FACULTY OF ELECTRICAL ENGINEERING
CTU IN PRAGUE

Diploma Thesis

Hardware Acceleration of Channel Decoding in Software Defined 5G Base Station

Hardwarová akcelerace kanálového dekódování v softwarově definované základnové stanici 5G

Master Candidate

Jaroslava Fiedlerova

Supervisor

doc. Ing. Zdeněk Bečvář, Ph.D.

Co-supervisor

Prof. Florian Kaltenberger

EURECOM

Academic Year
2022/2023

Declaration
Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravé vysokoškolských závěrečných prací.

I declare I have accomplished my diploma thesis by myself and I have named all the sources used in accordance with the Guidelines on ethical preparation of university theses.

In Prague, ...
Signature:

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Fiedlerová Jaroslava** Personal ID number: **456898**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Radioelectronics**

Study program: **Electronics and Communications**

Specialisation: **Technology of the Internet of Things**

II. Master’s thesis details

Master’s thesis title in English:

Hardware Acceleration of Channel Decoding in Software Defined 5G Base Station

Master’s thesis title in Czech:

Hardwarová akcelerace kanálového dekódování v softwarově definované základnové stanici 5G

Guidelines:

Study a concept of 5G new radio physical layer processing, especially channel (de)coding, and the possibility of its hardware acceleration. Review the current solution for offloading of the channel decoding using Xilinx T1 Telco Accelerator Card in the software defined 5G base station in OpenAirInterface. Propose an improvement of the implementation of the channel decoding offloading and consider implementation of the offloading of the channel decoding using Xilinx T2 Telco Accelerator Card. Test performance of the proposed solutions with OpenAirInterface simulation tool for 5G uplink. Compare the performance (e.g., in terms of average decoding time, SNR required to achieve certain error rate) using Xilinx T1 Telco Accelerator Card and/or Xilinx T2 Telco Accelerator Card.

Bibliography / sources:

[1] E. Dahlman, S. Parkvall, and J. Skold, “5G NR,” Sept. 2020. ISBN: 9780128223215
[2] M. Enescu, “5G New Radio: A Beam-based Air Interface,” 2020. ISBN: 978-1-119-58236-6
[3] F. Kaltenberger, H. Wang, S. Velumani, “Performance evaluation of offloading LDPC decoding to an FPGA in 5G baseband processing,” International ITG Workshop on Smart Antennas (WSA 2021), Nov. 2021

Name and workplace of master’s thesis supervisor:

doc. Ing. Zdeněk Bečvář, Ph.D. Department of Telecommunications Engineering FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **16.09.2022** Deadline for master's thesis submission: _____

Assignment valid until: **19.02.2024**

doc. Ing. Zdeněk Bečvář, Ph.D. Supervisor’s signature	doc. Ing. Stanislav Vítek, Ph.D. Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Abstract

The topic of this diploma thesis is the channel decoding offload using hardware accelerators, namely Xilinx T1 and T2 Telco Accelerator Card in the OpenAirInterface 5G New Radio platform. OpenAirInterface is a project focused on the implementation of the 3GPP compliant 5G and 4G protocol stack, that enables deployment of the radio access network and core network on general purpose computing platforms. Xilinx accelerator cards are hardware components, dedicated to performing computationally intensive Low Density Parity Check decoding with high speed. Software LDPC decoder and offload of the LDPC decoding to the Xilinx T1 Telco Accelerator card is currently implemented in the OAI. The solution with the Xilinx T2 Telco Accelerator card is newly integrated into the OAI protocol stack. Performance evaluation of the newly integrated solution in comparison with the existing solutions is done. Experiments are conducted using OAI physical layer simulator and further with an Over-the-air (OTA) testbench for throughput measurements.

Abstract

Diplomová práce je zeměřena na využití akcelerace kanálového dekódování v softwarově definované 5G základnové stanici. Praktická část práce spočívá v implementaci a testování hardwarového akcelerátou, T2 Xilinx Telco Accelerator Card, pro dekódování kanálu v OpenAirInterface základnové stanici. OpenAirInterface je organizace, která se zabývá implementací protokolových vrstev 5G a 4G sítí v souladu se standardy 3GPP. Akcelerační karty Xilinx jsou hardwarové komponenty určené ke zpracování výpočetně náročného dekódování kanálu. V rámci softwaru OpenAirInterface je v současné době implementována možnost offloadu dekování kanálu na T1 akcelerační kartu. Nově implementované řešení s použitím T2 akcelerační karty je testováno pomocí simulátoru uplink kanálu a Over-the-Air testu.

Contents

List of Figures

List of Tables

List of Acronyms

RAN Radio Access Network

3GPP Third Generation Partnership Project

ARQ Automatic Repeat Request

AWGN Additive White Gaussian Noise

BLER Block Error Rate

BS Base Station

BW Bandwidth

CN Core Network

CB Code Block

COTS Commercial Off-The-Shelf

CRC Cyclic Redundancy Check

DL Downlink

DPDK Data Plane Development Kit

EAL Environment Abstraction Layer

EPC Evolved Packet Core

FEC Forward Error Correction

FR Frequency Range

HARQ Hybrid Automatic Repeat Request

LDPC Low Density Parity Check

LLR Log-Likelihood Ratio

MAC Medium Access Control

MCS Modulation Coding Scheme

NSA Non-Standalone

OAI OpenAirInterface

OSA OpenAirInterface Software Alliance

OTA Over-the-air

PDCP Packet-Data Convergence Protocol

PDU Protocol Data Unit

PHY Physical Layer

PMD Poll Mode Driver

POSIX Portable Operating System Interface

QC-LDPC Quasi Cyclic LDPC

QoS Quality Of Service

RAN Radio Access Network

RLC Radio-Link Control

ROHC Robust Header Compression mechanism

SA Standalone

SCS Sub Carrier Spacing

SDAP Service Data Adaptation Protocol

SDR Software Defined Radio

SDU Service Data Unit

TLB Translation Lookaside Buffer

TB Transport Block

UE User Equipment

UL Uplink

USRP Universal Software Radio Peripheral

1 Introduction

With the evolution of mobile networks, 5G New Radio is the latest technology offered to end users. This technology offers backward compatibility with existing 4G networks. Compared to the 4G network, 5G NR two main frequency ranges are defined in 5G NR: FR1 for sub 6GHz and FR2 for a frequency range above 24.5 GHz. In the millimeter wave spectrum, high capacity and data rate of the network is possible. The scalable numerology of the network provides higher network flexibility. Another advantage over the previous generation brings the design of the network, which enables lower interference and power consumption by reduction of the always-on transmissions. There are three main target applications of this technology: enhanced mobile broadband (eMBB0, ultra-reliable and low latency communications (URLLC), and massive machine-type communication. Compared to the previous generation of mobile networks, these application requires significant improvements in terms of throughput, latency, reliability, and scalability of the entire system.

OpenAirInterface (OAI) is an open-source project focused on the implementation of the Third Generation Partnership Project (3GPP) compliant 4G and 5G solutions on the general purpose computing platforms and Commercial Off-The-Shelf (COTS) software defined radios. This enables the user to deploy a compliant 4G Long-Term-Evolution (LTE) and 5G New Radio (NR) network and run it with commercial hardware [1].

OpenAirInterface Software Alliance (OSA) is a non-profit organization, which is getting together developers all over the world to implement the key elements of 5G Radio Access Network (5G-RAN) and 5G Core Network (5G-CN), as well as the previous generation of mobile networks. Four main project groups can be defined as follows: OAI 5G RAN, 5G Core Network, MOSAIC5G, and CI/CD project group.

The aim of the 5G Core Network project group is a development of the open-source 3GPP compliant 5G Core Network (CN) implementation with a large set of features

enabling a wide spectrum of use cases. Between the basic supported features belongs connection and session management, other features are for example N2 handover or network slicing. Currently, OAI CN is also able to support multiple User Equipment (UE) connectivity with multiple PDU sessions.

The latest created project group called MOSAIC5G targets the transformation of the OAI Radio Access Network (RAN) and OAI CN into the agile and open network-service platform. Current developments are now focused on the creation of an orchestration framework with extendable APIs on top of the OAI CN and OAI RAN, enabling monitoring and programmability of the underlying network infrastructure.

The continuous Integration/Continuous Development (CI/CD) project group is oriented on maintenance and improvements of the current resources and CI/CD pipelines, implementation of the new test scenarios and development and improvements of the documentation.

The main objective of this diploma thesis is a review of the current implementation of the Low Density Parity Check (LDPC) decoding in the OAI protocol stack and investigation of its possible improvements in terms of the processing time and performance parameters. Major emphasis is here put on the usage of the hardware accelerators: Xilinx T1 Telco Accelerator Card and Xilinx T2 Telco Accelerator Card, and its benefits for the performance of the whole system. T1 card is already integrated and validated in the OAI gNB. whereas integration of the T2 card has to be done.

The rest of the thesis is organized as follows. Section 2 aims to provide a theoretical framework of the 5G NR, with focus on the physical layer procedures and particularly channel decoding. Implementation of the hardware acceleration for LDPC decoding in the OAI protocol stack is described in Section 3. Then, in Section 4, evaluation of the implemented and existing solution of the LDPC decoding is described and results of the simulations and measurements are analyzed. Obtained results and possible future improvements are then described in Section 5 of this thesis.

2

Theoretical framework

This chapter aims to provide a theoretical introduction into the topic of 5G with an emphasis on the transport-channel processing. Design principles, involved procedures and technology components behind the physical layer processing chain are described in this chapter. This theoretical knowledge is beneficial for deeper understanding of the main focus of this work - 5G channel decoding acceleration. Several sources related to this topic are available in form of books, scientific articles or also 5G specifications, which is a crucial source of information for both 5G technology understanding and also for 5G software implementation in OAI protocol stack. 5G NR [2] and 5G New Radio: Beam based interface [3] are main sources of the theoretical knowledge, specifically chapters in the books related to the physical layer aspects and processing. In the next step, implementation of the LDPC decoder (both software solution and decoding offload on the T1 card) in the OAI code is studied. This also involves a study of DPDK software framework, which provides an interface between the OAI L1 stack and the hardware accelerators. OAI simulation tools serves a vital role for further understanding of behavior and functionality of all parts of the system. Integration of the T2 card into the OAI is the next step of this project. In the final stage, testing and measurements are carried out to validate, test and to compare performance of the existing and proposed solution.

