3

The design of a system architecture for mobile multimedia computers

This chapter[1] discusses the system architecture of a portable computer, called Mobile Digital Companion, which provides support for handling multimedia applications energy efficiently. Because battery life is limited and battery weight is an important factor for the size and the weight of the Mobile Digital Companion, energy management plays a crucial role in the architecture. As the Companion must remain usable in a variety of environments, it has to be flexible and adaptable to various operating conditions.

The Mobile Digital Companion has an unconventional architecture that saves energy by using system decomposition at different levels of the architecture and exploits locality of reference with dedicated, optimised modules. The approach is based on dedicated functionality and the extensive use of energy reduction techniques at all levels of system design. The system has an architecture with a general-purpose processor accompanied by a set of heterogeneous autonomous programmable modules, each providing an energy efficient implementation of dedicated tasks. A reconfigurable internal communication network switch exploits locality of reference and eliminates wasteful data copies.

3.1 Introduction

One of the most compelling issues in mobile computing is to keep the energy consumption of the mobile low. This chapter discusses the system architecture of a portable computer, called *Mobile Digital Companion*, which provides support for handling multimedia applications energy efficiently. The *Mobile Digital Companion* was designed as part of the MOBY DICK project [62][50]. This project addresses fundamental

[1] Major parts of this chapter have been presented in two presentations at the first *Euromicro summer school on mobile computing '98,* August 1998 [30][62].

issues in the architecture, design and implementation of low-power hand-held computers, with particular emphasis on energy conservation.

3.1.1 Mobile systems today

The research community and the industry have expended considerable effort toward mobile computing and the design of portable computers and communication devices. Inexpensive gadgets that are small enough to fit in a pocket (like PDAs, palmtop computers and digital cameras) are joining the ranks of notebook computers, cellular phones and video games. Present day portable computers run most common interactive applications like word processors and spreadsheets without any noticeable computation delay. These devices now support a constantly expanding range of functions, and multiple devices are converging into a single unit [37]. Personal computers are becoming an integral part of daily life, as portable appliances such as wristwatches and cellular phones have become over the last few years. The emergence of wireless communication and the enormous improvements in technology that allows us to integrate many functions in one chip has opened up many possibilities for mobile computing. Communication, data processing and entertainment will be supported by the same platform, enhanced by the world-wide connectivity provided by the Internet.

We first take a brief look at the various mobile systems on the market today. Note that many of these systems have no built-in wireless networking capability, but rather rely on an external wireless modem for wireless connectivity. The wireless modem is in general based on a cellular phone, or on wireless LAN (WLAN) products. Wireless LANs are generally intended for short-range (several hundred meters) indoor use, as opposed to the outdoor several-kilometres range of cellular systems. Wireless LANs have a higher data rate than cellular phones, on the order of megabits per second, and the size is smaller, the power consumption is comparable.

Current mobile systems can be classified into the following categories based on their functions and form factors.

- *Laptops* – Laptops are not really mobile systems since they are too large and too heavy. In essence they are just battery operated small desktop machines. Wireless communication is generally based on WLAN products that can be plugged in as a PC-card.
- *Pen tablets* – Pen tablets can be viewed as laptops without keyboards. Interaction with the pen tablet is through pen input. In most cases, the pen replaces the mouse as pointer device. Some tablets have an internal radio modem, whereas others require an external radio modem. Generally spoken, these terminals are no different from the average desktop.
- *Virtual books* – Recently several products have been introduced that replace paper as the medium for reading and browsing a wide variety of material [13][59][65]. These systems have good quality displays, and a rather conventional architecture. User input is limited to a few buttons, and a pen.

- *Handheld Personal Computers (HPC)* – Systems of this category are basically miniature laptops. They are characterised by a reduced form factor keyboard and a half-VGA resolution display. They usually run reduced versions of Windows applications, including word processing, presentation, and scheduling software. The communication is usually a wire-line modem and an infrared port.
- *Personal Digital Assistants (PDA)* – the PDA is generally a monolithic device without a keyboard (although some have small sized keyboards) and fits in the user's hand. As such, pen input is the norm, and handwriting recognition is common. Communication abilities involve a docking port or serial port for connecting to and synchronising with a desktop computer, and possibly a modem.
- *Smart phones* – Although cellular phones may have several peripheral functions like a calculator, date book, or phone book, they are foremost a communication tool. Combination devices like the Nokia 9000 are essentially PC-like devices attached to a cellular phone.
- *Wireless terminal* – These systems are basically nothing more than the wireless extended input and output of a desktop machine which acts as the server. These systems are designed to take advantage of high-speed wireless networking to reduce the amount of computation required on the portable.

It will be clear that current mobile systems are primarily either data processing terminals or communication terminals. The trend in data processing terminals has been to shrink a general-purpose desktop PC into a package that can be conveniently carried. Even PDAs have not ventured far from the general-purpose model, neither architectural nor in terms of usage model.

3.1.2 The future: Mobile Digital Companion

Topic of this research is the architecture of a future handheld device, called *Mobile Digital Companion* (in this thesis also referred to as *Companion*). A *Mobile Digital Companion* will be a personal machine, and users are likely to become quite dependent on it.

The *Mobile Digital Companion* is a small personal portable computer and wireless communications device that can replace cash, cheque book, passport, keys, diary, phone, pager, maps and possibly briefcases as well [50]. It will resemble a PDA, that is, it looks like a normal PDA, but the functionality and typical use of the system are very different. Typical applications of a *Mobile Digital Companion* are diary, e-mail, web browsing, note-taking, walkman, video player and electronic payments. The *Mobile Digital Companion* is a hand-held device that is *resource-poor*, i.e. small amount of memory, limited battery life, low processing power, and connected with the environment via a (wireless) network with variable connectivity. Our primary objective in designing the architecture has been to support a wide variety of applications for mobile devices that make efficiently use of the available resources. Such companions must meet several major requirements: high performance, energy efficient, a notion of Quality of Service (QoS), small size, and low design complexity.

The *Mobile Digital Companion* is more than just a small machine to be used by one person at a time like the traditional organisers and desktop assistants. We distinguish two types of systems: 'desktop companions' and '*Mobile Digital Companions*'. A desktop companion is a handheld machine that is designed to give roaming users access to their business data and applications while on the road. Desktop companions are designed and optimised for compatibility and communication with the user's desktop machine(s), e.g. via modem, infrared or a docking station. A typical example of a desktop companion is a PDA or (sub)notebook running Windows CE [26].

The *Mobile Digital Companion* extends the notion of a desktop companion in several ways.

- It will run applications typically found in desktop companions, but it will also run other applications using *external public services*. A *Mobile Digital Companion* interacts with the environment and so is part of an open distributed system. It needs to communicate with – possibly hostile – external services under varying communication and operating conditions, and not only to its desktop 'master'.
- *Multimedia computing* will also be an essential part of the *Mobile Digital Companion*. If a mobile computer has to be used for every day work, then multimedia devices, such as audio and video have to be included in the system. Nowadays, there are several portable multimedia devices available (digital cameras, MP3man, etc.), but all these systems are no more than dedicated devices. What lacks is a good integration between all these devices.
- All current desktop companions have communication facilities to communicate with the desktop master. However, as the dependence on network-accessible information storage and computation increases, the desire to ubiquitously access the network requires a much more *sophisticated wireless networking* capability. The network access should support heterogeneity in many dimensions (transport media, protocols, data-types, etc.).

The most important factors, which will determine the success of the *Mobile Digital Companion*, are the utility and convenience of the system. An important feature will be the interface and interaction with the user: voice and image input and output (speech and pattern recognition) will be key functions. The use of real-time multimedia data types like video, speech, animation and music greatly improve the usability, quality, productivity, and enjoyment of these systems. Multimedia applications require the transport of multiple synchronised media streams. Some of these streams (typically video streams) have high bandwidth and stringent real-time requirements. These applications also include a significant amount of user interaction. Most of the applications we consider require not only a certain Quality of Service for the communication (like high bandwidth and low latency), but also a significant amount of computing power. The compute requirements stem from operations such as compression/decompression, data encryption, image and speech processing, and computer graphics.

The Mobile Digital Companion is thus quite a versatile device. Nevertheless these functions have to be provided by relatively small amount of hardware because a main requirement for the Companion is small size and weight. As most current battery research does not predict a substantial change in the available energy in a battery, energy efficiency plays a crucial role in the architecture of the *Mobile Digital Companion*. An integrated solution that reduces chip count is highly desirable.

3.1.3 Approach

The approach to achieve a system as described above is to have autonomous, reconfigurable modules such as network, video and audio devices, interconnected by a switch rather than by a bus, and to offload as much as work as possible from the CPU to programmable modules that are placed in the data streams. Thus, communication between components is not broadcast over a bus but delivered exactly where it is needed, work is carried out where the data passes through, bypassing the memory. Modules are autonomously entering an energy-conservation mode and adapt themselves to the current state of resources, the environment and the requirements of the user. The amount of buffering is minimised, and if it is required at all, it is placed right on the data path, where it is needed. To support this, the operating system must become a small, distributed system with co-operating processes occupying programmable components – like CPU, DSP, and programmable logic – among which the CPU is merely the most flexibly programmable one.

The interconnect of the architecture is based on a switch, called *Octopus*, which interconnects a general-purpose processor, (multimedia) devices, and a wireless network interface. The *Octopus* switch is subject of Chapter 4. Although not uniquely aimed at the desk-area, our work is related to projects like described in [4][20] and [32] in which the traditional workstation bus is replaced by a high speed network in order to eliminate the communication bottleneck that exists in current systems.

3.1.4 Outline

We first indicate in Section 3.2 the main challenges in mobile system design which will provide the motives why there is a need to revise the system architecture of a portable computer. Section 3.3 then describes the philosophy behind the architecture of the *Mobile Digital Companion*, and introduces the various basic mechanisms used: the connection-centric approach, the timing control, the Quality of Service framework, and finally presents the basic system architecture. Then we will give an overview of the state of the art in mobile multimedia computing in Section 3.4. Finally, we present the summary and conclusions in Section 3.5.

3.2 Design issues of mobile systems

There is a new and growing class of users whose primary computing needs are to access the information infrastructure, computing resources, and real-time interactive systems as well as direct communications with other people. These applications, which are communication oriented rather than computation oriented, is the motivation for a re-examination of the requirements of the system architecture and the hardware that is needed. These applications require a personal mobile digital companion that primarily has support for high bandwidth real-time communication and multimedia capabilities [70]. High-performance general-purpose computing is not a prominent requirement.

Recent improvements in circuit technology and software development have enabled the use of real-time data types like video, speech, animation, and music. As mobile computers evolve, support for multimedia-rich applications will become the standard. It is expected that by 2000 90 percent of the desktop cycles will be spent on multimedia applications [16].

The computer industry has made enormous progress in the development of mobile computing and the design of portable computers. This is partly due to recent advances in technology. These systems are generally based on architectures of high performance personal computers that have some provisions for wireless computing and a rudimentary form of power management. In this section we will show that such an approach is not sufficient if we want to be able to have a hand-held machine with multimedia capabilities that can be used conveniently in a wireless environment.

3.2.1 Mobility

The emergence of novel multimedia applications and services that leverage the growth in mobile computing depends on the availability of a flexible broadband wireless infrastructure. Key technical issues of this infrastructure include Quality-of-Service control and application software integration. Mobile systems will have a set of challenges arising from the diverse data types with different quality-of-service (QoS) requirements they will handle, their limited battery resources, their need to operate in environments that may be unpredictable, insecure, and changing, and their mobility resulting in changing set of available services.

The following are the key technological challenges that we believe will need to be addressed before mobile systems like the Mobile Digital Companion will become real.

- *Energy efficiency* – As the current portable computers have shown to be capable of assisting mobile users in their daily work, it is becoming increasingly evident that merely increasing the processing power and raising raw network bandwidth does not translate to better devices. Weight and battery life have become more important than pure processing speed. Energy consumption is becoming the limiting factor in the amount of functionality that can be placed in portable computers like PDAs and laptops.
- *Infrastructure* – The design of mobile systems cannot be done in isolation. The mobile system of the future is likely to be designed to operate autonomously, but it is also very likely that it relies on an external infrastructure to access information of any kind. The mobile will likely encounter many, very diverse environments and various network infrastructures. Furthermore, mobiles may vary along many axes, including screen size, colour depth, processing power, and available functions. Servers (or proxy agents that are placed between mobiles and servers) can perform

computation and storage on behalf of clients. Partitioning of functions between the wireless system and servers residing on the network is an important architectural decision that dictates where applications can run, where data can be stored, the complexity of the terminal, and the cost of communication services.

