Advances in Radio Science (2003) 1: 165–169

Advances in Radio Science

Study of heterogeneous and reconfigurable architectures in the communication domain

H. T. Feldkaemper, H. Blume, and T. G. Noll

Chair of Electrical Engineering and Computer Systems, RWTH Aachen University, Schinkelstr. 2, 52062 Aachen, Germany

Abstract. One of the most challenging design issues for next generations of (mobile) communication systems is fulfilling the computational demands while finding an appropriate trade-off between flexibility and implementation aspects, especially power consumption. Flexibility of modern architectures is desirable, e.g. concerning adaptation to new standards and reduction of time-to-market of a new product. Typical target architectures for future communication systems include embedded FPGAs, dedicated macros as well as programmable digital signal and control oriented processor cores as each of these has its specific advantages. These will be integrated as a System-on-Chip (SoC). For such a heterogeneous architecture a design space exploration and an appropriate partitioning plays a crucial role.

On the exemplary vehicle of a Viterbi decoder as frequently used in communication systems we show which costs in terms of *ATE* complexity arise implementing typical components on different types of architecture blocks. A factor of about seven orders of magnitude spans between a physically optimised implementation and an implementation on a programmable DSP kernel. An implementation on an embedded FPGA kernel is in between these two representing an attractive compromise with high flexibility and low power consumption. Extending this comparison to further components, it is shown quantitatively that the cost ratio between different implementation alternatives is closely related to the operation to be performed. This information is essential for the appropriate partitioning of heterogeneous systems.

1 Introduction

Today's mobile communication standards like GPRS, EDGE, UMTS or CDMA2000 enable high-performance wireless applications as they offer mobile high-speed data rates. For those systems it is required to provide a high degree of flexibility and highest computational capabilities. But the computational demands are beyond the capacities of today's programmable platforms. For example in (Hausner, 2001) the increase in computational demands evolving from one standard to the next one has been compared to the increase in performance of digital signal processor kernels (DSPs). At the time of the standard release the computational requirements are constantly beyond the available performance of on-chip DSP kernels (Fig. 1). Even more severe, future generations of communication standards tend to strengthen this computational gap.

In addition to these computational demands a high degree of flexibility is required because of:

- Avoidance of the design of a new platform for future products within the design cycles as sufficient flexibility is provided for an adaptation to changing demands e.g. for the integration of new features from one product generation to the next one, or for required adaptations to standard updates (e.g. variation of chiprate with evolution of a communication standard like UMTS). By this, short innovation cycles and longer product lifetimes can be achieved.
- Runtime adaptivity due to switching between several cells/standards (e.g. handover between standards), respectively adaptation to channel quality.

Therefore, the underlying architecture of a communication system has to include architecture blocks that support these aspects of flexibility. Dedicated hardware implementations offer orders of magnitude better performance with respect to throughput and power dissipation. But flexibility of those implementations is restricted to weak programmability (e.g. switching of coefficient sets) considered at design time. Altogether, a well-balanced architecture of a communication system has to include different types of architecture blocks in order to provide the required performance at reasonable costs (e.g. area and power dissipation) on one hand and ensuring sufficient flexibility on the other. Future communication systems will consist of a variety of system blocks. These

Correspondence to: H. T. Feldkaemper
(feldkaemper@eecs.rwth-aachen.de)

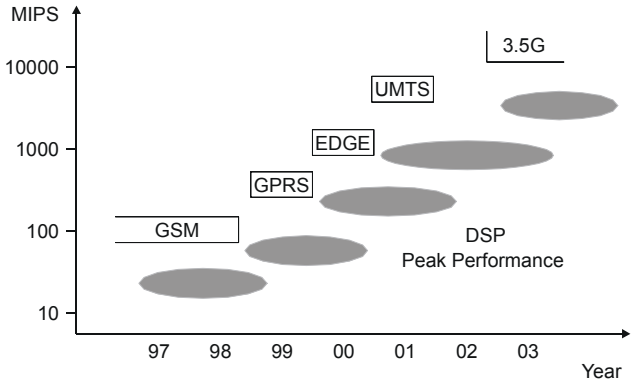


Fig. 1. Yearly increase in computational complexity and DSP-performance (Hausner, 2001).

