Dynamic Reconfiguration in Mobile Systems

Gerard J.M. Smit, Paul J.M. Havinga, Lodewijk T. Smit, Paul M. Heysters, Michel A.J. Rosien

University of Twente
Department of Computer Science
Enschede, The Netherlands
{smit, havinga, smitl, heysters, rosien}@cs.utwente.nl

Abstract. Dynamically reconfigurable systems have the potential of 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 growth in the use of handheld mobile devices, such as cell phones, personal digital assistants (PDAs), digital cameras, 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*) 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. 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.

[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.

M. Glesner, P. Zipf, and M. Renovell (Eds.): FPL 2002, LNCS 2438, pp. 171-181, 2002.

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 [2].

Traditionally, (embedded) systems that have demanding applications e.g., driven by portability, performance, or cost lead to 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 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 kinds of applications running on 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 with the computation model of the task to be performed. In the CHAMELEON project 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.

The key issue 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. In this paper we give a state-of-the-art report of the ongoing CHAMELEON project. First we give an overview of the hardware architecture (section 2) and the design flow (section 3) and after that a typical application will be presented in the field of wireless communication (section 4).

1.1. Reconfiguration 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 a 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. Therefore it should 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 defined 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.

Until recently only a few reconfigurable architectures have been proposed for wireless devices. There are a few exceptions, for example, the Maia chip from Berkeley [1][4]. Most reconfigurable architectures were targeted at simple glue logic or at dedicated high-performance computing. Moreover, conventional reconfigurable processors are bit-level reconfigurable and are far from energy efficient.

However, there are quite a number of good reasons for using reconfigurable architectures in future wireless terminals:

- New emerging multimedia standards such as JPEG2000 and MPEG-4 have many adaptivity features. This implies that the processing entities of future wireless terminals have to support the adaptivity needed for these new standards.
- Although reconfigurable systems are known to be less efficient compared to ASIC implementations they can have considerable energy benefits. For example: depending on the distance of the receiver and transmitter or cell occupation more or less processing power is needed. When the system can adapt - at run-time - to the environment significant power-saving can be obtained [7].
- Standards evolve quickly; this means that future systems have to have the flexibility and adaptivity to adapt to slight changes in the standards. By using reconfigurable architectures instead of ASICs costly re-designs can be avoided.
- The cost of designing complex ASICs is growing rapidly, in particular the mask costs of these chips are very high. With reconfigurable processors it is expected that less chips have to be designed, so companies can save on mask costs.

Dynamically reconfigurable architectures allow to experiment with new concepts such as software-defined radios, multi-standard terminals, adaptive turbo decoding, adaptive equalizer modules and adaptive interference rejection modules.

Reconfigurability also has another more economic motivation: it will be important to have a fast track from sparkling ideas to the final design. Time to market is crucial. If the design process takes too long, the return on investment will be less.

2. Heterogeneous Reconfigurable Computing

In the CHAMELEON project we are designing a heterogeneous reconfigurable System-On-a-Chip (SOC). This SOC contains a general-purpose processor (ARM core), a bit-level reconfigurable part (FPGA) and several word-level reconfigurable parts (FPFA tiles; see Section 2.3) (see Figure 1).

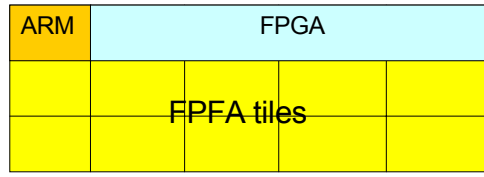

Figure 1: Chameleon heterogeneous architecture

We believe that in future 3G/4G terminals heterogeneous architectures are needed. The main reason is that the efficiency (in terms of performance or energy) of the system can be improved significantly by mapping application tasks (or kernels) onto the most suitable processing entity. Basically we distinguish three processor types in our heterogeneous reconfigurable system: bit-level reconfigurable units, word-level reconfigurable units, and general-purpose programmable units. The programmability of the architecture enables the system to be targeted at multiple applications. The architecture and firmware can be upgraded at any time (even when the system is already installed). In the following sections we will discuss the three processing entities in more detail.

2.1. General-Purpose Processor

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. However, general-purpose processors are very good in control type of applications; e.g. applications with frequent control constructs (if-then-else or while loops).

