IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 14, 2021, accepted November 4, 2021, date of publication November 9, 2021,
date of current version November 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3127039

LDPC Hardware Acceleration in 5G Open Radio Access Network Platforms

ELISSAIOS ALEXIOS PAPATHEOFANOUS[1], DIONYSIOS REISIS[2], AND KONSTANTINOS NIKITOPOULOS[1], (Senior Member, IEEE)

[1]Wireless Systems Lab, 5G and 6G Innovation Centre, Institute for Communication Systems, University of Surrey, Guildford GU2 7XH, U.K.
[2]Digital Systems Team, Department of Physics, National and Kapodistrian University of Athens, 157 72 Athens, Greece

Corresponding author: Konstantinos Nikitopoulos (k.nikitopoulos@surrey.ac.uk)

ABSTRACT The Open Radio Access Network (RAN) concept has been gaining wide acceptance in recent network architectures, including 5G New Radio (NR) network deployments. Current Open RAN radio implementation efforts, aim at integrating several white-box hardware elements and executing digital processing on open-source software. When building such a software-based, 5G Open RAN platform, challenges include achieving real-time execution times for demanding computational blocks of the 5G NR physical layer processing, such as Low Density Parity Check (LDPC) decoding. In this context, having already identified both the capabilities as well as the challenges that Field-programmable Gate Arrays (FPGAs) offer for accelerating LDPC, we present our novel LDPC FPGA accelerator system. In this paper, we contribute the implementation details of our FPGA accelerator design as well as the process of integrating the accelerator with OpenAirInterface (OAI), the basis for our 5G NR platform. For the first time in the literature, we show an FPGA-based LDPC accelerator fully integrated with a complete software platform, that is able to achieve more than 1.6*Gbps* decoding throughput and up to 13 times faster execution times compared to single core software implementations. Finally, in our results, we show that LDPC encoding is more challenging to accelerate due to lower computational complexity.

INDEX TERMS 5G New Radio (5G NR), accelerator, field-programmable gate array (FPGA), low density parity check (LDPC), radio access network (RAN).

I. INTRODUCTION

A. BACKGROUND

Open Radio Access Network (RAN) architectures rely on a split of the RAN functionalities, among the Radio Unit (RU), the Distributed Unit (DU) and the Central Unit (CU). The RU acts as the analog and digital front end for the radio signal while the baseband unit (BBU) functions are split between the DU and CU. The DU is responsible for the digital signal processing of the physical layer. One of the most ambitious promises of the Open RAN concept is to enable the implementation of most RAN functionalities on white-box hardware and open-source software, including the physical layer processing on the DU. Several previous works exploit the concept of implementing the physical layer processing of the base station on software and general-purpose processor machines. BigStation [1] aims at supporting multi-user multiple-input multiple-output (MU-MIMO) in 4G Long Term Evolution (LTE) by utilizing several commodity machines while Agora [2] showcases 5G New Radio (NR)-like massive MIMO with a single many-core server. Work in SWORD [3] presents a research-oriented, software-based platform for exploring advancements in the field of massive MIMO. A variety of open-source software projects exists either for LTE or 5G NR, such as OpenAirInterface (OAI) [4]. Closed source software solutions include Intel's FlexRAN [5] and Amarisoft's LTE and 5G NR suite [6].

However, computationally demanding processes such as the Low Density Parity Check (LDPC), the channel coding scheme for the physical uplink shared channel (PUSCH) and physical downlink shared channel (PDSCH) of 5G NR, impose the need for hardware accelerators. Works on software-based processing highlight the need of offloading LDPC decoding to Graphics Processing Units (GPUs) or Field-programmable Gate Arrays (FPGAs) [2], [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Manuel Rosa-Zurera.

The O-RAN Alliance already works on standardizing the interfacing of such hardware accelerators with the software frameworks [7].

B. RELATED WORK

The researchers' focus on processing LDPC in hardware has led to notable results. Ref. [8] introduces a sub-base matrix pruning scheme and the architecture for LDPC, improving the throughput. Ref. [9] presents an improved hard-decision LDPC decoder and explores the trade-off between latency and performance. Both of the above works target 5G NR applications but are not integrated in 5G platforms. Regarding FPGA implementations, [10] and [11] propose novel architectures on Altera FPGAs for improving LDPC decoding in terms of throughput, complexity and performance. LDPC decoders on Xilinx FPGAs are proposed in [12], [13], and [14]. In contrast to the above hardware description languages (HDL) designs, the work in [15] shows LDPC decoders using high-level synthesis (HLS). Even though some of the above FPGA works consider the 5G NR environment, not all of them support all the 5G LDPC configurations. Furthermore, none of them reports the design of a complete accelerator, including integration and evaluation with a complete physical layer processing platform. The GPU decoder for LDPC shown in [16] improves the sum-product algorithm (SPA) and shows FPGAs to be faster but GPUs to prevail with respect to development time. Similarly, [17] targets 5G Cloud RAN improving GPU resources utilization. The above GPU-based LDPC decoders are not integrated or evaluated in a complete physical layer system. An application-specific integrated circuit (ASIC) LDPC decoder, fully supporting WLAN and Wi-MAX is in [18]. Regarding 5G NR LDPC decoding, ASICs are presented in [19] and [20]. Ref. [19] improves the hardware area while it maintains the decoding performance. Ref. [20] proposes the combined min-sum (CMS) decoder achieving low complexity, area efficiency and high throughput. Ref. [21] proposes a 5G NR LDPC encoding ASIC with decreased hardware complexity and increased encoding throughput. None of the above ASIC solutions are reported as part of a complete 5G processing system.

