

# Design techniques for low power systems

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

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

## *Abstract*

Portable products are being used increasingly. Because these systems are battery powered, reducing power consumption is vital. In this report we give the properties of low power design and techniques to exploit them on the architecture of the system. We focus on: minimizing capacitance, avoiding unnecessary and wasteful activity, and reducing voltage and frequency. We review energy reduction techniques in the architecture and design of a hand-held computer and the wireless communication system, including error control, system decomposition, communication and MAC protocols, and low power short range networks.

**Keywords:** low power, system architecture, mobile computing, wireless communication.

## 1 Introduction

The requirement of portability of hand-held computers and portable devices places severe restrictions on size and power consumption. 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. These devices often require real-time processing capabilities, and thus demand high throughput. *Power consumption is becoming the limiting factor* in the amount of functionality that can be placed in these devices. 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.

### *The advance of technology*

Over the past two decades the semiconductor technology has been continuously improved and has lead to ever smaller dimensions of transistors, higher packaging density, faster circuits, and lower power dissipation. The trend is expected to continue beyond the year 2000. Over the past five years, feature sizes have dropped from about  $f=0.8\mu$  to about  $f=0.35\mu$ . Semiconductor Industry Associates (SIA) have developed a road map for the next few years (see figure 1). It is expected that a feature size of  $f=0.1\mu$  will be reached in 2007 within the context of our current CMOS technology. Such advances provide an effective area increase of about an order of magnitude. To avoid the effect of high electric fields which would be present in very small devices, and to avoid the overheating of the devices, power supply must be scaled down. The power supply voltage is expected to be as low as 1.2 V in 2007.

This rapid advance in technology can be used for several goals. It can be used to increase performance, to add functionality, but also to reduce energy consumption. The current trend is to focus on high performance processors as this is the area in which a semiconductor vendor can enhance its status [5]. Therefore, the architecture of a general purpose processor is most

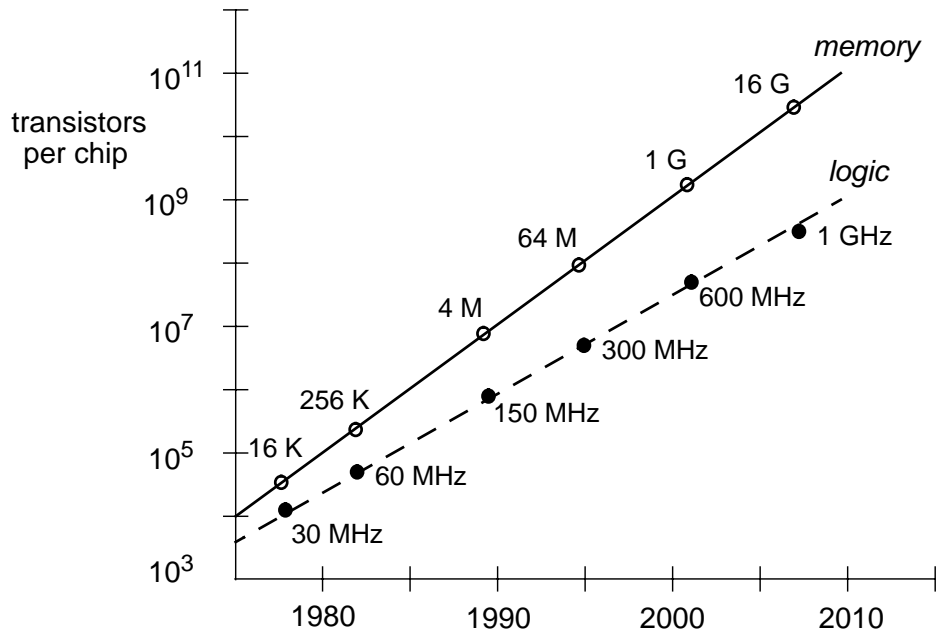


Figure 1: 1994 SIA road map summary

widely studied, and optimizations for *processor performance* is the main goal. The advance in technology has lead to a number of processor improvements like superscalar technology, reduction in cycle time, large on-chip caches, etc.

However, another environment that will rapidly become more important in the near future is that of application specific or embedded processors. The goal of these processors is to optimize the *overall cost-performance* of the system, and not performance alone. The modern application specific processors can use the same technology to *increase functionality* to provide services such as compression and decompression, network access, and security functions. For example, instead of the compromise of including ‘main memory’ on the chip, one could explicitly include dedicated memory space for the functional units. This memory can be used locally for network buffers, video buffers, etc. In this way the data flow in the system is reduced and a bottleneck (access to main memory) is avoided. This is also a good example of how area space can be used to *decrease energy consumption* as by exploiting this locality of reference, the energy consumption of the system is reduced as well. In fact, many energy reduction techniques will show to have the trade-off between chip area and energy consumption.

### Background

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 [12]. 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) [25]. 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 contribution of the CPU and memory to power consumption has been on the rise the last few years [6]. 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.

Note that there is an inherent trade-off between energy consumption and performance, since low-power techniques have associated disadvantages. For instance, decreasing the CPU fre-

quency can raise response time, and spinning down the disk causes a subsequent disk access to have a high latency.

### ***Outline of the paper***

With the increasing integration levels, energy consumption has become one of the critical design parameters. Consequently, much effort has to be put in achieving lower dissipation at all levels of the design process. It was found that, in spite of the progress in the field of low-power computing, most of that progress is in components research: better batteries with more power per unit weight and volume; low power CPUs; very low power radio transceivers; low power displays. But there is very little *systems* research on low power systems. While low-power components and subsystems are essential building blocks for portable systems, we concentrate on dedicated low-power hardware and software architectures. A system wide architecture is beneficial because there are dependencies between subsystems, e.g. optimization of one subsystem may have consequences for the energy consumption of other modules.

