Decoder Implementation for Cloud Based Architectures

Dirk Wübben[1], Henning Paul[1], Philippe Balleydier[2], Valentin Savin[2], Peter Rost[3]
[1]University of Bremen, [2]CEA-LETI, [3]NEC Laboratories Europe

***Abstract*—Implementation of radio access network functions on centralized cloud platforms is envisioned for on-demand provisioning of computing resources in mobile networks. In principle, this allows for more advanced algorithms and offers the ability to balance the computational load, but also imposes challenges on the design of the applied algorithms. In this paper, we analyse the implementation of decoding algorithms on general purpose hardware, as decoding draws the main computational burden of signal processing in the uplink.**

I. Introduction

For the evolution towards 5G mobile networks several technologies are currently investigated. The dense deployment of small cells in combination with (partly) centralized processing is one promising candidate. In centralized radio access networks (C-RAN) part of the digital baseband processing is shifted from the radio access points (RAPs) to a central processing unit. Although this allows for efficient resource usage and advanced multi-cell algorithms, dedicated and special hardware like DSPs is still required [1]. As a long term goal, the deployment of cloud computing platforms running on general purpose hardware (GP-HW) leading to Cloud-RAN systems would be more beneficial [2].

In cloud platforms the required computational resources are provided on-demand by introducing the concept of virtualization. Thus, Cloud-RAN will allow deploying algorithms that scale with the current needs and leverage massive parallelism. On the other hand, cloud-implementations also impose challenges for implementing baseband processing on GP-HW due to the tight constraints caused by the protocol stack of mobile communication standards. For example, in 3GPP LTE the hybrid automatic repeat request (HARQ) process requires to execute all physical processing of codewords within 3 ms [3]. Due its complexity, this poses a significant challenge especially for FEC decoding which is usually implemented on specialized hardware such as ASICs or FPGAs. In order to meet the stringent requirements on data rates, cloud-based FEC decoders will need to fully exploit the available parallelism of a cloud-computing platform. In this context, Low Density Parity Check (LDPC) and Turbo Codes (TC) are two promising candidates because both allow for accommodating various degrees of parallelization.

In this paper, we discuss the challenges of cloud-implementation of FEC decoders and present current results for implementing message passing algorithms for LDPC codes and for implementing LTE Turbo Decoder.

II. Implementation of Decoding Algorithms

A. Parallel Decoding on Multiprocessor Platforms

Two main approaches can be used to exploit parallelism in multiprocessor/multicore platforms as visualized in Fig. 1. First, fully or partially parallel decoders can be implemented through the use of concurrent threads, with every processor or core executing a separate thread. The efficiency of such a parallel implementation depends also on the degree of parallelism allowed by the implemented algorithm.

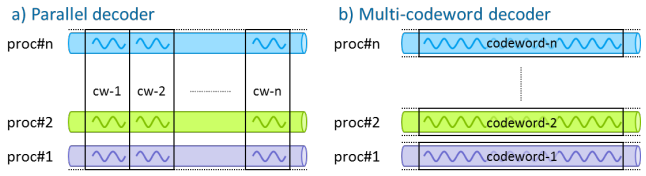

Fig. 1. Two approaches to exploit parallelism in multiprocessor platforms

The second approach consists of using multi-codeword decoders, with each processor running a separate image that decodes a different codeword. Both approaches allow increasing the throughput, although in two different ways: the first by reducing the latency per decoded codeword, the second by increasing the number of codewords decoded within the same latency period. In addition, both approaches can be combined, depending on the implemented algorithm and the specific characteristic of the computing platform.

B. LDPC decoder

This section presents experimental results on the achievable throughput of software-based LDPC decoders. Due to computational complexity and convergence speed reasons, the Min-Sum decoder with layered scheduling has been chosen for our investigation. In order to assess the achievable throughput on GP-HW, a C++ implementation of a multi-codeword decoder has been carried out. Multithreading was implemented by using Message Passing Interface (MPI) and Open Multi-Processing (OpenMP) directives. The maximum number of decoding iterations has been set to 20. The decoder stops when whether a codeword have been found (syndrome equal zero) or the maximum number of iterations is reached. In particular, the average number of decoding iterations decreases with increasing SNR, or equivalently, with improving error correction performance. The performance of the Min-Sum decoder for WiMAX LDPC codes with rate 1/2 and 5/6 is presented in Table I, in terms of required SNR for a target frame error rate (FER) of 10^{-2} and 10^{-4} (QPSK modulated A). The corresponding average number of decoding iterations is also shown in the table.

Table I: Required SNR and average number of decoding iterations for target FER of 10^{-2} and 10^{-4}

	Target FER = 10^{-2}		Target FER = 10^{-4}	
	Required SNR	Ave Iter Nb	Required SNR	Ave Iter Nb
LDPC@rate 1/2	2.1 dB	6.9	2.5 dB	5
LDPC@rate 5/6	5.9 dB	4	6.3 dB	2.8

The Min-Sum multi-codeword decoder has been run on two Intel Xeon x5650 @2.67GHz processors, each one composed of 6 physical cores. Each physical core is further com-

posed of 2 logical cores, which leads to a total of 24 logical cores divided in two processors.

