Energy Management for Dynamically Reconfigurable Heterogeneous Mobile Systems

Paul J.M. Havinga, Lodewijk T. Smit, Gerard J.M. Smit, Martinus Bos, Paul M. Heysters

University of Twente
department of Computer Science
Enschede, the Netherlands

{havinga, smitl, smit, tbos, heysters}@cs.utwente.nl

Abstract

Dynamically reconfigurable systems offer the potential for realising efficient systems as well as providing adaptability to changing system requirements. Such systems are suitable for future mobile multimedia systems that have limited battery resources, must handle diverse data types, and must operate in dynamic application and communication environments. We propose an approach in which reconfiguration is applied dynamically at various levels of a mobile system, whereas traditionally, reconfigurable systems mainly focus at the gate level only. The research performed in the CHAMELEON project [1]aims at designing such a heterogeneous reconfigurable mobile system. The two main motivations for the system are 1) to have an energy-efficient system, while 2) achieving an adequate Quality of Service for applications.

1. Introduction

We are currently experiencing an explosive increase in the use of handheld mobile devices, such as cell phones, personal digital assistants (PDAs), digital camera's, global positioning systems, and so forth. Advances in technology enable portable computers to be equipped with wireless interfaces, allowing networked communication even while on the move. *Personal mobile computing* (often also referred to as *ubiquitous computing* [16]) will play a significant role in driving technology in the next decade. In this paradigm, the basic personal computing and communication device will be an integrated, battery-operated device, small enough to carry along all the time. This device will be used as a replacement of many items the modern human-being carries around. It will incorporate various functions like a pager, cellular phone, laptop computer, diary, digital camera, video game, calculator and remote control. An important issue will be the user interface: the interaction with its owner. To enable this, the device will support multimedia tasks like speech recognition, video and audio. Whereas today's notebook computers and personal digital assistants (PDAs) are self contained, tomorrow's networked mobile computers are part of a greater computing infrastructure. Furthermore, consumers of these devices are demanding ever-more sophisticated features, which in turn require tremendous amounts of additional resources.

The technological challenges to establishing this paradigm of personal mobile computing are non-trivial. In particular, these devices have limited battery resources, must handle diverse data types, and must operate in environments that are insecure, unplanned, and show different characteristics over time [6].

Traditionally, (embedded) systems that have demanding applications – those driven by portability, performance, or cost – require the development of one or more custom processors or application-specific integrated circuits (ASICs) to meet the design objectives. However, the development of ASICs is expensive in time, manpower and money. In a world now running on 'Internet time', where product life cycles are down to months, and personalization trends are fragmenting markets, this inertia is no longer tolerable. Existing design methodologies and

[1] This research is supported by the PROGram for Research on Embedded Systems & Software (PROGRESS) of the Dutch organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the technology foundation STW.

integrated circuit technologies are finding it increasingly difficult to keep pace with today's requirements. An ASIC-based solution would require multiple design teams running simultaneously just to keep up with evolving standards and techniques.

Another way to solve the problems has been to use general purpose processors, i.e. trying to solve all kind of applications running a very high speed processor. A major drawback of using these general-purpose devices is that they are extremely inefficient in terms of utilising their resources.

To match the required computation with the architecture, we apply in the CHAMELEON project an alternative approach in order to meet the requirements of future low-power hand-held systems. We propose a heterogeneous reconfiguration architecture in combination with a QoS driven operating system, in which the granularity of reconfiguration is chosen in accordance to the model of the task to be performed. In the CHAMELEON project [15] we apply reconfiguration at multiple levels of granularity. 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. Partitioning is an important architectural decision, which dictates where applications can run, where data can be stored, the complexity of the mobile and the cost of communication services. Our approach is based on a dynamic (i.e. at run-time) matching of the architecture and the application. Partitioning an application between various hardware platforms is generally known as hardware/software co-design. In our approach we investigate whether it is possible and useful to make this partitioning at run-time, adapting to the current environment of the mobile device.

1.1. Mobile multimedia systems

Today, the choice of mobile devices is largely limited to simple wireless phones on the one hand, to complex and bulky laptops with wireless communication capability on the other. While these devices serve their purposes, they are neither the most integrated nor the most general: their functionality is often limited, they can operate for just a short time, and they are incapable of fully exploiting the emerging integrated wireless networks. Even while current devices have the ability to communicate and process data, they are and by large primarily either data processing devices *or* communication devices. Simply shrinking the processing devices and communication devices, and packaging them together does not alleviate the architectural bottlenecks of integrated mobile multimedia devices [12]. The real challenge is to design a device where data processing and communication share equal importance.

Multimedia functionality is a driving force for many research challenges. For example, due to the size constraints on a portable computer, the user interface must be small. The shortage of area on a mobile device can cause us to trade buttons in favour of recognising the user's intention from analogue input devices such as handwriting, gesture and voice. Speech generation and recognition seem an ideal user interface since they require no surface area and allow hands-free and eye-free operation. However, general-purpose speech input and output places substantial storage and processing demands on a mobile device. Other research investigates the use of head-mounted virtual reality displays.

