Reducing the Energy Footprint of 5G: A gNB on MPSoC with Low-Power FFT Acceleration

Journal:	*IEEE Transactions on Green Communications and Networking*
Manuscript ID	TGCN-RP-A2-25-0176
Manuscript Type:	Regular Paper - Area 2: Green Wireless Communication Networks
Date Submitted by the Author:	06-Jun-2025
Complete List of Authors:	Bourenane, Abdelghani; Scuola Superiore Sant'Anna Paolini, Emilio ; Scuola Superiore Sant'Anna Andriolli, Nicola; Universita degli Studi di Pisa Dipartimento di Ingegneria dell'Informazione Valcarenghi, Luca; Scuola Superiore Sant'Anna
Keyword:	Acceleration, Cellular neural networks, Computer networks, Green function

Reducing the Energy Footprint of 5G: A gNB on MPSoC with Low-Power FFT Acceleration

Abdelghani Bourenane*, Emilio Paolini*, Nicola Andriolli†, Luca Valcarenghi*

*TeCIP Institute, Scuola Superiore Sant'Anna, Pisa, Italy

†Department of Information Engineering (DII), University of Pisa, Pisa, Italy

Abstract

Open, virtualized architectures are reshaping cellular networks but purely software-based 5G deployments often struggle to meet strict real-time requirements. Thus, accelerators are often utilised even at the expense of a high energy consumption. In this paper, we propose and demonstrate an energy-efficient solution that accelerates Fast Fourier Transform (FFT) computations by placing Low-PHY functions in an MPSoC FPGA. Moreover, by placing High-PHY layer functions in the CPU of the same MPSoC, we eliminate host-to-accelerator bottlenecks (e.g., accelerator to processor transfer time) that typically limit hardware speedups and contribute to unnecessary energy expenditure. Though implemented into a single MPSoC, the solution is compatible with Open RAN solutions, such as the one proposed by the Small Cell Forum (SCF) in the 5G function application platform interface (5G-FAPI). We validate our MPSoC-based gNB architecture using OpenAirInterface. Experimental results demonstrate that the MPSoC-based approach achieves up to $13\times$ speedup compared to CPU-based implementations, particularly for larger FFT sizes, which directly contributes to reduced energy consumption per processed task. Furthermore, a detailed power consumption analysis shows that, while the processing system remains a significant energy consumer, hardware acceleration enhances overall energy efficiency by shifting computationally intensive tasks to dedicated, optimized hardware, thereby substantially reducing the energy consumption for these demanding functions and paving the way for greener 5G deployments.

Index Terms

Hardware Acceleration, Open RAN, FFT, Wireless Network, 5G, Green Networking

Manuscript created May, 2025; this study is supported by the Chips Joint Undertaking (JU), European Union (EU) HORIZON JU-IA, and its members (including top-up funding by MIMIT), under grant agreement No. 101140087 (SMARTY). It is also supported by the European Union under Italy's National Recovery and Resilience Plan (NRRP) of NextGenerationEU, ("Telecommunications of the Future" partnership PE00000001 - program "RESTART"), and by the Italian MUR under the FoReLab Project (Departments of Excellence).

I. Introduction

Next Generation (NextG) mobile networks are aiming to support a wide range of use cases, that have been categorised into four classes, namely *Enhanced Human Communication* (e.g., immersive experience, telepresence and multimodal interaction), *Enhanced Machine Communication* (e.g., robotic communication and interaction), *Enabling Services* (e.g., positioning, mapping, automatic protection, smart health, and manufacturing), and *Network Evolution* (e.g., Native Artificial Intelligence (AI) exposed as a service, energy efficiency, and coverage) [1]. Meeting these use case demands necessitates flexible, scalable, and, crucially, energy-efficient network architectures [2].

However, current Radio Access Network (RAN) components are designed as monolithic solutions, where all layers of the cellular protocol stack are tightly integrated into a single, vendor-specific implementation [3]. This rigid design limits flexibility, hampering equipment reconfigurability and coordination among network nodes, while also reinforcing vendor lock-in. As a result, achieving optimized radio resource management and efficient spectrum utilization becomes significantly more difficult [4].

To address the aforementioned challenges and, additionally, vendor lock-in, high costs, and limited scalability, Open RAN architectures, such as the one specified by the O-RAN Alliance [5] and Small Scale Forum (SCF) [6], promote disaggregated, virtualized, and interoperable solutions. These architectures utilize open interfaces and intelligent controllers to enable flexible, data-driven, and efficient network operations [3].

Open RAN operates on open, virtualized, and disaggregated network functions hosted on Commercial Off-The-Shelf (COTS) hardware, with a Software Defined Networking (SDN) control for resource allocation and dynamic adaptation [3], [7]. This design introduces significant benefits, including flexible network reconfiguration to support evolving protocols, multi-application spectrum sharing through enhanced resource orchestration, and the ability to place network functions across the edge-cloud continuum [5]. However, purely software-based Open RAN compliant 5G implementations running on commodity x86 hardware often struggle with high computational overheads when handling real-time PHY layer processing, including functions like FFT in the Low-PHY, and channel estimation and decoding in the High-PHY [8], [9].

To address performance trade-offs, the Open RAN framework offers the flexibility to leverage hardware accelerators, such as Field Programmable Gate Arrays (FPGAs), to accelerate network functions and offload compute-intensive tasks. This approach reduces latency and power consumption compared to alternative programmable hardware like GPUs or DPUs, while offering runtime and field programmability to adapt to evolving scenarios [10].

Despite these benefits, challenges remain. For instance, while FPGA acceleration enhances throughput via parallelism and pipelining, the exchange of data through off-chip interfaces introduces latency and consumes considerable power, potentially negating performance gains and overall energy efficiency benefits. Studies have demonstrated that integrating accelerators with in-chip memory or optimizing off-chip memory bandwidth can alleviate host-to-device communication overhead and associated energy expenditure, paving the way for more efficient and scalable implementations in Open RAN deployments [10].

While traditional offloading strategies using discrete accelerators (e.g., GPUs or external FPGAs connected via Peripheral Component Interconnect Express (PCIe)) can mitigate some bottlenecks, they still suffer from data transfer latency and limited energy efficiency [9]. To overcome these constraints, in-package Multi-Processor System-on-Chip (MPSoC) solutions have emerged as an appealing alternative. These systems combine embedded CPU cores (e.g., ARM-based) and FPGA fabric on the same die, thus reducing host-to-device overhead and improving real-time performance. Although on-chip communication fabrics are not immune to congestion under heavy load, our approach leverages hardware where accelerator and processor are integrated in the same die and interconnected by AXI bus. This solution inherently eliminates the significant latency overhead associated with traversing the PCIe bus for host-to-accelerator data exchange.

