CEPIS UPGRADE is the European Journal for the Informatics Professional, published bi-monthly at <http://cepis.org/upgrade>

Publisher
CEPIS UPGRADE is published by CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>), in cooperation with the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática,* <http://www.ati.es/>) and its journal Novática

CEPIS UPGRADE monographs are published jointly with Novática, that publishes them in Spanish (full version printed; summary, abstracts and some articles online)

CEPIS UPGRADE was created in October 2000 by CEPIS and was first published by Novática and INFORMATIK/INFORMATIQUE, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies)

CEPIS UPGRADE is the anchor point for UPENET (UPGRADE European NETwork), the network of CEPIS member societies' publications, that currently includes the following ones:
- inforewiew, magazine from the Serbian CEPIS society JISA
- Informatica, journal from the Slovenian CEPIS society SDI
- Informatik-Spektrum, journal published by Springer Verlag on behalf of the CEPIS societies GI, Germany, and SI, Switzerland
- ITNOW, magazine published by Oxford University Press on behalf of the British CEPIS society BCS
- Mondo Digitale, digital journal from the Italian CEPIS society AICA
- Novática, journal from the Spanish CEPIS society ATI
- OCG Journal, journal from the Austrian CEPIS society OCG
- Pliroforiki, journal from the Cyprus CEPIS society CCS
- Tölvumál, journal from the Icelandic CEPIS society ISIP

Editorial Team
Chief Editor: Llorenç Pagés-Casas
Deputy Chief Editor: Rafael Fernández Calvo
Associate Editor: Fiona Fanning

Editorial Board
Prof. Vasile Baltac, CEPIS President
Prof. Wolffried Stucky, CEPIS Former President
Prof. Nello Scarabottolo, CEPIS President Elect
Luis Fernández-Sanz, ATI (Spain)
Llorenç Pagés-Casas, ATI (Spain)
François Louis Nicolet, SI (Switzerland)
Roberto Carniel, ALSI – Tecnoteca (Italy)

UPENET Advisory Board
Dubravka Dukic (inforeview, Serbia)
Matjaz Gams (Informatica, Slovenia)
Hermann Engesser (Informatik-Spektrum, Germany and Switzerland)
Brian Runciman (ITNOW, United Kingdom)
Franco Filippazzi (Mondo Digitale, Italy)
Llorenç Pagés-Casas (Novática, Spain)
Veith Risak (OCG Journal, Austria)
Panicos Masouras (Pliroforiki, Cyprus)
Thorvardur Kári Ólafsson (Tölvumál, Iceland)
Rafael Fernández Calvo (Coordination)

English Language Editors: Mike Andersson, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennemore, Hilary Green, Roger Harris, Jim Holder, Pat Moody.

Cover page designed by Concha Arias-Pérez
"Luminous Recharge" / © ATI 2011
Layout Design: François Louis Nicolet
Composition: Jorge Llácer-Gil de Ramales

Editorial correspondence: Llorenç Pagés-Casas <pages@ati.es>
Advertising correspondence: <info@cepis.org>

Subscriptions
If you wish to subscribe to CEPIS UPGRADE please send an email to info@cepis.org with 'Subscribe to UPGRADE' as the subject of the email or follow the link 'Subscribe to UPGRADE' at <http://www.cepis.org/upgrade>

Copyright
© Novática 2011 (for the monograph)
© CEPIS 2011 (for the sections Editorial, UPENET and CEPIS News)
All rights reserved under otherwise stated. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, contact the Editorial Team

The opinions expressed by the authors are their exclusive responsibility

ISSN 1684-5285

Monograph of next issue (December 2011)
"Risk Management"

CEPIS UPGRADE
The European Journal for the Informatics Professional
http://cepis.org/upgrade
Vol. XII, issue No. 4, October 2011

Monograph
Green ICT: Trends and Challenges
(published jointly with Novática*)
Guest Editors: *Juan-Carlos López-López, Giovanna Sissa, and Lasse Natvig*

2 Presentation. Green ICT: The Information Society's Commitment for Environmental Sustainability — *Juan-Carlos López-López, Giovanna Sissa, and Lasse Natvig*

6 CEPIS Green ICT Survey – Examining Green ICT Awareness in Organisations: Initial Findings — *Carol-Ann Kogelman on behalf of the CEPIS Green ICT Task Force*

11 The Five Most Neglected Issues in "Green IT" — *Lorenz M. Hilty and Wolfgang Lohmann*

16 Utility Computing: Green Opportunities and Risks — *Giovanna Sissa*

22 Good, Bad, and Beautiful Software – In Search of Green Software Quality Factors — *Juha Taina*

28 Towards the Virtual Power Grid: Large Scale Modeling and Simulation of Power Grids — *Peter Feldmann, Jinjun Xiong, and David Kung*

41 Artificial Intelligence Techniques for Smart Grid Applications — *María-José Santofimia-Romero, Xavier del Toro-García, and Juan-Carlos López-López*

49 Green Computing: Saving Energy by Throttling, Simplicity and Parallelization — *Lasse Natvig and Alexandru C. Iordan*

59 Towards Sustainable Solutions for European Cloud Computing — *Kien Le, Thu D. Nguyen, Íñigo Goiri, Ricardo Bianchini, Jordi Guitart-Fernández, and Jordi Torres-Viñals*

67 A State-of-the-Art on Energy Efficiency in Today's Datacentres: Researcher's Contributions and Practical Approaches — *Marina Zapater-Sancho, Patricia Arroba-García, José-Manuel Moya-Fernández, and Zorana Bankovic*

UPENET (UPGRADE European NETwork)

75 From **Mondo Digitale** (AICA, Italy)
IT for Education
IT in Schools. A European Project for Teachers Training — *Pierfranco Ravotto and Giovanni Fulantelli*

CEPIS NEWS

81 Selected CEPIS News — *Fiona Fanning*

* This monograph will be also published in Spanish (full version printed; summary, abstracts, and some articles online) by **Novática**, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática*) at <http://www.ati.es/novatica/>.