The trend in mobile system architectures 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. Both the notebook and the personal computer generally use the same standard PC operating system such as Windows (CE) 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 often use low power components.

Even though battery technology is improving continuously and processors and displays are rapidly improving in terms of power consumption; battery life and battery weight are issues that will have a marked influence on how hand-held computers can be used. Energy consumption is becoming the limiting factor in the amount of functionality that can be placed in these devices. More extensive and continuous use of network services will only aggravate this problem since communication consumes relatively much energy.

Another 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. 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 completely different wireless communication infrastructure when walking from their office to the

street. 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. Adaptability and programmability should be major requirements in the design of the architecture of a mobile computer.

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 more 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. Just integrating current PC or PDA architecture with a 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 centre of control, but also the centre of data flow in the system and a main cause of energy consumption. Another drawback is that highly optimised ASICs are inflexible, have a relatively long development time, and can only be applied optimally in a limited range of applications.

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 energy-efficient balance between flexibility and efficiency through the use of a hybrid mix of general-purpose and the application-specific approaches [6]. A system wide approach that covers the whole spectrum between physical layer and applications can provide significant energy savings [6].

1.2. Why reconfigurability in mobile systems

A key challenge of mobile computing is that many attributes of the environment vary dynamically. Mobile devices operate in a dynamically changing environment and must be able to adapt to the new environment. For example, a mobile computer will have to deal with unpredicted network outage or should be able to switch to a different network, without changing the application. It should therefore have the *flexibility* to handle a variety of multimedia services and standards (like different video decompression schemes and security mechanisms) and the *adaptability* to accommodate the nomadic environment, required level of security, and available resources. Mobile devices 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. They might require another air interface, other network protocols, and so forth. A *software radio* that allows flexible and programmable transceiver operations is expected to be a key technology for wireless communication. Reconfigurable systems have the potential to operate efficiently in these dynamic environments.

Reconfigurability also has another more economic motivation: it will be important to have a fast track from sparkling ideas to the final design. If the design process takes too long, the return on investment will be less. It would further be desirable for a wireless terminal to have architectural reconfigurability such that downloading new functions from network servers may modify its capabilities. Such reconfigurability would also help in field upgrading as new communication protocols or standards are deployed, and in implementing bug fixes [12]. Summarising, the key issues in the design of portable multimedia systems is to find a good balance between flexibility and high-processing power on one side, and area and energy-efficiency of the implementation on the other side.

2. The spectrum of solutions

2.1. Solution domains

In general there is a whole range of possible solutions for a certain problem. In a direct hardware implementation, problems are solved by mapping an algorithm directly into hardware. That is, the resulting hardware reflects the solution directly. We call this *fixed resources and fixed algorithms*. With a processor we have a way to implement almost any algorithm using a fixed set of resources. The processor provides fixed resources in the form of ALUs, floating-point units, etc. The processor solves problems by providing

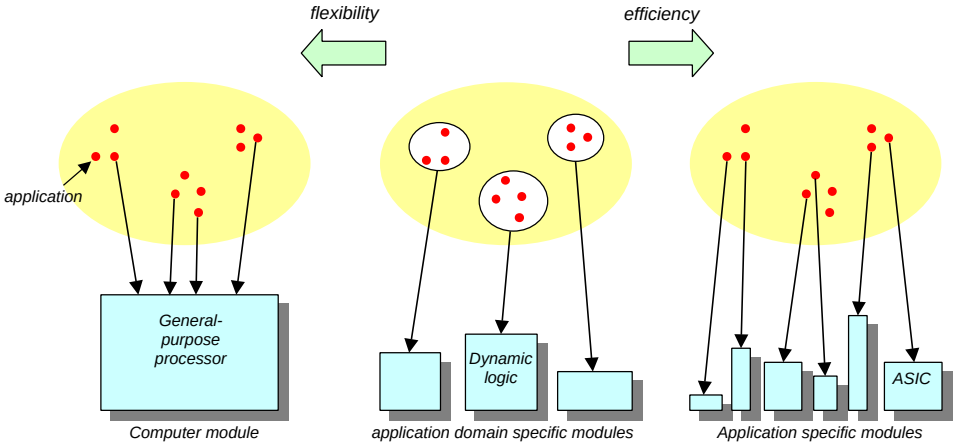

Figure 1: The spectrum of applications and hardware implementations [11].

limited, fixed hardware resources and by sharing those resources over time. In addition, the same hardware can support multiple problems. The processor solution is one with *fixed resources and dynamic algorithms*. These two problem-solving mechanisms have been competing with each other. There is a trade-off between the flexibility of the processor and the efficiency of the direct hardware implementation. If the application requires the ultimate in performance and efficiency, a direct hardware solution is preferred. But if the problem is too expensive to solve with a direct hardware approach, a processor-based solution is indicated. Between these two extremes there is a large world of dynamic logic. With dynamic logic the engineer can select both the resources and the algorithm. Both the resources and the algorithms may vary over time to solve the problem. *Dynamic resources and dynamic algorithms*. Dynamic logic can be efficient because the appropriate resources for the problem can be chosen to exactly match the algorithm at any point in time.

