2

Design techniques for energy efficient and low-power systems

Portable systems are being used increasingly. Because these systems are battery powered, reducing energy consumption is vital. In this chapter we give an overview of low-power design and provide a review of techniques to exploit them in the architecture of the system. We focus on: minimising capacitance, avoiding unnecessary and wasteful activity, and reducing voltage and frequency. We review energy reduction techniques with applications in the architecture and design of a hand-held computer including its wireless communication system.

2.1 Introduction

The portability requirement of hand-held computers and other 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 policies for careful management of the energy consumption while still providing the appearance of continuous connections to system services and applications. In this chapter[1] we will explore sources of energy consumption and provide a variety of energy reduction techniques at various levels in the design flow of a computer system. We will

[1] Major parts of this chapter will be published in the *Journal of Systems Architecture*, 2000 [25] and were presented at the *IEEE International Conference on Personal Wireless Communications* (ICPWC'97), 1997 [24].

try to point out the main driving forces in current research. This provides the foundation of the techniques we have applied in the design of the *Mobile Digital Companion* that is topic of the research presented in this thesis.

2.1.1 The advance of technology

The semiconductor technology has continuously improved and has lead to ever smaller dimensions of transistors, higher packaging density, faster circuits, and lower power dissipation. Over a three year period from 1998 to 2001 there will be a factor 100 increase in 3D graphics performance and nearly a factor 10 increase in hard disk capacity – far outstripping Moore's law [81]. The bandwidth of wireless networks has doubled every six months. Significant new features are being added. Video capturing, for example, is becoming a mainstream feature with MPEG-2 video encoding and decoding available on low-cost video adapters. These dramatic improvements are occurring even as the cost of computing for the average user is quickly dropping. This has been possible due to the use of parallel hardware, on-chip memory (RAM), new algorithms, and the increased level of integration of IC technology. Over the past five years, feature sizes have dropped from about 0.8μm to about 0.35μm. Semiconductor Industry Associates (SIA) have developed a road map for the next few years [62]. It is expected that a feature size of 0.1μm 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 is 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 0.9 V in 2007.

The rapid advance in technology can be used for several purposes. It can be used to increase performance, to add functionality, but also to reduce energy consumption. One way to use this opportunity would be to continue advancing our chip architectures and technologies as just more of the same: building microprocessors that are simply more complicated versions of the kind built today. For more than thirty years, performance optimisation has been extensively studied at all abstraction levels. The current trend in industry is to focus on high-performance processors, as this is the area in which a semiconductor vendor can enhance status [10]. Therefore, the architecture of a general-purpose processor is most widely studied, and optimisations for processor performance is the main goal. Technology innovation has lead to a number of processor improvements like superscalar technology, multi-level pipelines, large on-chip caches, etc.

Another environment that will become more important in the near future is that of application specific or embedded processors. The goal of these processors is to optimise the overall cost-performance of the system, and not performance alone. The modern application-specific processors can use the novel technology to increase functionality such as compression and decompression, network access, and security functions.

Energy consumption

Power consumption has become a major concern because of the ever-increasing density of solid-state electronic devices, coupled with an increasing use of mobile computers and portable communication devices. The technology has thus far helped to build low-power systems. The speed-power efficiency has indeed gone up since 1990 by 10 times each 2.5 years for general-purpose processors and digital signal processors (DSPs). Table 1 shows the performance and power consumption of some recent processors [71]. However, this help will slow down, because physical limits seem to be reached soon.

Processor	MHz	Year	SPECint-95	Watts	Watts/SPECint
P54VRT (Mobile)	150	1996	4.6	3.8	0.83
P55VRT (Mobile MMX)	233	1997	7.1	3.9	0.55
PowerPC 603e	300	1997	7.4	3.5	0.47
PowerPC 740 (G3)	300	1998	12.2	3.4	0.28
Mobile Celeron	333	1999	13.1	8.6	0.65

Table 1: Speed and power characteristics of some recent processors.

Design for low-energy consumption is certainly not a new research field, and yet remains one of the most difficult as future mobile system designers attempt to pack more capabilities such as multimedia processing and high bandwidth radios into battery operated portable miniature packages. Playing times of only a few hours for personal audio, notebooks, and cordless phones are clearly not very consumer friendly. Also, the required batteries are voluminous and heavy, often leading to bulky and unappealing products. The primary problem is that in the case of battery technology, there is no equivalent of Moore's Law which forecasts a doubling of the complexity of microelectronic chips every 18 months, and Gilder's Law, which theorises a similar exponential growth in communication bandwidth. In contrast, battery technology has improved very slowly, and only a 20% improvement in capacity is expected over the next 10 years [63]. These trends are depicted in Figure 1 [71].

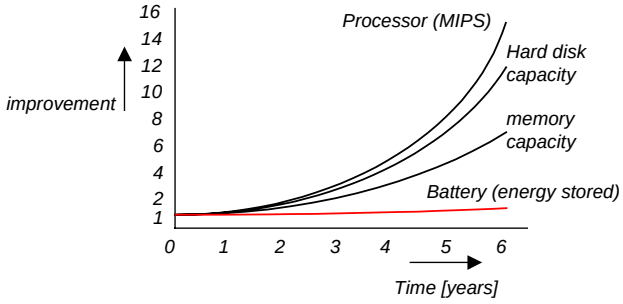


Figure 1: Improvement in technology.

With increasing computation and communication functions desired for wireless mobile systems, the energy density of existing battery technologies are far from what is needed. Table 2 shows the energetic potentials of current battery technology.

Battery	rechargeable	Wh/kg	Wk/litre
Alkaline MnO2	no	130	347
Li/MnO2	no	210	550
Zinc Air	no	280	1150
Lead acid	yes	30	80
Nickel-Cadmium NiCd	yes	40	130
Nickel-metal hybride NiMH	yes	60	200
Lithium-ion	yes	60	200
Methanol fuel cell	yes	6200	4900

Table 2: The energetic potentials of batteries.

The most recent advances in laptop batteries are in the form of better ‘fuel gauging’ of the battery, to give a more precise measure of the charge level and to estimate the time left before a recharge is needed [84]. Although this is a useful technique, it does not extend battery life.

A promising technique might be *fuel cells*. A fuel cell is an electrochemical device that converts the chemical energy of a fuel directly to usable energy – electricity and heat – without combustion. The energetic potential is very high, a fuel cell running on methanol could provide power for more than 20 times longer than traditional nickel cadmium batteries in a comparably sized package [15]. They are theoretically quiet and clean like normal batteries. Another benefit is that fuel cells do not require lengthy recharching; they can instead be replenished quickly, simply by adding more fuel. Fuel cells were once prohibitively expensive. But sophisticated engineering has recently

driven cost down considerably. However, designing a miniature fuel cell that can be mass-produced cheaply is a formidable task. Although initial prototypes have been built, no one has yet demonstrated a compact device that could be mass-produced at a cost lower than that of comparable rechargeable batteries.

Several researchers have studied the power consumption pattern of mobile computers. 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. However, because they studied different platforms, their results are not always in agreement, 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 [36]. 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) [73]. If the computer can 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 [27].

Even though it is difficult to compare these results because the measurements are made for different architectures, operating systems, communication interfaces, and benchmarks, there is a common pattern: there is no primary source of energy consumption. The energy spent is distributed over several devices and for several operations. The conclusion is that implementing an energy efficient system involves looking at all the functions in the system, and not just a single function such as for example, network protocol processing.

2.1.2 Outline

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 most low-power research is concentrated on components research: better batteries with more power per unit weight and volume; low-power CPUs; very low-power radio transceivers; low-power displays. We found that there is very little systems research on low-power systems. While these low-level circuit and logic techniques have been well established for improving energy efficiency, they do not hold promise for much additional gain. While low-power components and subsystems are essential building blocks for portable systems, a system-wide architecture that incorporates the low-power vision into all layers of the system is beneficial because there are dependencies between subsystems, e.g. optimisation of one subsystem may have consequences for the energy consumption of other modules.

The key to energy efficiency in future mobile systems will be designing higher layers of the mobile system, their system architecture, their functionality, their operating system, and indeed the entire network, with energy efficiency in mind. 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 chapter we will discuss a variety of energy reduction approaches that can be used for building an energy-efficient system. We have no intention to give an exhaustive overview of existing methodologies and tools for low-power systems, but try to point out the main driving forces in current research. 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.

2.2 Fundamentals of low-power design

Throughout this chapter, 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 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's may or may not consume more energy to run 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.

Therefore, we introduce a metric: *energy efficiency*. We define the energy efficiency *e* as the energy dissipation that is essentially needed to perform a certain function, divided by the actually used total energy dissipation.

$$e = \frac{\textit{Essential energy dissipation for a certain function}}{\textit{Actually used total energy dissipation}} \qquad (1)$$

The function to be performed can be very broad: it can be a limited function like a multiply-add operation, but it can also be the complete functionality of a network protocol. Let us for example consider a medium access (MAC) protocol that controls access to a wireless channel. The essential energy dissipation is the energy dissipation needed to transfer a certain amount of bits over the wireless channel, and the total energy dissipation also includes the overhead involved in additional packet headers, error control, etc.

Note that the energy efficiency of a certain function is independent from the actual implementation, and thus independent from the issue whether an implementation is low-power. It is possible to have two implementations of a certain function that are built with different building blocks, of which one has a high energy efficiency, but dissipates more energy than the other implementation which has a lower energy efficiency, but is built with low-power components.

2.2.1 Design flow

The design flow of a system constitutes various levels of abstraction. When a system is designed with an emphasis on power optimisation as a performance goal, then the design must embody optimisation at all levels of the design. In general there are three main levels on which energy reduction can be incorporated. The *system* level, the *logic* level, and the *technological* level. For example, at the system level power management can be used to turn off inactive modules to save power, and parallel hardware may be used to reduce global interconnect and allow a reduction in supply voltage without degrading system throughput. At the logic level asynchronous design techniques can be used. At the technological level several optimisations can be applied to chip layout, packaging and voltage reduction.

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

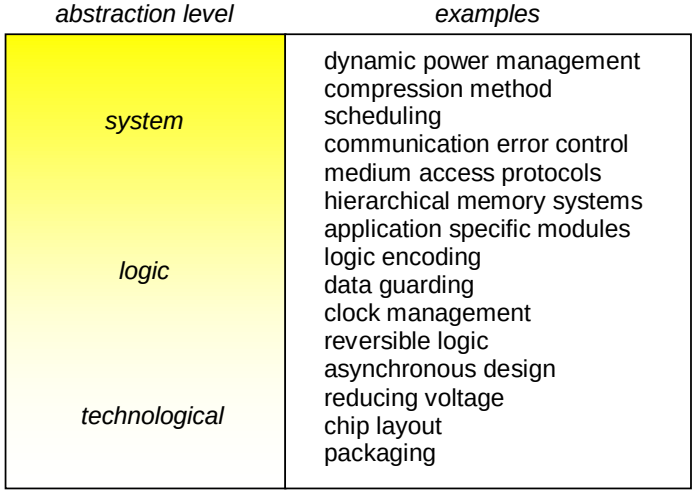


Figure 2: General design flow and related examples for energy reduction.

Given a design specification, a designer is faced with several choices at different levels of abstraction. The designer has to select a particular algorithm, design or use an architecture that can be used for it, and determine various parameters such as supply voltage and clock frequency. This multi-dimensional design space offers a large range of possible trade-offs. At the highest level the design decisions have the most influence. Therefore, the most effective design decisions derive from choosing and optimising architectures and algorithms at the highest levels. It has been demonstrated by several researchers [63] 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 high level design decisions because

implementation details can only be accurately modelled or estimated at the technological level and not at the higher levels of abstraction. Furthermore, the specific energy reduction techniques that are offered by the lower layers can be most effective only when the higher levels are aware of these techniques, know how to use them, and apply them.

2.2.2 CMOS component model

Most components are currently fabricated using CMOS technology. Main reasons for this bias is that CMOS technology is cost efficient and inherently lower power than other technologies. The sources of energy consumption on a CMOS chip can be classified as *static* and *dynamic* power dissipation. The main difference between them is that dynamic power is frequency dependent, while static is not. Bias (P_b) and leakage currents (P_l) cause static energy consumption. Short circuit currents (P_{sc}) and dynamic energy consumption (P_d) is caused by the actual effort of the circuit to switch.

$$P = P_d + P_{sc} + P_b + P_l \qquad (2)$$

The contributions of this static consumption are mostly determined at the circuit level. While statically-biased gates are usually found in a few specialised circuits such as PLAs, their use has been dramatically reduced in CMOS design [5]. 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.

