Enhancing the Low-PHY functions Efficiency in 5G Networks through Hardware-based SmartNIC Acceleration using OpenCL

Youssef Hassan Gharib[1], Gabriel Pedro Paião[1], Leandro Ronchini Ximenes[1], Rangel Arthur[1]

***Abstract*— The goal of 5G technology is to support new services based on three main usage scenarios: (I) eMBB (Enhanced Mobile Broadband), (II) mMTC (Massive Machine Type Communication), and (III) URLLC (Ultra-Reliable Low Latency Communication). To provide better signal processing within the network, this research aims to propose the implementation of the Low-PHY layer functions of the Distributed Unit (DU) in 5G on hardware (FPGA), leveraging the capabilities of the Open Computing Language (OpenCL). This framework is compatible with heterogeneous computing environments, enabling parallel processing. Since OpenCL programming allows a single program written by the host to be executed across different heterogeneous platforms, it becomes feasible to allocate specific functions to run on the FPGA or other hardware while others run on the CPU, thereby reducing the processing load on the processing unit. Consequently, the processing time of signal samples will decrease.**

***Keywords*— 5G, Low-PHY, Hardware, OpenCL**

I. Introduction

Since 2G wireless networks, Radio Access Network (RAN) architectures have been based on monolithic blocks, where functions are contained in boxes called Baseband Units (BBUs), at the base of radio towers, where signal processing takes place. From the initial phases of the New Radio (NR) of 5G, an attempt was made to disaggregate the BBUs into distributed units (DUs) and centralized units (CUs). The rationale for this was flexibility, allowing network operators to decide where to position processing functions and maximize performance, as well as more cost-effective network deployment. [1]

The 5G structure establishes a highly adaptable and versatile network technology that creates a robust cloud-native mobile network, enabling comprehensive backing for network segmentation. Its objective is to facilitate novel services through three primary usage scenarios: (1) enhanced mobile broadband (eMBB), enhancing broadband accessibility, accelerating connections, and elevating resolution; (2) massive machine-type communications (mMTC), catering to densely connected, cost-effective, and energy-efficient IoT devices; and (3) ultra-reliable low-latency communications (URLLC), catering to critical applications necessitating minimal latency and maximal reliability.

Youssef Hassan Gharib, e-mail: y256870@dac.unicamp.br; Gabriel Pedro Paião, e-mail: g238080@dac.unicamp.br; Leandro Ronchini Ximenes, e-mail: ronchini@unicamp.br; Rangel Arthur, e-mail: rangelft@unicamp.br. [1]Laboratório de Processamento de Sinais, Faculdade de Tecnologia, Universidade Estadual de Campinas, Limeira/SP. This work was funded by the São Paulo Research Foundation (FAPESP). (2023/04992-0).

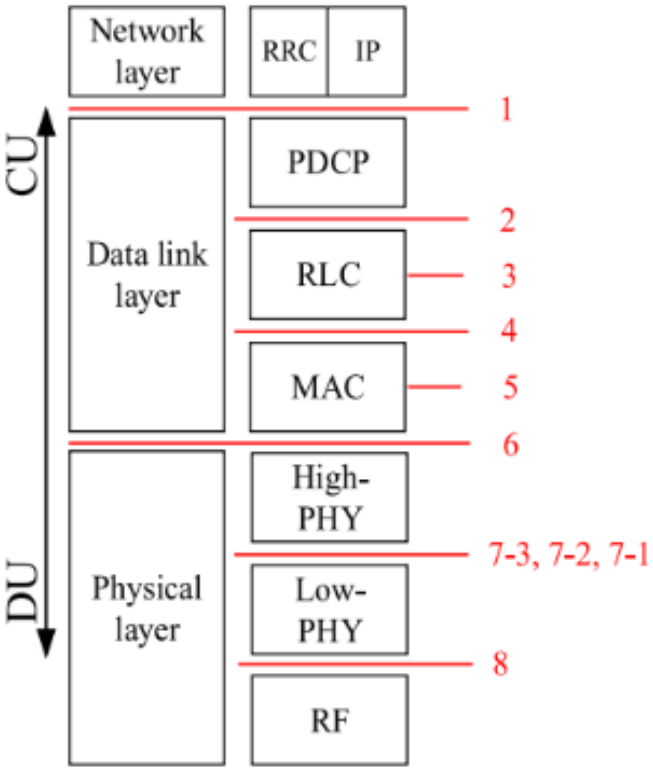

Fig. 1
THE LTE PROTOCOL STACK WITH LAYERS AND SUBLAYERS, INCLUDING THE NUMBERED FUNCTIONAL SPLIT OPTIONS PROPOSED BY THE 3GPP [2]