Recently, such integrated solutions have attracted a lot of attention. For instance, commercial CPUs like Intel's Xeon processors with vRAN Boost now incorporate integrated accelerators for functions such as FFT, aiming at improving vRAN performance by reducing the need for discrete accelerator cards [11]. Despite these significant commercial advancements, a key limitation remains in the general-purpose nature of these processors, which lack the reprogrammability and fine-grained control offered by FPGA-based solutions.

Designing an Open RAN-compliant solution capable of simultaneously meeting latency, throughput, and programmability requirements poses significant challenges, especially with dynamic 5G/6G workloads [12], [13]. As highlighted in our previous work [10], a critical aspect of tackling these challenges is the need to eliminate host-to-accelerator communication bottlenecks. Without addressing these challenges, offloading computationally intensive tasks may fail to outperform traditional general-purpose CPU implementations.

By building upon the approach presented in [10], in this paper we provide a solution where Low-PHY Fast Fourier Transform (FFT) computations are offloaded from the MPSoC processor to the FPGA while all the other layers of the next generation eNB (gNB) protocol stack are deployed in the ARM cores within the MPSoC. Although the solution is integrated into a single device it is flexible enough to support Open RAN interfaces. In particular, its full potentials could be exploited in the SCF where split option 6 is considered [6]. In the following, the architecture of our integrated solution, its implementation, and the specific experimental results concerning FFT acceleration, resource utilization, and power consumption are discussed.

II. Background

The benefits of offloading selected gNB functions has been shown in [14], where an FPGA accelerated Low Density Parity Check (LDPC) has been integrated into a 5G NR software stack OAI. The proposed solution was able to achieve a $13\times$ speedup in processing time when compared to a single-core software solution. In [15] authors demonstrate the high throughput and low-latency capabilities of Graphics Processing Units (GPUs) for LDPC decoding as an alternative to FPGA and Application Specific Integrated Circuit (ASIC) decoders, effectively providing high performance while maintaining the benefits of a software-based solution, though their power consumption can be a concern compared to FPGAs for certain sustained workloads.

Additionally, many efforts have been put into investigating the best design choices and computing platforms to host the Orthogonal Frequency Division Multiplexing (OFDM) functions. For instance, the study in [16] introduces a Software Defined Radio approach using high-level abstraction tools for developing and testing OFDM-based transceivers. The approach is practically demonstrated in FPGA platforms, showing the effectiveness in addressing the computational demands of OFDM, often with better energy profiles than CPU-only solutions.

Other recent works, such as [9], have demonstrated the potential of FPGA-accelerated SmartNICs to enhance the performance of Low-PHY functions in 5G gNB Distributed Unit (DU). These implementations leverage hybrid architectures, offloading specific computationally intensive tasks, such as FFT/inverse FFT (iFFT) and Cyclic Prefix addition/removal, into FPGA hardware while maintaining other layers on general-purpose processors. By utilizing functional splits, including 7-1 and 2, these solutions effectively reduce processing time and energy consumption compared to CPU-based alternatives. However, such architectures face key limitations. The reliance on hybrid designs, where data must traverse PCIe interfaces between host

Central Processing Units (CPUs) and FPGA devices, introduces additional latency and synchronization overheads. Indeed a significant latency overhead is associated with traversing the PCIe bus for host-to-accelerator data exchange. This data exchange not only adds latency but also incurs an energy cost for data transfer. Furthermore, the need to manage data transfer between global and local memory within the FPGA can mask the benefits of hardware acceleration, particularly for real-time applications like Ultra Reliable Low Latency Communications (URLLC). These bottlenecks limit the scalability and overall efficiency of such solutions in fully virtualized and disaggregated 5G networks.

Building on this foundation, our work introduces a novel approach by fully integrating all layers of the gNB protocol stack, from Low-PHY to higher layers, in the ARM cores of a System on Chip (SoC). This fully on-chip design eliminates reliance on external accelerators, such as external GPUs, and resolves host-to-device communication bottlenecks, thereby reducing both latency and the power associated with off-chip data movement. Moreover, this architecture simplifies the deployment process by consolidating all gNB functionalities within a single hardware platform, paving the way for more streamlined and adaptive Open RAN implementations.

III. Open RAN with offloaded FFT

A. Open RAN Functional Splits

Open RAN implementations are based on a disaggregated and virtualized architecture to enable flexibility, scalability, and cost-efficiency in NextG wireless networks [3], [17], [6], [18]. A central aspect of this architecture is the functional split between the Central Unit (CU), the DU, and Radio Unit (RU), which allows the distribution of computational and signal processing tasks across different hardware and software platforms [19]. This modularity supports various deployment scenarios, including centralized, distributed, and hybrid configurations that differ by the type of functions that are deployed in the RU, DU, and CU respectively.

Among the functional splits defined by 3GPP [20], *Option 7-2x* is adopted by the O-RAN alliance for its balance between computational efficiency and latency. This split divides the physical layer into: (i) Low-PHY, located in the RU, handling the FFT/iFFT, precoding (in downlink only and if not implemented in the DU), digital and analog beamforming, and analog/digital conversion tasks; (ii) High-PHY, located in the DU, managing resource element mapping, precoding (if not implemented in the RU), layer (de)mapping, (de)modulation, (de)scrambling, en/decoding, and equalisation, Indirect Forwarding Tunnel (IDFT), and channel estimation (in uplink only). The DU handles also Medium Access Control (MAC) and Radio Link Control (RLC) layers, while the CU is responsible for higher layers, including the Packet Data Convergence Protocol (PDCP) and Radio Resource Control (RRC)/Service Data Adaption Protocol (SDAP). This hierarchical distribution ensures efficient processing of RAN functionalities while maintaining flexibility and scalability across various deployment scenarios.

Another utilised split option is the split option 6 adopted by the SCF. SCF defines the 5G nFAPI [6] that is a standardized interface that separates the MAC and PHY layers in 5G networks, allowing the MAC layer to be virtualized and run on centralized or cloud infrastructure. This approach builds on similar work done for 4G and supports the Option 6 functional split defined by 3GPP. In particular, the 5G nFAPI enables more flexible, scalable, and cost-efficient deployment of 5G small cells, which are essential for dense network coverage.

The Telecom Infra Project (TIP) Open RAN initiative [18] architecture considers several split options. In particular, the following solutions are considered: an all integrated RAN with disaggregation at SW and HW level, a split RAN with RU, BBU (DU/CU), a split RAN with RU, DU and CU, a split RAN with integrated RU/DU, CU.