Dynamic power can be partitioned into power consumed internally by the cell and power consumed due to driving the load. Cell power is the power used internally by a cell or module primitive, for example a NAND gate or flip-flop. Load power is used in charging the external loads driven by the cell, including both wiring and fanout capacitances. So the dynamic power for an entire chip is the sum of the power consumed by all the cells on the chip and the power consumed in driving all the load capacitances.

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 ($I_{crowbar}$). This effect causes a power dissipation of approx. 10 to 15%.

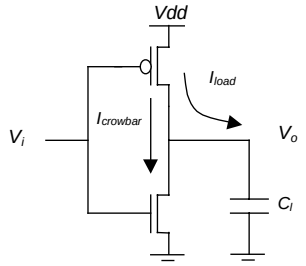


Figure 3: Dynamic power in a CMOS inverter.

The more dominant component of dynamic power is capacitive power. This component is the result of charging and decharging parasitic capacitances in the circuit. Every time a capacitive node switches from ground to V_{dd} an vice-versa energy is consumed.

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

$$P_d = C_{eff} V^2 f \qquad (3)$$

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 [33]. The power dissipation arises from the charging and discharging of the circuit node capacitance found on the output of every logic gate. Every low-to-high logic transition in a digital circuit incurs a voltage change ΔV, drawing energy from the power supply. C_{eff} combines two factors C, the capacitance being charged/discharged, and the *activity weighting* α, which is the probability that a transition occurs.

$$C_{eff} = \alpha C \qquad (4)$$

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

2.2.3 Power modelling and analysis

The search for the optimal solution must include, at each level of abstraction, a ‘design improvement loop’. In such a loop a power analyser/estimator ranks the various design, synthesis, and optimisation options, and thus helps in selecting the one that is potentially more effective from the energy consumption standpoint. Obviously, collecting the feedback on the impact of the different choices on a level-by-level basis, instead of just at the very end of the flow (i.e. at the gate level), enables a shorter development time. On the other hand, this paradigm requires the availability of power estimators, as well as synthesis and optimisation tools, that provide accurate and reliable results at various levels of abstraction. Power analysis tools are available primarily at the gate and circuit levels, and not at the architecture and algorithm levels where they could really make an impact. Current research is trying to fill this gap [33][43][86].

2.2.4 How much is a picojoule?

In Table 3 we compare the energy of a transition of a CMOS gate (about 1 picojoule) with a variety of energy quantities.

Note that neural transitions are an order of magnitude more efficient (in average). An assignment consumes about 100 picojoules for a word of eight bits. Note that the execution of a single instruction of the most efficient commercial available 32-bit microprocessor around, the ARM, dissipates two order of magnitude more. The

instruction of a DEC Alpha consumes even a thousand times more energy. This clearly suggests that micro-processors, due to their general-purpose nature, are not particularly energy efficient.

Quantity	Energy	Remark
Uv photon	-18	
Neural transition	-13	Varies with size
CMOS transition	-12	Gate with 100fF load
α-particle	-12	From space or IC package
8-bit assignment	-10	
PCB transition	-10	10 pF load
ARM instruction	-8	
8-bit access 16Mb SRAM	-8	
DEC Alpha instruction	-7	
Correcting DCC word	-6	
NiCd penlight battery	3	
Can of beer	6	600 kJ
Lead-acid car battery	6	5kg x 40Wh/kg
Kg coal	7	
Daily human consumption	7	2500 kilocalories
Man-made nuclear explosion	14	Trinity (July 16, 1944)
1906 San Francisco earthquake	17	8.3 on the Richter scale
Nova	37	
Big bang	73	

Table 3: The 10-log of the energy for various quantities [10-log Joules] (from [7]).

2.3 Low-power technological-level design

The previous section has presented the theoretical foundation for the analysis of energy consumption. From this section onwards, we will discuss the energy reduction techniques and trade-offs that involve energy consumption of digital circuits. We use a bottom-up organisation starting at the lowest level of abstraction and proceed upward. As we move up the abstraction level, the techniques and trade-offs become less exact due to more freedom in design configuration and decision.

The Equations (3) and (4) suggest that there are essentially four ways to reduce power:

- reduce the capacitive load *C*,
- reduce the supply voltage *V*,
- reduce the switching frequency *f*,
- reduce the switching activity α.

Despite the differences in optimisation and trade-off possibilities at the various levels of abstraction, the common themes of the low-power techniques are quite similar.

The technological level comprises the technology level, dealing with packaging and process technologies, the layout level that deals with strategies for low-power placement and routing, and the circuit level that incorporates topics like asynchronous logic and dynamic logic.

2.3.1 Minimise capacitance

Energy consumption in CMOS circuitry is proportional to capacitance *C*. Therefore, a path that can be followed to reduce energy consumption is to minimise 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.

Connection capacity and packaging

A significant fraction of the chip's energy consumption is often contributed to driving large off-chip capacitances, and not to core processing. Off-chip capacitances are in the order of five to tens of picofarads, while on-chip capacitances are in tens of femtofarads. For conventional packaging technologies, [3] suggests that pins contribute approximately 13-14 pF of capacitance each (10 pF for the pad and 3-4 pF for the printed circuit board). Since Equation (3) indicates that energy consumption is proportional to capacitance, I/O power can be a significant portion of the overall energy consumption of the chip. Therefore, in order to save energy, use few external outputs, and have them switch as infrequently as possible.

Packaging technology can have a impact on the energy consumption. For example, in multi-chip modules where all of the chips of a system are mounted on a single substrate and placed in a single package, the capacitance is reduced. Also, accessing external memory consumes much energy. So, a way to reduce capacitance is to reduce external accesses and optimise the system by using on-chip resources like caches and registers.

Example

To indicate the contribution of the interconnect to the total energy consumption we will compare the energy consumption that is required to compute a 32 x 32 bit multiply with the energy that is needed to fetch a 32-bits word from external memory.

Energy needed to perform a 32 x 32 multiply

The minimal amount of energy consumed for a single *m* x *n* multiply is given by [64]:

$$E_{mul} = m \,.\, n \,.\, \rho_{mul} \,(E_{fa} + E_{and}) \qquad (5)$$

When implemented in a 1 μm, 5V CMOS process, the energy for a full adder E_{fa} = 2.41 pJ and the energy to perform the AND E_{and} = 0.35 pJ. The ripple factor ρ_{mul} represents any extra energy due to ripples within the arithmetic element. Depending on the architecture it can take values between 1.5 and 2.5. So, with a ρ_{mul} = 2, the amount of energy for a single 32 x 32 multiplication equals approximately 5.7 nJ.

Energy needed to transfer data from memory

We are not interested here in the energy dissipation of the memory-core itself and concentrate on the energy consumption to transfer data bits over capacitive I/O pins. This amount of energy consumed can be expressed with:

$$E_{pad} = p \,.\, ½\, C_{IO} \,.\, V^2 \qquad (6)$$

in which *p* equals the transition probability. When we estimate the transition probability to be 0.5, use a capacitance C_{IO} of an I/O pad of 5 pF and an I/O voltage of 5 V, then one I/O pad needs an energy E_{pad} of 31.25 pJ. With a memory organisation consisting of 16 banks of 2 bits connected to the processor with an interface chip (e.g. an MMU) we have 19 I/O pads for the address (1 to the processor, 2 to the memory interface, and 16 to the memory), and 4 I/O pads for the data. The amount of energy required transferring a 32-bit data over the I/O pins between memory and processor using 24 bits address and 3 control lines thus requires (24.19 + 3.19 + 32.4). E_{pad} = 20 nJ.

The amount of energy consumed just to transfer one 32 bit word between memory and processor is thus almost four times the amount of energy needed for a 32 x 32 bit multiplication. When a better process – like 0.25 μm and 1.8 V – is used, the multiply will become much less energy consuming because it is smaller (16 times) (see the next paragraph) and because of the lower voltage (7.6 times less energy). The I/O pad has no advantage of the smaller feature size because its capacitance will remain about the same.

❑

Technology scaling

The process technology has been improved continuously, and as the SIA roadmap indicates, the trend is expected to continue for years [62]. Scaling of the physical dimension involves reducing all dimensions: thus transistor widths and lengths are reduced, interconnection length is reduced, etc. Consequently, the delay, capacitance and energy consumption will decrease substantially. For example, MIPS Technologies attributed a 25% reduction in power consumption for their new processor solely to a migration from a 0.8 μm to a 0.64 μm process [90].

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

Chip layout

There are a number of layout-level techniques that can be applied. Since the physical capacitance of the higher metal layers are smaller, there is some advantage to select upper level metals to route high-activity signals. Furthermore, traditional placement involves minimising area and delay, which in turn translates to minimising the physical capacitance (or length) of wires. Placement that incorporates energy consumption, concentrates on minimising the activity-capacitance product rather than capacitance alone. In general, high-activity wires should be kept short and local. Tools have been developed that use this basic strategy to achieve about 18% reduction in energy consumption [12].

Conclusion on capacitance reduction

The capacitance is an important factor for the energy consumption of a system. However, reducing the capacity is not *the* distinctive feature of low-power design, since in CMOS technology energy is consumed only when the capacitance is switched. It is more important to concentrate on the switching activity and the number of signals that need to be switched. Architectural design decisions have more impact than solely reducing the capacitance.

2.3.2 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 5 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 [58]. In other cases, vendors have introduced ‘true’ low voltage versions of their processors that run at the same speed as their 5 Volt counterparts. The majority of the techniques employing concurrency or redundancy incur an inherent penalty in area, as well as in capacitance and switching activity. If the voltage is allowed to vary, then it is typically worthwhile to sacrifice increased capacitance and switching activity for the quadratic power improvement offered by reduced voltage.

The variables voltage and frequency have a trade-off in 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 4). Therefore, major themes in

many power optimisation techniques are to optimise the speed and shorten the critical path, so that the voltage can be reduced. These techniques often translate in larger area requirements, hence there is a new trade-off between area and power.

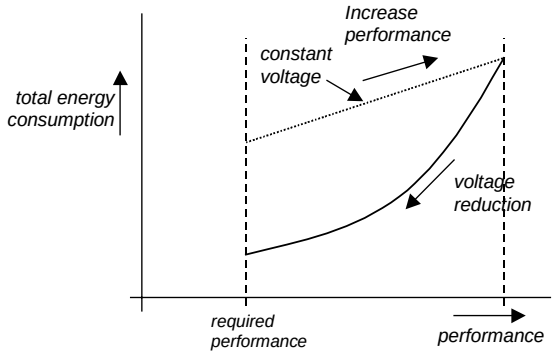


Figure 4: Impact of voltage scaling and performance to energy consumption.

Weiser et al. [76] 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. In order to operate properly at a lower voltage, the clock rate must be simultaneously reduced.

The main limitation of all voltage scaling approaches is that they assume the designer has the freedom of choosing the voltage supply for the design. Unfortunately, for many real-life systems, the power supply is not a variable to be optimised. Furthermore, in current and future technologies, voltage scaling will become increasingly difficult because of reduced noise margins and deterioration of device characteristics [5].

2.3.3 Avoid unnecessary activity

We can summarise the previous subsections as follows: the capacitance can only marginally be changed and is only important if switched, the voltage is usually not under designer's control, and the clock frequency, or more generally, the system throughput is rather a constraint than a design variable. The most important factor contributing to the energy consumption is the switching activity. Actually, once the technology and supply voltage have been set, major energy savings come from the careful minimisation of the switching activity α of Equation (4).

While some switching activity is functional, i.e. it is required to propagate and manipulate information, there is a substantial amount of unnecessary activity in virtually any digital circuit. Unnecessary switching activity is due to 1) spurious transitions due to unequal propagation delays (glitches), 2) transitions occurring within units that are not participating in a computation or 3) whose computation is redundant.

Reordering of logic inputs to circuits can have significant energy consumption consequences. For example, Figure 5 shows two functional identical circuits, but with a

different energy consumption due to the different signalling activity. The normalised energy consumption equals 0.11 of circuit *a*, and 0.021 for circuit *b*.

Thus, much energy can be saved by minimising the amount of switching activity needed to carry out a given task within its performance constraints. The activity weighting α of Equation (4) can be minimised by avoiding unnecessary and wasteful activity. There are several techniques to achieve this. In this section we will only mention the techniques at the technological level and circuit level. The techniques that are possible at the logic, architectural and system level are discussed in later sections.

