GPU-based LDPC Decoding for vRAN Systems in 5G and Beyond

Chance Tarver*†, Matthew Tonnemacher†, Hao Chen†, Jianzhong (Charlie) Zhang†, and Joseph R. Cavallaro*
*Dept. of Electrical and Computer Engineering, Rice University, Houston, TX, USA
†Standards & 5G Mobility Innovation Lab, Samsung Research America, Plano, TX, USA

***Abstract*—Next-generation virtual radio access n (vRAN) will benefit from the flexibility provided by virtua in proposed Cloud-RAN configurations. These systems for beyond may consist of commodity hardware such as GPUs centers with multiple connected base stations (gNBs) flex ceiving allocated resources depending on time-varying, r demands. In this paper, parallel reconfigurable algorith architectures for channel decoding are proposed. In pa flexible rate and block length LDPC decoders for the ne (NR) physical layer on GPU are characterized. We im these GPU decoders using reduced word lengths of 8 represent the log-likelihood ratios during decoding, and w multiple GPU streams to process multiple blocks of co in parallel. These techniques allow our implementation t the device transfer overhead and achieve the low-latency throughput targets for 5G and beyond. Moreover, we i our decoder into the Open Air Interface (OAI) NR s stack to investigate virtualization capabilities when containerizing vRAN functionality such as the LDPC decoder.**

***Index Terms*—LDPC, SDR, GPU, OAI, vRAN**

I. Introduction

A key challenge in 5G and beyond is the flexibility necessary for the radio access network (RAN) to be able to support the many possible applications ranging from 4K video streaming, which requires high data rates, to high-precision remote surgery, which requires ultra-low latency. The various applications are typically described to fit into one of the following categories: enhanced mobile broadband (eMBB), ultra-reliable low-latency communications (URLLC), and massive machine-type communications (mMTC). These descriptions are used to encapsulate the IMT-2020 5G requirements. eMBB is designed to support peak downlink data throughputs of 20 Gbps. URLLC is designed to support end-to-end latencies of 1 ms. The mMTC category is designed to be able to support connection densities of 1 million devices per km^2 [1]. These targets are 50 to 100$\times$ above the targets for 4G. To be able to support the next 100$\times$ improvement for beyond 5G, the systems will need to embrace the flexibility.

A. The need for software-based, vRAN systems

Software-based solutions excel at flexibility, and the development of standards-compliant software-defined radio (SDR) projects is an emerging theme that highlights this. For example, Open Air Interface (OAI) is a software-defined platform for 4G and 5G systems [2]. Completely on commodity CPUs, the entire 4G base station and core network can run and support multiple user equipments (UEs). This sort of software-based approach can be rapidly developed and tested and provides an opportunity for researchers to experiment with new algorithms for any task in the stack. Software is a more natural fit to possible future deployments based on cloud radio access network (C-RAN). In new C-RAN systems, baseband processing may be done at a central data center on enterprise-grade servers. More recently, virtualized radio access networks (vRAN) systems have also been proposed [3], [4] that virtualize the baseband, decoupling the hardware resources from processing tasks to allow for more dynamic orchestration of resources based on real-time network demands. This sort of consolidation, centralization, and virtualization of baseband tasks allows for the flexibility to support multiple standards and provide easier deployment of future standards. C-RAN/vRAN systems can also be cheaper in that processing can be done on enterprise grade servers and other commercially available off-the-shelf (COTS) equipment that is already widely available instead of on expensive, highly specialized equipment. Moreover, centralized processing like what is seen in C-RAN/vRAN is likely necessary for advanced algorithms such as coordinated multipoint (CoMP) and cell-free systems to be possible. However, performance is typically a concern when considering a software based solution.

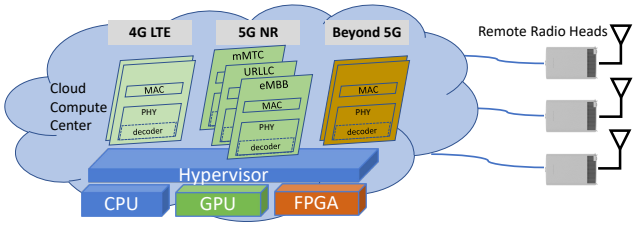

Fig. 1. Illustration of the flexible, multi-standard vRAN concept.

