Battery-Powered Distributed Systems
Extended Abstract for EW'98

Paul J.M. Havinga, Arne Helme, Sape J. Mullender[1], Gerard J.M. Smit, Jaap Smit
University of Twente, Enschede, Netherlands

1 Introduction

Mobile personal computers will be a vital technology for making electronic information processing available to people on the move. We expect personal mobile computers, *'mobile digital companions'*, to be small enough that they can be carried along all day, versatile enough that they can be used for all kinds of information processing — diary, notebook, pager, telephone, walk man, dictation, e-mail, e-money, keys, ID — and frugal enough that they can be used all day without recharging.

This paper reports ongoing work on Moby Dick, a research project that addresses fundamental issues in the architecture, design and implementation of low-power hand-held computers, with particular emphases on energy conservation and security. The goal is to investigate architectural issues in hardware and software design in concert, so that opportunities in hardware design can be exploited by supportive software.

The approach to energy conservation is to use independently clocked components, interconnected by a switch rather than by a bus and to offload as much work as possible from the CPU to programmable logic that is placed in the data streams. Thus, clocks are turned off for inactive components, communication between components is not broadcast over a bus but delivered exactly where it is needed, and work is carried out where and when data passes through, bypassing the memory and avoiding expensive cache misses. To support this, the operating system must be adapted to become a very small distributed system with cooperating processes occupying programmable components among which the CPU is merely the most flexibly programmable one.

The system is designed as a *real-time* system in order to be able to schedule component clocks. The applications assist the system in its power-saving effort by indicating their service deadlines and required quality of service. Participation of cognitive ergonomists in the project is desirable in order to establish the real-time envelope in which interactive applications can operate, as well as the issues of designing interfaces between people who think of their companion as being *on* continuously and the companion itself which will be *off* for as much of the time as is possible.

The privacy provided by the digital companion must be strong enough that people can securely carry very personal data with them and the integrity strong enough that they can be used for any authentication or payment function. Such security requires a tamper-proof environment for encryption, decryption and certificate validation to take place that includes the human interface.

We have devised the architecture of a tamperproof security system that makes use of programmable hardware in the data path to display and (pen- or keyboard-based) user input device.

The combined presence of expertise in the areas of security and energy management in the project is fortuitous, because they can be realized efficiently in a single architecture which we shall discuss in the remainder of this extended abstract.

2 Where does the energy go?

Lorch [1995] reported that the energy use of a typical laptop computer is dominated by the backlight of the display, the disk and the processor. Stemm *et al.* [1996] concluded that the network interface consumes at least the same amount of energy as the rest of the system (a Newton PDA). If the computer is able to receive messages from the network even when it is 'off', the energy consumption increases dramatically. Ikeda [1994] observed that the contribution of the CPU and memory to power consumption has been on the rise the last few years. Laptops use several techniques to reduce this energy consumption, primarily by turning them off after a period of no use, or by lowering the clock frequency.

There is consensus that disk, display and network interface are the big spenders in terms of energy. It is relatively easy to conceive of mobile companions without disk; after all, DRAM and SRAM are both getting cheaper and more compact. Research effort on compact program representations (e.g., Limbo[1] or Java byte codes [Arnold and Gosling, 1996]) and data representations will remain important.

[1] Contact: Sape Mullender <Sape@Huygens.org>

[1] http://cm.bell-labs.com/inferno/limbo.html

But any reasonable mobile companion must have a display and a (wireless) network interface. Although we recognize the importance of reducing the energy consumed by these, we do not address display and radio technology in our project, but we do address avoidance of communication (see Section 5).

On the hardware side, our interest concerns the energy consumed by data processing, data transport and data storage. Due to the space limitations of this extended abstract, we can only give the most cursory and simplified background information here: Data processing basically consumes energy because (CMOS) gates need energy to transition from low to high and high to low. In the steady state, the energy consumption of a gate is minimal. As a result, SRAM memories need very little energy to retain their information and consume most of their energy during read and write operations[2].

The clock in an electronic circuit forms a source of constant gate activity all over a circuit. Lowering the supply voltage (which implies slowing down the clock), can lead to power savings. Weiser *et al.* [1994] showed how this may be used to reduce the energy consumption in a general-purpose operating system. Research in asynchronous, clockless circuits is another attempt to confine gate activity to where it is productive.

Transporting gate transitions over long distances takes more energy than over short distances. Communication between chips is therefore much more expensive than communication within a chip. A low-power CPU, such as the Strong Arm[3] can only be truly low power as long as it has no cache misses. Addressing memory to resolve a cache miss is expensive because of the distances involved and because of the number of gates activated in finding the location of the data; the bigger the memory the more address bits activate gates. Big memories are thus more expensive than little ones even if the amount of activity is constant.

3 What about security?

The exchange of electronic cash and electronic contracts require encryption as a basic tool for security services. Maintaining the secrecy of keys is essential, as well as guaranteeing that signature keys can never be used without the consent of the owner. An owner will place trust in the mobile companion if it guarantees not to divulge keys, to sign only with explicit permission, and if it can guarantee to verify countersignatures honestly. If owners believe the companion's operating system is truly secure, they can allow it to manage their e-cash.