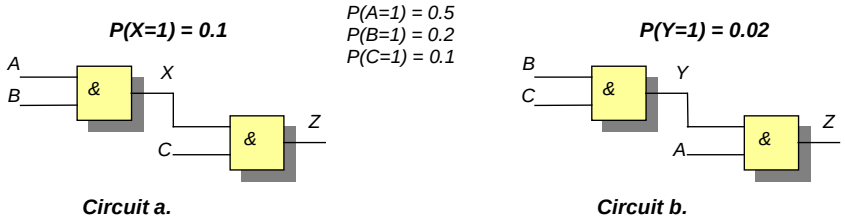


Figure 5: Reordering logic inputs.

Asynchronous design

One way to avoid unnecessary activity is by applying an asynchronous design methodology [6][47][23][55]. CMOS is a good technology for low power as gates mainly 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. The circuit activity is often low (below 10%), for one of the following reasons [7].

- The clock frequency often exceeds the sample frequency by several orders of magnitude, or order to allow for time sharing of resources such as busses, I/O pads, memories, etc.
- Large ICs consist of a number of more-or-less independent modules. These modules generally have different optimal clock frequencies. Nevertheless, the number of different clock frequencies on the IC is kept small to avoid problems with interfacing, clock distribution, and testing.
- The clock frequency must be such that a worst-case workload can be handled in the allocated time. This generally implies an excessively high clock frequency for the average case.

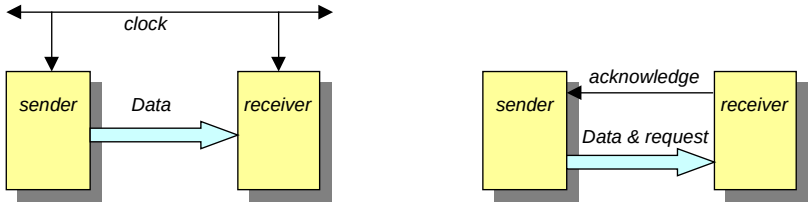
Many gates switch because they are connected to the clock, not because they have new inputs to process. A synchronous circuit therefore wastes power when particular blocks of logic are not utilised, for example, in a floating point unit when integer arithmetic is being performed. The biggest gate of 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.

Example [86]

The chip size of a CPU is 15 x 25 mm with clock frequency of 300 MHz operating at 3.3V. The length of the clock routing is estimated to be twice the circumference of the chip. Assume that the clock signal is routed on a metal layer with width of 1.2 µm and the parasitic capacitance of the metal layer is 1 fF/µm^2. Using Equation (3) the power dissipation of the clock signal is then 627 mW.

❑

This example is even conservative: in the DEC Alpha the distribution of the single-wire 200 MHz clock requires 7.5 Watts out of a total of 30 Watts. The clock driver dissipates another 4.5 Watts [7]!



a) synchronous communication protocol ***b) self-timed protocol***

Figure 6: Synchronous and self-timed protocols.

In *asynchronous systems* there is no global synchronisation of the modules within the system, but modules do synchronise locally through their communication protocols [20]. The set of asynchronous communication protocols that use some form of handshake between sender and receiver are known as *self-timed*. Asynchronous circuits are inherently data driven and are only active when performing useful work. An important property of self-timed protocols is that they are speed independent: the protocols work regardless of the speed that the modules produce results. In addition, protocols such as the dual-rail code are delay-insensitive: arbitrary delays may occur on the wiring and the protocol will still work. Among several advantages like modularity and robustness, power consumption is low.

The main drawback of asynchronous systems is that extra circuitry and signals is required to generate the timing information.

2.3.4 Technological and circuit-level conclusions

Clearly, numerous techniques at the technological and circuit level are available to the low-power system designer. These techniques include optimisation in packaging, technology scaling and layout level, a careful selection of the techniques used at circuit level, and applying energy saving techniques at gate level.

The gains that can be reached at these levels are, however, limited. The technology scaling for example offers significant benefits in terms of energy consumption only up to

a certain point. Once parasitics begin to dominate, the power improvements slack off or disappear completely. So we cannot rely on technology scaling to reduce energy consumption indefinitely. We must turn to other techniques for lowering energy consumption. Some of the techniques can be applied in conjunction with higher-level energy-reduction techniques.

The main concepts that are used at gate-level are associated with avoiding unnecessary activity and trading energy for performance through low-voltage concurrent processing. The gains reported by low-power designers working at the gate level are typically on the order of a factor of two or less [33]. However, technology- and circuit-level techniques can have major impact because some circuits are repeated thousands of times on a chip.

So, while these techniques should be exploited whenever possible, this should not be done at the expense of the larger gains achievable at the architecture and algorithm levels, which will be discussed in the following sections.

2.4 Low-power logic-level design

We consider the logic level as the level between the technological related issues and the system level. Issues in the logic level relate to for example state-machines, clock gating, encoding, and the use of parallel architectures.

At the logic level, opportunities to economise on power exist in both the capacitance and frequency spaces. The most prevalent theme in logic-level optimisation techniques is the reduction of switching activities. We will now give some typical examples of reducing energy consumption at this level.

2.4.1 Cell library

The choice of the cell library to use for a chip design provides the first obvious opportunity to save energy. Standard cells have lower input capacitances than gate arrays because they use a variety of transistor sizes. For the same reason, the cells themselves consume less power when switching. Using libraries designed for low power can also reduce capacitance. These libraries contain cells that have low-power micro-architectures or operate at very low voltages. Some of the leading application-specific IC (ASIC) vendors are providing such libraries today, and many captive design groups are producing specialised libraries for low-power applications. But no matter which type of library is utilised, the logic designer can minimise the power used by each cell instance by paying careful attention to the transition times of input and output signals. Long rise and fall times should be avoided in order to minimise the crowbar current component of the cell power.

2.4.2 Clock gating

Several power minimisation techniques work especially well at the logic level. Most of them rely on switching frequency. The best example of which is the use of *clock gating*. 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 [28][35]. In clock gating, a control signal enables a clock signal so that the clock toggles only when the *enable* signal is true, and is held steady when the *enable* signal is false. Gated clocks are used, in power management, to shut down portions of the chip, large and small, that are inactive. This saves on clock power, because the local clock line is not toggling all the time.

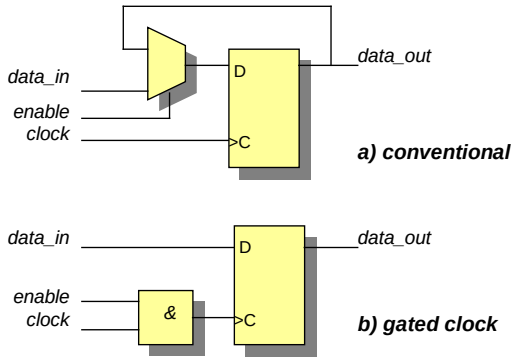


Figure 7: Clock gating.

Consider the case of a data bus input register as depicted in Figure 7. With the conventional scheme, the register is clocked all the time, whether new data is to be captured or not. If the register must hold the old state, its output is fed back into the data input through a multiplexer whose enable line controls whether the register clocks in new data or recycles the existing data. With a gated clock, the signal that would otherwise control the select line on the multiplexer now controls the gate. The result is that the energy consumed in driving the register's clock input is reduced in proportion to the decrease in average local clock frequency. The two circuits function identically, but utilisation of the gated clock reduces the power consumption.

Clock gating can be implemented locally by gating the clocks to individual registers, or globally, by building the gating structures into the overall architecture to turn off large functional modules. While both techniques are effective at reducing energy, global gating results in much larger energy reductions and is often used in implementing power-down and power-management modes. 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. Control can be done at the hardware level or the operating system or the application can manage it. The PowerPC603, for example, contains three power management modes – doze, nap, and sleep – that are controlled by the operating system and cut power use overall when the processor is idle for any extended period of time. With these modes, chip power can go

from 2.2 W in active mode to 358 mW in doze, 126 mW in nap, and as low as 1.8 mW in sleep.

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 stabilise 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 system has to predict whether it is profitable to shut down parts of the system.

2.4.3 State-machine modifications

A state-machine is an abstract computation model in which the designer specifies a state-transition graph. This can be implemented using Boolean logic and flip-flops. Among the modifications that can be made to achieve a higher energy efficiency are *decomposition* and *restructuring*. These approaches try to minimise the activity along the lines connecting the sub-machines, which tend to drive heavier loads. Shutdown techniques like clock-gating can be applied to the individual machines because only one is active at any point in time.

The *encoding* of the state machine (which is the encoding of states to bits in the state register) is another important factor that determines the quality (area, energy consumption, speed, etc.) of the system. Several key parameters have been observed to be important to the energy efficiency of state encoding. One such parameter is the expected number of bit transitions in the state register. Another parameter is the expected number of transitions of output signals.

Consider for example two functionally identical state machines M1 and M2 with different encoding shown in Figure 8 (figure and example from [89]). The labels at the state transitions edges represent the probability that transitions will occur at any given clock cycle. The expected number of state-bit transitions $E[M]$ is given by the sum of products of edge probabilities and their associated number of bit-flips as dictated by the encoding.

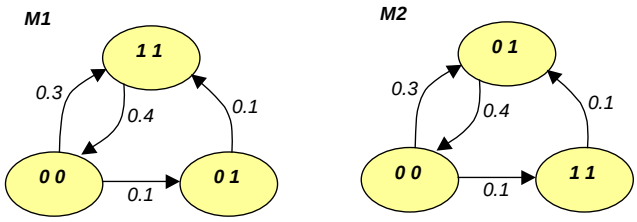


Figure 8: Functionally identical state machines with different encoding

The expected transitions per clock cycle for the two machines are:

$$E[M1] = 2\ (0.3 + 0.4) + 1\ (0.1 + 0.1) = 1.6$$

$$E[M2] = 1\ (0.3 + 0.4 + 0.1) + 2\ (0.1) = 1.0$$

In general, machines with lower E[M] are more energy efficient because there are fewer transitions of the state register and fewer transitions are propagated into the combinatorial logic of the machine.

Note that encoding affects the energy dissipation as well as the required area of the machine. If we encode the states to minimise the expected transition, the area is often large because the logic synthesis system has lost the freedom of assigning state codes that favour area optimisation. One practical solution is to encode only the subset of states that spans the high probability edges. The remaining state codes can be left to the logic synthesis system

Unlike area optimisation, the power optimisation of state machines requires the knowledge of the probabilities of input signals and hence state transitions. Without this knowledge, power optimisation of state encoding is impossible.

The principle of clock-gating discussed above can also be applied in state-machines: for example in the design of synchronous finite state machines (FSM) the clock can be disabled if the unit is not needed at a certain moment. For example Koegst et al. [31] use gated clocks in FSM designs to disable the state transition of so called self-loops.

2.4.4 Logic encoding

Energy consumption is proportional to the frequency at which signals change state and to the capacitance on the signal line. This is true for every signal 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 minimising the number of transitions. The designer of a digital circuit often has the freedom of choosing the encoding scheme. Different encoding implementations often lead to different power, area and delay trade-off. A correct choice of the representation of the signals can have a large impact on the switching activity. Usually, encoding techniques require the knowledge of signal statistics.

The frequency of consecutive patterns in the traffic streams is the basis for the effectiveness of encoding mechanisms. 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 [57]. Therefore, a combination can be used in which only the most frequently changing LSB bits use a Gray code.

A simple, yet effective, low-power encoding scheme is the bus-invert code [72]. If the Hamming distance between two successive patterns is larger than $N/2$, where N is the bus width, the current pattern is transmitted with inverted polarity. A redundant bus line is needed to signal to the receiving end of the bus which polarity is used for the transmission of the incoming pattern. The method guarantees a maximum of N/2

transitions, and performs well when the patterns are randomly distributed in time and no information about their correlation is available.

2.4.5 Data guarding

Switching activity is the major cause of energy dissipation in most CMOS digital systems. In order to transfer data and do computation, switching activities cannot be avoided. However, switching activities that do not contribute to the actual communication and computation should be eliminated. The basic principle is to identify logical conditions at some inputs to a logic circuit that is invariant to the output. Since those input values do not affect the output, the input transitions can be disabled. The approach is based on reducing the switching activities by placing some *guard logic*, consisting of transparent latches with an enable signal, at the inputs of each block of the circuit that needs to be power managed. This logic will guard not useful switching activities to propagate further inside the system. The latches are transparent when the data is to be used. Otherwise, if the outputs of a unit are not used, then they do not change.

The position of the registers within a design may greatly affect the area and performance of the implementation. The transformation that repositions the registers of a design without modifying its external behaviour is called *retiming*. If a register can be positioned before the output of the circuit, some spurious transitions (i.e. glitches) are filtered by the register and thus not propagated further.