In current Open RAN software, several approaches are utilised to implement the Low-PHY functions. For example, OpenAirInterface utilizes a single instruction multiple data (SIMD) FFT/iFFT implementation in a CPU [21], [22]. However, such implementation in the RU would require a SoC with a processor capable of supporting it, potentially increasing latency (because of the data transfer to the CPU) and energy consumption. Indeed, functions such as FFT and precoding are computationally intensive and require real-time processing to meet strict performance standards. Moreover, notably, RUs are often resource-constrained and operate under strict energy limitations, making these challenges particularly relevant [23]. As traditional general-purpose software implementations struggle to achieve the required efficiency under these constraints [10], hardware acceleration becomes crucial. Efficiently implementing Low-PHY functionalities using hardware accelerators not only ensures compliance with performance requirements, but also optimizes resource utilization in RU deployments. For example, private company initiatives, such as those by Comcores [24], have demonstrated the practical application of FPGA-based hardware acceleration in RUs. These efforts highlight the industry's drive towards open, efficient, and high-performance RAN solutions while leveraging advanced hardware architectures to meet modern wireless communication demands.

B. MPSoC-based gNB

The proposed system addresses the computational and energy challenges of FFT processing by integrating accelerated Low-PHY functionalities into a fully operational 5G RAN stack, utilizing the OpenAir Interface (OAI) software RAN stack [25]. Our integrated 5G gNB architecture, depicted in Fig. 1, implements these 5G gNB functionalities on a single MPSoC. This MPSoC hosts the CU functions and the DU functionalities (High-PHY, MAC, and RLC layers), which run on the embedded CPU. Crucially, it also integrates the Low-PHY processing, with FFT computations accelerated on the FPGA fabric. Although the implementation is in a single MPSoC, the proposed solution supports both split option 7.2x and split option 6. In particular, the latter one can be supported without any expected performance degradation. Indeed, in this case both Low-PHY and High-PHY are colocated in the same functional element (i.e., the RU).

In the considered implementation, as illustrated in Fig. 1, the FPGA-based Low-PHY processing performs the functional role typically associated with an Open RAN RU, while the CPU-based High-PHY, MAC, and RLC layers perform DU functionalities. The co-location potentially allows the best performance that, as previously said, could be reachable even in case of functions residing in separate RU and DU hardware units in a physically disaggregated Open RAN deployment (i.e., split option 6). Indeed, this single-chip, integrated gNB approach is a key aspect of our design, specifically aimed at eliminating host-to-accelerator and off-chip communication latencies that can arise between Low-PHY and High-PHY processing in disaggregated systems, thereby enhancing overall system performance and energy efficiency.

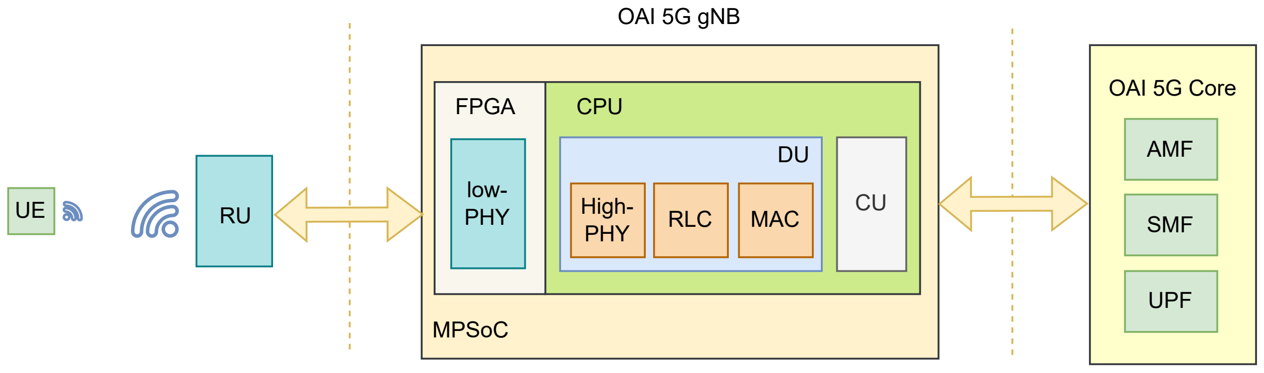

Fig. 1. Accelerated 5G OAI RAN Stack. Integrated MPSoC-based gNB with on-chip RU (Low-PHY on FPGA) and DU (High-PHY on CPU) functionalities.

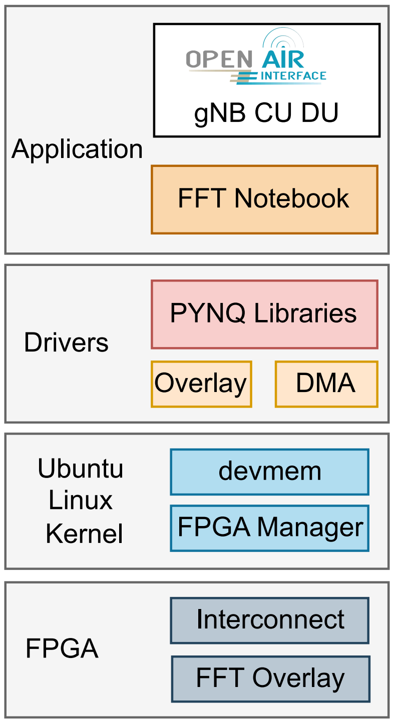

Fig. 2. Accelerated-PHY OAI5G on FPGA.

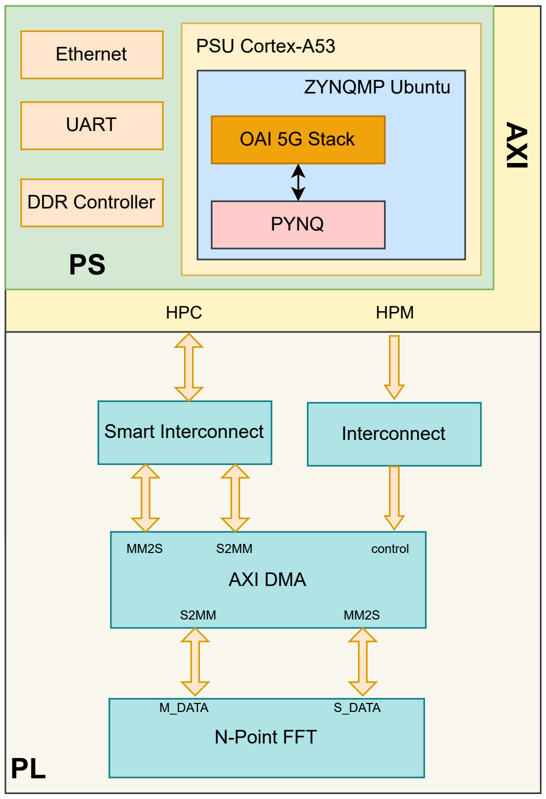

Fig. 3. Developed hardware design on UltraScale+ MPSoC architecture.

