Octopus – an energy-efficient architecture for wireless multimedia systems

Paul J.M. Havinga
University of Twente
department of Computer Science
the Netherlands
+31 (0)53 4894619
havinga@cs.utwente.nl

Gerard J.M. Smit
University of Twente
department of Computer Science
the Netherlands
+31 (0)53 4893734
smit@cs.utwente.nl

***Abstract* – Multimedia computing and mobile computing are two trends that will lead to a new application domain in the near future. However, the technological challenges to establishing this paradigm of computing are non-trivial. Personal mobile computing offers a vision of the future with a much richer and more exciting set of architecture research challenges than extrapolations of the current desktop architectures. In particular, these devices will have limited battery resources, will handle diverse data types, and will operate in environments that are insecure, dynamic and which vary significantly in time and location.**

The approach we made to achieve such a system is to use autonomous, adaptable modules, 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. A reconfigurable internal communication network switch called *Octopus* exploits locality of reference and eliminates wasteful data copies.

***Keywords* – Handheld computers; energy efficiency; switching fabric; multimedia; Quality of Service**

I. INTRODUCTION

Recent advances in wireless networking technology and the exponential development of semiconductor technology have engendered a new paradigm of computing, called *personal mobile computing* or *ubiquitous computing*. Two trends – multimedia applications and mobile computing – will lead to this new application domain and market in the near future. Users carrying portable devices will have access to a shared infrastructure independent of their physical location. The technological challenges to establishing this paradigm of computing are non-trivial, however. Personal mobile computing offers a vision of the future with a much richer and more exciting set of architecture research challenges than extrapolations of the current desktop architectures. In particular, these devices will have limited battery resources, will handle diverse data types, and will operate in environments that are insecure, dynamic and which vary significantly in time and location.

A key challenge of mobile computing is that many attributes of the environment vary dynamically. Mobile devices face many different types of variability in their environment. Therefore, they need to be able to operate in environments that can change drastically in short term as well as long term in available resources and available services. Some short-term variations can be handled by adaptive communication protocols that vary their parameters according to the current condition. Other, more long-term variations generally require a much larger degree of adaptation. Merely algorithmic adaptations are not sufficient, but rather an entirely new set of protocols and/or algorithms may be required. For example, mobile users may encounter a complete different wireless communication infrastructure when walking from their office to the street. They might require another air interface, other network protocols, and so forth. A possible solution is to have a mobile device with a reconfigurable architecture so that it can adapt its operation to the current environment and operating condition.

Outline

In this paper we first motivate why there is a need to revise the system architecture of a portable computer. Then we present our approach in Section III. In Section IV we will give an architectural overview of the *Mobile Digital Companion*, and give a short overview of the state of the art in mobile multimedia computing. We then present in Section V the architecture and design of the interconnection network, the *Octopus* switch. Topics of special interest are the buffer organisation, the scheduling techniques used, and the internal communication protocol. In Section VI the testbed implementation of the *Octopus* switch is described, and performance and energy consumption measurements are presented. Finally, we present the summary and conclusions in Section VII.

II. MOTIVATION

The research community and the industry have expended considerable effort toward mobile computing and the design of portable computers and communication devices. These devices now support a constantly expanding range of functions, and multiple devices are converging into a single unit. 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. 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.

Basically, there are two types of computer devices for use on the road: the palm-top computer and the notebook computer. 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 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 if these behave poor in a wireless environment [2]), 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.

The future: Mobile Digital Companion.

Topic of this paper is the architecture of a future handheld device, called *Mobile Digital Companion* (also referred to as *Companion*), which provides support for handling multimedia applications energy efficiently. A *Mobile Digital Companion* will be a personal machine, and users are likely to become quite dependent on it.

The *Mobile Digital Companion* is envisioned to be 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 [9]. It is resource-poor, i.e. small amount of memory, limited battery life, low processing power, and connected to the environment via a (wireless) network with variable connectivity. An important feature is 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 are characterised by their requirement for transport of multiple synchronised media streams. Some of these streams (typically video streams) have high bandwidth and stringent real-time requirements. Most of the applications we consider require not only a certain Quality of Service for the communication, 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 will be a versatile device that combines multimedia and communication functionality in one single portable device. Nevertheless these functions have to be provided by relatively little hardware and consume little energy because a main requirement for the Companion is small size and weight.

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. 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 [11].

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.