2.1 Related work

Various research work is done to discover and improve the capabilities of the LDPC codes. LDPC codes were first introduced in the year 1962 by Gallager [4], however at that time were not widely used due to their high complexity. With the development of more capable computation machines, the application of these codes suited perfectly for wired

and wireless communication systems. LDPC codes became a part of the 3GPP standards for the 5G NR. In this paper, the implementation of the LDPC and turbo codes for 5G networks is well described [5]. The research focused on implementation of the LDPC on the hardware accelerators is presented in [6], where the FPGA-based LDPC accelerator system using Xilinx LDPC decoder. Here the accelerator system with up to 1.6 Gbps decoding throughput can achieve up to 13 x times lower processing time for the channel decoding compared to the software LDPC decoder implementation, accelerator system is here integrated into OAI 5G NR platform. Offload of the LDPC processing on the Xilinx FPGA is further investigated in these articles Moving on to the implementation of the hardware accelerator with the 5G NR protocol stack, the following paper [7] provides a survey on high-performance packet processing solutions, where multiple software frameworks (DPDK, Netslice, Netmap, PF_RING) are described and compared. Multiple FPGA solutions for LDPC processing are available in the industry. LDPC FEC IP cores are provided by Xilinx [8], Intel [9], or Hitek Systems [10]. Accelercomm [11] or VVDN are also providing adding additional functionalities for the processing chain of channel decoding in 5G technology.

2.2 5G New Radio

5G networks consist of two main components: Radio Access Network (RAN) and Core Network (CN). The RAN is responsible for functions related to the radio part, such as physical signal transmission, modulation, coding, Hybrid Automatic Repeat Request (HARQ) management and scheduling. Within the overall network architecture, the concept of user plane and control plane split is used. User data are carried in the user plane, while the control signaling responsible for functions such as connection establishment, maintenance, or user mobility is carried by the control plane. Following this concept, user plane protocols and control plane protocols are defined. An overview of defined protocols is shown in Figure 2.1. Above the RAN, there is the CN, which is responsible for functions related to UE control and overall network management. Two main architectures are defined by 3GPP specifications: non-standalone and standalone, based on their interoperability with the previous generation of telecommunication technology, 4G network. In the case of non-standalone architecture, both 4G core network, called Evolved Packet Core (EPC), and 5G core network are deployed, as well as both LTE and 5G NR base stations are present in the network. Standalone architecture does not contain any components belonging to LTE technology. A brief description of functions provided by individual layers of the RAN protocol stack is given in the following subsections. Service Data Unit (SDU) and Protocol Data Unit (PDU) are commonly used terms for the description of protocol layers. An SDU is a data unit received from the upper layer using services of the current layer. A PDU is a data unit created in the

layer by attachment a header to the received SDU before forwarding to the bellow layer. Attached header carries information about processing at the layer [12].

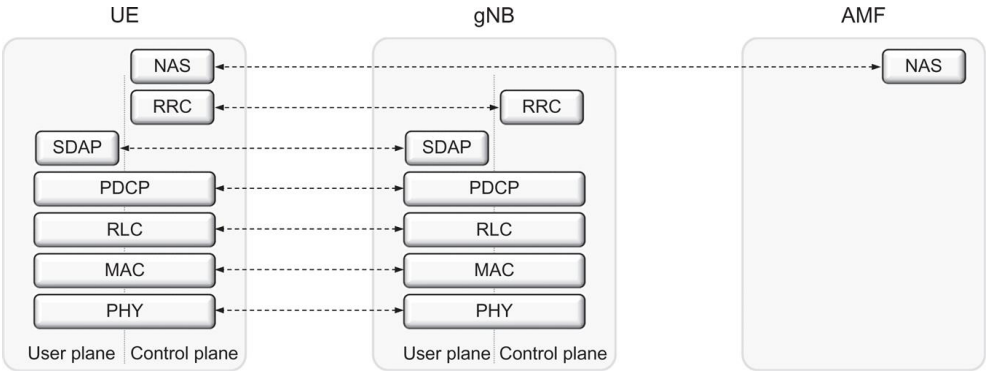

Figure 2.1: 5G NR Protocol Stack [3]

2.2.1 SDAP

SDAP layer performs a function of mapping between Quality Of Service (QoS) flows and data radio bearer, according to the QoS requirements. To enable QoS handling, UL and DL packets are marked with quality-of-service flow identifiers. This identifier brings knowledge about the mapping of IP flows to QoS flow and radio bearer [3]. Th1is layer is not present in the LTE standards, in the non-standalone 5G network, this layer is not used.

2.2.2 PDCP

Functions done by this layer are different for the user plane and control plane. In the case of the user plane, IP header compression/decompression using Robust Header Compression mechanism (ROHC) is performed. Regarding the control plane Packet-Data Convergence Protocol (PDCP) layer is responsible for integrity protection in a downward direction and integrity verification in an upward direction.

2.2.3 RLC

Radio-Link Control (RLC) layer ensures segmentation and retransmission management in L2. Based on the service requirements, RLC can act in three modes: transparent mode (TM), unacknowledged mode (UM), or acknowledged mode (AM). In the TM, the data unit flows via these layers without additional processing by the layer, no header is attached in this case. In UM, segmentation of RLC SDU in a downward direction or reassembly in an upward direction is performed. For AM mode, both segmentation and retransmission management functions are provided [12].

2.2.4 MAC

The main function provided by this layer is the multiplexing of logical channels, handling HARQ retransmission and scheduling-related operations. In a wireless communication channel, there is a certain level of signal degradation caused by multiple factors, such as propagation loss, noise and power loss. As a consequence, decoding of the incoming transport block at the receiver can be unsuccessful, which is usually determined by the CRC error detection mechanism. The transmitter is informed about the result of decoding using ACK/NACK. Retransmission of the transport block is required if an error is detected. HARQ stands for a mechanism providing robustness against transmission errors and it is ranging over Medium Access Control (MAC) and Physical (PHY) layers. PHY is in charge of the actual soft combining, while the HARQ protocol is part of the MAC layer. 4 redundancy versions are adopted.

2.2.5 PHY

PHY performs physical layer-related functions, such as modulation/demodulation, coding/decoding and mapping of the signal to the physical time-frequency resources. Its input in the case of the upward direction or output in the case of a downward direction is a transport block. The following section is focused primarily on the physical layer function of LDPC encoding and decoding.

2.3 Channel Coding in 5G

Channel coding is the process of adding redundant bits to the useful bits to enable error detection and error correction in the received message. In 5G NR following techniques are implemented: CRC mechanism for error detection, LDPC coding scheme is used for shared channels, polar codes are used for the control channels and HARQ is a retransmission mechanism operating on the MAC and PHY layer [12]. As the topic of this diploma thesis is oriented on the decoding of the uplink channel, LDPC coding scheme and processing chain of the channel are discussed in greater detail.

2.3.1 CRC

Cyclic Redundancy Check (CRC) is an error detection mechanism, to verify if received information is correct or not. CRC mechanism allows the detection of all burst errors that affects an odd number of bits, all burst errors of length less than or equal to the degree of the generator polynomial. In 5G NR, data is organized by Medium Access Control (MAC) layer into the transport blocks and sent to the Physical Layer (PHY) layer. As depicted in Figure 2.2, CRC is attached to the transport block. Based on the

specifications 3GPP TS 38.212 version 15.3.0 Release 15 [1], if the size of the transport block exceeds 3824 bits, a 24-bit CRC is attached. For transport blocks smaller than this threshold, a 16-bit CRC is attached to reduce the overhead. Further, the transport block is segmented into code blocks of equal size, if the transport block size exceeds 8448 bits for LDPC base graph 1. In the case of the LDPC base graph 2, this threshold is set to 3840 bits. An additional 24-bit CRC is attached to each segment. A detailed description of the CRC processing scheme is provided in the 3GPP specifications.

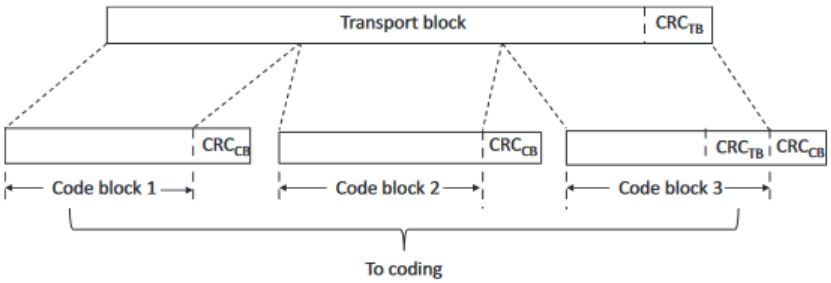

Figure 2.2: NR Transport block [12]

2.3.2 LDPC

Introduction of the redundancy into the data can serve not only for error detection but also for error correction. LDPC codes adopted in 5G NR, were first proposed by Gallager in 1962. These codes can be decoded using soft decision iterative algorithms, for example, propagation belief or sum product. It has been presented, that the performance of the LDPC codes is close to the channel capacity for long block length [13]. Nowadays, this coding technique is widely used in wired and wireless standards, such as WLAN (IEEE 802.11n), WiMax (IEEE 802.16e), or optical fiber communications [14]. Compared to convolutional codes employed in 4G, LDPC codes provide higher coding gain and lower error floor with reduced computational complexity [12]. Focusing on the 5G deployment scenarios, high encoding/decoding efficiency, high throughput (especially for eMBB deployment), and support for the HARQ is required. Class of the LDPC codes called Quasi Cyclic LDPC (QC-LDPC) are used in 5G NR, which enables easier encoding/decoding hardware implementation, without perceptible performance degradation. These codes have the feature that each codeword is a shifted version of another codeword [13]. Based on their structure, encoding/decoding parallelism is naturally supported for these codes, which is beneficial for layered decoding [15]. These codes are used both for downlink and uplink data transmission [12].