2.4.6 Conclusion

With the advance of logic synthesis tools and structured VLSI design practice today, logic design is seldom performed manually. However, the logic-level design can have a high impact on the performance and energy-efficiency of the system. Even with the use of hardware description languages like VHDL, there are still many techniques for the designer to reduce energy consumption at the logic level. The most effective technique used at this level is the reduction of switching activities.

2.5 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. We already concluded that the impact of these techniques is limited due to several factors. It is unlikely that the combined effort of technology-level, gate-level and circuit level reduce power by more than a factor of two in average [5]. Technology roadmaps and trend analyses [62] clearly show that this result is far from being sufficient. In this section we will concentrate on the energy reduction techniques at architecture and system level, and we will evaluate the relevance for low-power system design.

We define a *system* as an interconnection of possibly heterogeneous resources (electronic, electro-mechanical, optical, etc.) which are often separately designed. System-level design deals with connecting the resources in a functionally correct and efficient fashion. In this definition all practical digital devices like computers and PDAs are systems. A chip can be a system if its components are designed and optimised as separate resources.

The two main themes that can be used for energy reduction at these higher levels are:

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

Typical examples at these levels include algorithmic transformations, partitioning, memory organisations, power management, protocol design and selecting dedicated versus programmable hardware.

2.5.1 Optimise communication channels

The observation has already been made that energy in real-life systems is to a large extend dissipated in *communication channels*, sometimes even more than in the computational elements. Experiments have demonstrated that in designs, about 10 to 40% of the total power may be dissipated in buses, multiplexers and drivers [1]. This amount can increase dramatically for systems with multiple chips due to large off-chip bus capacitance.

The power consumption of the communication channels is highly dependent on algorithm and architecture-level design decisions. Two properties of algorithms and architectures are important for reducing the energy consumption due to the communication channels: locality and regularity.

Locality relates to the degree to which a system or algorithm has natural isolated clusters of operation or storage with 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 with a low electrical capacity 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 minimisation or performance optimisation. However, for power reduction it is, for instance, better to minimise the number of accesses to long global buses and have the local buses be accessed more frequently. In a direct implementation targeted at area optimisation, 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) localisation of interconnect allows a more compact layout and (2) fewer (access to) multiplexers and buffers are needed.

Localisation reduces the communication overhead in processors and allows the use of minimum sized transistors, which results in reductions of capacitance. Pipelining and

caching are examples of localisation. Another way to reduce data traffic over a 'long' distance is to integrate a processor in the memory, as for example proposed by Patterson in intelligent RAM [53][48]. This approach also reduces the processor-memory bottleneck.

At system level locality can be applied by dividing the functionality of the system into dedicated modules [1][44]. When the system is decomposed into application-specific modules, 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.

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. Several researchers (e.g. Mehra [49] and Rabaey [59]) have exploited these techniques, but mainly in the DSP domain where a large set of applications inherently have a high degree of regularity.

2.5.2 Low-power memory organisation

Closely related to the previous section that discussed the optimisation of communication channels, is the memory organisation. In processor based systems, a significant fraction of the total energy budget is consumed in memories and buses. Minimising the memory accesses, and clustering them, minimises the cost of bus transactions and can reduce energy consumption. Accesses can be reordered and special bus-encoding techniques can be used to minimise the transition activity on memory busses.

There are furthermore various techniques to reduce the energy consumption of the secondary storage. Secondary storage is in general non-volatile and is used to store large amounts of data. Caching techniques can be used for both main memory and secondary storage to increase performance and reduce energy consumption.

Main memory

Main memory is generally implemented using Dynamic Random Access Memories (DRAM or SDRAM). These chips can be in three modes: active, standby, and off. In active mode, the chip is reading or writing. In standby mode, the chip needs to be refreshed periodically to maintain the data stored in the memory cells. The energy consumption of DRAM can be significant, for example 8 MB of EDO DRAM memory from Micron consumes about 580 mW in active mode, and 1.65 mW in standby mode. Static memory (SRAM) does not need to be refreshed, and therefore consumes less energy in standby mode. SRAM is in general faster than DRAM, requires more chip area, and is more expensive. For example, the Samsung KM616FS1000ZI 128K*16 100 ns SRAM consumes 216 mW when active, and 0.1 mW when standby. Note that using more energy-consuming memory can be more energy-efficient when it is faster, and can thus remain longer in a lower energy consuming mode [64].

The 'off' state of main memory chips can only be used when it is determined that the entire system will be idle for a significant period of time. The content of all main memory is saved to secondary storage before the main memory system can be turned off.

There are various ways to reduce the energy consumption of a memory array. Tierno and Martin describe various memory organisations in relation to their energy consumption [74]. They conclude that a memory organisation as a multidimensional array of memory cells gives an improvement in energy per access, but requires more chip area.

Breaking up the memory into several sub-arrays can further reduce the energy per access, so that only one of the smaller sub-arrays is accessed in each memory reference. This technique is for example applied in the Direct Rambus DRAM system (RDRAM) [80]. The Rambus physical layer contains 30 high speed, controlled impedance, matched transmission lines. These high-speed signals are terminated at their characteristic impedance at the RDRAM end of the channel. Power is dissipated on the channel only when a device drives a logic '1' (low-voltage) on the pin. All high-speed signals use low-voltage swings of 800 mV. The RDRAM has several built-in operating states to reduce energy consumption. In active mode the RDRAM is ready to immediately service a memory transaction request. At the end of a transaction an RDRAM automatically transitions to standby mode.

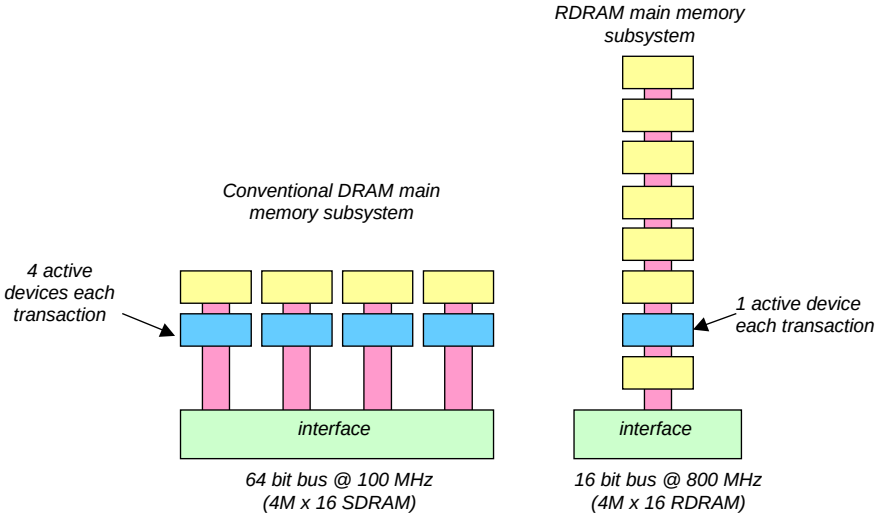


Figure 9: Conventional SDRAM and RDRAM block diagram

The organisation of a RDRAM device allows an RDRAM main memory subsystem to consume less power than a conventional system. Figure 9 shows a conventional DRAM memory subsystem topology compared to a RDRAM design. Note that for each transaction request from the memory subsystem an entire bank of conventional DRAM devices are activated and consume energy versus a single device activation with the Rambus design. Since a conventional DRAM device's throughput is much lower than an

RDRAM, several DRAM devices are operated in parallel as a bank of memory to provide the desired data bandwidth onto a 64-bit data bus. When a memory transaction request is sent out to the memory array the appropriate RDRAM device services the request and moves to active mode while the other RDRAMs remain in standby mode.

Employing nap mode can further reduce power consumption. In nap mode the energy consumption is reduced to 10% of the energy consumption in standby mode. Some extra latency is introduced to return from nap mode to standby.

A similar memory organisation technique can be used with conventional memory devices. When the memory is divided into several small blocks that can be individually powered down, then the memory allocation strategy and garbage collector of the operating system can take benefit of this by allocating the memory in clustered memory blocks, such that unused memory is not spread over all memory banks.

Caching

The previously described memory organisation techniques try to minimise the energy cost of a single access to memory, under the assumption that all addresses are equally probable. In most applications, address distributions are far from random: the past history of the address sequence can be used to increase memory throughput and decrease the average energy per access. The assumptions made about the address sequence are *spatial* and *temporal locality*. Spatial locality indicates that once an address has been accessed, there is a strong probability that a nearby address will be accessed in the near future. Temporal locality indicates that, once an address has been accessed, there is a strong probability that the same address will be accessed again in the near future. Spatial locality is used by pre-fetch mechanisms. The cost per word of fetching a multi-word line from memory decreases with the number of words on the line (the energy cost of decoding the address is shared among more words). Temporal and spatial locality can be used to store a copy of the contents of the memory locations most likely to be needed in the future, in a small, fast, energy-efficient memory. If the locality is strong enough, most of the memory references will be serviced by the small memory (e.g. a cache memory), with a corresponding improvement in energy performance.

By employing an on-chip *cache* significant power reductions can be gained together with a performance increase. For example, improvements in cache organisations for high performance processors also reduce traffic on high capacitance busses. As, most of the time, only the cache is read, the energy consumption is reduced. This phenomenon clearly helps to save energy although the primary goal is improving performance. However, as these caches are typically implemented with high-performance static RAM cells, these caches often consume a significant amount of energy. For example, two modern embedded RISC microprocessors, the StrongARM 110 and a PowerPC, the energy consumption of the cache is either the largest or second largest power-consuming block [29].

A cache designed for low-energy has to optimise the hit ratio at relatively small cache sizes. This is because in general caches with good hit ratios use complicated architectures, which make the energy cost of a cache access high. Special techniques,

like using a small *filtering cache* that is placed just before the normal (large, complex and high energy consuming) cache, trade off performance for energy consumption. Experimental results across a wide range of embedded applications show that the filter cache results in improved memory system energy efficiency (for example a direct mapped 256-byte filter cache can achieve a 58% energy reduction while reducing the performance by 21%) [22].

Note, however, that, although for many applications and data streams these techniques are profitable for performance and energy consumption, for streaming data, these caches might even become an obstacle to high performance and low power. This kind of traffic – typically multimedia traffic – is characterised by its one-time reference [2]. The locality principle, the key property behind the use of caches, is not valid for this kind of traffic. The media functions typically involve processing a continuous stream of input [32], thereby effectively emptying the cache of useful processor data. It has been observed that future processor designs spend a large fraction of their area budget on local caches, and not on energy conserving techniques. The *Computer* journal produced a special issue on "Billion-transistor architectures" that discussed problems and trends that will affect future processor designs [10]. Most of these designs focus on the desktop and server domain. The majority use 50 to 90 percent of their transistor budget on caches, which helps mitigate the high latency and low bandwidth of external memory. In other words, in the vision of future computer designers most of the billion-transistor budget is spent on redundant, local copies of data normally found elsewhere in the system [32].

The compiler used to make a program can also utilise the locality principles by reducing the number of instructions with memory operands. Much energy can be saved by a proper utilisation of registers [75]. It was also noted that writes consume more energy, because a processor with a write-through cache (like the Intel 486) always causes an off-chip memory operation.

Secondary storage

Secondary storage in modern computers generally consists of a magnetic disk supplemented by a small portion of main memory RAM used as disk cache. Having the motor of the disk off saves energy. However, when it needs to be turned on again, it will take considerable time and energy to return to full operation.

The larger the cache, the better the 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. Unfortunately, there is a trade-off in the size of the cache memory since the required amount of additional DRAM can use as much energy as a constantly spinning hard disk [86].

Because of size, energy consumption and weight limitations of handheld machines a possible technology for secondary storage is flash memory. Like a hard disk, flash memory is non-volatile and can hold data without consuming energy. Furthermore, when reading or writing flash memory, it consumes only 0.15 to 0.47 W, far less than a

hard disk. It has read speed of about 85 ns per byte, quite like DRAM, but write speed of about 4-10 us, about 10-100 times slower than hard disk. However, since flash memory has no seek time, its overall write performance is not that much worse than a magnetic disk; in fact, for sufficiently small random writes, it can actually be faster [42]. Since flash is practically as fast as DRAM at reads, a disk cache is no longer important for read operations.

The cost per megabyte of flash is about 17-40 times more expensive than hard disk, but about 2-5 times less expensive than DRAM. Thus, flash memory might also be effective as a second level cache below the standard DRAM disk cache.

2.5.3 Programmability

Programmability is an attractive feature for any system, since it allows one system to be used for many applications. Programmability is even more important for mobile systems because they operate in a dynamically changing environment and must be able to adapt to the new environment. For example, a mobile computer will have to deal with unpredicted network outage or should be able to switch to a different network, without changing the application. It should therefore have the *flexibility* to handle a variety of multimedia services and standards (like different video decompression schemes and security mechanisms) and the *adaptability* to accommodate the nomadic environment, required level of security, and available resources [63][70].