B. GPU-based high-performance baseband processing

This gap can be filled by graphics processing units (GPUs) which provide high performance while maintaining flexibility. GPUs have become commonplace in many high-performance computing fields such as deep learning. They have also been considered for many baseband processing tasks for 4G and 5G implementations. For example, in [5] GPUs are used for detection and beamforming in a multi-user (MU) multiple-input multiple-output (MIMO) base station. Additionally, GPUs can

This work was supported by Samsung Research America and US NSF grants CNS-1717218 and CNS-1827940 for the PAWR Platform POWDER-RENEW: A Platform for Open Wireless Data-driven Experimental Research with Massive MIMO Capabilities.

978-1-7281-3320-1/20/$31.00 ©2020 IEEE

be containerized via tools like NVIDIA-Docker to run on vRAN systems.

1) LDPC for 5G: One particular task that is ideal for implementations on a GPU in 5G is the low-density parity-check (LDPC) error correction codes (ECC). ECCs add redundancy to wirelessly transmitted bits so that the receiver can detect and correct errors. LDPC was chosen to replace turbo codes from 4G for the data traffic in 5G [6]. Although LDPC codes have near capacity-achieving decoding performance, the decoding complexity is extreme. This puts a challenge for the 5G next-generation NodeBs (gNBs), which need to potentially decode many codewords (CWs) from multiple users at high data rates.

GPUs are a great platform for LDPC because LDPC decoding can cleanly map to the single-instruction multiple-thread (SIMT) parallel architecture of GPUs. Decoding is an iterative process where log-likelihood ratio (LLR) messages are exchanged to correct any bits. Messages can easily be computed in parallel on the thousands of processing elements such as the compute unified device architecture (CUDA) cores found in NVIDIA GPUs. Moreover, being software-based, a GPU project can be rapidly developed and deployed. Scaling to support more users and higher throughput can also be trivial in that multiple GPUs can work together in parallel. This lends itself to being a natural fit for data-center based C-RAN systems. Moreover, as GPU hardware development continues to progress rapidly, an operator could see throughput and latency improvements by upgrading devices over time.

A new challenge in 5G for any LDPC implementation that is not found in other radio access technologies (RATs) that use LDPC such as Wi-Fi is the flexibility necessary. In eMBB a high decoding throughput is necessary. For URLLC, being able to quickly decode a CW will likely be one of the major bottlenecks in round-trip time. For mMTC, it will be difficult to decode CWs from many users simultaneously. However, the scalability and reconfigurability of the GPU make this possible. At runtime, we can reconfigure the GPU to change configuration to prioritize latency or throughput.

2) Related works: GPUs have been used for a variety of ECC. For example, in [7] they are used for turbo codes found in 4G. LDPC codes were implemented in [8] and [9] with the latter work also deploying multiple GPUs together to increase total throughput. However, these works target Wi-Fi and WiMAX. There are few reported architectures and software realizations for the recent LDPC code for 5G New Radio (NR) physical layers. In [10] LDPC is implemented on a Xeon processor using AVX instructions. While the latency results are exceptional at 31 μs, the throughput is limited to 270 Mbps. Xilinx and Intel offer FPGA based solutions, but performance benchmarks are not publicly available.

In this work, we fill the gap and provide a GPU solution for 5G NR LDPC targeting vRAN deployments. Our main contributions are the development of a GPU-based solution for LDPC with support for the 5G NR standard with the flexibility necessary to become a research platform for beyond 5G. We present throughput and latency numbers across multiple GPU devices. In our implementation, we present multiple optimizations to improve performance such as quantization to shorter word lengths to save time on data transfers. We also develop our GPU software as a library to run in a container and use it in the "develop-nr" git branch of OAI [11] as part of a vRAN testbed.