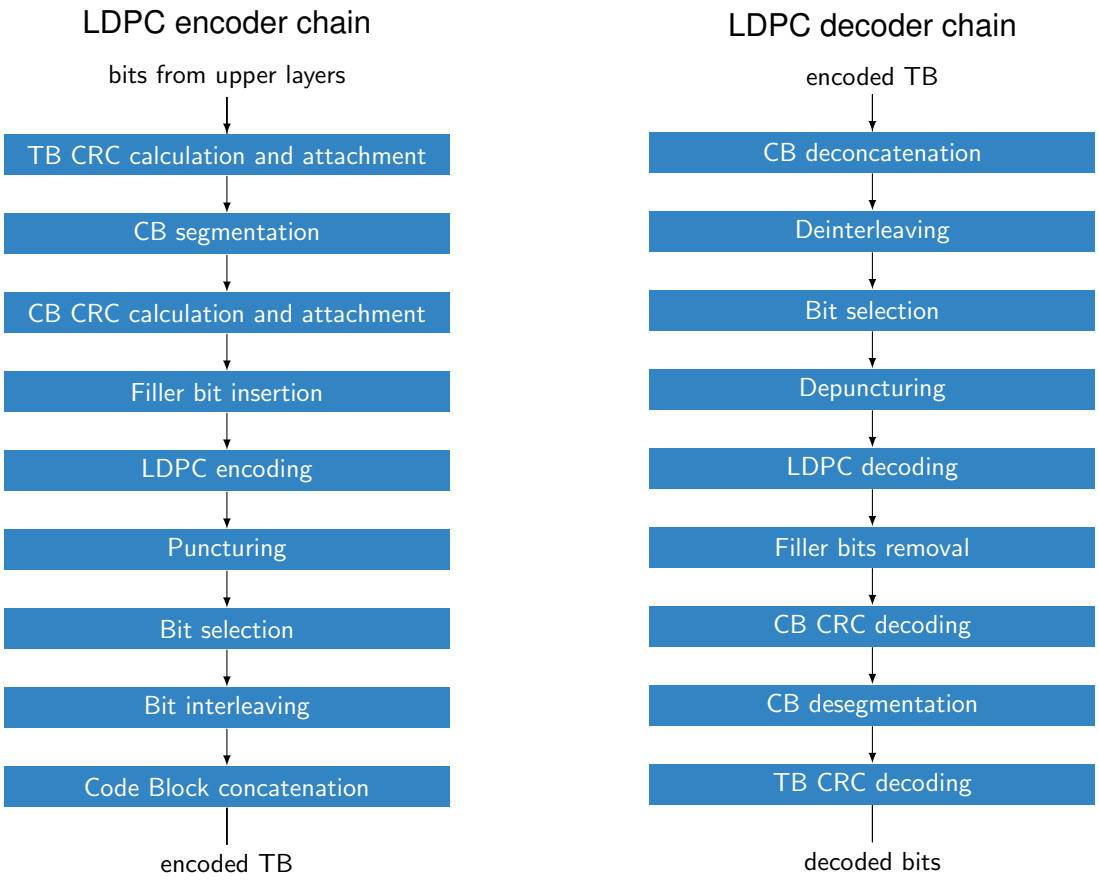

Figure 2.3: 5G NR LDPC encoder/decoder chain

2.4 QC-LDPC ENCODING CHAIN

The procedure of channel encoding is performed on the UE side in case of uplink transmission and on the gNB side in case of downlink transmission. Individual steps of this procedure, as defined in the 3GPP specifications TS 138.212 [16], are described in the following section. In 5G NR, data from the upper layers is delivered in the form of transport blocks to the physical layer. For these transport blocks of variable size, CRC is calculated and attached. The next step is a selection of LDPC base graph based on the coding rate and transport block size, followed by segmentation into the code blocks of equal size. CRC is then calculated and attached to each code block. LDPC encoding operation is then performed on each code block with its attached CRC, followed by the rate matching operation and HARQ management. Individual steps are shown in Figure 2.3.

2.4.1 TB/CB CRC calculation

Let's assume $a_0, a_1, ...a_{A-1}$ an information bits and $p_0, p_1, ..., p_{L-1}$ parity bits of the transport block, where A is the size of the number of input bits and L is the number parity bits. Based on the value of A, a cyclic generator polynomial is selected. Bits after this operation are denoted as $b_0, b_1, ..., b_{B-1}$. B is calculated as $B = A + L$, and denotes the size of the transport block after the CRC attachment.

2.4.2 Base graph selection

A base graph is selected based on the size of the transport block and target code rate. Selection of the base graph as a function of code rate and the transport block size is shown in Figure 2.4 Based on the 5G specifications, base graph 1 has a dimension of 46 x 68 with 22 systematic information columns. The maximum lifting size is set to 384. This leads to a maximum information payload of 8448 bits. Base graph 2 is dedicated to smaller payloads with lower code rates, so the dimension is 42 x 52, with 6,8,9 or 10 systematic information columns with respect to the payload size. The maximum lifting size is 384, the same as for the base graph 1. Maximum information payload, in this case is 3840 bits.

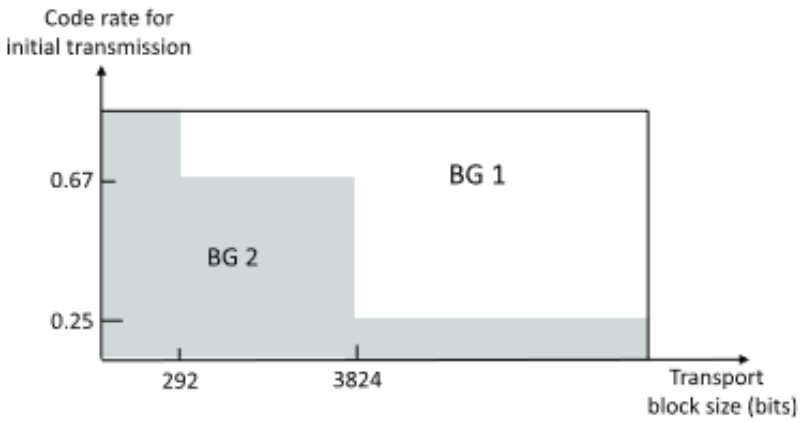

Figure 2.4: Base graph choice [12]

2.4.3 Segmentation and CRC attachment

Let's put $b_0, b_1, ..., b_{B-1}$ with size B presents an input message to the code block segmentation. When LDPC base graph 1 is selected, if B is higher than the maximum code block size $K_{cb} = 8448$, the input bit sequence is segmented, and an additional CRC of size $L = 24$ is attached. In the case of the base graph 2, the maximum code block size K_{cb} is set to 3840. Rules for code block segmentation are described in detail in the TS 38.212 3GPP specifications. Output sequence after the code block segmentation, and

additional CRC attachement is denoted by $c_{r0}, c_{r1}, \ldots, c_{r(K_r-1)}$, where $0 < r < C$. Parameter C denotes the number of code block after the segmentation. K_r denotes a number of bits in each code block r.

2.4.4 LDPC encoding

After the segmentation, each code block is encoded independently. There are two important parameters for the QC-LDPC codes: base graph and lifting size. Two versions of base graphs are defined in the 5G specifications and its selection depends on the code rate and size of the transport block. Once the base graph is specified, the parity check matrix for the LDPC encoder is created by the transformation of the base graph into the parity check matrix. Each entry in the base graph is replaced by the Z x Z identity matrix cyclically shifted to the right, where Z stands for a lifting size. The higher the lifting size is selected, the more elements the parity check matrix has. Entries of the base graph are values in a range from -1 to Z-1. Components of the base graph with value -1 are by convention replaced by all zero matrix and components with value 0 are replaced by identity matrix [12].

2.4.5 Rate matching

As for the encoding, rate matching and HARQ functionality are performed for each code block separately. The purpose of this operation is to adapt the code rate as well as provide a redundancy version demanded by HARQ functionality. There are three main processes behind this operation: bit puncturing, bit selection, and bit interleaving [12]. Based on the code block size, up to one third of the information bits is punctured. Remaining bits are then placed into the circular shift buffer. First, information bits are written to the buffer, followed by parity bits. There are four redundancy versions defined in 5G NR standards. Different set of bits representing the same set of information bits is then selected in each retransmission. Starting position of each redundancy version in the circular buffer is calculated based on the base graph selection and lifting size Z_c value. Order of redundancy version for particular retransmission round is specified to RV0 for initial transmission, followed by RV2, RV3 and RV1 for retransmission. The last stage of the rate matching is interleaving, which is a technique for mitigation of the burst errors in the transmission. The output sequence from the rate matching is denoted by $f_0, f_1, \ldots f_{E-1}$, where E is the number of bits of the rate matched code block.

2.4.6 Concatenation

Bit sequences from the rate matching stage are sequentially concatenated into the transport block sequence, which is then transmitted through the physical channel.

2.5 QC-LDPC DECODING CHAIN

Process of decoding is inverse to the encoding process as is presented in Figure 2.3. Signal received on the antenna is first processed by the lower physical layer functions and in form of Log-Likelihood Ratio (LLR) values delivered to the receiver decoding chain. First step in this procedure is the code block deconcatenation. Compared to classical block codes, which are usually decoded via an ML decoder, LDPC codes are decoded iteratively based on the Tanner graph, where a message passing algorithm provides a function of passing the message along individual lines of the Tanner graph between the isolated nodes. In each iteration of the algorithm, message is passed from the message node to the check nodes. Parity check condition $CH^T = 0$ has to be fulfilled in order to assume, that a received code word is valid. Estimates of the bit value are in to form of LLR, where negative LLR presents a stronger, that the value of the associated bit is 1. Positive LLR value indicates higher confidence, that the associated bit is value of 0. There are two main variants of the LDPC decoding algorithms: soft decision and hard decision forward error correction. These two approaches are compared in the paper [17], the results shows, that the BER performance of the soft decision LDPC decoder is higher regardless of the SNR level. All components of the processing chain performed for uplink transmission on gNB side are shown in Figure 2.3, the numbers listed in each function block of diagram refer to the corresponding section in the 5G specifications, concretely TS38.212 and TS38.211.

3 Implementation description

3.1 Software

In this section, sofware for operating the T1 and T2 card is described. There are two main software frameworks, which are further discussed. First one is DPDK framework, and the tesbbdev application on top of it, second part is dedicated to the OAI protocol stack, which is the main software tool used in this project for T1 and T2 performance testing. As the T1 card is already integrated and tested with the OAI stack, integration of the T2 card is one of the main goals of this project.

3.1.1 DPDK

Nowadays a few software frameworks for high-speed data processing exists, such as Netmap, Netslice, Intel DPDK or PF_ring. Widely used strategies by these software frameworks to achieve high-speed processing are usage of zero-copy processes, kernel bypass and circular buffers [7]. As the DPDK is chosen for implementation with the accelerator cards, main principles of this framework are further discussed.
DPDK is an open-source set of software libraries and drivers developed by Intel for fast packet processing. DPDK framework provides a core components for memory management, lockless buffer rings, support for multicore processing and zero-copy communication with PCIe devices. Kernel bypass is one of the key techniques allowing to reach a high throughput. DPDK Environment Abstraction Layer (EAL) provides a generic interface to the application, while the environment components, such as operation system or hardware is abstracted away. Typical tasks performed by EAL are DPDK launching, enumerating and loading hardware buses, devices and Poll Mode Driver (PMD), management of the CPU cores and memory (non-uniform memory

access), making hardware devices available to DPDK PMD by mapping their registers into memory [9]. Typically, DPDK application runs in the user space using pthread library. In order to increase performance, physical memory allocation is performed using huge pages, which are exposed to DPDK service layers through mempool library. Hugepages are contiguous blocks of memory with size larger than the size of regular pages used in the system, which is 4 kilobytes on Intel 64 and IA-32 architectures [18]. In order to access page memory location during the code runtime, translation from virtual addresses to physical addresses is performed. Mapping of of virtual addresses used by software to physical addresses used by hardware is stored in page tables. For performance improvement, recently used page addresses are stored in Translation Lookaside Buffer (TLB) cache with limited size. TLB miss occurs, when access to the memory address, which is not withing the page contained in the TLB, is required. In this case, relatively performance expensive operation of fetching of the page address from the global TLB is performed. In order to access large memory locations (DPDK usually uses up to tens of gigabites), usage of hugepages enables a reduction of a number of page entries, as well as page faults and TLB misses, compared to regular memory [18]. This can lead to significant improvement of the data plane performance [19]. Main core initialization and launch of the DPDK application is done by *rte_eal_init()* call. »» rewrite

Poll Mode Driver

Main functionality of the poll mode drive is data forwarding between the application and the HW accelerator. Kernel bypass is usually used for queue manipulation in order to avoid multiple copies of the data between the application and the hardware. If the vendor specific PMD is required to use with DPDK, build option has to be set in DPDK build folder to enable the compilation of the driver as a shared library or everything as a single combined library [20].

testbbdev Application

testbbdev is an utility provided by DPDK, dedicated for the performance measurement of the PMD in the bbdev framework. Tests for measuring throughput, latency, Block Error Rate (BLER), validation and sanity test are available. Each test can be customized by passing various parameters to the application [**9.**]. Usage of this utility allows us to compare performance parameters of the PMD used with the particular accelerator card. In case of the Xilinx T1 Telco Accelerator Card, BBDEV PMD is provided by the company VVDN, for the Xilinx T2 Telco Accelerator card is the PMD provided by company AccelerComm. Performance measurements and their analysis are presented in the following chapter. Testing with this utility is done with test vector file, which

contain data for processing and parameters for the LDPC decoder, which need to be passed to the accelerator card.

