Performance evaluation of offloading LDPC decoding to an FPGA in 5G baseband processing

Florian Kaltenberger
EURECOM
Sophia Antipolis, France
florian.kaltenberger@eurecom.fr

Hongzhi Wang and Sakthivel Velumani
OpenAirInterface Software Alliance
Sophia Antipolis, France
hongzhi.wang@openairinterface.org

***Abstract*—Offloading of computationally expensive physical layer processing such as the forward error correction, is one of the key enablers for a fully virtualized open radio access network (RAN). In this paper we show such an offloading architecture and will demonstrate it using the OpenAirInterface 5G New Radio open source software and the Xilinx T1 telco accelerator card. We will show the feasibility and the potential savings of such an architecture.**

I. Introduction

5G mobile radio networks are currently being rolled out worldwide and are bringing us higher bandwidth, lower latency and a lot of other new features. However, this new technology is not cheap and, operators are scrambling to re-architecture their networks to keep up with the increasing demand and flat revenue. One of the key methods to reduce costs and improve service is by moving from a traditional radio access network deployment to what is today called Open RAN deployments, also called cloud RAN or virtualized RAN.

In an open RAN design, interfaces such as fronthaul and midhaul but also management and operation are open, and functionality is implemented in software on general purpose computing hardware and can thus be run in the cloud. The open RAN interfaces and architecture are defined by the O-RAN alliance and depicted in Figure 1. The most important elements are the radio unit (O-RU), the distributed unit (O-DU), and the centralized unit (O-CU), which make up the main elements of a traditional base station. The O-RU and the O-DU are connected over the fronthaul (FH) interface, which is also called the O-RAN 7.2 interface while O-DU and O-CU are connected over the F1 interface O-RU and O-DU also have an E2 interface which allows the near-real time RAN intelligent controller (RIC) to get feedback from the RAN (e.g., channel measurements, load, etc.) and control certain elements in the RAN (e.g., scheduling decisions, handover decisions etc.). This design allows the RIC to run artificial intelligence algorithms that can control certain elements in the RAN. O-RAN also specified an O1 interface which allows for management and orchestration of the RAN entities.

OpenAirInterface[1] (OAI) is an open-source initiative that today provides a 3GPP compliant reference implementations of key elements of 4G, 5G Radio Access Network (RAN) and core network that run on general purpose computing platforms (x86) together with Off-The-Shelf (COTS) Software Defined Radio (SDR) cards like the ETTUS Universal Software Radio Peripheral. It allows users to set up a compliant 4G/5G network and inter-operate with commercial equipment. OpenAirInterface is today seen by many key players in the industry as the most complete open-source software solution to implement such open RAN networks. It is the ideal starting point for any company who wants to provide solutions and products in this new landscape.

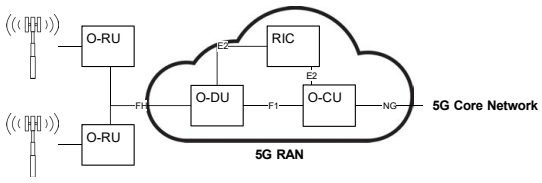

Fig. 1. O-RAN architecture and interfaces.

II. Physical Layer Offloading

One of the key challenges in virtualized, open-RAN networks is the computational complexity of the physical layer processing. And within the physical layer, the most computationally expensive task is the LDPC forward error correction encoding and decoding. As an example, lets consider the physical layer processing of the physical uplink shared channel (PUSCH) with the following configuration: MCS 28 (64 QAM), Length: 12 OFDM symbols, Bandwidth: 106 PRB. In ideal channel conditions (AWGN with SNR of 30dB) on a single processor the channel decoding alone takes 965 μs, which is 91% of the total RX processing time[2]. This result has been achieved with the `nr_ulsim` function of OAI, which already includes highly optimized code for LDPC decoding. This time can be reduced by parallelizing the decoding of multiple segments over multiple cores (in this example there are 9 segments), but the overall computational complexity remains the same.

Therefore, a key element of open RAN networks is the possibility to offload part of the computational expensive

[1]http://www.openairinterface.org

[2]This result was achieved on an Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz.