The 3rd Generation Partnership Project (3GPP) proposed eight functional division options, including several sub-options. The red lines in Figure 1 illustrate different options for functional divisions. The functions below the red line will be those implemented in the DU, and the functions above the red line will be implemented in the CU. The functions in the DU are very close to the user, as they will be located on the antenna mast, and those located in the CU will benefit from centralization and high processing power. Thus, the more functions located in the DU, the more processing has already been done before the data is transmitted in the fronthaul network. [2]

To achieve results that meet the needs of each 5G category, it is necessary to have fast and enhanced signal processing in certain specific blocks of the architecture, to achieve greater offered bandwidth, extended coverage, reduced latency, among other objectives referred to by the 3GPP as Key Performance Indicators (KPIs). [3]

This work proposes the implementation of an hardware-based SmartNIC to be installed within the edge cloud micro data center, positioned near the antenna site housing a DU, to enhance the performance of gNB Low-PHY functions. Specifically, the targeted functions for offloading include the Inverse Fast Fourier Transform (IFFT) and the addition of Cyclic Prefix (CP) for Orthogonal Frequency Division Multiplexing (OFDM) in downlink transmission. The Open Computing Language (OpenCL) was chosen to implement the functions, where a single host can interact with one or more devices using only one program running on these heterogeneous platforms.

II. Literature Review

There are many works that exploit the use of OpenCL framework on hardwares, mainly on FPGAs, for diverse applications, like acceleration techniques for 5G functions, image processing and cryptography.

The paper from [4] provides an overview of using OpenCL-based hardware acceleration techniques to implement computationally intensive 5G functions, specifically the Inverse Fast Fourier Transform (IFFT) and Cyclic Prefix Addition. It discusses how OpenCL can be used to offload these functions onto reconfigurable hardware like FPGAs and GPUs, which can provide significant performance improvements compared to general-purpose CPUs. It covers optimization techniques like exploiting data parallelism, loop unrolling, and using sliding windows to improve pipeline throughput. The ability to accelerate critical 5G functions in hardware is an important enabler for the virtualization and cloud-native deployment of 5G networks. By offloading computationally intensive tasks, it can help meet the low-latency and high-throughput requirements of 5G while also reducing power consumption and costs.

The work from [5] presents an FPGA accelerator design for the transform and quantization functions of the new H.266/Versatile Video Coding (VVC) standard, implemented entirely using an OpenCL-based high-level design approach. VVC significantly increases the computational complexity compared to previous video coding standards like HEVC, particularly in the intra-frame coding part. The authors leverage the flexibility and parallelism of OpenCL to design a modular, pipelined accelerator architecture that can efficiently implement the required 2D DCT, quantization, and inverse operations. By exploiting data-level parallelism within each kernel and pipeline-level parallelism across kernels, the FPGA implementation achieves processing speeds ranging from 6.5 to 83.3 frames per second, with an average of 16.1 fps - a 3.22x speedup over a CPU-only implementation.

Another work, reported in [6], presents an OpenCL-based methodology for efficiently implementing pipelined architectures of the real-valued Fast Fourier Transform (RFFT) on FPGA platforms. RFFT is an ideal candidate for high-speed, low-power FFT processing as it requires approximately half the number of arithmetic operations compared to traditional complex-valued FFT (CFFT). The authors identify a regular computational pattern in the RFFT signal flow graph and design corresponding butterfly structures. They also develop novel twiddle factor access schemes to enable efficient pipelining. By encoding the RFFT algorithm in a single for-loop, the OpenCL compiler can fully unroll the loop to automatically construct the pipelined architecture. Experiments show that the proposed RFFT implementation on an Intel Stratix-10 FPGA achieves a 1.48x speedup over an Intel CFFT design on the same FPGA, while consuming 12% less logic resources and 16% fewer DSP blocks. Compared to GPU and CPU FFT libraries, the FPGA-based RFFT demonstrates significant advantages, with 2.49x and 21.12x speedups over CUFFT and FFTW respectively, along with 3.09x and 16.09x better energy efficiency.