In all the aforementioned works there are no reported integration results for a complete LDPC accelerator system. Efforts towards implementing a complete FPGA-based accelerator are reported in the OAI open-source 5G software project. The FPGA design and driver are provided from the industry and no implementation details are available to the public. Their paper [4] reports a throughput of 300 *Mbps* over the channel with no LDPC configuration specifications. The most complete and notable effort is in [22] introducing a GPU LDPC decoding accelerator for NVIDIA GPUs, integrated with the OAI 5G full-stack software. Its evaluation takes into account not only the processing times but the complete offloading process between the host and the GPU.

The industry offers individual LDPC IP Cores for FPGAs such as the LDPC IP Core of Intel [23] and Xilinx [24]. FPGA solutions are also provided by Creonic [25] and Accelercomm [26] with additional functionalities of the 5G NR LDPC decoding chain. Finally, the NVIDIA Aerial [27] offers Software Development Kits (SDKs) for GPUs in software-based 5G RANs, including LDPC processing.

C. PROBLEM STATEMENT

Previously, our works were focused on our Soft and Open Radio Design for Rapid Development, Profiling, Validation and Testing (SWORD) platform [3] targeting LTE and utilizing the codebase of OpenAirInterface [28]. SWORD serves as a testbed for research, development and validation of advanced concepts for base station physical layer processing [29], [30]. Currently, our efforts focus on extending the platform towards supporting computationally demanding 5G NR modes in real-time operation [31]. In order to address the LDPC challenge we opt for hardware acceleration based on FPGAs given that hitherto published LDPC accelerators were not integrated in a complete software-based 5G system. Consequently, the research question left unanswered are:

1) Which are the challenges and design considerations for efficiently integrating the two platforms together, FPGA and software on a general purpose machine?
2) How much can LDPC encoding and decoding benefit, in terms of execution times, from FPGA offloading, in the context of an entire physical layer system?
3) Where do the bottlenecks lie and how much do they affect the speedup gains that can be achieved?

D. CONTRIBUTIONS

The current paper introduces a 5G NR compliant, FPGA-based, LDPC accelerator system using commercial off-the-self (COTS), open-source and custom developed components, including the Xilinx LDPC IP Core. The accelerator is integrated with the open-source software OpenAir-Interface, which is also the basis of our research platform. For the first time in the literature, we aim to answer the aforementioned research questions by contributing (a) the FPGA design and software integration details, as well as (b) the results that can be achieved from the LDPC accelerator when evaluated in a complete system. This work further promotes research in the field of FPGA-based LDPC processing architectures since it highlights the necessary steps and potential challenges for integrating such novel architectures into complete systems.

This work, is an extension of the FPGA prototype of our previous work in [32]. A more detailed analysis is performed and several additional features are introduced by studying the previously identified bottlenecks and future improvements:

1) We develop an optimized scheme for offloading packs of multiple code blocks in a single PCIe transfer and achieve by extensive pipelining a speedup of 13.5 compared to the single core OAI software decoder; a result improving significantly our previous [32] work.
2) We offload the Cyclic Redundancy Check (CRC) of the 3GPP 5G NR coding chain [33] and achieve savings in

execution times of up to 76.8 *us* compared to the software CRC implementation; these savings are additional to the LDPC acceleration.

3) Moreover, we fully integrate our FPGA accelerator with the full-stack 5G NR OAI software for the base station and validate its functionality and decoding performance.

The paper is structured as follows. First, in Section II we present a detailed analysis of our improved FPGA design of the LDPC accelerator. Then, in Section III we contribute the integration steps of the accelerator with the OAI codebase. In Section IV we validate the FPGA accelerator system in both physical layer as well as full-stack simulations and we present the results that we achieve. Finally, in Section V we indicate future directions before concluding in Section VI.

II. FPGA ACCELERATOR DESIGN

The current section presents the updated FPGA design of our LDPC accelerator based on our initial work in [32]. The high-level architectural overview of both the FPGA-based accelerator and the software framework where it is integrated, is shown in Fig. 1. On the FPGA side, we utilize two readily available components which we integrate together and build all the additional functionalities on top of them. First, for the LDPC encoding and decoding processing we use the commercially available Xilinx LDPC IP Core. Second, the implementation of the Gen2, 8-lane PCIe external interface of the FPGA is based on the Reusable Integration Framework for FPGA Accelerators (RIFFA) [34]. RIFFA has proven to be a satisfying solution so far, utilizing the available PCIe bandwidth efficiently and with the *RIFFA PCIe IP Core* exposing simple and easy to integrate hardware interfaces on the FPGA side, the *Channel Interfaces*.