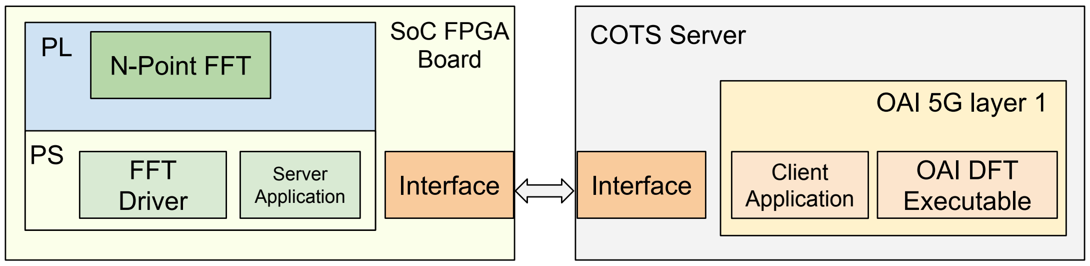

Fig. 4. OAI FFT offloaded to an external FPGA, as discussed in [9].

C. Implementation Details

The proposed MPSoC solution offloads the FFT/iFFT operations from the ARM processor to the FPGA, leveraging hardware acceleration to improve computational efficiency, reduce execution time and lower energy per operation.

To accomplish this, the OAI stack has been deployed in the MPSoC. As shown in Fig. 2, a custom root file system has been created to support the execution of the OAI codes. This includes generating the Boot image to program the FPGA with the FFT IP design and combining it with a custom root file system that was built. The procedure involved importing the FFT hardware design into the petalinux tool and generating relevant boot files by using the petalinux-build command. Additionally, a custom root file system has been developed by using the minimal distribution created by Debian packages and designed for ARM64 architectures. Both the boot and root files have been flashed onto an SD card and inserted into the board in SD boot mode. Furthermore, an application driver has been developed in C to utilize the hardware design. The driver writes data to the Direct Memory Access (DMA) address, which streams it to the FFT and sends it back to the DMA which makes data available for reading by the driver. With providing custom compiling directives, the OAI codes is able to run on the MPSoC CPU, which hosts a custom Debian version. The whole process is illustrated in Fig. 2.

The utilized MPSoC is a Zynq UltraScale+ MPSoC FPGA platform [26], as shown in Fig. 3. This heterogeneous architecture features a dual-core ARM Cortex A-53 processor in the Processing System (PS) and high-performance programmable logic (PL), which enables seamless execution of both software and hardware tasks on a single board. The selection of an MPSoC with respect to an RFSoC is motivated by the fact that the objective of the study is to proof the performance improvements of the performed solution. As depicted in our system architecture (Fig. 1), an external USRP SDR can be utilized for RF operations. This modular approach allowed us to focus specifically on the challenges and benefits of on-chip baseband processing, software stack deployment, and PHY-layer acceleration within the MPSoC environment. RFSoC devices (e.g., the XCZU67DR), which integrate RF data converters, can be also utilized for the realisation of an O-RU featuring the proposed solution. Integrating direct RF capabilities via an RFSoC is a logical next step for an even more compact and potentially more power-efficient system.

In the proposed design, the OAI stack operates natively on the PS, with all gNB protocol stack layers—including Low-PHY—fully integrated. Key peripherals, including Ethernet, UART, and the DDR Controller, are integrated within the PS to support system-level operations and high-level protocol handling. Computationally intensive functions, such as the FFT, are offloaded to the PL using the AMD LogiCORE IP FFT core, which is configured to handle Discrete Fourier Transform (DFT) operations efficiently. The forward FFT output sequence $X'(k)$ is computed as:

$$X'(k) = \frac{1}{s} \sum_{n=0}^{N-1} x(n) e^{-jnk\frac{2\pi}{N}}, \quad k = 0, \ldots, N-1,$$

where $x(n)$ are the input samples, N is the FFT size, and s is the scaling factor. The use of a fixed-point format for input and output samples is for alignment with OAI, where the real and imaginary part of each sample is represented with an integer of 16 bits.

To facilitate seamless communication between the PS and PL, the design employs high-bandwidth Advanced eXtensible Interface (AXI) interconnects, as depicted in Fig. 3, ensuring efficient data transfer and minimizing bottlenecks. Furthermore, the FFT core configuration is highly customizable, allowing it to adapt to different FFT sizes and workloads, providing flexibility for various deployment scenarios.

The choice of memory for buffering these samples is crucial. On-Chip Memory (OCM), for example, offers the lowest latency for exchanges between the PL and PS but its limited size restricts its use to small FFT sizes and low-throughput scenarios. In contrast, for large FFTs or continuous high-rate data streams, DRAM is more suitable due to its higher capacity and sustained bandwidth. Furthermore, as OCM is also utilized by the CPU for critical tasks like stack, code, and interrupts, overloading it with large data buffers can degrade system performance and potentially lead to instability.

IV. Experimental Evaluation

To evaluate the presented solution, we performed a series of experiments focusing on latency, resource utilization, and power consumption. It is important to note that while this integration is functional for the entire stack (Low-PHY, High-PHY, RLC, MAC, CU), the detailed performance breakdown and

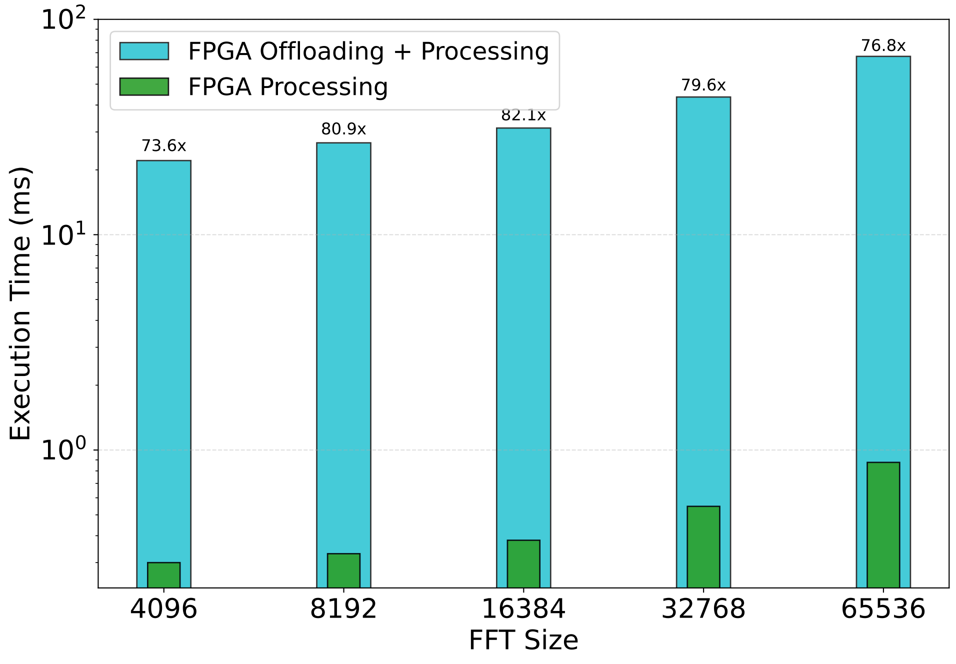

Fig. 5. FPGA FFT execution time breakdown: green bars show pure FPGA processing, while blue bars include data transfer and computation.

optimization results presented in this section focus primarily on the hardware acceleration of the FFT component within the Low-PHY and its direct system impact.