Shared library API

The main DPDK operation structure for the LDPC decode is called *rte_bbdev_ldpc_dec*. There are multiple members of this structure which specifies parameters for a single decode operation. In frame of the DPDK, operation can be performed in two mode: code block mode or transport block mode. In the current setup, only code block mode is supported, so decode operation is performed on one code block at a time. One of the mandatory member structures for the decoding operation is *input* structure, which presents the input encoded code block. This is a Virtual Circular Buffer data stream, carrying LLR values of the original code block. Another member structure is a *hard_output*, which carries a bits of the decoded code block. Another important parameters, which specifies the decoding operation are rate matching redundancy version, maximum number of iterations of the decoder, base graph, lifting size length of the circular buffer and modulation order [21].

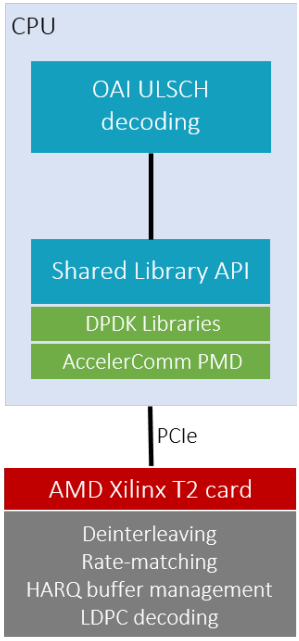

Figure 3.1: Shared library API integration

3.1.2 System setup

Device binding

In proposed solution, device has to be binded to the kernel driver *igb_uio* to ensure proper functionality with the DPDK. Following command is used to perform an operatio

of binding the device specified by its interface address with the specific kernel driver. DPDK utility *dpdk-devbind* is used, which allows to bind/unbind available devices from the driver and check their status. Address of the device is discovered using *lspci* command. Status of the devices after sucessful binding is shown in Appendix 6.1.

```
$ ./dpdk-devbind.py -b igb_uio 21:00.0
```

Hugepages setup

As previously mentioned, EAL perform the function memory management for the DPDK application. During the overall application initialization, allocation of large contiguous physical memory is performed using huge pages. Application called *dpdk-hugepages* is available in DPDK for adjustment of the huge pages setup on the system. In proposed solution implementing T1 and T2 cards for channel decoding offload, 32 hugepages of the size 1 Gb are required to ensure correct functionality with the DPDK. This setup is done using following command. Status of the hugepages is shown in Appendix 6.2.

```
$ ./dpdk-hugepages.py -p 1G --setup 32G
```

3.1.3 OpenAirInterface Protocol Stack

In order to integrate accelerator cards with OAI, shared library is created to implement the interface for DPDK. It also provides a translation of the OAI descriptors for ULSCH decoding, such as code rate or lifting size into the required DPDK parameters for LDPC decoding. On top of the DPDK, testbbdev application is used. Only codeblock mode is supported by for T2 card, which means that each time only one code block is decoded. Support for the decoding of the whole transport block is not available for the card yet.

OAI implementation of the channel processing

nr_ulsim

Multiple physical layer unitary simulators are available in the OAI 5G, such as nr_dlsim, nr_prachsim, nr_puschsim or nr_ulsim. All these tests are designed for testing the behavior of particular physical channels. These tests serve as a tool for evaluation of the new features implemented into to the code. In the process of OAI code development and integration, all tests has to be passed before the new code is added to the repository.
nr_ulsim is designed for simulation of the ULSCH transmitter, wireless channel and ULSCH receiver. This application is important in the process of T2 card integration,

as the channel decoding offload can be tested. Besides the testing of the behavior and stability of the newly implemented code, it returns error statistics and statistics of the processing time needed to perform individual steps from the processing chain of the PHY channel. In further development stages, HARQ functionality is primarily tested using this application.

NR-SOFTMODEM

One of the main executables available in the 5G OAI is nr-softmodem, which aims to provide 3GPP compliant implementation of the 5G gNB. Parameters of the gNB are configurable through configuration file with specific struscture and parametrs to set, which has to be passed as an command line parameter for running the *nr-softmodem* application. As the *nr_ulsim* provides a valuable testing tool in process of the development, this solution allows for the setup of the end to end scenario, alongside with OAI CN and COTS UE.

3.2 Hardware

In this chapter, hardware used for setting up the testbeds and for simulations is briefly described. More details are provided about the hardware accelerator cards, where the channel decoding is offloaded.

3.2.1 Xilinx T1 Telco Accelerator Card

Xilinx T1 Telco Accelerator Card aims to provide high throughput, energy efficiency and low latency suitable for 5G O-DU, 5G O-CU and vBBU deployments. As displayed on figure 3.2, this card is equipped by two FPGA device for the lookaside acceleration of L1 processing, O-RAN 7.2 fronthaul termination and functions. 16nm Zynq® UltraScale+™ MPSoC offers a 3GPP O-RAN split 7.2 fronthaul termination for eCPRI/RoE/SyncE 5G protocols with reference clock timing circuit located on the board for IEEE Std 1588 timing functionality. Zynq UltraScale+ RFSoC FPGA device includes hardened soft-decision Forward Error Correction (FEC), which provides decoding and encoding acceleration for 4G/5G systems along with rate matching and CRC logic wrapper functions. This card supports PCIe Gen3 interface with 16 lines, bifurcated into two PCIe Gen3 x8 lines in order to ensure separate communication of the onboard FPGAs with the host [22].

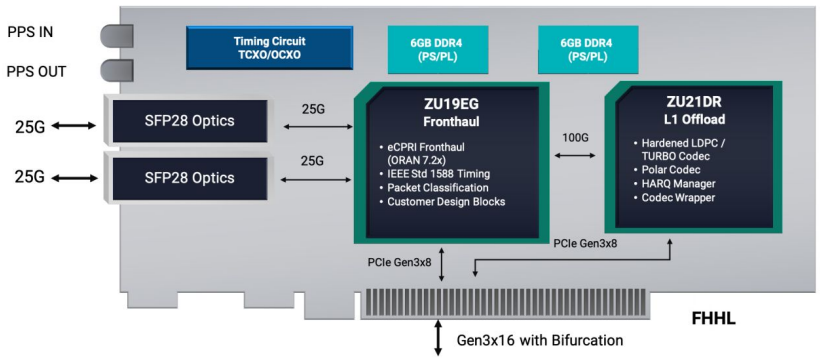

Figure 3.2: Block diagram of the Xilinx T1 Telco Accelerator Card [22]

3.2.2 Xilinx T2 Telco Accelerator Card

Target application of the Xilinx T2 Telco Accelerator Card is L1 processing acceleration, which brings increase of performance in terms of lower latency, higher throughput and lower energy consumption for 5G O-DU deployments, as well as massive MIMO radio configurations. This card is equipped by one FPGA 16 nm Zynq® UltraScale+™ RFSoC ZU48DR, with hardened soft-decision FEC and wrapper functions to provide rate matching, HARQ buffer management and CRC logic. This card supports PCIe Gen3 x 16 or 2x PCI Gen4 x 8 interface for communication and data transmission with the host [23].

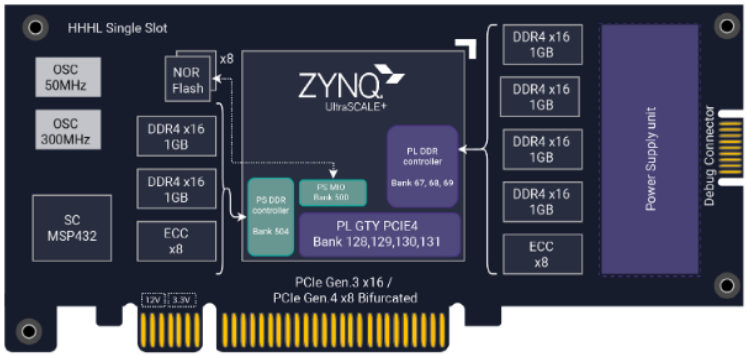

Figure 3.3: Block diagram of the Xilinx T2 Telco Accelerator Card [23]

3.2.3 Comparison of the accelerator cards

Comparison of the cards is done based on performance metrics measurements provided by the vendor of the card, available in [23] [22]. Comparison of important performance metrics of the used cards is presented in the Table 3.1 below.

	Xilinx T1 Telco Accelerator Card	**Xilinx T2 Telco Accelerator Card**
Optimization	Fronthaul and L1	L1
FPGA	ZU19P+ZU21DR	ZU48DR
PCIe	2x Gen 3x8	Gen 3x16 or 2x Gen 4x8
L1 Encode	17.7 Gbps	35 Gbps
L1 Decode	8.1 Gbps	12 Gbps

Table 3.1: Comparison of T1 and T2 accelerator card

3.2.4 Installation of the cards

Vivado Lab Solutions – 2019.2 is used to program the flash of the cards with bitstream image file provided by Accelercomm (T2 card) or VVDN (T1 card), solution partners of AMD Xilinx. After the successful programming, cards are recognized by the host.

3.2.5 USPR N310

USRP N310 is networked Software Defined Radio (SDR) developed by Ettus Reserch. In the proposed test setup, this device serves a function of gNB radio. Key feature of this SDR is a high channel density with 4 RX and 4 TX channels, where each channels offers up to 100 MHz of instantaneous bandwidth, Xilinx Zynq-7100 SoC adopted as a baseband processor provides real-time and low latency processing [24].