But this is not enough. In order to trust the signed statement of strangers, one must have assurance that *they* cannot easily lose *their* signature keys. Because, if they can, they stand a good chance of avoiding being held to a contract when they claim they 'must have lost their key' as a result of buying software from the wrong manufacturer. The point we make here is that a secure operating system may be sufficient assurance for its owner (although we also believe such owners may be somewhat irresponsible), but it is not a sufficient assurance for the parties at the far end of a communication link — and will probably never be so for banks. One of us, Helme [1997], made this point with far more arguments than we have space for here.

The consequence is that, for secure electronic transactions, financial or otherwise, a secure processing environment is of the essence. This implies that it may not, in any way, be tampered with by the operating system. The security environment must also include user input and output devices. We propose that the security environment forms an integral part of the screen controller, thus putting both display and touch-screen input in the trusted computing base [Helme, 1997].

4 Micro Distribution

We see an important path to reducing energy consumption by taking processing activities (that cause cache misses, for instance) away from general-purpose processors and realizing them instead on programmable-logic devices. This is not always possible, but often enough and for activities important enough that it can constitute a significant power saving.

Mobile companions, being multimedia systems, will process compressed audio and, if we have our say, video as well; they will also encrypt and decrypt data streams; and they will assist network-protocol processing (e.g, calculate checksums, or error-correcting codes). These are all activities that programmable logic can be good at. Code is downloaded into the chip, the chip is placed in the data path and the calculation takes place with a minimum of fuss. A processor, in contrast, requires the data it processes to be loaded into memory first. This costs energy that the programmable logic does not have to spend. Both program and data are stored in memory and need to be fetched into the CPU before processing takes place. If the processing code fits the cache, then, at best, the CPU uses as much energy as the programmable logic device. If it doesn't, the CPU does much worse.

The traditional interconnect for CPU, memory and devices is the bus. A bus broadcasts address and data. This requires a substantial amount of energy. We designed a *switch* to replace the bus as interconnect [Smit and Havinga, 1998]. The CPU controls the switch by

[2] The USRobotics PalmPilot, for example, can retain a megabyte of information for several months, being powered by just two AAA batteries.

[3] See `http://www.digital.com/semiconductor/-strongarm/strongar.htm`.

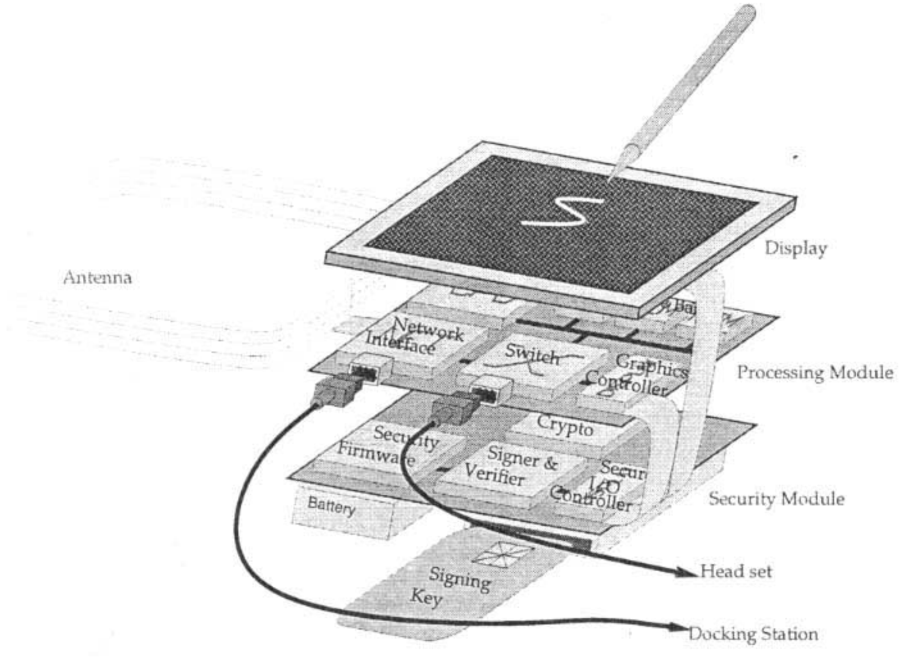

Figure 1: Moby Dick architecture

managing *connections* through it, quite similar in concept to the Desk-Area Network [Hayter and McAuley, 1991]. These can connect CPU to devices, memory to devices, or devices to each other. Instead of doing DMA transfers, devices send and receive data using a message-stream protocol. Connection identifiers can be between ten and sixteen bits in size, whereas memory addresses must be between 24 and 32 bits wide. The switch is clocked by the incoming or outgoing data streams and, therefore, consumes energy only for active connections and on a confined part of the chip.

Putting the system together produces the system illustrated in Figure 1. The figure shows the switch as the central interconnect for CPU, network interface, graphics controller, and audio subsystem.