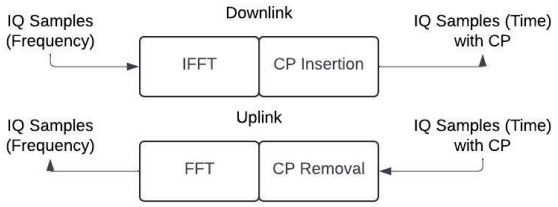

Fig. 2
Low-PHY layer protocol to be implemented in hardware

III. Considered functions deployment

The offloaded functions on the SmartNIC, presented by Figure 2, will operate as follows: in the downlink direction, the in-phase and quadrature (IQ) signal samples, consisting of the real part and the complex part in the frequency domain, arrive at the hardware (FPGA) and undergo IFFT to convert the samples to the time domain. The cyclic prefix (CP) is inserted as a guard interval to prevent intersymbol interference (ISI). In the uplink direction, the IQ signal samples in the time domain, including the cyclic prefix, are received, the guard interval is removed, followed by the FFT operation that converts the samples to the frequency domain. The insertion/removal of the cyclic prefix is performed by adding/removing redundant bytes before each symbol of Orthogonal Frequency Division Multiplexing (OFDM). [7]

For this research, the considered OFDM structure, in Figure 3, uses the modulation scheme known as Quadrature Amplitude Modulation (QAM). In the conventional OFDM modulator, the modulated symbols are converted from serial to parallel format, and then mapped onto the subcarriers of the OFDM system. After that, the Low-PHY functions discussed before, IFFT and CP addition, are applied. The insertion of the cyclic prefix is to mitigate ISI caused by multipath propagation. Finally, the OFDM symbol with CP is then transmitted over the wireless channel.

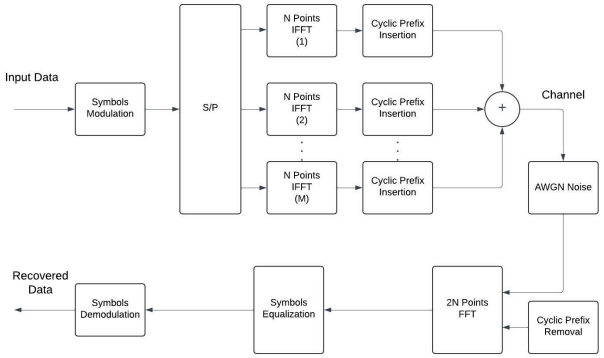

Fig. 3
OFDM TRANSCEIVER

In the receiver, the process is inverse, applying the CP removal and Fast Fourier Transform (FFT) with a factor of 2N, followed by equalization and signal demodulation.

The integration of FFT/IFFT stands out as a pivotal element, representing the most intricate and resource-demanding segment within the Orthogonal Frequency Division Multiplexing (OFDM) framework. FFT represents a refined computation method of the Discrete Fourier Transform (DFT), achievable through the application of the following formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} = \sum_{n=0}^{N-1} \omega^{kn} x_n \tag{1}$$

where X_k are the samples in the frequency domain, x_n are the samples in the time domain, N is the number of FFT/IFFT points, and ω^{kn} is the twiddle factor.

In papers such as [8], [9] and [10], the authors explore the use of different FFT algorithms that use different radix numbers. One of them is the Radix-2 algorithm, a type of FFT calculation that decomposes the DFT into smaller DFTs, reducing computation time. The Radix Butterfly configuration divides FFT computation into different stages, where the higher the Radix number, the fewer stages required, but with more complex twiddle factors. For example, to compute a 16-point FFT using radix-2, four stages will be required, but with twiddle factors at angles 0° and 180°. On the other hand, to compute a 16-point FFT using radix-4, only two stages will be needed, but with twiddle factors at angles 0°, 90°, 180°, and 270°. [7]

The Radix-2 algorithm separately computes the DFTs of the input values with even indices (x_0, x_2, ..., x_{N-2}) and those with odd indices (x_1, x_3, ..., x_{N-1}), then combines these two results to produce the DFT of the entire sequence. This concept can be recursively applied to reduce the total computation time from O(N^2) to O(N logN). This algorithm requires that N be a power of 2.