The newly developed, custom VHDL modules, *LDPC Encoder* and *Decoder Wrappers*, include all the required logic to perform the corresponding LDPC processing. Each of these two modules is connected to its own RIFFA Channel Interface as shown in Fig. 1. The distinct interfaces provide independent and parallel access for both the LDPC encoding and decoding functionalities. In this architectural configuration, RIFFA is responsible for (de)multiplexing data to/from the two Channel Interfaces on the PCIe link.

The FPGA architecture has a streaming functionality where code blocks and their configurations are not stored anywhere in the FPGA, instead they are just propagated from one component to another. To support this organization, individual code blocks or packs of code blocks carry their configuration and status data on a 128-bit header. The above design decisions allow for a minimal latency pipeline, since:

- Buffering is minimized and thus there is no need for external memory which has increased read/write latency when compared to fabric memory.
- Each of the LDPC Encoder and Decoder Wrappers can still accommodate, by using buffers local to the components, the maximum number of code blocks of the maximum size that OAI supports, in a single PCIe transfer. This is 34 code blocks of 26112 bits/Log-Likelihood Ratio (LLR) values each.
- There is no need for storing per-code block parameters, a feature that allows the pipelining of multiple code blocks with different configurations.

In the following two subsections, the VHDL design of the LDPC Encoder and Decoder Wrappers is described in detail.

A. FPGA LDPC DECODER WRAPPER

The LDPC Decoder Wrapper is the VHDL entity responsible for performing all LDPC decoding related processing. Its internal structure is shown in Fig. 2, which also highlights the additions on our previous LDPC prototype in [32]. It includes the Xilinx LDPC IP core and all the logic required to interface it with the rest of the system, as well as to perform additional functionalities. All components are designed to be fully compatible to the AXI4-Stream interface.

The novel modules that have been developed and added to the LDPC Decoder Wrapper are three. First, are the *Pack* and *Un-Pack* components. These modules support multiple code blocks to be offloaded in the same, continuous, PCIe transfer. This allows for better utilization of the PCIe link since PCIe is optimized for larger transfers. Also, a larger number of code blocks can better take advantage of our highly pipelined design and efficiently utilize the available LDPC decoder IP Core throughput. Second novel module is the *Decoder CRC* that accelerates the CRC functionalities of the 5G NR decoding chain. It is depicted in detail in Fig. 3. The *CRC I/O Control* component enables or bypasses the CRC functionalities based on the header of each code block. If CRC is enabled, the data are routed to the *CRC Core*. The CRC Core calculates, 8-bits in parallel, the CRC value of each decoded code block and compares it with the original, decoded CRC. It supports all the polynomials, CRC24A, CRC24B and CRC16 that the specification defines in [33]. The third module is the *RIFFA Channel to AXI4-Stream* of Fig. 2 that has been significantly revised since our previous work in [32]. It translates the RIFFA Channel Interfaces to AXI4-Stream interfaces that the rest of the design uses. It is responsible for initiating and controlling the length of the transmit and receive PCIe data transfers that can now carry several code blocks.

With the use of the above three novel modules the complete flow of the FPGA receiving, processing and transmitting a single code block or a pack of code blocks is as follows:

(a) The first component of the pipeline, the RIFFA Channel to AXI4-Stream module, handles the reception of PCIe transfers via the receive RIFFA Channel Interface. It performs the translation, forwards the incoming transfers to a master AXI4-Stream interface and keeps them in local buffering. Then, in the case of a pack of multiple code blocks in the same PCIe transfer, the Un-Pack module performs the unpacking.

(b) Next is the *In FSM* that handles the three distinct AXI4-Stream input interfaces of the LDPC Decoder

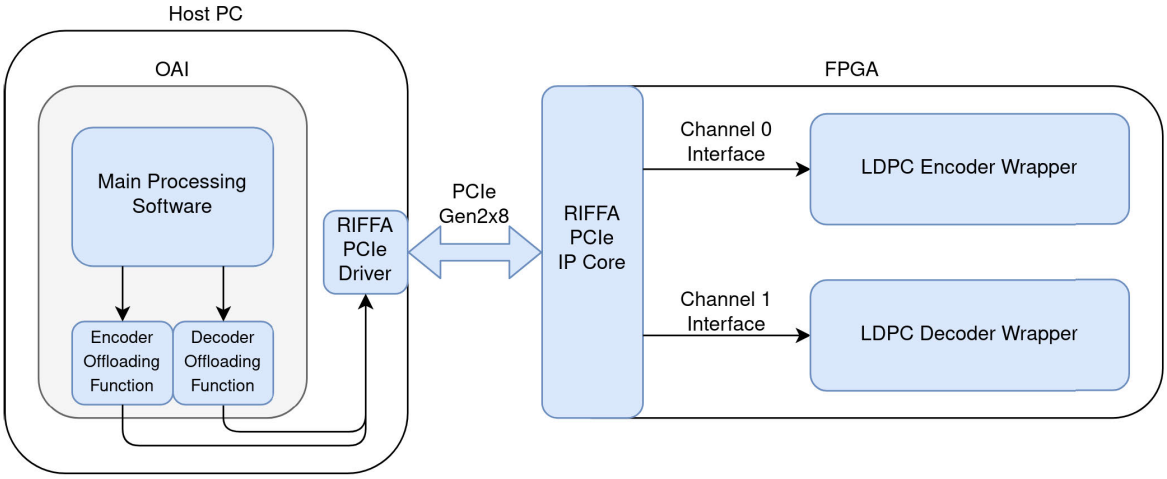


FIGURE 1. Architecture of the FPGA-based LDPC accelerator system.