There is a vital relationship between hardware architecture, operating system and applications, where each benefits from the others. Therefore, energy reduction techniques have to be applied in all design levels of the system. First of all, we have to use components that use the latest developments in low power technology. Furthermore, as the most effective design decisions derive from the architectural and system level, a cautious design at these levels can reduce the power consumption considerable. However, it is not just a problem of the power-conscious 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.

In this paper we will discuss a variety of energy reduction approaches that can be used for building an energy efficient system. We first explore sources of energy consumption and show the basic techniques used to reduce the power dissipation. Then we give an overview of energy saving mechanisms at the system and architectural level. Finally, we will show as an example techniques used in the Moby Dick project to reduce energy consumption at the architectural and system level.

## **2 Properties of low power design**

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 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.

### **2.1 Design flow**

The design flow of a system constitutes of various levels of abstraction. When a system is designed with the emphasis on power optimization as a performance goal, then the design must embody optimization at all levels of the design flow. In general there are three levels on which energy reduction can be incorporated. The *system level*, the *architecture level*, and the

*technological level*. For example, at the system level inactive modules may be turned off to save power. At the architectural level, parallel hardware may be used to reduce global interconnect and allow a reduction in supply voltage without degrading system throughput. At the technological level several optimisations can be applied at the gate level.

The system and architecture have to be designed targeted to the possible reduction of energy consumption at the gate level. An important aspect of the design flow is the relation and feedback between the levels. Figure 2 shows the general design flow of a system with some examples of where or how energy reduction can be obtained.

<i>abstraction level</i>	<i>examples</i>
system	compression method energy manager scheduling medium access protocols system partitioning
architecture	communication error control parallel hardware hierarchical memories compiler
technological	asynchronous design clock frequency control reducing voltage reduce on-chip routing

Figure 2: General design flow and related examples for energy reduction

Given a design specification, a designer is faced with several different choices on different levels of abstraction. The designer has to select a particular algorithm, design or use an architecture that can be used for it, and determines various parameters such as supply voltage and clock frequency. This multi-dimensional design space offers a large range of possible trade-offs. The most influence on the properties of a design is obtainable at the highest levels. Therefore the most effective design decisions derive from choosing and optimizing architectures and algorithms at the highest levels. It has been demonstrated by several researchers [24] that system and architecture level design decisions can have dramatic impact on power consumption. However, when designing a system it is a problem to predict the consequences and effectiveness of design decisions because implementation details can only be accurately modelled or estimated at the technological level and not at the higher levels of abstraction.

## 2.2 CMOS component model

Most components are fabricated using CMOS technology. The sources of energy consumption on a CMOS chip can be classified as static and dynamic power dissipation. *Static* energy consumption is caused by short circuit currents ( $P_{sc}$ ), bias ( $P_b$ ) and leakage currents ( $P_l$ ). *Dynamic* energy consumption ( $P_d$ ) is caused by the actual effort of the circuit to switch.

$$P = P_d + P_{sc} + P_b + P_l \quad (1)$$

The contributions of this static consumption are mostly determined at the circuit level. During the transition on the input of a CMOS gate both  $p$  and  $n$  channel devices may conduct simultaneously, briefly establishing a short from the supply voltage to ground. This effect causes a power dissipation of approx. 10 to 15%. Also, lower operating voltages as being used nowadays, tend to reduce the short circuit component. While statically-biased gates are usually found in a few specialized circuits such as PLAs, their use has been dramatically reduced in CMOS design [3]. Leakage currents also dissipate static energy, but are also insignificant in most designs (less than 1%).

In general we can say that careful design of gates generally makes their power dissipation typically a small fraction of the dynamic power dissipation, and hence will be omitted in further analysis.

The dominant component of energy consumption (85 to 90%) is CMOS is therefore *dynamic*. A first order approximation of the dynamic power consumption of CMOS circuitry is given by the formula:

$$P_d = C_{eff} V^2 f \quad (2)$$

where  $P_d$  is the power in Watts,  $C_{eff}$  is the effective switch capacitance in Farads,  $V$  is the supply voltage in Volts, and  $f$  is the frequency of operations in Hertz [9]. The power dissipation arises from the charging and discharging of the circuit node capacitances found on the output of every logic gate. Every low-to-high logic transition in a digital circuit incurs a voltage change  $\Delta V$ , drawing energy from the power supply.  $C_{eff}$  combines two factors  $C$ , the capacitance being charged/discharged, and the activity weighting  $\alpha$ , which is the corresponding probability that a transition occurs.

$$C_{eff} = \alpha C \quad (3)$$

A designer at the technological and architectural level can try to minimize the variables in these equations to minimize the overall energy consumption. However, as will be shown in the next sections, power minimization is often a subtle process of adjusting parameters in various trade-offs.

### 2.3 Battery model

A significant amount of work on the development of new batteries has been done in recent years. Batteries have become smaller and they have more capacity. The capacity of the battery is strongly influenced by the available *relaxation time* between current pulses. By taking into consideration the dynamic charge recovery, it is possible for most types of batteries to get more out of a given battery. In [30] the authors studied cylindrical alkaline cells subject to a periodically pulsed current discharge, and found that the cell capacity increases as the duty cycle decreases and the frequency increases. When the system has knowledge of these battery characteristics, the behaviour and energy demands of the system can be adapted such that it tries to discharge the battery only when completely recovered from the previous discharge.

## 3 Reducing power at the technological level

The equations 2 and 3 suggest that there are essentially four ways to reduce power:

- reduce the capacitive load  $C_{eff}$
- reduce the supply voltage  $V$ ,
- reduce the switching frequency  $f$ ,
- reduce the activity  $\alpha$ .

