

# A survey of energy saving techniques for mobile computers

Gerard J.M. Smit, Paul J.M. Havinga

University of Twente  
department of Computer Science  
P.O. Box 217, 7500 AE Enschede  
the Netherlands  
e-mail {smit, havinga}@cs.utwente.nl

## *Abstract*

Portable products such as pagers, cordless and digital cellular telephones, personal audio equipment, and laptop computers are increasingly being used. Because these applications are battery powered, reducing power consumption is vital.

In this report we first give a survey of techniques for accomplishing energy reduction on the hardware level such as: low voltage components, use of sleep or idle modes, dynamic control of the processor clock frequency, clocking regions, and disabling unused peripherals. System-design techniques include minimizing external accesses, minimizing logic state transitions, and system partitioning using application-specific coprocessors.

Then we review energy reduction techniques in the design of operating systems, including communication protocols, caching, scheduling and QoS management. Finally, we give an overview of policies to optimize the code of the application for energy consumption and make it aware of power management functions. Applications play a critical role in the user's experience of a power-managed system. Therefore, the application and the operating system must allow a user to control the power management. Remarkably, it appears that some energy preserving techniques not only lead to a reduced energy consumption, but also to more performance.

## **1 Introduction**

The technologies of PDA, digital cellular phone and smart card, when combined and integrated well, have the potential of replacing all of the things people have to carry around with them by one small device, the Pocket Companion. It is a small portable computer and wireless communications device that can replace cash, cheque book, passport, keys, diary, phone, pager, maps and possibly briefcases as well. The combination of an intelligent information system and a location system engenders many new types of applications, such as admission control, digital chequebook, paging, and an automatic diary that keeps track of where you were and with whom.

The Moby Dick project [Moby Dick 95] is a joint european project (Esprit Long Term Research 20422) to develop and define the architecture of a new generation of mobile hand-held computers. The design challenges lie primarily in the creation of a single architecture that allows the integration of security functions, externally offered services, personality, and communication. Research issues include: security, energy consumption and communication, hybrid networks, data consistency, and environment awareness.

In the Moby Dick architecture, Quality of Service (QoS) is no longer a networking issue alone, but a framework to model integration and integrated management of all the system services and applications in the Pocket Companion.

### *Energy consumption*

Portable and hand-held computers must be careful not to waste the scarce energy resources in their batteries. Even though battery technology is improving continuously and processors and displays are rapidly improving in terms of power consumption, battery life and battery weight are issues that will have a marked influence on how hand-held computers can be used. More extensive and continuous use of network services will only aggravate this problem since communication consumes relatively much energy. Research is needed to provide intelligent policies for careful management of the power consumption while still providing the appearance of continuous connections to system services and applications.

Several researchers have studied the power consumption pattern of mobile computers. However, because they studied different platforms, their results are not always in line, and sometimes even conflicting. Lorch reported that the energy use of a typical laptop computer is dominated by the backlight of the display, the disk and the processor [Lorch 95]. Stemm et al. concluded that the network interface consumes at least the same amount of energy as the rest of the system (i.e. a Newton PDA) [Stemm 96]. If the computer is able to receive messages from the network even when it is 'off', the energy consumption increases dramatically. Ikeda et al. observed that the CPU and memory power consumption has been on the rise the last few years (see table 1) [Ikeda 94].

*Table 1: the breakdown of power consumption for IBM ThinkPad laptops*

	1992	1993	1994	1995
LCD	55%	52%	43%	37%
Video System	9%	6%	7%	6%
DC-DC Converter	9%	8%	6%	9%
Hard Disk	8%	11%	12%	8%
CPU & Memory	18%	22%	32%	40%

Laptops use several techniques to reduce this energy consumption, primarily by turning them off after a period of no use, or by lowering the clock frequency. Some researchers proposed to replace the hard disk by a flash RAM.

There is an inherent trade-off between energy consumption and performance, since low-power techniques have associated disadvantages. For instance, decreasing the CPU frequency can raise response time, and spinning down the disk causes a subsequent disk access to have a high latency.

Throughout this paper, we discuss 'power consumption' and methods for reducing it. Although they may not explicitly say so, most designers are actually concerned with reducing energy consumption. This is because batteries have a finite supply of energy (as opposed to power, although batteries also put limits on peak power consumption as well). Energy is the time integral of power; if power consumption is a constant, energy consumption is simply power multiplied by the time during which it is consumed. Reducing power consumption only saves energy if the time required to accomplish the task does not increase too much. A processor that consumes more power than a competitor may or not may not consume more energy for a certain program. For example, even if processor A's power consumption is twice that of processor B, A's energy consumption could actually be less if it can execute the same

program more than twice as quickly as B.