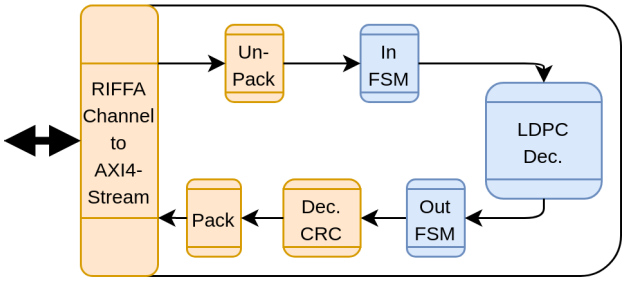


FIGURE 2. Internal architecture of the LDPC decoder wrapper. Highlighted in orange are the components that have been currently developed or completely re-worked compared to our previous work [32].

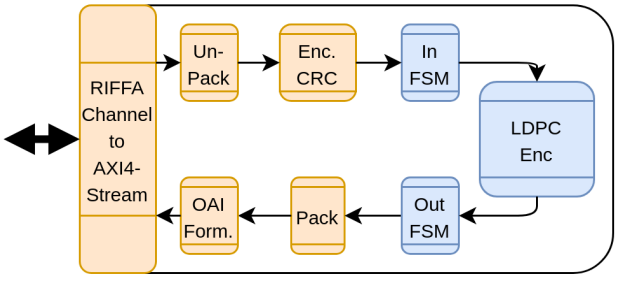


FIGURE 4. Internal architecture of the LDPC encoder wrapper. Highlighted in orange are the components that have been currently developed or completely re-worked compared to our previous work [32].

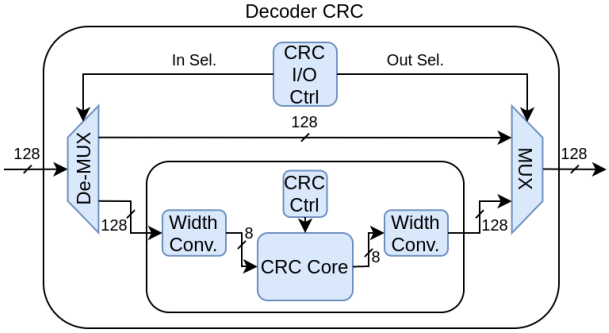


FIGURE 3. Internals of the decoder CRC module.

IP Core. It feeds the header of each code block to the control interfaces while the data that follow are forwarded to the data interface. When the LDPC decoding is complete, the *Out FSM* handles the IP Core's output interfaces. First, it appends the output of the status interface as a header to the outgoing code block and then, it extracts the decoded data from the data interface. These two modules were developed in our previous work [32].

(c) The Decoder CRC module follows, which performs the CRC checking, and stores the result in the position of the original CRC at the end of the code block.

(d) Then, the Pack module performs the code block packing into a single PCIe transfer. Finally, the RIFFA Channel to AXI4-Stream module forwards the PCIe transfer to the transmit RIFFA Channel Interface after translating it from the slave AXI4-Stream interface.

B. FPGA LDPC ENCODER WRAPPER

The LDPC Encoder Wrapper handles all LDPC encoding related functionalities and its architecture is shown in Fig. 4. Similarly to the LDPC Decoder Wrapper, the newly developed Pack and Un-Pack modules regarding the code block packing are included in the design. The *Encoder CRC* module that is additionally developed, performs a part of the 5G NR specified CRC functions. In the case of code blocks that constitute segments of a transport block, the corresponding CRC is calculated, 8-bits in parallel, and is appended at the correct position at the end of each code block. After CRC appending, the number of filler bits that is defined in the 3GPP specification is appended as well. Finally, having identified in [32] the LDPC output representation that OAI requires, we also offload the OAI format matching functionality that was previously being performed in software. The *OAI Format* module re-arranges the encoder IP Core output

of continuous bits, to the format of each bit being represented in a whole byte word as OAI requires. Having analysed the functionalities of the above, newly developed blocks, the flow of data being received, encoded and transmitted back to host, is the same as for the LDPC Decoder Wrapper.

III. OAI SOFTWARE INTEGRATION

The completion of the FPGA-based LDPC accelerator system concludes with the integration into the OpenAirInterface codebase. OAI has been our target software platform of choice from previous works [3], [31] since it has several features that make it fit our needs for building a research-grade platform. It is (a) entirely software-based which makes it appropriate for the Open RAN concept, (b) open-source and thus suitable for research purposes, (c) modular enough to add/remove components with limited effort and (d) closely following the 3GPP 5G NR specification.