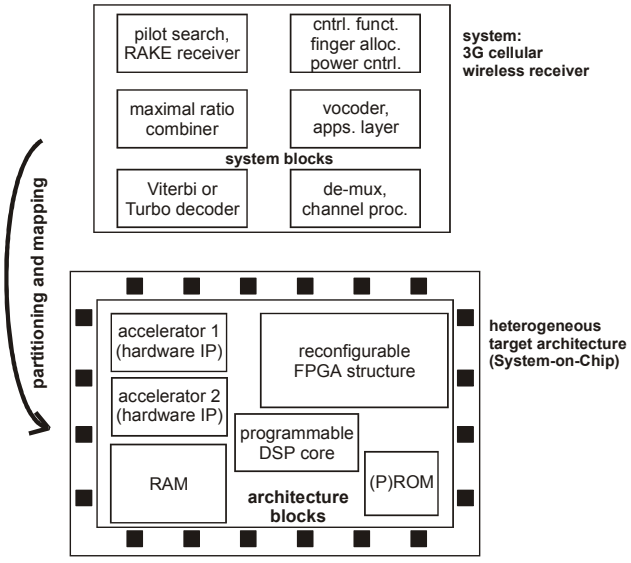


Fig. 2. Partitioning and mapping from system to architecture.

systems have to be partitioned and mapped to the architecture blocks of such heterogeneous target architectures (Fig. 2). In order to meet the challenging demands sketched above it is important to elaborate methodologies which assist designers with metrics and with an early assessment of the capabilities of a given platform.

The next section lists classical quantitative metrics. An exemplary implementation of a key component in digital communication systems – the Viterbi decoder – is shown in the following section considering principle architecture blocks. Finally, the results for the different implementation alternatives are discussed.

2 Possible evaluation metrics for quantitative optimisation

For the evaluation of basic operations in the field of digital signal processing several aspects have to be considered. Various metrics considering silicon area A and symbol rate

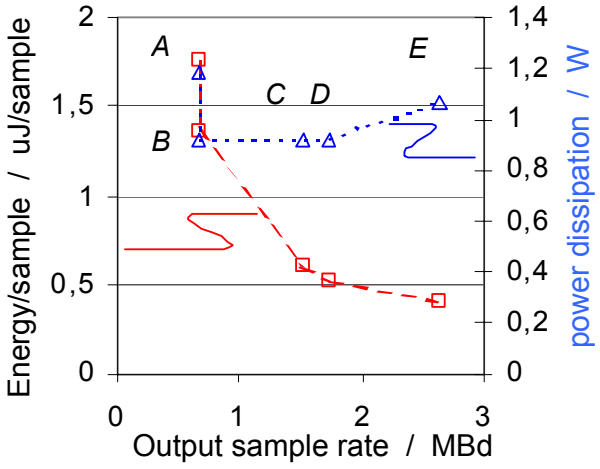


Fig. 3. Software optimisations for the TM 1300.

$1/T$ have been proposed in the past e.g. (De Hon, 2000). Due to the importance of low-power operation especially in the communication domain the energy per output sample $E_{persample}$ has to be taken into account. Therefore, a combined cost function is taken exemplarily in the following as evaluation metric

$$cost = A \cdot T \cdot E_{per\ sample}.$$

A common primary design challenge is to implement a system achieving a specified throughput rate at minimised energy per sample and silicon area. According to scaling theory (general scaling for short channel devices) all parameters are normalised with respect to the minimal feature size L_{min}

$$T_{norm} = T \cdot \left(\frac{1\mu\mathrm{m}}{L_{min}}\right), \quad A_{norm} = A \cdot \left(\frac{1\mu\mathrm{m}}{L_{min}}\right)^2,$$

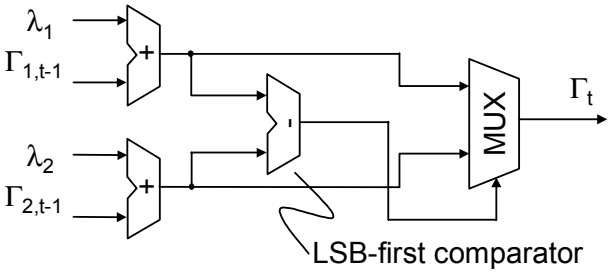
$$E_{per\ sample,norm} = E_{per\ sample} \cdot \left(\frac{1\mathrm{V}}{V_{DD}}\right)^2 \cdot \left(\frac{1\mu\mathrm{m}}{L_{min}}\right)^{0.75}.$$

Overhead for time-sharing e.g. of free computational resources of a DSP or parallelisation is neglected here.