The requirement for programmability in systems is also triggered by economical reasons. The high costs involved in designing and implementing a chip does not justify the design of a system that implements only a single application. Furthermore, because the requirements of applications are increasing rapidly, new chips need to be designed more often.

The structure of systems and applications is usually based on a modular design. This is needed to manage the complexity of the design, and also allows the designer to compose different applications of different combinations of the modules. The functions expressed by the modules typically consist of algorithms comprised of basic arithmetic operations (e.g. add, subtract, multiply). Descriptions in these basic operations we will call *fine-grain*, whereas descriptions of functions such as algorithms in which the complexity of the functions is large in comparison to the basic mathematical primitives will be called *coarse-grain*. Microprocessor designers are generally committed to the fine-grain level since they must provide generally applicable elementary operations. Implementing functions at the coarse-grain level usually yields sufficient flexibility for the application domain.

Each level of granularity has its own preferred and optimal application domain. Some of these levels are illustrated in Figure 10, which shows three different approaches in the spectrum of applications and hardware implementations.

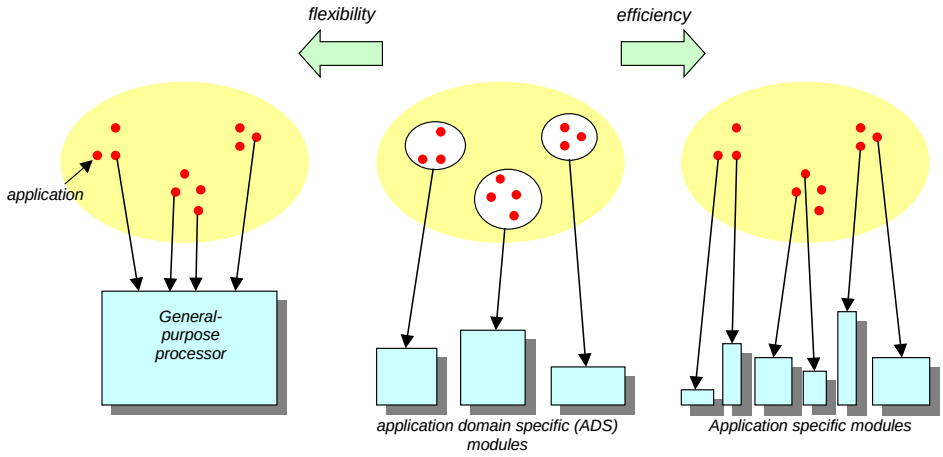


Figure 10: The spectrum of applications and hardware implementations [38].

General-purpose processors

There are two conflicting demands in the design of a high-performance architecture: efficiency and flexibility. The current trend is to focus on flexibility with *high performance general-purpose processors* as this is the area in which a semiconductor vendor can enhance its status [17][32]. Therefore, the architecture of a general-purpose processor is most widely studied, and optimisations for processor performance is the main goal. The advance in technology has lead to number of processor improvements like superscalar technology, VLIW architectures, reduce in cycle time, large on-chip caches, etc. With the general-purpose processors it is possible to map any type of application on the hardware simply by writing the right software for it. The only limitations are processing capacity and storage. It has been reported that 50 to 90 percent of the number of transistors in current high performance processors and future processor architectures is used for caches and main memory [32].

This flexibility and high-performance poses high demands on technology, and the resulting chips are usually large in terms of chip area and dissipate much energy. While general-purpose processors and conventional system architectures can be programmed to perform virtually any computational task, they have to pay for this flexibility with a high energy consumption and significant overhead of fetching, decoding and executing a stream of instructions on complex general-purpose data paths. General-purpose processors often have to perform tasks for which they are not ideally suited. Although they can perform such tasks, they will take longer, and will use more energy, than a custom hardware implementation (see Section 2.2.4). The energy overhead in making the architecture programmable most often dominates the energy dissipation of the intended computation. For example, when executing a typical DSP application program on the TMS320C5x family of general-purpose DSPs from Texas Instruments with a 20% mix of multiply-accumulate operations, instruction fetching and decoding is

responsible for 76% of the total supply current of the processing core [1]. Clearly a heavy penalty is paid for performing computations on a general-purpose data path under the control of an instruction stream.

Application specific modules

Another environment that will rapidly become more important in the near future is that of *application specific processors or modules*. The goal of these processors is to optimise the overall cost-performance of the system, and not performance alone. The modern application processor can use the technology to increase functionality to provide services such as multimedia devices, compression and decompression, network access, and security functions. Application specific solutions present the most effective way of reducing energy consumption and have been shown to lead to huge power savings [49][38].

Performing complex multimedia-data processing functions in dedicated hardware that is optimised for energy-efficient operation reduces the energy-per-operation by several orders of magnitude relative to software. Conventional general-purpose processors (e.g. Alpha, Pentium) focus entirely on instruction-per-second metrics, and typically require 100 mW/MIP; energy optimised general-purpose processors such as the StrongARM require 1-10 mW/MIP. Fully dedicated, optimised hardware, on the other hand requires less than 0.01 mW/MIP [84]. However, the disadvantage of dedicated hardware is the lack of flexibility and programmability, their functionality is restricted to the capabilities of the hardware. Current implementations of these systems have focussed on teleconferencing type applications and are limited to those. The wide range of multimedia applications being described in the literature cannot all be implemented with this specialised hardware [2].

Application domain specific modules

The difference in area and power dissipation between a general-purpose approach and application specific architectures can be significant. Furthermore, the technological challenges in the design of custom ASICs are usually significantly smaller than the design of general-purpose circuits. This means that high-performance custom chips can be designed and manufactured at relatively low cost. However, this comes at the price of less flexibility, and consequently a new chip design is needed for even the smallest change in functionality.

A hybrid solution with *application domain specific modules* should offer enough flexibility to be able to implement a predefined set of (usually) similar applications, while keeping the costs in terms of area, energy consumption and design time to an acceptable low level. The modules are optimised for one specific application domain. A system designer can use the general-purpose processor for portions of algorithms for which it is well suited, and craft an application domain specific module for other tasks. Unused parts of the system must be switched off when not needed. 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.

When the system is partitioned in multiple modules that each are able to perform some application-domain specific tasks, then *voltage-scaling* might become attractive. The key idea is that energy consumption can be reduced by first increasing the speed beyond the required timing constraints, then by reducing the voltage supply slowing them down until the timing constraints are not exactly met. As practical difficulties and cost were important obstacles in applying this technique at the technological level, at the system level the large size and small number of components may actually allow multiple (and adjustable) voltage supplies.

Reconfigurable computing

Reconfigurable computing systems combine programmable hardware with programmable processors to capitalise on the strengths of hardware and software [87]. While low-power solutions are already available for application specific problems, applying these solutions in a reconfigurable environment is a substantially harder problem, since programmable devices often incur significant performance and energy-consumption penalties [45][46]. To reduce the energy overhead in programmable architectures, the computational granularity should be matched to the architectural granularity. Performing multiplications on a FPGA is bound to carry huge amount of waste, so does executing large dot-vector products on a microprocessor.

2.5.4 Operating system

Up to this point, we have mainly discussed low-power techniques related purely to hardware components of the system. Software and algorithmic considerations can also have a severe impact on energy consumption. Digital hardware designers have promptly reacted to the challenge posed by low-power design. Designer skills, technology improvements and CAD tools have been successful in reducing the energy consumption. Unfortunately, software engineers and system architects are often less energy-conscious than digital designers, and they also lack suitable tools to estimate the energy consumption of their designs. As a result, energy-efficient hardware is often employed in a way that does not make optimal use of energy saving possibilities.

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

Dynamic power management

The essential characteristic of energy consumption for static CMOS circuits is that quiescent portions of a system dissipate a minimal amount of energy. Power management exploits periods of *idleness* caused by system under-utilisation. Especially in mobile systems, the utilisation is not constant. Designers naturally focus on worst-case conditions, peak performance requirements and peak utilisation. As a result, systems are often designed to operate under high utilisation, but they are actually fully

exploited during a relatively small fraction of their lifetime. Dynamic power management refers to the general class of techniques that manage the performance and throughput of a system based on its computational needs within the energy constraints.

We can identify two basic flavours of dynamic power management.

- *Binary power management*

  The most conservative and simple, although quite effective, is to deactivate some functional units when no computation is required. This can be done at different hierarchies and at different levels of design. The main problems involved in dynamic power management is the cost of restaring a powered down module or component. Restarting induces an increase in latency (e.g. time to restore a saved CPU state, spin-up of a disk), and possibly also an increase in energy consumption (e.g. due to higher start-up current in disks). The two main questions involved are then: 1) when to shut-down, and 2) when to wake-up.

  The activity of components in a computing system is usually *event driven*: for example the activity of display modules, communication interfaces, and user interface functions is triggered by external events and is often interleaved with long periods of quiescence. To take advantage of low-power states of devices, 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. Alternatively, the modules use a demand- or data-driven computation to automatically eliminate switching activity of unused modules. The trade-off is to justify the additional hardware and design complexity. The effectiveness is further determined by the extra energy required to implement this technique, and the time (and thus energy) that is required to determine when a module can be shut down (the so-called *inactivity threshold*). The inactivity threshold can be assigned statically or dynamically. In a *predictive* power management strategy the threshold is adapted according to the past history of active and idle intervals. The effectiveness at system level can be high because little additional hardware and design complexity is needed.

  Another question is *when to wake-up*, where the typical policy is to wake up in response to a certain event such as user interaction or network activity. The problem with such a demand policy is that waking up takes time, and the extra latency is not always tolerable. Again, a predictive approach, where the system initiates a wakeup in advance of the predicted end of an idle interval, often works better.

- *Dynamic power management of adaptive modules*

  More advanced power management schemes are based on a quality-of-service framework. Computer subsystems may be designed for multiple levels of reduced energy consumption at the expense of some other system performance measure (e.g. throughput). Key to the approach is a high degree of adaptability of the modules. What really matters in many cases is not sheer performance, but a balance of performance and availability. Users may tolerate performance degradation if functionality is provided for a longer period.

A hierarchical – QoS based – model of the whole system (covering the architecture, wireless communication, distributed processing, and applications) is of great importance for this technique. It is needed to adapt to the changing operating conditions dynamically in the most (energy) efficient way. Besides the functional modules and their ability to adapt (e.g. the effects on its energy consumption and QoS when the image compressor changes its frame rate, its resolution, or even its compression mechanism) this model also includes the interaction between these modules. Such a model should predict the overall consequences for the system when an application or functional module adapts its QoS. Using this model the inherent trade-offs between e.g. performance and energy consumption can be evaluated and a proper adaptation of the whole system can be made.

The whole system (hardware and software) should be designed taking power management into account. The division of the system into modules must be such that the modules provide a clustered functionality which are mutually exclusive and which can be idle for a large fraction of the operation time. 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 co-operation 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 is crucial for the energy reduction of future mobile systems. Power management directly influences the design cycle, and is not only a matter of post-optimisation and tuning.

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 co-ordination 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.

The applicability of dynamic power management is not limited to the system level. The same concept has been exploited successfully at the logic level [52].

Scheduling

In a system scheduling is needed when multiple functional units need to access the same object. In operating systems 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.

Subsystems of a computer, such as the CPU, the communication device, and storage system have small usage duty cycles. That is, they are often idle and wait for the user or network interaction. Furthermore, they have huge differences in energy consumption between their operating states (such as on, standby, sleep).

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

- *Processor time scheduling*

  Most systems spend only a fraction of the time performing useful computation, and 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 [5]. Since all power-down approaches incur some overhead, the task of an energy aware scheduler is to collect requests for computation and compact the active time-slots into bursts of computation.

  Experiments at UCLA with an X server and typical applications on a wireless terminal show that, in theory, a large reduction in CPU power consumption can be obtained if the terminal is shut down whenever it is idle [39]. They noticed that 96 to 98% of the time was spent in the blocked state, and that the average time in the blocked state is very short (much less than a second). Potential energy reduction is from 29 to 62 times.