### 3.1 Minimize capacitance

Energy consumption in CMOS circuitry is proportional to capacitance. Therefore a path that can be followed to reduce energy consumption is to *minimize the capacitance*. This can not only be reached at the technological level, but much profit can be gained by an architecture that exploits locality of reference and regularity. Connections to *external components* typically have much greater capacitance than connections to on-chip resources. Therefore, in order to save energy, use few external outputs, and have them switch as infrequently as possible. For example, accessing external memory consumes much energy. So, a way to reduce capacitance is to reduce external accesses and optimize the system by using on-chip resources like caches and registers.

*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, energy reduction can be reached by optimizing the clock frequency of the design even if the resulting performance is far in excess of the requirements [28].

Another way to reduce capacitance is to *reduce chip area*. However, note that a sole reduction in chip area could lead to an energy-inefficient design. For example, a energy efficient architecture that occupies a larger area can reduce the overall energy consumption, e.g. by exploiting locality in a parallel implementation.

### 3.2 Reduce voltage and frequency

One of the most effective ways of energy reduction of a circuit at the technological level is to reduce the supply voltage, because the energy consumption drops quadratically with the supply voltage. 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 low voltage 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 [21]. In other cases, vendors have introduced 'true' low voltage versions of their processors that run at the same speed as their five volt counterparts.

The variables *voltage and frequency* have a trade-off between delay and energy consumption. Reducing clock frequency  $f$  alone does not reduce energy, since to do the same work the system must run longer. As the voltage is reduced, the delay increases. A common approach to power reduction is to first increase the performance of the module - for example by adding parallel hardware -, and then reduce the voltage as much as possible so that the required performance is still reached (figure 3). Therefore, a major theme in many power optimization techniques is to optimize the speed and lower the critical path, so that the voltage can be reduced. However, these techniques often translate in larger area requirements, hence there is a new trade-off between area and power.

Weiser et al. [27] have proposed a system in which the clock frequency 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.

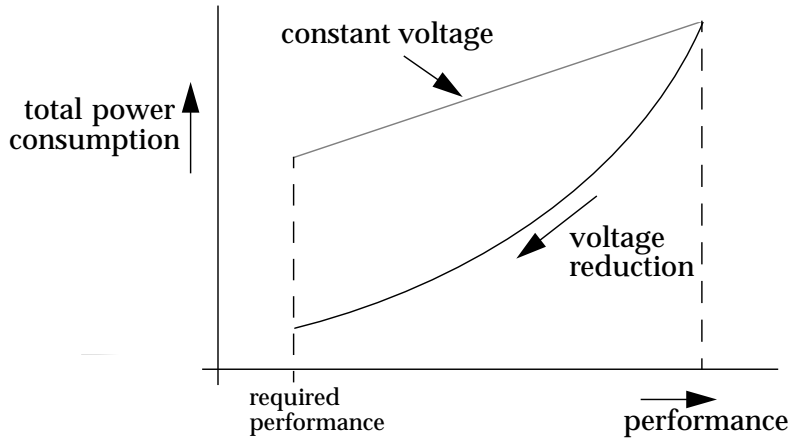


Figure 3: Impact of voltage scaling and performance to total power consumption

### 3.3 Avoid unnecessary activity

The activity weighting  $\alpha$  of equation 3 can be minimized by *avoiding unnecessary and wasteful activity*. There are several techniques to achieve this.

#### **Clock control**

Because CMOS power consumption is proportional to the clock frequency, dynamically turning off the clock to unused logic or peripherals is an obvious way to reduce power consumption [7, 10]. Control can be done at the hardware level or it can be managed by the operating system or the application. Some processors and hardware devices have *sleep or idle modes*. Typically they turn off the clock to all but certain sections to reduce power consumption. While asleep, the device does no work. A wake-up event wakes the device from the sleep mode. Devices 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 device is expected to sleep for a relatively long time.

The technique of dynamically turning off the clock can also be applied to the design of synchronous finite state machines (FSM). For example Koegst et al. [8] use gated clocks in FSM designs to disable the state transition of so called self-loops.

#### **Minimizing transitions**

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 for 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. In this context we can state that a correct choice of the *number representation* can have a large impact on the switching activity. For example, program counters in processors generally use a binary code. On average, two bits are changed for each state transition. Using a Gray code, which will typically result in single bit changes, can give interesting energy savings. However, a Gray code incrementer requires more transistors to implement than a ripple carry incrementer [20]. Therefore a combination can be used in which only the most frequently changing LSB bits use a Gray code.

#### **Asynchronous design**

Another way to avoid unnecessary activity is by applying an *asynchronous design methodology*. 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 that must distribute a clock signal evenly to all parts of a circuit, and it must switch all the time to provide the timing reference even if only a small part of the chip has something useful to do. A synchronous circuit therefore wastes power when particular blocks of logic are not utilized, for example, to a floating point unit when integer arithmetic is being performed.

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, extra logic is required for synchronization, so asynchronous circuits are larger than synchronous circuits.

### **Reversible logic**

*Reversible logic* [17] 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.

## **4 Low power system level design**

In the previous section we have explored sources of energy consumption and showed the low level design techniques used to reduce the power dissipation. In this section we will concentrate on these techniques at *system level* and the relevance for low power system design.

The two main themes that can be used for energy reduction at system level are:

- avoid unnecessary activity, and
- exploit locality of reference.