ISBN 978-3-8007-5686-5

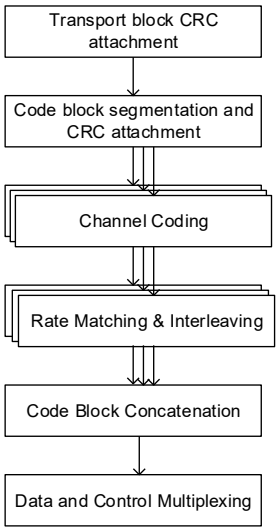

Fig. 2. ULSCH tranport chain.

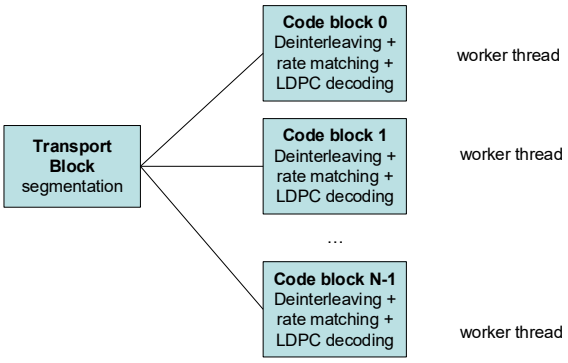

Fig. 3. Current parallel architecture of LDPC decoding.

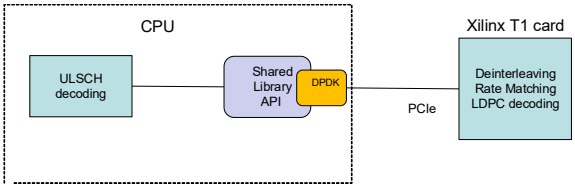

Fig. 4. New architecture using offloading to T1 card.

processing to dedicated hardware. Several hardware manufacturers are currently bringing such hardware to the market. One example is Xilinx with its T1 telco accelerator card[3]. This card is a PCI-e card and contains one ZU19EG Multi-processing Platform System on Chip (MPSoC) and ZU21DR Radio Frequency System on Chip (RFSoC). The card has been designed to offload the forward error correction code processing (LDPC and Turbo encoding and decoding) and O-RAN 7.2 fronthaul processing (O-RAN and eCPRI framing and de-framing, as well as PTP synchronization). The card allows for 50Gbps Fronthaul Bandwidth, 17.7Gbps Encoder Throughput, and 7.8Gbps Decoder Throughput.

In the current contribution we are showing the first integration of OpenAirInterface with the Xilinx T1 card by offloading the LDPC channel decoding of the physical uplink shared channel (PUSCH). We start by describing briefly the 5G NR PUSCH transmission procedure in section III. In section IV we describe in more detail the software defined PUSCH receiver as well as the offload architecture. Section V presents the simulation results of the performance as well as the timing. Section VI concludes the paper.

III. The physcial uplink shared channel

The PUSCH transport process is described in [1], section 6.2 and shown in Figure 2. First a cyclic redundancy check (CRC) is appended to the transport block (TB) coming from the MAC. Next, the TB is segmented into code block and another CRC is added to each code block. This procedure allows for (i) re-transmission of only the code blocks that were in error and (ii) early termination of the LDPC decoder. Next, LDPC channel coding is applied to each of the code blocks and rate matching and interleaving is applied. Then the code blocks are concatenated again and if control information is transmitted on the PUSCH, it is multiplexed with it. The resulting bits are then passed to the scrambling, modulation, layer mapping, and precoding. The resource elements of the PUSCH and the corresponding demodulation reference signals are then mapped to the OFDM resource grid and sent to the OFDM modulation. This part is not shown as it is not the focus of the paper.

[3]https://www.xilinx.com/applications/wired-wireless/telco.html

IV. Implementation of the receiver

The receiver first does OFDM de-modulation, channel estimation based on the DMRS, channel compensation, demodulation, and de-scrambling. The resulting log-likelihood ratios (LLRs) are then separated into code blocks and passed to the channel decoding function.

The soft LDPC decoder used in OAI is described in [2]. In our implementation we set the number of iterations to 10. Further, OAI uses a thread-pool to decode multiple segments in parallel on multiple cores. Each thread/core handles de-interleaving, de-rate-matching and LDPC decoding. The process is also shown in Figure 3.

In our new offloading architecture, all these elements are offloaded to Xilinx T1 card. The T1 card supports the bbdev API from DPDK to offload the channel encoding and decoding. A shared library is created to implement the interface for DPDK. The shared-library translates the OAI descriptors for ULSCH decoding into the required parameters for LDPC decoding used by the DPDK API (see Figure 4).