2.2. Approaches

Figure 1 shows three different approaches in the spectrum of applications and hardware implementations, and illustrates the trade-off between efficiency and flexibility.

General-purpose processors

A trend in computer architecture is to focus on flexibility with *high performance general-purpose processors* as this is the area in which a semiconductor vendor can enhance its status. Therefore, the architecture of a general-purpose processor is most widely studied, and optimisations for processor performance are the main goal. In general-purpose processors it is possible to map a large set of applications on the hardware simply by writing the right software for it.

While general-purpose processors and conventional system architectures can be programmed to perform virtually any computational task, they have to pay for this flexibility with a high energy consumption and significant overhead of fetching, decoding and executing a stream of instructions on complex general-purpose data paths. The energy overhead in making the architecture programmable most often dominates the energy dissipation of the intended computation [11].

Application specific modules (ASIC)

The goal of *application specific modules* is to optimise the overall cost-performance of the system, and not performance alone. Application specific solutions present the most effective way of reducing energy consumption and have shown to lead to huge power savings. Performing complex multimedia-data processing functions in dedicated hardware that is optimised for energy-efficient operation reduces the energy-per-operation by several orders of magnitude relative to software. However, the disadvantage of dedicated hardware is the lack of flexibility and programmability, their functionality is restricted to the capabilities of the hardware.

The technological challenges in the design of custom ASICs are usually significantly smaller than the design of general-purpose circuits. This means that high-performance custom chips can be designed and

manufactured at relatively low cost. However, this comes at the price of less flexibility, and consequently a new (re)chip design is needed for even the smallest change in functionality.

Application domain specific modules

From the discussion above, it is apparent that neither general-purpose processors, nor ASICs will be capable of satisfying the power and efficiency requirements of the future mobile devices. A hybrid solution with *application domain specific modules* could offer enough flexibility to be able to implement a predefined set of (usually) similar applications, while keeping the costs in terms of area, energy consumption and design time to an acceptable low level [1]. The modules are optimised for one specific application domain. A system designer can use the general-purpose processor for portions of algorithms for which it is well suited, and craft an application domain specific module for other tasks. Unused parts of the system must be switched off when not needed.

Summarising, an ASIC solution is too rigid, and a processor solution is too inefficient. In fact, we want to make the machine fit the algorithm as opposed to making the algorithm fit the machine. This is the area of *reconfigurable computing systems*.

2.3. Reconfigurable computing

Reconfigurable computing systems combine programmable hardware with programmable processors to capitalise on the strengths of hardware and software. The earliest configurable computing machine was proposed, designed, and implemented by Professor Gerald Estrin at UCLA in the early 1960s [4]. Estrin proposed the "Fixed plus variable structure computer", where some fixed hardware was dedicated to an inflexible abstraction of a programmable processor and a flexible component implemented digital logic. Today, the most common devices used for reconfigurable computing are *Field Programmable Gate Arrays* (FPGA). FPGAs present the abstraction of gate arrays, allowing developers to manipulate flip-flops, small amounts of memory, and logic gates.

Currently, many reconfigurable computing systems are based on FPGAs. However, these systems have a number of limitations:

- *Limited functionality* – Not all computations can be implemented efficiently with today's FPGAs: they are well suited to algorithms composed of bit-level operations, but they are ill suited to numeric operations, such as high-precision multiplication or floating point calculations. No circuit constructed with an FPGA can be as efficient as the same circuit in dedicated hardware. General-purpose processors (including digital signal processors) use optimised function units that operate in bit-parallel fashion on long data words. Compared with general-purpose processors, FPGAs are inefficient in performing ordinary arithmetic operations.
- *Gate capacity* – Available FPGAs provide an equivalent of 10K to 1000K gates. These devices are often large enough to experiment with the basic strategies, but limit the scope of the designs. Future FPGAs will be much larger and will have much broader application, including highly complex communications and signal-processing algorithms.
- *Configuration speed* – Most existing FPGAs use relatively slow paths for device configuration, and few have the ability to reconfigure only selective parts of the device. The configuration speed determines the characteristics of the computation model: it should change frequently enough to take advantage of programmability, but slowly enough to mask hardware configuration time. In general purpose machines, task switches occur constantly, 10 millisecond time-slices are normal. With reconfigurable hardware it is likely that task-switches are very expensive. Whereas saving CPU state consists mostly of backing up register contents, saving the state of a reconfigurable piece of hardware includes saving all configuration bits.
- *Memory structures and interface* – FPGAs currently provide little on-chip memory for storage of intermediate results in computation; thus many reconfigurable computing applications require large external memories. The transfer of data to and from the FPGA increases energy consumption and may slow down the computations.
- *Tools* – FPGAs employ the same design tools and methodologies as used for ASICs, including the use of hardware description languages (HDLs). Although HDLs are good at representing hardware, they are not particularly efficient at describing algorithms. This means that it is largely left to the designer to decide how to implement the algorithm in hardware, and then to describe this proposed implementation using the HDL.