To implement Radix-2, the main formula of the DFT is divided into two parts: a sum over indices numbered with even values and a sum over indices numbered with odd values:

$$X_k = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)k} + e^{-\frac{2\pi i}{N}k} \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{-\frac{2\pi i}{N}(2m)k} \tag{2}$$

Let's denote the DFT of the input values with even indices as E_k and the DFT of the input values with odd indices as O_k, then we obtain:

$$X_k = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{-\frac{2\pi i}{(\frac{N}{2})}mk} + e^{-\frac{2\pi i}{N}k} \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{-\frac{2\pi i}{(\frac{N}{2})}mk} \tag{3}$$

$$E_k + e^{-\frac{2\pi i}{N}k} O_k \tag{4}$$

With the functions E_k and O_k of k being periodic with a period of N/2. [11]

Therefore, by splitting the DFT into reduced sums, we achieve faster computation of the transform, thus reducing processing time.

IV. OPENCL FRAMEWORK

The programming of this proposed work will be done using the OpenCL framework [12]. It enables reconfigurable computing applications and is applicable across different architectures, from CPUs and GPUs to FPGAs. While each platform requires specific optimizations, a single OpenCL code can be applied across all of them, enhancing scalability. Moreover, OpenCL offers various features that leverage parallelism to improve performance and processing time, such as loop unrolling, reduction of function calls, and maximizing global memory bandwidth. [13][14]

Despite processing on the hardware, the system will still require a computer to function as a controller, as the OpenCL language mechanism follows the master-slave approach, where a host software (computer) controls the execution of the OpenCL program (kernel) within a computing device (FPGA) supporting the OpenCL framework. Before initiating its routine, the host must initialize the context and communication with the device, all made possible through the OpenCL Application Programming Interface (API).

V. RESULTS AND DISCUSSIONS

As shown in Figure 4 and Figure 5, the performance analysis of the IFFT using the clFFT client program on both CPU and GPU reveals significant differences, highlighting the potential benefits of offloading certain tasks to more specialized hardware. The tests conducted on an Intel(R) Core(TM) i7-10510U CPU and an NVIDIA GEFORCE® MX110 GPU demonstrate that the GPU consistently outperforms the CPU in terms of execution time, regardless of the batch size. This is particularly notable given that the CPU is typically responsible for managing a multitude of background operations simultaneously, which can impact its ability to execute computationally intensive tasks efficiently.

For this implementation, the number of points used are 128, 256, 512, 1024 and 2048. All power of two, taking into

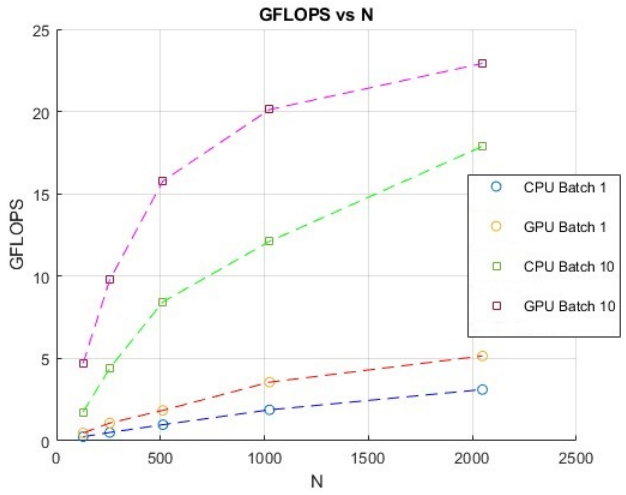

Fig. 4
IFFT GFLOPS VS LENGTH N FOR CPU AND GPU WITH BATCH SIZES 1 AND 10

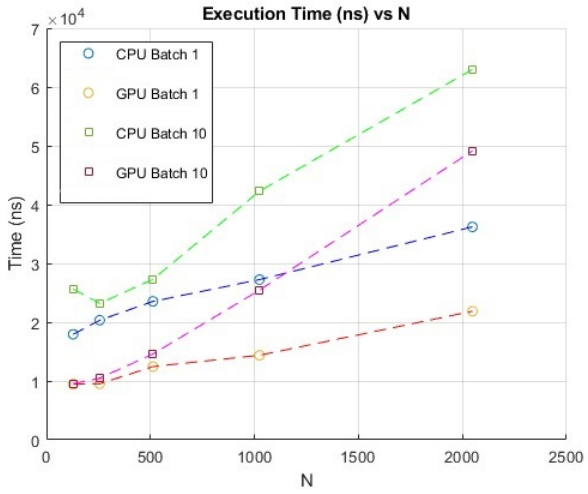

Fig. 5
IFFT EXECUTION TIME VS LENGTH N FOR CPU AND GPU WITH BATCH SIZES 1 AND 10

account applications of the radix-2 algorithm, or mixing radix-2 with radix-4.