### **4.1 Hardware system architecture**

The implementation dependent part of the power consumption of a system is strongly related to a number of properties that a given system or algorithm may have [22]. The component that contributes a significant amount of the total energy consumption is the *interconnect*. Experiments have demonstrated that in designs, about 10 to 40% of the total power may be dissipated in buses, multiplexers and drivers. This amount can increase dramatically for systems with multiple chips due to large off-chip bus capacitance. The power consumption of the interconnect is highly dependent on algorithm and architecture-level design decisions. Two properties of algorithms are important for reducing interconnect power consumption: locality and regularity.

*Locality* relates to the degree to which a system or algorithm has natural isolated clusters of operation or storage with a few interconnections between them. Partitioning the system or algorithm into spatially local clusters ensures that the majority of the data transfers take place within the clusters and relatively few between clusters. The result is that the local buses are shorter and more frequently used than the longer highly capacitive global buses. Locality of



reference can be used to partition memories. Current high level synthesis tools are targeted to area minimization. For power reduction, however, it is better to minimize the number of accesses to long global buses and have the local buses be accessed more frequently. In a direct implementation targeted at area optimization, hardware sharing between operations might occur, destroying the locality of computation. An architecture and implementation should preserve the locality and partition and implement it such that hardware sharing is limited. The increase in the number of functional units does not necessarily translate into a corresponding increase in the overall area and energy consumption since (1) localization of interconnect allows a more compact layout and (2) fewer (access to) multiplexers and buffers are needed.

*Regularity* in an algorithm refers to the repeated occurrence of computational patterns. Common patterns enable the design of less complex architecture and therefore simpler interconnect structure (buses, multiplexers, buffers) and less control hardware. These techniques have been exploited by several researchers (e.g. Mehra [16] and Rabaey [22]), but mainly in the DSP domain where a large set of applications inherently have a high degree of regularity.

We will now show two mechanisms that exploit locality of reference to reduce energy consumption.

#### ***A. Application specific modules***

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. Another way to reduce data traffic is to integrate a processor in the memory, as for example proposed by Patterson in intelligent RAM [19, 15].

At system level locality can be applied to divide the functionality of the system into dedicated modules [1]. When the system is decomposed out of application-specific coprocessors the data traffic can be reduced, 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.

Furthermore, 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.

By careful repartitioning a system, not only the power consumption can be reduced but the performance is actually improved as well [14].

#### ***B. 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). As most of the time only a small memory is read, the energy consumption is reduced.

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 requires less data traffic. Furthermore, the energy consumption is reduced because less disk and network activity is required.

The compiler can have impact on power consumption by reducing the number of instructions with memory operands. The most energy can be saved by a proper utilization of registers [26]. 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.

When the memory is divided into several small blocks that individually can be powered down, then the memory allocation strategy and garbage collector of the operating system can take benefit of this by allocating the memory being used in clustered memory blocks, such that memory that is not used is not spread around all memory banks.

## 4.2 Energy reduction in communication

The wireless network interface of a mobile computer consumes a significant fraction of the total power [25]. 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.

The concept of Quality of Service will be an important issue for the management of multimedia traffic in which delay, jitter, latency, packet loss, etc. are important parameters. When energy constraints are taken into account, then classic network protocols might perform badly.

There are a number of energy reduction techniques possible in the architecture of a wireless communication system. A careful design of *all* network layers is required. There are several ways to achieve a reduction in energy consumption here: e.g. by applying dedicated wireless error control, by system decomposition, by using hybrid networking with low power short range networks, and by applying power aware MAC protocols.

### A. Error control

Wireless networks have a much higher error rate than the normal wired networks. The errors that occur on the physical channel are caused by phenomena like multipath, fading and user mobility. The most relevant errors that occur are related to *block errors* that have a bursty nature. Packet switching communication schemes will likely be the basis for most currently designed wireless systems. Therefore most errors will occur as packet errors.

Error correcting codes like FEC (Forward Error Correction) are mainly used at the data link layer to reduce the impact of errors in the wireless connection. In most cases, these codes provide less than perfect protection and some amount of residual errors pass through. Higher level protocol layers employ various block error detection and retransmission schemes such as Go-Back-N and Selective Repeat.

In the design of error control mechanisms, the current issues are mainly complexity, buffering requirements, throughput and delay. When energy constraints are taken into account, then these classic ARQ (Automatic Repeat reQuest) protocols, perform badly because these protocols keep retransmitting and spending energy. Alternative schemes, that try to avoid transmitting during bursty error conditions, might perform more energy efficient with a slight loss

in throughput [31].

Forward Error Correction (FEC) can be used to improve the performance and to reduce energy consumption not only at the data link level, but also in higher levels of the protocol stack [23]. In particular, FEC can be beneficial for reliable multicast communication, because it reduces the number of acknowledgements. Current error correction codes perform well when the errors are independent, whereas their efficiency is usually less in the presence of bursty errors.

Errors on the wireless link can be propagated in the protocol stack. In the presence of a high packet error rate and periods of intermittent connectivity characteristics of wireless links, some network protocols (such as TCP) may overreact to packet losses, mistaking them for congestion. TCP responds to all losses by invoking congestion control and avoidance algorithms. These measures result in an unnecessary reduction in the link's bandwidth utilization and increases in energy consumption because it leads to a longer transfer time. The limitations of TCP can be overcome by a more adequate congestion control during packet errors. These schemes choose from a variety of mechanisms to improve end-to-end throughput, such as local retransmissions, split connections and forward error correction. In [2] several schemes have been examined and compared. These schemes are classified into three categories: *end-to-end* protocols, where the sender is aware of the wireless link; *link-layer* protocols, that provide local reliability and shields the sender from wireless losses; and *split-connection* protocols, that break the end-to-end connection into two parts at the base station. Their results show that a reliable link-layer protocol with some knowledge of TCP provides good performance, more than using a split-connection approach. Selective acknowledgement schemes are useful, especially when the losses occur in bursts.