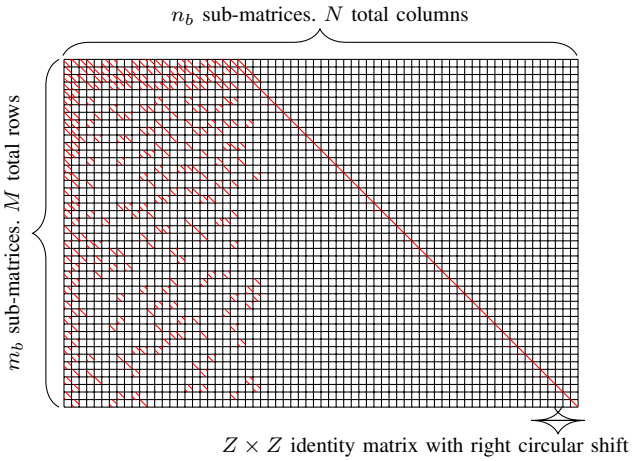

Fig. 2. Illustration of a QC-LDPC parity check matrix, $\mathbf{H}$. Red slashes correspond to ones in the sub-matrices. This illustration is modeled after base graph 1 in 5G NR with $m_b = 46$ and $n_b = 68$ and set index $i_{LS} = 1$.

II. LDPC Overview

A binary LDPC code is defined by an $M \times N$ sparse parity check matrix, $\mathbf{H}$, and an $N \times 1$ CW vector, $\mathbf{x}$. For the vector $\mathbf{x}$ to be a valid CW, $\mathbf{Hx} = \mathbf{0}$ must be true. In a communications system, we only transmit valid CWs. If we receive a CW that is not valid, we assume that elements of $\mathbf{x}$ are incorrect and attempt to find the valid CW that was most likely sent.

One common family of codes is quasi-cyclic (QC)-LDPC codes. Here, each parity check matrix is constructed from an $m_b \times n_b$ base matrix which is an array of shifted identity matrices of size Z known as the lifting factor. An example QC-LDPC parity check matrix is shown in Fig. 2.

From the parity check matrix, a bipartite Tanner graph can be constructed with rows in $\mathbf{H}$ corresponding to check nodes (CNs), columns corresponding to variable nodes (VNs), and $H_{ji} = 1$ corresponding to an edge connecting CN j and VN i. The main idea behind most LDPC decoders is that we can send messages back-and-forth from VNs to CNs to find and correct any incorrect bits. Decoding is typically performed via iterative message passing on the Tanner graph between CNs and VNs. The calculation of a posteriori probabilities (APP) is done through a sum-product algorithm (SPA). However, a common simplification that we implement in this work is the min-sum algorithm (MSA). When implemented with a scaling parameter or an offset, it offers low complexity with minor loss in bit error rate (BER) performance. For complete details on SPA, MSA, and other LDPC algorithm details see [12].

A. LDPC for 5G NR

The LDPC encoding and decoding specification for 5G NR is defined in TS.38.212 of Rel. 15. In 5G, LDPC is used for the downlink shared channel (DL-SCH), uplink shared channel (UL-SCH), and paging channel (PCH) transport channels, which mostly carry data. In contrast, polar codes are used for the broadcast channel (BCH), which mostly carries system information. In the below section, we present a brief overview

of the relevant parts of the specification for reference. See [13] for the complete details and [14] for a discussion on the design of LDPC for 5G.

There are two QC base graphs (BGs) used to derive the parity check matrices. In general, BG 1 is used for larger, higher rate data while BG 2 is reserved for short payload sizes and code rates less than 0.25. For BG 1 there are $m_b = 46$ rows and $n_b = 68$ columns while BG 2 has $m_b = 42$ and $n_b = 52$. The maximum length of a sequence of bits in a code block, K_{cb}, to be encoded is 8448 bits for BG 1 and 3840 bits for BG 2. Each support a variety of lifting factors, Z, from 2 up to 384. This makes the largest possible 5G NR code an irregular (25344, 8448) rate 1/3 code which uses BG 1. Fig. 2 shows the parity check matrix corresponding to this code.

To better understand LDPC for 5G, we discuss the encoding procedure below. Given a sequence of input bits to be encoded, represented by the vector $\mathbf{a}$, we start by appending a cyclic redundancy check (CRC) to make a new vector, $\mathbf{b}$. For $\mathbf{b}$ with length B and the code rate, R, provided by the modulation and coding scheme (MCS), a BG is chosen according to the above rules. If $B > K_{cb}$ for the selected BG then the input sequence is segmented into multiple code blocks, each with their own CRC forming a new sequence, $\mathbf{c}$. The size of the individual code blocks is used to calculate the appropriate lifting factor, Z. Based on the lifting factor, a table lookup provides the set index, i_{LS}, which is used to construct the final parity check matrix, $\mathbf{H}$. There are 8 set indices in total leading to eight possible variations of each BG. In the process of encoding, a vector in the nullspace of $\mathbf{H}$ is constructed in the form of $\begin{bmatrix}\mathbf{c} & \mathbf{w}\end{bmatrix}^T$ where $\mathbf{w}$ is a vector of parity bits to be calculated with length $N+2Z-K$ where $N = 66Z$. After encoding, the CW bits corresponding to the indices after $2Z$ undergo rate matching via bit selection and interleaving. If segmentation took place, the code blocks are concatenated together before modulation and transmission.