III. APPROACH

In the traditional design of a mobile, a number of problem areas in hardware and software architectures can be identified concerning the energy consumption. The major reason is that current operating system software and networking software emphasises flexibility and performance, and is constructed from components developed by independent groups. A good working practise is to define interfaces in a hierarchical way, since it reduces the complexity of the system to manageable proportions. However, the result of this flexibility and this development approach is that numerous unnecessary data copies occur between different modules. Operations such as data copying, servicing of interrupts, context switches, software compression, are currently often responsible for poor performance and high energy consumption.

Research has shown that there is no single approach for reducing energy in systems like the *Mobile Digital Companion* [5]. While low-power components and subsystems are essential building blocks for portable systems, little effort has been directed towards dedicated low-power hardware architectures by considering the system as a whole. 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.

The vision in the MOBY DICK project 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.

The approach to achieve such a system 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 placed in the 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. Thus, communication between components is not broadcast over a bus but delivered exactly where it is needed, work is carried out where and when the data passes through, bypassing the memory. We use dynamic programmable and adaptable devices that convert incoming or outgoing data streams. Modules are autonomously entering an energy-conserving mode and adapt themselves to the current state of the resources, the environment and the requirements of the user. To support this, the operating system must become a small, distributed system with co-operating processes occupying programmable components, 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.

IV. SYSTEM ARCHITECTURE OF A MOBILE DIGITAL COMPANION

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 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. In our infrastructure all connections are associated with a certain QoS.

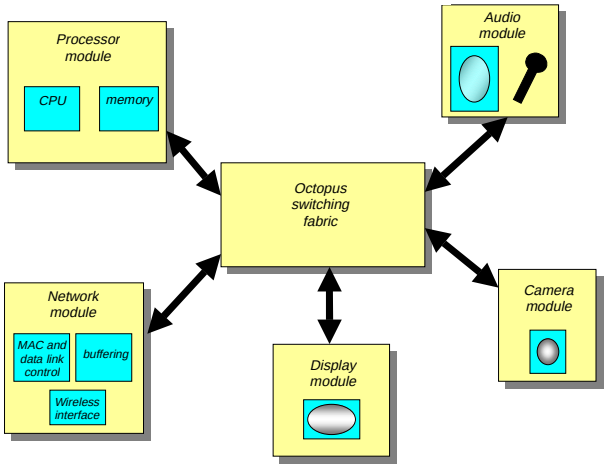

Figure 1: A typical Mobile Digital Companion architecture

Note that 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.

A. *Application Domain Specific Modules*

The *Mobile Digital Companion* 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. Note that our devices are not merely that of 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 these devices in multiple ways. First, each module is an autonomous sub-system that can operate without intervention from the main CPU. Secondly, it has a control processor that performs diverse operations, including connection management and energy management. Finally, most modules are able to adapt their behaviour to the 'wishes' of the client or application, and try to operate in the most efficient way.

Advantages – The main advantages of this approach with multiple autonomous modules are:

- *Efficient processing* – Instead of executing all computations in a general-purpose datapath, as is commonly done in conventional architectures, the energy- and computation-intensive tasks are executed on optimised modules. The modules are capable of efficiently performing device or application specific tasks.
- *Eliminate useless data copies* – When the data flows directly between the modules that need to process them, unnecessary data copies can be eliminated. 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 our system the 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 communicate with a local device.
- *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.
- *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.

Of course there are also some disadvantages. The most apparent disadvantage seems to be that using 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. Another disadvantage is that to make a new architecture gain wide acceptance, it must run a significant body of software, which will require a major software development.

B. Related Work

Different architectures have been proposed to address mobile multimedia computing. There are few systems that address energy reduction. Systems like the InfoPad [15] and ParcTab [9] 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 that uses a hardware architecture to localise data for both communication and video in order to increase performance and reduce energy consumption [11][12]. 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 granularity of these tasks is relatively small. Some examples include address generators, multiply-accumulate processors for computing vector dot products, etc.

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) [10]. We have adopted this concept, and believe 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 [7], *VuNet* from MIT [8] and the *APIC* architecture from the University of Washington [3]. However, their main motivation was performance and interoperability between (ATM) networks and devices. Our main motivation is reducing energy consumption. Furthermore, another main distinction is that our primary target is an architecture that can be used for a small portable computer, and not a high performance networked workstation. 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.