ISBN 978-3-8007-5686-5

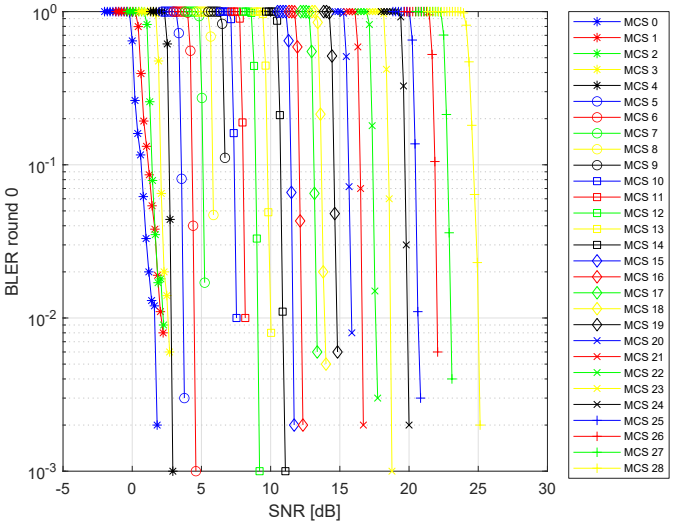

Fig. 5. Soft LDPC decoder block error rate (BLER) of the first round for all MCS in AWGN channel.

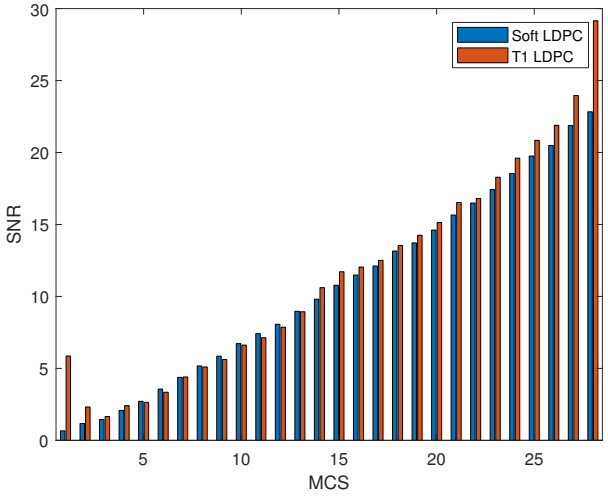

Fig. 6. Soft LDPC and LDPC offload comparison of the SNR required to achieve a 10% BLER in the first round in AWGN channel.

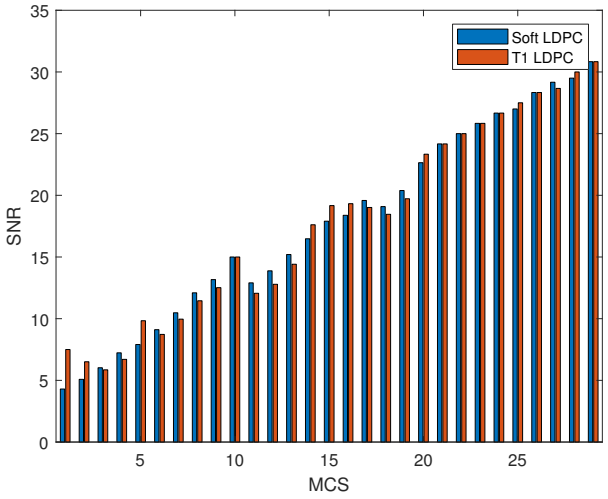

Fig. 7. Soft LDPC and LDPC offload comparison of the SNR required to achieve a 10% BLER in the first round in TDL-C channel with 30ns delay spread.

V. Simulation Results

The simulations have been carried out with the nr_ulsim tool of OAI, which simulates the ULSCH transmitter, the channel, and the ULSCH receiver. We use the following configuration of the ULSCH: 30kHz subcarrier spacing, 273 PRB (corresponding to a channel bandwidth of 100MHz), 12 OFDM symbols, DMRS config type 1, 1 DMRS symbol. The simulations were carried out on an Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz with 36 cores (hyperthreading disabled).

We start by showing in Figure 5 the soft LDPC decoder block error rate (BLER) of the first round for all MCS in AWGN channel. To compare the performance of the soft LDPC decoder with the hardened LDPC decoder on the T1 card we compute the SNR required to achieve a BLER of 10% in the first round for each MCS. We show these results for the AWGN channel in Figure 6 and for the TDL-C channel with 30ns delay spread in Figure 7.