- *Adaptability* – Wireless mobile systems face many different types of variability in their environment in both the short and the long term. Mobile systems will need the ability to adapt to these changing conditions, and will require adaptive radios, protocols, codecs and so on. Adaptive error control and adaptive compression are examples of such techniques.
- *Reconfigurability* – To combat a higher degree of variations in operational environment than is possible with adaptable systems, reconfigurable architectures can be used that allow new software and hardware functions to be downloaded. Thus rather than changing parameters of algorithms to current conditions, an entirely new set of protocols and algorithms can be used. An alternative approach to adapt to a change in environment would be to have a mobile system with all possible scenarios built-in. Such multimode systems become costly, and relatively inflexible.
- *Security* – When computers become more involved in people's personal and business activities security i.e. confidentiality, privacy, authenticity and non-repudiation become important concerns. Judicious application of cryptography can satisfy these concerns, provided systems provide a secure environment for users in which the appropriate cryptographic algorithms can do their work without any risk of compromising or losing keys or confidential data.
- *User interfaces* – Traditional keyboards and display based interfaces are not adequate for the mobile systems of the future because of the required small size and weight of these system. Instead, intrinsically simpler interfaces based on speech, touch, pen and so forth are more likely to be used and more adequate to the small form factors of these systems. Because these systems will be consumer appliances that are used by non-experts, the complex environment should remain hidden from the user, or presented at a level that can easily be understood by the user.

In the remainder we shall focus on the issues that are related to energy consumption, i.e. energy-efficiency and adaptability.

3.2.2 Multimedia

The systems that are needed for multimedia applications in a mobile environment must meet different requirements than current workstations in a desktop environment can offer. The basic characteristics that multimedia systems and applications needs to support are [17]:

- *Continuous-media data types* – Media functions typically involve processing a continuous stream of data, which implies that temporal locality in data memory accesses no longer holds. Remarkably, data caches may well be an obstacle to high performance and energy efficiency for continuous-media data types because the processor will incur continuous cache-misses.

- *Provide Quality of Service (QoS)* – Instead of providing maximal performance, systems must provide a QoS that is sufficient for qualitative perception in applications like video.
- *Fine-grained parallelism* – Typical multimedia functions like image, voice and signal processing require a fine-grained parallelism in that the same operations across sequences of data are performed. The basic operations are relatively small.
- *Coarse-grained parallelism* – In many applications a pipeline of functions process a single stream of data to produce the end result.
- *High instruction reference locality* – The operations on the data demonstrate typically high temporal and spatial locality for instructions.
- *High memory bandwidth* – Many multimedia applications require huge memory bandwidth for large data sets that have limited locality.
- *High network bandwidth* – Streaming data – like video and images from external sources – requires high network and I/O bandwidth.

Distributed multimedia applications running in a mobile environment have a number of special characteristics. Many future wireless mobile systems will operate in various, relatively unregulated environments such as home and workplace LANs with time varying interference levels. Existing cellular telecommunication networks can also be used to provide wireless access to wired computer networks. Applications cannot rely on the wireless network to provide high throughput or fast response times. Two of the most fundamental characteristics are:

- A *heterogeneous processing environment* (including relatively low-power mobile hosts) and,
- *rapid and massive fluctuations* in the quality of service provided by the underlying communication infrastructure.

QoS control is a key feature for efficient utilisation of resources in wireless networks supporting mobile multimedia. Traditional static resource-allocation models lack flexibility, and thus cope poorly with multimedia interactivity and session mobility.

The challenge is to maintain a high perceived end-to-end quality without limiting applications to the point where they are no longer useful. Multimedia networking requires at least a certain minimum bandwidth allocation for satisfactory application performance [58]. The minimum bandwidth requirement has a wide dynamic range depending on the users' quality expectations, application usage models, and applications' tolerance to degradation. In addition, some applications can gracefully adapt to sporadic network degradation while still providing acceptable performance. For example, while video-on-demand applications may in general tolerate bit rate regulations within a small dynamic range, applications such as teleconferencing may have a larger dynamic range for bit rate control. Other multimedia applications may allow a larger range of bit rate control by resolution scaling.

3.2.3 Limitation of energy resources

Although current portable computers have shown to be capable of assisting the mobile user in their daily work, it is becoming increasingly evident that merely increasing the processing power and raising raw network bandwidth does not translate to better devices. Weight and battery life has become more important than pure processing speed. These two factors are related by battery size: to operate a computer for a longer time without recharging, we need a larger, heavier, battery. The limitation is therefore the total amount of electric energy stored in that battery that is available for operation. Battery technology has improved at a glacial pace compared to the pace at which the amount of processing power in mobile systems is increasing while their size is decreasing. Energy consumption is becoming the limiting factor in the amount of functionality that can be placed in portable computers like PDAs and laptops.

To extend battery life, we have to design the system to be more efficient in the way it uses this energy. However, even today, research is still focussed on performance and circuit design. Due to fundamental physical limitations, progress towards further energy reduction will have to be found beyond the chip-level. Key to energy efficiency in future mobile systems will be the higher levels: energy-efficient architectures and protocols, energy aware applications, etc.

The vast majority of energy-critical electronic products are far more complex than a single chip. In most electronic products, the digital components consume a fraction of the energy consumed. Analog, electro-mechanical and optical components are often responsible for a large contributions to the power budget of a portable computer [42]. One of the most successful techniques employed by designers at the system level is *dynamic power management*, in which parts of the system have different energy modes, and can even be completely powered down.

3.2.4 System architectural problems

Within the traditional design of a mobile, a number of problem areas in hardware and software architectures can be identified concerning the energy consumption [8]. We will just mention a few.

- A key issue is the lack of interaction between hardware facilities for energy management (power saving modes, device interrupts that ‘wake up’ the CPU, etc.) and the operating system and application software. In particular, the device drivers, the operating system and the applications attempt to autonomously control the hardware. This mis-coordination causes inexplicable erratic behaviour. Examples of this problem are unexpected screen blanks during a presentation, or when the disk spins up and spins down unannounced (causing annoying delays).
- Second, opportunities for saving energy are not exploited because devices are controlled at a too low level, ignoring high-level information on what the user actually needs during system operation.
- Third, applications assume that the computer is always on. This assumption often causes excessive energy consumption. For example, polling cycles, when an

application is waiting for a response, are very inefficient from an energy point of view.

- Finally, the current operating system software and networking software emphasises flexibility and performance, and is constructed from components developed by independent groups. Within system design a key role lies in the development of interfaces. A good working practise is to define interfaces in a hierarchical way, since the complexity of the system is reduced to manageable proportions. Such an approach is also directed to by standardisation approaches like the ISO/OSI network layer structure. However, the result of this flexibility and this development approach is that in many cases numerous unnecessary data copies occur between different modules. Non-copying protocol stacks do exist [67], but are not widely used. Operations such as data copying, servicing of interrupts, context switches, software compression, are in current systems often responsible for poor performance and high energy consumption. Not well designed network protocols that do not efficiently make use of one of the most energy demanding devices of the mobile, the wireless interface, waste also a lot of energy.

Our vision is that there is a vital relationship between hardware architecture, operating system architecture, applications' architecture and human-interface architecture, where each benefits from the others: the applications can adapt to the power situation if they have an appropriate operating system API for doing so; the operating system can minimise the energy consumption by keeping as many as components turned off as possible; the hardware architecture can be designed to route data paths in such a way that, for specific functions, only a minimum of components need to be active.

3.2.5 System level integration

The design flow of a system consists of various levels of abstraction. By carefully designing all components that make up the mobile system (i.e. the hardware components, the architecture, the operating system, the protocols, and the applications) in a coherent and integrated fashion, it is possible to minimise the overhead resulting from the use of these operations and reduce the energy consumption. Any single application, device driver, or hardware module does not have sufficient knowledge of the status of the entire system to effectively make autonomous decisions concerning energy management.

An important aspect of the design flow is the relation and feedback between the levels. Given a design specification, a designer is faced with several different choices on different levels of abstraction. The designer has to select a particular algorithm, design or use an architecture that can be used for it, and determines various parameters such as supply voltage and clock frequency. This *multi-dimensional design space* offers a large range of possible trade-offs. The most effective design decisions derive from choosing and optimising architectures and algorithms at the highest levels. It has been demonstrated by several researchers [14] that system and architecture level design decisions can have dramatic impact on power consumption. However, when designing a system it is a problem to predict the consequences and effectiveness of design decisions because implementation details can only be accurately modelled or estimated at the technological level and not at the higher levels of abstraction.

The ability to integrate diverse functions of a system on the same chip provides the challenge and opportunity to do system architecture design and optimisations across diverse system layers and functions. Especially a mobile computing device that combines multimedia computing and communication functions exemplifies the need for system level integration. Functions ranging from audio and video processing, radio modem, wireless interface, security mechanisms, and user interface oriented applications have to be integrated in a small portable device with a limited amount of energy. Information generated by a device or an application has to traverse and be processed at all these layers, providing the system architect with a rich design space of trade-offs.

3.2.6 Programmability and adaptability

As mobile computers must remain usable in a variety of environments, they have to support different encoding and encryption schemes and protocols to conform to different network standards, and to adapt to various operating conditions. A mobile computer will therefore require a large amount of circuits that can be customised for specific applications to stay versatile and competitive. *Programmability* and *adaptability* is thus an important requirement for mobile systems, since the mobiles must be flexible enough to accommodate a variety of multimedia services and communication capabilities and adapt to various operating conditions in an (energy) efficient way.

The requirement for programmability in systems on a chip is also triggered by economical reasons [72]. A well designed ASIC will solve the specific problem for which it was designed, but probably not a slightly modified problem introduced after the design was finished. Furthermore, even if a modified ASIC can be developed for the new problem, the original hardware circuits may be too highly customised to be reused in successive generations. Moreover, 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 and new standards are emerging quite fast, new chips need to be designed very often.

3.2.7 Discussion

Basically, there are two types of computer devices for use on the road: the palm-top computer and the notebook computer. Palm tops are mainly used for note-taking, electronic appointment books, and address books. Notebook computers are battery powered personal computers, and the current architectures for mobile computers are strongly related to the architecture of high-performance workstations. Both the notebook and the personal computer generally use the same standard PC operating system such as Windows 98 or Unix, same applications, use the same communication protocols and use the same hardware architecture. The only difference is that portable computers are smaller, have a battery, a wireless interface, and sometimes use low-power components. The problems that are inherent to mobile computing are either neglected (e.g. the communication protocols are still based on TCP/IP, even though these behave poor in a wireless environment [6]), or tried to solve with brute force neglecting the increase in energy consumption (e.g. extensive error control, or software decompression). Adaptability and programmability should be major requirements in the design of the architecture of a mobile computer.

In the near future other small electronic gadgets like Web-phones, MP3 man, games and digital cameras will be integrated with the portable computers in a *personal mobile computing* environment. This not only leads to greater demand for computing power, but at the same time the size, weight and energy consumption must be small.

We are entering an era in which each microchip will have billions of transistors. 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 complicated versions of the kind built today [5]. However, simply shrinking the data processing terminal and radio modem, attaching them via a bus, and packaging them together does not alleviate the architectural bottlenecks. The real design challenge is to engineer an integrated mobile system where data processing and communication share equal importance and are designed with each other in mind. Connecting current PC or PDA designs with an off-the-shelf communication subsystem, is not the solution. One of the main drawbacks of merely packaging the two is that the energy-inefficient general-purpose CPU, with its heavyweight operating system and shared bus, becomes not only the center of control, but also the center of data flow in the system [41].

Clearly, there is a need to revise the system architecture of a portable computer if we want to have a machine that can be used conveniently in a wireless environment. A system level integration of the mobile's architecture, operating system, and applications is required. The system should provide a solution with a proper balance between flexibility and efficiency by the use of a hybrid mix of general-purpose and the application-specific approaches.