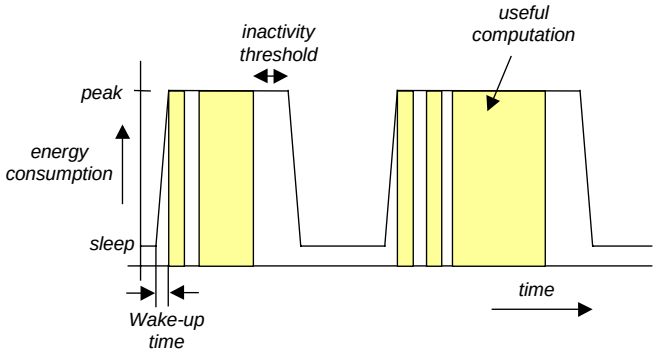


Figure 11: Power consumption in time of a typical processor system.

Weiser et al. [76] 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 minimises 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 needs only 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.

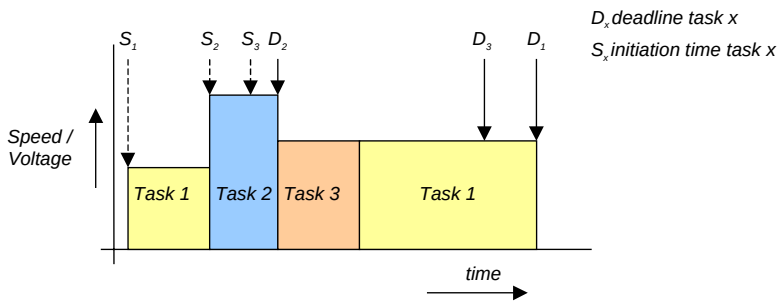


Figure 12: Voltage scheduling under deadline constraints.

Weiser et al. 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.

  Several people have investigated powering down disk drives on portable computers to conserve energy (e.g. [14][86]). Shutting down a disk is a relatively easy task, but unfortunately turning it on is much more expensive in time and energy. Golding et al. tackle the general problem of finding the best way to exploit idle periods during operations of a hard disk [21]. A power management policy that spins down the disk when it is idle can be successful only if it can predict with sufficient accuracy the start time and the duration of the idle interval. The main feature of their model is that it is convenient to spin down the disk as long as it remains in sleep for a period of time longer than a threshold T_{min}.

  It is one thing to turn off a disk when it has been idle for some time, 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.

- *Battery relaxation*

  In recent years batteries have become smaller and they have got more capacity. The capacity of the battery is strongly influenced by the available relaxation time between periods of operation. The relationship between how much of the battery capacity is recovered during an off period depends on the cell chemistry and geometry. 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 [78] 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. However, this may not be synchronised with the current demand of the application.

The energy efficiency of communication protocols can be enhanced through the use of communication protocols that exploit the charge recovery mechanism [13]. Primarily delaying some power consuming activity such as transmission of a packet will perform the discharge demand shaping. Intuitively, it seems clear that the fundamental trade-off here is between delay and energy efficiency.

System decomposition

System decomposition can be applied at various levels: at the computer system internally, or externally by a functional partitioning between a wireless terminal and the network infrastructure. Let us first consider the *internal system decomposition*, that is decomposition of functionality within the mobile.

In a mobile multimedia system many trade-offs can be made concerning the required functionality of a certain mechanism, its actual implementation, and values of the required parameters. In an architecture with reconfigurable modules and data streams, functions can be dynamically migrated between functional modules such that an efficient configuration is obtained. For example, when we consider the transmission of an image over a wireless network, there is a trade-off between image compression, error control, communication, and energy consumption. Functionality can be partitioned between a program running on the general-purpose CPU, dedicated hardware components (like a compressor or error correction device), and field programmable hardware devices (like FPGAs). Of course, the actual trade-off will depend on the particularities of the system, the nature of the data sent, and so on.

The networked operation of a mobile system opens up additional opportunities for decomposition to increase energy efficiency. A careful analysis of the data flow in the system and decomposition of the system functions between wireless terminal and network infrastructure can reduce energy consumption considerably. One opportunity is offloading computation dynamically from the mobile system, where battery energy is at a premium, to remote energy-rich servers in the wired backbone of the network. In essence, energy spent in communication is traded for computation. Partitioning of functions is an important architectural decision, which indicates where applications can run, where data can be stored, the complexity of the terminal, and the cost of the communication service [39]. The key implication for this architecture is that the runtime hardware and software environment on the mobile computer and in the network should be able to support such adaptability, and provide application developers with appropriate interfaces to control it. Software technologies such as proxies and mobile agents, and

hardware technologies such as adaptive and reconfigurable computing are likely to be the key enablers.

A good example of such a decomposition that partitions the computational effort for *video compression* is described by Rabiner [60][61]. Due to the large amount of data needed for video traffic, efficient compression techniques are important when the video frames are transmitted over wireless channels. Motion estimation has been shown to help significantly in the compression of video sequences. However, since most motion estimation algorithms require a large amount of computation, it is undesirable to use them in power constrained applications, such as battery operated wireless video terminals. Since the location of an object in the current frame can be predicted from its location in previous frames, it is possible to optimally partition the motion estimation computation between battery operated portable devices and high powered compute servers on the wired network.

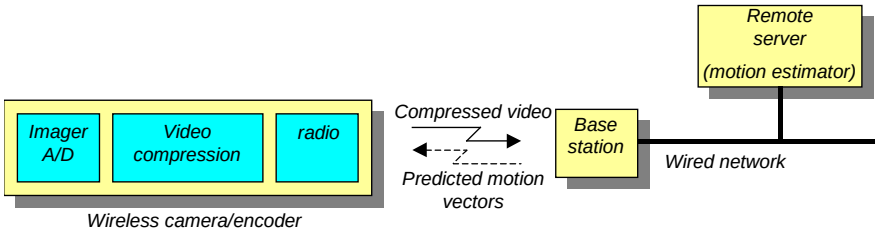


Figure 13: Partitioned video compression.

Figure 13 shows a block diagram of a wireless video terminal in a networked environment. A resource on the wired network with no power constraints can estimate the motion of the sequence based on the previous (decoded) frames and use this information to predict the motion vectors of the current frame. This can achieve a reduction in the number of operations performed at the encoder for motion estimation by over two orders of magnitude while introducing minimal degradation to the decoded video compared with full search encoder-based motion estimation.

Intelligent agents, sometimes called *proxies*, can be used to process control information, or to manipulate user information that is being exchanged between the mobile device and a network-based server. A proxy can be executed in a fixed location, or it may be mobile, and move with the user that it serves. The general benefits of a network-based proxy are that it can execute complex functions, perform aggressive computation and use large amounts of storage on behalf of clients. One example of a complex function that may be performed by a mobile device is the format translation of information sent for display. For example, a wireless web browser may provide a picture in 16 bits colour while the device can only display black and white. To have an end device perform this conversion requires a significant amount of storage and processing. Instead a proxy may perform this conversion, filter out any unusable graphics, and forward only the black and white pixels to the display. The Wireless Application Protocol (WAP) uses a similar approach to be able to support Internet content and services on differing wireless network technologies and device types [88].

Energy is saved because the amount of computation and communication for the mobile is reduced. Other advantages of proxies are that proxies may account for mobiles that are in a disconnected state, and that they provides a solution to the problem of client and network heterogeneity and allow interoperability with existing servers [18][22].

Network-based proxies may thus be used to perform various functions in lieu of the mobile. However, for applications and protocols developed specifically for wireless mobile devices, solutions inherent in their design may be more efficient. For example, the network protocols such as TCP/IP can be implemented more energy efficiently. Certain parts of the network protocol stack can be migrated to servers residing on the fixed network (i.e. the base station). 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 communication protocol with the mobile so that the mobile units can use an internal, lightweight, protocol to communicate with the base station rather than TCP/IP or UDP. The net result is saving in code and energy. In Section 2.5.6 we will exploit the possibilities of energy reduction in communication in more detail.

In the early phases of the design of any part of the system, either hardware or software, the designer needs to experiment with alternative designs. However, energy efficiency is not only a one-time problem that needs to be solved during the design phase. When the system is operational, frequent adaptations to the system are required to obtain an energy efficient system that can fulfil the requirements imposed in terms of a general QoS model. Finding the energy management policy that minimises energy consumption without compromising performance beyond acceptable levels is already a complex problem. If the resources are also flexible, and can adapt their functionality, this problem becomes even bigger.

2.5.5 Applications, compilation techniques and algorithms

Applications

The best policy for deciding when to shut down or wake up a specific part of the system is in general application-dependent, since applications are in the best position to know usage and activity patterns for various resources. Applications play a critical role in the user's experience of a power-managed system. Power management circuits and operating systems that lack application-specific knowledge can only rely on the generic policies discussed above. 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 has resulted 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 penetrate into the power management decision-making system in order to prevent the kinds of user problems described above.

This suggests that operating systems ought to provide application programming interfaces (APIs) so that energy-aware applications may influence the scheduling of the

system's resources. Obviously, careless application's use of the processor and hard disk drastically affects battery lifetime. For example, performing non-essential background tasks in the idle loop prevents the processor from entering a low power state (see for example [41]). So, it is not sufficient to be low power, but the applications running for a system have to be made energy aware as well.

Prefetching and caching of data has been used to improve performance in many applications and file systems. In a mobile environment, these techniques are used by many systems to limit communication and energy consumption caused by mobility, and to improve performance and availability of services. In [79] two systems have been discussed: a file system application and a browsing application.

Code and algorithm transformation

Software design for low power has become an active area of research in the last few years. Software optimisation that properly selects and orders the instructions of a program to reduce the instruction bus activity are based on the simple observation that a given high-level operation can be compiled into different machine instruction sequences. As much of the power consumed by a processor is due to the fetching of instructions from memory, high code density, or even instruction compression, can reduce energy consumption. This is not only because fewer processor cycles are required for a given function, but also because denser code means better cache performance. So, the reduction in bus traffic may be better than linear with decreasing code size [69]. However, this only works well when the execution cycle is not (much) longer. Luckily, power and performance can be improved at the same time by optimising the software. If an algorithm can be optimised to run in fewer cycles, it is faster, and consumes less energy.

Today, the cost function in most compilers is either speed or code size, so the most straightforward way to proceed is to modify the objective function used by existing code optimisers to obtain low-power versions of a given software program. The energy cost of each instruction (determined a priori) must be considered during code optimisation. 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 [75][64]. Research has shown that improvements that can be gained using ARM compiler optimisations are marginal compared to writing more energy efficient source code [64]. The largest energy savings are observed at the inter-procedural level that compilers have not been able to exploit. For DSP processors, low-power compilation techniques have produced more interesting results. In particular, it was shown in [36] that instruction scheduling has sizeable impact on global power consumption. This difference can be explained because in DSP processors the energy consumption is dominated by the functional units in the data-path, hence there is a strong correlation between the executed instruction and the execution unit involved in the computation.

Another technique that can be applied is to reduce the cost of memory accesses. Again, this is a similar objective of current compilers developed in the context of high-performance code generation. Further improvements in the power budget can be achieved by applying techniques that explicitly target the minimisation of the switching activity on the address bus and that best exploit the hierarchy in the memory system as described above.

Recent work has shown that with approaches as described above, a particular Fujitsu digital signal processor could run on 26 percent to 73 percent less power – with no hardware changes [37]. They did it by, among other things, carefully assigning data to specific memory banks and by using packed, instead of unpacked instructions. In the former technique, if operands will be needed together for computations, they are assigned to the same memory bank to take advantage of a double-operand move operation out of the one memory. Since the double-operand move takes only a single cycle, instead of two cycles for two single-operand moves, the access draws less power. For the same reason, using packed, instead of unpacked, instructions also consumes less power: instructions are chosen that reduce the number of execution cycles and so are fundamentally more efficient.

At the algorithm level functional pipelining, retiming, algebraic transformations and loop transformations can be used [49]. The system's essential power dissipation can be estimated by a weighted sum of the number of operations in the algorithm that has to be performed [10]. 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 [43].

2.5.6 Energy reduction in communication

Up to this point we have mainly discussed the techniques that can be used to decrease the energy consumption of digital systems and focussed on computer systems. In this subsection we will discuss some techniques that can be used to reduce the energy consumption that is needed for the (wireless) communication external of the computer. In [24] we give more detailed information.

Sources of energy consumption

In its most abstract form, a networked computer system has two sources of energy drain during operation:

- *Communication*, due to energy spent by the wireless interface. Communication energy is, among others, dictated by the signal-to-noise ratio (SNR) requirements.

- *Computation*, due to (signal) processing and other tasks required during communication. Computation energy is a function of the hardware and software used for tasks such as compression and forward error correction (FEC).

Broadly speaking, minimising energy consumption is a task that will require minimising the contributions of communication and computation, making the appropriate trade-offs between the two. For example, reducing the amount of transmitted data may be beneficial. On the other hand, the computation cost (e.g. to compress the data being sent) might be high, and in the extreme it might be such that it would be better to just send the raw data.