Green Computing: Saving Energy by Throttling, Simplicity and Parallelization

Lasse Natvig and Alexandru C. Iordan

The aim of this article is to give an overview of several techniques that are used in making computers more energy efficient. All segments of the ICT market are facing this challenge, starting with the smallest embedded systems, through mobile devices and laptops, to workstations and supercomputers. We start by giving a broad overview of energy saving techniques in hardware, continue at the operating system level, system software and finally techniques that can be used at the application level. Techniques for low-power electronics such as different levels of sleep modes, architectural techniques like multi-cores and heterogeneous processing, the role of customization and accelerators as well as reconfigurable hardware are outlined. We describe how parallelization in both HW and SW can help reduce the energy consumption, and discuss several examples of how throttling (slowing down) and simplicity give the same effect. The authors have developed an experimental framework for exploring these issues, and the paper ends by presenting a few recent results showing how parallelism expressed with Task Based Programming, TBP, can save energy.

Keywords: Energy Awareness, Energy Efficiency, Green Computing, Multi-Core, Power Management.

1 Introduction

The need for saving energy has become a top priority in almost *all segments of the ICT market.* In sensor network, tiny computers report periodically measurements over a time frame of 10 years or more. In such situations it is impossible or too costly to replace batteries. Most users of mobile phones and laptops have the experience of running out of battery too fast. Ordinary desktop computers could produce less heat, less noise from cooling fans and be cheaper to operate if they were to consume less energy. In High Performance Computing, HPC, the need for power efficiency has become even more important and is now a critical design factor and operational requirement. In almost all cases, reducing the energy consumption leads to a lower performance – a classical and challenging situation of conflicting goals.

This paper gives an introduction to *green computing*, focusing on what can be done in hardware and systems software to save energy when doing computations on computers of all kinds and sizes. This is a relatively narrow focus compared to other papers that include the environmental impact of the whole life cycle of computers and related equipment, such as the manufacturing process and disposal and recycling. These very important aspects, definitely within Green IT or Green ICT, are covered in other papers of this CEPIS **UP**GRADE Oct. 2011 Special Issue on Green ICT.

As indicated in the title, we will explain three general technological principles that have the potential to save energy in computations. *Throttling*, i.e. slowing down a processor or core to a lower frequency is a widespread technique today, but must be balanced against the disadvantage of lost performance. *Simplicity* is perhaps our most important design principle and it has self-evident effects on energy consumption. However the need for product portability and longevity has pushed towards increasingly complex ICT systems with lots of abstraction layers. Green computing gives increased focus to simplicity and a need to re-evaluate the design process that leads to all this complexity. *Parallelism*, i.e. distribution of computation on multiple processors or cores, might not look at first sight like a power saving technique since it generally requires some overhead and most often increased complexity. However, as made evident by the rapid and dominating technological shift to multi-core processors, it has become the main design factor in microprocessor development to save energy.

Green computing is an interdisciplinary, large and rapidly increasing area. In this overview the aim is to introduce the main concepts within the focus of the paper, as well as provide motivation for further studies within the area.

Authors

Lasse Natvig received the MS and Dr. Ing. degrees in Computer Science from the Norwegian Institute of Technology, NTNU, in 1982 and 1991, respectively. He is currently a Professor in Computer Architecture at the Norwegian University of Science and Technology (NTNU), Trondheim, Norway. His main research interests are computer architecture, parallel processing, multi-core systems, system-level computer simulation, memory architectures and green computing. Dr. Natvig is a full member of HiPEAC2 Webpage: <http://www.idi.ntnu.no/people/lasse>. <Lasse@computer.org>

Alexandru C. Iordan received his BSc and MS degrees from the *Politehnica Universitatea* of Bucharest, Romania. He is currently a PhD candidate at the Norwegian University of Science and Technology, focusing on energy efficient methods for multi-core programming. <iordan@idi.ntnu.no>

© Novática

Figure 1: Selected Aspects of Green Computing.

2 Green Computing — A Subset of Green ICT

We will focus on a subset of green ICT, namely those techniques that are used to reduce the energy needed to execute a computer program. The techniques we discuss are outlined in Figure 1. There is also an abundance of software *applications* that are green in the sense that they help the community to save energy. Well-known examples are intelligent control of combustion engines, routing of transportation to minimize travel distance and teleconferencing. Such applications are not covered in this overview.

In Section 2.1 and 2.2, we present a high level view of the main hardware techniques. Our main emphasis is at the architectural level since it is of paramount importance to have a holistic view of the interplay of both HW and SW when designing an energy efficient system. Section 2.3 discusses techniques such as processor throttling and load balancing that are controlled at the operating system level. These are normally invisible to the application programmer. Section 2.4 introduces energy aware algorithms and data-structures. Implicitly, good traditional design principles such as aiming at simplicity have helped produce such algorithms. However, the recent increased interest in green computing has stimulated new research in the area. Also, research into data-structures has gained renewed interest in the multi-core era. This can be explained by the need to make the data-structures more efficient for sharing in multithreaded programs. Also, the large on-chip shared caches of a modern multi-core processor offer new possibilities to implement such data sharing in a more energy efficient way.

Clock frequency (speed): f
Supply voltage: V
Dynamic power consumption: $P_{dynamic}$
Static power consumption: P_{static}
Physical capacitance: C
Activity factor: a
Execution time: T
Performance: $Perf = 1/T$
Energy Delay Product ([20]): EDP

(Eq. 1) $P_{dynamic} \sim aCV^2f$
(Eq. 2) $Power = P_{dynamic} + P_{static}$
(Eq. 3) $Energy = Power \times T$
(Eq. 4) $EDP = Perf^2/Power$

Figure 2: Microprocessor Speed, Power, and Energy — Simplified Definitions and Metrics.