3.3 The system architecture of a Mobile Digital Companion

In this section we describe the architecture of the *Mobile Digital Companion*. The properties to be achieved by the architecture are:

1. the *flexibility* to handle a variety of (multimedia) services and standards and
2. the *adaptability* to accommodate to its current environment for the changing conditions in communication connectivity, required level of security, and available resources.
3. Configuration parameters can be adapted according to the *QoS requirements*. The components of architecture should be able to adapt their behaviour to the current

environment and requirements to handle the required tasks efficiently. In doing this, the system should be fully aware of its *energy consumption.*

The difficulty in achieving all requirements into one architecture stems from the inherent trade-offs between flexibility and energy consumption, and also between performance and energy consumption. Flexibility requires generalised computation and communication structures that can be used to implement different kinds of algorithms. While conventional architectures (like used in current laptops) can be programmed to perform virtually any computational task, they achieve this at the cost of high energy consumption. Using system decomposition at different levels of the architecture and exploiting locality of reference with dedicated, optimised modules much energy can be saved.

3.3.1 Approach

The scope of this section is the architecture of systems hardware, firmware and software in general, and the following issues in particular:

- Eliminate as much as possible the CPU as an active component in all data streams. In particular we aim to eliminate the active participation of the CPU in media transfers between components such as network, display and audio system (e.g. when the companion functions as a phone, walk-man, TV, or electronic newspaper). Unlike a local CPU architecture, in which I/O peripherals enhance the functionality of the core processor, our goal was to design intelligent peripherals that are capable of processing I/O events and can manage data streams without relying on a centralised processor.
- Eliminate as much as possible memory as the intermediate station for all data transfers between devices. The energy required to transfer and store the data is wasted if the data only occupies memory in transit between two devices (e.g. network and screen or network and audio).
- Use dynamic programmable and adaptable devices that convert incoming or outgoing data streams, in particular network, security, display and audio devices. Because they are programmable, they can handle different data encoding standards and communication protocols autonomously. This has two effects. First, devices can be designed to communicate directly with each other, instead of requiring CPU intervention for adapting data streams.
    - A display device will convert between, for example, MJPEG-compressed data and pixel data. Multimedia applications can benefit from compression as a means of saving (energy wasting) network bandwidth, but require a low power platform for the necessary calculation.
    - A network device will convert between byte streams used internally and, for example, TCP/IP packet streams. Network protocol stacks can be installed on the network interface device, or even on the base station, where they can handle much of the communication functions while the CPU is turned off.
    - Security protocols can be run in an environment beyond the direct control of

the operating system or applications. Regular software is prone to many forms of attack (viruses, Trojan horses, bugs).

The second effect is that, for a large number of these data-conversion functions (or filter functions), digital signal processors (DSPs), field-programmable hardware, or dedicated hardware are both faster and more energy efficient than general-purpose CPUs.

To limit the communication overhead and the required buffering, the granularity of the tasks on the devices is rather coarse, and the application is partitioned in large blocks. The programmability of each device (or *module*) is more fine-grained and is controlled by the individual autonomous module. The module application can be partitioned over various computational resources, based on the granularity of their application. The proposed architecture of the *Mobile Digital Companion* is shown in Figure 1. The figure shows a typical system with Processor module, Network module, Display module, Camera module, and Audio module, all interconnected by a switching fabric (the *Octopus* switch).

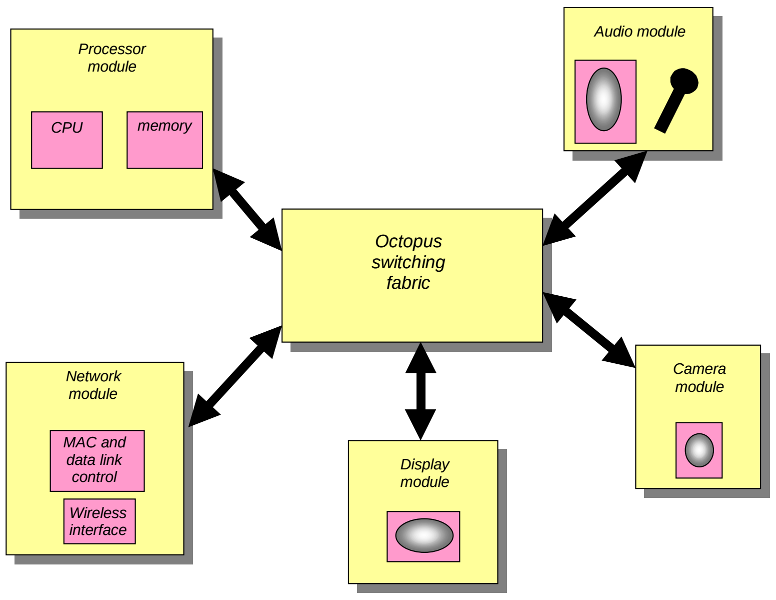

Figure 1: A typical Mobile Digital Companion architecture.

The system has a number of premises:

- An architecture with a general-purpose processor accompanied by a set of heterogeneous programmable modules, provides an energy efficient implementation of dedicated tasks.
- Communication between modules is based on *connections*. Connections are associated with a certain QoS. This identifier provides the mechanism to support lightweight protocols that provide data-specific transport services.
- A reconfigurable internal communication network exploits locality of reference and eliminates wasteful data copies. The data paths through the switch only consume energy when data is being transferred, leaving most of the switch turned off nearly all the time.
- A system design that avoids wasteful activity: e.g. by the use of autonomous modules that can be powered down individually and are data driven.
- A wireless communication system designed for low energy consumption by using intelligent network interfaces that can deal efficiently with a mobile environment, by using a energy aware network protocol, and by using an energy efficient MAC protocol that minimises the energy consumption of network interfaces [31].
- A Quality of Service framework for integrated management of the resources of the *Mobile Digital Companion* in which each module has its own – dedicated – local power management. The operating system will control the power states of devices in the system and share this information with applications and users.

The Mobile Digital Companion is quite a versatile device. Nevertheless these functions can be provided by relatively little hardware. All modules are programmable, but with the exception of the processor module, not as easily or flexibly programmable as conventional CPUs. The components of the modules in the prototype encompass (micro)processors, DSPs and programmable logic (Field Programmable Gate Arrays (FPGA), or Field Programmable Function Arrays (FPFA) [64]). Ultimately, all these components should be integrated into one large VLSI chip (a system-on-a-chip).

3.3.2 Philosophy

Our approach is based on dedicated functionality and the extensive use of energy reduction techniques at all levels of system design. We will use these techniques throughout the design of the *Mobile Digital Companion*, including technological level, architecture level, and system level. To preserve the locality inherent in the application or algorithm a hierarchical-granularity architecture is used that matches the computational granularity to the required operations.

The two main themes that can be used for energy reduction at system level are to *avoid waste*, and to *exploit locality of reference*.

Avoiding waste seems obvious, but in the design of a system it is difficult to avoid waste at various levels in the system. The reason for this is not only the carelessness of the designer, but is also due to the complexity of systems and the relations between the various levels in the system. What is needed is a proper model in which the consequences of a design decision for other parts in the system can be predicted.

The component that contributes significantly to the total energy consumption of a system is the interconnect. Experiments have demonstrated that in chip-designs, about 10 to 40% of the total power may be dissipated in buses, multiplexers and drivers [47]. This amount can increase dramatically for systems with multiple chips due to large off-chip bus capacitance.

The amount of energy required for the transport of data can be reduced by using special memory interfaces (e.g. the Rambus memory technology [56]), or by using on-chip (cache) memory. However, as mentioned before, data caches in processors for multimedia applications are of little use, and may well become an obstacle to high performance and low power, because these media functions typically involve processing a continuous stream of input [37], thereby effectively emptying the cache of useful processor data. The temporal locality property in data memory access does not hold for such data traffic.

As already described in Chapter 2 there are two properties of algorithms important for reducing interconnect power consumption: locality and regularity.

- *Locality* relates to the degree to which a system or algorithm has natural isolated clusters of operation or storage with a few interconnections between them. Partitioning the system or algorithm into spatially local clusters ensures that the majority of the data transfers take place within the clusters and relatively few between clusters. Localisation reduces the communication overhead in processors and allows the use of reduced sized transistors, which results in a reduction of capacitance. The result is that the local buses are shorter and more frequently used than the longer highly capacitive global buses.
- *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.

Most of the techniques for reducing energy consumption can be applied on general-purpose computing. However, for multimedia applications in particular, there is a substantial reduction in energy consumption possible as the computational complexity is high and they have a regular and spatially local computation. Also, the communication between modules is significant. Improving the energy efficiency by exploiting locality of reference and using efficient application-specific modules therefore has a substantial impact on a system like the *Mobile Digital Companion*.

Locality of reference is exploited at several levels. The main philosophy used is that operations on data should be done at the place where it is most energy efficient and where it minimises the required communication. This can be achieved by matching computational and architectural granularity. In the system we have a *hierarchical granularity* in which we differentiate three main grain-sizes of operations:

- *fine grained* operations in the modules that perform functions like multiply and addition,
- *medium grained* operations are the functions of the modules. These functions are dedicated to the basic functionality of the module, e.g. the display module decompresses the video-stream.
- *course grained* operations are those tasks that are not specific for a module and that can be performed by the CPU module, or even on a remote compute server.

We use a *micro-distribution* to migrate tasks between functional modules within the *Mobile Digital Companion*, and a *macro-distribution* to migrate tasks between modules in the external world and on the portable system. In the latter approach certain functions are migrated from the portable system to a remote server that has plenty of energy resources. The remote server handles those functions that cannot be handled efficiently on the portable machine. A typical example of a macro-distribution can be utilised for the network protocol handling. Several researchers have showed that some network protocols perform badly over wireless channels [6]. A solution can be to split the connection in a separate wireless connection between the mobile and a (base)station on the fixed network, and a different connection over the wired network. The base-station can than perform part of the network protocol stack in lieu of the mobile, and use a dedicated and energy efficient protocol over the wireless channel. In such a system it is also simpler and efficient to adapt the protocols for the specific environment it is used in. For example, the network module of the *Mobile Digital Companion* is capable of adapting its error control, its flow control, and its scheduling policy to the current environment and requirements of the system and applications [31].

These principles have lead to our architecture that is capable of handling media-streams efficiently.

3.3.3 Memory-centric versus connection-centric

The system architecture of the *Mobile Digital Companion* is connection (or media) centric, which means that the media type of the traffic drives the data flow in the system using connections. For example, the video traffic from the network interface is transferred directly to the display, without interference from the CPU. This is in contrast to the memory-centric (or CPU-centric) architecture that is centered around a general-purpose processor that controls the media streams in a computer using a memory-addressing.

Memory-centric

Modern high-performance network protocols require that all network access is handled by 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). The communication costs can be broken up in per packet and per-byte costs. The per-packet cost can be optimised, and for large packets, this overhead is amortised over a lot of data. However, the cost of per-byte operations such as data copying and checksumming is not reduced by increasing the packet size. Let us first take a look at the typical processing path that an information bit incurs in a multimedia networked computer. Typically, the information bits are generated at a module via a sensor such as a camera. Processing like coding and compression may be done by a codec at this stage. The control flow (and possibly also the information bits) passes through middleware / operating system layers. The bits will then be processed by an application for the transmission over the network. The bits are then sent by the application, again via middleware / operating system, to a network protocol stack which is composed of transport, network, link and medium access (MAC), and physical layer protocols. Typical functions in the protocol stack include routing, congestion control, error control, resource reservation, scheduling, etc. The bits are eventually sent over the network to other nodes where they traverse a reverse path. Notice how, instead of arithmetic functions like additions and multiplications, the primary importance in the system is processing for the protocols.

To address this performance problem, several *user-level communication architectures* have been developed that remove the operating system from the critical communication path [10] and to minimise the number of times the data is actually touched by the host CPU on its path through the system. The ideal scenario is a single-copy architecture in which the data is copied exactly once. Measurements of a single-copy protocol stack at Carnegie Mellon University show that for large reads and writes the single-copy path through the stack is five to seven times more efficient than the traditional implementation for large writes [67].