To address some of these concerns, various researchers have proposed building a machine that tightly couples reconfigurable hardware with a

conventional microprocessor [2][5]. The reconfigurable hardware is used to speed up what it is good at, while the main processor takes care of all other computations.

3. System modelling of reconfiguration

In personal mobile devices, the system must meet the conflicting demands of compute intensive algorithms, communication intensive applications, and long battery life. In general, a designer tries to design a system to be optimal for a certain application and environment. The designer has to select a particular algorithm, design or select an existing architecture that can be used for it, and determine various parameters such as supply voltage and clock frequency. However, energy efficiency in mobile systems is not only a one-time design problem that needs to be solved during the design phase. In a mobile system, power management extents the notion of hardware/software co-design, since we have to face a *highly dynamic application and communication environment*. This multi-dimensional design space offers a large range of possible trade-offs.

3.1. Energy management

Adaptability and flexibility are two recurring items when we mention energy efficiency and performance on mobile multimedia computers. This leads to a key problem of *energy policy optimisation*, which must be the central issue in any *energy management system*. The policy is the algorithm that decides what measures have to be taken to minimise the energy consumption. Traditional power management schemes only decide how and when to activate or shut down system resources to minimise the energy consumption, depending on usage patterns and performance constraints.

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

3.2. Mobile system functionality and partitioning

In a reconfigurable mobile system functions can be dynamically migrated between functional modules such that an efficient configuration is obtained. Functionality can be partitioned inside a mobile system between a program running on the general-purpose CPU, dedicated hardware components (like a compressor or error correction device), and field programmable hardware devices (like FPGAs).

The *networked* operation of a mobile system opens up additional opportunities for decomposition to increase energy efficiency. One opportunity is offloading computation dynamically from the mobile system, where saving battery energy is at a premium, to remote energy-rich servers in the wired backbone of the network. In essence, energy spent in communication is traded for computation. For example, when we consider the transmission of an image over a wireless network, there is a trade-off between image compression, error control, communication, and energy consumption.

Partitioning of functions is an important architectural decision, which indicates where applications can run, where data can be stored, the complexity of the terminal, and the cost of the communication service. The key implication for this architecture is that the run-time hardware and software environment on the mobile computer and in the network should be able to support such adaptability, and provide application developers with appropriate interfaces to control it. Software technologies such as proxies and mobile agents, and hardware technologies such as adaptive and reconfigurable computing are likely to be the key enablers.

For multimedia applications in particular, a substantial reduction in energy consumption is possible as the computational complexity is high and their computation is regular and spatially local. 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 mobile system.

3.3. Granularity of programming model

The term *reconfigurability* has a strong association with *programmability*. Programmability originates

from the 'stored-program' concept used in the computer world. A program is typically a set of instructions that modify dynamically the behaviour of statically connected modules such as memories, registers and datapaths. This has gradually been extended to include the dynamic reconfiguration of interconnect networks and logic functionality as well. Nowadays, reconfigurable systems are often constructed from (arrays of) field-programmable devices (FPGAs). These can give orders of magnitude in performance improvement for specific computational kernels over traditional computers by providing programmability (reconfigurability) at the gate level. However, this may come at the price of an increase in area, increase in energy consumption, and lower programming ease.

It is arguable whether one should refer to the dynamic loading of a hardware configuration as programming or reconfiguration. The difference is indeed a bit subtle. Of course, even in dynamic logic there will be static hardware that will not change over time. Reconfiguration can be viewed as an extra layer of abstraction between programming and hardware. In programming, fixed instructions are used to solve a problem. The characteristics and meaning of these instructions is determined by the micro-program memory of the device. The 'instruction set' of the reconfigurable device is changed to meet the requirements of the algorithm. In some sense, there is a resemblance to dynamic micro-programming, in which the instruction set of the computer is changed by downloading microcode, rather than to run a single instruction set for all algorithms.

Reconfigurability can actually be applied at many layers in a system, and at multiple levels of granularity, in which each has its own preferred and optimal application domain [13]. In the CHAMELEON project we use the term reconfigurability to denote all possible changes to solve a problem at various levels of the system architecture. So we extend the traditional notion of reconfiguration from being just the layer between programming and hardware, to actually a vertical parameter that involves many layers in the system.

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

The 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 multiple grain-sizes of operations. These levels are illustrated in Figure 2, and will be discussed in more detail in Section 5.