### *Outline of the paper*

In this paper we will discuss a variety of energy reduction approaches that can be used when building an energy efficient system. First of all, we have to use components that use the latest developments in low power technology. Energy reduction techniques can be applied in all levels of the system. It is not just a problem of the power-conscious system hardware designer, but also involves careful design of the operating system and application programs. Furthermore, because the applications have direct knowledge of how the user is using the system, this knowledge must be penetrated into the power management of the system.

We first study possible energy reduction strategies that can be applied in the system architecture. Then we give an overview of energy saving mechanisms in the operating system domain, including network protocols. Finally, since applications play the most critical role in the user's experience with a power-managed system we present an overview of current research in this area.

## **2 Energy reduction in the system architecture**

Most components are fabricated using CMOS technology. A first order approximation of the power consumption of CMOS circuitry is given by the formula:

$$P = C V^2 f$$

where  $P$  is the power in Watts,  $C$  is the effective switch capacitance in Farads,  $V$  is the supply voltage in Volts, and  $f$  is the clock frequency in Hertz [Lapsley 94]. This equation suggests that there are essentially three ways to reduce power:

- reduce the supply voltage  $V$ ,
- reduce the capacitive load  $C$ ,
- reduce the switching frequency  $f$ .

### **2.1 Reducing supply voltage**

Power consumption is proportional to the square of supply voltage, which means that reducing a processor's supply voltage can result in dramatic savings in power. For example, reducing a supply voltage from 5.0 to 3.3 volts (a 44% reduction) reduces power consumption by about 56%. As a result, most processor vendors now have three volt versions. The problem that then arises is that lower supply voltages will cause a reduction in performance. In some cases, low voltage versions are actually five volt parts that happen to run at the lower voltage. In such cases the system clock must typically be reduced to ensure correct operation. Therefore any such voltage reduction must be balanced against any performance drop. To compensate and maintain the same throughput, extra hardware can be added. This is successful up to the point where the extra control, clocking and routing circuitry adds too much overhead [Rabaey 94].

In other cases, vendors have introduced "true" three volt versions of their processors that run at the same speed as their five volt counterparts.

### **2.2 Clock frequency control**

Reducing clock frequency  $f$  alone does not reduce energy, since to do the same work the system must run longer.

### *Sleep or idle modes*

Some processors have sleep or idle modes. Typically they turn off the clock to all but certain sections of the processor to reduce power consumption. Because CMOS power consumption is proportional to the clock frequency, turning off the clock to the processor can greatly reduce power consumption. While asleep, the processor does no work. A wake-up event wakes the processor from the sleep mode. Processors may require different amounts of time to wake up from different sleep modes. For example, many 'deep sleep' modes shut down on-chip oscillators used for clock generation. A problem is that these oscillators may require microseconds or sometimes even milliseconds to stabilize after being enabled. So, it is only profitable to go into deep sleep mode when the processor is expected to sleep for a relatively long time.

### *Frequency and voltage control*

Weiser et al. [Weiser 94] have proposed a system in which the clock speed and operating voltage is varied dynamically under control of the operating system while still allowing the processor to meet its task completion deadlines. They point out that in order to operate properly at a lower voltage, the clock rate must be simultaneously reduced. This method can be used in the scheduler of the operating system (see section 3.3).

In most components there is a linear relation between voltage and clock rate. Suppose a task has a deadline of 100 ms, but it will only take 50 ms of CPU time when running at full speed to complete. A normal system would run at full speed for 50 ms, and the idle for 50 ms in which the CPU can be stopped. Compare this to a system that runs the task at half speed, so that it completes just before its deadline. If it can also reduce the voltage by half, then the task will consume a quarter of the energy of the normal system. This is because the same number of cycles are executed in both systems, but the modified system reduces energy use by reducing the operating voltage.

### *Clocking regions*

Turning off the system clock to unused logic or peripherals is an obvious way to reduce power consumption [Larri 96], [Intel486SX]. Control can be done at the hardware level or it can be managed by the operating system or the application (see also section 3.4).