For the latency evaluation of FFT operations (Section IV-A), we included FFT sizes ranging from 4096 up to 65536 points. Although the largest FFT size defined within the current 5G NR standards (e.g., 3GPP Release 18) is 4096, the decision to evaluate larger FFT sizes (i.e., 8192, 16384, 32768, and 65536) in our study serves several key purposes. Primarily, it allows us to rigorously test the performance scalability and demonstrate the processing limits of our MPSoC-based FFT accelerator design across a wide range of computational loads. Furthermore, this exploration aimed to showcase the hardware's capability to support potentially wider transmission bandwidths or different numerologies that might be considered in wireless systems beyond 5G or for specialized OFDM-based applications. For example, in [27] a number of subcarriers equal to 8192 is already considered.

A. Latency

To quantify the impact of the presented solution, we measured the execution time of the FFT operations across different implementation approaches: (i) CPU; (ii) a conventional approach where the FFT function is offloaded to an external FPGA, as shown in Fig. 4; (iii) the proposed MPSoC approach, with a fully integrated implementation on the MPSoC, as shown in Fig. 1. For all cases, timestamps were taken at the function call and return points, allowing us to capture the execution time.

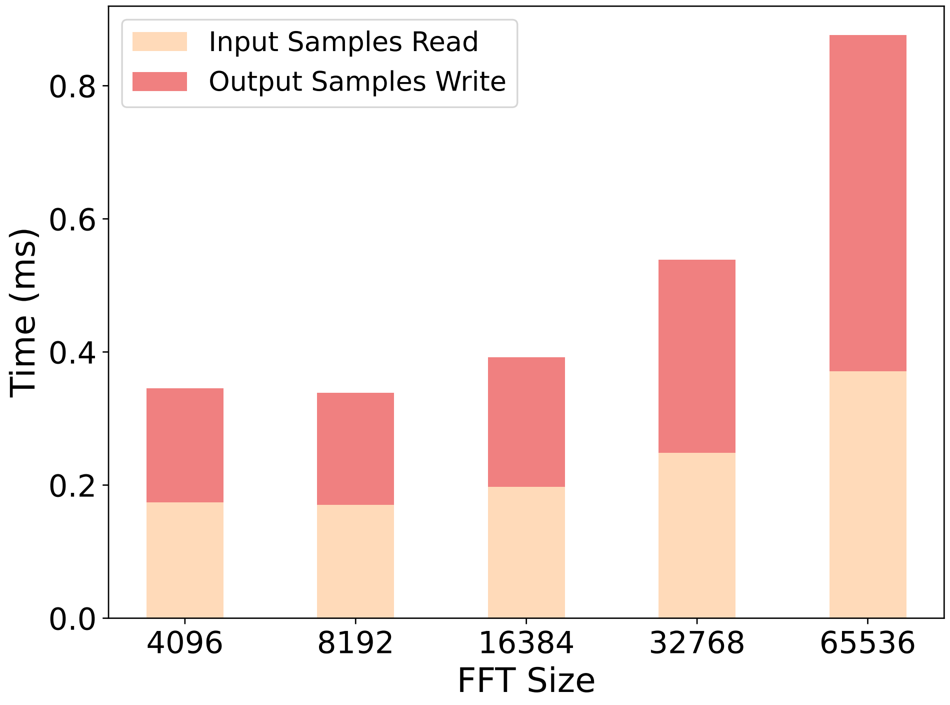

Fig. 6. MPSoC read and write data overhead.

In the external FPGA solution, an undesirable and energy overhead latency are attributed to sample exchange with the COTS server. This is depicted in Fig. 5, reporting the execution time for (i) the pure hardware processing of the FFT on the FPGA (green bars), i.e., the time required by the FPGA itself to compute the FFT once the data is already on-chip, and (ii) the overall offloading process (blue bars), which includes the time for transmitting the input samples to the FPGA and retrieving the results back to the host. The hardware processing time is very small (i.e., less than a millisecond) for all FFT sizes, highlighting the inherent speed of FPGA-based parallel computations. However, the total offloading time increases significantly with the FFT size due to the data transfer overhead. This overhead is substantial enough to dominate the entire offloading process, resulting in (i) a total latency that is nearly two orders of magnitude greater than the raw compute time and (ii) wasted energy during transfers and idle periods.

The behavior differs significantly in the proposed MPSoC-based design, as reported in Fig. 6. Indeed, a latency increase can be observed of only ≈ 0.2 ms (≈ 0.33 ms) going from 4096 to 65536 FFT size in case of read (write) operation.

Fig. 7 then reports the end-to-end execution time for multiple FFT sizes in a full OAI setup. The latency performance of a CPU-based (ARM processor of the MPSoC) OAI FFT (blue bars) is compared with the proposed MPSoC approach (green bars) in which the entire 5G RAN stack runs directly on the