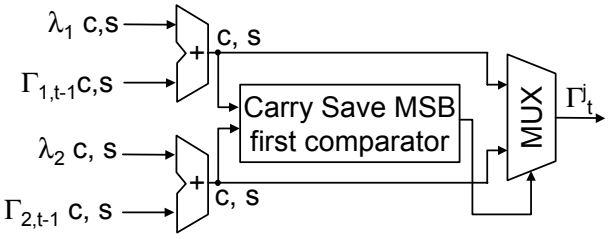
3 Exemplary vehicle: Viterbi decoder

In order to perform a fair comparison between the costs of an algorithm mapped on different architecture blocks, the implementations need to be optimised individually to the specific architecture block, e.g. by including algorithmic transformations. This will be shown in the following applying the exemplary vehicle of a Viterbi decoder.

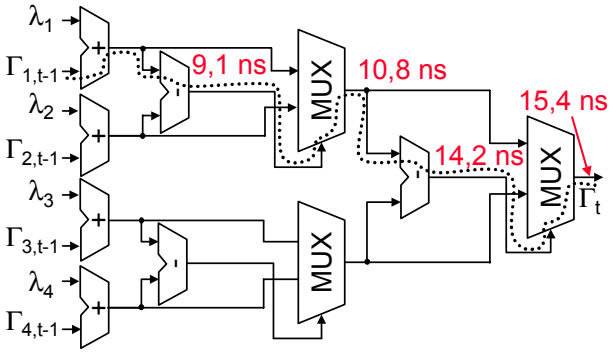
The architecture of the Viterbi decoder can be divided into three basic units: the branch metric unit (BMU), the path metric unit (PMU) and the survivor memory unit (SMU). The following subsections review optimisation techniques and present the costs of a rate-1/2 64-state Viterbi decoder with a survivor memory length of 128 and a path-metric word length of eight bits mapped to a DSP, an FPGA and a physically optimised macro.



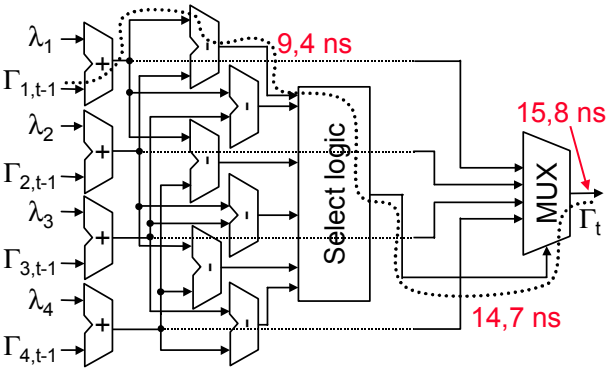
(a) radix-2 PE



(b) serial carry save radix-2 PE



(c) radix-4 PE with 3 comparisons with critical path timing



(d) radix-4 PE with 6 comparisons with critical path timing

Fig. 4. Simplified Signal flow graphs (SFG) of different PEs for the PMU.

3.1 Optimisation for a DSP implementation

Due to high data rates in the communication domain, very long instruction word (VLIW) DSPs like the TM 1300 processor are well suited for Viterbi decoding (Ahmad Khan et al., 2000). This processor features five issue slots (i.e. at max. five instructions/cycle) and 27 functional units dedicated to frequently used basic operations ranging from simple ALU operations to square root or division. It has been applied here in order to implement the Viterbi decoder. Special attention has been directed to the power dissipation as this is one of the most decisive factors for wireless applications and therefore is examined here for different kinds of software optimisation. For several optimisation steps power dissipation has been measured within continuous operation of the program code and is sketched in Fig. 3 with the achieved symbol rate. Furthermore, the resulting energy per output sample is depicted over the symbol rate.