For interfacing the host workstation with the FPGA accelerator over PCIe, the RIFFA framework and driver is used. On top of the RIFFA software, for integrating the main OAI processing application with the functions that RIFFA provides, the custom *Encoder* and *Decoder Offloading Functions* are developed. Each of the Offloading Functions is bound to the corresponding RIFFA Channel Interface that the LDPC Encoder or Decoder Wrapper resides on the FPGA side. The two different RIFFA Channel Interfaces can be accessed at the same time by different threads thus allowing independent transfers of data to and from the LDPC Encoder and Decoder Wrappers on the FPGA. The entire architecture is illustrated in Fig. 1. Both the Encoder and Decoder Offloading Functions perform similar functionalities. First, they formulate the 128-bit header that is attached to each code block or pack of code blocks. To do so, they map already existing OAI parameters to the format that the LDPC IP Core requires and calculate additional ones regarding the number of code blocks to be offloaded, the sizes of the PCIe transfers, the CRC offloading functionalities, etc. Code block packing follows where multiple code blocks are arranged inside the buffer to be transmitted by means of memory copies. The RIFFA function call for transmitting the buffer to the FPGA, over PCIe is then performed. Even though the transmit function only returns after the transmission is complete, the system is designed in order for the reception to be initiated from the FPGA side and start even before transmission is complete. This allows for a highly pipelined operation in the case of large number of code blocks in the same PCIe pack, where different code blocks of the pack can be at the same time in the transmit, the LDPC processing and the receive pipeline. The Offloading Functions then call the RIFFA receive function, and when the reception is complete, the resulting packed code blocks are copied to the output buffers. Finally, useful status parameters are extracted from the header of each received code block, such as the number of LDPC decoding iterations performed or the result of the decoded CRC checking. The above development of the Offloading Functions leads to the FPGA-based accelerator supporting all the available combinations of LDPC parameters at run-time, in an automated away and thus being fully 5G NR compliant. Furthermore, caution has been taken to minimize expensive software operations such as memory copies in order to keep the FPGA offloading overheads to the absolutely necessary ones.

Apart from the development of the custom Offloading Functions, complete integration with OAI includes modifications in the LDPC data formats. First, while both the OAI and Xilinx LDPC decoder accept LLR values of 8-bit size as their input, the Xilinx decoder requires that the values are saturated to 6-bits and have their sign inverted. To achieve this, the OAI code for performing the existing LLR 8-bit saturation is extended with Single Instruction Multiple Data (SIMD) optimizations resulting in negligible overhead for the additional operations. Second, the output representation of the OAI LDPC encoder accommodates a single encoded bit inside a byte word. To avoid re-arranging the output of the IP Core to the above format in software, by means of expensive bit-shifting operations, the format matching functionality is now being performed on the FPGA side.

IV. RESULTS AND ANALYSIS

In this section, we are presenting the implementation and integration results of our FPGA-based LDPC accelerator. The FPGA accelerator design of section II is implemented on a Xilinx VC707 development board which features the Virtex-7 XC7VX485T-2FFG1761 FPGA. The entire FPGA development and implementation workflow is done within the Xilinx Vivado software suite. The accelerator is physically mounted on a Gen 2 $\times$ 8 PCIe slot of a Linux workstation. The workstation is operating Ubuntu with a low-latency kernel and has an 18-core Intel(R) Core(TM) i9-7980XE CPU @ 2.60GHz and 64GB of RAM.

A. FPGA DESIGN IMPLEMENTATION RESULTS

All the user interfaces of the PCIe components as well as the LDPC encoder and decoder wrappers have achieved an operating frequency of 250MHz in the FPGA. The impact of the entire design on the fabric resources of the FPGA is shown in Table 1.

TABLE 1. FPGA resources utilization.

Resource	Utilized	Available	Utilization
LUT	98974	303600	32.60%
LUTRAM	7284	130800	5.57%
FF	96056	607200	15.82%
BRAM	407	1030	39.51%

The utilization is kept low to medium for most of the resource types, with the LDPC Decoder IP Core taking up the largest share of it. The above results indicate that it is possible to include more processing blocks inside the FPGA in the future. These could include more functionalities of the LDPC coding chain of 5G NR such as (de)segmentation, rate-(de)matching and (de)interleaving.

Also, given the availability of more independent RIFFA Channel Interfaces, entirely independent blocks could also potentially be offloaded. These could include more instances of LDPC Encoder/Decoder Wrappers for increased LDPC throughput or other demanding physical layer processing blocks such as Fast Fourier Transforms (FFTs) or Inverse FFTs (IFFTs).

Regarding power, we use the power estimation tool of the Xilinx Vivado software suite to provide a power measurement for the entire accelerator design regardless of run time parameters such as LDPC configuration, design idle time, etc. The tool reports a total on-chip power of 5.897W. Out of this, 0.332W is attributed to static power while 5.565W is dynamic power due to the design activity. The 5.565W of dynamic power is distributed among the different FPGA components as shown in Table 2.

TABLE 2. FPGA dynamic power distribution.