2.1 Low Power Design and HW Energy Saving Techniques

There is a large interest in improving the energy efficiency of a chip's logical gates. These efforts include basic research to find new materials that can be used to do computations and research in electronics as how to operate transistors at lower voltage levels, as in near-threshold and sub-threshold designs. This technology will reduce the possible speed, i.e. increased propagation delay, and therefore give a reduced clock frequency. It is predicted that near threshold computing can reduce energy requirements by 10 to 100 times or even more in future systems [1].

Within digital design we have a rich set of circuit and logic level techniques — extensively surveyed in the paper *Power Reduction Techniques For Multiprocessor Systems* by Venkatachalam and Franz [2]. To mention just a few: making transistor smaller, reordering transistors in a circuit, half-frequency clocks synchronizing events using both edges of the clock, logic gate restructuring, technology mapping where the components are selected from a library to meet energy constraints, and the use of low-power flip-flops. Placed slightly higher in the abstraction levels is *low power control logic design*. An example can be to implement a finite state machine (FSM) so that the switching activity in the processor is minimized, or to decompose the FSM into smaller sub-FSMs that can be deactivated when not in use.

> “The need for saving energy has become a top priority in almost *all* segments of the ICTmarket”

 © Novática

> “The aim of this article is to give an overview of several techniques that are used in making computers more energy efficient”

2.1.1 Dynamic Voltage and Frequency Scaling, DVFS

Another important technique is dynamic voltage scaling and frequency scaling, DVFS. We list it here since it requires special hardware features, but it is normally controlled by software. *Dynamic Voltage Scaling* has also been called *undervolting*, and is used to lower the supply voltage of a processor when the workload is getting so small that the processor can reduce its speed (frequency) and still have a performance that is sufficiently high to meet the systems requirements. Reducing the frequency also makes it possible to reduce the supply voltage (V) since the gates can use longer time to switch. Both result in lower dynamic power consumption (see Figure 2). DVFS is used in most modern laptop processors where it is controlled by the operating system to save energy under light load.

Much of the research on DVFS has focused on single-core processors, and similar techniques have been used to other components such as main memory, RAM, and hard disks [3]. Today, this kind of power management is also available in multi-core processors, also called chip multiprocessors, CMP. A simple example of such a processor is illustrated in Figure 3. Using such a CMP, where each core is operating at $f/4$, the same performance as a single-core processor running at frequency f can be achieved. Also, since the supply voltage V can be reduced when the frequency f is reduced, eq. 1 in Figure 2 tells us that we can get a cubic reduction in power used per core. In general, parallelizing a computation by executing it on many slow cores instead of one fast core makes it possible to do the same amount of work equally fast or faster and still use less energy. This is one of the main reasons for the rapid paradigm shift in the microprocessor market from single- to multi-cores. The underlying assumption is of course that the overhead incurred by the parallelization is not too large.

When using DVFS in a multi-core context the most straight-forward approach is to use one "*knob*" to control the whole chip, i.e. to reduce the frequency of all the cores at the same rate. However, improved energy efficiency can be achieved if the individual cores can be controlled separately or even turned off [4]. In mainstream Intel multi-cores, a technique called Turbo Boost Technology, TBT, is used to allow adjustments of core frequency at runtime [5]. Considering the number of active cores, estimated power requirements and CPU temperature, TBT can determine a maximum frequency for a core. The frequency can be incremented in steps of 133 MHz to give a boost in performance while still maintaining the power envelope. To save energy, it is possible to power down cores when they are idle. Other multi-core vendors offer similar techniques. In the Intel "Single-chip Cloud computer", 48 cores are integrated on a single chip and fine-grained power management is made possible by the 8 voltage islands and 28 frequency islands defined on the chip. The supply voltage can be scaled from 1.3 to 0V in 6.25mV steps. Voltage islands can be set to 0.7V for idle cores, a value that is a safe voltage for state retention, or completely collapsed to 0V, if retention is unnecessary [6]. State retention is closely linked to the concept of sleep modes covered in the next paragraph.

2.1.2 Sleep Modes

There are many real time applications where a task has to be completed within a given time frame. Often a performance guarantee for that deadline is of importance, but being faster has little or no value. In such cases, it is tempting to use DVFS or similar techniques to slow down the computation and save energy. However, if much of the system can be put into sleep mode after the execution has fin-

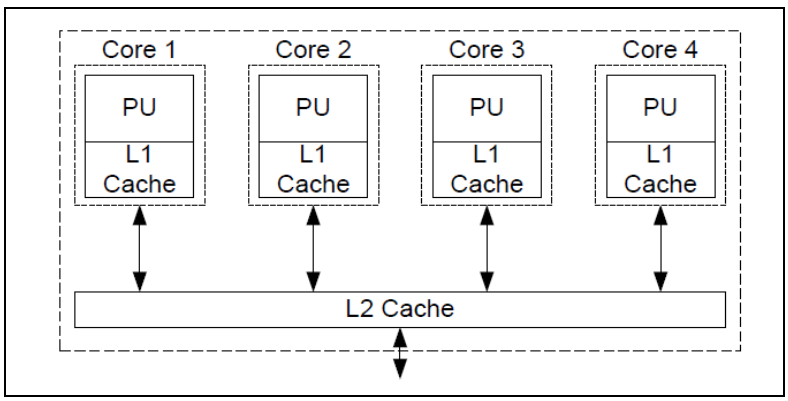

Figure 3: Quad-core Processor Example. PU = Processing Unit. L1 and L2 caches are explained in Section 2.2.