For long distance wireless links, the transmit-communication energy component dominates. However, for short distance wireless links and in harsh environments where much signal processing and protocol computation may be used, the computation component can be significant or dominant.

The wireless network interface of a mobile computer consumes a significant fraction of the total power [73]. Measurements show that on typical applications like web-browsing or handling e-mail, the energy consumed while the interface is 'on' and idle is more than the cost of actually receiving packets. That is because most applications have little demanding traffic needs, and hence the transceiver is *idling* most of the time. The access to the wireless channel is controlled by a MAC protocol. Many MAC protocols for wireless networks are basically adaptations of MAC protocols used in wired networks, and ignore energy issues [39]. For example, random access MAC protocols such as carrier sense multiple access with collision avoidance (CSMA/CA) and 802.11 typically require the receiver to be powered on continually and monitor the channel for traffic. The typical *inactivity threshold*, which is the time before a transceiver will go in the off or standby state after a period of inactivity, causes the receiver to be needlessly in an energy consuming mode for a significant time. Significant time and energy is further spent by the mobile in switching from transmit to receive modes, and vice-versa. In broadcast networks *collisions* may occur (during high load situations). This causes the data to become useless and the energy needed to transport that data to be wasted.

The next step is to *reduce the amount of data*, which must be pushed through the channel. This goal can be reached in a number of ways. One is to reduce the *overhead of a protocol* which influences the energy requirements due to the amount of 'useless' control data and the required computation for protocol handling. The high *error rate* that is typical for wireless links is another source of energy consumption for several reasons. First, when the data is not correctly received the energy that was needed to transport and process that data is spoiled. Secondly, energy is used for error control mechanisms. Finally, because in wireless communication the error rate varies dynamically over time and space, a fixed-point error control mechanism that is designed to be able to correct errors that hardly occur, spoils energy and bandwidth. If the application is error-resilient, trying to withstand all possible errors spoils even more energy for needless error control. Reducing the amount of data is also an application-layer issue. For example, the application might change the compression rate or possibly reduce the data resolution. Instead of sending an entire large full-colour image, one can send black-and-white half-size images with lossy compression.

Network protocol stack

Data communication protocols govern the way in which electronic systems exchange information by specifying a set of rules that, when followed, provide a consistent, repeatable, and well-understood data transfer service. In designing communication protocols and the systems that implement them, one would like to ensure that the protocol is correct and efficient.

Portable devices have severe constraints on the size, the energy consumption, the communication bandwidth available, and are required to handle many classes of data transfer over a limited bandwidth wireless connection, including delay sensitive, real-time traffic such as speed and video. Multimedia applications are characterised by their various media streams. Each stream can have different quality of service requirements. Depending on the service class and QoS of a connection a different policy can be applied to the communication protocol by the application to minimise energy consumption. For example, by avoiding error-control overhead for connections that do not need it and by never transmitting stale data, efficiency is improved. This combination of limited bandwidth, high error rates, and delay-sensitive data requires tight integration of all subsystems in the device, including aggressive optimisation of the protocols that suit the intended application. The protocols must be robust in the presence of errors; they must be able to differentiate between classes of data, giving each class the exact service it requires; and they must have an implementation suitable for low-power portable electronic devices.

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. This example, however, shows that the network protocols of a wireless system can be changed in such a way that it minimises its energy consumption.

Considerations of energy efficiency are fundamentally influenced by the trade-off between energy consumption and achievable Quality of Service (QoS). With the provision of universal roaming, a mobile user will be faced with an environment in which the quality of service can vary significantly within and across different wireless networks. In order to deal with the dynamic variations in networking and computing resources gracefully, both the mobile computing environment and the applications that operate in such an environment need to *adapt* their behaviour depending on the available resources including the batteries. Energy reduction should be considered in the whole system of the mobile and through all layers of the protocol stack, including the application layer. Adaptability of the protocols is a key issue. We will now provide various ways that can be used to reduce energy consumption at the layers of a typical network protocol stack.

- *Physical layer* – At the lowest level we need to apply an energy-efficient radio that can be in various operating modes (like variable RF power and different sleep modes) such that it allows a dynamic power management. Energy can also be saved if it is able to adapt its modulation techniques and basic error-correction schemes. The bandwidth offered by the radio also influences its energy consumption. The energy per bit transmitted or received tends to be lower at higher bit rates. For example, the WaveLAN radio operates at 2Mb/s and consumes 1.8 W, or 0.9 μJ/bit. A commercially available FM transceiver (Radiometrix BIM-433) operates at 40 kb/s and consumes 60 mW, or 1.5 μJ/bit. This makes the low bit-rate radio less efficient in energy consumption for the same amount of data. However, when a mobile has to listen for a longer period for a broadcast or wake-up from the base station, then the high bit-rate radio consumes about 30 times more energy than the low bit rate radio. Therefore, the low bit-rate radio must be used for the basic signalling only, and as little as possible for data transfer.

  To minimise the energy consumption, but also to mitigate interference and increase network capacity, the transmit power on the link should be minimised, if possible.

- *Medium access layer* – In an energy efficient MAC protocol the basic objective is to minimise all actions of the network interface, i.e. minimise 'on-time' of the transmitter as well as the receiver. Another way to reduce energy consumption is by minimising the number of transitions the wireless interface has to make. By scheduling data transfers in bulk, 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. An example of an energy-efficient MAC protocol is E^2MaC [26]. This is a TDMA protocol in which the QoS manager at the base-station schedules all traffic according to the QoS requirements and tries to minimise the energy consumption of the mobiles. The E^2MaC protocol is a subject of Chapter 5.

- *Logical Link Control layer* – Due to the dynamic nature of wireless networks, *adaptive error control* gives significant gains in bandwidth and energy efficiency [86][68]. This avoids applying error-control overhead to connections that do not need it, and it allows to selectively match the required QoS and the conditions of the radio link. Above these error control adaptations, a scheduler in the base-station can also adapt its traffic scheduling to the error conditions of wireless connections to a mobile. The scheduler can try to avoid periods of bad error conditions by not scheduling non-time critical traffic during these periods.

  Flow control mechanisms are needed to prevent buffer overflow, but also to discard packets that have exceeded the allowable transfer time. Multimedia applications are characterised by their various media streams. Each stream can have different quality of service requirements. Depending on the service class and QoS of a connection a different flow control can be applied so that it minimises the required bandwidth and energy consumption. For instance, in a video application it is useless to transmit images that are already outdated. It is more important to have the 'fresh' images. For such traffic the buffer is probably small, and when the connection is hindered

somewhere, the oldest data will be discarded and the fresh data will be shifted into the fifo. Flow control would needlessly spend energy for transmitting 'old' images and flow-control messages. An energy-efficient flow control adapts it control mechanism to the requirements of the connection.

- *Network layer* – 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 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 utilisation 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 [4] 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.
- *Operating system level* – Another way to avert the high cost (either performance, energy consumption or money) of wireless network communication is to avoid use of the network when it is expensive by predicting future access and fetching necessary data when the network is cheap. In the higher level protocols of a communication system caching and scheduling can be used to control the transmission of messages. This works in particular well when the computer system has the ability to use various networking infrastructures (depending on the availability of the infrastructure at a certain locality), with varying and multiple network connectivity and with different characteristics and costs [16]. True prescience, of course, requires knowledge of the future. Two possible techniques, *LRU caching* and *hoarding*, are for example present in the Coda cache manager [30]. In order to effectively support mobile computers, system designers must view the network as a first-class resource, expending CPU and possibly disk resources to reduce the use of network resources during periods of poor network connectivity.

Modern high-performance network protocols require that all network access be through the operating system, which adds significant overhead to both the transmission path (typically a system call and data copy) and the receive path (typically an interrupt, a system call, and a data copy). This not only causes performance problems, but also incurs a significant of energy consumption. Intelligent network interfaces can relieve this problem to some extend. To address the performance problem, several *user-level communication architectures* have been developed that remove the operating system from the critical communication path [8].

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 various layers of the communication stack, unnecessary data copies between successive layers of the protocol stack may be eliminated. This eliminates wasteful data transfers over the global bus, and thus saves much dissipation in buses, multiplexers and drivers. Note, however, that modern, optimised systems, like the x-kernel, avoid data copies between protocol layers religiously.
- Secondly, dedicated hardware can do basic signal processing and can move 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 efficiently 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 communication protocol with the mobile so that the mobile units can use an internal, lightweight, protocol to communicate with the base station rather than TCP/IP or UDP. The net result is saving in code and energy.

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 [68]. 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 in such a way 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).

2.6 Conclusions

As there will become an increasing numbers of portable, battery powered systems, more and more attention will be focused on low-power design techniques. The art of low-power design used to be a narrow speciality in analog circuit design. As the issue of energy efficiency becomes even more pervasive, the battle to use the bare minimum of energy will be fought on multiple fronts: semiconductor technology, circuit design, design automation tools, system architecture, operating system, and application design. It is now appearing in the mainstream digital design community affecting all aspects of the design process. Eventually, the concern for low-power design will expand from devices to modules to entire systems, including application software. 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 and system level. At system level, the system designer can take advantage of power management features where available, as well as decomposed system architectures and programming techniques for reducing power consumption.

Note that some low-power design techniques are also used to design high-speed circuits, and to increase performance. For example, optimised 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 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 a 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 is 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.

Programmability is important for mobile systems because they operate in a dynamically changing environment and must be able to adapt to the new environment. While low-

power solutions are already available for application specific problems, applying these solutions in a reconfigurable environment is a substantially harder problem, since programmable devices often incur significant performance and energy-consumption penalties. The research described in this thesis mainly focuses on higher level re-programmability. The *Chameleon* project [66], that just started and is a spin-off from the Moby Dick project and the research presented in this thesis, will deal with reconfiguration in more depth.

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 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.

References

[1] Abnous A., Rabaey J.: “Ultra-low-power domain-specific multimedia processors”, *VLSI Signal processing IX*, ed. W. Burleson et al., IEEE Press, pp. 459-468, November 1996.

[2] Adam J.: “Interactive multimedia – applications, implications”, *IEEE Spectrum*, pp. 24-29, March 1993.

[3] Bakoglu H. “Circuits, Interconnections, and Packaging for VLSI”, *Addison-Wesley*, Menlo Park, CA, 1990.

[4] Balakrishnan H., et al.: “A comparison of mechanisms for improving TCP performance over wireless links”, Proceedings *ACM SIGCOMM'96*, Stanford, CA, USA, August 1996.

[5] Benini L., De Micheli G.: “Dynamic Power Management, design techniques and CAD tools”, *Kluwer Academic Publishers*, ISBN 0-7923-8086-X, 1998.

[6] Berkel K., et al.: “A fully asynchronous low power error corrector for the DCC player”, *Digest of Technical Papers, International Solid-State Circuit Conference*, pp. 88-89, 1994.

[7] Berkel K. van, Rem M.: “VLSI programming of asynchronous circuits for low power”, *Nat.Lab. Technical Note Nr. UR 005/94*, Philips Research Laboratories, Eindhoven, the Netherlands, 1994.

[8] Bhoedjang, R.A.F., Rühl T., Bal H.E.: “User-level network interface protocols”, *Computer*, November 1998, pp. 53-60.

[9] Burd T.D., Brodersen R.W.: “Energy efficient CMOS microprocessor design”, *Proceedings 28th. annual HICSS Conference*, Jan. 1995, vol. I, pp. 288-297.

[10] Burger D., Goodman J.: “Billion-transistor architectures”, *Computer*, Sept. 1997, pp. 46-47.

[11] Chandrakasan A.P., et al.: “Optimizing power using transformations”, *Transactions on CAD*, Jan. 1995.

[12] Chao K., Wong D.: “Low power considerations in floorplan design”, *1994 International Workshop on low power design*, Napa Valley, CA, pp.45-50, April 1994.

[13] Chiasserini C.F., Rao R.R.: "Pulsed battery discharge in communication devices", *proceedings ACM/IEEE MobiCom'99*, pp. 88-95, August 1999.

[14] Douglis F., Krishnan P., Marsh B.: “Thwarting the power-hungry disk”, *proceedings of USENIX Winter 1994 Technical Conference*, pp.292-306, USENIX Association, 17-21 January 1994.

[15] Dyer C.K.: "Replacing the battery in portable electronics", *Scientific American*, pp. 70-75, July 1999.

[16] Ebling, M.R., Mummert, L.B., Steere D.C.: “Overcoming the Network Bottleneck in Mobile Computing”, *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, Dec. 1994, Santa Cruz, CA.

[17] Flynn M.J.: “What's ahead in computer design?”, *Proceedings Euromicro 97*, pp. 4-9, September 1997.