A key insight in the above exploration of the 5G standard is that the standard is broad and decoders should support 51 different lifting factors from $Z = 2$ to 384 giving rise to 8 reconfigurations of 2 separate parity check base graphs, all derived from a wide range of possible code rates and block lengths. Supporting all possible valid configurations is challenging for hardware-based decoding implementations. Moreover, being able to meet the various latency and throughput targets based on the real-time demands is another serious challenge. In contrast the flexible, software-based nature of the GPU makes decoding straightforward without sacrificing performance, as we will demonstrate in the remainder of the paper.

III. Implementation Details

A. GPU Overview

A GPU consists of an array of streaming multiprocessors (SMs) which each often contain 32 or 64 vector processing lanes. NVIDIA provides C++ language extensions, known as CUDA, to write highly parallel applications that target these devices. The developer writes kernels that will utilize a certain number of blocks and threads which are roughly a programming abstraction of the SMs and CUDA cores. There are multiple tiers of memory on the GPU. Global memory is typically a GDDR based memory that is available to all kernels over the lifetime of the application. Threads in a block all execute on the same SM and can have a shared memory for the lifetime of the kernel. Each thread also has its own local registers. There is also a constant memory that is globally available to all threads, but it is read only. The global memory is typically the slowest, so it is desirable to put data in constant and shared memory whenever possible.

B. Mapping of LDPC to GPU

To efficiently map the LDPC decoding to a GPU, we create two main kernels: a CN kernel and a VN kernel. Multiple CWs are decoded in parallel via two main mechanisms. First, we pack multiple CWs into what we refer to as a macro-codeword (MCW). These CWs are evaluated concurrently within the same CUDA block. Secondly, we copy multiple MCW to the GPU in a batched transfer to execute on separate blocks. Each block will work on 1 row of the base graph with the individual threads working on the subrows after applying the lifting factor. For more details on the architecture see [9].

1) Check node processing: In this kernel, each thread operates on a row of $\mathbf{H}$. Using the two-min algorithm for the scaled MSA, messages are computed for each connected variable node. The messages are stored in global memory.

2) Variable node processing: For the VN kernel, each thread corresponds with a column of $\mathbf{H}$ and computes its APP LLR based on the messages received from the CNs. The updated LLR is stored in global memory. In the final decoding iteration, a hard decision is made.

C. Optimizations Strategies:

1) Reduced word length: The main optimization that we offer in this work compared to previous GPU-based solutions is the reduction of the word lengths down to the 8 bit char format. Using as few as six bits is known to be adequate [15] and is common in many FPGA and ASIC implementations. We use this reduced word length to save up to 4x on data transfer times between host and device and for global memory accesses while casting to and from floats for each thread's computation.

2) Packing final hard decision: In many applications a hard decision is made, and then each CW bit only needs to be represented by a single bit. Past works often did not take advantage of this and stored the result with unnecessary data types. We implement a bit packing kernel before the final transfer to the host and only copy the final information bits to the host instead of the entire decoded CW.

3) Streams: To further alleviate the memory transfer between the host and device, we utilize multiple CUDA streams. Each stream acts like a separate process on a CPU application to be scheduled and executed in parallel. This allows for one stream to be computing while another may be copying new data to the GPU. This helps to ensure that the computational resources are fully utilized most of the time.

4) Caching of $\mathbf{H}$: Row-major and column-major compact versions of the parity check matrices are loaded by the host onto the GPU's constant memory. The row or column major version is used by the check or variable node, respectively, to allow for fast, coalesced memory accesses.

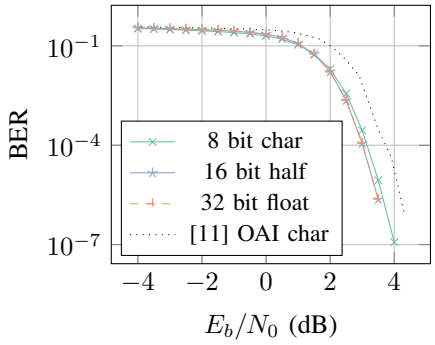

Fig. 3. BER curve comparing performance when using various data storage precisions on GPU. The (25344, 8448) code was used with 5 iterations.

