International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 05 | May - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

# GPU-Based, LDPC Decoding for 5G and Beyond

**Mohamed Madni[1], DR. Dilip R[2]**

**Department of Electronics and Communication Engineering**

## ABSTRACT

The New 5G Radio (NR) includes low-density parity-checking (LDPC) codes as errors correction code (ECC) for the data channel. For LDPC codes for low performance, near Shannon power, bit error rate (BER), they are also mathematically complex in physical representation applications. Moreover,5G LDPC has additional challenges not found in previous LDPC implementations, such as Wi-Fi. LDPC and.The specification in 5G includes many new configurations to support different rates, block sizes and use cases.5G is also targeted to support increased usage and lower latency. For this new, more flexible versionStandard, traditional hardware-based solutions in FGPA and ASIC may struggle to support all issues with.It can be costly in scale. The software solution can seamlessly support all possible reconfigurations however.Struggles in the workplace. This case shows its high throughput and low latencyGraphics Processing Units (GPUs) for LDPC decoding as an alternative to FPGA and ASIC decoders, .To effectively deliver the high performance required while maintaining the advantages of the based software answer. In particular, we focus on how by changing the parallelization scheme for GPU kernel mappingto blocks, we can use more GPU cores to compute a codeword faster to aim for low-latency, or We can use core to work on multiple codewords at the same time to aim at high throughput applications.This flexibility is particularly useful for virtualized radio- access networks(vRAN), which are the next generationa technology that is expected to become more prominent in the coming years. Hardware on vRAN audit resources will be released from specific audit projects in the RAN With virtualization, it allows for load balancing and other benefits.

## I.INTRODUCTION

A key challenge in 5G and beyond is resility to a requirement for a radio area network (RAN). 5G can support many possible applications from streaming 4K video, which requires higher data rates, to precision distal surgery, which requires minimal surgery.

Roles are often discussed fits in one of the following categories: Enhanced MobileBroadband (eMBB), reliable low- cost connecttions (URLLC), and device type correlation size(mMTC) is performed. These reasons have been analysed.IMT-2020 Requirements for 5G. Peak downlink data throughput of 20 Gbps and peak uplink data throughput are the goals of the eMBB architecture.10 Gbps bandwidth is available. The one-millisecond end-to-end delay is supported by the design of the URLLC. One million devices per square kilometre is the target connection density of the mMTC section. In 4G objectives, objectives range from 50 to 100×. To end with 5G, followed by a 100× improvement in connectivity.

### A.GPU-BASED HIGH PERFORMANCE BASEBAND PROCESSING

Change is needed in 5G. However, hardware-based accelerators can find it challenging to provide computationally efficient for every scenario and common approach. To close the flexibility/performance gap and preserve software speed for multipurpose cores, additional GPUs can be added for high parallelization performance GPUs are now widely used in high-performance computing environments, including learning including deep. Several baseband applications have also already been

considered for 4G and 5G services. For example, in a multi-user (MU) multiple input multiple output (MIMO) base station, a GPU is used for beam generation detection. Additionally, frameworks such as NVIDIA-Docker can be used to run GPUs in a VRAN configuration. Additionally, more recent models offer a multi-instance GPU (MIG), which enables virtualization of a single GPU to multiple GPUs.

1)LDPC FOR 5G

DPC ECC is a unique technology optimized for GPU applications in 5G.

To enable error detection and correction at the receiver, ECC adds redundancy to the bits transmitted wirelessly. LDPC was chosen to replace the turbo code in 4G for data traffic in 5G [3]. Although the decoding performance of the LDPC code is similar to that of the capacity seeker, the decoding complexity increases. Higher data speeds for 5G next-generation NodeBs (gNBs) may require more codewords (CWs) defined for more users, making LDPC decoding computationally demanding. GPUs are a good platform for LDPC. Since the single-instruction-multithread (SIMT) parallel GPU architecture maps to pure LDPC decoding methods, this combination works. Decoding is an iterative process in which each bit is changed by changing the log-likelihood ratio (LLR) message. Compute unified device architecture (CUDA) cores found on NVIDIA GPUs simplify computing messages in parallel to thousands of processing components, enabling GPU projects to be measured and executed far faster than other software projects Scale up to enable large codewords and throughputs and multiple GPUs can run in parallel. This can also be easy. Cloud Radio Access Network (C-RAN) solutions based on data centres are perfect due to its scalability. Furthermore, a manufacturer can quickly upgrade GPU hardware and develop new devices over time without incurring additional development costs, and any LDPC deployed in 5G will face additional challenges of throughput and latency improvements without a radio ambient new technology (RATs). and required for intended use cases, such as Wi-Fi. Probably, one of the main constraints to travel time recovery will be the speed at which CW decoding for URLLC can be performed. In the case of mMTC, it would be difficult to distinguish between CWs among multiple users at once.

2)RELATED APPLICATIONS:

ECC using GPU. It is used, for example, in [4] for 4G turbo codes. The codes were LDPC. In this study, we focus on the modification of the method and present a GPU solution for 5G NR LDPC decoding. Our core contribution is to create a GPU-based LDPC solution that meets the 5G NR standard and has the flexibility needed to be a research platform for 5G and beyond. We also show latency and throughput statistics for several GPU devices. Several improvements are applied to improve the efficiency of our implementation, including quantization to shorter word lengths to reduce the data transmission. Additionally, we have integrated our GPU system into the vRAN testbed within the Open-Air Interface (OAI) "develop-nr" git branch and have developed it as a library for use in the repository [4]. The assembly from reference [17] is detailed in this issue. We extend the design and implementation of the GPU algorithm in this paper extension, present new findings including new 5G LDPC designs, deepen our analysis to better understand the trade-off between throughput and latency, present a time segmentation it provides about GPU-based LDPC decoding, and we provide more context. The remaining letters are as follows. We summarize LDPC in the second section, and its performance in 5G NR. The GPU configuration and each of our kernels will be described in more detail in Part III, as well as how we can decode the algorithm to the GPU. The results, as well as comparisons with other works, are presented in Section IV. Section V provides an overview of projects implementing software-defined radio design (SDR) systems. In the sixth step, we then wrap the paper.