### *Asynchronous design methodology*

A synchronous circuit requires that a clock signal be distributed evenly to all parts of a circuit therefore wasting power when particular blocks of logic are not utilized, for example, to a floating point unit when integer arithmetic is being performed. CMOS is a good technology for low power as gates only dissipate energy when they are switching. Normally this should correspond to the gate doing useful work, but unfortunately in a synchronous circuit this is not always the case. Many gates switch because they are connected to the clock, not because they have new inputs to process. The biggest gate of all is the clock driver, and it must switch all the time to provide the timing reference even if only a small part of the chip has anything useful to do.

Asynchronous circuits though are inherently data driven and are only active when performing useful work. Parts of an asynchronous circuit that receives less data will automatically operate at a lower average frequency. Unfortunately, asynchronous circuits are larger than synchronous circuits.

## **2.3 Reducing the capacitive load**

### *Reduce external accesses*

Energy consumption in CMOS circuitry is proportional to capacitance. Connections to exter-

nal components, such as external memory, typically have much greater capacitance than connections to on-chip resources. As a result, accessing external memory can increase energy consumption. So, the system should be optimized to use on-chip resources e.g. caches and registers. Furthermore, use few external outputs, and have them switch as infrequently as possible.

### *Reduce logic state transitions*

As said before, energy consumption is proportional to the frequency at which signals change state from 0 to 1 or vice-versa and to the capacitance on the signal line. This is true of every signal path in a system, whether it is a clock signal, a data pin, or an address line. This implies that power consumption can be reduced by carefully minimizing the number of transitions.

For example, program counters in processors generally use a binary code. On average, two bits are changed for each state transition. A Gray code, in which typically a single bit changes, can give interesting energy savings. However, according to [Piguet 96], a Gray code incrementer requires more transistors to implement than a ripple carry incrementer. Therefore a combination can be used in which only the most frequently changing LSB bits use a Gray code.

### *Reversible logic*

Reversible logic [Merkle 93] or adiabatic logic tries to reduce energy consumption by not erasing information. Today's computers erase a bit of information every time they perform a logic operation. These logic operations are therefore called 'irreversible'.

We can improve the efficiency of erasing information with conventional methods, such as used in large cache systems.

An alternative is to use logic operations that do not erase information. These are called reversible logic operations, and in principle they can dissipate arbitrarily little heat. To achieve a completely reversible system (which erases no bits at all) is very difficult.

### *Hierarchical memory systems*

Hierarchical memory systems can be used in a processor system to reduce energy consumption. The basic idea is to store a frequently executed piece of code or frequently used data in a small memory close to or in the processor (a cache) (see also 2.4). As most of the time only a small memory is read, the energy consumption is reduced.

### *Minimize on-chip routing capacitance*

Power dissipation is proportional to capacity, and routing capacitance is the main cause of the limitation in clock frequency. Circuits that are able to run faster can do so because of a lower routing capacitance. Consequently, they dissipate less power at a given clock frequency. So, optimize the clock frequency of the design even if the resulting performance is far in excess of the requirements [Xilinx 95].

## **2.4 System partitioning and application specific coprocessors**

Processors often have to perform tasks for which they are not ideally suited. Although they can perform such tasks, they may still take considerably longer, and might be more energy demanding, than a custom hardware implementation.

Application-specific integrated circuits (ASICs) or dedicated processors placed around a standard processor can offer an alternative approach. A system designer can use the processor for portions of algorithms for which it is well suited, and craft an application-specific coprocessor (e.g. custom hardware) for other tasks. This is a good example of the difference

between power and energy: although the application-specific coprocessor may actually consume more power than the processor, it may be able to accomplish the same task in far less time, resulting in a net energy savings.

Furthermore, when the system is decomposed out of application-specific coprocessors the data traffic can be reduced, for instance because unnecessary data copies are removed. For example, in a system where a stream of video data is to be displayed on a screen, the data can be copied directly to the screen memory, without going through the main processor. By careful repartitioning a system, not only the power consumption can be reduced but the performance is actually improved as well [Mangione-Smith 96].

Abnous and Rabaey proposed an architecture that consists of a control processor (a general-purpose microprocessor core), surrounded by a heterogeneous array of autonomous, special-purpose satellite processors [Abnous 96].

Another way to reduce data traffic is to integrate a processor in the memory, as for example proposed by Patterson in intelligent RAM [Patterson 96], [McGaughy 96].