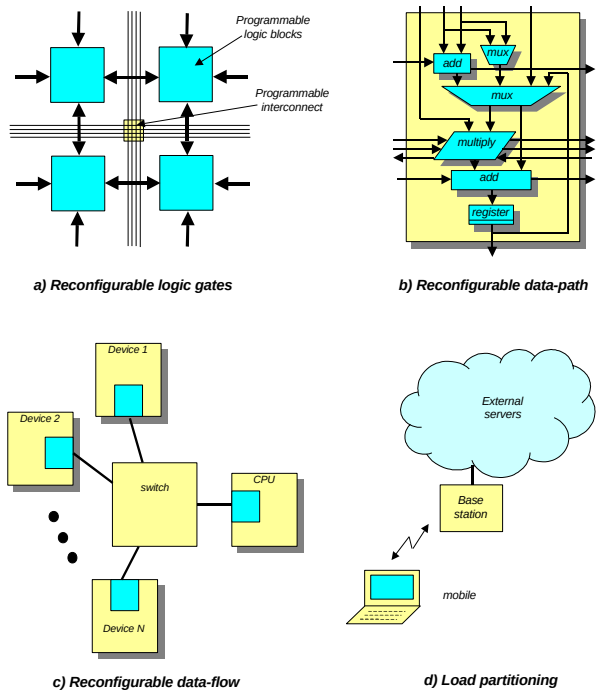

Figure 2: Granularity levels of reconfiguration.

At the gate level, we have the already mentioned *reconfigurable logic* of which FPGAs are a typical example. Other options for reconfiguration are at the *reconfigurable datapath*, in which the modules have various functions like multiplication and addition (the Field Programmable Function Array (FPFA) that will be introduced in Section 5 is a typical example of this); at the *reconfigurable dataflow* in which streams of

data are dynamically configured between various application domain specific modules; and at a high level we can identify reconfiguration in the *partitioning of the load* between the mobile and an external server.

To be able to handle this multi-level reconfigurability, a model is needed to describe the functionality and characteristics of the various processes, and also the interaction between them.

4. CHAMELEON system modelling

The reconfigurable system we envision consists of a heterogeneous dynamically reconfigurable architecture that spans many levels of the system. To deal with the hierarchical granularity, a system model is used in which Quality of Service (QoS) and energy consumption play a crucial role. This model is used to dynamically partition tasks of an application such that an energy efficient configuration is established.

4.1. Overview

An important issue of the research is to deal with the flexibility of the system. A trade-off has to be made where and when to run which (part of the) algorithm. Therefore, a high-level hierarchical – QoS based – model is needed to predict the behaviour of modules in terms of energy consumption, performance, cost etc., when running a certain set of applications. Besides describing the functionality of the modules and their ability to adapt (e.g. the effects on its energy consumption when the image compressor changes its frame rate, its resolution, or even its compression mechanism) this model also includes the interaction between these modules. Such a model is required to predict the overall consequences for the system when an application or functional module adapts its QoS, as in mobile systems, applications typically will have to adapt dynamically to the changing operating conditions. Using this model the inherent tradeoffs between for example performance, communication and energy consumption can be evaluated and a proper adaptation of the whole system can be made.

Applications that users run on a mobile need several functional resources of the system, such as processor, memory, wireless network interface, and compression/decompression logic. In our architecture we assume that such modules are programmable and can adapt to the demands of the applications and to the state of the environment, e.g. available bandwidth, bit error rate, available energy, etc. In general these modules are not independent and choices for the setting of one module may influence other modules. For example: when video has to be transmitted it can be compressed, which reduces the required bandwidth on the wireless network. However, more compression requires not only more processing power, it also needs better error-control. All these functional modules often have contradictionary effects on the resources needed, and a trade-off has to be made to find an optimal solution. Not only the parameters can be changed as it might also be profitable to migrate complete function from one module to another, possibly even to another machine. A complicating factor is that the dynamic nature of a mobile environment also demands an architecture that allows adapting to quickly changing conditions. The hardware and software architecture should be able to support such adaptability and should minimise the energy consumption by determining in run-time, things like:

- *Algorithm* – the most suitable algorithm(s) to execute the requested service(s).
- *Partitioning* – an appropriate partitioning of the algorithms over the different heterogeneous parallel processing units, which may include base stations, CPUs, FPGAs and dedicated hardware.
- *Parameters* – the most optimal parameters for execution of the algorithms.
- *Power state* – the suitable energy status mode of the different hardware components.

These decisions are related to each other and are strongly influenced by the environment, the current status of the system and the given constraints such as minimum required performance, real-time deadlines and minimum quality of service (QoS) parameters. For making these decisions a trade-off has to be made between computation, communication and initialisation costs (both time and energy) to adapt the status of the system dynamically to new circumstances in the best suitable manner. Multiple hierarchical levels will be used to hide complexity and achieve local adjustable systems when possible. A flexible communication mechanism is required to always offer the same communication interface independent of a specific processing unit, so allocation of algorithms to different processing units can be easily done.