## II.LDPC OVERVIEW

The M × N sparse parity test matrix, H, and the N×1 CW vector, x, define the two LDPC codes. For Hx = 0 to be a valid CW, the vector x must be real. We only

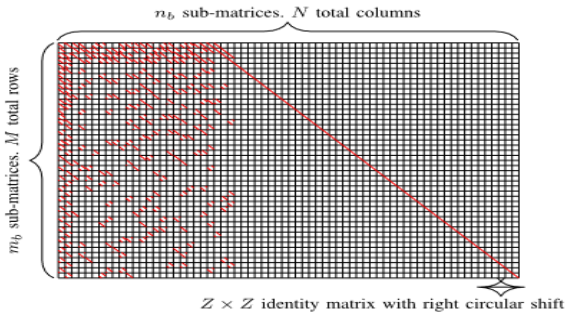
send valid CW within the network. If we get an invalid CW, we suspect that the content of x is wrong and look for the actual CW that should have been given. To generate valid codewords from a sequence of K information bits, s, encoding is usually performed using a generator matrix obtained from H . The ratio of the information bits to the codeword bits, R = K /NOT, determines the number r of the LDPC code. LDPC code is usually "systemic", meaning that the first K bits of a particular codeword contain information bits. This includes codes used in 5G. There are N − K bits remaining in the redundancy bits. One common signal family is the quasicyclic (QC)-LDPC code. In this case, an mb × nb basis matrix—which consists of several shift identification matrices of the form Z or lift factor—is used to generate each equality test matrix characterizing a combination of edges.

**A. DECODING ALGORITHMS**

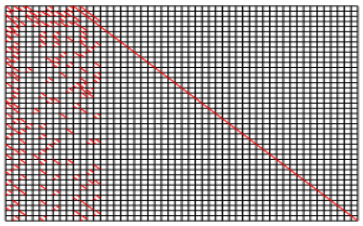
Most LDPC decoders operate on the basic principle that the transmission between VN and CN can be switched and corrected for odd bits. Repeated messages transmitting Tanner graphs between CN and VN are usually used for decoding. The sum-product algorithm (SPA) determines the posterior probability (APP). However, the min-sum algorithm (MSA) is a special characteristic that we often use in our work. By using offset or scaling parameters, BER provides reduced complexity with minimal operating costs. For a detailed description of SPA, MSA, and other LDPC algorithms, see [18]. We believe that instead of hardbits being passed from the demodulator to the decoder, the LLR values in the soft decision are passed in each case.

Minimum Sum Algorithm: MSA and its variants, such as scaled min-sum algorithm and offset min-sum algorithm, are weaknesses designed for SPA to improve decoding throughput with minimal loss in decoding performance In the basic approach, . decoding can only be done by comparing operations and changes. The algorithm starts with check node operation. The first messages for each CN are set to the first corresponding LLRs, Q(0) mn = L(0) n . Each CN continuously counts the number of messages to send to each associated VN, Rmn.

Eq.1.1.1.1.1.1.1.1.1.1. (2) denote these numbers where m and n are the indexes of check and variable nodes, respectively, Nm is the order of all variable nodes associated with check node m, and γ is the MSA scale factor:




(a) Base graph 1. Here, $m_b = 46$, $n_b = 68$, and the set index $i_{LS} = 1$.



(b) Base graph 2. Here, $m_b = 42$, $n_b = 52$, and the set index $i_{LS} = 0$.

FIGURE 1. Illustration of the QC-LDPC parity check base graphs, $H_{BG}$, used in 5G NR. Red slashes correspond to ones in the sub-matrices.

**B.LDPC FOR 5G NR**

3GPP TS.38.212 in Rel. 15, LDPC encoding and decoding requirements for 5G NR are explained. Most of the data including the 5G paging channel (PCH), the uplink shared channel (UL-SCH), the downlink shared channel (DL-SCH), and the transport channel are all transmitted using LDPC. In contrast, 5G uses polar codes for broadcast channels (BCH), most of which carry system data. For your convenience, we've included a quick summary of resources related to the presentation in the area below. See [19] for details; See [20] for a discussion on LDPC configuration for 5G. Homogeneity test matrices were constructed using two QC base graphs (BGs). BG 2, shown in Fig. 1(b), is shown for large payload and low coding rate because BG 1 = 46 rows and nb = 68 columns is less than 0.25 mb while BG 2 has mb = 42 and nb = 52 Overall. BG 1, seen in Fig. 1(a), is used for high data sizes. The maximum bit sequence length in encoding code part Kcb is 3840 bits for BG 2 and 8448 bits for BG 1. Z, using Z = 384, the longest conceivable 5G NR code is irregular (25344, 8448) rate 1/3 code utilizing BG 1. per Supports, Z. Various boosters from 2 to 384 Below is a discussion of the encoding method for LDPC for

5G. We start by applying a cyclic redundancy check (CRC) to vector a, in order to create a new vector, b. The BG is chosen according to the B length B and the coding rate, R, given by the Modulation and Coding Scheme (MCS). The input sequence is split into many code segments, each with its own CRC that generates an extra sequence, c, if B > Kcb for the chosen BG. 1) LDPC and HARQ. In 5G NR, hybrid automated repeat request (HARQ) is utilized to boost redundancy. The eNB must respond to uplink messages in less than 4 milliseconds, either with an ACK or a NACK indicating failure. This will serve as the basis for our calculation of the LDPC decoding delay in the results section. However, all uplink operations must go through the CRC check to update the appropriate HARQ configuration, which means that more time needs to be allocated for proper LDPC decoding and demodulation procedures.