There are several ways to build flexible, high performance communication protocols. Advanced protocol design techniques include *application-level framing*, in which the protocol buffering is fully integrated with application-specific processing, and *integrated-layer processing*, in which many protocol layers are collapsed into highly efficient, monolithic code paths. Integrating buffer management between the application and network interface is important in eliminating data copies and reducing allocations and de-allocations. This integration, however, gives rise to additional complexity due to virtual addressing mechanisms and protection [22] and may need a substantial amount of memory [67].

The *traditional architecture* of a mobile, shown in Figure 2, is centered around a general-purpose processor with local memory and a bus that connects peripherals to the CPU. The long arrow in the figure indicates the essential data stream through the system when data arrives from the network, is transferred through the receive buffers on the network interface, copied to the 'main' memory, and then processed by the application. After the data is processed by the application, the data will traverse via 'main' memory, over the bus, to the output device (i.e. in the figure the display module). In general additional bus transfers between CPU and memory are introduced while traversing several protocol layers (e.g. for data conversion of the packets like Ethernet to IP, and subsequently IP to TCP). A large fraction of system time and power budget is thus devoted to bus transactions.

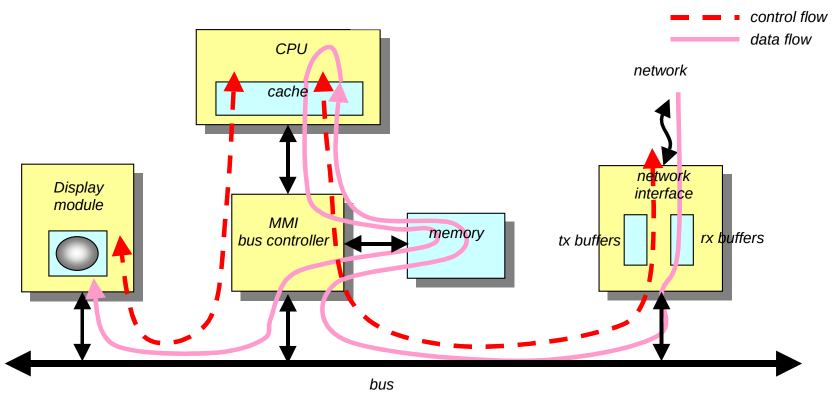

Figure 2: Data flow through a traditional architecture.

As can be seen in the figure, the CPU is required not only to handle the control path, but also to transfer the data between the devices. A better approach is to transfer vast amounts of data through the system by using Direct Memory Access (DMA) functionality of the interfaces and modules. DMA is used for the data transfer between the main memory and the buffers on the network interface. Figure 3 shows the separate data and control flows of such an optimised architecture. The CPU is now only required to perform the control flow between the devices, e.g. – like in our previous example – to the network interface and to the display module. Although this already reduces the demands laid on the processor drastically, the processor still needs to be active during the data transaction to perform the control flow.

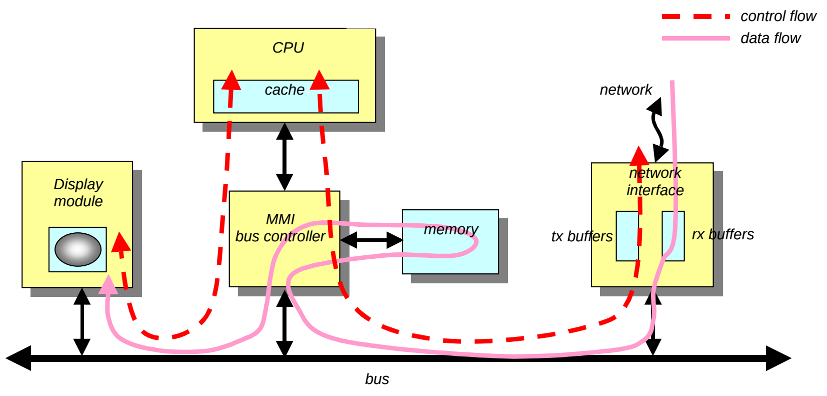

Figure 3: Separated data and control flow traditional architecture.

DMA can introduce some drawbacks. Checksumming needs to be done while copying the data, i.e. checksumming needs to be done in hardware. On workstations, the use of DMA to transfer data between network buffers and main memory is made more complicated by the presence of a cache and virtual memory. DMA can create inconsistencies between the cache and main memory. Hosts can avoid this problem by flushing the data to memory before transferring it using DMA on transmit and invalidating the data in the cache before DMAing on receive. User pages have to be wired in memory to insure that they are not paged out while the DMA is in progress. Because of these extra overheads, it might sometimes be more efficient to use the CPU to copy packets between user and system space. Furthermore, it can be desirable to alter the data stream online, such as decompressing an MPEG audio or video stream. In such cases the CPU generally needs to perform the conversion.

In some special cases it might be possible to forward the data directly from source to destination. This can for example be applied to display graphics data on the screen if the display memory is accessible by the source. In this case the data only needs to travel once across the interconnect between source and sink.

Note that even in this optimised case, there are still one or two data copies required over the shared bus. Busses are significant sources of power dissipation due to high switching activities and large capacitance. Modern systems are typically characterised by wide and high-speed busses, which means that the capacitance and frequency factor of the power dissipation dominates.

The conventional memory-centred shared-bus architecture requires frequent traversal of multimedia streams over the highly capacitive central bus and through the layers of the operating system software for the simplest operations such as multiplexing/demultiplexing and interstream synchronisation. Indeed, measurements with a prototype wireless multimedia terminal at the University of California at Los Angeles (UCLA) with an embedded PC-based architecture show that large amounts of time and power go into memory and I/O transactions across the shared bus [41].

The same trend can be observed in microprocessor designs. *Computer* produced a special issue on “Billion-transistor architectures” that discussed problems and trends that will affect future processor designs, and several proposed microprocessor architectures and implementations [12]. 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 help mitigate the high latency and low bandwidth of external memory. In other words, the conventional vision of future computers spends most of the billion-transistor budget on redundant, local copies of data normally found elsewhere in the system [37].

Current systems based on a shared bus architecture are able to deliver the required performance for various multimedia applications not only by using the rapid advance in technology, but also by careful design and use of the interface modules. The process to achieve this requires a huge amount of effort of both the hardware designer of the I/O interfaces and the system designer. The hardware designer has to very carefully wire the devices to fit the needs of the application, tailor the circuits so that the wires are as short as possible and all the signals get from their originating point to the right place at exactly the right time. Then, the software designer must carefully determine in detail what the devices are capable of before designing a multimedia system [11]. There are many subtle device issues that can influence the overall I/O performance of a system. When, after a lot of fine-tuning, finally the system is running satisfactory, performance problems can arise easily when the (hardware or software) configuration of the system is (slightly) changed, the operating system is updated, or the user is using a new application. The reason for these problems are often caused by the interconnect and the interconnection protocols. Since a shared bus cannot give QoS guarantees, a single device or application can reduce the throughput that is available for all devices.

Connection-centric

By designing an architecture that moves processing power closer to the data stream, it is possible to bypass the CPU altogether. This approach is especially well suited for continuous media data (e.g. audio, video, etc.), where the processing is actually of a very specialised nature (e.g. signal processing, compression, encryption, etc.) and needs to be carried out in real-time. The CPU is thus moved out of the data flow datapath, although it still participates in the control flow. The role of the CPU is reduced to a controller that initialises the system and handles complex protocol processing that are most easily implemented in software on a general-purpose processor.

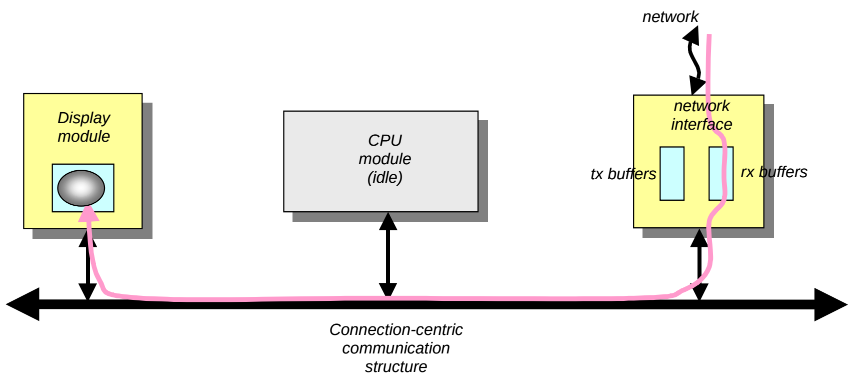

Figure 4: Data flow in a connection-centric architecture.

In contrast to memory-centric systems, a connection-centric system is decomposed out of application-specific coprocessors that communicate using connections. The operating system plays a crucial role in this architecture, as it is responsible to set-up the connections between the modules. The CPU and the operating system do not participate in the control flow during a transaction. The interconnection structure is not based on a bus that uses addresses, but is based on a connection-oriented communication structure (such as the *Octopus* switching fabric as depicted in Figure 1). In such a system the data traffic is reduced, mainly 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 display module without going through the main processor. The display module possibly converts the data stream and forwards it to its screen memory. The result is that instead of the eight data transactions (of which two over a large bus) that were needed in the traditional architecture, in the connection-centric architecture only two local transactions are required (network interface to switch, switch to display module). The CPU module is mainly used to initiate the connections, and can be idling during the transaction.

In a connection-centric architecture, each connection can be associated with a certain QoS using a connection identifier. This identifier provides the mechanism to support lightweight protocols that provide data-specific transport services that are associated with a certain QoS. The careful design and fine-tuning process that is needed by system designers for the memory centric architecture, is not needed if QoS can be guaranteed throughout the whole system.

3.3.4 Application domain specific modules

Figure 1 gives a schematic overview of the *Mobile Digital Companion* architecture. In it, we distinguish a switch surrounded by several modules that are each optimised for a certain application domain. Moreover, these modules are reconfigurable, so that they are able to be adapted when this is required.

In general there is always a module with a general-purpose main processor that performs control-type operations. Other modules can be devices like display controllers, network interfaces and stable storage. The architecture comprises many devices normally found in multimedia workstations, but since our target is a portable computer, these devices generally do not have the performance and size of their workstation counterparts. Our ultimate target is to have a *system-on-a-chi*p, where all the functionality of a system is integrated on a single chip.

Note that our devices are not merely dedicated I/O devices in the traditional sense. We prefer to call the devices *modules*, or *I/O subsystems*, to emphasise the fact that they provide more functionality than a simple device. The modules differentiate to I/O devices in multiple ways. First, each module is an autonomous sub-system that can operate without intervention from the main CPU. Second, it has a control processor that performs diverse operations, including connection management and energy management. Finally, most modules are able to adapt their behaviour autonomously to the ‘wishes’ of the client or application, and try to operate in the most efficient way.

Advantages – It is often more sensible to implement device (or media) specific adaptation layers within the module, rather than requiring the network interface or the main CPU to implement a plethora of different adaptation layers [7]. The main reasons for this are:

- *Efficient processing* – The modules are capable of efficiently performing device or application specific tasks. It can for example decompress a video stream, just before it is displayed on the screen (this is a typical example of the locality of reference principle). Dedicated modules can be optimised to execute specific tasks efficiently, with minimal energy overhead. Instead of executing all computations in a general-purpose datapath, as is commonly done in conventional programmable architectures, the energy- and computation-intensive tasks are executed on optimised modules. For example, even when the application-specific coprocessor consumes more power than the processor, it may accomplish the same task in far less time, resulting in net energy savings. The processor can, for example, be offloaded with tasks like JPEG and MP3 decoding, encryption, and some network protocol handling. A system designer can apply an application-specific coprocessor (e.g. custom hardware) for those tasks the module can handle efficiently, and use the processor for those portions of the algorithm for which the hardware is not well suited (e.g. initialisation).