The reference software taken from (Karn, 1996) is denoted by the symbol A. In the first step, unused blocks in the DSP for example video I/O units are powered down, resulting in B. Compiler options were adjusted appropriately within the next optimisation (C). Custom operations were applied in order to get linear program code (D). All of these optimisation steps were provided for the PMU, since it requires about 79% of the total execution time. The variable type of the path metrics were changed from an array type to a scalar one resulting in the most energy efficient implementation (E). Though the power dissipation is increasing, from B to E, the energy per sample is decreased as the symbol rate features at the same time a more significant increase.

3.2 Optimisation for FPGA implementations

As an exemplary SRAM-based reconfigurable FPGA an Altera Apex 20KE device (EP20K200EQC240-2) was applied (Altera 2002). The available logic resources allow an implementation on a single device. This FPGA offers arrays of logic cells and additional embedded system blocks as memories. The logic cells include a 4-bit look-up table, a subsequent register and additional logic e.g. for the implementation of fast carry ripple adders.

The PMU and the SMU were examined and optimised for the FPGA implementation because of their significance for the overall area and speed of the design (Black, 1993). For the SMU a trace back method was chosen, because the register exchange method requires too many registers and routing resources for this specification. In order to use a simple clocking scheme, the two pointer trace back method was chosen, since hybrid architectures with a trace forward method again result in inefficient usage of FPGA resources.

In order to find an optimal solution different implementa-

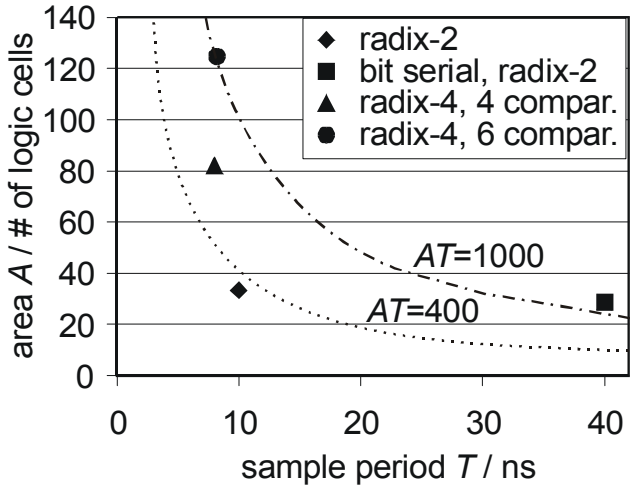


Fig. 5. Comparison of different PEs.

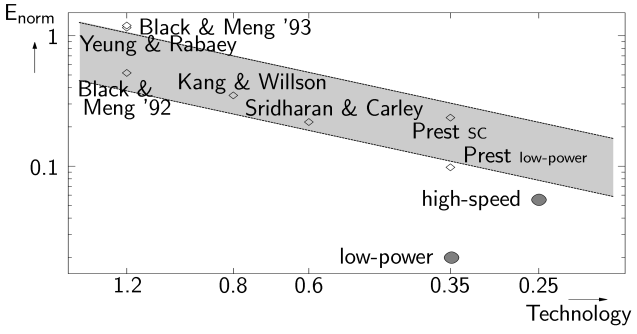


Fig. 6. Normalised energy conversion (E_{norm}) per symbol of Viterbi decoder implementations (Gemmeke et al., 2001); (E_{norm} is defined as power dissipation normalised to the Viterbi decoder specifications of DVB-S and throughput rate).

tions of the add-compare-select processing element (PE) of the PMU were examined regarding the area (assumed to be proportional to the number of required logic cells) and the maximal symbol rate $1/T$ (Fig. 4). The radix-2 butterfly PE has been presented several times in literature (Kivioja et al., 1999; Pandita et al., 1999). With an LSB first comparator, it utilises the mentioned fast carry chain efficiently (Fig. 4a). It requires 33 logic cells with a symbol rate of 99 Megabit/s. A bit serial carry save MSB first implementation was further examined providing the advantage, that the path metric does not need to be stored. For this implementation a carry save MSB first comparator is required which is rather complex.