V. Architecture and Design of the *Octopus* Switch

In this section we will present the architecture and design of the *Octopus* switch that is used as the interconnection network of the Mobile Digital Companion. A key goal motivating the design has been simplicity, flexibility and energy efficiency.

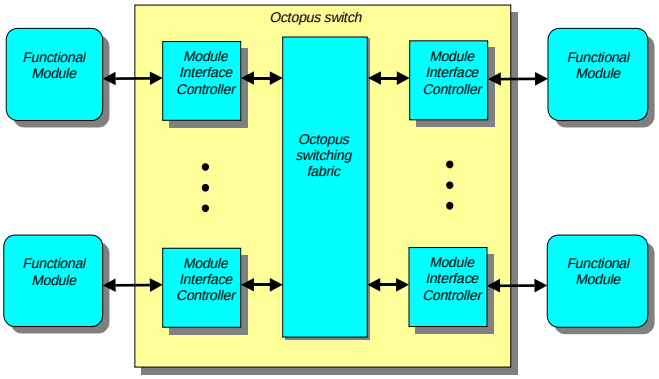

Figure 2: System architecture Mobile Digital Companion

Figure 2 shows an architectural view of the system of a *Mobile Digital Companion*. The *Octopus* switch provides the interconnection infrastructure between the functional modules in the system.

A. Octopus architecture

At the heart of the *Octopus* architecture is the *Octopus switching fabric*. The fabric connects eight *Module Interface Controllers* (MIC) that interface to the *Functional Modules*. These MICs decouple the modules from the *Octopus* switch and the other modules. The MICs contain small transmission and reception queues that store ATM cells. The MICs further

perform operations like connection setup and arbitration for the connections between the modules. In the architecture we have chosen to adopt the size and the structure of an ATM cell, i.e. 5 bytes header and 48 bytes payload. This format has shown to be sensible for several reasons, among a very practical one is that it allows for a simple connection to the ATM network that we are using. The size is small enough to allow for a fast and flexible scheduling of communication streams, and large enough to have a relatively small overhead.

B. Octopus Switching Fabric Architecture

The *Octopus* switching fabric behaves like a reconfigurable 8 port ATM switch. The switching provides a simple mechanism for the exchange of cells, regardless of their payload. The *Octopus* switch simply routes the traffic according to (a part of) the Virtual Channel Identifier (VCI) in the header. In contrast to full-blown ATM switching fabrics, the responsibility for ATM functions, such as VCI mapping and flow control, has been teased out of the switch fabric and assigned to the devices that plug into the switching fabric. The MICs are responsible for translating the VCI to the address of the destination module, and – when a connection has to be established – initialises the switching fabric with that address.

The architecture of the switching fabric is therefore basically very simple; most of the complexities are migrated to the Module Interface Controllers. A switch consists of 8 input sections, 8 output sections, and an 8 x 8 interconnection structure. A MIC is connected to an input section and an output section. The connection between the MIC and an input and output section is shared, so the *Octopus* switch can support half-duplex connections only.

The input section basically consists of only three registers: an *address register* that determines the output section of the connection, a *control register* for energy management and general control, and a *status register*. The dataflow of a connection passes the input section transparently. The control register determines when a request for a connection will be made. The status register will contain status information about the current connection. The output section contains two registers (*control register* and *status register*) and a *synchroniser* that synchronises the data stream so that the receiver MIC will handle it correctly in time. The status register is in fact shared with the input section. All requests for an output section are stored in this register and can be read by the MIC. Both the input and the output section share the *control unit*. This unit generates the *clock signal* for the attached MIC, and generates an *attention signal* when the MIC has to perform an operation. The attention signal is used to wake-up the MIC from sleep mode.

Figure 3 shows one input section and one output section that are involved in one connection between two modules. The interconnection network uses the address stored in the address register of the input section to select the output section it wants to make a connection to.