In other words, *localization* reduces the communication overhead in processors and allows the use of minimum sized transistors, which results in drastic reductions of capacitance. Pipelining and caching are examples of localization.

### 3 Energy reduction in the operating system

There are a number of system design or programming techniques in the design of an operating system that can be used to reduce power consumption. Some of these are outlined below.

#### 3.1 Communication

The wireless network interface of a mobile computer consumes a significant fraction of the total power [Stemm 96]. Measurements show that on typical applications like a web-browser or e-mail, the energy consumed when the interface is on and idle is more than the cost of receiving packets. This is because the interface is generally longer idle than actually receiving packets. Furthermore, switching between states (i.e. off, idle, receiving, transmitting) consumes time and energy.

There are several ways to reduce the power consumption involved in communication.

##### *System decomposition*

In normal systems much of the LAN protocol stack is implemented on the main processor. Thus, the network interface and the main processor must always be on for the LAN to be active. Because almost all data is transported through the processor, performance and energy consumption is a significant problem.

Just like the system partitioning as discussed in a previous section, decomposition of the network protocol stack and the data flow in the system can reduce the energy consumption. First, when the system is constructed out of independent components that implement different layers of the communication stack, unnecessary data copies are eliminated. Secondly, dedicated hardware can do basic signal processing and can move the data directly to its destination, thus keeping data copies off of the system bus. Finally, a communications processor can be applied to handle most of the lower levels of the protocol stack, thereby allowing the main processor to sleep for extended periods of time without affecting system performance or functionality.

##### *Error rates*

Wireless networks have a much higher error rate than the normal wired networks. In the presence of a high packet error rate, some network protocols (such as TCP) may overreact to packet losses, mistaking them for congestion. This leads to backing off to a lower transfer rate which increases the energy consumption because it leads to a longer transfer time. Any protocol that leaves a mobile receiver idle unnecessarily wastes energy. The limitations of TCP can be overcome by a more adequate congestion control during packet errors.

Buffering of data on a base station can be used to perform only local retransmissions that are caused by errors in the wireless network.

### *Sleep mode*

The network protocols can be modified to support full connectivity while being in a deep power down mode. Store-and-forward schemes for wireless networks, such as the IEEE 802.11 proposed sleep mode, not only allow a network interface to enter a sleep mode but can also perform local retransmissions not involving the higher network protocol layers. However, such schemes have the disadvantage of requiring a third party, e.g. a base station, to act as a buffering interface.

### *Medium access protocol*

The limited bandwidth of current wireless networks may cause needless energy consumption. The medium access protocols of a wireless system can be adapted and tuned for low energy consumption. A base station that divides the available bandwidth equally among 10 mobiles causes them to consume 10 times as much power (100 times as much power total!) compared to a base station that uses a TDMA protocol to coordinate delivery of data to receivers. An example of an energy aware MAC protocol is LPMAC [Mangione-Smith 96]. It uses peer-to-peer wireless networking to reduce power consumption across the entire network. One host or base station is responsible for traffic scheduling. Mobiles with scheduled traffic are indicated in a list, which allows mobiles without traffic to rapidly reduce power. By explicitly scheduling all bulk data transfers, a terminal is allowed to doze (and power off the receiver) as long as the network interface is reactivated at schedule time to receive the data at full speed.

Furthermore, as switching between states (i.e. off, idle, receiving, transmitting) consumes time and energy, the number of state-transitions have to be minimized.

## **3.2 Caching**

Memory considerations must also be taken into account in the design of any system. By employing an on-chip cache significant power reductions together with a performance increase can be gained.

Apart from caching data and instructions at the hardware level, caching is also applied in the filesystem of an operating system. The larger the cache, the better performance. Energy consumption is reduced because data is kept locally, and thus there is less data traffic. Furthermore, the energy consumption is reduced because less disk and network activity is required.

The file system can also try to collect disk operations. It therefore has to postpone low priority disk I/O only until the hard drive is running already or has enough data.

In a situation with varying and multiple network connectivity it may be wise to prefetch some information or postpone the actual transmission until a more power economic network is available. For example an application can schedule times to turn on the processor when it is connected to a wired network so that the application can download information from the network when it consumes less energy or does not need its batteries.

### 3.3 Scheduling

Weiser et al. [Weiser 94] have proposed a system that reduces the cycle time of a processor for power saving, primarily by allowing the processor to use a lower voltage. By detecting the idle time of the processor, they can adjust the speed of the processor while still allowing the processor to meet its task completion deadlines.