[18] Fox A., Gribble S.D., Chawathe Y., Brewer E.A.: "Adapting to network and client variation using infrastructural proxies: lessons and perspectives, *IEEE Personal Communications*, pp. 10-19, August 1998.

[19] Frenkil J.: "A multi-level approach to low-power IC design", *IEEE Spectrum*, Volume 35, Number 2, February 1998.

[20] Gao B., Rees D.J.: "Communicating synchronous logic modules", *proceedings 21st Euromicro conference*, September 1995.

[21] Golding R., Bosch P., Staelin C., Sullivan T., Wilkes J.: "Idleness is not sloth", Winter'95 USENIX Conference proceedings, New Orleans, Jan. 1995.

[22] Han R., Bhagwat P., LaMaire R., Mummert T., Perret V., Rubas J.: "Dynamic adaptation in an image transcoding proxy for mobile web browsing", *IEEE Personal Communications*, pp. 8-17, December 1998.

[23] Hauck S.: "Asynchronous design methodologies: an overview", *Proceedings of the IEEE*, Vol. 83, No. 1, pp. 69-93, January 1995.

[24] Havinga, P.J.M., Smit, G.J.M.: "Minimizing energy consumption for wireless computers in Moby Dick", *proceedings IEEE International Conference on Personal Wireless Communication ICPWC'97*, Dec. 1997.

[25] Havinga P.J.M., Smit G.J.M.: "Design techniques for low power systems" *Journal of Systems Architecture*, Vol. 46, Iss. 1, 2000, a previous version appeared as CTIT Technical report, No. 97-32, Enschede, the Netherlands, ISSN 1381-3625

[26] Havinga P.J.M., Smit G.J.M., Bos M.: "Energy-efficient wireless ATM design", *proceedings wmATM'99*, June 2-4, 1999.

[27] Ikeda T.: "ThinkPad Low-Power Evolution", *IEEE Symposium on Low Power Electronics*, October 1994.

[28] Intel486SX: URL: http://134.134.214.1/design/intarch/prodbref/272713.htm.

[29] Kin J., Gupta M., Mangione-Smith W.H.: "The filter cache: an energy efficient memory structure", *Micro* 30, 1997.

[30] Kistler J.J.: "Disconnected operation in a distributed file system", PhD thesis, *Carnegie Mellon University*, School of Computer Science, 1993.

[31] Koegst, M, et al.: "Low power design of FSMs by state assignment and disabling self-loops", *Proceedings Euromicro 97*, pp 323-330, September 1997.

[32] Kozyrakis C.E., Patterson D.A.: "A new direction for computer architecture research", *Computer*, Nov. 1998, pp. 24-32

[33] Landman P.E.: "Low-power architectural design methodologies", Ph.D. thesis, *University of California at Berkeley*, 1994.

[34] Lapsley, P: "Low power programmable DSP chips: features and system design strategies", *Proceedings of the International Conference on Signal Processing*, Applications and Technology, 1994.

[35] Larri G.: "ARM810: Dancing to the Beat of a Different Drum", *Hot Chips 8: A Symposium on High-Performance Chips*, Stanford, August 1996.

[36] Lee M.T.-C et al. V. Tiwari et al. : "Power analysis and low-power scheduling techniques for embedded DSP software", *International Symposium on System Synthesis*, pp. 110-115, Sept. 1995.

[37] Lee M.T.-C, Tiwari V., Malik S., Fujita M. “Power Analysis and Minimization Techniques for Embedded DSP Software”, *IEEE Transactions on VLSI Systems*, March 1997, Vol. 5, no. 1, pp. 123-133.

[38] Leijten J.A.J.: “Real-time constrained reconfigurable communication between embedded processors”, *Ph.D. thesis, Eindhoven University of Technology*, November 1998.

[39] Lettieri P., Srivastava M.B.: “Advances in wireless terminals”, *IEEE Personal Communications*, pp. 6-19, February 1999

[40] Lorch, J.R.,: “A complete picture of the energy consumption of a portable computer”, *Masters thesis, Computer Science, University of California at Berkeley*, 1995

[41] 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.

[42] Lorch, J.R., Smith, A.J.: "Software strategies for portable computer energy management", Report No. UCB/CSD-97-949, Computer Science Division (EECS), *University of California at Berkeley*, May 1997.

[43] Macii, E., Pedram M., Somenzi F.: “High-level power modeling, estimation, and optimization”, *IEEE transactions on computer-aided design of integrated circuits and systems*, Vol. 17, No. 11, pp. 1061-1079, November 1998.

[44] Mangione-Smith, B. 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 (ISLPED) 1996*, Monterey CA, USA, pp. 23-28, August 1996.

[45] Mangione-Smith W.H., et al.: “Seeking solutions in configurable computing”, *IEEE Computer*, pp. 38-43, December 1997.

[46] Mangione-Smith W.H., Hutchings B.L.: “Configurable computing: the road ahead”, *1997 reconfigurable architectures workshop*, 1997.

[47] Martin A.J., Burns S.M., Lee T.K., Borkovic D., Hazewindus P.J.: “The first asynchronous microprocessor: the test results”, *Computer Architecture News*, 17(4):95-110, June 1989.

[48] McGaughy, B: “Low Power Design Techniques and IRAM”, March 20, 1996, URL: http://rely.eecs.berkeley.edu:8080/researchers/brucemcg/iram_hw2.html.

[49] Mehra R., Lidsky D.B., Abnous A., Landman P.E., Rabaey J.M.: “Algorithm and architectural level methodologies for low power”, section 11 in “*Low power design methodologies*”, editors J. Rabaey, M. Pedram, Kluwer Academic Publishers, 1996.

[50] Mehra R., Rabaey J.: “Exploiting regularity for low power design”, *Proceedings of the International Conference on Computer-Aided Design*, 1996

[51] Merkle, R.C.: “Reversible Electronic Logic Using Switches”, *Nanotechnology*, Volume 4, pp 21 - 40, 1993 (see also: http://nano.xerox.com/nanotech/electroTextOnly.html)

[52] Monteiro J, Devadas S., Ashar P., Mauskar A.: “Scheduling techniques to enable power management”, Proceedings of the 33rd Design Automation Conference, pp. 349-352, Las Vegas, Nevada, June 1996.

[53] “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.

[54] Nieuwland A.K., Lippens P.E.R.: "A heterogeneous HW-SW architecture for hand-held multi-media terminals", *proceedings IEEE workshop on Signal Processing Systems*, SiPS'98, pp. 113-122.

[55] Payne R.E.: "Self-Timed FPGA Systems", *Proceedings of the 5th International Workshop on Field Programmable Logic and Applications*, LNCS 975, September 1995.

[56] Pedram M.: "Power minimization in IC design: principles and applications", *ACM Transactions on Design Automation*, Vol. 1, no. 1, pp. 3-56, Jan 1996.

[57] Piguet, C, et al.: "Low-power embedded microprocessor design", *Proceeding Euromicro-22*, pp. 600-605, September 1996.

[58] Rabaey J. et al.: "Low Power Design of Memory Intensive Functions Case Study: Vector Quantization", *IEEE VLSI Signal Processing Conference*, 1994.

[59] Rabaey J., Guerra L., Mehra R.: "Design guidance in the Power Dimension", *Proceedings of the ICASSP*, 1995.

[60] Rabiner W., Chandrakasan A.: "Network-Driven Motion Estimation for Wireless Video Terminals", *IEEE Transactions on Circuits and Systems for Video Technologies*, Vol. 7, No. 4, August 1997, pp. 644-653.

[61] Rabiner W., Chandrakasan A.: "Network-Driven Motion Estimation for Portable Video Terminals", Proc. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)*, April 1997, Vol. 4, pp. 2865-2868.

[62] Semiconductor Industry Association: "The national technology roadmap for semiconductors: Technology needs", *Sematech Inc*., http://www.sematech.org, Austin, USA, 1997.

[63] Sheng S., Chandrakasan A., Brodersen R.W.: "A Portable Multimedia Terminal", *IEEE Communications Magazine*, pp. 64-75, vol. 30, no. 12, Dec., 1992.

[64] Simunic T., Benini L., De Micheli G: "Cycle-accurate simulation of energy consumption in embedded systems", *Proceedings Design Automation Conference DAC 99*, 1999.

[65] Smit J., Bosma M.: "Graphics algorithms on Field Programmable Function Arrays", *proceedings of the 11th EuroGraphics workshop on graphics hardware*, Eds. B.O. Schneider and A. Schilling, pp.103-108, 1996.

[66] Gerard J.M. Smit, Martinus Bos, Paul J.M. Havinga, Sape J. Mullender, Jaap Smit: "Chameleon - reconfigurability in hand-held multimedia computers", *proceedings First International Symposium on Handheld and Ubiquitous Computing*, HUC'99, September 1999.

[67] Smit J., Stekelenburg M., Klaassen C.E., Mullender S., Smit G., Havinga P.J.M.: "Low cost & fast turnaround: reconfigurable graph-based execution units", *proceedings 7th BELSIGN workshop*, Enschede, the Netherlands, May 7-8, 1998

[68] Smit G.J.M., Havinga P.J.M., van Opzeeland M., Poortinga R.: "Implementation of a wireless ATM transceiver using reconfigurable logic", *proceedings wmATM'99*, June 2-4, 1999.

[69] Snyder J.H., et al.: "Low power software for low-power people", *Symposium on low power electronics*, October 1994.

[70] Srivastava M.: "Design and optimization of networked wireless information systems", *IEEE VLSI workshop*, April 1998.

[71] Srivastava M.: "Designing energy effuicient mobile systems", Tutorial during *MobiCom'99*, http://www.janet.ucla.edu/~mbs/tutorials/mobicom99, August 1999.

[72] Stan M.R., Burleson W.P.: “Bus-invert coding for low-power I/O”, *IEEE Trans. VLSI Syst.*, Vol. 3, no. 1, pp. 49-58, 1995.

[73] Stemm, M, et al.: “Reducing power consumption of network interfaces in hand-held devices”, *Proceedings mobile multimedia computing MoMuc-3*, Princeton, Sept 1996.

[74] Tierno J.A., Martin A.J.: “Low-energy asynchronous memory design”, http://www.cs.caltech.edu/~alains/publications/pub.html.

[75] Tiwari V. et al.: “Compilation Techniques for Low Energy: An Overview”, *IEEE Symposium on Low Power Electronics*, October 1994.

[76] 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.

[77] “Minimizing power consumption in FPGA designs”, *XCELL* 19, page 34, 1995.

[78] 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.

[79] Porta La T.F., Sabnani K.K., Gitlin R.D.: “Challenges for nomadic computing: mobility management and wireless communications”, *Mobile networks and applications*, Vol. 1, No. 1, pp. 3-16, August 1996.

[80] Rambus Inc.: “Direct Rambus Memory for mobile PCs”, http://www.rambus.com.

[81] Rashid R.F.: “Personal Computing – the new future”, *keynote speech MobiCom’99*, August 1999.

[82] Sheng S., Chandrakasan A., Brodersen R.W.: “A Portable Multimedia Terminal”, *IEEE Communications Magazine*, pp. 64-75, vol. 30, no. 12, Dec., 1992.

[83] Simunic T., Benini L., De Michelle G.: “Energy-efficient design of battery-powered embedded systems”, *Proceedings ISLPED’99*, 1999.

[84] Truman T.E., Pering T., Doering R., Brodersen R.W.: The InfoPad multimedia terminal: a portable device for wireless information access”, *IEEE transactions on computers*, Vol. 47, No. 10, pp. 1073-1087, October 1998.

[85] Udani S., Smith J.: “The power broker: intelligent power management for mobile computers”, Tech. report MS-CIS-96-12, *Department of Computer Information Science*, University of Pennsylvania, May 1996.

[86] Udani S., Smith J.: “Power management in mobile computing”, Tech. report MS-CIS-98-26, *Department of Computer Information Science*, University of Pennsylvania, August 1996.

[87] Villasenor J., Mangione-Smith W.H.: “Configurable Computing”, *Scientific American*, June 1997.

[88] Wireless Application Protocol Forum Ltd.: “Official Wireless Application Protocol”, *Wiley Computer Publishing*, 1999, http://www.wapforum.org.

[89] Yeap G.K.: “Practical low power digital VLSI design”, *Kluwer Academic Publishers*, ISBN 0-7923-80.

[90] Yeung N. et al.: “The design of a 55 SPECint92 RISC processor under 2W”, *Proceedings of the international solid-state circuits conference ’94*, San Francisco, CA, pp. 206-207, February 1994.

[91] 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.