Figure 3.4: USRP N310

3.2.6 Quectel RM500Q-GL

Quectel RM5000Q-GL, 3GPP Release 15 compliant module optimized for eMBB and IoT applications, is used as an COTS UE in E2E test setup. Between the key feature of this module belongs its support of 3G, 4G LTE and both 5G standalone (SA) and non-standalone (NSA) modes for sub 6Hz frequency range, coverage of all the mainstream carries worldwide and support of multiple location technologies (GPS, GLONASS, BDS and Galileo). For 5G Standalone (SA) mode in Frequency Range (FR) 1, this module support data rate up to 2.1 Gbps in downlink and 900 Mbps in uplink. Configuration of this module is performed through AT commands [25].

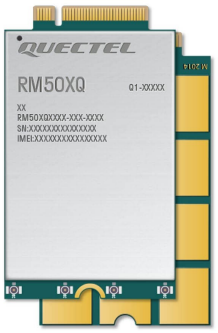

Figure 3.5: Quectel RM500Q-GL module [25]

4

Analysis of the Results

4.1 DPDK simulation results

4.1.1 testbbdev simulation

As mentioned in the previous chapter, *testbbdev* is the DPDK utility, that allows for performance measurements of the PMD. As the cards are using different PMD implementation, based on the Xilinx solution partner, which provides not only the software PMD, but also bitstream image for the programming of the card. Testing of performance parameters provided PMD is done using *testbbdev* application. PMD for T1 card provided by VVDN, PMD for T2 card provided by Accelercomm. DPDK *testbbdev* testing tool is based on using predefined vector files, where parameters for LDPC decoder, as well input values and expected output values are listed. In order to compare performance of the T1 and T2 accelerator card, following parameters are passed to the application:

Parameter	**Value**
Number of queues	2
Burst size	32
Number of operations	512
Test mode	throughput
Max. number of LDPC decoder iterations	3

Table 4.1: bbdev configuration

An example command for running this application is displayed bellow.

```
$ sudo ../../build/app/dpdk-test-bbdev -l 0-6 -a 21:00.0 -- -n 512 -l 2 -c
  throughput -v test_vectors/ldpc_dec_v8480.data -t 3
```

Both cards are tested using *testbbdev* tool in throughput mode with the same vector file. T1 card achieved a total throughput of 3.76 Gbps per 2 cores. Total throughput of the T2 card is 6.473 Gbps per 2 cores.

4.2 NR_ULSIM SIMULATION RESULTS

In the following chapter, results of the simulations using physical uplink channel simulator provided by OAI are shown and analysed. All the simulations are ran on the server AMD EPYC 7282 CPU @ 2.8 GHz with 32 cores.

4.2.1 SIMULATION SCENARIO

In order to test performance and behavior of the implemented solution, simulations are carried out using *nr_ulsim*. There are multiple physical simulators provided by OAI for testing individual transport channels and coding schemes. *nr_ulsim* is a physical simulator dedicated for testing uplink shared channel in a controlled scenario, which is adjusted by multiple available options such as coding scheme, bandwidth or DMRS configuration. As an result of the simulation, simulation parameters, values of performance indicators and processing time of individual steps of physical channel processing chain are displayed. Shell scripts are developed in order to automatize testing and processing of the obtained results. Sample command for running this simulator is displayed below.

```
$sudo ./nr_ulsim -n1000 -s30 -m10 -r273 -R273 -P -o t2
```

In this command, *-n* option stands for the number of iterations of the simulation, *-s* option set up SNR, by *-m* option modulation and coding scheme is chosen, *-r* option set up the minimum number of physical resource blocks and *-R* option the maximum number of physical resource blocks used in the simulation. Offload of the decoding to the accelerator card is selected by option *-o*. In frame of this project, support for both Xilinx T1 Telco Accelerator card and Xilinx T2 Telco Accelerator Card is enabled, concrete card is then selected using *-t1* or *-t2* option. By using *-P* option, time statistic of the simulation are displayed. Sample output of the OAI *nr_ulsim* simulator is present in 6.6. Configuration of the simulator is shown in Table 4.2.

Parameter	**Value**
Bandwidth	100 MHz 273 PRBs
Subcarrier spacing	30 kHz
OFDM symbols	12
Channnel model	AWGN
MCS	0-28
SNR	30 dB
Max. number of LDPC decoder iterations	10

Table 4.2: Simulator configuration

All simulations are carried out on AMD EPYC 7282 CPU @ 2.8 GHz with 32 cores.

4.2.2 Comparison of the processing time

Processing time needed for successful channel decoding for all 5G NR Modulation Coding Scheme (MCS) is displayed in Figure 4.1 Four solutions of the channel decoding are compared. As previously mentioned, software implementation of the LDPC decoder in the OAI protocol stack allows for the multi-thread execution, where multiple segments are decoded in parallel. This solution significantly reduces processing time of the decoding. Results of simulation with software LDPC decoder parallelized over 9 CPU cores are marked on figure 4.1 as *SW LDPC multi-thread*. *SW LDPC single-thread* stands for the software decoder implementation, where only one core is employed. In case of the channel decoding offload, only one core is selected during the EAL initialization and employed for the LDPC offload.

With respect to the results of the simulation, best performance is achieved by the offload of the channel decoding to the T1 accelerator card. The worst performance is observed for single-thread software implementation of the decoder. Using multi-thread solution proves a good results also compared to the offloading solution, however with penalty of higher CPU utilization of the host machine. Focusing on the accelerator cards, there is an almost linear dependence of the processing time and MCS selected. For all MCS, measured processing time is lower than 500 us for the T2 card and lower than 350 us for the T1 card.

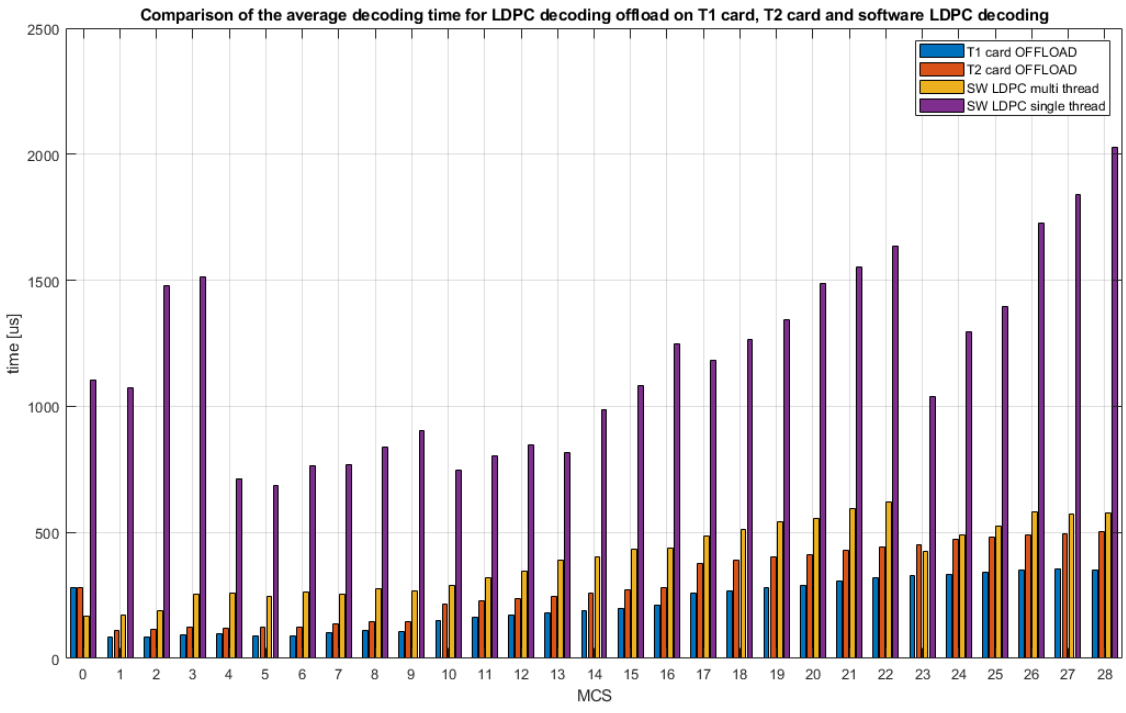

Figure 4.1: Comparison of the average processing time

4.2.3 Comparison of the SNR to achieve 10% BLER

Results of the SNR required to achieve 10 % block error rate (BLER) in the first transmission are shown on figure 4.2. Performance of the solution with channel decoding offload to the T1 card, T2 card and software implementation of the LDPC decoder are comaped. HARQ functionality is disabled in this simulation. Based on the results, the biggest difference of these solution is for low MCS values (MCS < 10) and for high MCS values (MCS > 24). For low MCS values, software implementation provides the most stable solution. For MCS values higher than 24, offload to the T2 cards demonstrates better results, than the other solutions. BLER lower than 10% in the first transmission is achieved for all MCS values with SNR lower than 20 dB. On the other hand, T1 card in this region proves significantly worse results than the other solutions. Difference in the T1 offload and T2 offload performance could be caused by bitstream image installed on the card and PMD provided by different vendor.

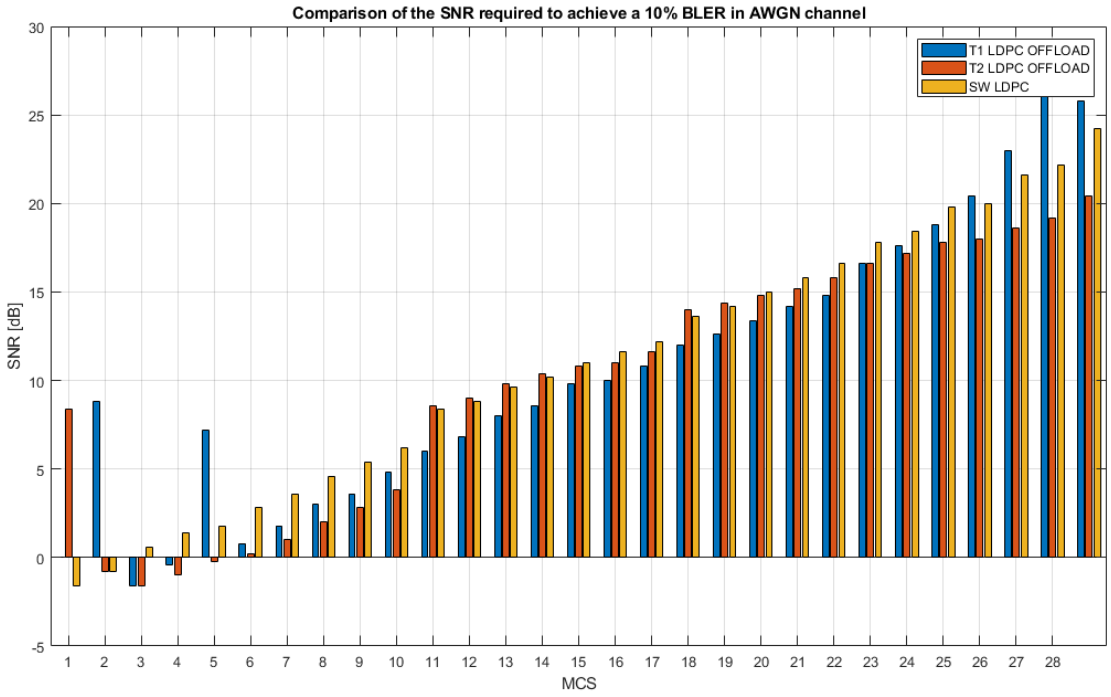

Figure 4.2: Comparison of SNR to achieve 10% BLER

4.3 Over-the-air test results

OTA test is performed in order to validate solutions using accelerator cards and to compare the throughput performance with the software LDPC decoder available in the OAI. Throughput measurements are not performed for the solution with the channel

decoding offload to the Xilinx T1 Telco Accelerator card.