For batch size 1, the GPU shows remarkable performance improvements over the CPU. For instance, with N = 128, the GPU achieves an execution time of 9,461 ns compared to the CPU's 17,944 ns. As the value of N increases, the GPU continues to maintain its lead, with the time taken for N = 2048 being 21,859 ns on the GPU versus 36,211 ns on the CPU. The GFLOPS metrics further support these findings, where the GPU consistently achieves higher GFLOPS across all tested values of N. This indicates that the GPU is able to handle more floating-point operations per second, a critical factor in high-performance computing applications.

When increasing the batch size to 10, both CPU and GPU exhibit increased execution times, but the GPU still retains a clear advantage. For example, at N = 128, the GPU execution time is 9,528 ns, while the CPU takes 25,655 ns. Even at N = 2048, the GPU completes the task in 49,144 ns compared to the CPU's 62,977 ns. Notably, the GFLOPS for the GPU with batch size 10 reach up to 22.92 for N = 2048, demonstrating the GPU's superior capacity to scale performance with increased workloads.

A key observation from the tests is the relationship between GFLOPS and execution time. Despite the increase in execution time with larger batch sizes, the GFLOPS achieved by both CPU and GPU also increase, with the GPU showing a more pronounced improvement. This suggests that while individual operations may take slightly longer with larger batch sizes, the overall throughput (number of operations completed per second) improves significantly. This is particularly relevant for applications requiring high computational throughput, such as signal processing in 5G networks.

In the context of 5G functions, such as the IFFT in OFDM (part of the Low-PHY functions), offloading the computation to an OpenCL device like a powerful GPU or FPGA can enhance overall system performance. By reducing the computational load on the CPU, which is already handling other critical functions, the system can achieve higher efficiency and faster processing times. This approach aligns with the broader trend of utilizing specialized hardware to handle specific tasks in high-performance and real-time systems, ensuring optimal resource utilization and improved performance metrics.

VI. CONCLUSION

This work proposed the implementation of a hardware-based SmartNIC using the OpenCL framework to enhance the performance of the disaggregated gNB Low-PHY functions in the distributed units (DUs). The performance of the Low-PHY implementation in the NVIDIA GeForce® MX110 GPU was compared against the Intel(R) Core(TM) i7-10510U CPU performance of the same function.

The results showed a significant performance advantage for the GPU over the CPU in both execution time and GFLOPS across various values of N and batch sizes, demonstrating the GPU's superior parallel processing capabilities. Even with larger batch sizes, the GPU consistently handled higher throughput of operations, making it a more suitable choice for computationally intensive tasks in 5G networks. The variability in CPU performance is likely due to the additional functions being executed alongside Low-PHY tasks, highlighting the complexity of network function offloading and the importance of considering overall system architecture when designing hardware accelerators.

Considering that FPGAs should be used in this type of implementation, we can achieve better performance for the Low-PHY functions, alongside lower energy consumption per operation. As per the example cited [7], employing the FPGA-based SmartNIC avoided the latency introduced by the transfer of data between the host and the device, as well as between off-chip and on-chip memory transfers.

Even though the proposed approach offers a promising solution for enhancing signal processing performance in 5G

networks by leveraging hardware-based accelerators and the OpenCL framework, further research and optimization efforts are warranted. Future work should focus on fully realizing the potential of this approach and addressing challenges associated with heterogeneous network architectures and varying processing requirements.

ACKNOWLEDGMENTS

This work is the result of a Scientific Initiation Project linked to Process nº 2023/04991-3, funded by the São Paulo Research Foundation (FAPESP).

REFERENCES

[1] E. Jordan, *Open RAN functional splits, explained*. 5G Technology World, 2021, https://www.5gtechnologyworld.com/open-ran-functional-splits-explained/. Last Accessed: (2024-05-04)

[2] LARSEN, L. M. P.. CHECKO, A.. CHRISTIANSEN, H. L, *A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks*. IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 146-172.