### C. SUMMARY OF NOTES

In this section, we provide a list of all the symbols used throughout the work. Matrices such as H are capitalized and shown in bold. Uppercase letters in bold are used to represent scalar constants, such as K. Lowercase letters in bold are used as vectors, such as c. The l0 criterion is used to return the number of null elements of the vector, denoted by ‖.‖0. Table 1 lists all the markers and their significance.

## III.IMPLEMENTATION DETAILS

### A.GPU OVERVIEW

An array of stream multiprocessors (SMs), about 32 or 64 vector processing lenses each connected to the GPU. Applications that use an Nvidia C++ language extension called the GPU are designed to run in high parallel on these devices. To create many blocks and threads—about a programming abstraction involving SMs and CuDA cores—the developer creates the kernel. The GPU has multiple layers of memory. Typically, global memory is off-chip, GDDR-based memory that is accessible to all kernels for the duration of the program. During the duration of the kernel, all threads in the block share the same low latency, high throughput SRAM memory running on the same SM. This memory is called "shared". Each thread also has n 'own local registers. Furthermore, "permanent" read-only memory is globally accessible to all threads on the chip. Whenever possible, it is advisable to store data in shared memory because global memory is the slowest. Each new generation of GPU computing brought significant improvements. The 2020 generation of Nvidia Ampere introduced features such as Mellanox-based RDMA support, improved data processing capabilities previously limited to GPUs and enabled real-time reception, such as PHY performance an advanced multi-instance GPUs, which enabled single GPU virtualization for many smaller ones, the use of NVLink can be useful for vRAN applications that do so.

### B.MAPPING OF LDPC TO GPU

When implementing any algorithm on a GPU, it is important to properly map the algorithm to the hardware to achieve the desired speed. Deciding where to draw kernel boundaries in this mapping algorithm for GPU systems, deciding how to build parallel calculations in formulas, arranging the information to enable proper placement in the memory hierarchy and using profiling techniques to mimic design choices. A vast array of algorithms, such as LDPC decoding, may be set up to execute on a GPU. By offering a kernel for processing nodes and a kernel for processing variable nodes, we expand the architecture outlined in [5] and [7] in this work. Individual threads will fail, which will be accounted for by either a check node or a variable node. For more details on the trade-offs involved in choosing an architectural design, see [5]. A CN kernel and a VN kernel are essentially the two kernels that we design. There are two primary techniques for defining many CWs simultaneously. Initially, we gather $\alpha \in N$ CWs, which we refer to as macro-codewords (MCW). Currently, the same CUDA block is being used to assess these CWs. Secondly, we perform a batch transfer of $c \in N$ MCWs to the GPU so that they may be executed in discrete blocks. Every block will function on a single row of the underlying graph, and each thread will apply the lift factor by working on a sub-row. With this approach, the GPU may receive and decode the $\alpha \cdot \beta$ total code words at the same time. When constructing the system, the operator's willingness to compromise

on hardware, lift factor Z, delay, and throughput will determine which values of α and β are used. There is a maximum of around 1024 threads per block in CUDa. Thus, $\alpha Z \leq 1024$ serves as a restriction. The delay per codeword offered by $\alpha \cdot \beta$ grows as the total number of codewords transferred to the GPU increases every batch. Throughput usually increases as the amount of codewords transmitted every batch throughout increases. There's no need in increasing either option further if the group's overall codeword count is so high that GPU cores will lag significantly. The system builder should experiment with Nvidia profiling tools to find the best settings for their hardware platform and design goals. A figure is displayed. Table 1 provides a description of each memory array along with information about how long it is. The length of the array multiplied by the size of the chosen data type determines the final size of memory for each GPU stream in most of these arrays. We also keep a structure in continuous memory that contains the different parameters needed for decoding in addition to these basic arrays.

1)Check Node Processing

In this kernel, each thread operates in a row of H. Using a two-minute algorithm for scaled MSA, messages are created for each associated variable node and then these messages are stored in global memory. Blocks with dimensions (mb, β, 1) and threads with size (Z, α, 1) are used to implement the CUD mesh. In kernel 1, the check node kernel is displayed. A thread may operate on a single check node, which is the fundamental principle of the kernel. To find the variable nodes connected to it, the individual thread will load a collapsed parity check matrix. It will then take incoming messages from global memory and check the lowest and second lowest messages. It then calculates the messages for each VN according to (2) in the second iteration. Two optimizations are to be noted in the method of Kernel 1. Initially, in accordance with (4), the check node counts the messages delivered by every variable node. We can enhance speed by lowering the number of global memory accesses needed by calculating Q rather than check nodes. Second, we save the difference in our own array dt for use by kernel 2, again saving memory space, as the kernel already has access to both R iterations.