4.2. System modelling

At all levels of abstraction, modern computing systems are built in terms of components and communication (or, at least, synchronisation) between components. Communicating systems imply

concurrency. Data dominated applications such as multimedia usually consist of a number of different complex sub-modules. In a high-level description (e.g. C) of the system the different sub-modules correspond to procedures that communicate to each other by exchanging several large array signals. Many applications can be structured as a set of processes or threads that communicate via channels. These threads can be executed on various platforms (e.g. general purpose CPU, DSP, FPGA, etc). In our research we apply channels as a basic communication mechanism between threads in a reconfigurable system. Channels first showed up in Tony Hoare's Communicating Sequential Processes (CSP) [9]. Occam is one language that includes them. More recently, they appeared in the language Limbo, that comes with Inferno [2] and also in reconfigurable systems [10]. In Inferno, channels and threads are fully integrated into the system and their use is really natural.

The creation of an abstract system model that will act as a frame for run-time partitioning of the functions to be performed using the available resources is of key importance to be able to deal with the dynamic application and communication environment of mobile systems.

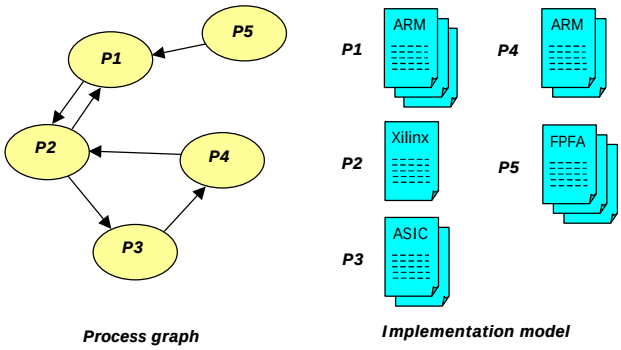

Figure 3: System model.

We use a *process graph* model, which abstracts system functionality into a set of *processes* represented as nodes in a graph, and represents functional dependencies among processes (states) with graph edges. The functionality of a process graph will be referred to as *task*. This model emphasises communication and concurrency between system processes. Edge and node labelling are used to enrich the semantics of the model. For instance, edge labels are used to represent communication bandwidth requirements, while state labels may store a measure of process computational requirements. Process graph models may include structural models, which describe systems as an assembly of tasks. The root of such a hierarchy of tasks is called the *application*. Figure 3 shows our system model with the process graph and the implementation model of the tasks. The *implementation model* describes characteristics and requirements of the available implementations of the processes on devices.

The costs associated with a process graph in the context of reconfiguration can be divided into *communication* costs between the processes, *computational* costs of the processes and *initialisation* costs of the task. The costs can be expressed in energy consumption, resource usage, and aspects of time (latency, jitter, etc). These costs are also described in the implementation model.

4.3. Mapping

In practice, most systems are realised using libraries of complex components. In a reconfigurable system, application instantiation consists first of all of finding a suitable partition of the system specification into the available resources of the system (processors, memories, reconfigurable logic, and ASICs) (*mapping phase*). The allocation of system functions into hardware and software during the design phase is already a complex task, to do it dynamically at run time, in response to the changed environment, available resources, or demands from the user, is an even more challenging task. The search of the 'best' mapping is typically a very hard problem, due to the size of the search space. The costs associated with the mapping cannot be ignored.

Mapping and algorithm selection assumes the existence of a library with multiple implementations for the computation of some (commonly used) processes. Furthermore, it is assumed that the characteristics (e.g. energy consumption and performance) of the library elements on a given architecture are known beforehand (and described in the implementation model like depicted in Figure 3). A characterisation of the energy consumption for each library element can be specified in a so called *Power State Machine*.

4.4. Operating system support

The life-time of an application in a reconfigurable system has four major phases. At first, during the *mapping phase*, a suitable partitioning of the system specification into the available resources of the system is made. Then, the *constellation phase* initialises the required hardware (loading the programs), and builds the communication structure required (setting up

communication channels and if necessary applying some plumbing to convert data formats). Then, the program can be executed (*execution phase*). In our model we do not assume that the mapping will be changed. So an application will run unaltered until its termination. Note, however that the application can use its resources in any way it likes. Also partial reconfiguration of its logic should be possible. In the *termination phase* all resources will be freed. An application may be terminated when the environment has changed in a way that a new mapping may be beneficial, or that the original mapping and performance cannot be maintained.

The research involves providing system level functions to describe the setup of communicating threads, which may now either run timeshared on a general CPU or in dynamically-setup special purpose logic that runs on reconfigurable hardware. The use of channels and processes running in whatever implementation model (software or hardware), will be made transparent for the application level programmers by the system level functions.

By first describing the processes and how they are connected and then letting the operating system decide on 'geographical' placement of the processes and buffers, multiprogramming will be supported and programs will be able to run on different setups of hardware (i.e. different amount of CPUs or available programmable logic).

QoS will be key factor in future hand-held multimedia computers. A QoS driven operating system should integrate QoS management into every (software or reconfigurable) module, and all modules are responsible for the collection of the QoS management information they require. In the design of a module, it is important to express both the resources it needs from other modules and the adaptation that is required based on which resources the module actually gets. The design of hardware/software modules for mobile terminal therefore focuses on co-operation and adaptation issues rather than just performance.