- *Eliminate useless data copies* – When the data flows directly between the modules that need to process them, unnecessary data copies can be eliminated. Eliminating unnecessary data copies can reduce the traffic on the bus. For example, in a system where a stream of video data is to be displayed on a screen, the data can be copied directly from the network into the screen memory, without going through the main processor.
- *Relieve the general-purpose CPU* – In a connection-centric system data can flow between modules without any involvement of the main CPU and without using any processor cycles. The main CPU is also relieved of having to service interrupts and to perform context switches every time new data arrives, or needs to be sent from a local device. Instead of having one central system that needs to control and process fine-grained operations, a distributed control system is less complex. It is easier to provide real-time support for devices if a dedicated processor controls them.
- *Easy adaptations* – The modules can easily adapt their behaviour. If a module adapts it behaviour, it is able to react on changes in the environment, either imposed by the user (when it starts a new or different application) or by resource changes (for example when the network module notices a change in the wireless channel conditions).
- *Adequate energy management* – Each module contains specific knowledge about the usage patterns and the specific requirements for a device. Therefore, each module has its own responsibility and has some autonomy in deciding how to manage its state of operation to minimise its energy consumption without compromising its quality of service. This is in contrast to current systems in which the main CPU can control the power-state of the connected devices. We have given the modules their own responsibility in deciding how to manage their resources. The individual modules are controlled by a global energy policy that makes high-level decisions on the state of the entire system.
- *Flexible and adaptable* – Because the modules are programmable, they can offer the flexibility to provide support for various standards that a Companion might need to use (e.g. different encoding and encryption schemes), and the adaptability to adapt its mechanisms, algorithms and techniques to the various operating conditions. Of all the programmable modules, the general-purpose processor is merely the most flexible one. The processor will be used for all tasks that the application specific modules are not capable of, or when the implementation would not be efficient. The general-purpose processor will perform all computations that are too complex or would require too much area if they were implemented with the hardware modules. The general-purpose processor can thus also be seen as an application domain

specific module: its domain covers all areas that are not covered by the other modules.

Of course there are also some disadvantages. Most of the trade-offs involved have already been discussed in Section 3.2.6. The most apparent disadvantage seems to be that having application domain specific modules requires more hardware. Instead of processing all tasks on one general-purpose processor, these tasks are distributed over several modules. However, it is expected that the advance in technology give enough possibilities to take advantage of the increased effective chip-area and provide more functionality while keeping the energy consumption low.

Examples of modules

The modules can be very diverse and have different characteristics and requirements. The modules must be capable of handling connections with other modules. They typically contain some intelligence for connection setup and energy management. Typical examples of modules are:

- *CPU-module* – This is the module that can perform general-purpose applications, and provides a broad range of services. One important task of this module is that is responsible for connection management. If a connection has to be established that requires some quality of service guarantees, then the CPU module negotiates with the modules that take part in the connection. The CPU-module is the central place where all QoS related connections are managed.
- *Network module* – This module provides the access to and from the external (wireless) network. In our research we have developed an energy efficient network module that can handle multimedia traffic. This module is a major topic of Chapter 5. The module is able to route traffic according to the VCI of the ATM cells directly to the destination module. The base station plays a crucial role, since it handles most of the network protocol stack layers in the communication over the wired network in lieu of the mobile. In this way the wireless channel peculiarities are decoupled from the network protocol layers, and can provide a more efficient communication [6]. The base station can also act as a proxy server that adjusts the data to the format the mobile can use in an efficient way. For example, a video stream from the fixed network is converted in the base station to a format that the display can easily interpret [74]. The communication between the network module and the base station uses an energy efficient MAC protocol (E^2MaC) that is able to provide QoS guarantees over the wireless channel [31]. The architecture of the network module uses a dynamic error control adapted to the QoS and traffic type of a connection, and has dedicated connection queues and flow control for each connection.

  Note that while we propose to eliminate the dependency on software-based protocol stacks from the mobile, there is no reason to dogmatically preclude the involvement of the general-purpose processor. For example, for research purposes, it is desirable to have the ability to develop and test new algorithms and protocols. In a sense the stack remains a software stack: the devices are programmable. Traditional disadvantages of hardware stacks (inflexibility, cost in concrete) do not hold here.

- *Display module* – On a portable computer the display will generally be small and have a low resolution. The best approach is to adapt the data that is transmitted to the display module such that it can easily interpret the data and display it directly on the screen. If the display module contains an decompression engine, then energy and bandwidth can be saved because no uncompressed data would have to traverse the interconnect, and possibly also traverse the wireless network.
- *Reconfigurable computing module* – This module is basically a device that is capable to handle a wide variety of services. The module contains reconfigurable logic that can be (re)configured dynamically to the requirements. A typical example where such a module can provide a large flexibility is when it is used as an encryption/decryption engine. Data destined or coming from the fixed network could be made to ‘pass through’ the encryption device on the way to the destination module.

The architecture is modular and can be extended with modules that have a different functionality.

3.3.5 The interconnection network

The interconnection network is a key component for providing flexibility in reconfigurable systems [76]. All modules in the system communicate over a reconfigurable communication network that is organised as a switch. Conceptually, the architecture is analogous to a self-routing packet switch. The exact implementation of the interconnect is not a vital issue in the architecture of the *Mobile Digital Companion*. Just as rings, crossbars and busses have all been used in ATM switches [7], so they may be used in the Companion (although they differ in their complexity and energy consumption). It is the connection-oriented approach using fixed sized cells and the asynchronous multiplexing that are key factors. As in switching networks, the use of a multi-path topology will enable parallel data flows between different pairs of modules and thus will increase the performance.

The switch interconnects the modules and provides a reliable path for communication between modules. Addressing is based on connections rather than memory addresses. This not only eliminates the need to transfer a large number of address bits per access, it also gives the system the possibility to control the QoS of a task down to the communication infrastructure. This is an important requirement since in a QoS architecture all system components, hardware as well as software, have to be covered end-to-end along the way from the source to the destination.

QoS is a general theme in our research. It is used as a means to deal with the dynamic behaviour of the communication channels, and to be able to provide the various streams in a multimedia mobile computer a satisfactory quality at the lowest cost. The whole system is based on connections between modules. Each connection is associated with a certain QoS. Applications must indicate the QoS is expects from the system, and its ability and willingness to change these QoS requirements. All modules communicate using these connections only. The network module uses the same mechanism to communicate over the wireless channel with a base station that is connected to a wired environment. If the wired environment also provides mechanisms that are able to deal with QoS (like an ATM network), then we are in principal able to establish an *end-to-end QoS*. In this way we are able to establish a connection from applications on the wired network, through the wireless network, right to the destination where the data will be processed. In the path from source to destination we apply the QoS demands to the various resources involved including the error control over the wireless channel, the communication link layer, and the medium access protocol. The goal is to satisfy the QoS requirements of the connections at the lowest energy consumption. The consequences for the operating system are sketched in Section 3.3.8.

In our infrastructure all connections are identified with a connection identifier which is used to identify the type of data, and the module destination address. This identifier provides the mechanism to support lightweight protocols that provide data-specific transport services that are associated with a certain QoS.

An architecture in which a generalised packet switched interconnect is used to connect processors, memories, and devices has widely come to be known as a ‘desk-area network’ (DAN). Leslie, McAuley and Tennenhouse first introduced the concept of DAN [40]. We have adopted this concept, and will show that such an architecture is also suitable for low-power portable computers. Our architecture has therefore some similarities to for example the *Desk Area Network* from Cambridge [32], *VuNet* from MIT [34] and the *APIC* architecture from the University of Washington [18]. However, their main motivation was performance and interoperability between (ATM [54]) networks and devices. Our main motivation is reducing energy consumption and not only performance.

Ultimately, the architecture should be implemented in just a single chip. Therefore, we would not call the architecture a *Desk*-area network, but merely a *Chip*-area network. In the *Rattlesnake* ATM switch we already showed that it is quite feasible to build a cost effective ATM switching system that meets multimedia requirements [27][61]. Some of the ideas that were introduced in the architecture of the Rattlesnake switch have been used in the design of the *Octopus* switch that is subject of Chapter 4.

Every device is equipped with an interface to the ATM interconnect. This attachment is no more complicated than the equivalent bus interface, and even simpler than complicated high performance busses (Chapter 4). The header of a standard ATM cell is subdivided into a virtual path and a virtual channel. Within the *Mobile Digital Companion*, this subdivision is not significant. If we would generalise our architecture so that it becomes a full-fledged ATM network that can interface with different ATM devices, then we arrive at a more futuristic DAN-based architecture. While such a system would likely deliver a higher performance than the architecture of the *Mobile Digital Companion*, it does not come without some drawbacks. Among these is the high energy consumption that would needed to implement the full-blown ATM network stack. Furthermore, the costs would be higher because important considerations when designing a network architecture are scalability and tolerance to malfunctioning links and nodes. In a chip-area network the trust boundaries and the operating conditions are much different from those of the desk-area network and local or wide-area networks. Larger networks need protection from hostile or faulty clients and a great amount of processing power must be put into devices to manage control and security functions. An interconnect architecture designed for interconnecting the components of a mobile system has to fulfil less stringent requirements. In our architecture, the end nodes are dedicated to one work environment, they can be controlled more easily, they can be trusted to be fair and do not exceed their resource allocations.

3.3.6 Energy analysis

In this section we will evaluate the energy impact of a connection-centric architecture based on a switch, compared to a memory-centric architecture based on a shared bus. Figure 5 shows the two architectures. In this analysis we will only deal with the energy effects of communication. Note that the energy consumption required for communication is only a small fraction of the total energy consumption of a system using a typical multimedia application (i.e. 1/10th, see Chapter 6[2]).

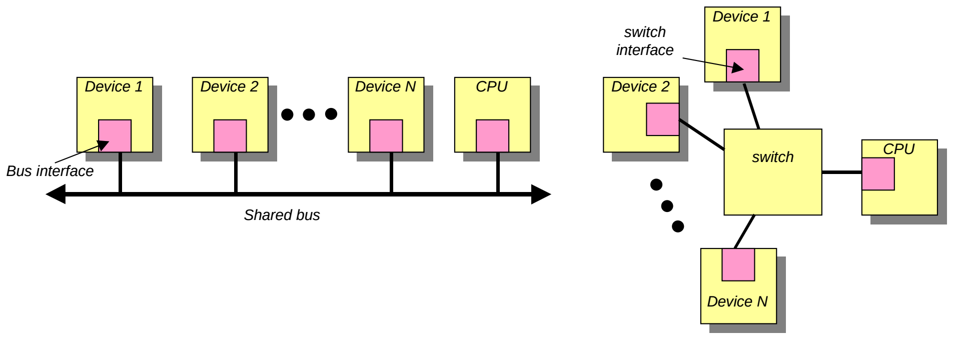

Figure 5: Shared bus architecture and switched architecture.

From Chapter 2 we know that a first order approximation of the dynamic energy consumption of CMOS circuitry is given by the formula:

$$P_d = C_{eff} V^2 f \qquad (1)$$

where P_d is the energy consumption in Watts, C_{eff} is the effective switching capacitance in Farads, V is the supply voltage in Volts, and f is the frequency of operations in Hertz. C_{eff} combines two factors, the capacitance C being charged/discharged, and the *activity weighting* α, which is the probability that a transition occurs.

$$C_{eff} = \alpha C \qquad (2)$$

Most parameters in these equations are affected by the choice of the architecture. In our analysis we make a few assumptions.

[2] In Chapter 6 we will evaluate the energy consumption of a mobile system for typical applications.

1. The elements we will consider to contribute to the energy consumption in both architectures are the energy consumption of connection interfaces (P_{bi} for the bus interface and P_{si} for the switch interface) and the energy consumption caused by the wiring (P_{bw} for each interface to the bus, and P_{sw} for each interface to the switch). The energy consumption caused by the switching fabric is P_{switch} per device interface.
2. The shared bus architecture is memory centric, which implies that to transfer a packet an address has to be provided. We will assume that the bus interface controller provides a burst mode, such that only one address is required for the whole packet. The switched architecture is connection centric, which implies that a connection identifier has to be provided per packet. In our analysis we will neglect the differences caused by these addressing schemes.
3. We assume that the N devices (modules) have a half-duplex connection. This implies that the aggregate throughput is thus maximal $N/2$ times the throughput of a single connection. For example, in the Octopus architecture there can be four simultaneous connections when the source and destination of these connections are disjoint. However, because connections in a system are not always disjoint, we will assume an average aggregate throughput of $N/4$.
4. We assume in our analysis that the complexity of bus interface logic is equal to that of a switch interface. Note that it is, however, more likely that the bus interface will be more complex. This is because a bus interface needs to be flexible (because it must be capable to handle a wide variety of devices) and have a high performance (because the bus is shared it for instance needs to implement burst data transfer modes to achieve the required data rates). Because a bus interface has to operate at a higher frequency, the energy consumption of a bus interface will be higher. Using Assumption 3 we will assume that the energy consumption of a bus interface is $N/4$ times the energy consumption of a switch interface, thus $P_{bi} = N/4 \, . \, P_{si}$.
5. From Equation (1) we know that there is a linear dependence of capacitance on the energy consumption. The capacitive load C_{out} of a CMOS logic gate G consists mainly of a) gate capacitance C_{fo} of transistors in gates driven by G, b) capacitance C_w of the wires that connect the gates and c) parasitic capacitance C_p of the transistors in gate G [8]. In symbols:

   $$C_{out} = C_{fo} + C_w + C_p \qquad (3)$$

   The fan-out capacitance depends on the number of logic gates driven by G and the dimensions of their transistors. In a bus architecture, this number is equivalent to the number of connected devices (including the CPU) N. The size of the transistors that need to drive a high speed bus is also larger than a transistor that only needs to drive one gate. It is extremely hard to estimate C_w accurately because it depends on the topology and routing of the wires and their size. Coupling between wires is becoming the most important factor for the wiring capacitance. The wiring capacitance dominates C_{out} for busses. The parasitic capacitance C_p is probably the component causing the least concern, as it is relatively small compared to the other two contributions.