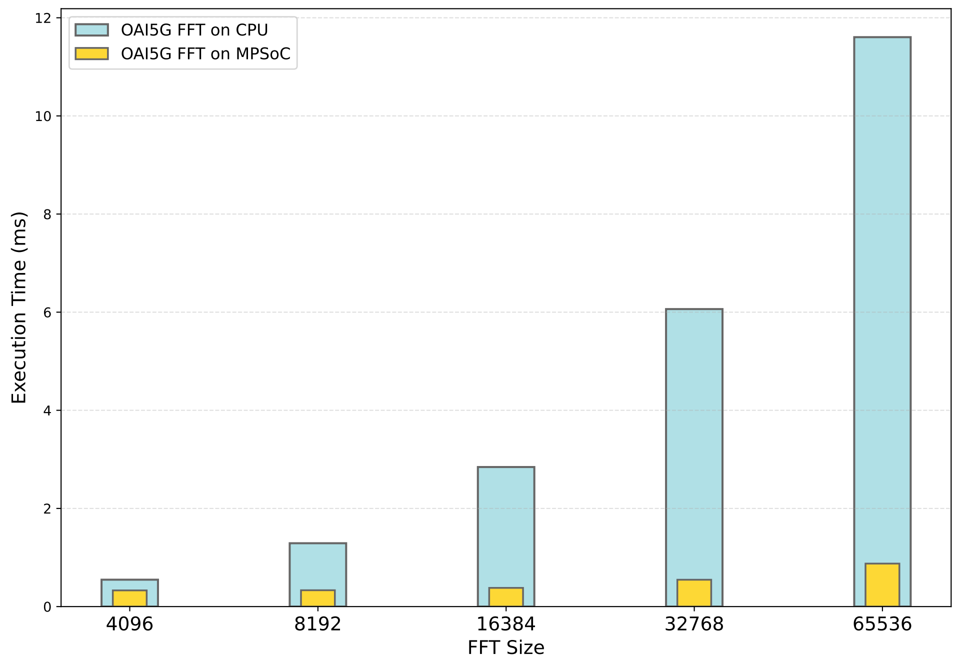

Fig. 7. OAI FFT execution time on CPU and MPSoC.

ARM cores and FPGA within the same chip. As the FFT size increases, the CPU implementation scales poorly, ultimately reaching around ≈ 11.6 ms for 65536-point. In contrast, the MPSoC-based solution shows significantly lower execution times, with improvements becoming more pronounced as the FFT size increases, meaning less active time and thus lower energy consumed for the task.

For instance, at 8192 points, the CPU requires 1.3 ms, whereas the MPSoC implementation completes the computation in 0.3 ms, achieving a speedup of approximately $4\times$. This trend continues for larger FFT sizes, with the 65536-point FFT on the MPSoC showing an acceleration of over $13\times$ over the CPU.

These results confirm that MPSoC integration significantly reduces the processing latency, and, consequently, operational energy for FFT tasks. This is achieved by eliminating the host-to-accelerator communication overhead and by leveraging the parallel processing capabilities of the FPGA fabric.

B. Resource Utilization

In this section, we analyze the resource utilization of our design in terms of LookUp Table (LUT) and registers. Efficient resource utilization can contribute to overall system power efficiency by enabling the use of appropriately sized FPGAs or integrating more functions within a given power budget.

Fig. 8 indicates that the FFT block utilizes the highest number of LUTs and registers. This is indicative of the complexity involved in its arithmetic computations, such as butterfly operations, as well as the

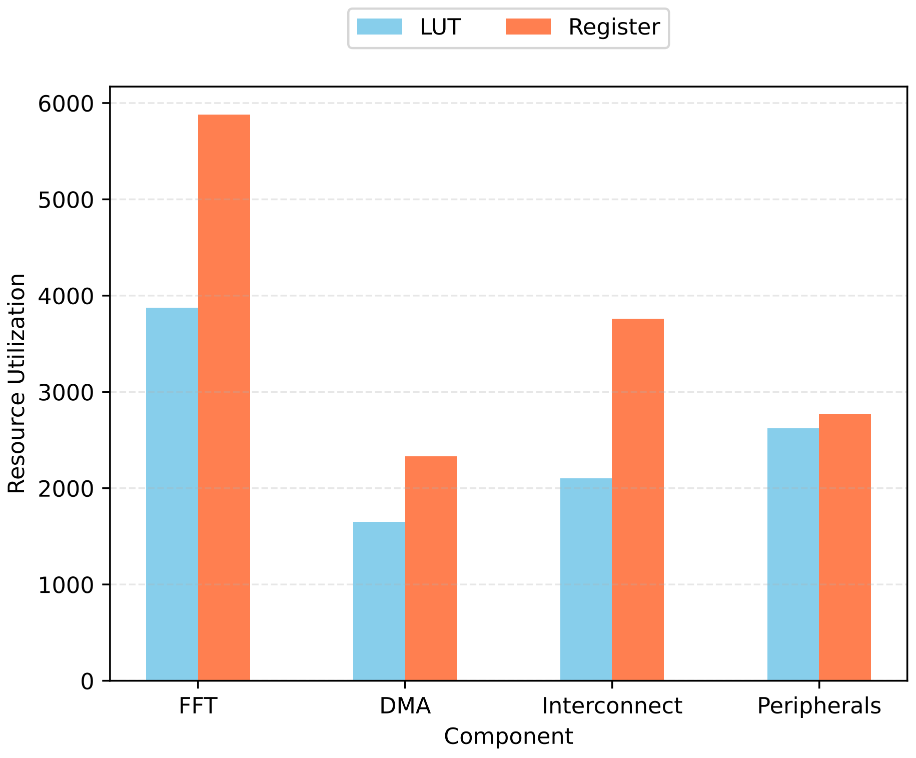

Fig. 8. Hardware resource utilization.

multiple pipeline stages necessary to achieve high-throughput performance. The Interconnect block also shows a significant resource usage because it manages multiple AXI channels to transfer data between the PS and PL. In comparison, the DMA requires fewer LUTs and registers since its primary role is to handle burst-based data transfer, while the peripherals consume a relatively small amount of logic to provide essential support functions (e.g., timers, status registers). Overall, these results confirm that the specialized FFT accelerator and the on-chip network fabric (Interconnect) dominate the hardware footprint, while the DMA and peripherals are more lightweight.

C. Power Consumption

Power consumption measurements were obtained by using Vivado implementation reports, which analyze switching activity, resource utilization, and clock frequencies to estimate dynamic and static power consumption across the PS and PL. The power consumption of the ARM CPU is estimated by using Vivado implementation reports, providing a value in Watts based on operational activity. The energy consumed by the CPU for a specific operation is then calculated by multiplying this estimated CPU power by the operation's measured execution time.