Component	Power(W)	Percentage
RIFFA PCIe IP Core	3.295	59%
LDPC Dec. Wrapper	1.822	33%
LDPC Enc. Wrapper	0.446	8%

We observe that the absolute largest amount of power is attributed to the RIFFA PCIe IP Core component, and to its internal FPGA transceiver in particular, that handles the PCIe interfacing of the accelerator. This finding should be taken into consideration when designing and evaluating the power of individual LDPC processing architectures on FPGAs. Significant power overheads can occur when such an architecture is integrated into a complete system.

B. VALIDATION AND DECODING PERFORMANCE RESULTS

After integrating the FPGA accelerator with the OpenAirInterface codebase for the base station processing, we validate its functionality in terms of correctness. To do so, first, we utilize the physical layer simulators that OAI provides. In *nr_dlsim* and *nr_ulsim* simulators, the FPGA accelerated encoder and decoder are validated respectively, for all the available 5G NR Modulation Coding Scheme (MCS) and number of resource blocks (RBs) values that OAI supports. Additionally to the physical layer validation in previous work [32], the FPGA accelerator has now also been verified in the context of the entire 5G NR stack by using the *RF Simulator* full-stack tool that OAI provides.

Regarding LDPC decoding performance, the Xilinx LDPC IP Core that we use for the FPGA decoding, is implementing normalized and offset min-sum decoding algorithms. The above algorithms offer increased decoding performance since they constitute a better approximation than the min-sum approximation that the OAI software decoder utilizes. In the context of our system, we have experimentally identified the normalized min-sum algorithm, with a normalization value of 0.75, to yield the best performance results. The Xilinx LDPC IP Core user guide provides an extensive evaluation

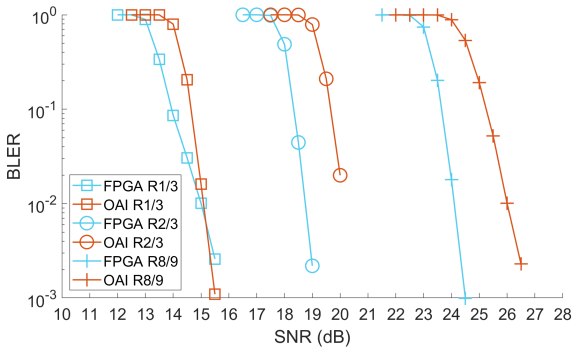


FIGURE 5. FPGA-based decoding performance against OAI decoder. Results presented are for QAM64, code block size $K = 8448$, 5 decoding iterations and all supported code rates of base graph 1, $R = 1/3, 2/3$ and $8/9$ correspondingly.

of the decoder performance with regards to several LDPC parameters and our FPGA implementation does not alter this performance in any way. Still, in order to justify our choice with numbers and show that there is no sacrifice in decoding performance, we provide an indicative comparison of the decoding performance of the OAI and FPGA implementations. Measurements taken include the entire physical layer stack in order to be able to extract conclusions regarding the decoding performance in the context of a complete system. For the same reasons, we present the results of the decoding performance in terms of Block Error Rate (BLER) instead of Bit Error Rate (BER), if a single code block does not pass the CRC check then it is reported as erroneous. In Fig. 5 we show the decoding performance for both the FPGA-based decoder and the OAI software decoder for selected configurations. We observe that, for 5 decoding iterations, for the code rate of $R = 8/9$ there is a performance advantage of about $1.5dB$ of the FPGA decoder. As the code rate becomes lower and the number of available parity bits increases the decoding performance of the two implementations slowly converges. Results are consistent with the findings in [35] where the min-sum OAI decoder is significantly outperformed by normalized min-sum and offset min-sum decoders.

C. SPEEDUP AND EXECUTION TIME RESULTS

In order to evaluate the latency and throughput capabilities of our FPGA-based LDPC accelerator, we perform execution times benchmarking by utilizing the *ldpctest* testbench that OAI provides. All FPGA related execution time measurements take into consideration the entire offloading process and include, host-to-FPGA, LDPC processing and FPGA-to-host times. All OAI software execution time measurements are extracted by executing the LDPC encoder and decoder on a single core of the host general purpose machine. The LDPC decoding throughput numbers presented for our FPGA accelerator refer to decoded throughput. In contrast to encoded throughput that also includes parity bits, decoded throughput takes into account only the decoded, original, information bits which are the ones that matter to the end user.

It is calculated as

$$Throughput = \frac{CBs \times K}{T_{dec}} \quad (1)$$

where K is the code block size, CBs the number of decoded code blocks and T_{dec} the execution time required to decode all the aforementioned code blocks.

1) LDPC DECODER ACCELERATION RESULTS

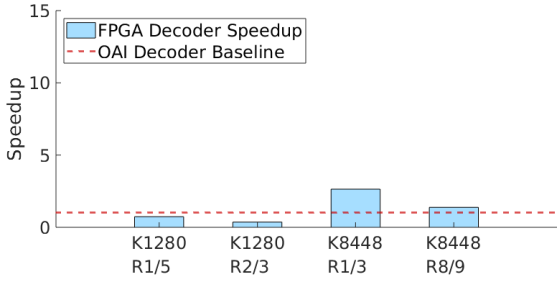
Regarding LDPC decoding in OpenAirInterface, the LDPC parameters that affect the computational complexity and thus the execution times of the OAI software decoder are:

- Code block size K that includes all the original, uncoded, information bits.
- Code rate R, that indicates the ratio of original information bits K to the total number of encoded bits N that also include parity bits. For a given K, a smaller R means more parity bits to decode.
- Base Graph (BG) that is inferred based on K and R as defined in the 5G NR specification [33].
- Number of iterations for the decoding algorithm that can affect the complexity in a linear fashion.
- Number of distinct code blocks (CBs) to be decoded. If the original message is larger than the maximum CB size of $K = 8448$ it has to be segmented into several CBs.