The security system is shown as a separate layer, for clarity. Note that the security system controls the data path between the programmable components and the human interface. The security system, although it is made up of programmable-logic components similar to those in network interface and graphics controller, is not programmable by any CPU software. Thus, users have the guarantee that their payments and contracts cannot be tampered with by Trojan horses and viruses, that signatures cannot be placed without their consent and that their PINs cannot be intercepted[4]. The security system is personalized with a smart card that contains the user's signature key. The smart card authenticates the security system and will refuse to interact with a security system of dubious reputation. The security protocols have been designed such that physical tampering with the security system jeopardizes the security of the owner, but not that of remote parties in payments and contract exchanges.

Programmable devices will be (combinations of) field-programmable gate arrays (FPGAs), field-programmable *function* arrays (FP*F*As), microcontrollers, and digital signal processors (DSPs). FPGAs and DSPs are familiar components. FP*F*As have been developed under the supervision of J. Smit [Remmelink, 1997]. An FPFA resembles an FPGA: it is a regular grid of identical functional components, that can be connected to buses along the horizontal and vertical grid lines. In contrast to the boolean-logic functions in FPGAs, FPFAs use *arithmetic* functions (ALUs) and 16-bit buses. In addition, there are columns of small memories (64 16-bit words), called *lookup tables* (LUTs) interspersed between the ALUs.

An ALU with an adjacent LUT can be configured to calculate functions (using the LUT as a function-value table and the ALU for interpolation), and a collection of these can be configured to do FFT (useful in the radio interface)[5], data compression, MPEG/JPEG decoding, encryption, etc.

FPFAs do this at a much lower power consumption than either CPUs or DSPs. And, for most processing that involves arithmetic operations, FPFAs are easier to use than FPGAs and consume less energy as well.

5 Macro Distribution

The mobile companion forms a distributed system together with other mobile systems and fixed-base systems. Fixed-base systems of particular interest to mobile-companion users are their home and work computers and server computers for services encountered

[4] Details can be found on line in `http://www.huygens.org/reports/97-01.html`

[5] An FPFA algorithm for a *digital audio broadcast* (DAB) receiver has already been demonstrated.

on the road.

Home and work computers will act as long-term repositories for data replicated in the mobile companion. One can view the companion's data store as a cache for a larger fixed-base (distributed) file system. The connectivity between the companion and the rest of the file system will vary over time. When the companion is in its cradle for recharging, it will also be efficiently connected to the rest of the file system. This the best time for restoring any consistency that might have been lost, but also for preloading information that is likely to be needed on the road.

The issues of maintaining cache consistency and storage reliability through low-bandwidth and high-cost communication infrastructures is still an item of research in our group. We are also investigating application-dependent algorithms for consistency management and cache preloading. Doing this right can be enormously beneficial to perceived performance (data is there when required and does not have to be uploaded first) and radio energy consumption (one of the big spenders).

Another aspect of distribution is interaction between a mobile companion on the move and service offerings encountered. We envision that companions will be used in shops, theatres, railway stations and airports, where they will interact with specific information services. How mobiles find those services is another item of investigation. It has not such a great bearing on energy management, but an enormous one on functionality and security.

6 Conclusions

This extended abstract reports on work that is very much in progress. By seeking solutions in the combined hardware, firmware and software domains, we are convinced that we can solve energy-management problems in portable computers much more fundamentally than existing efforts.

We can experiment with a switch as a replacement for the bus, because we are in the position to write a new operating system to control the switch; we can play with programmable logic, because we can write our own device drivers, and we can experiment with new Quality-of-Service concepts that allow applications to indicate the services they require of the hardware and then build the hardware to make use of these features.

The work is, as we said, in progress. We have presented the furthest-reaching ideas we have for power management, but we have not actually produced any proof that our ideas work. This is something we aim to do in the next four years. We do, however, believe that our ideas are already worth being noted at the SIGOPS workshop, because we think they will engender interesting discussions.

References

Arnold, K. and Gosling, J. [1996], *The JAVA Programming Language*, Addison-Wesley, Reading, MA.

Hayter, M. and McAuley, D. [October 1991], The Desk-Area Network, *ACM Operating System Review* **25**(**4**), 14–21.

Helme, A. [August 1997], A System for Secure User-controlled Electronic Transactions, University of Twente, Ph.D. Dissertation, Enschede, Netherlands.

Ikeda, T. [October 1994], ThinkPad Low-Power Evolution, *Proceedings of the IEEE Symposium on Low Power Electronics*.

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

Remmelink, G. B. [June 1997], Design of an Architecture for a Field-Programmable Function Array, University of Twente, Master's Thesis, Electrical Engineering Report EL-BSC-038N97.

Smit, G. J. M. and Havinga, P. J. M. [1998], A Low-Power Switch. as a new Interconnect for Hand-Held Computers, Huygens Internal Memorandum, Conference publication in preparation.

Stemm, M., Gauthier, P., Harada, D. and Katz, R. H. [September 1996], Reducing power consumption of network interfaces in hand-held devices, *Proceedings of the Mobile Multimedia Computing MoMuc-3*, Princeton.

Weiser, M., Welch, B., Demers, A. and Shenker, S. [November 1994], Scheduling for Reduced CPU Energy, *Proceedings of the Usenix Conference on Operating Systems Design and Implementation*, Monterey, California.