© Novática

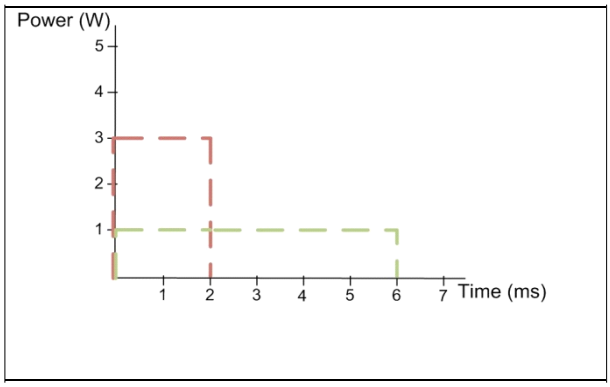

Figure 4: Alternative Designs, Simple Example.

ished, it might turn out to be more energy efficient to combine a fast and power-hungry period of execution with the use of sleep modes. This is illustrated in Figure 4 where case A (red) is one design doing a given computational task in 2 ms using 3 watt while case B (green) is the same task slowed down to 6 ms but using 1 watt. We assume that the task is done repeatedly, and must be completed within 6 ms when it occurs.

In the example above, the two alternatives would give the same energy consumption if we can assume that the unit can be completely turned off while not executing. However, if the unit has to retain some state during the 4 ms of time saved, the total energy consumption will be higher. But, on the other side, if some peripheral units can be turned off when the unit is not executing, the fastest alternative will be most energy efficient. This latter effect often motivates embedded system designers to aim at "unnecessarily fast" execution to save energy. DVFS and other slow down techniques (see also Section 2.2) must therefore be balanced against the use of sleep-modes. The situation is in practice much more complex. First of all, the energy consumed during execution will normally vary and not be constant as we assumed in Figure 4. Further, a computer will have different components that can be put into different levels of sleep. As an example, the EFM32 energy friendly microcontrollers from Energy Micro use 5 energy modes: run, sleep, deep sleep, stop and shutoff mode [7]. More details about different kinds of sleep modes can be found as a part of the large ACPI Specification [8]. ACPI is short for Advanced Configuration and Power Interface and describes a common industrial interface enabling operating system directed configuration and power management. Sleep modes are also known as standby modes or suspend modes.

2.2 Parallelism and Other Architectural Techniques

The architecture sub-tree of Figure 1 refers to a vast field of research. It contains many different subfields each of which is a large research area. Some examples are CPU design (also called micro-architecture), memory systems (which includes among others cache hierarchies, memory compression and 3D stacking technology) and system interconnections. In Figure 1, we used only two broad classes: multi-cores and customization. There are also architectural techniques that apply to single-cores. Kontorinis et al. have described a highly adaptive processor design using a table-driven approach that is used to configure the core's elements allowing tradeoffs between execution time and energy consumption. The dynamic configuration is based on run-time resource demands and user-defined performance. Bhattacharjee and Martonosi have investigated how custom design of Translation Lookaside Buffers, TLB, can improve overall performance of a CMP. At first sight, this topic appears to be related only to performance and not to energy consumption. However, "multi-core aware" TLBs will improve performance for the same energy budget, which leads to an improvement in energy-efficiency.

2.2.1Multi-core Processors

A simple homogeneous multi-core architecture was sketched in Figure 3. As explained in the previous section there is a cubic relation between processor speed (clock frequency), supply voltage and power. 2 cores running at $f/2$ can do a computation in the same time T seconds as one core running at f. In a very simplistic view, considering only dynamic power, and assuming that supply voltage can be scaled down to $V/2$ when frequency is set to $f/2$, these two cores will use 1/8 of the power each, and the total power consumption will be 1/4 for 2 cores doing the same work equally fast. In reality, it is not that simple, and some of the energy savings are lost. Normally, quite a bit of overhead is involved when transforming an application into a parallel program. This depends on the parallelism available in the application, the programming tools used and the underlying architecture. There are some, so-called embarrassingly parallel applications that can be executed on a large number of cores with very little coordination and communication between the cores, i.e. having a large computation/communication ratio.

However, very often lots of communication are involved in the parallelisation, and a higher degree of parallelism will reduce the computation/communication ratio to a point

> “We describe how parallelization in both HW and SW can help reduce the energy consumption, and discuss several examples of how throttling (slowing down) and simplicity give the same effect”

 © Novática

> “Green computing is an interdisciplinary, large and rapidly increasing area”

where the parallelisation gives no further speedup. When energy-efficiency is taken into account in addition to execution time, it becomes even more important to avoid too much overhead. This explains the development of different *throttling techniques* in addition to processor slowdown via DVFS. These can restrict or reduce the degree of thread level parallelism, TLP, as in Feedback-Driven threading, FDT [9] and Dynamic Concurrency Throttling , DCT [3]. A related technique is Fetch Throttling that refers to the ability to control the number of instructions a processor fetches each cycle. This will reduce the amount of instruction level parallelism, ILP, exploited by a processor. Throttling is not a new technique in computer architecture. It was used as early as in 1980s in the Manchester Dataflow Machine, MDM where an architectural technique called Throttle was designed to control the parallelism [10]. The dataflow computing paradigm allows for very fine-grained parallelism and applications could easily express large amounts of parallelism that gave the MDM excessive use of storage, hence it had to be throttled. A somewhat similar situation is so called thrashing in operating systems, a situation that occurs when so many processes have to be context-switched that this overhead drastically reduces the processors ability to do computations.

The simplistic multi-core in Figure 3 also shows another aspect of crucial importance for energy efficiency in all contemporary computers. For several decades we have seen a growing performance gap between processor and memory (also called the memory wall) — the speed of processors (core) has increased much faster than the speed of memory access. A direct result is the need for memory hierarchy, including several levels of fast and small memory (caches). Today’s multi-cores have at least two levels of cache: in the example in Figure 3 there is a level 1 (L1) cache in each core and one shared level 2 (L2) . These caches are beneficial for energy saving in two main ways. All memory access that can be kept locally, i.e. on-chip, will save huge amounts of both time and energy. Note also that the off-chip memory bandwidth is expected to become an increasingly severe bottleneck. As a result, many new programming languages have keywords that can be used by programmers to specify data locality.