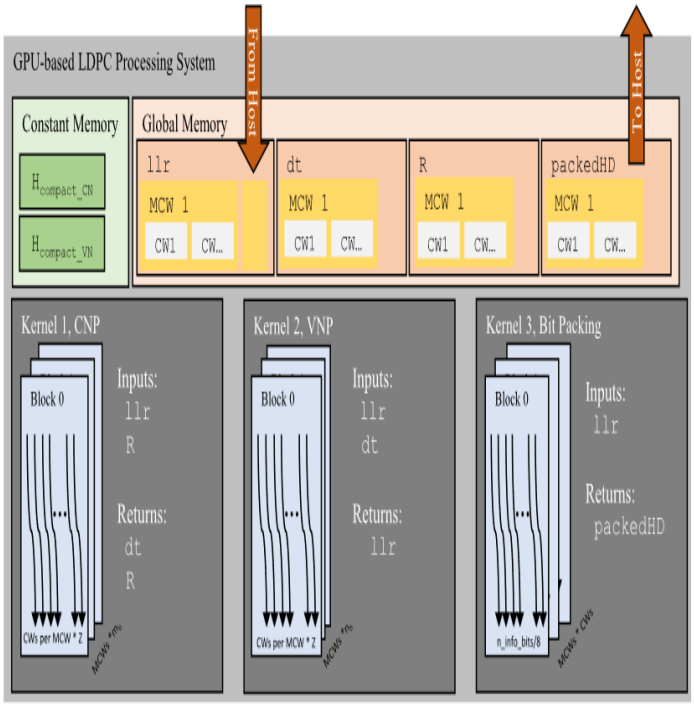



FIGURE 2. GPU Architecture Diagram. The LLRs for multiple MCWs are copied from the host to the device. The LDPC decoding is performed by iterating / times between Kernel 1 and 2 for check-node and variable-node processing. After / iterations, a hard decision is made, and Kernel 3 packs the hard decision bits into a smaller memory footprint format for transfer back to the host.

2) Processing of Variable Nodes

Each thread in the VN kernel corresponds to a H column, and it uses kernel 1's R variables, which are stored in dt, to determine its APP LLR. To update the LLR as in (3), the kernel retrieves the dt elements in accordance with h_compact_vn and saves the variables in a message. Next, the modified LLR is kept in global memory. In kernel 2, the complete algorithm is displayed. Dimensional threads (Z, α, 1) and blocks (nb, β, 1) are used to implement the CUDA mesh.

## C.OPTIMIZATION STRATEGIES

1)Reduced Word Length

The primary optimization we provide in this study is a reduction in word length to an 8-bit char format, in comparison to earlier GPU-based methods. It is well known that six bits is sufficient [23] and is frequently used in FPGA and ASIC applications.

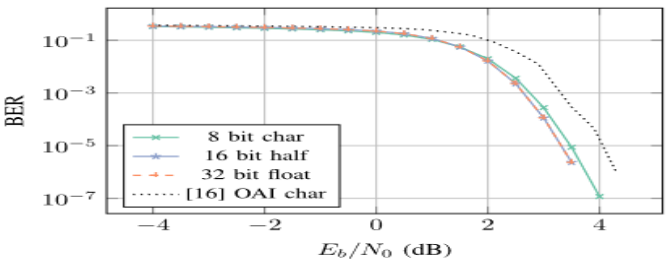



FIGURE 3. BER curve comparing performance when using various data storage precisions on GPU. The (25344, 8448) code was used with 5 iterations.

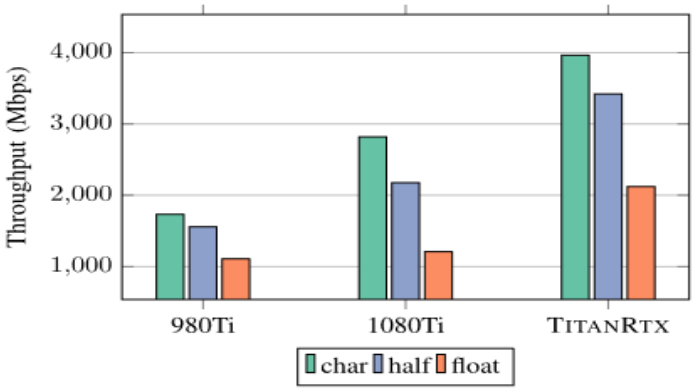



FIGURE 4. Throughput for the considered datatypes on three different GPUs.

We toss to floats for each thread count and leverage this shortened word length to save 4x on data transmission time between hosts and devices as well as global memory accesses.

2) Packing: The Last Hard Choice
A complicated choice can be taken after I total iterations, and each LLR can be represented by a single bit. Many times, this was not utilized in previous study, and the results are obscured by redundant data sets.
Prior to final transmission to the host, we employ a bit-packing kernel. We then make the tough choice to transfer only the final information bits to the host, rather than the complete decoded CW. The expression for this kernel is kernel 3. After retrieving eight components from llr, each thread uses (5) to guide its intricate decision-making. After that, this is somewhat altered to the matching byte. For the host program to return to the CPU, the bytes are kept in the packed array in global memory once each of the eight llr elements has been determined by the thread in charge.

3)Streams
To further reduce the memory transfer between host and device, we use $\eta \in N$ CUDA streams. Each stream acts as a separate process in the CPU application to be organized and synchronized. This allows one stream to calculate, while the other may be updating data to the GPU. This helps ensure that computing resources are fully utilized most of the time and that the device does not wait for new data.

4)Caching of H
The equilibrium rows and column versions, H_compact_cn and h_compact_vn, are assigned to the daily memory of the column version, that is, the fastest nodel, VN or CN path to the memory record.

