussion results of its predecessor [1], and continued the discussion of challenges in the area. We think that the results, partly described in this report, partly described in the papers published as part of the seminar proceedings, are suited to give the involved communities ideas for future challenges.

We would like to thank all participants of the seminar for making it a fruitful and inspiring event—and especially Dagstuhl's wonderful staff, for their support both before and during the seminar.

References

1. Benini, L., Kremer, U., Probst, C.W., Schelkens, P., eds.: Power-aware Computing Systems, 3.-8. April 2005. Volume 05141 of Dagstuhl Seminar Proceedings., Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2005)
2. Homepage of Dagstuhl Seminar 05141: "Power-aware Computing Systems". http://www.dagstuhl.de/05141 (2005)
3. Homepage of Dagstuhl Seminar 07041: "Power-aware Computing Systems". http://www.dagstuhl.de/07041 (2007)