In a radix-4 butterfly PE two succeeding stages of a radix-2 add-compare-select operation are performed in a single PE. For a hierarchical comparison three comparators are required (Fig. 4c). One would expect that a doubled throughput rate should be achievable with a radix-4 approach, but due to the underlying hardware structure, the resulting sample rate is only almost as fast as in the case of a radix-2, but much more area is required. For a parallel, pairwise comparison six comparators are necessary (Fig. 4d). The critical path is shown in both PEs with the corresponding timing values. However, due to the inefficient placement of the logic cells in the select logic, the critical path is even worse although the number of logic cells was increased for the intention of speedup. For a dedicated macro, this implementation is often chosen, because of better implementation results.

The AT complexities of different PEs are shown in the diagram in Fig. 5. The radix-2 element is by a factor of 2.5 more efficient in terms of AT complexity than a bit serial radix-2 PE. Overall, here a radix-2 PE is the most cost efficient implementation. This is due to the use of fast carry chains in the logic cells, which can be applied to efficiently implement the add-compare operation (implemented as a subtraction).

3.3 Physically optimised implementation

A dedicated hardware implementation offers higher throughput rates and less power dissipation compared to any programmable/reconfigurable solution. A physically oriented design style allows to fully exploit the throughput potential of a technology, and to reach the lowest possible energy conversion per operation.

To reduce the design effort of a hardware implementation the regularity of common digital signal processing datapaths can be exploited. Common arithmetic algorithms inherently contain a high degree of locality. Preserving this down to the layout ensures high throughput at low silicon area and even more important low power dissipation. The use of a datapath generator (Weiss et al., 2001) automates the macro generation.

Applying quantitative optimisation on all levels of design hierarchy two Viterbi decoders were designed (Gemmeke et al., 2002): one optimised for high speed operation, the other for low power dissipation.

A comparison of these two designs to the trend of other published leading edge implementations is shown in Fig. 6. Apparently, the designs fall into a band of decreasing power dissipation. Its slope indicates an exponential trend according to the minimum feature size in VLSI technology. The physically optimised low-power implementation disrupts the common trend by approximately one decade. Whereas, the high performance design traded some power dissipation for high-speed operation.

4 Final comparison

The preceding implementation results are compared by means of the *ATE* cost function. As a consequence Fig. 7a can be derived. Besides the three implementation alternatives discussed before also the cost value for a dedicated standard cell implementation is depicted. The cost values are normalised to that of the physically optimised implementation. It is shown that the normalised cost of a DSP and a physically optimised implementation differ by about seven orders of magnitude. This is compared to the costs of further basic operations. In Fig. 7b a comparison for a variety of digital signal processing operations is depicted (Blume et al., 2002).