**D.MEMORY TRANSFER TIME OVERVIEW**
The main bottleneck in LDPC implementation is the need to transfer motion data. First we need to get the LLRs on the GPU which is usually limited by a bus like PCIe. Once the data is in the global memory of the GPU, it must be transferred to the SM for processing by the kernels. This data movement is performed by both kernels for each iteration. In Table 2 , we presented the specifications of each array in memory for LDPC decoding. In this section we state the required memory transfer time in bytes per second as a function of the number of iterations I, and the bus throughput rate r of each.

**IV.GPU MEASUREMENT RESULTS**
a) Performance: First, we assess the BER performance in relation to relative power. Fig. 4. Throughput for three distinct GPUs' worth of data kinds. Eb/N0 is the spectral density. This examines the LDPC's decoding performance and displays the results for various word lengths, as seen in Fig. 3. Bit char is used for 8-bit data storage, and the performance of semi-precision and float data types is almost the same. We also offer an OAI implementation from [16] for comparison, which uses char formats for both data storage and processing to carry out CPU decoding. For a given decoder algorithm, such as min-sum,

performance should be consistent between implementations. Any differences can be attributed to differences in algorithms and quantization. The difference between the OAI implementation and ours here can be attributed to the difference between the

min-sum algorithm used in OAI and the scaled min-sum implemented in this work.

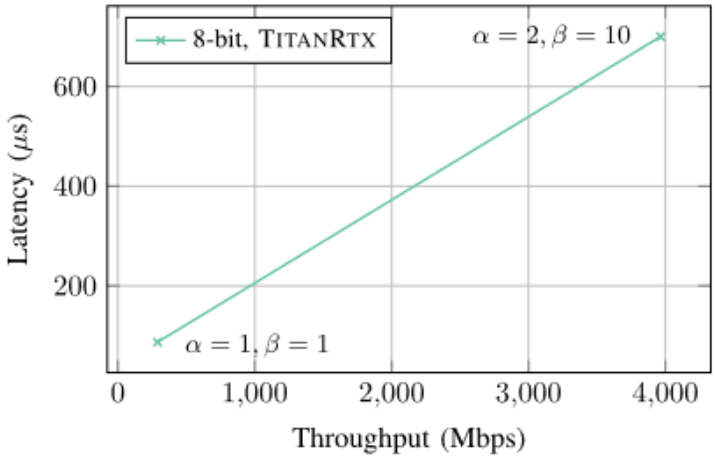



**FIGURE 6.** Latency vs throughput for five iterations. The GPU system architecture can be reconfigured to target latency constraints or throughput.

b)Throughput vs Latency: We tested maximum throughput for three different GPUs. For this test, we configured the GPU to use 6 streams, 20 MCW of replacement, and 2 CW per MCW. This setting helped ensure that the GPU was always full, reducing the impact of PCIe relocations. In Figure 4 we can see that the performance of each generation GPU increases as the number of CUDA cores also increases. The best performance is char data type in T ITAN RTX with 3964 Mbps decoding throughput, including transport time for CPU and GPU In this configuration, the latency is high and is of the order of 700 µs, although this can has been acceptable for eMBB applications.

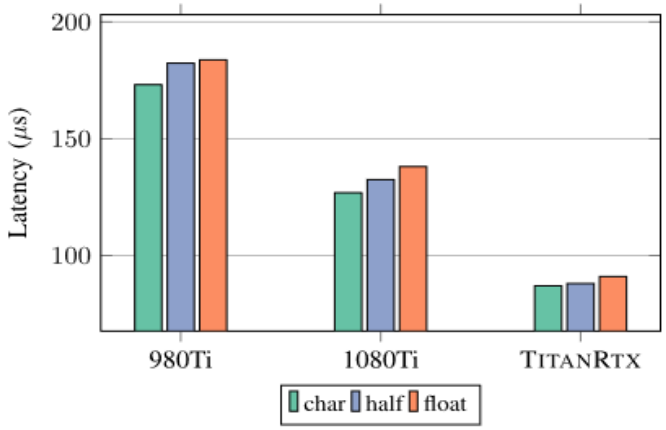



**FIGURE 5.** Latency for the considered datatypes on three different GPUs.

The decoding time can be reduced by targeting URLLC to reset the GPU. To achieve low latency, we can reduce MCWs, MCWs to CWs, and streams. Where there is a good signal-to-noise ratio (SNR), the number of decoding iterations can also be reduced or an earlier termination can be used Testing on several GPUs as shown in Figure 5, we find that T ITAN The RTX is capable in order to achieve latency as low as 87 µs Also, there is travel time for both GPUs The capacity of this system is 290 Mbps.
To better highlight the tradeoff between latency and throughput, the two are plotted against each other in Figure 6. Latency increases by combining more codewords to increase throughput With this batched transfer so the observed latency to send all code words to the GPU, decode them all, and transfer them again is 700 us. This delay is much less than the 4 ms time limit for sending each HARQ response to a user device (UE).

c)Latency versus iteration: In Figure 7 we examine latency as a function of the number of LDPC iterations, I. This also gives an estimate of the performance to use early termination where I if decoding ends before current reaches The LLR vector is set to represent the correct codeword. As the number of iterations increases, the latency also increases at a linear rate. For 8-bit LLR data with 1080 TI, the latency increases at 13.7 µs per iteration.