It can be seen that from a performance point of view the two decoders preform more or less equally well. The soft LDPC decoder is actually a bit better sometimes, especially for the very low and high MCS values. But that might also be outliers or due to some bad configuration of the hard LDPC decoder (to be investigated further).

Next we compare in Figure 8 the average decoding time of the soft LDPC decoder with the hardened LDPC decoder on the T1 card for each MCS in AWGN channel. The results for the TDL-C channel are similar and not shown here. It can be seen that the offloaded LDPC decoder is up to double as fast, except for the high MCS where the savings are not that significant. The reason for this could be that in our implementation we are only sending only code block at a time to the T1 card and then wait for the decoded result to return before decoding the next code block.

VI. Conclusions

We have shown the feasibility to offload the LDPC decoding to the Xilinx T1 acceleration card using the OpenAirInterface software. This is a necessary step to enable a fully virtualized deployment of OAI in an open RAN environment.

We have shown that the performance of the software LDPC decoder available in OAI is comparable with the one of the hardened LDPC decoder on the Xilinx T1 card. From the perspective of processing time the offload can be more than double as fast. However, this is not as fast as we would have expected and this is probably due to the fact that we only send individual segments sequentially. The T1 card actually also support a mode where one can send the whole transport block at once instead of sending the individual code blocks. However the current version of the driver and bitstream does not support this yet.

ISBN 978-3-8007-5686-5

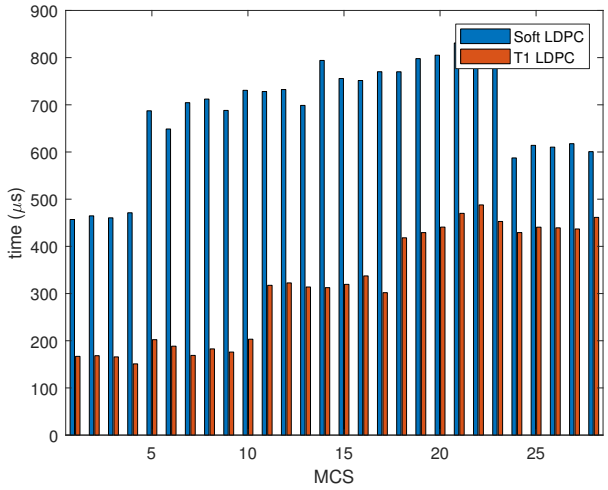

Fig. 8. Soft LDPC and LDPC offload comparison of the average decoding time for each MCS in AWGN channel.

As a future work we need to integrate the support for hybrid automatic repeat request (HARQ). Once this is done, we shall be able to test the offload also in the real-time softmodem and do end-to-end testing with a real UE. Last but not least we are also working on a mechanism to share the T1 card between multiple processes so it can be shared in a virtualized environment.

ACKNOWLEDGEMENTS

The authors would like to thank Xilinx for their financial support and for providing us with a T1 card and VVDN for providing us with the bitstream and the necessary drivers to operate the card. Further thanks goes to all the developers at the OpenAirInterface project!

REFERENCES

[1] 3GPP, "Multiplexing and channel coding," 3GPP, Technical Specification 38.212 15.9, Jun. 2021.

[2] S. Wagner, "Nr ldpc decoder," TCL 5G Labs, techreport, Oct. 2019. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/openair1/PHY/CODING/nrLDPC_decoder/doc/nrLDPC/nrLDPC.pdf

[3] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, "OpenAirInterface: Democratizing innovation in the 5G era," *Computer Networks*, vol. 176, no. 107284, Jul. 2020. [Online]. Available: http://www.eurecom.fr/publication/6243

[4] F. Kaltenberger, R. Knopp, C. Roux, P. Matzakos, F. Mani, and S. Velumani, "OpenAirInterface 5G NSA system with COTS phone," in *WINTECH 2020, 14th ACM Workshop on Wireless Network Testbeds, Experimental evaluation and Characterization*, London, UK, Sep. 2020. [Online]. Available: http://www.eurecom.fr/publication/6314

[5] T.-H. Wang and R. Knopp, "OpenAirInterface: A pipeline structure for 5G," in *DSP 2018, IEEE International Conference on Digital Signal Processing, 19-21 November 2018, Shanghai, China*, Shanghai, CHINA, 11 2018. [Online]. Available: http://www.eurecom.fr/publication/5733