2.2. Bit-Level Reconfigurable Unit

Today, *Field Programmable Gate Arrays* (FPGAs) are the common devices for reconfigurable computing. 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. FPGAs are particularly useful for applications with bit-level operations. Typical examples are PNcode generation and Turbo encoding.

2.3. Word-Level Reconfigurable Units

Many DSP-like algorithms (like FIR and FFT) call for a word-level (reconfigurable) datapath. In the CHAMELEON project we have defined a word-level reconfigurable datapath, the so called Field-Programmable Function Array (FPFA) [3] [8].

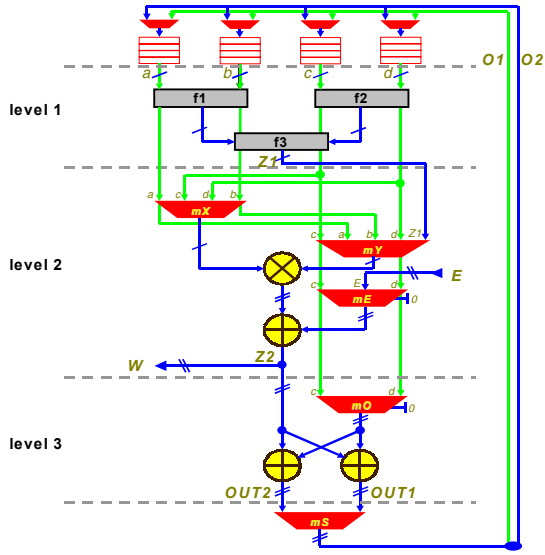

Figure 2: Structure of one ALU of the FPFA

It consists of multiple interconnected processor tiles. Within a tile multiple data streams can be processed in parallel in a VLIW manner. Multiple processes can coexist in parallel on different tiles. Each processor tile contains five reconfigurable ALUs, 10 local memories, a control unit and a communication unit. Figure 3 shows a FPFA tile with the five ALUs. Each FPFA can execute a fine grain computational intensive process. We call the inner loops of a computation, where most time is spent during execution, *computational kernels*. A computational kernel can be mapped onto an FPFA tile and interfaces with the less frequently executed sections of the algorithm that may run on the general-purpose processor.

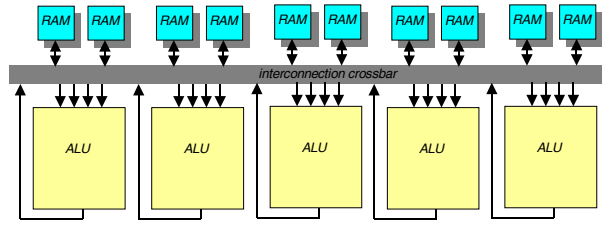

Figure 3: FPFA tile with five ALUs.

FPFAs have resemblance to FPGAs, but have a matrix of word-level reconfigurable units (e.g. 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.

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. The ALUs are relatively complex (see Figure 2), for instance they can perform a multiply-add operation, a complex multiplication or a butterfly operation for a complex FFT in one cycle.

2.4. Implementation Results

The FPFA has been designed and implemented. The FPFA architecture is specified in a high-level description language (VHDL). Logic synthesis has been performed and a one FPFA tile design fits on a Xilinx Virtex XCV1000. In CMOS .18 one (un-optimized) FPFA tile is predicted to have an area of 2.6 mm^2 and it can run at least at 23 MHz. In this technology we can have approx. 20 FPFA tiles in the same area as an embedded PowerPC. For the prototype we probably will use CMOS .13 technology. Several often-used DSP algorithms for SDR have been mapped successfully onto one FPFA tile: e.g. linear interpolation, FIR, correlation, 512-point FFT and Turbo/SISO decoding. Of course, these are only a few of the algorithms that the FPFA should be able to handle.