d)Time allocation: In Table 3 , we allocate the time required for how much time it takes to transfer data against computation. For 1080 Ti, we perform a series of tests varying the MCW rate, α to MCW, β and CW rate. By making this adjustment, we increase the total number of code words in each batch and find that as the total number of code words in the batch increases, there is a proportional increase in each time due

to memory transfer which is increased

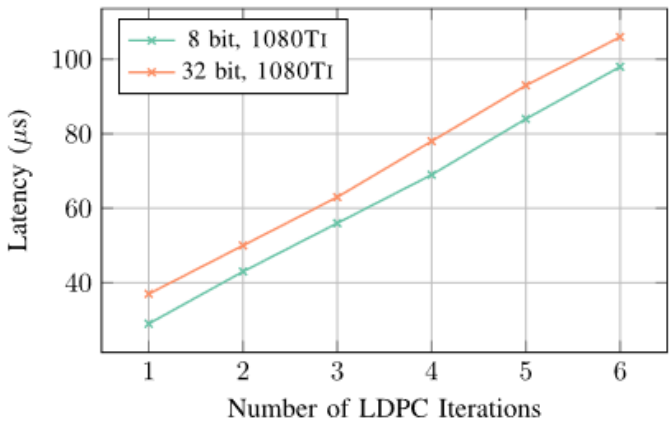



**FIGURE 7.** Latency vs. Iteration. As the number of iterations *I* increases, we see the latency increase at a linear rate of 13.7 $\mu$s per iteration. In this test, we configure the GPU for minimum latency by setting $\alpha = 1$ and $\beta = 1$.

Currently, few publicly published results are available for 5G LDPC decoders. To understand how our decoder fits in the space of 5G NR LDPC decoders in the open book, we surveyed the available metrics, as shown in Table 4. In this work, we construct a variation that lies between maximum throughput and minimum delay are highlighted, and we configured Table 4 to highlight these trade-offs. However, not all tasks reported the lowest latency and the highest throughput, which may be due to code wording schemes. Many short commercial products do not indicate the exact LDPC code used or the decoding frequency. Although we show the application space of each in the table, it is quite impossible to properly compare each without knowing the exact resource consumption and power consumption of each This table is presented for purposes giving a rather functional perspective to look at. However, we should not use it to draw strong conclusions about how our decoder compares to others.

OAI provides a software-based implementation and reports results in [25]. These results were observed on the same platform and the corresponding delay was measured. The LDPC codes used were (25344, 8448) codes from BG1 which had 5 repeats. [8] also used a software-based solution. However, this implementation also uses the AVX SIMD instructions for the architecture. Using a single core, a performance of 271.8 Mbps throughput can be achieved with a delay of 31.08 µs for 10 iterations of the layered decoder. With 18 cores, a maximum throughput of 5 Gbps is reported. [26] also uses a software-based decoder for OAI. A layered min-sum decoder is adopted with the AVX-256 instruction. Using i7, a decode time of 240 µs is reported for an unspecified code and number of iterations. Creonic offers FPGA-based, NR, LDPC decoder accelerators for use throughout the OAI community. In their product brief they show a maximum throughput of 574 Mbps for an unspecified FPGA and LDPC code structure [27]. [28] present a new GPU benchmark for Nvidia 2080Ti for (2080, 1760) code using 10 iterations of the layered minimal decoder. They achieve a maximum throughput of 1380 Mbps with unspecified latency.

## V. vRAN DISCUSSION AND OAI INTEGRATION

vRAN will be the dominant technology going forward. In this section, we discuss vRAN and demonstrate how GPU solutions can be used for baseband computing tasks. As an example, we now discuss the integration of our GPU-based LDPC decoder with OAI

### A.THE NEED FOR SOFTWARE-BASED, VRAN SYSTEMS

Software-based solutions excel in flexibility, and the development of standards-compliant SDR projects is an emerging theme that highlights this. For example, OAI and srsLTE are examples of software-defined platforms for 4G and 5G systems [29], [30]. With this initiative it is now possible to deploy 4G base stations and core networks entirely on object CPUs, and

support a wide range of commercially available off- the-shelf (COTS) Ues. This software-based approach is feasible fast and tested with analysts for each job in the stack Provides an opportunity to test new algorithms Software solutions such as OAI are a more natural fit for potential future deployments based on C-RAN, where operators perform baseband operations in central cloud data center on enterprise grade servers recently vRAN An -RAN extension has been proposed [31], [32]. We illustrate this concept in Figure 8. Here, a cloud computing environment provides a baseband usage accelerator pool that can be used by multiple remote radio headers and RAT standards Virtualization is added to the computing center in vRAN. By virtualizing baseband usage, hardware resources can be freed from process work. Many telecom companies participate in vRAN efforts through the O-RAN Alliance which pushes RAN virtualization through new standards and software development [33]. Virtualization provides many benefits such as improved system management, increased fault tolerance, better scalability, and reduced costs. By decoupling hardware resources from computing, more dynamic resource management based on real-time network requirements and constraints is possible.

Scalability is about adding additional hardware resources to existing resources. Adding the right software and connecting it to a hardware resource pool also makes it possible to switch to multiple standards.

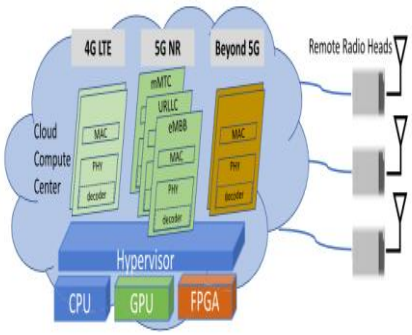



FIGURE 8. Illustration of the flexible, multi-standard vRAN concept. A cloud-compute center can be used in beyond 5G to provide high-performance, baseband processing in a virtualized manner across various hardware accelerators, including GPU and FPGA.