Taking these considerations into account, we will assume in our analysis that the capacity per device on the shared bus (C_{bw}) is twice of that capacity of the wires on a switch architecture (C_{sw}), thus $C_{bw} = 2\ C_{sw}$.

6. As described in Section 3.3.3, the shared bus architecture requires at least one, and in most situations two data transfers over the shared bus for a stream between two devices. We will assume that the data transfers over the interconnect are based on Direct Memory Access (DMA) performed by DMA controllers of the devices. The number of DMA copies is D. The complexity of the DMA controllers that are required for the shared bus architecture and for the switched architecture is assumed to be the same.
7. We ignore in this analysis the energy consumption that is due to the CPU controlling the data flow. Note, however, that this can be a significant part of the total energy consumption since in a shared bus architecture where the CPU has to manage the data flow in the machine, the CPU cannot enter a sleep mode for a reasonable time. In the switched architecture, the CPU is out of the data path and can enter sleep mode whenever the connection has been setup. We further ignore the energy consumption caused by the intermediate buffering in the CPU-node.
8. Voltage scaling is an effective way to reduce energy consumption. A switched architecture can operate at a lower voltage than a shared bus architecture because it can operate at a lower speed. In a shared bus architecture all data streams must be performed sequentially using one shared resource (the bus). To achieve the required throughput, the bus has to be fast. The switched architecture allows several parallel data streams. Because of this, the bandwidth of the individual connections in a switched architecture does not have to be as high as the bandwidth in a shared bus architecture. Because of the lower bandwidth required for the connections on a switched architecture, the voltage can be reduced. The delay of a CMOS inverter can be described by the following formula [8]:

$$T_d = \frac{C_{out} V_{dd}}{I} = \frac{C_{out} V_{dd}}{\eta (W/L)(V_{dd} - V_t)^2} \qquad (4)$$

where η is a technology-dependent constant, W and L are respectively the transistor width and length, and V_t is the threshold voltage. Many simplifying assumptions are made in the derivation of Equation (4). Nevertheless, the equation contains the variables on which gate delay actually depends, and the nature of their effect is correctly represented.

We will assume that the lower required bandwidth of the switched architecture (Assumption 3), together with the lower capacity of the wires of the switch, corresponds to a potential reduction in voltage of 50% ($V_{bus} = 2\ V_{switch}$), corresponding to a four times lower energy consumption.

The energy consumption P_{ba} required to transfer a certain amount of data (a packet) over a shared bus architecture with N devices is given by:

$$P_{ba} = N \,.\, D \,.\, (P_{bi} + P_{bw}) \qquad (5)$$

Similarly, the energy consumption P_{sa} required to transfer a packet over a switched architecture (from source device to switch, and from switch to sink device) is given by:

$$P_{sa} = 2 \,.\, (P_{si} + P_{sw} + P_{switch}) \qquad (6)$$

in which P_{switch} is the energy consumption of the switching fabric per interface connection. This leads to the following ratio R between the energy consumption of a shared bus architecture and the energy consumption of a switched architecture.

$$R = \frac{P_{ba}}{P_{sa}} = \frac{N \,.\, D\,(P_{bi} + P_{bw})}{2\,(P_{si} + P_{sw} + P_{switch})} \qquad (7)$$

P_{bw} and P_{sw} depend on the capacity and voltage of the wires. When we incorporate the effects of Assumption 5 ($C_{bw} = 2\ C_{sw}$), and 8 ($V_{bus} = 2\ V_{switch}$), we can deduce that $P_{bw} = 8\ P_{sw}$. Using this and Assumption 3 ($P_{bi} = N/4 \,.\, P_{si}$) we can rewrite Equation (7) as:

$$R = \frac{P_{ba}}{P_{sa}} = \frac{N \,.\, D\,(N/4 \,.\, P_{si} + 8\,P_{sw})}{2\,(P_{si} + P_{sw} + P_{switch})} \qquad (8)$$

To give a feeling of the effect of how this equation works out in a real system, we will use the values obtained by our testbed implementation of a switched architecture (the *Octopus* switch that is described in Chapter 4), and compare this with a shared-bus based system with an equivalent performance.

In the testbed of the Octopus architecture we are using a small low-power microcontroller that operates as the interface controller. The power consumption of this microcontroller is about 26 mW (at 3.3V, 33 MHz)[3]. So a reasonable value of P_{si} is thus 26 mW. The energy consumption P_{switch} of the switching fabric is about 15 mW per interface connection of our testbed implementation[4].

To calculate P_{sw} we assume an 8 bits wide interface operating at 3.3 V with a capacitance of 5 pF per wire. When we assume the operating frequency to be 33 MHz and a toggling probability of 0.25 at each clock cycle, then the power dissipation P_{sw} will be: (8 x 5 pF) x (0.25 x 33 MHz) x $3.3^2\ V^2 \approx$ 4 mW.

When we have the number of devices N=8, and the number of DMA transfers on a shared bus D=2, then the ratio R = 1344/90 ≈ 15. We can conclude that, even while the

[3] A dedicated controller would consume less energy. Note that a PCI bus controller has a much higher power consumption, e.g. the PCI9060 PCI bus master from PLX technology requires 680 mW to operate [54].

[4] The testbed was designed to be flexible. It is energy efficient, and is not low power (see Chapter 4). A dedicated implementation would consume much less energy.

assumptions we have made are very conservative and that we have used the power consumption of a *testbed* switched architecture, the switched architecture is much more energy efficient than the shared bus architecture.

In the previous discussion we did not consider dynamic power control, but assumed a connection between two devices with a continuous data stream. In situations where this is no activity on the interconnect, a well designed switched architecture is capable of operating at a low-power sleep mode, whereas in the shared bus architecture the bus interface has to be active all the time.

3.3.7 Timing control

The *Mobile Digital Companion* has a connection-centric approach for several reasons like performance, QoS provisions, energy efficiency, and complexity. There are yet other reasons to choose for a connection-centric architecture. These reasons all stem from the timing-control mechanism in the architecture.

Basic timing control structures

The choice and design of a proper timing control structure for a system is a vital and yet a very practical issue. The synchronous timing scheme is often the first choice in the system design because of the low hardware complexity and logic design simplicity. In a *synchronous system* clock signals serve two purposes: as a sequence reference, and as a time reference. As a sequence reference, a clock transition defines the instance at which the system may change state (so that random state changes and interference can be eliminated). As a time reference, the interval between clock level transitions defines a time region during which data can either move between successive processing stages or are processed in stages isolated from others. A clock signal can thus be viewed as a guard that controls when and what is to be done or not to be done. To ensure the correct system operations, a clock distribution scheme must be used to generate logically equivalent clock signals across an entire system. However, clock skews in the system are unavoidable and caused by many random factors such as various signal propagation delays on wires and in logic, capacitive loading variations at different points, and variations in device and process parameters. The clock signal typically drives a large load because it has to reach many sequential elements distributed throughout the chip. Therefore, clock signals have been a notorious source of energy dissipation because of high frequency and load. It has been observed that clock distribution can take up to 45% of the total energy dissipation of a high performance microprocessor [75].

In an adaptive and (re)configurable system, the synchronous timing method suffers from even more problems. In such a system the delay characteristics (in both communication delays and computation delays) may be very different with different configurations, and cannot, or very limited, be estimated in advance. Therefore, it will be very difficult to determine an appropriate clock speed for the system.

Because it is becoming more and more difficult to distribute a proper global clock network over a large area of silicon, and it is increasingly expensive to design an efficient schedule for a synchronous system with millions of transistors, *asynchronous* *design methods* might give a solution. Such a self-timed system is built by decomposing the system into a set of combinatorial logic blocks and inserting an asynchronous hand-shaking control between each pair of connected blocks. Because the complexity of the hand-shaking circuits increases drastically with the number of inputs and outputs, the building blocks perform relatively simple functions. Because there is no global clock in such a system, the system performance and energy consumption is data dependent at run-time. Energy is dissipated only when the circuit is active. As a consequence, asynchronous circuits can have remarkable energy performance [9][46]. However, the circuit complexity to implement the handshaking control logic and the required area to implement such a system are relatively high if the size of the associated logic is small. Also, there are extra delays caused by the hand-shaking protocol and the logic needed to implement it. Asynchronous logic has failed to gain acceptance on the circuit level, mainly based on area and performance criteria, but also due to the design difficulty.

Timing control in a connection-centric architecture

A system can in general – and in a connection -centric system in particular – be composed into two essential parts: a set of functional modules and a communication network connecting these modules. The most efficient system with the highest performance can be achieved if both the modules and the communication network are running at their highest possible and/or most efficient performance, and these performances are well matched with each other. If all functional modules and the communication network of the system are timed separately, then there is a better chance to achieve this goal. The feasibility of meeting such a requirement depends not only on the timing scheme, but also on the architecture of the system.

Synchronous and asynchronous design approaches represent two extremes, and many variants in between exist. In a connection-centric system an interesting combination is to use clocks local to individual logic modules for synchronous operation in each module, and an asynchronous protocol between functional modules for asynchronous communication in the interconnection network. Recently several studies (e.g. [2][25]) indicate that it would be worthwhile to consider such an approach to eliminate the necessity of distributing a global clock between block of larger granularity. In this way, the interface circuitry would represent a very small overhead component, and the most energy consuming aspects of synchronous circuitry (i.e. global clock distribution) would be avoided. The timing difficulties in a synchronous system are localised to the logic inside a module, and do not affect the correct data transfer. An interface between the modules and the inter-communication network synchronises the events in a hand-shaking protocol at the input of a synchronous module with a local clock in the module.

3.3.8 Quality of Service framework

Applications must adapt to ever changing environments and they need the help of the operating system to provide the information for it. Traditional operating systems do not tell applications when the network is down, how much communication costs (in terms of cost per bit or energy consumption), or how much CPU resources are available.

Adaptation to available network bandwidth already exists in the context of multimedia communication. It can be very useful for mobile-computing applications as well to be aware of network outages and network communications cost. A *Mobile Digital Companion* may be in a location where communication over the network is expensive and of low bandwidth. When this is the case, a file system (to mention just one example) may do well to adapt its behaviour and stop prefetching to increase performance.

If one investigates by what methods applications can adapt their Quality of Service (QoS), one notices that, in order to bridge substantial changes in resource allocation (CPU, energy consumption and network bandwidth are the resources most affected), merely changing parameters is not sufficient. In a dynamic mobile environment more drastic changes are required, e.g. by changing algorithms. In the MOBY DICK architecture, Quality of Service is a framework to model integration and integrated management of all the system services and applications in the *Mobile Digital Companion*. The consumption of resources by one application might affect other applications, and as resources run out, all applications are affected. If the availability of a resource changes, whether it is a file, CPU cycles, or energy consumption, applications that use them are notified, and they can adapt their behaviour. For example, an application that maintains a distributed diary would request, for its highest QoS, to make use of a consistent view of its files, but, if this cannot be made available due to a network partition, it would accept an inconsistent version as the next best thing. Since communication bandwidth, energy consumption and application behaviour are closely linked, we believe that a QoS framework is a sound basis for integrated management of the resources of the *Mobile Digital Companion*.