4.3.1 Test setup 1 - Validation

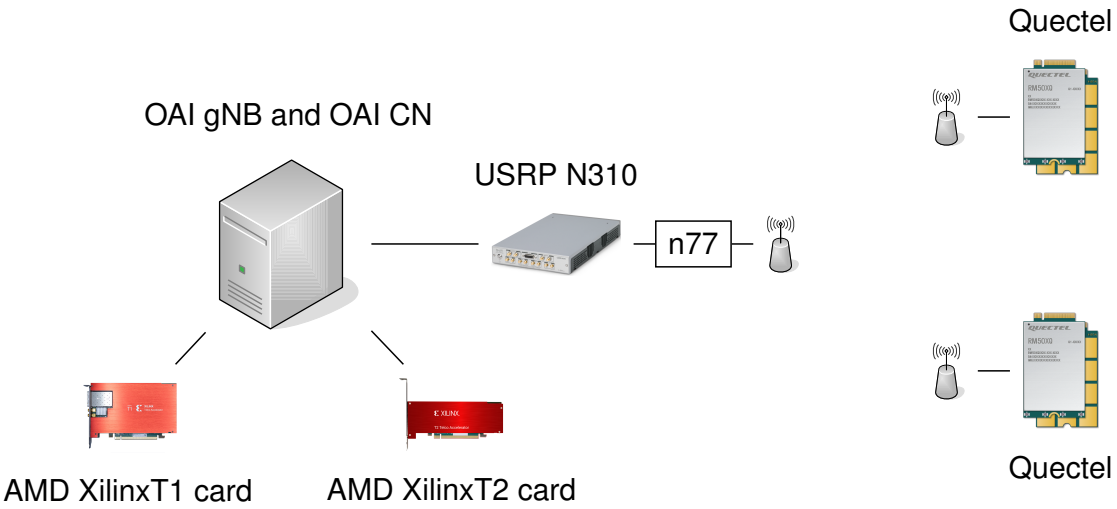

Figure 4.3: Testbench for validation of T1 and T2 offload

As depicted on figure 4.3, both OAI gNB and OAI CN are running on the server AMD EPYC 7282 CPU @ 2.8 GHz with 32 cores. OAI Core network is deployed in Docker containers. OAI gNB is deployed using *nr-softmodem* executable. USPR N310 is used as an radio unit. Finally, two Quectel QM500G modules, connected to host laptops, are used as COTS UEs. Configuration of the network and the radio unit is done via the configuration file, where parameters such as Public Land Mobile Network (PLMN), Data Network Name (DNN), parameters for the USRP setup or IP address of the USRP are defined. Parameters selected for the OTA setup are as follows:

Parameter	**Value**	
Band	77	
Bandwidth	60 Mhz	162 PRBs
Subcarrier spacing	30 kHz	
TDD configuration	2.5 ms	DDXUU
OFDM symbols	12	
Antenna setup	SISO	

Table 4.3: Test setup 1 - configuration

For the purpose of validation of the accelerator card functionality in real conditions, two COTS UEs are connected to the network. One of the UEs is streaming a video, the other UE is receiving this video. In this setup, heavy downlink and uplink traffic is

carried over the network. Validation test is performed with both, T1 and T2 accelerator cards.

4.3.2 Test setup 2 - Uplink throughput measurements

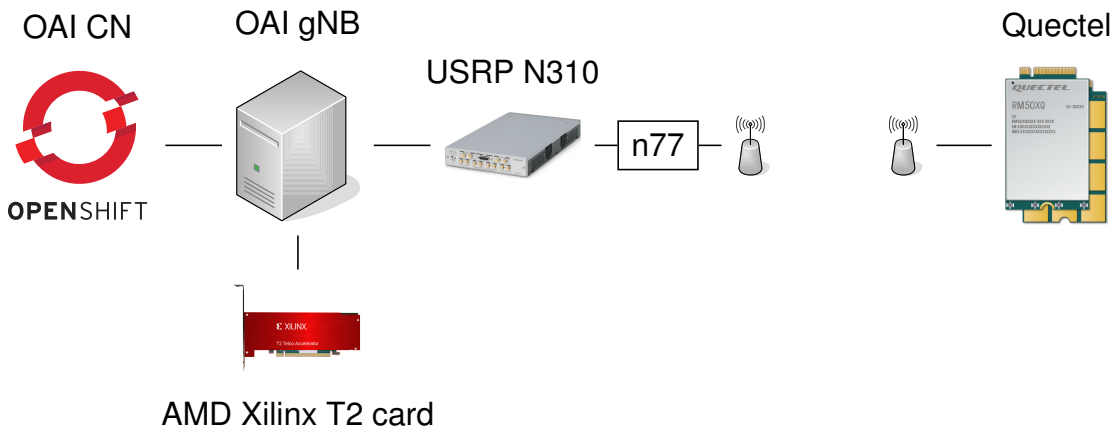

Figure 4.4: Testbench for measurements with T2 LDPC offload

In the following test setup, OAI gNB and is deployed on the server AMD EPYC 7282 CPU @ 2.8 GHz with 32 cores. OAI core network in this setup is deployed in the Openshift Cluset. COTS UE - Quectel module is connected to the host laptop. Example comamnd for running OAI *nr-softmodem* application is shown below. Using an option of *–ldpc-offload-enable 1*, LDPC offload to the accelerator card is enabled and with *–loader.ldpc_offload.shlibversion* option version of the accelerator card is selected. Configuration file for the gNB is selected by the *-O* option.

```
sudo numactl --cpunodebind=1 --membind=1 ./nr-softmodem -O conf_20899_n310.
    conf --sa --tune-offset 30000000 --ldpc-offload-enable 1 --loader.
    ldpc_offload.shlibversion _t2
```

Setup of the Quectel module is done via AT commands as descibed in Appendix 6.4. For the connection management and IP address setup on the UE side is used Quectel Module Connection Manager.

Uplink throughput tests are performed using *iperf3*, which is a command-line tool dedicated for the network analysis and performance measurements. Traffic in this setup is generated and transmitted from the COTS UE and received at the OAI gNB. Example of the *iperf3* command is displayed here.

Transmitter (UE) side: UE in this setup acts as an client with an IP address 12.1.1.152, connected to the server with IP address 172.21.6.12. *-i* option presents an interval (in seconds) between the periodic test, followed by a measurement report. Duration of the test is expressed by *-t* option.

```
$ iperf3 -c 172.21.6.12 -B 12.1.1.152 -i 1 -t 300-u -b 75M -p 5201
```

Receiver (gNB) side: Traffic server is deployed with the OAI CN on Openshift Cluster. This pod is exclusively used for testing. Multiple iperf3 processes in server mode with different ports are simultaneously running on this pod, which allows a UE to perform uplink or downlink (in reverse iperf3 mode) iperf3 testing. Sample command to run an iperf3 server is displayed bellow:

```
$  iperf3 -s -p 5201 -D
```

Iperf3 parameters selected for the testing are as follows: duration of 300s, UDP protocol, target bitrate 75 Mbps.

4.3.3 Uplink throughput measurement

Comparison of the throughput measurements in case of the OAI software LDPC decoder and proposed solution using LDPC decoding offload to the Xilinx T2 Telco Accelerator Card are displayed in Figure 4.5. Statistical evaluation of the throughput measurements is then shown in Table 4.4. Based on the test results, solution with the LDPC decoding offload to the T2 card performed better in terms of the average throughput compared to the software OAI LDPC decoder implementation. It is also important to mention, that the measurements are carried out in the laboratory, where measurements can be influenced by many factors. Statistics of the uplink throughput measurement were obtained by processing of the iperf3 test report.

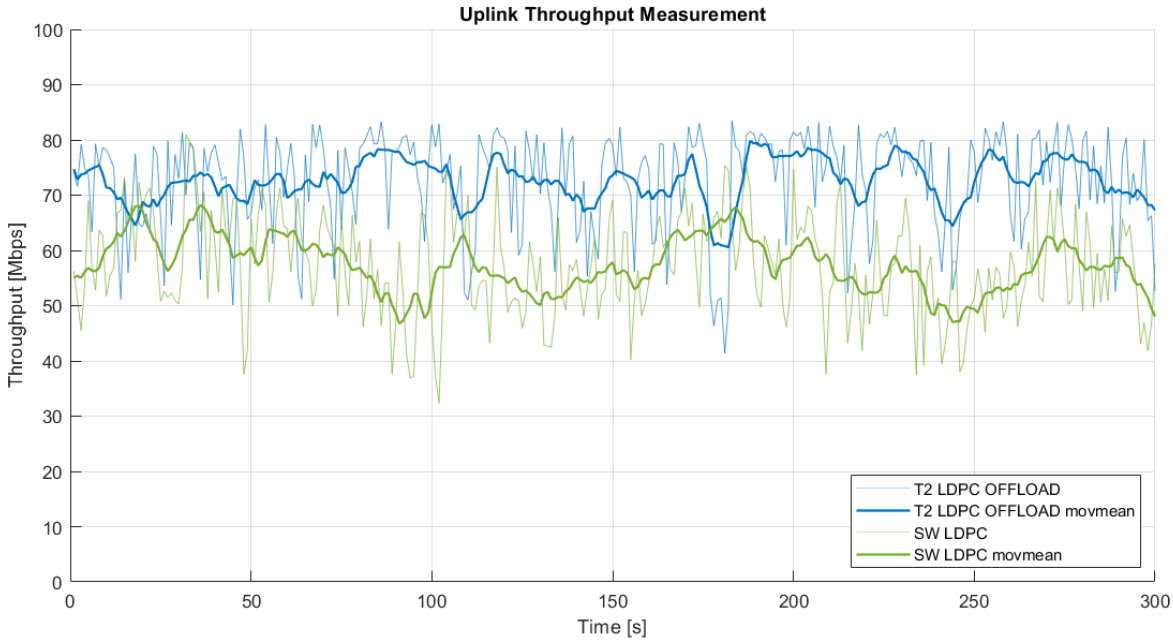

Figure 4.5: Uplink throughput measurement

	OAI SW LDPC decoder	OAI T2 LDPC Offload
Min UL Throughput	32.3 Mbps	41.4 Mbps
Avg UL Throughput	57.3 Mbps	72.6 Mbps
Max UL Throughput	81 Mbps	83.5 Mbps

Table 4.4: Uplink throughput statistics

4.3.4 CPU Utilization measuremnt

CPU utilization of the OAI gNB is measured during the iperf uplink test. Comparison between the setup without the LDPC offload and setup with LDPC offload to the T2 card is provided. In Table 4.5, CPU utilization, as well as measured throughput is displayed for both setups. Monitoring of CPU utilization and capture of the measured data is accomplished with s-tui terminal tool [26].