C-RAN/vRAN configurations are also economical because enterprise-grade servers and other COTS devices are widely available to replace application functionality with expensive, highly specialized devices In addition to the above advantages, centralized processing can allow the development of new algorithms and architectures such as coordinated multipoint (CoMP) and cell-free .

These

future vRAN systems will likely be heterogeneous compute clusters with CPUs, FPGAs, application-specific integrated circuits (ASICs), GPUs, and even tensor processing units (TPUs) CPUs can be used for sequential tasks such as planning and medium access control ( MAC) function. FPGAs and ASICs can accelerate specific tasks such as precoding and detection. GPUs can bridge the performance/flexibility gap for any task that requires both. GPUs can also provide high performance for any algorithm without developing any dedicated FPGA/ASIC accelerators. In next-generation applications, machine learning has been proposed for various networking applications and TPUs are available to provide dedicated application components for such applications.

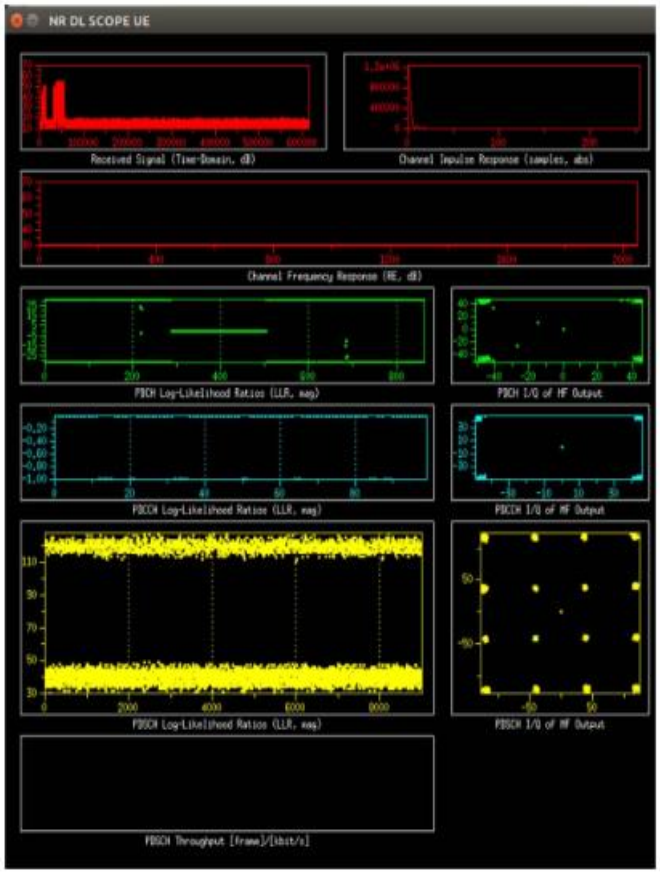




**FIGURE 10. Example calling the custom, LDPC accelerator from the OAI ldpctest.**

vRAN systems. To highlight this portability on SDR platforms, we integrated our decoder with OAI.

Integration is done by adding our CMAKE file to the master CMAKE and updating the function call in the decoder. We evaluated the performance of our GPU decoder in two scenarios. First, we tested with the OAI independent "ldpctest" value. "ldpctest" checks BER performance and latency. Testing OAI current C-based LDPC decoder in "ldpctest", the best decoding delay is FIGURE 10. Example for custom call, LDPC accelerator from OAI ldpctest is 178 µs In contrast our GPU-based decoder latency is 87 µs, 51% reduction in latency .

With the BER data in this test we were able to verify the performance and collect the data used in Figure 3. We then moved the GPU-based LDPC decoder to the larger "RFSimulator" target, driven by the entire 5G NR protocol Strew around. Testing with a full 5G stack, we were able to coordinate with OAI and ensure that CRC passed as expected. Two screenshots from the OAI experiments are shown. First, in Fig. 9 we show the NR performance of the OAI with each star for the physical distance. Figure 10 shows a screenshot of the terminal during OAI. In this experiment, the individual blocks are sent to the GPU for written decoding.

## VI.CONCLUSION

Our top level GPU broker code can catch new code words and start processing them with kernels. Here, we add a message to the log file indicating that the function call to the GPU LDPC accelerator was successful.

When the decoded information bits are sent back to the OAI, a CRC check is performed for the HARQ. In Figure 10 we confirm CRC check passing after the call to the GPU.

## REFERENCES

[1] International Telecommunication Union. (Nov. 2017). Minimum Requirements Related to Technical Performance for IMT-2020 Radio Interface(s). [Online]. Available: https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2410-2017-PDF-E.pdf