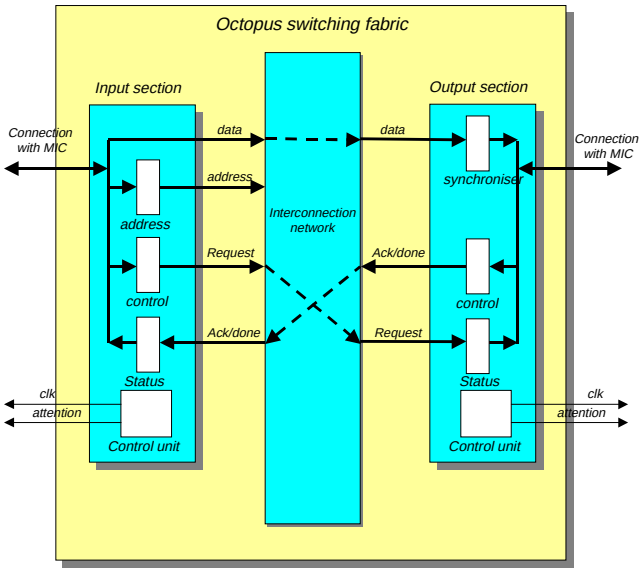

Figure 3: Structure of one input section and one output section of the Octopus *switch*

C. Module Interface Controller Architecture

The main task of the Module Interface Controller is to provide a data-path between the modules. The MIC basically contains the following units (see Figure 4): a *transmission queue* that stores ATM cells originating from the module in transit to the switching fabric; a *reception queue* receives the ATM cells coming from the switching fabric; a *VCI mapping table* that determines the destination module, and is indexed by the VCI in the header of the ATM cell; and an *arbiter* that performs the actual establishment of a connection and performs scheduling in case multiple simultaneous requests are received via the switch.

Using a typical communication stream between two MICs we will briefly describe the basic functions of these units.

When a MIC receives an ATM cell from the module it is attached to, it will first store this ATM cell in its *transmission queue*. The connection identifier contained in the VCI header of the ATM cell indexes the VCI mapping table to lookup the output port. The MIC will then establish a connection with this destination MIC. Cells with a VCI that is not known will be forwarded to a default module, which in general will be the CPU-module. The CPU-module contains the *connection manager* (*ConMng*) that is responsible for the management of all connections in the system. The ConMng uses special *management cells* to communicate with the MICs and the modules in the system. The VCI mapping table will be initialised by the ConMng using these management cells.

When the destination MIC receives a request for a connection, the *arbiter* determines when and whether the connection can be established. If there are multiple simultaneous connection requests, it uses a scheduling algorithm to determine which request can be honoured first.

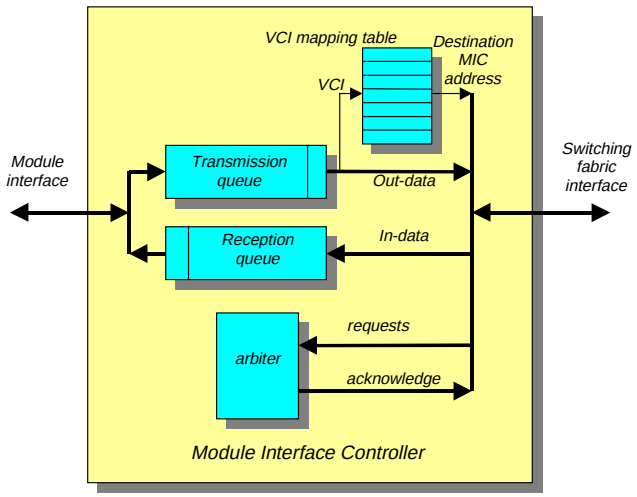

Figure 4: Module Interface Controller architecture

When the connection is established the source MIC forwards the data from its transmission queue over the *Octopus* switching fabric to the destination MIC. The destination MIC then first stores the ATM cells in its *reception queue* before it is further forwarded to the attached module. Note that the buffering of cells in the transmission and reception queue is not always required: when the module is capable of handling the cells at the required rate, then buffering can be omitted.

D. Connections

All communication between modules is based on connections. Prior to a communication transfer between two modules, a connection has to be set up. The ConMng schedules traffic between the modules. All connections are uni-directional. There are basically two types of connections:

- *Ad-hoc connections* – Modules that need no guaranteed flow of data use the ad-hoc connections. These type of connections are thus unreliable, and a higher protocol is needed e.g. for flow control. Ad-hoc connections are also used during the connection setup phase in which the module establishes a guaranteed connection with a different module.
- *Guaranteed connections* – Guaranteed connections are used to transfer data between modules that require bandwidth guarantees between the modules. Once a guaranteed connection is established, the actual data transfer still has to be announced by the source. In this way, the destination MIC can determine whether the reserved bandwidth is actually used, or otherwise assign that bandwidth to other connections.

Individual MICs in the *Octopus* switch can be remotely configured by using special control cells. The operating system running on the CPU-module issues these control-cells.