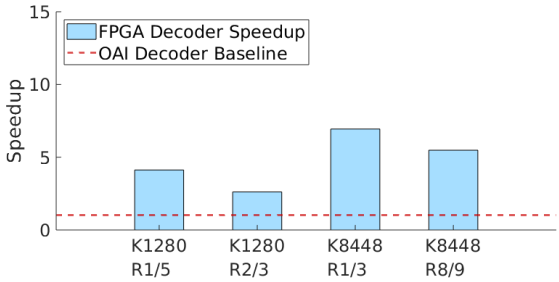
Additional results and a detailed quantitative analysis regarding the above relation of LDPC parameters to the OAI software decoder implementation can be found in [35].

Based on the above explanation of LDPC parameters, in Fig. 6 and 7 we present extensive speedup results of the FPGA-based LDPC decoder, when compared with the software OAI LDPC decoder, at 2 and 5 decoding iterations respectively, and for 4 different numbers of code blocks packed in the same PCIe transfer. The selected LDPC configurations presented include the largest code block size, $K = 8448$, of BG1, a relatively small code block size, $K = 1280$ of BG2 and the minimum and maximum code rates, $R = 1/5, 2/3$ and $R = 1/3, 8/9$, of both BGs, correspondingly. The results of both figures, for offloading a single code block, validate the findings of our previous work in [32] where not all LDPC configurations are favoured by FPGA offloading. Smaller code block sizes and small number of iterations result in less computations where hardware acceleration can be overshadowed by the overheads of the PCIe data transfers. However, in the context of high throughput 5G NR configurations where the number of code blocks increases due to increased QAM modulation order and increased number of allocated resource blocks, the picture changes drastically.

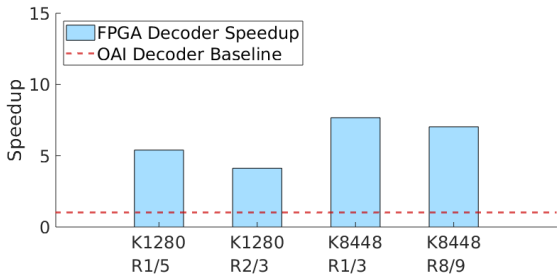
Our latest improvements of packing multiple code blocks in a single PCIe transfer allow for a more efficient utilization of the available PCIe bandwidth and pipelining capabilities of our FPGA design, which in turn, results to significant speedup gains. From the 2 figures, the most significant results occur for 5 decoding iterations. For the most computationally demanding decoding configuration of 5 iterations, $K = 8448$ and $R = 1/3$, the speedup plateaus after the number of offloaded code block increases to 10 due to approaching the maximum throughput of the LDPC decoder IP Core. It is worth noting, that for the less demanding code rate of 8/9, at 5 iterations, our pipelined offloading design can efficiently utilize the increased LDPC IP Core throughput as the number of code blocks increases, reaching a speedup of 13.5.



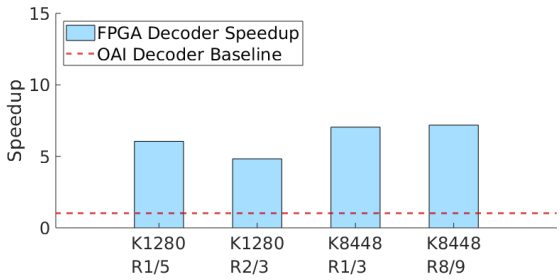
(a) LDPC Decoding of 1 CB at 2 decoding iterations