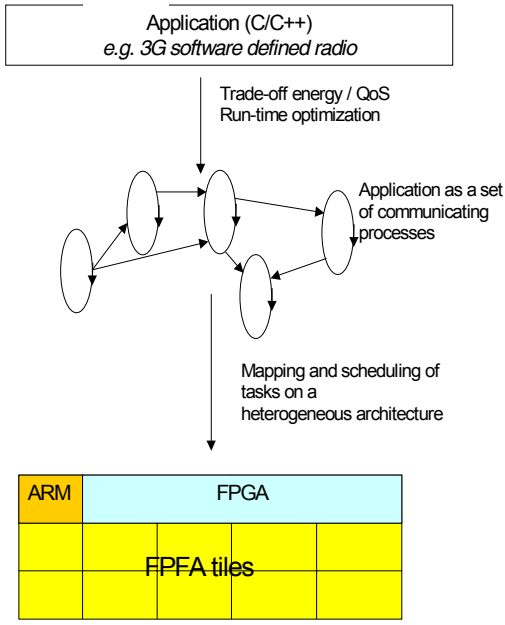

Figure 4: Chameleon design flow

3. CHAMELEON System Modeling

The design of the above-mentioned architecture is useless without a proper tool chain supported by a solid design methodology. At various levels of abstraction, modern computing systems are defined in terms of processes and communication (or, at least, synchronisation) between processes. 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, FPFA, FPGA, etc).

We use a Kahn based *process graph* model, which abstracts system functionality into a set of *processes* represented as nodes in a graph, and represents functional dependencies among processes (channels) with graph edges. The functionality of a process graph will be referred to as *task*. This model emphasizes communica-tion and concurrency between system processes. Edge

and node labeling are used to enrich the semantics of the model. For instance, edge labels are used to represent communication band-width requirements, while state labels may store a measure of process computational requirements. Process graph models may include hierarchical models, which describe systems as an assembly of tasks. The root of such a hierarchy of tasks is called the *application*.

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 *initialization* costs of the task. The costs can be expressed in energy consumption, resource usage, and aspects of time (latency, jitter, etc).

The mapping of applications (a set of communicating tasks) is done in two phases. In the first phase (macro-mapping) for each task the most- (or near) optimal processing entity is determined. This phase defines what is processed where and when. This phase is supported by a number of profiling tools. In the second phase (micro-mapping) for each task a detailed mapping is derived to the platform of choice.

3.1. Macro-mapping

In practice, most complex systems are realized using libraries of components. In a reconfigurable system, application instantiation consists first of all of finding a suitable partition of the system specification into parts that can be mapped onto the most appropriate resources of the system (processors, memories, reconfigurable entities). Because of the dynamics of the mobile environment we would like to perform the macro-mapping *at run-time*. In Section 4 we will show an example how macro-mapping can be used to save energy.

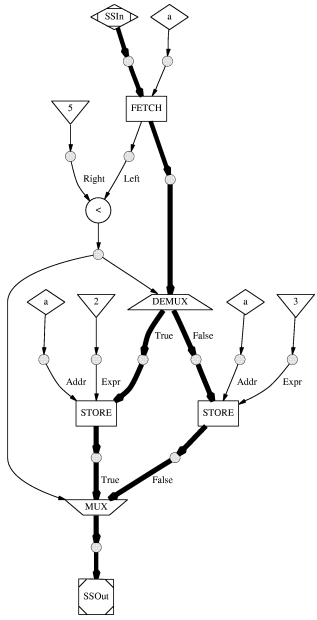

Figure 5: DFGC of a C statement

The traditional allocation of system functions into hardware and software during the design phase is already a complex task, doing 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. Moreover, the costs associated with the mapping cannot be ignored.

Macro-mapping and algorithm selection assumes the existence of a library with multiple implementations for the computation of some (commonly used) processes or adequate profiling tools. Furthermore, it is assumed that the characteristics (e.g. energy consumption and performance) of the library elements on a given architecture are known beforehand.

3.2. Micro-mapping

Once the designer has decided what to map where, the micro-mapping comes into play. This is normally done at design-time, as this is a quite time-consuming operation. We assume that all processes are written in C. The mapping of C processes to the general-purpose processor is straightforward; we use the standard GNU tools for that. In this paper we concentrated on mapping C tasks to the FPFA.

We first translate C to a Control Data-Flow Graph (CDFG). In this graph control (order of execution) and data are modeled in the same way. As an example Figure 5 shows the automatically generated CDFG graph of a simple C statement (IF (A<5) A=2; ELSE A=3;). The bold arrows in the figure indicate the State-Space of the architecture, where the State-Space denotes the mathematical representation of the C memory model. Store and fetch are operations on the State-Space. We have defined several behavior-preserving transformations on these graphs e.g. constant propagation, loop unrolling and removing of intermediate variables. The FPFA tile can also be described in terms of a CDFG graph; the *architecture graph*. With the help of these transformations we can derive a simple CDFG that is suitable for mapping onto an architecture graph. In general the mapping of algorithm graphs to an architecture graph is NP complete. Fortunately, the size of the algorithms tasks is usually quite small (no more than two nested loops). We have performed several mappings by hand, and currently we are implementing a method to automate this process.