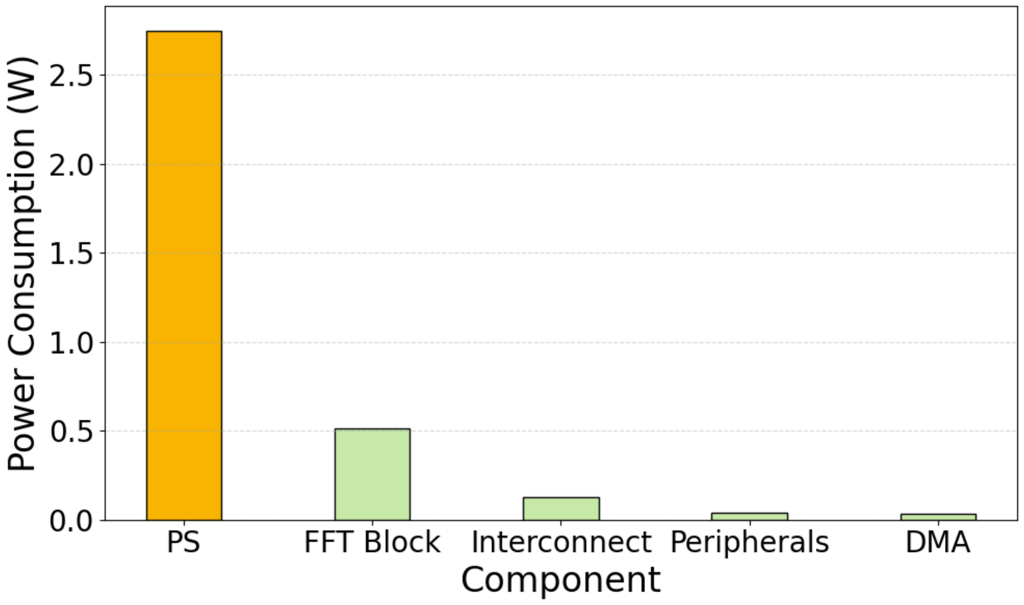

Fig. 9. Power consumption of design components in the PS and PL for FFT. The value for the FFT Block represents its general power draw during processing; the total energy consumed for a specific FFT operation is then determined by this power and the operation's duration (see Fig. 10 for energy details per FFT size).

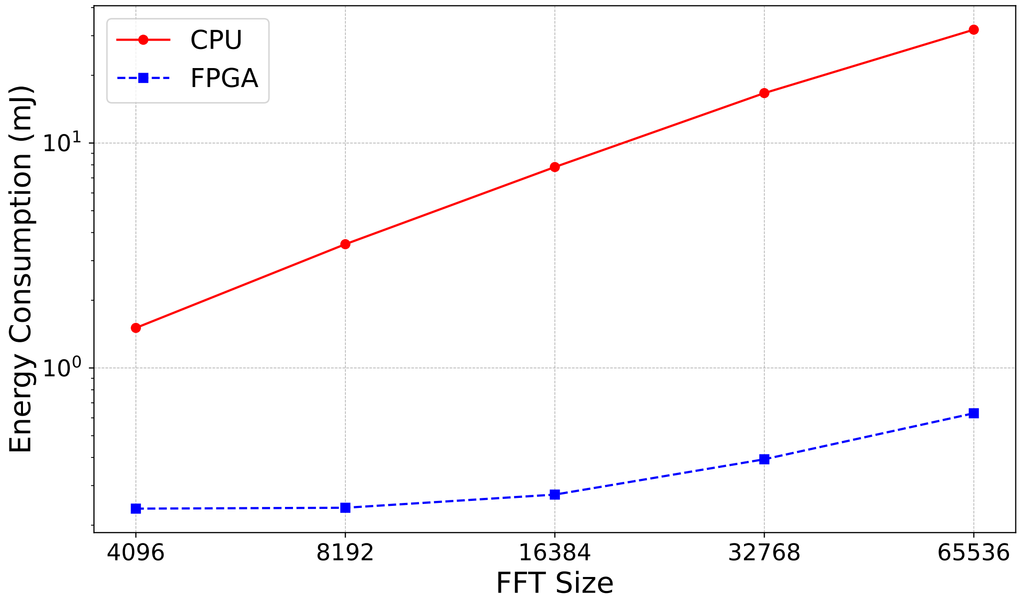

Fig. 10. Energy Consumption per FFT operation (log scale) in CPU and MPSoC FPGA for different FFT sizes.

Fig. 9 shows that PS is the most power-consuming component of the device. This is expected since it includes the CPU cores, memory controllers, and other primary system interfaces. Meanwhile, the FFT block reaches the second-highest consumption, reflecting its intensive arithmetic operations and high clock speed. In comparison, the interconnect, peripherals, and DMA draw significantly less power, indicating that on-chip data movement and support components contribute only a small fraction of the total. Overall, these results highlight that most of the energy budget is spent in the PS and the specialized accelerator, while the remaining logic in the programmable fabric has a comparatively modest power footprint. This substantiates the importance of offloading to the PL for tasks where it is significantly more energy-efficient.

Additionally, to provide detailed insights on the energy usage gains when performing end-to-end low-PHY operations on FPGA instead of CPU, in Fig. 10 we report the energy consumption per FFT operation for different FFT sizes both on CPU and FPGA in logarithmic scale. The energy consumption is computed as the power consumption of the PL multiplied by the execution time of a single operation. Thanks to the inherent efficiency of FPGA technology, these designs operate with significantly lower power consumption compared to CPUs. In fact, the FPGA shows a steady energy consumption over most FFT sizes, increasing for the two largest FFT sizes due to the higher execution time. On the other hand, the CPU usage grows exponentially with increasing FFT size due to the sequential nature of CPU code execution: processing more inputs requires additional clock cycles, contributing to a higher energy consumption per operation. The CPU energy consumption continues to double until it reaches the largest FFT size, resulting in a 50.74x increase in energy per operation compared to FPGA.

V. Conclusion

This work addressed the challenge of efficiently implementing Open RAN architectures, where disaggregation increases flexibility but can introduce performance and energy consumption trade-offs. Indeed, deploying RAN functions as modular software running on COTS hardware components can create inefficiencies, especially for compute-intensive PHY-layer tasks. While offloading to external accelerators improves performance, the data transfer latency and associated energy overhead often negatively impact the expected gains, limiting the effectiveness of these solutions.

To overcome these limitations and promote greener 5G solutions, we propose an MPSoC FPGA-based solution, where gNB Low-PHY and High-PHY functions are deployed in the same chip but in the FPGA and in the processor respectively. In addition, as proof of concept of the feasibility of integrating the entire 5G protocol stack, the remaining upper part of the gNB is deployed in the MPSoC processor. By eliminating host-to-accelerator communication overhead, our approach achieves significant latency reductions while improving the overall system efficiency. Experimental results demonstrate that our MPSoC implementation achieves a $4\times$ speedup for an 8192-point FFT and a $13\times$ speedup for a 65,536-point FFT compared to a CPU implementation, reducing the execution time from 11.6 ms to 0.87

ms for the largest FFT size. Crucially, this acceleration translates to substantial energy savings, with the FPGA consuming up to 50.74 times less energy per FFT operation than the CPU.