During *connection setup* of a guaranteed connection, the source module contacts the ConMng that it requires a channel to a module with a certain amount of bandwidth. It therefore transmits an ATM cell containing the request to the CPU-module using an ad-hoc connection. Upon receiving the request, the CPU-module will determine whether there is enough bandwidth available between both modules involved and that the connection can thus be established. The ConMng then might negotiate with the destination module to verify that this module is capable and prepared to connect. If the connection is not possible (because either the ConMng knows that there is not enough bandwidth in the *Octopus* switch, or the destination module cannot accept the connection), then the ConMng will reply to the requesting module that the connection is currently not possible. If the connection is possible, then it will inform the destination module and the source module that it a new connection has been set-up.

The aggregate bandwidth available inside the *Octopus* switch allows each module to communicate at a rate that is probably much higher than it can source or sink. The *Octopus* switch allows up to four parallel connections between eight modules. Given the small number of modules and the rate at which our present modules are able to generate traffic, this provides enough bandwidth.

In the prototype of the *Octopus* switch we use *static scheduling* for guaranteed connections, and *dynamic scheduling* for ad-hoc connections.

VI. IMPLEMENTATION *OCTOPUS* SWITCH

A key goal motivating the design has been simplicity and flexibility. Our goal was to build a testbed from off-the-shelf VLSI components that was easy to design and test. Therefore, the prototype interconnection module is build using a Field Programmable Gate Array of Xilinx (i.e. XC4010XL) surrounded by six low-end and low-power micro-controllers (i.e. Microchip PIC 16C66).

A. Implementation

The FPGA can be programmed to operate as the switching fabric that connects the modules. Each port of the switch is connected to one micro-controller, so in our testbed we have six micro-controllers. The micro-controllers implement the *Module Interface Controllers* (*MIC*) as described in the architecture. The datapath between the micro-controllers and the FPGA is eight bits wide. The internal datapath in the switch is also 8 bits wide. All data in the system is based on the size of an ATM cell. We have implemented the interconnection as a fully connected crossbar switch. The switch does not have ATM sized buffers, but just some synchronisation and pipeline registers.

In the current prototype the MICs perform several tasks: connection establishment with the other MICs that are connected to the switch, routing of traffic between the modules, scheduling of traffic at the output port of the switch destined for the module, and the actual data-transfer between the module's device and the input port of the switch.

The design has been implemented and tested. The design allows us to do experiment with and make performance measurements of various architectures and interconnection protocols. We used VHDL as a design tool.

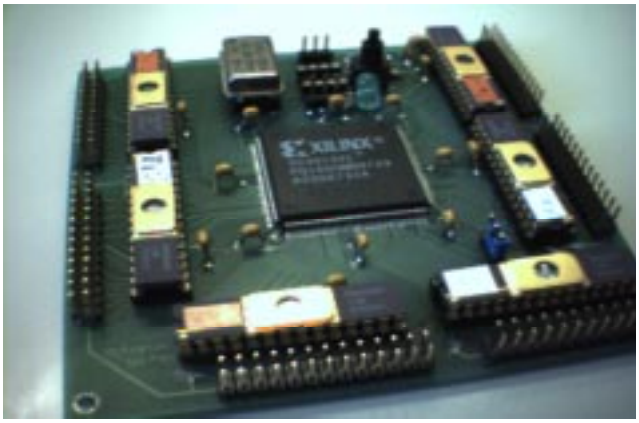

Figure 5: Testbed implementation Octopus.

B. *Performance*

We have measured the performance and energy consumption of the *Octopus* switch including the MICs. All measurements are performed with a clock frequency applied to the switch and MICs ranging from 0.1 to 32 MHz. This is equivalent to a raw data-rate per connection of 0.1 to 32 Mb/s. The *Octopus* switch is capable to support up to three simultaneous active connections when the connections are disjoint. This makes the total maximal throughput to be 96 Mb/s. This data-rate is more than sufficient to support all our expected data-streams on the mobile computer.