A second advantage of cache hierarchies in multi-core systems is that the on-chip shared cache gives a fast way for the processes on the different cores to communicate. This can be used by the programmer as traditional shared memory programming, which is considered to be much easier than the alternative of message passing. Both these parallel programming paradigms have been used on multiprocessors for decades. For the future, we believe the advent of multi- and many-cores and large on-chip shared caches will foster great improvements in both per chip performance and energy efficiency.

2.2.2 Amdahl’s Law and Asymmetric Multi-cores

If we generalize the homogeneous multi-core architecture of Figure 3 to a chip with *n* equally powerful processor cores it will become an ideal execution vehicle for an application that can undergo an ideal parallelization into *n* equally sized computational tasks. However, as stated by Amdahl’s law, the part of an application that is serial, i.e. cannot be parallelized, will severely limit the maximum speed-up that can be obtained by parallelisation. A direct consequence is the increasing interest in asymmetric multi-cores where at least one of the cores is more powerful, and well suited to "crunch" the serial part of the computation. The effect of such an architectural choice and the more general discussion about few fat cores vs. many thin cores is elegantly discussed in the paper *Amdahl’s Law In the Multicore Era* by Hill & Marty [11]. A related study with focus on energy-efficiency is the paper *Maximizing Power Efficiency with Asymmetric Multicore Systems* by Fedorova et al. [12].

2.2.3 Customization and Heterogeneity

A multi-core processor extended to include a single core with enhanced computational power or asymmetric multi-cores are in the category of heterogeneous multi-cores. This is a large class of very diverse microprocessor products and many of them contain very different processors on the same chip — with different architectures and even programmed in different languages. Generally, this makes programming of such chips much more complex. Nevertheless, the reason for its popularity is the benefits of customization. Most applications consist of different phases or subtasks that can be executed more efficiently by a special purpose processor or some dedicated HW unit. This is even truer for energy-efficiency, as witnessed by a large amount of different customized MPSoC, Multiprocessor System-on-Chip, chips used in embedded systems.

2.2.4 The Green500 List

Also among supercomputers, when it comes to energy efficiency, heterogeneous multi-cores are dominating today. This is evident from the recent June 2011 Green500 ranking list of power efficient computers, <http://www.green500. org/lists/2011/06/top/list.php>. The list resembles the more famous top500.org list of the most powerful supercomputers, but Green500 use performance/watt as ranking criteria.

IBM supercomputers lead the Green500’s latest list of the world’s most energy efficient high-performance com-

© Novática

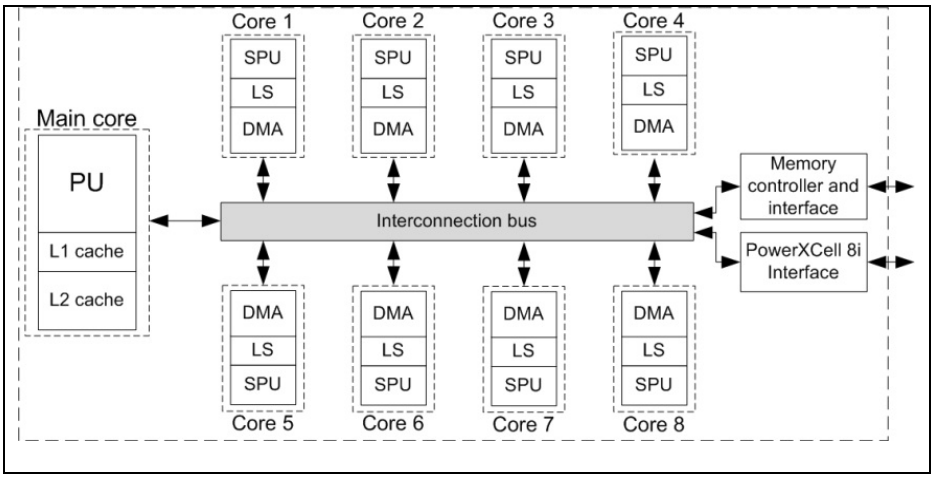

Figure 5: PowerXcell 8i, Simplified View.

puters. The two Blue Gene/Q prototypes use a 18 core processor that is based on a modified version of the Power A2 chip, but one running at a lower speed – 1.6GHz versus 2.3GHz. 16 cores are used for computing and 1 core for running the operating system. The 18th core is used as a spare. The University of Nagasaki built a hybrid cluster called DEGIMA with Intel Core i5 processors matched to ATI Radeon graphics controllers all linked together with 40Gb/sec InfiniBand switches. Its components (7,600 cores total) are relatively inexpensive and their energy efficiency gave DEGIMA the number three ranking on the Green 500 list. The fourth most energy-efficient supercomputer is the Tsubame 2.0 hybrid built by Tokyo Institute of Technology, comprising HP ProLiant servers and Nvidia Tesla GPU coprocessors.

Another IBM hybrid – the PLX Cluster at the Cineca/SCS research consortium in Italy, made up of 274 iDataPlex servers using six-core Xeon 5600 processors and Nvidia M2070 GPU coprocessors is occupying the fifth position. The most powerful supercomputer (as ranked in the Top500), the K supercomputer built by Fujitsu for the Japanese government, is ranked as the sixth most energy efficient machine. This is a monolithic parallel machine, using the eight-core Sparc64-VIIIfx processors. Other machines at the top of the Green500 list include a set of rather special supercomputers, the QS22 blade servers, based on IBM's Cell PowerXCell 8i chips and a 3D torus interconnect. Figure 5 shows a simplistic view of the architecture that is the 2nd generation product of the Cell Broadband Engine architecture developed by Sony, Toshiba and IBM. The processor consists of a quite standard main processor based on the Power PC architecture, and 8 specialised cores called synergistic processing units (SPU). These have a local store (LS) and a DMA engine for fast memory transfers. By explicit use of these two resources the programmer can achieve high power efficiency, but the price paid is the increase in software development complexity.