### **B. System decomposition**

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

In a communication system locality of reference can be exploited by decomposition of the network protocol stack and cautious management of the data flow. This can reduce the energy consumption for several reasons:

- First, when the system is constructed out of independent components that implement different layers of the communication stack, unnecessary data copies between successive layers of the protocol stack are eliminated. This eliminates wasteful data transfers over the global (!) bus, and thus saves much dissipation in buses, multiplexers and drivers.
- Secondly, dedicated hardware can do basic signal processing and can move merely the necessary data directly to its destination, thus keeping data copies off of the system bus. Moreover, this dedicated hardware might do its tasks much more energy efficient than a general purpose processor.
- 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.

This decomposition can also be applied beyond the system level of the portable: certain functions of the system can be migrated from the portable system to a remote server that has plenty of energy resources. This remote server handles those functions that can not be handled efficiently on the portable machine. For example, a base station could handle parts of the network protocol stack in lieu of the mobile. The remote server has a private dedicated com-

munication protocol with the mobile so that the mobile units can use an internal, light weight, protocol to communicate with the base station rather than TCP/IP or UDP. The net result is saving in code and energy. In such a system it is also efficient to adapt the protocols for the specific environment it is used in. For example, as described in the previous section wireless networks have a much higher *error rate* than the normal wired networks.

In order to save energy a normal mode of operation of the mobile will be a sleep or power down mode. To support full connectivity while being in a deep power down mode the network protocols need to be modified. 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.

In the higher level protocols of a communication system caching and scheduling is used to control the transmission of messages. In a situation with varying and multiple network connectivity it may be wise to prefetch some information or postpone the actual transmission until the quality of the connection is better, or until another, more power economic, network is available. An application can for example schedule the 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.

### ***C. Low power short range networks***

Portable computers need to be able to move seamlessly from one communication medium to another, for example from a GSM network to an in-door network, without rebooting or restarting applications. Applications require that networks are able to determine that the mobile has moved from one network to another network with a possible different QoS. The network that is most appropriate in a certain location at a certain time depends on the user requirements, network bandwidth, communication costs, energy consumption etc. The system and the applications might adapt to the *cost* of communication (e.g. measured in terms of ampère-hours or telephone bills).

Over short distances, typically of up to five metres, high-speed, low-energy communication is possible. Private houses, office buildings and public buildings can be fitted with ‘micro-cellular’ networks with a small antenna in every room at regular intervals, so that a mobile computer never has to communicate over a great distance — thus saving energy — and so that the bandwidth available in the aether does not have to be shared with large numbers of other devices — thus providing high aggregate bandwidth. Over large distances (kilometres rather than metres), the mobile can make use of the standard infrastructures for digital telephony (such as GSM).

### ***D. Energy aware MAC protocol***

Current wireless networks do not pay much attention to the energy consumption, but mainly focus on maximizing the throughput. In a wireless system the medium access protocols can be adapted and tuned to enhance energy efficiency. An example of an energy aware MAC protocol is LPMAC [14]. The basic objective is that the protocol tries to minimize all actions of the network interface, i.e. minimize ‘on-time’ of the transmitter as well as the receiver. In a power aware TDMA protocol that coordinates the delivery of data to receivers is a base station responsible for *traffic scheduling* [11]. The base station dictates a frame structure within its range. A frame consists of a number of data-cells and a traffic control cell. Mobiles with scheduled traffic are indicated in a list, which allows mobiles without traffic to rapidly reduce power (see Figure 4). The traffic control is transmitted by a base station and contains the infor-

mation about the subsequent data-cells, including when the next traffic control cell will be transmitted.

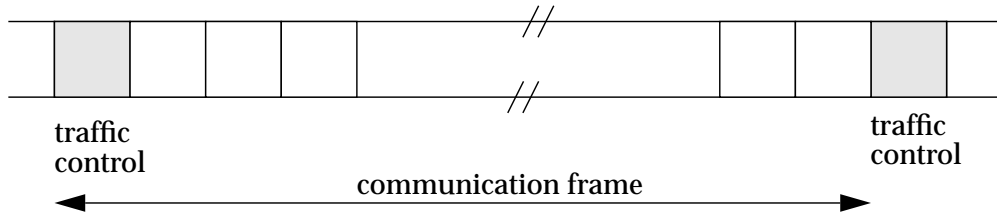


Figure 4: example of a TDMA frame structure

As switching between states (i.e. off, idle, receiving, transmitting) consumes time and energy, the number of state-transitions have to be minimized. By *scheduling bulk data transfers*, an inactive terminal is allowed to doze and power off the receiver as long as the network interface is reactivated at the scheduled time to transceive the data at full speed.

The overhead and energy consumption involved in the frame mechanism with traffic control, can be reduced when the frame size can be adapted to the situation. For example in case of a room in which only one mobile communicates, the frame size can be increased. There is a trade-off between frame size and the latency. When a low latency is required the frame size can be adapted accordingly.

Some applications require a distribution of information to a group of users. A *multicast* or *broadcast* mechanism can reduce energy consumption since the information is sent only once, and the base station - with plenty of energy - disseminates the information. Notice that the performance (total aggregated throughput) is increased as well. The network and MAC protocols need to be adapted for these mechanisms.

Another way to increase the energy efficiency is that the MAC protocol has a strategy that tries to avoid transmission during bad channel periods. In this way useless transmissions are avoided.

### 4.3 Operating system and applications

In this section we will show several approaches that can be applied at the operating system level and to the applications to reduce energy consumption.