5. Target CHAMELEON architecture

In our model we have a number of modules that communicate through a communication channel. The modules can be a wide range of hardware devices, from general-purpose processors, via reconfigurable logic to ASICs. The communication channels can also be very diverse (e.g. ranging from a shared memory, via a bus, to wireless channels).

In the following we will describe three levels of reconfigurability that can be distinguished in the target CHAMELEON architecture and that will be part of our model.

5.1. Reconfigurable data path: the FPFA

As a starting point of the reconfigurable datapath we take the Field-Programmable Function Array (FPFA) architecture [8], but the research is not restricted to this architecture only; other reconfigurable architectures will be evaluated as well.

The operational environment of a FPFA is at a low level in the hierarchy, executing fine grain computational intensive processes. This exploits the 'law', which states that most execution time is spent on a small fraction of a program. We call *computational kernels* the inner loops of a computation, where most time is spent during execution. Each computational kernel is optimised as a stand-alone application and implemented on hardware that interfaces with the less frequently executed sections of the algorithm.

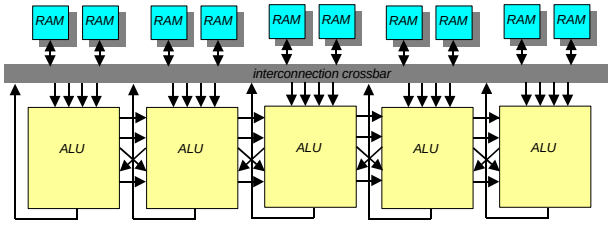

Figure 4: FPFA tile with five ALUs.

FPFAs have resemblance to FPGAs, but have a matrix of ALUs and lookup tables instead of Configurable Logic Blocks (CLBs). Basically the FPFA is a low power, reconfigurable accelerator for an application specific domain. Low power is mainly achieved by exploiting locality of reference. High performance is obtained by exploiting parallelism. A FPFA consists of interconnected processor tiles. Multiple processes can coexist in parallel on different tiles. Within a tile multiple data streams can be processed in parallel. Each processor tile contains multiple reconfigurable ALUs, local memories, a control unit and a communication unit. Figure 4 shows a FPFA tile with five ALUs.

The ALUs on a processor tile are tightly interconnected and are designed to execute the (highly regular) inner loops of an application domain. ALUs on the same tile share a control unit and a communication unit. The ALUs use the locality of reference principle extensively: an ALU loads its

operands from neighbouring ALU outputs, or from (input) values stored in lookup tables or local registers. Each memory has 256 20-bit entries. A crossbar-switch allows flexible routing between the ALUs, registers and memories.

5.2. Reconfigurable dataflow: the Octopus switch

In a multimedia system various communication streams (like audio and video) exist each with its specialised nature. Typically, multimedia streams are both communication-intensive and computation-intensive. In the MOBY DICK project [14] we already developed a system in which we have autonomous, reconfigurable modules such as network, video and audio devices, interconnected by a *switch* rather than by a bus.

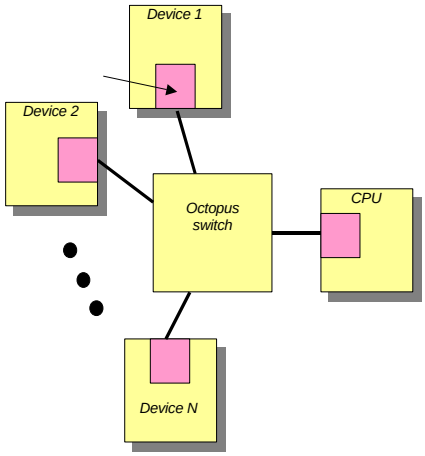

Figure 5: Reconfigurable dataflow: the Octopus switch.

We offload as much work as possible from the CPU to programmable modules placed in the data streams. In particular we 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 efficiently 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.

The interconnect of the architecture is based on a switch, called *Octopus* [7], which interconnects a general-purpose processor, (multimedia) devices, and a wireless network interface.

5.3. Load partitioning

One important advantage of integrating wireless communication with computing is that it facilitates user mobility and connectivity to the network while carrying a portable computer. The application layer in a mobile and wireless system is responsible for such things as partitioning of tasks between the fixed and mobile hosts, audio and video source encoding and decoding, and context adaptation in a mobile environment. A careful analysis of the data flow in the system and decomposition of the system functions between wireless terminal and network infrastructure can reduce energy consumption considerably.

6. Conclusion

Reconfigurable systems are suitable for the dynamic application and communication environment of wireless multimedia devices. Reconfigurable systems provides flexibility to design new equipment that can adapt to changing standards and algorithms once/year, add new features once/month or adaptively modify the algorithm once/microsecond based on the contents of the data stream.