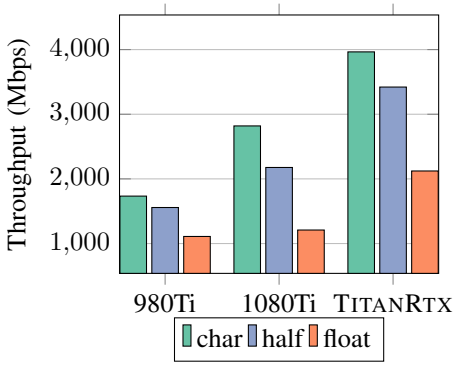

Fig. 4. Throughput for the considered datatypes on 3 different GPUs.

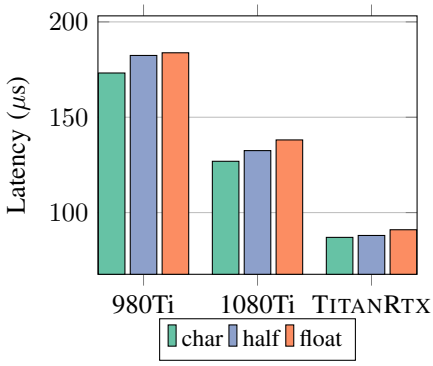

Fig. 5. Latency for the considered datatypes on 3 different GPUs.

IV. Results

a) Performance: We begin by evaluating the BER performance as a function of energy per bit to noise power spectral density (E_b/N_0). This tests the LDPC decoding performance and shows the performance for the various word lengths, as shown in Fig. 3. The float and half precision data types offer nearly identical performance while there is a minor increase in BER when quantizing down to the 8-bit char for data storage. For comparison, we also include the OAI implementation from [11] which performs decoding on CPU using char formats throughout the data storage and computation.

b) Throughput: For three different GPUs, we tested the maximum throughput. For this test, we configured the GPU to use 6 streams, 20 MCWs per transfer, and 2 CW per MCWs. This configuration helped to ensure the GPU was always full and the effect of the PCIe transfers to and from the CPU and GPU was reduced. In Fig. 4, we see that with each generation of GPU performance increases as the number of CUDA cores also increases. The best performance was for the char data type on the TITAN RTX with 3964 Mbps of decoding throughput, including the transport time to-and-from the CPU and GPU. For this configuration, the latency is increased and on the order of 700 μs, though this can be acceptable for eMBB applications.

c) Latency: The GPU can be reconfigured to target URLLC to reduce the decoding times. To achieve lower latency, we can reduce the number of MCWs, CWs per MCW, and streams. In cases where there is a good signal-to-noise ratio (SNR), the number of decoding iterations can be reduced as well or early termination can be applied. When testing across multiple GPUs as shown in Fig. 5, we see that the TITAN RTX is able to achieve latencies as low as 87 μs, including the transport time to-and-from the CPU and GPU. For this configuration, the throughput is 290 Mbps.

d) Comparison to other works: Currently, there are few published results available publicly for 5G LDPC decoders. We compare and summarize them in Table I. In these works, it is not always clear what the chosen code and rate are. For each category, we report the best metric reported by the work. Not all works reported both the lowest latency and the highest throughput, which may occur for different code word configurations.

TABLE I
COMPARISON OF NR LDPC DECODERS

	Platform	Latency (μs)	Throughput (Mbps)
This work	TITAN RTX	87	3964
[16]	Intel i7-6700K	177.7 [1]	30
[10]	Xeon Gold 6154	31.08	271.80
[17]	i7-4770	240	NA
[18]	FPGA	NA	574

V. OAI Integration

Our GPU software is designed to be compiled into a generic library that can accelerate SDR platforms for a vRAN. We evaluated the performance of our GPU decoder in two scenarios. First we tested with their standalone ldpctest. Then we tested with their larger RFSimulator, which includes the entire 5G NR protocol stack. The ldpctest evaluates BER performance and latency. When testing OAI's current C-based LDPC decoder in the ldpctest, the best decoding latency achieved was 178 μs In contrast, our GPU-based decoder had a latency of 87 μs, a 51% reduction in latency. When testing with the full 5G stack, we were able to integrate with OAI and verify that the CRC passed as expected.