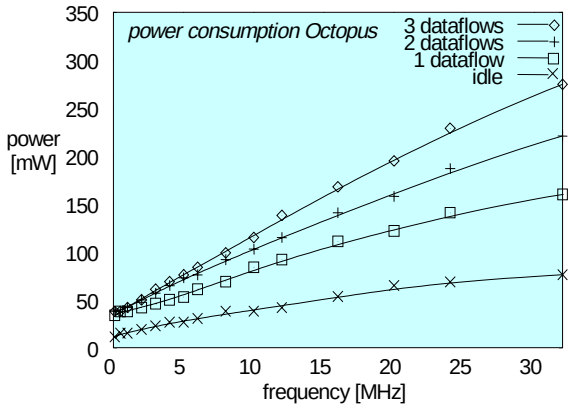

Figure 6: Power consumption Octopus switch

In Figure 6 we have plotted the measurement results for varying frequencies, and with a different number of simultaneous data-flows. In this setup a maximum of three disjoint connections were active, involving six modules, three sending and three receiving MICs. Since the connections are disjoint, no congestion can occur and all units are able to operate at maximum speed. The graphs show clearly that the energy consumption increases linearly with the frequency, which was expected. It also shows that the required amount of energy depends strongly on the number of dataflows in the switch. This effect is mainly due to our aggressive power management. All parts of the system that at some moment have no functionality, are in a low-power mode. The micro-controllers are in a very low-power mode when they don't have traffic, and their contribution to the energy consumption can be neglected. The switch however has always a large energy overhead due to the implementation in an FPGA.

The MICs operate in the following sequential phases:

- *Module I/O phase* in which data is transferred between the module and the MIC
- *Arbitration phase*, in which a connection between source and sink MIC is established
- *Data-transfer phase*, in which the data is transferred in the switching fabric
- *Release phase*, that acknowledges the transfer and releases the connection.

Note that the phases of both micro-controllers that take part in the connection operate in parallel, and that the effect is that there is a pipelined dataflow between the modules.

We have also measured the time in which the individual phases contributed to the total time needed to transfer one ATM cell. In the setup the arbitration phase took 9% of the time, the data-transfer phase, including the release phase, took 27%, and the I/O phase took 64%. Note that the interface with the switch is highly optimised, and that the interface with the module is more general, and requires more time.

The measurements show that the costs of having a flexible dynamic scheduling can be significant. The overhead introduced by the arbitration and release phase is about 30% when only one ATM cell is actually being transferred. A solution can be to have larger packets of multiple ATM cells, but this will lead to a bigger latency.

VII. SUMMARY AND CONCLUSIONS

In this paper we considered the problem of designing an architecture for a handheld mobile multimedia computer. 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.

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 modules, 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 where it is needed, work is carried out where the data passes through, bypassing the memory, modules are autonomously entering an energy conserving mode and adapt themselves to the current state of the resources and the requirements of the user. 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.

The interconnect of the architecture is based on a switch, called *Octopus*. In our model, the interconnection network is transparent and provides only a direct connection between two functional modules. The switch supports two basic connection

types: ad-hoc connections for traffic with no hard real-time requirements, and guaranteed connections for traffic with (hard) real-time requirements.

We have built a testbed of this architecture from off-the-shelf VLSI components that was easy to design and test. A key goal motivating the design has been simplicity, flexibility and energy efficiency. The performance of this prototype provides bandwidth guarantees and enough bandwidth for many multimedia applications. The power management that was used showed to be very effective. We have built the network module that 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 [6]. An energy-efficient MAC protocol is used that is able to provide near optimal energy efficiency for the mobile within the QoS constraints. Both the base station and the mobile use the operating system *Inferno* [4] which allows applications and system functions to be split and migrated dynamically between client and server.

Future research

The testbed showed that it is already feasible with standard components to build an energy efficient architecture that allows many devices in the system to be turned off (including the CPU), while still providing enough performance to support multimedia applications. Having an energy efficient architecture that is capable to handle 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 [14]. A model that encompasses different levels of granularity of the system is essential in the design of a 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.

REFERENCES

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

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

[3] 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.

[4] Dorward S., Pike R., Presotto D., Ritchie D., Trickey H., Winterbottom P.: "Inferno", *Proceedings COMPCON Spring'97*, 42nd IEEE International Computer Conference, 1997 (more information on URL: http://www.lucent.com/inferno).

[5] 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.

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

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

[8] Houh H.H., 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 (URL: http://www.tns.lcs.mit.edu/ViewStation/src/html/publications/JSAC95.html)

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

[10] Leslie I., D. McAuley, D. L. Tennenhouse: "ATM Everywhere?", *IEEE Network*, March 1993.

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

[12] 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.

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

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

[15] 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

.