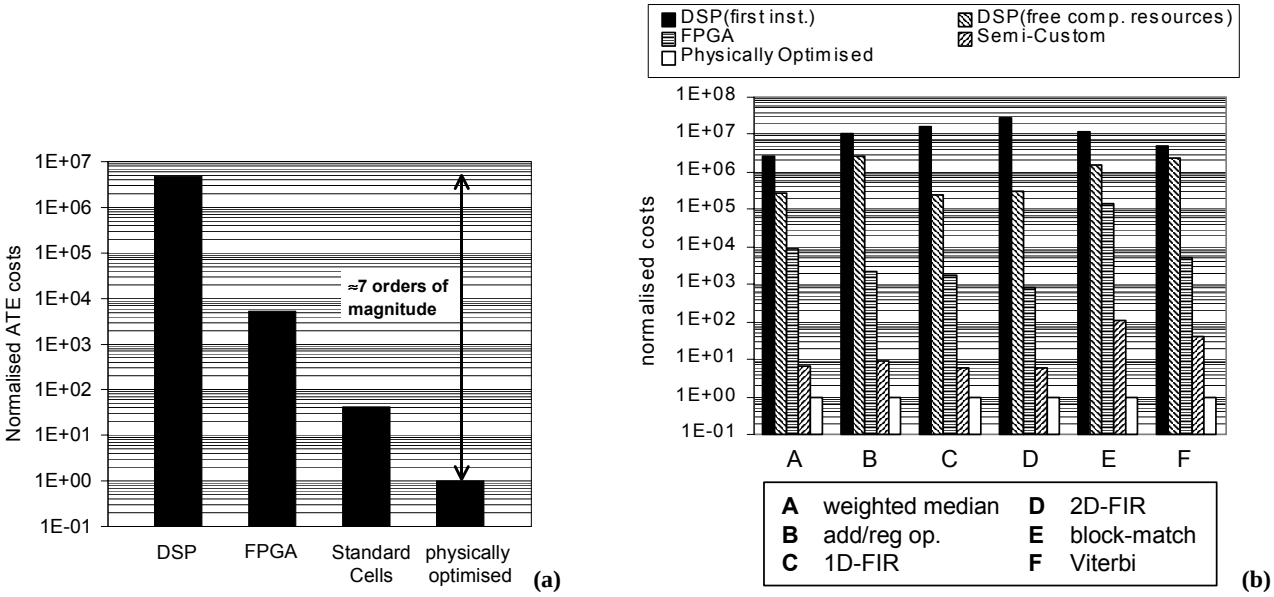


Fig. 7. Comparison by means of normalised costs, **(a)** Comparison of a Viterbi decoder, **(b)** Comparison for different operations.

It is distinguished between differential and absolute power consumption of an operation on a given architecture block. Differential power consumption applies to the availability of free computational resources. Therefore, only the operation-dependent power consumption is considered. If the computational resources are exhausted, a new device needs to be instantiated. This is referred to as absolute power consumption, where additionally operation-independent power consumption (overhead e.g. for the clock system) has to be considered.

For the case of a DSP with free computational resources the cost ratio between a physically optimised and a DSP based implementation spans from at least four to six orders of magnitude. FPGA based implementation costs mostly lie between the physically optimised and the DSP implementations. Considering the absolute power consumption, the cost ratio between DSP and physically optimised implementation increases up to two additional orders of magnitude. All the investigations discussed here were based on discrete devices. Future SoCs will embed several architecture blocks of the types described before on one chip.

5 Conclusion

As the cost differences between dedicated and programmable implementations of system blocks are rather huge and flexibility is required for future SoCs, partitioning of a system to a target architecture is most important. It demands for an analysis of implementation specific parameters for basic operations. This is required in order to choose the optimum implementation for basic operations with demanding specifications. Cost modelling of basic operations on different architecture blocks like DSPs, FPGA like structures, semi-custom and physically optimised macros is required for this partitioning.

Using the Viterbi decoder as an exemplary component a quantitative cost function based analysis for heterogeneous SoCs has been shown. Future work has to be directed to refined partitioning strategies utilising approved models for key elements allowing an early assessment of possible implementation alternatives.

References

Ahmad Khan, S., Saqib, M., and Ahmed, S.: Parallel Viterbi algorithm for a VLIW DSP, Proc. ASSP, pp. 3390–3393, 2000.

Altera: APEX 20K Programmable Logic Device Family Data Sheet, Version 4.3, Feb. 2002.

Black, P.: Algorithms and Architectures for High Speed Viterbi Decoding, PhD dissert., Stanford, March 1993.

Blume, H., Huebert, H., Feldkaemper, H., and Noll, T. G.: Model based exploration of the design space for heterogeneous Systems on Chip, ASAP, pp. 29–40, 2002.

De Hon, A.: The Density Advantage of Configurable Computing, IEEE Computer, pp. 41–49, 4/2000.

Gemmeke, T., Gansen, M., and Noll, T. G.: Scalable, Power and Area Efficient High Throughput Viterbi Decoder Implementations, IEEE, Journal of Solid-State Circuits, pp. 941–948, July 2002.

Hausner, J.: Integrated Circuits for Next Generation Wireless Systems, Proc. ESSCIRC, pp. 26–29, 2001.

Karn, P.: http://people.qualcomm.com/karn/code/index.html, 1996.

Kivioja, M., Isoaho, J., and Vanska, L.: Design and Implementation of a Viterbi Decoder with FPGAs, Journal of VLSI Signal Processing, pp. 5–14, Kluwer, Netherlands, May 1999.

Pandita, B. and Roy, S.: Design and Implementation of a Viterbi Decoder using FPGAs, Proc. VLSI Design, pp.611–614, 1999.

Weiss, O., Gansen, M., and Noll, T. G.: A flexible Datapath Generator for Physical Oriented Design, ESSCIRC, pp. 408–411, 2001.