	OAI SW LDPC decoder	OAI T2 LDPC Offload
Min CPU utilization	4.8 %	5.3 %
Avg CPU utilization	9.9 %	6.7 %
Max CPU utilization	13.8 %	7.7%

Table 4.5: CPU utilization statistics

4.3.5 Analysis of the results

In the case of the first scenario, functionality of the LDPC decoding on accelerator cards is tested in the test setup, where two Quectel modules are connected to the OAI gNB and OAI CN. Validation of the correct functionality of the LDPC decoding, is first done using the *iperf* tool. One module is sending uplink traffic to the OAI gNB, the other module is receiving downlink traffic from the OAI gNB. An additional step of validation is a video streaming from one UE to the other. This scenario is set up for both hardware accelerators, Xilinx T1 Telco Accelerator Card and Xilinx T2 Telco Accelerator Card. In the case of the T1 card, HARQ management function is disabled, as this function is not supported by the current software of the card. In the case of the T2 card, HARQ functionality is not fully available. Based on the observations, only the second retransmission (redundancy version 3) allows for an error correction. Even with the mentioned limitations with the HARQ management, both solutions proved stable connection and smooth video transmission.
Uplink throughput measurements is performed only with the T2 card. Compared to the OAI software implementation of the LDPC decoder, uplink throughput is improved

by using T2 card for the channel decoding offload. Average uplink throughput for the software decoder is 57.3 Mbps, for the setup with channel decoding offload, average uplink throughput is 72.6 Mbps. Improvement of the channel throughput when accelerator card is employed could be caused by the difference in the decoding algorithm and also in the decoder setup. Observations during testing shows that maximum number of decoder iterations can be higher without the significant processing time increase in case of the LDPC offload to the T2 accelerator card. Optimal value of the number of decoder iterations is set ,based on observations, to 10. It is also important to mention, that software decoder was not optimized, as this is out of the scope of this diploma thesis. Based on the CPU utilization information captured during the OTA throughput test, employment of the hardware accelerator for the LDPC processing enables reduction of the CPU utilization as presented in Table 4.5.

5

Conclusions and Future Works

This diploma thesis aims to provide a review of the current implementation of the hardware acceleration for the LDPC decoding in the OAI protocol stack. A comparison of the existing solution with the Xilinx Telco T1 Accelerator Card and the newly integrated solution with the Xilinx Telco T2 Accelerator Card dedicated exclusively to L1 processing acceleration is provided.

In the first part of the thesis, a study of both software and hardware components is done. Understanding of OAI protocol stack and available simulation tools, with a focus on the channel decoder, as well as a study of the DPDK software framework is necessary in the entire process of the integration of the accelerator cards. Simultaneous support for both T1 and T2 cards was implemented within the OAI code, which enables for a conditional compilation of T1 or T2 shared library, as well as selection of the device for the LDPC offload in the case of running OAI physical uplink simulator *nr_ulsim* or deploying OAI gNB application *nr-softmodem*.

Simulations were carried out to compare several implementations of the channel decoding. Results of the simulations prove, that the T2 card is able to achieve better performance in terms of SNR required to achieve 10% BLER compared to the results obtained with the T1 card or software LDPC decoder, improvement is noticed especially for higher MCS, where the T2 card performs up to 10% better on average. In terms of the processing time of the decoding operation, better performance shows the T1 accelerator card, processing time is on average 100 us lower than for the T2 card. Both cards were tested in OTA setup, where OAI gNB and OAI CN were deployed on a server, SDR USRP N310 served the function of gNB radio, and Quectel 5G module was used as the COTS UE. Compared to the software implementation of the LDPC decoder, the solution with hardware accelerators achieves better results for the uplink performance, while the CPU utilization of the system is reduced.

In the current solution, buffers of LLR values are copied between the OAI and DPDK interface, as a future improvement, a zero-copy approach could be investigated. In order to implement this solution, usage of the DPDK could be extended for the entire gNB receiver. Another important feature of the 5G RAN is the HARQ functionality. Due to the fact, that HARQ was not fully supported by the software for operating the card, it opens an option for further development and testing with HARQ functionality. Finally, *bbdev-lite* solution, provided by AccelerComm, could be further explored. This solution provides an additional abstraction layer on top of the DPDK, as well as multi-queue processing, which could bring further improvements to the current implementation.

References

[1] Florian Kaltenberger et al. "OpenAirInterface: Democratizing innovation in the 5G Era". In: *Computer Networks* 176.C (). DOI: 10.1016/j.comnet.2020.107284. URL: https://par.nsf.gov/biblio/10215366 (cit. on pp. 1, 7).

[2] Mihai Enescu. *New Radio*. Apr. 2020. ISBN: 9781119582335 (cit. on p. 3).

[3] Erik Dahlman, Stefan Parkvall, and Johan Skold. *5G NR*. Sept. 2020. ISBN: 9780128223215 (cit. on pp. 3, 5).

[4] Robert G. Gallager. *Low Density Parity Check Codes*. 1963. URL: https://web.stanford.edu/class/ee388/papers/ldpc.pdf (cit. on p. 3).

[5] Shao et al. *Survey of turbo, LDPC and Polar Decoder ASIC implementations*. Jan. 2019. URL: https://eprints.soton.ac.uk/427712/ (cit. on p. 4).

[6] Elissaios Alexios Papatheofanous, Dionysios Reisis, and Konstantions Nikitopoulos. *LDPC Hardware Acceleration in 5G Open Radio Access Network Platforms*. Sept. 2021. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9610039 (cit. on p. 4).

[7] Nanda Kishore et al. *Survey on software solution for High Performance Packet Processing*. Jan. 2021. URL: https://www.ijert.org/research/survey-on-software-solution-for-high-performance-packet-processing-IJERTV10IS020266.pdf (cit. on pp. 4, 13).

[8] *Soft-Decision FEC Integrated Block LogiCORE IP Product Guide (PG256)*. 2022. URL: https://docs.xilinx.com/r/en-US/pg256-sdfec-https://docs.xilinx.com/r/en-US/pg256-sdfec-integrated-block/IP-Facts?tocId=HkDYNxA%2014Aek~XI7Cw61Ow-block%5C%2FIP-Facts%5C%3FtocId (cit. on p. 4).

[9] *Memory in DPDK part 2: Deep Dive into Iova*. URL: https://www.intel.com/content/www/us/en/developer/articles/technical/memory-in-dpdk-part-2-deep-dive-into-iova.html (cit. on pp. 4, 14).

[10] *5G LDPC IP Core Solution Product Brief (HTK-5G-LDPC)*. HITEK SYSTEMS. URL: https://hiteksys.com/pdf/5G-LDPC-IP-Core-Product-Brief.pdf (cit. on p. 4).

[11] *Lookaside Hardware Acceleration*. URL: https://www.accelercomm.com/resources (cit. on p. 4).

[12] Douglas H. Morais. *Key 5G Physical Layer Technologies*. Aug. 2020. ISBN: 9783030514402 (cit. on pp. 5–7, 9, 10).

[13] Martin Tomlinson et al. *Error-correction coding and decoding bounds, codes, decoders, analysis and applications*. Springer International Publishing, 2018 (cit. on p. 7).

[14] Anuj Verma and Rahul Shrestha. *A New Partially-Parallel VLSI-Architecture of Quasi-Cyclic LDPC Decoder for 5G New-Radio*. 2020 (cit. on p. 7).

[15] Jung Hyun Bae et al. *An overview of channel coding for 5G NR cellular communications: APSIPA Transactions on Signal and Information Processing*. June 2019 (cit. on p. 7).

[16] 3GPP. *5G; NR; Multiplexing and channel coding*. Technical Specification (TS). Version 15.3.0. URL: https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf (cit. on p. 8).

[17] Jose Rinu and Ameenudeen Pe. *Analysis of Hard Decision and Soft Decision Decoding Algorithms of LDPC Codes in AWGN*. 2015 (cit. on p. 11).

[18] Antanoly Burakov. *Memory in DPDK, Part 1: General Concepts*. Aug. 2019. URL: https://docs.vmware.com/en/VMware-Telco-Cloud-Platform---5G-Edition/2.0/telco-cloud-platform-5g-edition-data-plane-performance-tuning-guide/GUID-E4BF8181-C2FF-43FE-8D25-848996C8D55C.html (cit. on p. 14).

[19] vmware. *Data Plane Acceleration Using DPDK*. May 2021. URL: https://docs.vmware.com/en/VMware-Telco-Cloud-Platform---5G-Edition/2.0/telco-cloud-platform-5g-edition-data-plane-performance-tuning-guide/GUID-E4BF8181-C2FF-43FE-8D25-848996C8D55C.html (cit. on p. 14).

[20] Nayan Gadre. *Writing a PMD for DPDK*. Oct. 2020. URL: https://medium.com/powerof2/writing-a-pmd-for-dpdk-56e6388467a8 (cit. on p. 14).

[21] *API*. 23.03.0. URL: https://doc.dpdk.org/api/ (cit. on p. 15).

[22] *Solution Brief, Xilinx T1 Telco Accelerator Card*. AMD XILINX, 2021. URL: https://www.xilinx.com/content/dam/xilinx/publications/product-briefs/xilinx-t1-product-brief.pdf (cit. on pp. 17–19).

[23] *Product brief - xilinx T2*. URL: https://www.xilinx.com/content/dam/xilinx/publications/product-briefs/xilinx-u25N-product-brief.pdf (cit. on pp. 18, 19).

[24] *USRP N310 Simplifying SDR Deployment*. Ettus Research, 2019. URL: https://www.ettus.com/wp-content/uploads/2019/01/USRP_N310_Datasheet_v3.pdf (cit. on p. 19).

[25] *Quectel RM50XQ series*. URL: https://www.quectel.com/wp-content/uploads/2021/09/Quectel_RM50xQ_Series_5G_Specification_V1.1.pdf (cit. on p. 20).

[26] *The Stress Terminal UI: s-tui*. https://github.com/amanusk/s-tui. version 1.1.4 (cit. on p. 29).