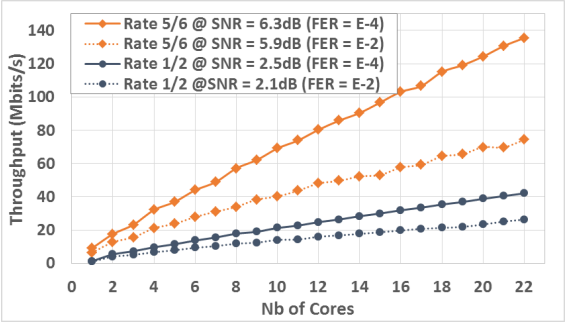

Fig. 2. Average Throughput as function of the number of cores,

Threads decoding different codewords are synchronized, in the sense that they all start in same time and wait for the slowest one to complete decoding (since codewords decoded by different threads can take a different number of decoding iterations). The average throughput, as function of the number of cores is presented in Fig. 2 for SNR values reported in Table I. The throughput is expressed as number of useful (information) bits per second. The difference between rate-1/2 and rate-5/6 throughput is explained by (i) the faster convergence of the rate-5/6 decoder, which results in a reduced average number of decoding iterations, and (ii) the increased number of information bits for each decoded codeword. It can also be observed that the achieved throughput significantly increases when the FER improves from 10^{-4} to 10^{-2}, corresponding to an SNR increase of about 0.4 dB. For an SNR of 6.3 dB (FER = 10^{-4}), the rate-5/6 decoder achieves an average throughput of 140 Mbits/s, by decoding 22 codewords in parallel (22 logical cores are used).

C. *LTE Turbo Decoder*

In order to assess the computational diversity offered by multi-core implementations we present experimental results for spectral efficiency and required computational complexity of an 3GPP LTE uplink decoder. The turbo decoder has been implemented on a default VMWare ESXi server with Ubuntu Linux host operating system, GNU C++ compiler, and codeword multi-threading in order to account for the virtualization overhead. We measured the required CPU time to decode one codeword and determined the average CPU time within the 90% confidence interval.

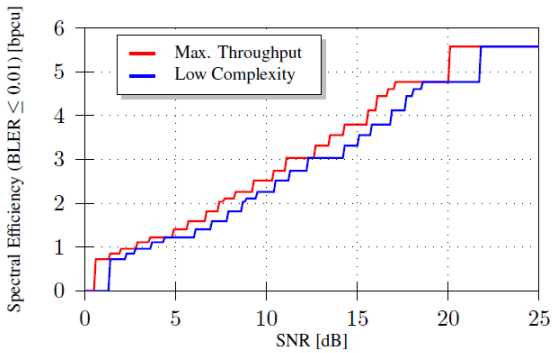

Fig. 3. Spectral Efficiency achieved for turbo decoding on GP-HW

Fig. 3 shows the achievable spectral efficiency for a given SNR (AWGN, no fading) and target block error rate $BLER_{tar} = 0.01$ for two cases: maximum throughput and low complexity by limiting the number of iterations to 2. Obviously, reducing the complexity results in a performance penalty of 1-2 dB.

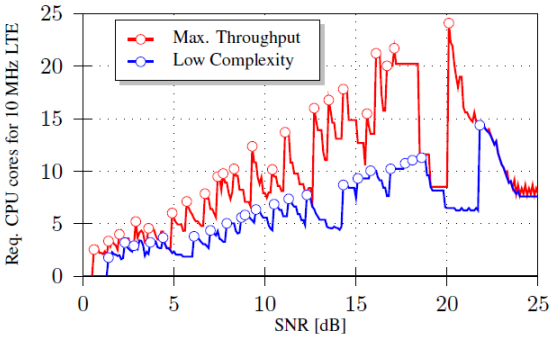

Fig. 4. Required number of CPU cores for turbo decoding on general purpose hardware

In Fig. 4, we show the required computational resources for a 10MHz 3GPP LTE system. The required complexity strongly depends upon the SNR. Firstly, it increases linearly with the number of information bits which implies a logarithmic increase of complexity in SNR. Secondly, the complexity increases with the number of iterations that are necessary to decode a codeword. Markers show the SNR where the next higher MCS has been chosen. We can notice at each of these markers an increase of the computational demand which is then quickly decreasing in SNR. Apparently, this strongly varying computational demand allows for exploiting multi-user computational diversity at the centralized processor. For instance, computational load balancing across multiple users to reduce the ratio of peak to average computational efforts can be performed or the computational demand can actively been shaped by selecting MCS to satisfy a computational constraint.

III. Conclusion and Outlook

Cloud-RAN offers massive parallel computing and allows for computational load balancing. We present throughput and complexity results for decoder implementations on commodity hardware and point out design criteria that allow for flexible load balancing.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Program FP7/2007-2013 under grant agreement n° 317941 – project iJOIN. The European Union and its agencies are not liable or otherwise responsible for the contents of this document; its content reflects the view of its authors only. We gratefully recognise the great contributions of many colleagues from iJOIN, who in fruitful cooperation, contributed with valuable insight, surveys and vision.

References

[1] K. Chen, C. Cui, Y. Huang, and B. Huang, "C-RAN: A Green RAN Framework," in Green Communications: Theoretical Fundamentals, Algorithms and Applications, J. Wu, S. Rangan, and H. Zhang, Eds. CRC Press, 2013.

[2] P. Rost, C.J. Bernados, A. De Domenico, M. Di Girolamo, M. Lalam, A. Maeder, D. Sabella, and D. Wübben, "Cloud technologies for flexible 5G radio access networks", *IEEE Communications Magazine*, vol. 52, no. 5, May 2014.

[3] D. Wübben, P. Rost, J. Bartelt, M. Lalam, V. Savin, M. Gorgoglione, A. Dekorsy, and G. Fettweis, "Benefits and Impact of Cloud Computing on 5G Signal Processing", *IEEE Signal Processing Magazine,* submitted.