[2] K. Li, "Decentralized baseband

processing for massive MU-MIMO systems," Ph.D. dissertation, Dept. Elect. Comput. Eng., Rice Univ., Houston, TX, USA, 2019.
[3] D. Hui, S. Sandberg, Y. Blankenship, M. Andersson, and L. Grosjean, "Channel coding in 5G new radio: A tutorial overview andperformance comparison with 4G LTE," IEEE Veh. Technol. Mag., vol. 13, no. 4, pp. 60–69, Dec. 2018.
[4] M. Wu, G. Wang, B. Yin, C. Studer, and J. R. Cavallaro, "HSPA;/LTE-A turbo decoder on GPU and multicore CPU," in Proc. Asilomar Conf.Signals Syst. Comput., Nov. 2013, pp. 824–828.
[5] G. Wang, M. Wu, Y. Sun, and J. R. Cavallaro, "A massively parallel implementation of QC-LDPC decoder on GPU," in Proc. Symp. Appl.Specific Processors (SASP), 2011, pp. 82–85.
[6] 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.
[7] G. Wang, M. Wu, B. Yin, and J. R. Cavallaro, "High throughput low latency LDPC decoding on GPU for SDR systems," in Proc. IEEE Glob. Conf. Signal Inf. Process., Dec. 2013, pp. 1258–1261.
[8] Y. Xu, W. Wang, Z. Xu, and X. Gao, "AVX-512 based software decoding for 5G LDPC codes," in Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS), 2019, pp. 54–59.
[9] (Jun. 2020). 5G LDPC Intel FPGA IP User Guide. [Online]. Available: https://www.intel.com/content/www/us/en/programmable/documentation/ond1481066696968.html
[10] (Dec. 2019). LDPC Encoder Decoder v2.0 LogiCORE IP Product Brief (PB052). [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/ldpc/v2_0/pb052-ldpc.pdf
[11] A. Balatsoukas-Stimming and C. Studer, "Deep unfolding for communications systems: A survey and some new directions," in Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS), 2019, pp. 266–271.
[12] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in Proc. IEEE Int. Symp. Inf. Theory (ISIT), 2017, pp. 1361–1365.
[13] L. Lugosch and W. J. Gross, "Learning from the syndrome," in Proc. 52nd Asilomar Conf. Signals Syst. Comput., 2018, pp. 594–598.
[14] S. Velayutham. (Oct. 2019). NVIDIA Aerial Software Accelerates 5G on Nvidia Gpus—NVIDIA Blog. [Online]. Available: https://blogs.nvidia.com/blog/2019/10/21/aerial-application-framework-5g-networks/
[15] Nvidia. (Jul. 2020). 5G CloudRAN and Edge AI End-to-End System Featuring NVIDIA Aerial SDK and EGX Platform. [Online]. Available: https://news.developer.nvidia.com/5g-cloudran-and-edge-ai-end-to-end-system-featuring-nvidia-aerial-sdk-and-egx-platform/
[16] (Nov. 2019). OAI Develop-nr Gitlab Repository. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/tree/develop-nr
[17] C. Tarver, M. Tonnemacher, H. Chen, J. C. Zhang, and J. R. Cavallaro, "GPU-based LDPC decoding for vRAN systems in 5G and beyond," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), 2020, pp. 1–5.
[18] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.Y. Hu, "Reduced-complexity decoding of LDPC codes," IEEE Trans. Commun., vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[19] NR; Multiplexing and Channel Coding Version 15.7.0, 3GPP
Standard TS 36.212, Sep. 2019. [Online]. Available: https://portal.
3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?
specificationId=3214
[20] T. Richardson and S. Kudekar,
"Design of low-density parity check
codes for 5G new radio," IEEE Commun.
Mag., vol. 56, no. 3,
pp. 28–34, Mar. 2018.
[21] (2020). NVIDIA A100 Tensor Core
GPU Architecture. [Online].
Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-
Center/nvidia-ampere-architecture-whitepaper.pdf
[22] Mellanox OFED GPUDirect RDMA. Accessed: Aug. 9, 2020.
[Online]. Available:
https://www.mellanox.com/products/GPUDirect-RDMA
[23] T. Zhang, Z. Wang, and K. K. Parhi, "On finite precision implementation of low density parity check codes decoder," in Proc. IEEE Int.
Symp. Circuits Syst. (ISCAS), vol. 4, May 2001, pp. 202–205.
[24] S. Shao et al., "Survey of turbo, LDPC, and polar decoder ASIC
implementations," IEEE Commun. Surveys Tuts., vol. 21, no. 3,
pp. 2309–2333, 3rd Quart., 2019.
[25] F. Kaltenberger. (Sep. 2019). 5G-NR-Development-and-Releases.
[Online].Available:
https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/5g-nr-development-and-releases
[26] 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
Proc. IEEE Globecom Workshops (GC Wkshps), Dec. 2018, pp. 1–6.
[27] Creonic. (2019). 5G LDPC Decoder
and HARQ Buffers Product Brief.
[Online]. Available:
https://www.creonic.com/wp-content/uploads/PB_Creonic_5G_RL15_Decoder.pdf
[28] R. Li, X. Zhou, H. Pan, H. Su, and Y. Dou, "A high-throughput
LDPC decoder based on GPUs for 5G new radio," in Proc. IEEE
Symp. Comput. Commun. (ISCC), 2020,
pp. 1–7.
[29] N. Nikaein et al., "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/10.1145/2677046.26775
[30] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C.
Cano, and D. J. Leith, "srsLTE: An open source platform for LTE
evolution and experimentation," in Proc.
10th ACM Int. Workshop Wireless Netw. Testbeds Exp. Eval.
Characterization, 2016, pp. 25–32.
[31] Wind River. Nov. 2017. vRAN: The
Next Step in Network Transformation.
[Online]. Available:
https://builders.intel.com/docs/networkbuilders/vran-the-next-step-in-network-transformation.pdf
[32] A. Garcia-Saavedra, X. Costa-Perez, D. J. Leith, and G. Iosifidis,
"FluidRAN: Optimized vRAN/MEC
orchestration," in Proc. IEEE
Conf. Comput. Commun., Apr. 2018, pp. 2366–2374.
[33] O-RAN Alliance. O-RAN Use Cases and Deployment Scenarios. Accessed: Aug. 9, 2020.
[Online]. Available:
https://static1.squarespace.com/static/5ad774cce74940d7115044b0/t/5e95a0a306c6ab2d1cbca4d3/1586864301196/O-RAN+Use+Cases+and+Deployment+Scenarios+Whitepaper+February+2020.pdf