The QoS framework influences a large number of parameters of various components in the system. Most of these parameters have also a significant impact on the energy consumption, in general a higher quality requires also more energy. Energy consumption is thus an important parameter in the QoS framework. In order to integrate power awareness in the QoS framework, changes must be made to hardware, drivers, firmware, operating system, and applications. The system needs to be flexible, and have several implementations of a function of which one can be chosen depending on the QoS and available resources. The operating system will control the power states of devices in the system and share this information with applications and users.

One of the key aspects of our QoS approach is to move power management policy decisions to the user 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, however, does not imply that modules have no responsibility. Each module has its own – dedicated – local power management. Only the module is able to, and has the knowledge to implement the necessary power management fine-tuning of the internal functions. However, the overall power management control of the modules is done by the operating system and the user. To take advantage of low-power modes of the system's modules, the operating system needs to direct these modules to change its power mode when it is predicted that the net savings in energy will be worth the time and overhead of switching over and restarting.

3.4 Related work

In this section we will provide an overview of related work in the various topics that are covered by the architecture of the Mobile Digital Companion: i.e. multimedia architectures, heterogeneous architectures, network attached devices, and energy management.

3.4.1 Multimedia architectures

The problem of hardware architecture design for high-performance processors is a topic that is covered widely in the literature. Various architectures have been proposed to address the problems involved with multimedia computing. These approaches are based on high-performance technology and are mostly simple extensions to current architectures. These systems fail to exploit the opportunities for energy reduction offered by multimedia.

Systems like the *InfoPad* [60][70] and *ParcTab* [36] are designed to take advantage of high-speed wireless networking to reduce the amount of computation required on the portable. These systems are a kind of portable terminal and take advantage of the processing power of remote compute servers. This approach simplifies the design and reduces power consumption for the processing components, but significantly increases the network usage and thus potentially increases energy consumption because the network interface is energy expensive. These systems also rely on the availability of a high bandwidth network connectivity and cannot be used when not connected.

UCLA has constructed a network testbed [43] that uses a hardware architecture to localise data for both communication and video. In this way the data streams are reduced and efficiently transferred directly to their destination. The granularity of this system is much larger than the previous systems. Performance evaluation using the testbed has revealed the relative importance of the overhead incurred by the application and network protocols as well as the signal processing in the video and radio hardware [15]. For a high performance node, the overheads due to bus transfers, memory copies, and network processing are high. Bus transfer is the main source of limitations to system throughput for applications requiring movement of large blocks of data across the system bus.

Recent years show an increase in the use of application specific architectures in the general-purpose world. In this approach frequently used operations that are expensive in computation time are implemented in dedicated hardware inside the microprocessor. The hardware units are often called *hardware accelerators*. A typical example can be found in Intel's *MMX* ™ architecture [39]. To further increase performance several instructions can be performed in parallel, an example can be found in VLIW (Very Large Instruction Word) architectures. The term media processor is often used for a class of multimedia processors, predominantly aimed at the multi-media-PC market. For example, the *TriMedia* processor uses a VLIW architecture with hardware accelerators and a data highway to be able to handle applications such as decompression of real-time audio and video [57]. Although hardware accelerators enable the designer to implement higher-level operations, this level is still limited by the requirement of generally applicable instructions to support a high degree of programmability. The amount of parallelism that can be obtained at such a level is rather limited, typically a factor of 3 to 5 [38]. The amount energy consumption required is generally no concern for the designers, and is high.

3.4.2 Heterogeneous parallel architectures

By adding special coprocessors next to the general-purpose processor, the grain of operations is increased to the level of complete functions that are executed on dedicated hardware. However, the coprocessors cannot operate independently from the general-purpose processor that performs the synchronisation of tasks. This leads to a significant overhead in execution time and limits the increase in concurrency that can be obtained. Furthermore, a communication bottleneck can easily occur because in multimedia applications that require a large amount of data, the bandwidth that is offered is highly insufficient because all processors must communicate over the same bus. Making use of function-level parallelism can increase the processing performance and efficiency.

Abnous and Rabaey propose an architecture for signal processing applications that is flexible and uses low power [1]. The architecture consists of a control processor surrounded by a heterogeneous array of autonomous, special-purpose satellite processors. The computational demand on the control processor is minimal, its main task is to configure the system and manage the overall control flow of a given signal-processing algorithm. The satellite processors perform the dominant, energy-intensive computational tasks of algorithms. The granularity of these tasks is relatively small. Some examples include address generators, multiply-accumulate processors for computing vector dot products, etc. The architecture does not allow multiplexing of different tasks on the same processor. This restricts the degree of efficiency, since for every task contained in an application a separate processor is required.

Nieuwland and Lippens [52] propose a heterogeneous multiprocessor architecture that supports a global memory model. Such a model allows for easy re-map of current typical programs on heterogeneous processing elements. A bus connects the heterogeneous processing elements. Local memory on the processing elements is positioned within a single global mapping of the application and is accessible by all other processing elements. Due to a well-defined communication interface, allocating tasks to another processing element does not require changes in the remaining application software. Experiments in software show that although the communication protocol runs rather efficient, a significant part of the speed up is lost in communication due to the small grain size of communication with the coprocessor task.

Leijten has proposed a heterogeneous multiprocessor template to be able to obtain a processing performance [38]. This is obtained by replacing the coprocessors by processors that have their own thread of control, that is, autonomous processors can execute tasks completely independently from the microprocessor. In the resulting multiprocessor solution the general-purpose microprocessor executes low-performance tasks requiring a high degree of programmability, while the other processors execute high-performance tasks requiring only limited programmability. These high-performance processors are *application-domain-specific* (ADS) processors optimised in terms of speed, area and power, and tuned towards a well-defined set of tasks. The granularity of the operations of processors is relatively small, the main target of the system is to implement a multimedia processor.

The University of Twente has developed an architecture that is suitable for reconfigurable low-power DSP-like algorithms. *Field-Programmable **Function** Array* (FP*F*A) devices are reminiscent of FPGAs, but have a matrix of ALUs and lookup tables instead of CLBs (Configurable Logic Blocks). The construction of an ALU from multiple 1-bit-wide lookup tables is energy inefficient [64]. For a wide range of multimedia functions that use digital filtering algorithms on parallel data: video (de)compression, data encryption and digital signatures these devices do not posses the required processing power. For these functions 16/32 bit calculations (multiply, add) are required. Newer architectures are based on 'chunky' function units such as complete ALUs and multipliers. For example, a collection of multipliers might be available along with a crossbar interconnect to efficiently support a wide range of infinite-impulse response (IIR) filters. These architectures present an abstraction that is much higher than logic gates and flip-flops, but highly irregular computations will likely be a poor match.

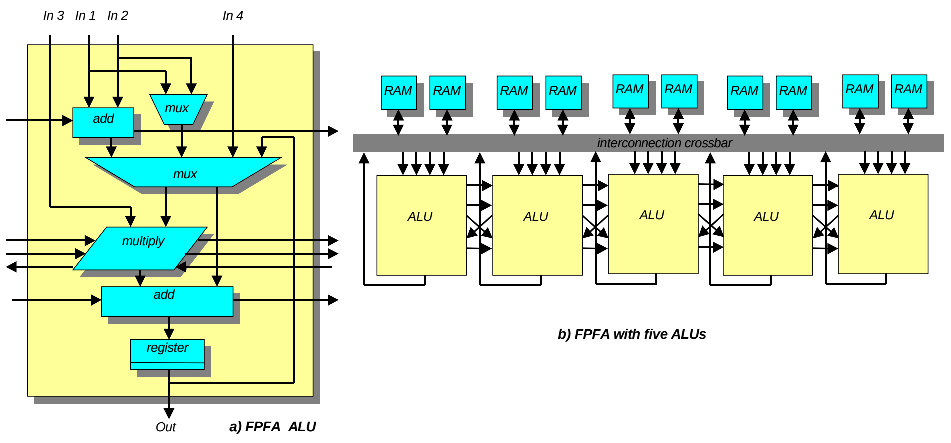

Figure 6: FPFA architecture.

The instruction set of an FPFA-ALU can be thought of as the set of ordinary ALU instructions, with the exception that there are no load and store operations which operate on memories. Instead, they operate on the programmable interconnect; that is, the ALU loads its operands from neighbouring ALU outputs, or from (input) values stored in lookup tables or local registers. The graph-based execution of the FPFA is used to execute the inner loop of an application. The regular, general-purpose structure of the device makes a rapid context switch from one inner loop to another possible, hence on-the-fly reconfiguration. This is how a broad class of compute intensive algorithms can be implemented on an FPFA [63].

At the M.I.T. Laboratory for Computer Science a new architecture is being developed that eliminates the traditional instruction set interface and instead uses a replicated architecture directly to the compiler [73]. This allows the compiler to determine and implement the best resource allocation for each application. They call systems based on that approach *Raw architectures* because they implement only a minimal set of mechanisms in hardware. The architecture is based on a set of interconnected tiles, each of which contains instruction and data memories, an arithmetic logic unit, registers, configurable logic, and a programmable switch that supports both dynamic and compiler-orchestrated static-routing. Raw architectures are best suited for stream-based signal-processing computations.

The TMS320C80 device is a single-chip, parallel processor intended for applications such as real-time audio/video processing, high-end data communications, and image processing [69]. This chip proves the possibility of placing multiple interconnected processors on a single chip. This complex device contains four parallel processing DSPs (PP) with 64-bit instructions and 32-bit fixed-point data; a RISC master processor (MP) with a floating point unit; 50 Kbytes of on-chip RAM; a Video Controller; and a transfer controller that services data requests and cache misses by the MP and PPs. A crossbar switch provides the access of the MP and PPs to on-chip memory.

3.4.3 Network attached devices

A network-attached peripheral (NAP) is a computer peripheral that communicates via a network rather than a traditional I/O bus, such as SCSI [20]. Several research projects using network attached peripherals in multimedia workstations are ongoing in various universities. The canonical example of the uses for NAPs in multimedia is the desire to transmit data directly from a camera to a frame buffer without passing through the system’s backplane, where it unproductively consumes bandwidth. Captures of video to disk and playback from disk are similar. We will now mention only some typical examples.

Desk Area Network – One way to provide real-time guarantees when transferring data inside a workstation is to also use an ATM network switch to interconnect the components of a workstation system. This work has been done at Cambridge [32] and some of this work has now been commercialised by Nemesys. The Desk-Area Network (DAN) carries this idea to the extreme in that the ATM switches are also used to interconnect the memory.

VuNet – The VuNet architecture was designed as part of the ViewStation multimedia project at MIT [4]. The VuNet is a gigabit per second network using ATM, which interconnects general-purpose workstations and multimedia devices. The VuNet is intended to be used as both a desk-area and local-area network. In their approach the multimedia information is channelled all to the workstation processor rather than bypassing it with specialised hardware. They expect that given the current rate of progression of workstation performance, performance levels that allow multimedia tasks to execute in parallel with other tasks, will be reached soon. Due to the software intensive approach to multimedia, the VuNet and custom video hardware were designed to provide efficient support for software driven handling of multimedia streams. The VuNet switch fabric is constructed out of a high performance 4 port crossbar chip and

FIFOs that can buffer 64 cells in the transmit direction and 256 cells on the receive direction.

Switcherland – This scalable communication architecture is based on crossbar switches that provide QoS guarantees for workstation clusters in the form of reserved bandwidth and bounded transmission delays [21]. Similar to ATM technology Switcherland provides QoS guarantees with the help of service classes. Their main target is to provide a high performance and good availability of processors and I/O devices by allowing any arbitrary topology. The switches can be used as an I/O interconnection fabric of a workstation as well as a network interconnection fabric of a workstation cluster.

3.4.4 Energy management

One of the most successful techniques employed by designers of current computers at the system level is dynamic power management [8]. There are, however, few operating systems designed specifically for portable computing equipment. Microsoft's Windows CE [26] is one, USRobotics PalmOS [71] is another. The power management in these systems consists almost exclusively of powering down the CPU and other devices when the system becomes idle and turning off the screen after a few minutes of user inactivity.