A further step toward using specialized or customized HW units is reconfigurability as offered by the successful FPGA, Field Programmable Gate Array, technology. FPGA is hardware that can be reconfigured during run-time — and is often called programmable HW. A FPGA chip can be configured to be ideal for executing one phase of an application, and then be reconfigured to be well suited for a later phase with different computational demands. Zain *et al* implemented an autofocus algorithm used in synthetic aperture radar systems in a FPGA as a case study. They showed that two different parallel implementations using many cores running on much lower clock frequency could achieve both better performance and highly improved energy efficiency [13]. Again, the price paid for excellent energy efficiency results is more difficult programming, although the researchers raised the abstraction level of the FPGA programming by using a language called Occam-pi.

Thus, good energy efficiency and high programmability can be conflicting goals, and we will sketch a few more of these challenging trade-offs in Section 4.

2.3 Power Management and Techniques at the OS Level

Fortunately, software tools and systems have been developed that can improve energy efficiency without the end user or programmer being aware of it. There are also tools giving good help during energy efficiency engineering.

> "It is predicted that near threshold computing can reduce energy requirements by 10 to 100 times or even more in future systems"

 © Novática

> “Another important technique is dynamic voltage scaling and frequency scaling, DVFS. It requires special hardware features, but it is normally controlled by software”

Several of these techniques *control the amount of parallelism* dynamically. Suleman *et al* have developed a framework called *feedback-driven threading*, FDT, that dynamically controls the number of threads using runtime information. FDT can be used to implement Synchronization-Aware Threading, SAT, controlling the number of threads depending on the amount of synchronization. Further, it can form the basis for Bandwidth-Aware Threading, BAT, that predicts how many threads we can execute before the off-chip bus gets saturated. Both these techniques, and their combined use, can reduce the execution time and power consumption with up to more than 70% [9]. Singh *et al* have developed a power aware thread scheduler that can be used to ensure that the system executes within a maximum power envelope [14].

Both the above mentioned techniques exploit the *performance counters* of modern microprocessors giving low level programmers deep insight into processor state and dynamic behaviour. Bertran *et al* have shown that such models can be used to not only model power consumption with good accuracy, but also to provide per component power consumption [15]. Bhattacharjee and Martonosi propose and evaluate simple but effective methods for predicting critical threads in parallel applications. It is shown by the above mentioned research that accurate predictors can be built using performance counters. If a system can accurately find the critical, or slowest, threads of a parallel program, this information can be used for load rebalancing, energy optimization, and capacity management on constrained resources.

A lot of research has been done in *power management* tools used at the Operating System or Run Time System level. Dynamic power management policies perform better than static power management and it has been shown that a global view of the activity of all the cores in the system can gives the most optimal control of the power consumption [4]. Borkar and Chien outline a hypothetical heterogeneous processor consisting of a few large cores and many small cores, where the supply voltage and frequency of each core is controlled individually. This fine-grained power management improves energy-efficiency without burdening the application programmer, i.e. controlled by the run time system.

A recent example of an innovative tool that helps in developing energy efficient software is the *energyAware profiler* developed by Energy Micro for use with their energy friendly microcontrollers [7]. This tool plots a graph of the use of current during execution, and by clicking on a power-peak the developer is automatically directed to the source code causing the peak. We expect that similar tools will be provided for larger processors in the future.

2.4 Energy Aware Algorithms and Data Structures

A recent paper by Susanne Albers [16] presents a survey of algorithmic techniques to solve problems in energy management. The goal of these algorithms is to reduce energy consumption while minimizing compromise to service. One of their major challenges is that the power management problem normally is a so called online problem meaning that at any time the system is not aware of future events. A power down operation typically uses very little energy, but since a power up operation will consume some extra energy it might be a wrong decision to power down a unit being idle if the idle-time appears to be very short. The challenge becomes even larger when there are more than two power states.

The same survey paper also covers dynamic speed scaling as made possible by DVFS described in Section 2.1. This gives the OS scheduler a more challenging job but also more options for saving energy. Further, scheduling of jobs with deadlines, trying to minimize processor temperature or the more classical goal of minimal response time all give different algorithmic challenges. We agree with Albers that speed scaling techniques in multiprocessors and multicore based computers will become increasingly important. In Section 3, we present some early experiments in a related area. The future will show new ways of expressing parallel algorithms and new ways for the corresponding application to interact with the scheduling and power management algorithm in the underlying OS and RTS.

Also in the area of algorithms the value of *simplicity* must never be forgotten. Traditional algorithmic studies use asymptotical analysis and complexity classes. As an example, a $O(\log n)$ sorting algorithm is considered to have a better performance than a $O(\log^2 n)$ algorithm (n is the number of items to sort). However, as exemplified by a study of Cole’s parallel merge sort, complex algorithms have large complexity constants, so a $O(\log^2 n)$ algorithm might be faster than a $O(\log n)$ algorithm for all practical problem sizes.

The term "simplicity" has several interpretations. In the example above, it was synonymous with low complexity constants. Another interpretation is simple to understand. Again using sorting as example, the old odd-even transposition sort is perhaps the easiest to understand parallel algorithm, however being $O(n^2)$ it can also be said to be a brute force algorithm where simplicity in the algorithm is achieved by adding redundant operations. In this paper, simplicity refers to low implementation complexity or little parallelisation overhead. The value of simplicity is well known among engineers and known as "principles" such as Occam’s razor and the KISS principle ("Keep It Simple

© Novática

Stupid"). The latter has inspired the paper *The KILL Rule for Multicore*. Here, "Kill" stands for Kill If Less than Linear, and describes an approach for power-efficient multi-core design.