While this work demonstrates significant benefits in latency reduction and energy efficiency by integrating FFT acceleration onto the MPSoC, some aspects require further investigation for a comprehensive system understanding. Our implementation in the PS utilizes a custom Debian-based Linux distribution, which is a non-real-time operating system. A detailed analysis of real-time operating system and its influence on deterministic processing for latency-critical RAN functions could valuably extend this work. Additionally, porting RF processing (including Digital FrontEnd functionalities) to an RFSoC system that combines SDR functions with our developed RU-DU-CU system could be a future contribution, enabling a more compact and power-efficient 5G network solution. Finally, evaluating the performance of the upper protocol layers (High-PHY, RLC, MAC, CU) executing on the MPSoC's ARM cores, and understanding any associated processing trade-offs, is also recognized as critical. A comprehensive, layer-by-layer performance characterization of all software-defined gNB components (High-PHY, RLC, MAC, CU) running on the PS might be helful for a complete evaluation.

REFERENCES

[1] NGMN, "6G Use Cases and Analysis," NGMN, Tech. Rep. v1.0, 2022, accessed: 2025-05-28. [Online]. Available: https://www.ngmn.org/work-programme/6g-use-cases-and-analysis.html

[2] S. Giménez-Antón, E. Grasa, J. Perelló, and A. Cárdenas, "6g-rupa: A flexible, scalable, and energy-efficient user plane architecture for next-generation mobile networks," *Computers*, vol. 13, no. 8, p. 186, 2024.

[3] M. Polese, L. Bonati, S. D'oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[4] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and learning in O-RAN for data-driven NextG cellular networks," *IEEE Commun. Mag.*, vol. 59, no. 10, pp. 21–27, 2021.

[5] A. Garcia-Saavedra and X. Costa-Perez, "O-RAN: Disrupting the virtualized RAN ecosystem," *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 96–103, 2021.

[6] SCF, "5G nFAPI specification," SCF, Tech. Rep. 225.3.0, July 2022, accessed: 2025-06-03. [Online]. Available: https://www.smallcellforum.org/technology/5g-fapi-standard/

[7] "O-RAN: Towards Open and Smart RAN," https://www.o-ran.org/, 2020, white Paper.

[8] L. Kundu, X. Lin, E. Agostini, V. Ditya, and T. Martin, "Hardware Acceleration for Open Radio Access Networks: A Contemporary Overview," *IEEE Communications Magazine*, vol. 62, no. 9, pp. 160–167, 2024.

[9] J. C. Borromeo, K. Kondepu, N. Andriolli, and L. Valcarenghi, "FPGA-accelerated SmartNIC for supporting 5G virtualized Radio Access Network," *Computer Networks*, vol. 210, p. 108931, 2022.

[10] A. Bourenane, E. Paolini, N. Andriolli, and L. Valcarenghi, "A Programmable 5G DU-RU SmartNIC based on MPSoC FPGA," in *2024 IEEE 25th International Conference on High Performance Switching and Routing (HPSR)*, 2024, pp. 209–214.

[11] Intel Corporation, “Integrated acceleration for high-performance, energy-efficient vrans: 4th gen intel® xeon® scalable processors with intel® vran boost,” Intel Corporation, Tech. Rep., February 2023. [Online]. Available: https://www.intel.com/content/dam/www/central-libraries/us/en/documents/2023-02/4th-gen-xeon-scalable-vran-product-brief-final.pdf

[12] E. Municio, G. Garcia-Aviles, A. Garcia-Saavedra, and X. Costa-Pérez, “O-RAN: Analysis of Latency-Critical Interfaces and Overview of Time Sensitive Networking Solutions,” *IEEE Communications Standards Magazine*, vol. 7, no. 3, pp. 82–89, 2023.

[13] O-RAN Alliance, “Architecture Principles for a Cloud-Friendly Future 6G RAN Architecture,” `https://mediastorage.o-ran.org/`

[14] E. A. Papatheofanous, D. Reisis, and K. Nikitopoulos, “LDPC hardware acceleration in 5G open radio access network platforms,” *IEEE Access*, vol. 9, pp. 152 960–152 971, 2021.

[15] C. Tarver, M. Tonnemacher, H. Chen, J. Zhang, and J. R. Cavallaro, “GPU-based, LDPC decoding for 5G and beyond,” *IEEE Open Journal of Circuits and Systems*, vol. 2, pp. 278–290, 2021.

[16] C. Ribeiro and A. Gameiro, “A software-defined radio FPGA implementation of OFDM-based PHY transceiver for 5G,” *Analog Integrated Circuits and Signal Processing*, vol. 91, pp. 343–351, 2017.

[17] A. Umesh, T. Yajima, T. Uchino, and S. Okuyama, “Overview of O-RAN Fronthaul Specifications,” *NTT DOCOMO Technical Journal*, vol. 21, no. 2, pp. 46–59, 2019.

[18] T. I. Project, “OpenRAN,” https://telecominfraproject.com/openran/, 2025, accessed: June 4, 2025.

[19] M. Polese, M. Dohler, F. Dressler, M. Erol-Kantarci, R. Jana, R. Knopp, and T. Melodia, “Empowering the 6G Cellular Architecture With Open RAN,” *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 245–262, 2024.

[20] “Study on new radio access technology: Radio access architecture and interfaces,” 3rd Generation Partnership Project (3GPP), Tech. Rep. TR 38.801, 2017, release 15. [Online]. Available: https://www.3gpp.org/DynaReport/38801.htm

[21] L. H. Jamieson, P. T. Mueller Jr, and H. J. Siegel, “Fft algorithms for simd parallel processing systems,” *Journal of Parallel and Distributed Computing*, vol. 3, no. 1, pp. 48–71, 1986.

[22] F. Civerchia, M. Pelcat, L. Maggiani, K. Kondepu, P. Castoldi, and L. Valcarenghi, “Is OpenCL Driven Reconfigurable Hardware Suitable for Virtualising 5G Infrastructure?” *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 849–863, 2020.

[23] M. K. Motalleb, V. Shah-Mansouri, S. Parsaeefard, and O. L. A. López, “Resource allocation in an open RAN system using network slicing,” *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 471–485, 2022.

[24] Comcores, “Part 1: Accelerating 5G virtual RAN deployment,” https://www.comcores.com/wp-content/uploads/2024/08/Comcores-ORAN-Intro-whitepaper.pdf, 2024, accessed: January 28, 2025.

[25] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, “OpenAirInterface: Democratizing innovation in the 5G Era,” *Computer Networks*, vol. 176, p. 107284, 2020.

[26] Z. U. M. Xilinx. (DS891 (v1.10) November 7, 2022) Zynq ultrascale+ mpsoc data sheet: Overview. Accessed on 2025-01-20. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ds891-zynq-ultrascale-plus-overview

[27] L. Electronics, “Method and apparatus for handling dc subcarrier in nr carrier in wireles communication system,” International Patent WO 2018021 676 A1, Feb. 1, 2018. [Online]. Available: https://patents.google.com/patent/WO2018021676A1/en