[3] CHAMOLA, V.. PATRA, S.. KUMAR, N.. GUIZANI, M.. *FPGA for 5G: Reconfigurable Hardware for Next Generation Communication*. IEEE Wireless Communications, vol. 27, no. 3, pp. 140-147.

[4] J. C. Borromeo, K. Kondepu, N. Andriolli and L. Valcarenghi, *An Overview of Hardware Acceleration Techniques for 5G Functions,*. 2020 22nd International Conference on Transparent Optical Networks (ICTON), Bari, Italy, 2020, pp. 1-4.

[5] H. M. Waidyasooriya et al., *OpenCL-Based Design of an FPGA Accelerator for H.266/VVC Transform and Quantization*. 2022 IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS), Fukuoka, Japan, 2022, pp. 1-4.

[6] Z. Qian and G. Gan, *Accelerating Real-Valued FFT on CPU-FPGA Platforms*. in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

[7] BORROMEO, Justine Cris. KONDEPU, Koteswararao. ANDRIOLLI, Nicola. VALCARENGHI, Luca. *FPGA-accelerated SmartNIC for supporting 5G virtualized Radio Access Network*. Computer Networks. Volume 210. 2022

[8] R. H. Neuenfeld, M. B. Fonseca, E. A. C. da Costa and J. P. Oses, *Exploiting addition schemes for the improvement of optimized radix-2 and radix-4 fft butterflies* 2017 IEEE 8th Latin American Symposium on Circuits & Systems (LASCAS), Bariloche, Argentina, 2017, pp. 1-4.

[9] Y. Zhou, S. Li, C. Yang, J. Kuang, J. Chen and C. Zhang, *Hardware Efficient Radix-3/4/11 based FFT Processor for 5G NR Application* 2022 56th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 2022, pp. 634-639.

[10] C. Yang, J. Wu, S. Xiang, L. Liang and L. Geng, *A High-Throughput and Flexible Architecture Based on a Reconfigurable Mixed-Radix FFT With Twiddle Factor Compression and Conflict-Free Access* in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 31, no. 10, pp. 1472-1485, Oct. 2023.

[11] UNIVERSITY OF CONNECTICUT Department of Physics. *radix-2 fast fourier transform* https://www.phys.uconn.edu/. Last Accessed: (2024-05-03)

[12] MUNSHI, A.. GASTER, B. R.. MATTSON, T. G.. FUNG, J.. GINSBURG, D.. *OpenCL: Programming Guide* Addison-Wesley, 2011.

[13] CIVERCHIA, F.. PELCAT, M.. MAGGIANI, L.. KONDEPU, K.. CASTOLDI, P.. VALCARENGHI, L., *Is OpenCL Driven Reconfigurable Hardware Suitable for Virtualising 5G Infrastructure?* IEEE Transactions on Network and Service Management, vol. 17, no. 2, pp. 849-863, June 2020.

[14] ZOHOURI, Hamid Reza. *High Performance Computing with FPGAs and OpenCL*. Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2018.

[15] A. Ghosh, A. Maeder, M. Baker and D. Chandramouli, *5G Evolution: A View on 5G Cellular Technology Beyond 3GPP Release 15*. IEEE Access, vol. 7, pp. 127639-127651, 2019.

[16] electronicsnotes. *CP-OFDM Cyclic Prefix OFDM Explained* https://www.electronics-notes.com/articles/radio/multicarrier-modulation/ofdm-cyclic-prefix-cp.php. Last Accessed: (2024-04-16)

[17] NVIDIA Developer. *CUDA Toolkit* https://developer.nvidia.com/cuda-toolkit. Last Accessed: (2024-04-20)

[18] Clfft, 2021, https://github.com/clMathLibraries/clFFT. Last Accessed: (2024-04-06)

[19] F. Civerchia, P. Castoldi, L. Valcarenghi and M. Pelcat, *Exploiting reconfigurable computing in 5G: a case study of latency critical function: Invited Paper.* 2019 IEEE 20th International Conference on High Performance Switching and Routing (HPSR), Xi'an, China, 2019, pp. 1-6.

[20] S. Ghosh *Performance Evaluation on the Basis of Bit Error Rate for Different Order of Modulation and Different Length of Subchannels in OFDM System* International Journal of Mobile Network Communications & Telematics (IJMNCT) Vol. 4, No.3, June 2014