They classified idle periods into 'hard' and 'soft' events. Obviously, running slower should not allow requests for a disk block to be postponed. However, it is reasonable to slow down the response to a keystroke, such that processing of one keystroke finishes just before the next. Another approach is to classify jobs or processes into classes like background, periodic and foreground. With this sort of classification the processor can run at a lower speed when executing low priority background tasks only.

### 3.4 Energy manager

To take advantage of low-power states, the operating system needs to direct the processor to turn off (or down) when it is predicted that the net savings in power will be worth the time and energy overhead of turning off and restarting.

In the OnNow approach [OnNow] power management hardware is used to put a Personal Computer into a low-power sleeping state instead of shutting down completely, so that the system can resume working quickly. The operating system will enter the sleeping state when the computer is idle or when the user indicates to do so.

In order to achieve the OnNow vision of the 'always ready' computer, changes must be made to current designs for hardware, drivers, firmware, operating system, and applications. One of the key aspects is to move power management policy decisions and coordination of operations into the operating system. The operating system will control the power states of devices in the system and share this information with applications and users. This knowledge can be used and integrated in the Quality of Service model of operating systems.

In addition to that the ACPI specification defines a new hardware interface to the system board that enables the operating system to implement operating system-directed power management. This enables the system to turn on and off peripherals such as CD-ROMs, network cards, hard disk drives, and printers, as well as consumer devices connected to the PC such as VCRs, TVs, phones, and stereos. What's more, with this technology, peripherals will be able to activate the computer as well.

## 4 Energy efficient software

Finally, we can optimize the code of the application for energy consumption and make it aware of power management functions. At this level the user is involved. We now give an overview of current research in this area.

### 4.1 Compiler

As much of the power consumed by a processor is due to the fetching of instructions from memory, high code density can reduce energy consumption. However, this only works well when the execution cycle is not (much) longer. Today, the cost function in most compilers is either speed or code size. An energy aware compiler has to make a trade-off between size and speed in favour of energy reduction.

The energy consumed by a processor depends on the previous state of the system and the current inputs. Thus, it is dependent on instruction choice and instruction ordering. Reordering of instructions can reduce the switching activity and thus overall energy consumption. How-

ever, it was found not to have a great impact [Tiwari 94]. The compiler can have a greater impact on power consumption by concentrating on reducing the number of instructions with memory operands. The most energy can be saved by a better utilization of registers. It was also noted that writes consumes more energy, because a processor with a write-through cache (like the Intel 486) always causes an off-chip memory operation.

## 4.2 Applications

Applications play the most critical role in the user's experience of a power-managed system. In traditional power-managed systems, the hardware attempts to provide automatic power management in a way that is transparent to the applications and users. This results in some legendary user problems such as screens going blank during video or slide-show presentations, annoying delays while disks spin up unexpectedly, and low battery life because of inappropriate device usage. Because the applications have direct knowledge of how the user is using the system to perform some function, this knowledge must be penetrated into the power management decision-making in the system in order to prevent these kinds of user problems.

Obviously, careless application's use of the processor and hard disk drastically affects battery life time. For example, performing non-essential background tasks in the idle loop prevents the processor from entering a low power state (see for example [Lorch 96]). So applications have to be written energy aware.

## 5 Conclusions

More and more attention will be focused on low power design techniques as there will become an increasing numbers of portable, battery powered systems. Processor and system designers have a number of ways to decrease energy consumption. Energy consumption can be decreased by reducing the supply voltage, reducing the capacitive load and by reducing the switching frequency. Furthermore, they can take advantage of power management features where available, as well as decomposed system architectures and programming techniques for reducing power consumption. As the communication of a mobile station consumes a significant amount of energy, a careful design of *all* network layers is required.

Remarkably, it appears that some energy preserving techniques not only lead to a reduced energy consumption, but also to more performance. For example, optimized code runs faster, is smaller, and therefore also consumes less energy. Using a cache in a system not only improves performance, but, - although requiring more space - uses less energy since the data is kept locally. The approach of using application specific coprocessors is not only more efficient in terms of energy consumption, but has also a performance increase because the specific processors can do their task more efficient than a general purpose processor. Energy efficient asynchronous systems also have the potential of a performance increase, because the speed is no longer dictated by a clock, but is as fast as the flow of data.