VI. Conclusions

In this work, we presented a flexible, 5G NR compliant LDPC decoder for GPU that was targeted for enabling 5G and beyond-5G vRAN. The GPU-based solution, being software based, is shown to have the flexibility needed to support all possible configurations of LDPC for 5G and beyond. Moreover, by adjusting system parameters, the GPU can be configured to target codeword throughput or latency depending on the needs of the base station. For future work, we plan on extending the vRAN testbed to include more UEs and gNBs to explore hardware orchestration.

[1]The latency metric was measured locally using the "ldpctest" executable from OAI's "develop-nr" git branch [11].

REFERENCES

[1] International Telecommunication Union, "Minimum requirements related to technical performance for IMT-2020 radio interface(s)," Nov. 2017. [Online]. Available: https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2410-2017-PDF-E.pdf

[2] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "Openairinterface: A flexible platform for 5G research," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 33–38, Oct. 2014. [Online]. Available: http://doi.acm.org/10.1145/2677046.2677053

[3] Wind River, "vRAN: The Next Step in Network Transformation," 11 2017. [Online]. Available: https://builders.intel.com/docs/networkbuilders/vran-the-next-step-in-network-transformation.pdf

[4] A. Garcia-Saavedra, X. Costa-Perez, D. J. Leith, and G. Iosifidis, "Fluidran: Optimized vRAN/MEC orchestration," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, April 2018, pp. 2366–2374.

[5] K. Li, "Decentralized baseband processing for massive MU-MIMO systems," Ph.D. dissertation, Dept. of Elect. and Comput. Eng., Rice University, 2019.

[6] D. Hui, S. Sandberg, Y. Blankenship, M. Andersson, and L. Grosjean, "Channel coding in 5G new radio: A tutorial overview and performance comparison with 4G LTE," *IEEE Vehicular Technology Magazine*, vol. 13, no. 4, pp. 60–69, Dec 2018.

[7] M. Wu, G. Wang, B. Yin, C. Studer, and J. R. Cavallaro, "HSPA;/LTE-A turbo decoder on GPU and multicore CPU," in *2013 Asilomar Conference on Signals, Systems and Computers*, Nov 2013, pp. 824–828.

[8] G. Falcao, L. Sousa, and V. Silva, "Massively LDPC decoding on multicore architectures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 2, pp. 309–322, Feb 2011.

[9] G. Wang, M. Wu, B. Yin, and J. R. Cavallaro, "High throughput low latency LDPC decoding on GPU for SDR systems," in *2013 IEEE Global Conference on Signal and Information Processing*, Dec 2013, pp. 1258–1261.

[10] Y. Xu, W. Wang, Z. Xu, and X. Gao, "AVX-512 based software decoding for 5G LDPC codes," presented at the 2019 IEEE International Workshop on Signal Processing Systems (SiPS), Nanjing, China, 2019.

[11] "OAI develop-nr gitlab repository," Nov. 2019, commit: 517c8585b3662cd38edaf7cb542fed5eca3d89e5. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/tree/develop-nr

[12] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1288–1299, Aug 2005.

[13] 3GPP, "NR; Multiplexing and channel coding," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 36.212, 09 2019, version 15.7.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3214

[14] T. Richardson and S. Kudekar, "Design of low-density parity check codes for 5G new radio," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 28–34, March 2018.

[15] T. Zhang, Z. Wang, and K. K. Parhi, "On finite precision implementation of low density parity check codes decoder," in *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No.01CH37196)*, vol. 4, May 2001, pp. 202–205 vol. 4.

[16] F. Kaltenberger, "5G-NR-development-and-releases," 9 2019. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/5g-nr-development-and-releases

[17] W. Ji, Z. Wu, K. Zheng, L. Zhao, and Y. Liu, "Design and implementation of a 5G NR system based on LDPC in open source SDR," in *2018 IEEE Globecom Workshops (GC Wkshps)*, Dec 2018, pp. 1–6.

[18] Creonic, "5G LDPC decoder and HARQ buffers product brief," 2019. [Online]. Available: https://www.creonic.com/wp-content/uploads/PB_Creonic_5G_RL15_Decoder.pdf