Currently several system developers and vendors are pursuing a long-term, wide scope strategy to greatly simplify the task of large and complex power-managed systems. The strategy is based on a standardisation initiative known as *Advanced Configuration and Power Interface* (ACPI). The *OnNow* initiative targets the migration of power management algorithms and policies into the computer's operating system [49]. OnNow and ACPI provide a framework for designers to implement power management strategies. The choice of the policy is left to the designer. OnNow is an initiative proposed by a single software company, and is tightly bound to the abstract model of a personal computer. Although OnNow requires ACPI as the interface between the operating system and the hardware, ACPI is more general in scope and does not depend on any operating system or hardware model. However, both ACPI and OnNow assume a CPU and operating system centric system, where the activities of the system are managed by a single entity. ACPI and OnNow are developed to support the implementation of power managed computer systems, and are too detailed to effectively support design exploration [8].

A modelling approach that is aimed at providing support for system-level architectural exploration of power-managed systems is described in [8]. In their model, a system is defined by a set of components and a communication pattern between components. Communication is modelled by abstract events. The abstraction of the model is much higher than ACPI, and no details are specified about the functional behaviour of a service provider (like disk driver unit or video driver).

3.5 Summary and conclusions

In this chapter we considered the problem of designing an architecture for a handheld mobile multimedia computer. The architecture of the *Mobile Digital Companion* is connection-centric in which the modules communicate using an asynchronous hand-shaking interface. These modules can be combinatorial or controlled by clocks local to each module. The CPU is moved out of the data stream, although it still participates in the control flow. Such a design approach offers a solution in the design of multimedia, low-power wireless terminals. The architecture presents several advantages over the traditional memory-centric models.

Energy management is the general theme in the design of the system architecture since battery life is limited and battery weight is an important factor. We have shown that there is a vital relationship between hardware architecture, operating system architecture and applications architecture, where each benefits from the others. In our architecture we have applied several supplementary energy reduction techniques on all levels of the system. Achieving high energy efficiency requires first of all the elimination of the waste that typically dominates the energy consumption in general-purpose processors. The second main principle used is to have a high locality of reference. The philosophy is that all operations that are required on the data should be done at the place where it the most efficient, thereby also minimising the transport of data through the system.

As the Mobile Digital Companion must remain usable in a wide variety of environments, it must be flexible enough to accommodate a variety of multimedia services and communication capabilities and adapt to various operating conditions in an (energy) efficient way. The approach made to achieve such a system is to use autonomous, adaptable components, interconnected by a switch rather than by a bus, and to offload as much as work as possible from the CPU to programmable modules that is placed in the data streams. Thus, communication between modules is delivered exactly to where it is needed, work is carried out where the data passes through, bypassing the ‘main’ memory, modules are autonomously entering an energy-conservation mode and adapt themselves to the current state of the resources and the requirements of the user. If buffering is required at all, it is placed right on the data path, where it is needed. The application domain specific modules offer enough flexibility to be able to implement a predefined set of (usually) similar applications, while keeping the costs in terms of area and energy consumption to an acceptable low level.

Having an energy-efficient architecture that is capable of handling adaptability and flexibility in a mobile multimedia environment requires more than just a suitable hardware platform. First of all we need to have an operating system architecture that can deal with the hardware platform and the adaptability and flexibility of its devices. Optimisations across diverse layers and functions, not only at the operating systems level, is crucial. Managing and exploiting this diversity is the key system design problem. A model that encompasses different levels of granularity of the system is essential in the design of an energy management system and in assisting the system designer in making the right decisions in the many trade-offs that can be made in the system design. Finally, to fully exploit the possibilities offered by the reconfigurable hardware, we need to have proper operating system support for reconfigurable computing, so that these components can be reprogrammed adequate when the system or the application can benefit from it.

Although our design assumes a low-power, wireless multimedia computer, most of our ideas are applicable (perhaps with some modification) to many other types of computer (sub)systems, including high performance workstations and network interfaces.

References

[1] Abnous A., Seno K., Ichikawa Y., Wan M., Rabaey J.: "Evaluation of a low-power reconfigurable DSP architecture", *proceedings 5th Reconfigurable Architectures workshop (RAW'98)*, March 30, 1998, Orlando, USA. (URL: http://xputers.informatik.uni-kl.de/RAW/RAW98/adv_prg_RAW98.html)

[2] 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.

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

[4] Adam J.F., Houh H.H., Tennenhouse D.L.: "Experience with the VuNet: a network architecture for a distributed multimedia system", *Proceedings of the IEEE 18th Conference on Local Computer Networks*, pp. 70-76, Minneapolis MN, September 1993.

[5] Agarwal A.: "Raw computation", *Scientific American*, pp. 44-47, August 1999.

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

[7] Barham P., Hayter M., McAuley D., Pratt I.: "Devices on the Desk Area Network", March 1994.

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

[9] 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.

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

[11] Bosch P.: "Mixed-media file systems", *Ph.D. Thesis University of Twente*, June 1999.

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

[13] Chaiken D., Hayter M., Kistler J., Redell D.: "The Virtual Book", *SRC Research report 157*, Digital Systems Research Center, November 1998.

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

[15] Chien C., et al.: "An integrated testbed for wireless multimedia computing", *Journal of VLSI Processing Systems* 13, pp. 105-124, 1996.

[16] Dally W.: "Tomorrow's Computing Engines", keynote speech, *Fourth International symposium High-performance Computer Architecture*, Feb. 1998.

[17] Diependorff K., Dubey P.: "How multimedia workloads will change processor design", *Computer*, Sept. 1997, pp.43-45.

[18] Ditta Z.D., Cox R.C., Parulkar G.M.: "Catching up with the networks: host I/O at gigabit rates", *Technical report WUCS-94-11*, Washington University in St. Louis, April 1994.

[19] Dorward S., Pike R., Presotto D., Ritchie D., Trickey H., Winterbottom P.: “Inferno”, *Proceedings COMPCON Spring’97*, 42nd IEEE International Computer Conference, 1997, URL: http://www.lucent.com/inferno.

[20] Doyle van Meter, R.: “A brief survey of current work on network attached peripherals”, *ACM Operating Systems Review*, Jan. 1996.

[21] Eberle H., Oertli E.: “Switcherland: a QoS communication architecture for workstation clusters”, *Proceedings ISCA ’98 – 25th annual Int. Symposium on Computer Architecture*, Barcelona, June 1998.

[22] von Eicken, T., Vogels, W.: “Evolution of the Virtual Interface Architecture”, *Computer*, pp. 61-68, November 1998

[23] Estrin G.: “Organization of Computer Systems: The Fixed-plus Variable Structure Computer”, *Proceedings of the Western Joint Computer Conference*, pp. 33-40, 1960.

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

[25] Gao B., Rees D.R.: “Communicating synchronous logic modules”, *21th Euromicro conference*, September 1995.

[26] O'Hara, R.: “Microsoft Windows CE: History and Design”, *Handheld systems 5.1*, Jan./Feb. 1997, available at http://www.cdpubs.com/Excerpts.html.

[27] Havinga P.J.M., Smit G.J.M.: “Rattlesnake – a single chip high-performance ATM switch”, *proceedings International conference on multimedia networking (MmNet’95*), pp. 208-217, Aizu, Japan, September 26-29, 1996.

[28] 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.

[29] Havinga P.J.M., Smit G.J.M.: “Low power system design techniques for mobile computers”, *CTIT technical report series 97-32*, Enschede, the Netherlands, 1997

[30] Havinga P.J.M., Smit G.J.M.: “The Pocket Companion's architecture”, *Euromicro summer school on mobile computing ’98*, Oulu, pp. 25-34, August 1998

[31] Havinga P.J.M., Smit G.J.M., Bos M.: “Energy efficient wireless ATM design”, *proceedings wmATM’99*, June 1999.

[32] Hayter M.D., McAuley D.R.: “The desk area network”, *ACM Operating systems review*, Vol. 25 No 4, pp. 14-21, October 1991.

[33] Helme A.: “A system for secure user-controlled electronic transactions”, *PhD. thesis University of Twente*, August 1997.

[34] H.H. Houh, Adam J.F., Ismert M., Lindblad C.J., Tennenhouse D.L.: “The VuNet desk area network: architecture, implementation and experience”, *IEEE Journal of Selected Areas in Communications* (JSAC), 13(4):710-121, May 1995 (see also: http://www.tns.lcs.mit.edu/ViewStation/src/html/publications/JSAC95.html)

[35] Hui J.: “Switching and traffic theory for integrated broadband networks”, *Kluwer Academic Press*, 1990.

[36] C. Kantarjiev et al.: “Experiences with X in a wireless environment”, *Mobile and location-independent computing symposium*, Cambridge MA, August 1993.

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

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

[39] Lempel, O., Peleg A., Weiser U.: “Intel’s MMX ™ Technology – a new instruction set extension”, *Proceedings IEEE COMPCON*, pp. 255-259, 1997.

[40] Leslie I., D. McAuley, D. L. Tennenhouse: “ATM Everywhere?”, *IEEE Network*, March 1993.

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

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

[43] 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.

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

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

[46] 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.

[47] Mehra R., Rabaey J.: “Exploiting regularity for low-power design”, *proceedings of the international Conference on computer-aided design*, 1996.

[48] 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.

[49] Microsoft: “OnNow and Power Management”, http://microsoft.com/hwdev/onnow.htm.

[50] Mullender S.J., Corsini P., Hartvigsen G. “Moby Dick – The *Mobile Digital Companion*”, LTR 20422, Annex I – Project Programme, December 1995 (see also http://www.cs.utwente.nl/~havinga/pp.html).

[51] Mullender S.J., Smit G.J.M., Havinga P.J.M., Helme A., Hartvigsen G., Fallmur T., Stabell-kulo T., Bartoli A., Dini G., Rizzo L., Avvenuti M.: “The Moby Dick Architecture”, *CTIT Technical report series*, No. 98-18, Enschede, the Netherlands, 1998.

[52] 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.

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

[54] PLX technology: “PCI9060, PCI Bus master interface chip for adapters and embedded systems”, datasheet, 1995, http://www.plxtech.com/download/9060/datasheets/9060-12.pdf.

[55] Prycker: “Asynchronous Transfer Mode”, 1991.

[56] Rambus Inc.: “Direct Rambus Technology Disclosure, http://www.rambus.com.

[57] Rathnam S., Slavenburg G.: “An architectural overview of the programmable multimedia processor, TM-1”, *Proceedings IEEE COMPCON*, pp. 319-326, 1996.

[58] Reiniger D., Izmailov R., Rajagopalan B., Ott M., Raychaudhuri D.: “Soft QoS control in the WATMnet broadband wireless system”, *IEEE Personal Communications*, pp. 34-43, February 1999.

[59] Rocket eBook, http://www.rocket-ebook.com.

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

[61] Smit G.J.M.: “The design of central switch communication systems for multimedia applications”, *Ph.D. thesis, University of Twente*, 1994.

[62] Smit G.J.M., Havinga P.J.M., et al.: “An overview of the Moby Dick project”, *1st Euromicro summer school on mobile computing*, pp. 159-168, Oulu, August 1998.

[63] 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.

[64] 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.

[65] SoftBook Reader, http://www.softbook.com.

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

[67] Steenkiste P.A. Zill B.D., Kung H.T., Schlick S.J., Hughes J., Kowalski B., Mullaney J.: "A host interface architecture for high speed networks", *Proceedings 4th IFIP conference on high performance networking*, pp. A3-1 A3-16, December 1992.

[68] Steenkiste P.: “Design, implementation and evaluation of a single-copy protocol stack”, *Software – practice and experience*, January 1998.

[69] Texas Instruments, SMJ320C80 Digital Signal Processor, http://www.ti.com/sc/docs/products/sm320C80.html.

[70] 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.

[71] USRobotics PalmOS, URL: http://palmpilot.3com.com.

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

[73] Waingold E, Michael Taylor, Devabhaktuni Srikrishna, Vivek Sarkar, Walter Lee, Victor Lee, Jang Kim, Matthew Frank, Peter Finch, Rajeev Barua, Jonathan Babb, Saman Amarasinghe, and Anant Agarwal: “Baring it all to Software: Raw Machines”, *IEEE Computer*, September 1997, pp. 86-93.

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

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

[76] Zhang H., Wan M., George V., Rabaey J.: “Interconnect architecture exploration for low-energy reconfigurable single-chip DSPs”, *Proceedings of the WVLSI*, Orlando, Fl, April 1999.