A recent paper by Nir Shavit explains how the advent of multi-cores gives a need for new research in data structures [17]. To improve parallel performance we should try to reduce the serial fraction that may be caused by using a traditional shared data structure that allows access to only one core at a time. New research in lock-free data structures and relaxation methods can be used to implement slightly more complex but *concurrent* data structures that in many cases can reduce this serial fraction. This can give higher speed up and better performance, and if the added complexity is not too high, it can improve the energy efficiency. In addition, the new multi-cores also have very large on-chip shared caches. Since the increased cost both in time and energy. for memory accesses that miss in this last-level cache (LLC). might be as much as 100 times larger or more, there is great importance in keeping the data on-chip. We believe this fact also provides motivation for research into new energy-efficient multi-core data structures.

3 Case Study: Energy-aware Task Based Programming

To build competence in how to save energy by parallelization we have developed an experimental framework combining an architecture simulator with full system simulation with a power and area estimation tool. We studied the impact of load-balancing on energy-efficiency in parallel executions and we identified several issues that limit the energy-efficiency of a system as the number of cores grows. For our experiments we used a Task Based Programming, TBP, approach to parallelize our test benchmarks. The base concept of the TBP model is that the programmer should identify and annotate pieces of code (tasks) which can be executed concurrently with other tasks, while the complexity of the hardware is covered by an abstraction. Each application was organized as a set of computational units (called tasks) that were scheduled across different cores.

The experimental framework we put in place for this study combines an architectural simulator and a power estimation tool. We used GEM5 [18] to perform full-system simulations of several multi-core platforms. The simulations were run on the Kongull compute cluster made available by the NTNU HPC Division (http://docs.notur.no/ntnu). The performance statistics gathered from GEM5 were used as input for a power, area and timing estimation tool called McPAT [19]. This setup allowed us to record the behaviour (for both performance and energy consumption) for all key components of the system: cores, cache hierarchy, main memory etc.

We simulated our test benchmarks on systems with 2 -

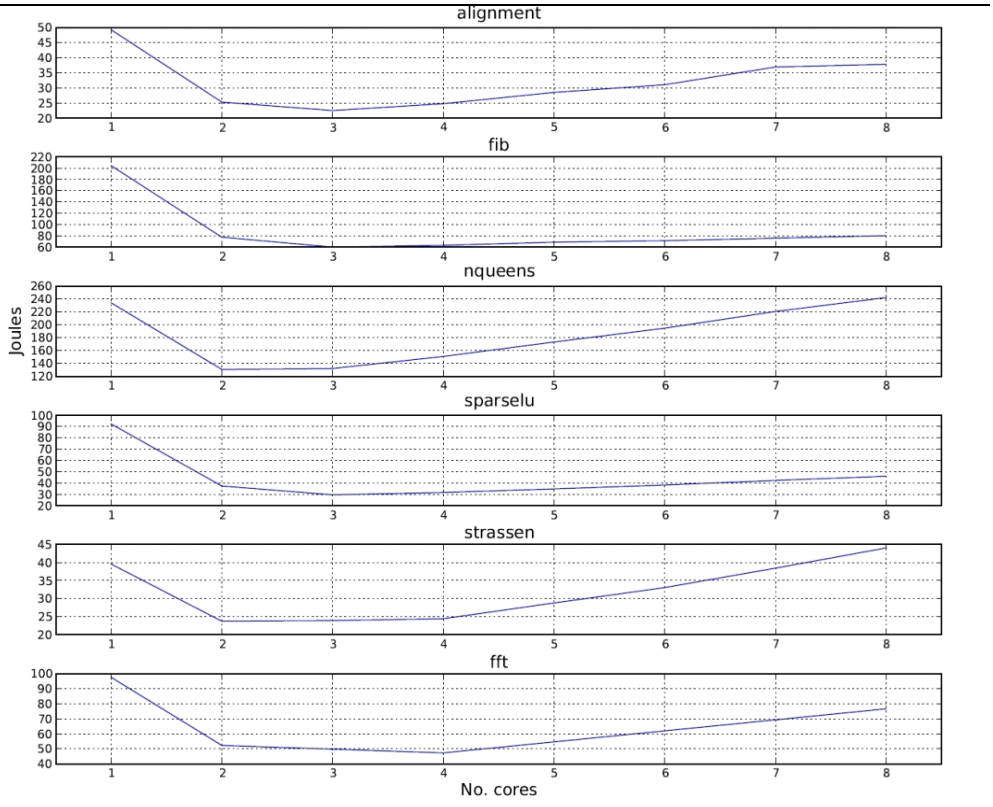

Figure 6: Energy consumed by running 6 Test Applications on Different Numbers of Cores.

 © Novática

> “Also among supercomputers, when it comes to energy efficiency, heterogeneous multi-cores are dominating today”

8 cores and for various levels of work imbalance. For our measurements we used the Energy-Delay-Product, EDP, in order to ensure that a complete profile of the application (one that includes performance and energy consumption) is created. For each core count we recorded the load balancing characteristics with the best EDP value. Figure 6 shows the energy consumption of the most power efficient execution for number of cores ranging from 1 to 8 for 6 different benchmark applications. We see that all applications reach minimum energy consumption for 2, 3 or 4 cores. This reflects that our benchmarks are all relatively small — a necessary choice for avoiding our detailed simulations to becoming too lengthy. The "bathtub" curves are due to two issues we observed: parallelization overhead and stalls caused by the synchronization portions in the code. All these negative effects grow together with the core count and at a certain point the decrease in energy consumption stops. Fib and SparseLU perform better than the other benchmarks when scaling up the core count because they require less task synchronization. Fib uses a recursive approach to generate a very extensive task tree. This ensures that every core will find work every time it needs it. SparseLU generates far less tasks, but they require less synchronization so the cores are stalled a very small amount of time during the execution.

Our experiments show a significant potential for energy-efficiency improvements of parallel executions on multi-cores compared to the serial version of the same application on a single-core system. In all, we think the results we have found so far are promising and motivate further research into energy-efficiency through parallelization.