#### A. Scheduling

In a system *scheduling* is needed when multiple functional units need to access the same object. Scheduling is used by the operating system to provide each unit a share of the object in time. Scheduling is applied at several parts of a system for processor time, communication, disk access, etc. Currently scheduling is performed on criteria like priority, latency, time requirements etc. Power consumption is in general only a minor criterion for scheduling, despite the fact that much energy could be saved.

We will now show several possible mechanisms in which an energy aware scheduling can be beneficial.

- *Processor time scheduling*

Most systems spend only a fraction of the time performing useful computation. The rest of the time is spent idling. The operating systems energy manager should track the periods of computation, so that when an idle period is entered, it can immediately power off major parts of the system that are no longer needed [3]. Since all power-down approaches incur some overhead, the task of an energy aware scheduler can be to collect requests for compu-

tation and compact the active time-slots into bursts of computation.

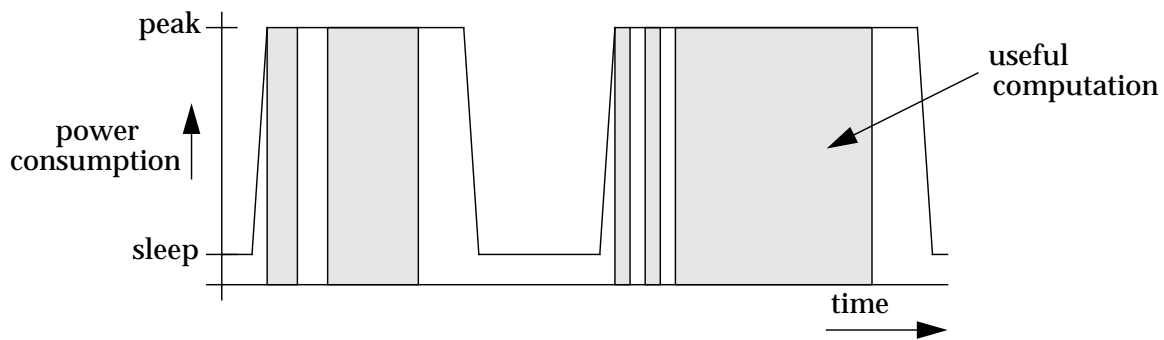


Figure 5: Power consumption in time of a typical processor system.

Weiser et al. [27] 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. For background and high latency tolerable tasks, the supply voltage can be reduced so that just enough throughput is delivered, which minimizes energy consumption. 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. 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 then be 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.

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.

- *File system*

The file system is another issue in the interaction between hardware facilities for power management and the system software. It is one thing to turn off a disk when it has been idle for a minute, but it is much better to design a file system in such a way that it takes advantage of the possibility of turning the disk off. For example the operating system’s file system a scheduler can try to collect disk operations in a cache and postpone low priority disk I/O only until the hard drive is running already or has enough data.

- *Communication*

The MAC protocol of a wireless system can be tuned in such a way that it tries to minimize state-transitions and have bulk data-transfer (see for more information the section about the power aware MAC protocol). In the higher level protocols of a communication system scheduling is used to control the transmission of messages. 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 con-

sumes less energy or does not need its batteries.

- **Battery relaxation**

The capacity of batteries is strongly influenced by the available relaxation time between current pulses. The relationship between how much of the battery capacity is recovered during an off period depends on the cell chemistry and geometry. A scheduler could try to find the optimum between required energy and available time.

### ***B. Energy manager***

Power down of unused modules is a commonly employed approach for energy reduction. To take advantage of low-power states of devices, either the operating system needs to direct (part of) the device 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, or the modules use a demand- or data-driven computation to automatically eliminate switching activity of unused modules. The device or system will enter the sleeping state when it is idle or when the user indicates to do so.

The division of the system into modules must be such that the modules provide a *clustered functionality*. For example, in the memory system, locality of reference can be exploited during memory assignment to induce an efficient and effective power down of large blocks of memory. This shows once again that a close cooperation between the operating system that performs the memory allocation, the energy manager that controls the power states of the devices, together with a suitable hardware architecture can be beneficial for energy reduction of the system.

In order to control the modules, changes must be made to current architectures 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 the system.

### ***C. 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 [13]). So, it is not sufficient to have the system to be low power, but the applications running on the system have to be written energy aware as well.

### ***D. Code and algorithm transformation***

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. However, it was found not to have a great impact [26].

At the algorithm level functional pipelining, retiming, algebraic transformations and loop transformations can be used [16]. The system essential power dissipation can be estimated by a weighted sum of the number of operations in the algorithm that has to be performed [4]. The weights used for the different operations should reflect the respective capacitance switched. The size and the complexity of an algorithm (e.g. operation counts, word length) determine the activity. Operand reduction includes common sub-expression elimination, dead code elimination etc. Strength reduction can be applied to replace energy consuming operations by a combination of simpler operations (for example by replacing multiplications into shift and add operations). Drawbacks of this approach are that it introduces extra overhead for registers and control, and that it may increase the critical path.

## 5 A case study: energy reduction in the Moby Dick project

The Moby Dick project [18] 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, called *Pocket Companion*. The design challenges lie primarily in the creation of a single architecture that allows the integration of security functions, externally offered services, personal-ity, and communication. Research issues include: security, energy consumption and communication, hybrid networks, data consistency, and environment awareness. In this section we will elaborate on the techniques used in the Moby Dick project for reducing energy consumption.

To support multimedia functionality for the intended applications of the Pocket Companion the system needs to have real-time properties. The increasing levels of performance and integration that is required will be accompanied by increasing levels of energy consumption. Without significant energy reduction techniques and energy saving architectures, battery life constraints will limit the capabilities of a Pocket Companion. More extensive and continuous use of network services will only aggravate this problem since communication consumes relatively much energy.