Central in our approach is the matching between granularity of computation and architecture. This by necessity leads to a heterogeneous reconfigurable system that spans many levels of the system. A hierarchical system model is used in which Quality of Service and energy consumption play a crucial role. This model is used to dynamically partition tasks of an application such that an energy efficient configuration is established while achieving a sufficient Quality of Service of the running applications.

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] Chameleon Systems Inc., http://www.chameleonsystems.com.

[3] Dorward S., David Presotto, Howard Trickey, Rob Pike, Dennis Ritchie, Phil Winterbottom: "Inferno", *Proceedings of Compcon 1997*.

[4] Estrin G.: “Organization of Computer Systems: The Fixed-plus Variable Structure Computer”, *Proceedings*

of the Western Joint Computer Conference, pp. 33-40, 1960.

[5] Hauser J.R., Wawrzynek J., Garp: A MIPS Processor with a Reconfigurable Coprocessor, *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines* (FCCM '97), April 16-18, 1997

[6] Havinga P.J.M., "Mobile Multimedia Systems", *Ph.D. thesis University of Twente*, February 2000, ISBN 90-365-1406-1, www.cs.utwente.nl/~havinga/thesis.

[7] Havinga P.J.M., Smit G.J.M.: "Octopus: embracing the energy efficiency of handheld multimedia computers" , *proceedings fifth annual ACM/IEEE international conference on mobile computing and networking (Mobicom'99)*, pp.77-87, August 1999.

[8] Heysters P.M., Smit J., Smit G.J.M., Havinga P.J.M.: "Mapping of DSP algorithms on Field Programmable Function Arrays", *FPL '2000 (Tenth International Workshop on Field Programmable Logic and Applications)*, Villach, Austria, August 28 - 30, 2000.

[9] Hoare C. A. R.: "Communicating Sequential Processes", *in Proceedings of the Joint IBM University of Newcastle upon Tyne Seminar*, Newcastle upon Tyne, UK, 1978, pp. 145-156.

[10] Jones M. et al.: "Implementing an API for Distributed Adaptive Computing Systems", *FCCM Conference Virginia Tech*, 1999.

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

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

[13] Rabaey Jan M., "Reconfigurable Computing: The Solution to Low Power Programmable DSP", *Proceedings 1997 ICASSP Conference*, Munich, April 1997.

[14] Smit G.J.M., Havinga P.J.M.: "Lessons learned from the design of a mobile multimedia system in the Moby Dick project", *proceedings HUC2k (Second International Symposium on Handheld and Ubiquitous Computing)*, Bristol, UK, September 2000.

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

[16] Weiser M.: "Some computer science issues in ubiquitous computing", *Communications of the ACM*, 36(7): 75-84, July 1993.

Biography

Paul J.M. Havinga obtained his Ph.D. degree from the University of Twente on mobile multimedia systems in February 2000. He has worked at the Computer Science department of the University of Twente on various projects: on a multiprocessor system, on a central ATM network switch for multimedia applications, and during the last five years in the *MOBY DICK* project on mobile computing. Main emphasis in the latter project has been on the design of an energy-efficient architecture and wireless networking capabilities for handheld multimedia systems. Currently he is involved with the *Chameleon* project that deals with reconfigurable computing for handheld multimedia systems. His research interests include: mobile multimedia systems, energy-efficient wireless communication systems, Quality of Service, hardware architectures, hardware-software co-design, reconfigurable computing, security.

Lodewijk T. Smit was born in Rijssen, the Netherlands in 1974. He received the M.Sc. degree in computer science from the University of Twente, Enschede, the Netherlands, in 1996. From 1997 to 2000, he was with Akzo Nobel. Since February 2000 he joined the computer science department of the University of Twente, where he is currently working towards a Ph.D. degree. He is working on software support for mobile multimedia devices within the computer architecture and embedded systems group. His research focuses on energy efficient and effective architectures for mobiles, which are adaptable to the environment.

Gerard J.M. Smit works since 1996 in the MobyDick project that investigates new hardware and software architectures for battery-powered hand-held computers. Currently we concentrate our research on the power consumption issue and in particular low power communication. Furthermore, he is interested in using reconfigurable computer architectures for energy reduction. For a large number of multimedia data-conversion functions (e.g. video (de)compression, multi-channel digital radio reception and transmission, data encryption and digital signatures), reconfigurable computers are both faster and more energy-efficient than general-purpose CPUs.

Paul M. Heysters was born in 1973 in the city of Leeuwarden in the Netherlands. In 1992 he began a study computer science at the University of Twente in Enschede, the Netherlands. He started his final year project at Ericsson Mobile Networking in 1998 and at the end of same year he received his M.Sc. degree in Computer Science. In 1999 he joined Philips Home Networking, however at the end of 1999 he returned to the University of Twente to join a Ph.D. programme. Both the departments of Computer Science and Electrical Engineering participate in this programme.

He developed an interest in the integration of voice and data networks. Currently, his research focuses on reconfigurable architectures for handheld devices. Reconfigurability is explored to find an energy-efficient architecture that offers both flexibility and high performance.