6 Appendix

6.1 Appendix I - Device binding

```
[eurecom@avra usertools]$ sudo ./dpdk-devbind.py -s

Network devices using DPDK-compatible driver
============================================
0000:81:00.0 'Device 903f' drv=igb_uio unused=

Baseband devices using DPDK-compatible driver
=============================================
0000:21:00.0 'Device 9048' drv=igb_uio unused=
```

6.2 Appendix II - Hugepages setup

```
[eurecom@avra usertools]$ ./dpdk-hugepages.py -s
Node Pages Size Total
0    32    1Gb  32Gb
1    32    1Gb  32Gb

Hugepages mounted on /dev/hugepages
```

6.3 Appendix III - card initialization

Xilinx T1 Telco Accelerator Card

```
EAL: Detected 32 lcore(s)
EAL: Detected 2 NUMA nodes
EAL: Multi-process socket /var/run/dpdk/b6/mp_socket
```

```
EAL: Selected IOVA mode 'PA'
EAL: 942 hugepages of size 2097152 reserved, but no mounted hugetlbfs found for that
    size
EAL: Probing VFIO support...
EAL: Probe PCI driver: hpac_sdfec_pmd (10ee:903f) device: 0000:81:00.0 (socket 1)
qdma_master_resource_create: New master resource created at 0

#### FPGA RELEASE Version: a20000

#### FPGA BUILD Date     : 1092021

#### Number of Encoders : 1e

#### Number of Decoders : 1d

#########################
 Feature support list
#########################

CRC Attachment   :Supported

Ratematching     :Supported

De-Ratematching  :Supported

HARQ Feature     :Supported

CRC Detach       :Supported

EAL: No legacy callbacks, legacy socket not created
```

XILINX T2 TELCO ACCELERATOR CARD

```
EAL: Detected 32 lcore(s)
EAL: Detected 2 NUMA nodes
EAL: Detected shared linkage of DPDK
EAL: Multi-process socket /var/run/dpdk/rte/mp_socket
EAL: Selected IOVA mode 'PA'
EAL: 942 hugepages of size 2097152 reserved, but no mounted hugetlbfs found for that
    size
EAL: Probing VFIO support...
EAL: Probe PCI driver: net_qdma (10ee:9048) device: 0000:21:00.0 (socket 0)
PMD: QDMA PMD VERSION: 2020.2.1
ACL_LDPC_Lib:[vr2p0_int_96 (built_against)DPDK_20.11.3 (date)'Jul 11 2022 20:44:21']
ACL_DPDK_Lib:[vr2p0_int_96 (built_against)DPDK_20.11.3 (date)'Jul 11 2022 20:44:18']
No Error
Devices: 1
-----------------------------------------------------
Platform String       : '09af1123c71a7fa8'
-----------------------------------------------------
Max Descriptor Size   : 16384
Number PCIe lanes     : 11
Device A Frequency    : 201 MHz
Device B Frequency    : 201 MHz
```

```
Device C Frequency     : 250 MHz
Device D Frequency     : 125 MHz
-----------------------------------------------------
Number of Connected Devices     : 2
-----------------------------------------------------
Device[0]: QID:0Number of Queues4
----------------------------------------
__DeviceDescription__
m_deviceID            : 1
m_deviceType          : SD_FEC_Decoder
m_deviceVersion       : 2
m_deviceParallelism   : 16
----------------------------------------
__InstanceDescription__
m_totalInstances      : 6
m_activeInstances     : 6
Device:0 Type: 7
-----------------------------------------------------
Device[1]: QID:4Number of Queues4
----------------------------------------
__DeviceDescription__
m_deviceID            : 2
m_deviceType          : SD_FEC_Encoder
m_deviceVersion       : 2
m_deviceParallelism   : 128
----------------------------------------
__InstanceDescription__
m_totalInstances      : 2
m_activeInstances     : 2
Device:1 Type: 6
-----------------------------------------------------
Device[2]: QID:8Number of Queues0
-----------------------------------------------------
Device[3]: QID:8Number of Queues0
-----------------------------------------------------
ACL FPGA:[ 6SD-P16-201MHz_2SE-P128-201MHz_09af1123c71a7fa8 ]
QDMA Chaining support: SUPPORTED
QDMA C2H Burst Size support: SUPPORTED
ACL : PCIe LinkSpeed 4 : Width 8
ACL BBDEV PMD : [12:54:37:Jan  9 2023]
2101:q_setup() setup device queue 0 with ID ranges 0 - 255 (256) on QDMA Q 0
2101:q_setup() setup device queue 1 with ID ranges 0 - 255 (256) on QDMA Q 1
2101:q_setup() setup device queue 2 with ID ranges 0 - 255 (256) on QDMA Q 2
2101:q_setup() setup device queue 3 with ID ranges 0 - 255 (256) on QDMA Q 3
```

6.4 Appendix IV - Quectel RM500Q-GL (Release 15) setup

Commands to run from the terminal of the host machine:

```
# correct interface has to be used (wwan0, wwan1, ....)
sudo ./quectel-CM -s oai.ipv4 -4 -i wwan0
sudo minicom -D /dev/ttyUSB2
AT+QMBNCFG="Select","ROW_Commercial"
AT+QMBNCFG="AutoSel",0
```

```
AT+QNWPREFCFG="mode_pref",NR5G
AT+QNWPREFCFG= "nr5g_band",77
AT+QNWPRECFG="nr5g_band"          # check band
AT+ CGDCONT?                      # check APN/DNN
AT+CFUN=0                         # turn OFF (airplane mode ON)
AT+CFUN=1                         # turn ON (airplaine mode OFF)
```

6.5 Appendix V - Library for T1 and T2 implementation

Following modification was done in the *openairinterface/CMakeList.txt* in order to create library for T1 and T2 implementation linking necessary shared object files from the DPDK library.

```
# LDPC offload library
############################################################

add_boolean_option(BUILD_T1_OFFLOAD True "Build support for LDPC Offload to T1 library")
if(BUILD_T1_OFFLOAD)
  find_library(T1 NAME rte_pmd_hpac_sdfec_pmd REQUIRED)
  set(ENV{PKG_CONFIG_PATH} "$ENV{PKG_CONFIG_PATH}:/usr/local/lib64/pkgconfig/")
  pkg_search_module(LIBDPDK_T1 REQUIRED libdpdk=20.05.0)
  add_library(ldpc_offload_T1 MODULE ${OPENAIR1_DIR}/PHY/CODING/nrLDPC_decoder/nrLDPC_decoder_offload.c)
  set_target_properties(ldpc_offload_T1 PROPERTIES COMPILE_FLAGS "-DALLOW_EXPERIMENTAL_API")
  target_compile_options(ldpc_offload_T1 PRIVATE ${LIBDPDK_T1_CFLAGS})
  target_link_libraries(ldpc_offload_T1 ${LIBDPDK_T1_LDFLAGS} rte_pmd_hpac_sdfec_pmd "-Wl,-rpath
  /usr/local/lib64")
endif ()

add_boolean_option(BUILD_T2_OFFLOAD True "Build support for LDPC Offload to T2 library")
if(BUILD_T2_OFFLOAD)
  find_library(T2 NAME rte_baseband_accl_ldpc REQUIRED)
  set(ENV{PKG_CONFIG_PATH} "/usr/lib64/accelercomm/dpdklibs/pkgconfig/")
  pkg_search_module(LIBDPDK_T2 REQUIRED libdpdk=20.11.3)
  add_library(ldpc_offload_T2 MODULE ${OPENAIR1_DIR}/PHY/CODING/nrLDPC_decoder/nrLDPC_decoder_offload.c)
  set_target_properties(ldpc_offload_T2 PROPERTIES COMPILE_FLAGS "-DALLOW_EXPERIMENTAL_API")
  target_compile_options(ldpc_offload_T2 PRIVATE ${LIBDPDK_T2_CFLAGS})
  target_link_libraries(ldpc_offload_T2 ${LIBDPDK_T2_LDFLAGS} rte_baseband_accl_ldpc "-Wl,-rpath /usr/lib64
  accelercomm/dpdklibs")
endif ()

############################################################
if(BUILD_T1_OFFLOAD)
add_dependencies( nr-softmodem ldpc_offload_T1)
endif ()

```

```
33 if(BUILD_T2_OFFLOAD)
34 add_dependencies( nr-softmodem ldpc_offload_T2)
35 endif ()
36
37 if(BUILD_T1_OFFLOAD)
38 add_dependencies( nr_ulsim ldpc_offload_T1)
39 endif ()
40
41 if(BUILD_T2_OFFLOAD)
42 add_dependencies( nr_ulsim ldpc_offload_T2)
43 endif ()
```

6.6 Appendix VI - nr_ulsim sample output

```
****************************************
SNR 30.000000: n_errors (0/1000,0/0,0/0,0/0) (negative CRC), false_positive 0/1000,
    errors_scrambling (808/150696000,0/0,0/0,0/0)

SNR 30.000000: Channel BLER (0.000000e+00,-nan,-nan,-nan Channel BER (5.361788e-06,-nan
    ,-nan,-nan) Avg round 1.00, Eff Rate 50184.0000 bits/slot, Eff Throughput 100.00,
    TBS 50184 bits/slot
DMRS-PUSCH delay estimation: min 0, max 0, average 0.000000
****************************************

Total PHY proc rx                          363.60 us (1000 trials)
 Statistics std=3.65, median=0.00, q1=0.00, q3=0.00 us (on 0 trials)
|__ RX PUSCH time                         135.98 us (1000 trials)
    |__ ULSCH channel estimation time       42.55 us (1000 trials)
    |__ ULSCH PTRS Processing time           0.00 us (   0 trials)
    |__ ULSCH rbs extraction time            2.19 us (12000 trials)
    |__ ULSCH channel compensation time      2.76 us (12000 trials)
    |__ ULSCH mrc computation                0.01 us (12000 trials)
    |__ ULSCH llr computation                2.24 us (12000 trials)
|__ ULSCH unscrambling                      18.44 us (1000 trials)
|__ ULSCH total decoding time              199.34 us (1000 trials)
|__ ULSCH total encoding time               92.09 us (1000 trials)
    |__ ULSCH segmentation time              4.77 us (1000 trials)
    |__ ULSCH LDPC encoder time             31.83 us (1000 trials)
    |__ ULSCH rate-matching time             0.65 us (6000 trials)
    |__ ULSCH interleaving time              8.33 us (6000 trials)
|__ RX SRS time                             0.00 us (   0 trials)
    |__ Generate SRS sequence time           0.00 us (   0 trials)
    |__ Get SRS signal time                  0.00 us (   0 trials)
    |__ SRS channel estimation time          0.00 us (   0 trials)
    |__ SRS timing advance estimation time   0.00 us (   0 trials)
    |__ SRS report TLV build time            0.00 us (   0 trials)
        |__ SRS beam report build time       0.00 us (   0 trials)
        |__ SRS IQ matrix build time         0.00 us (   0 trials)

************
PUSCH test OK
************
```

```
Num RB: 273
Num symbols: 12
MCS: 10
DMRS config type: 0
DMRS add pos: 0
PUSCH mapping type: 1
DMRS length: 1
DMRS CDM gr w/o data: 1
```