### 5.1 The Pocket Companion's system architecture

The goal of the Pocket Companion's architecture is to optimize the *overall energy-performance* of the system, and not performance alone. The technology is used to *decrease energy consumption* and to *increase functionality* to provide services such as multimedia devices, compression and decompression, network access, and security functions.

The difficulty in achieving all requirements into one architecture stems from the inherent trade-offs between flexibility and energy consumption, and also between performance and energy consumption. Flexibility requires generalized computation and communication structures, that can be used to implement different kinds of algorithms. For multimedia applications in particular there is a substantial reduction in energy consumption possible as the computational complexity is high, they have a regular and spatially local computation, and the communication between modules is significant. Improving the energy efficiency by exploiting *locality of reference* and using *efficient application-specific modules* therefore has a substantial impact on a system like the Pocket Companion.

While conventional architectures (like used in current laptops) can be programmed to



perform virtually any computational task, they achieve this at the cost of high energy consumption. The Pocket Companion has a rather unconventional architecture that saves energy by using system decomposition at different levels of the architecture and exploiting locality of reference with dedicated, optimized modules.

Our approach is based on dedicated functionality and the extensive use of power reduction techniques at all levels of system design. The system has a number of premises:

- An architecture with a general purpose processor surrounded by a set of *heterogeneous programmable modules*, each providing an energy efficient implementation of dedicated tasks. Application-specific coprocessors might be able to perform their tasks more efficient - in terms of performance and/or energy consumption - than a general purpose processor. For example, even when the application-specific coprocessor consumes more power than the processor, it may accomplish the same task in far less time, resulting in a net energy saving.
- A *reconfigurable internal communication network* that exploits locality of reference and eliminates wasteful data copies. When the system is decomposed out of application-specific coprocessors the traffic on the bus is reduced by eliminating unnecessary data copies. For example, in a system where a stream of video data is to be displayed on a screen, the data can be copied directly from the network into the screen memory, without going through the main processor.
- A system design that avoids unnecessary activity: e.g. by the use of *autonomous modules* that can be powered down individually and are data driven. Each module can be optimized - apart for its main task - for energy consumption. It can have a separate clock, voltage control, power-down modes, and dedicated features to save energy.
- A wireless communication system designed for low energy consumption by the use of *intelligent network interfaces* that deal efficiently with a mobile environment, by using a power aware network protocol, by using an energy efficient MAC protocol, and by energy aware error control mechanisms.

In the Pocket Companion's architecture we distinguish a *switch* surrounded by several modules (see Figure 6). All modules in the system communicate over a communication

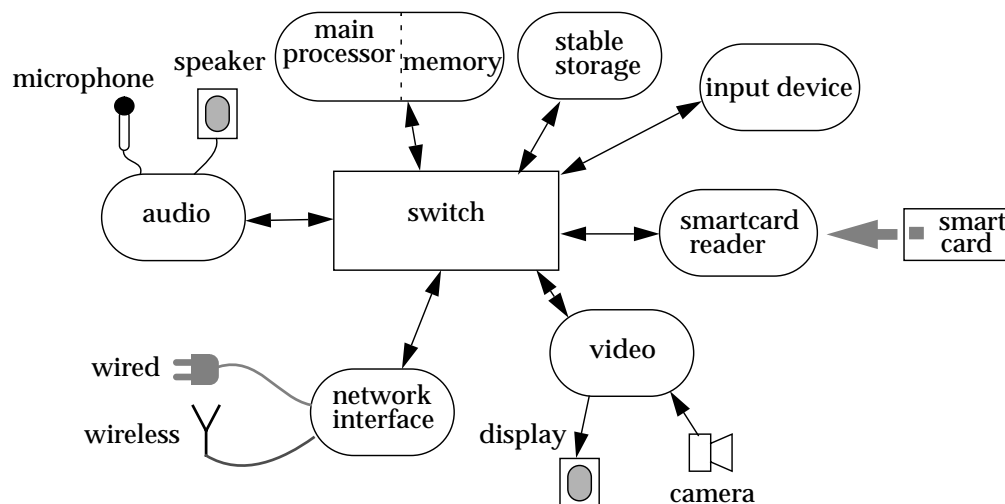


Figure 6: Pocket Companion system architecture

network. The switch interconnects the modules and provides a reliable path for communication between modules. As in switching networks, the use of a multi-path topology will enable parallel data flows between different pairs of modules and thus will increase the performance. Modules are autonomous and can communicate without involvement of the

main processor. Each module has its own functionality and responsibility. The modules are optimized for their function in terms of performance and energy consumption and have their own *local power management*.

## 5.2 Testbed for the Pocket Companion

Currently we are designing and building a testbed for the Pocket Companion using existing hardware components and subsystems. The hardware basis of the Pocket Companion is an interconnection switch. All data in the system is based on the size of an ATM cell (48 bytes data). This not only allows us to easily interconnect with an ATM environment, but the size is also adequate: it is small enough to buffer several cells in the switch and have small latency, and large enough to keep the overhead small.

The prototype of the switch is build using a Xilinx FPGA and six small micro-controllers. The switch is capable to connect six modules. It is a tiny ATM switch as it is able to transfer ATM cells to a destination according to its VCI (Virtual Channel Identifier). The basic functions of the micro-controllers is to perform routing, to establish a connection and to interface with the connected modules. The design methodology used is data-driven and based on asynchronous methods, combined with synchronous parts. Each part of the switch that is not used at some time does not receive any data and clock changes, thus minimizing energy consumption.