4 Concluding Remarks and Future Work

We have seen that low-power design, HW-techniques, multi-core architectures, parallelism, heterogeneity and novel software techniques both at the OS level and in user application all can contribute to improved energy efficiency. In Figure 7 we show the "5 Ps of parallel processing": performance, predictability, power-efficiency, programmability and portability. When designing for better energy-efficiency, it is important to have a holistic view and be aware of the possible conflicting goals. Optimizing for one of these Ps very often will reduce the possibility of achieving some of the others. As an example, the Cell processor has achieved outstanding results for power-efficiency at the cost of reduced programmability. Also, optimizing an application to use the underlying architecture in an energy-efficient way will very often introduce constructs and adaptations that reduce portability. Predictability denotes the wish of the user to know how long a computation will take. If it is of high importance, as in real time systems, the system must be designed to meet performance guarantees. But if predictability can be given low priority, the task can be postponed to a low-load period, running using less energy.

We plan to extend and improve the research sketched in Section 3 in different ways. On the programming side, we will test OpenMP programs and other variants of task based programming. On the execution side, we will test and measure the energy consumption on the Intel Sandy Bridge architecture (Core i7) and eventually other multi-cores. We wish to extend our experiments towards a larger number of cores and at the same time try to model a more recent and energy-focused processor. The full-system simulations used are very time consuming and limit the current approach. Therefore, we work on using statistical sampling methods to be able to simulate the main effects with drastically reduced execution time.

There are many challenges, but these are also opportunities for innovation in new products, solutions and exciting research. *We see the increased focus on energy efficiency*

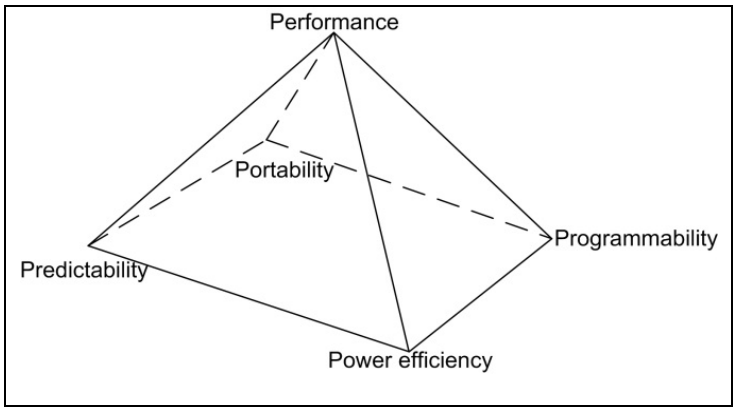

Figure 7: 5 Ps of Parallel Processing.

© Novática

> “We see the increased focus on energy efficiency as a revitalizing force for large parts of the Computer Science field”

as a revitalizing force for large parts of the Computer Science field.

References

[1] B. Zhai et al. "Energy efficient near-threshold chip multi-processing". Proceedings of the 2007 Int’l Symp. on Low Power Electronics and Design Series, ISLPED 2007.

[2] V. Venkatachalam, M. Franz. "Power reduction techniques for microprocessor systems". ACM Comput. Surv., September, 2005.

[3] M. Curtis-Maury et al. "Prediction models for multi-dimensional power-performance optimization on many cores". Proceedings of the 17th Int’l Conf. on Parallel Architectures and Compilation Techniques, PACT 2008.

[4] C. Isci et al. "An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget". Proceedings of the 39th annual IEEE/ACM Int’l Symp. on Microarchitecture, MICRO 2006.

[5] Intel Corp. "2nd Generation Intel Core Processor Family Desktop". <http://download.intel.com/design/processor/datashts/324641.pdf>.

[6] J. Howard et al. "A 48-core IA-32 Message-Passing Processor with DVFS in 45nm CMOS". IEEE International Solid-State Circuits Conference, 2010.

[7] Energy Micro website, <http://www.energymicro.com/>.

[8] Advanced Configuration and Power Interface Specification, Revision 4.0, June 16 2009. <http://www.acpi.info/spec40.htm>.

[9] M.A. Suleman et al. "Feedback-Driven Threading: Power-Efficient and High-Performance Execution of Multi-threaded Workloads on CMPs". Proceedings of the 13th int’l conf. on architectural support for programming languages and operating systems, ASPLOS 2008.

[10] C. Ruggiero, J. Sargeant. "Control of parallelism in the Manchester Dataflow Machine". In Lecture Notes in Computer Science - Functional Programming Languages and Computer Architecture, 1987.

[11] M. D. Hill, M. R. Marty. "Amdahl’s Law In the Multicore Era". IEEE Computer, July 2008.

[12] A. Fedorova et al. "Maximizing Power Efficiency with Asymmetric Multicore Systems". Communications of the ACM, December 2009.

[13] Zain-ul-Abdin et al. "Programming Real-time Autofocus on a Massively Parallel Reconfigurable Architecture using Occam-pi". Proceedings of IEEE 19th annual Int’l Symp. on Field-programmable Custom Computing Machines, FCCM 2011.

[14] K. Singh et al. "Real Time Power Estimation and Thread Scheduling via Performance Counters". SIGARCH Comput. Archit. News, July 2009.

[15] R. Bertran et al. "Decomposable and Responsive Power Models for Multicore Processors using Performance Counters". Proceedings of the 24th ACM Int’l Conf. on Supercomputing, ICS 2010.

[16] S. Albers. "Energy-efficient algorithms". Communications of the ACM, May 2010.

[17] S. N. Shavit. "Data structures in the multicore age". Communications of the ACM, March 2011.

[18] N. L. Binkert et al. "The M5 simulator: Modeling networked systems". IEEE Micro, July-August 2006.

[19] S. Li et al. "McPAT: An integrated power, area, and timing modeling framework for multicore and many-core architectures". Proceedings. of the 42nd annual IEEE/ACM Int’l Symp. on Microarchitecture, MICRO 2009.

[20] Suzanne Rivoire et al. "Models and metrics to enable energy-efficiency optimizations". Computer, December 2007.

© Novática