4. Sample Application: Reconfiguration in a Wireless Terminal

In this section we show how a reconfigurable architecture and macro-mapping can be used to save energy in wireless terminals. 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 contrast to ASIC implementations reconfigurable architectures offer the possibility to tune the settings of a software-defined radio (SDR) *at run-time* to the current wireless environment, even in continuously changing conditions. In this way overkill is avoided, which can be translated in a reduction in energy consumption for a mobile, or savings in the necessary computing resources for a base station.

To support this run-time adaptive behavior, trade-offs between different parameter sets should be made to determine the most optimal set for the current situation. We introduced a control system, which is based on a model that selects at run-time a set of parameters that minimizes the cost, while satisfying the requested quality.

In our initial approach, we reduce the set of performance indicators for a SDR to two: the *quality* and the *required effort*. Figure 6 depicts the relationship between the quality and the required effort. The dots represent a certain setting of the system in the quality/cost space. Due to the dynamic external environment of a SDR, the wireless link conditions change constantly and therefore the quality of the output of the SDR will change. In Figure 6, this implies that the dots will move in horizontal direction as a function of time. If the conditions of the external environment become worse, then a dot will move to the right and if the conditions become better, the dots will move to the left.

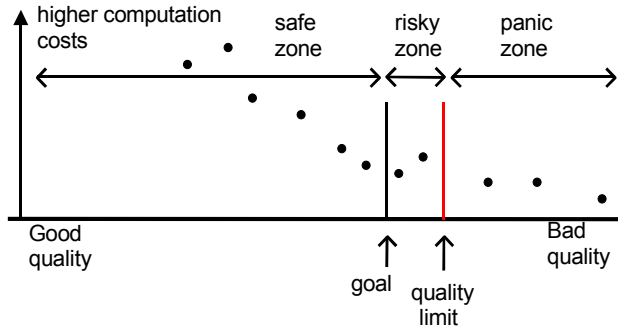

Figure 6: Quality vs cost trade-off

Given a specific application, a certain minimum quality limit will apply. A quality worse than this quality limit is not acceptable. Furthermore, a certain area left of the quality limit will be considered as a risky zone in the sense that the system is not allowed to stay too long in this area.

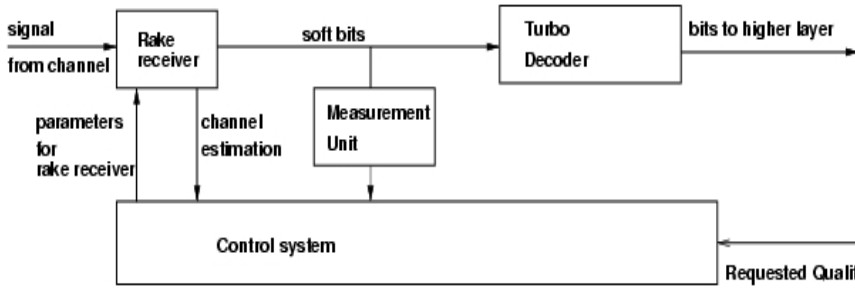

Figure 7: Rake receiver Turbo decoder combination

Therefore, when an optimal setting is determined, the optimization goal for the quality is left of the risky zone to maintain a certain ‘safety margin’, because otherwise a quality violation may occur, when the external environment changes only slightly. The optimal parameter setting in Figure 6 is the first dot on the left side of the goal line: it has the lowest costs (e.g. energy consumption) that satisfies the required quality of service. The quality limit is mainly dictated by the application and cannot be changed. For a specific application, the ‘goal’ line is at a fixed distance from the quality limit. Currently, the design of most SDR ensures that worst-case situations are handled well, which provides overkill in 'normal' situations. So, these SDRs operate almost always in the left most part of the safe zone that is mentioned in Figure 6. The added value of our approach is that we provide a run-time optimization to minimize the operation costs.