The attached modules can be similar to today's PDAs, notebook computers or 'handheld PCs' augmented with a security module and one or several wireless network devices. Currently we are using the WaveLAN modem as network interface to experiment with MAC protocols and error correction. With this testbed we can easily evaluate (parts of) the architecture concerning energy consumption, security, Quality of Service and communication. We gradually improve the architecture to the final architecture.

We also experiment with system software, as hardware architecture and system software are related. The operating system *Inferno* from Lucent Technologies [7] is quite well suited for this purpose.

## 6 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. System designers can decrease energy consumption at several levels of abstraction. At technological and architectural level energy consumption can be decreased by reducing the supply voltage, reducing the capacitive load and by reducing the switching frequency. Much profit can be gained by avoiding unnecessary activity at both the architectural as system level. At system level, they can take advantage of power management features where available, as well as decomposed system architectures and programming techniques for reducing power consumption.

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. These techniques often lead to less performance,

that only can be improved by adding more hardware. Most energy efficient systems use more area, not only to implement the new data flow or storage, but also to implement the control part. Furthermore, energy efficient systems can be more complex. Another consequence is that although the application specific coprocessor approach is more efficient than a general purpose processor, it is less flexible. Furthermore, 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 guide the power management.

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

## 7 References

- [1] Abnous A, Rabaey J.: "Ultra-Low-Power Domain-Specific Multimedia Processors," *Proceedings of the IEEE VLSI Signal Processing Workshop*, San Francisco, October 1996.
- [2] Balakrishnan H., et al.: "A comparison of mechanisms for improving TCP performance over wireless links", *Proc. ACM SIGCOMM'96*, Stanford, CA, USA, August 1996.
- [3] Burd T.D., Brodersen R.W.: "Energy efficient CMOS microprocessor design", *Proceedings 28th. annual HICSS Conference*, Jan. 1995, vol. I, pp 288-297.
- [4] Chandrakasan A.P., et al.: "Optimizing power using transformations", *Transactions on CAD*, Jan. 1995.
- [5] Flynn M.J.: "What's ahead in computer design?", *Proceedings Euromicro 97*, pp 4-9, September 1997.
- [6] Ikeda T.: "ThinkPad Low-Power Evolution", *IEEE Symposium on Low Power Electronics*, October 1994.
- [7] Intel486SX: information can be browsed on: <http://134.134.214.1/design/intarch/prodbref/272713.htm>
- [8] Koegst, M, et al.: "Low power design of FSMs by state assignment and disabling self-loops", *Proceedings Euromicro 97*, pp 323-330, September 1997.
- [9] Lapsley, P: "Low power programmable DSP chips: features and system design strategies", *Proceedings of the International Conference on Signal Processing, Applications and Technology*, 1994.
- [10] Larri G.: "ARM810: Dancing to the Beat of a Different Drum", *Hot Chips 8: A Symposium on High-Performance Chips*, Stanford, August 1996.
- [11] Linnenbank, G.R.J. et al.: "A request-TDMA multiple-access scheme for wireless multimedia networks", *Proceedings Third Workshop on Mobile Multimedia Communications (MoMuC-3)*, 1996.
- [12] Lorch, J.R.: "A complete picture of the energy consumption of a portable computer", Masters thesis, Computer Science, University of California at Berkeley, 1995

- [13] 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.
- [14] 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
- [15] 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)
- [16] Mehra R., Rabaey J.: "Exploiting regularity for low power design", *Proceedings of the International Conference on Computer-Aided Design*, 1996
- [17] Merkle, R.C.: "Reversible Electronic Logic Using Switches", *Nanotechnology*, Volume 4, pp 21 - 40, 1993 (see also: <http://nano.xerox.com/nanotech/electroTextOnly.html>)
- [18] 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>)
- [19] "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>
- [20] Piguet, C, et al.: "Low-power embedded microprocessor design", *Proceeding Euromicro-22*, pp. 600-605, September 1996.
- [21] Rabaey J. et al.: "Low Power Design of Memory Intensive Functions Case Study: Vector Quantization", *IEEE VLSI Signal Processing Conference*, 1994.
- [22] Rabaey J., Guerra L., Mehra R.: "Design guidance in the Power Dimension", *Proceedings of the ICASSP*, 1995.
- [23] L.Rizzo: "Effective Erasure Codes for Reliable Computer Communication Protocols", *ACM Computer Communication Review*, Vol. 27- 2, pp 24-36, April 97
- [24] Sheng S., Chandrakasan C., Brodersen R.W.: "A portable multimedia terminal", *IEEE Communications Magazine*, December 1992, pp 64-75
- [25] Stemm, M, et al.: "Reducing power consumption of network interfaces in hand-held devices", *Proceedings mobile multimedia computing MoMuc-3*, Princeton, Sept 1996.
- [26] Tiwari V. et al.: "Compilation Techniques for Low Energy: An Overview", *IEEE Symposium on Low Power Electronics*, October 1994.
- [27] Weiser, M, et al.: "Scheduling for reduced CPU energy", *proceedings of the first USENIX Symposium on operating systems design and implementation*, pp. 13-23, November 1994.
- [28] "Minimizing power consumption in FPGA designs", *XCELL* 19, page 34, 1995.
- [29] Smit, G.J.M., et al.: "Design and realization of a receiver for wireless ATM", submitted for publication.
- [30] Podlaha, E.J., Cheh, H.Y.: "Modeling of cylindrical alkaline cells. VI: variable discharge conditions", *Journal of Electrochemical Society*, vol 141, pp. 28-35, Jan. 1994.
- [31] Zorzi, M., Rao, R.R.: "Error control and energy consumption in communications for nomadic computing", *IEEE transactions on computers*, Vol 46, pp. 279-289, March 1997.