However, some trade-offs need to be made. Most energy efficient systems use more area, not only to implement the new data flow or storage, but also to implement the control part. These systems often have less speed, that only can be improved by adding more hardware. Furthermore, energy efficient systems are more complex and need more hardware. The application specific coprocessor approach is more efficient than a general purpose processor, but less flexible. The latency from the user's perspective might be increased, because a system in sleep has to be wakened up. For instance, spinning down the disk causes the subsequent disk access to have a high latency.

Applications play a critical role in the user's experience of a power-managed system. Therefore, the application and operating system must allow a user to have influence on the power management.

Any consumption of resources by one application might affect the others, and as resources run out, all applications are affected. Since communication bandwidth, energy consumption and application behaviour are closely linked, we believe that a QoS framework is a sound basis for integrated management of the resources.

## 6 References

- [Abnous 96] Abnous A, Rabaey J.: "Ultra-Low-Power Domain-Specific Multimedia Processors," Proceedings of the IEEE VLSI Signal Processing Workshop, San Francisco, October 1996.
- [Ikeda 94] Ikeda T.: "ThinkPad Low-Power Evolution", IEEE Symposium on Low Power Electronics, October 1994.
- [Intel486SX] information can be browsed on: <http://134.134.214.1/design/intarch/prodbref/272713.htm>
- [Lapsley 94] Lapsley, P: "Low power programmable DSP chips: features and system design strategies", Proceedings of the International Conference on Signal Processing, Applications and Technology, 1994.
- [Larri 96] Larri G.: "ARM810: Dancing to the Beat of a Different Drum", Hot Chips 8: A Symposium on High-Performance Chips, Stanford, August 1996.
- [Lorch 95] Lorch, J.R.: "A complete picture of the energy consumption of a portable computer", Masters thesis, Computer Science, University of California at Berkeley, 1995
- [Lorch 96] Lorch, J.R., Smith, A.J.: "Reducing power consumption by improving processor time management in a single user operating system", proceedings of 2nd ACM international conference on mobile computing and networking, Rye, November 1996.
- [Mangione-Smith 96] Mangione-Smith, W., et al.: "A low power architecture for wireless multimedia systems: lessons learned from building a power hog", proceedings of the international symposium on low power electronics and design, Monterey, August, 1996
- [McGaughy 96] McGaughy, B: "Low Power Design Techniques and IRAM", March 20, 1996, information can be browsed on [http://rely.eecs.berkeley.edu:8080/researchers/brucemcg/iram\\_hw2.html](http://rely.eecs.berkeley.edu:8080/researchers/brucemcg/iram_hw2.html)
- [Merkle 93] Merkle, R.C.: "Reversible Electronic Logic Using Switches", Nanotechnology, Volume 4, pp 21 - 40, 1993 (see also: <http://nano.xerox.com/nanotech/electroTextOnly.html>)
- [Moby Dick 95] Mullender S.J., Corsini P., Hartvigsen G. "Moby Dick - The Mobile Digital Companion", LTR 20422, Annex I - Project Programme, December 1995 (see also <http://www.cs.utwente.nl/~havinga/pp.html>)
- [OnNow] "OnNow and ACPI: Introduction and Specifications", available at: <http://www.microsoft.com/hwdev/onnow.htm>
- [Patterson 96] "A Case for Intelligent DRAM: IRAM", Hot Chips 8 A Symposium on High-Performance Chips, information can be browsed on: <http://iram.cs.berkeley.edu/publications.html>
- [Piguet 96] Piguet, C, et al.: "Low-power embedded microprocessor design", proceeding Euromicro-22, pp. 600-605, September 1996.
- [Rabaey 94] Rabaey J. et al.: "Low Power Design of Memory Intensive Functions Case Study: Vector Quantization", IEEE VLSI Signal Processing Conference, 1994.

- [Stemm 96] Stemm, M, et al.: "Reducing power consumption of network interfaces in hand-held devices", proceedings mobile multimedia computing MoMuc-3, Princeton, Sept 1996.
- [Tiwari 94] Tiwari V. et al.: "Compilation Techniques for Low Energy: An Overview", IEEE Symposium on Low Power Electronics, October 1994.
- [Weiser 94] Weiser, M, et al.: "Scheduling for reduced CPU energy", proceedings of the first USE-NIX Symposium on operating systems design and implementation", pp. 13-23, November 1994.
- [Xilinx 95] "Minimizing power consumption in FPGA designs", XCELL 19, page 34, 1995.