4.1. Example in Detail

Our control system will be demonstrated with a wideband code division multiple access (WCDMA) RAKE receiver [5], in combination with a turbo decoder [6], as

shown in Figure 7. This combination can be used in an UMTS terminal or base station. For an in-depth discussion about the simulation of this system, see [7]. The quality is expressed in bit error rate (BER) and the costs are expressed in number of operations needed for the datapath (excluding control costs). In our current implementation input samples are 6 bits wide and 5 fingers can be executed in one single tile. The main part of the Turbo-decoder, (SISO module) can also be executed in one tile. We built a control system that adapts the system at run-time to the dynamic external environment. The goal is to operate with minimum use of resources and energy consumption, while satisfying an adequate quality of service. The control system is based on a model, which selects the most optimal configuration based on off-line gathered information and on-line measurements.

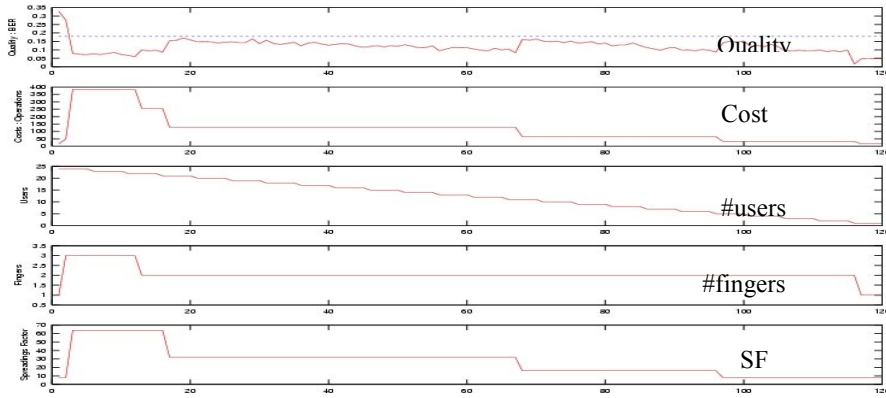

Figure 8: Simulation results

Figure 8 shows simulation results of the behaviour of the control system, when the number of users decreases. In this situation, the external environment becomes better. The five graphs in this figure represent (top - down): the quality of the output of the RAKE receiver expressed in BER, the costs of the RAKE receiver expressed in number of operations, the number of simultaneously transmitting users, the number of fingers of the RAKE receiver and the spreadingsfactor used by the RAKE receiver. On the horizontal axis the sequence number of the transmitted block is shown. Blocks contain 1000 bits.

As can be seen from the figure, the number of fingers and the spreadingsfactor are decreased as soon as possible, whereas the quality is maintained below the BER quality limit of 0.18. Note that the costs for the worst-case situation are much higher than the average costs, which indicates a considerable energy saving.

4.2. Evaluation

Compared to a system that is optimized for the worst-case situation, substantial savings can be achieved. In our simulations, savings of a factor three were no exception. The control system presented here has been applied to a specific RAKE/turbo case. The data required by the control system were: the quality limit

(application dependent), the width of the risky zone (application dependent), and per parameter the: the range and the estimated gain when a parameter changes. The control system can be used as a general framework. For example, the turbo decoder can very easy be replaced by another kind of forward error decoder (e.g. Viterbi). The presented control system has a number of attractive properties: it is able to handle an unpredictable time-variant changing environment with a lot of parameters, it is simple and therefore possible to compute at run-time, it is suitable for a dynamically reconfigurable mobile terminal with scarce energy resources, it is fast enough (within tenths of ms) to react to a fast changing environment.

5. 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/millisecond 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] 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.

[3] 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.

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

[5] G. L. Turin. "Introduction to spread-spectrum anti-multipath techniques and their application to urban digital radio", *Proc. of the IEEE, 68(3):*328–353, Mar. 1980.

[6] C. Berrou and A. Glavieux. "Near optimum error correcting coding and decoding: Turbo-codes", *IEEE Transactions on Communications, 44*(10):1261–1271, Oct. 1996.

[7] L. T. Smit, G. J. Smit, P. J. Havinga, J. L. Hurink, and H. J. Broersma. "Influences of rake receiver/turbo decoder parameters on energy consumption and quality" *2002 International Conference On Third Generation Wireless and Beyond,* pp. 175-180, May 2002.

[8] P.M. Heysters, H. Bouma, J. Smit, G.J.M. Smit, P.J.M. Havinga, "A reconfigurable function array architecture for 3G and 4G wireless terminals, *2002 International Conference On Third Generation Wireless and Beyond,* pp. 399-404, May 2002.