(b) LDPC Decoding of 10 CBs at 2 decoding iterations

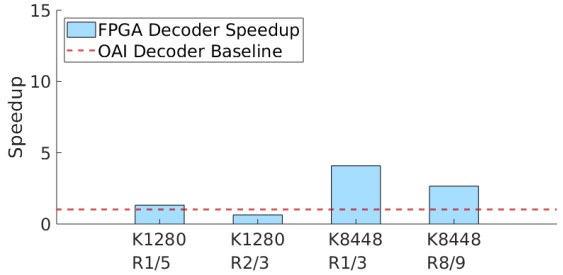


(c) LDPC Decoding of 25 CBs at 2 decoding iterations

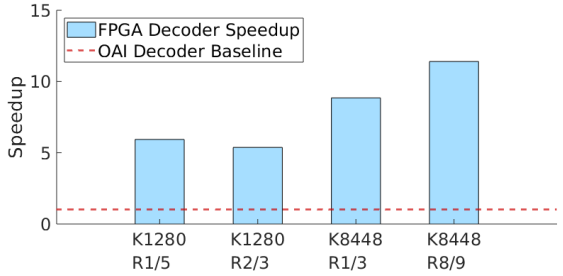


(d) LDPC Decoding of 34 CBs at 2 decoding iterations

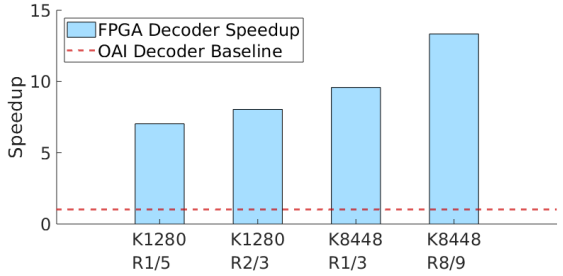
FIGURE 6. FPGA accelerated LDPC decoding speedup, when compared to OAI software decoder, for 2 decoding iterations. The configurations include the minimum and maximum code rates of selected code block sizes of K = 1280 and 8448, of both BGs and for number of code blocks (CBs) 1, 10, 25, and 34. In previous work [32] only the results for 1 CB were extracted.



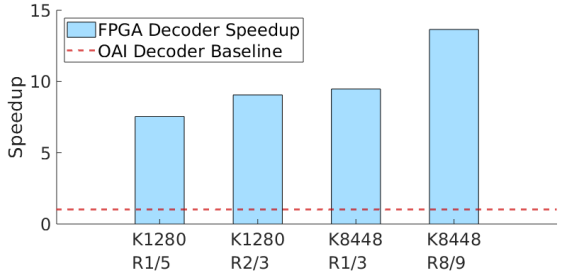
(a) LDPC Decoding of 1 CB at 5 decoding iterations



(b) LDPC Decoding of 10 CBs at 5 decoding iterations



(c) LDPC Decoding of 25 CBs at 5 decoding iterations



(d) LDPC Decoding of 34 CBs at 5 decoding iterations

FIGURE 7. FPGA accelerated LDPC decoding speedup, when compared to OAI software decoder, for 5 decoding iterations. The configurations include the minimum and maximum code rates of selected code block sizes of K = 1280 and 8448, of both BGs and for number of code blocks (CBs) 1, 10, 25 and 34. In previous work [32] only the results for 1 CB were extracted.

In order to highlight the advantages of our FPGA accelerator, in Table 3, we isolate the best performing LDPC decoding configurations out of the extensive benchmarking results presented in Figures 6 and 7. We showcase execution times for both the FPGA decoder and the OAI software decoder as well as the throughput that the FPGA decoder achieves. The configurations presented are:

TABLE 3. FPGA LDPC decoding best metrics & configurations.

Conf.	FPGA(us)	OAI(us)	Speedup	Throughput(Gbps)
(a)	174.0	1.2×10^3	6.9	1.650
(b)	810.6	7.7×10^3	9.5	0.354
(c)	222.0	3.0×10^3	13.5	1.294

(a) 2 Iterations, 34 CBs, $R = 8/9, K = 8448$ with the highest decoding throughput of 1.650*Gbps*.
(b) 5 Iterations, 34 CBs, $R = 1/3, K = 8448$ with the largest reduction in absolute execution time of $6.9 \times 10^3 us$ when compared to single core OAI decoder.
(c) 5 Iterations, 34 CBs, $R = 8/9$, $K = 8448$ with the greatest speedup of 13.5.

The results comparison with the software OAI decoder is done for a single core but modern systems utilize several general-purpose processor cores to decode multiple blocks in parallel. This, in the best case, can reduce the execution time of processing multiple code blocks down to the time of processing a single code block. Still, our results indicate that the latency of our FPGA LDPC decoder accelerator can potentially compete against such multi-core implementations as well. For example, for decoding a single code block of $K = 8448, R = 1/3$ with 5 iterations the OAI software decoder requires 252.3*us*. Our FPGA accelerator can decode 8 code blocks of the same configuration in less time, at 211.8*us*. A trade-off analysis between number of available cores and latency/throughput requirements can be performed in a context specific to each system.

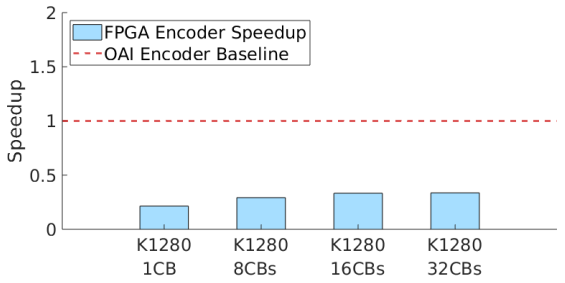
Finally, on top of the significant speedups that we have shown for LDPC decoding, our newest additions of offloading the CRC calculations further reduce execution times. To evaluate this, we compare the CRC execution times between the OAI software and the FPGA-based implementation of the CRC calculation and checking. In Table 4 we summarize the results for two different configurations:

(i) MCS 28 and 106 RBs, resulting in 10 code blocks and decoded in 5 iterations.
(ii) MCS 28 and 273 RBs, resulting in 25 code blocks and decoded in 5 iterations.

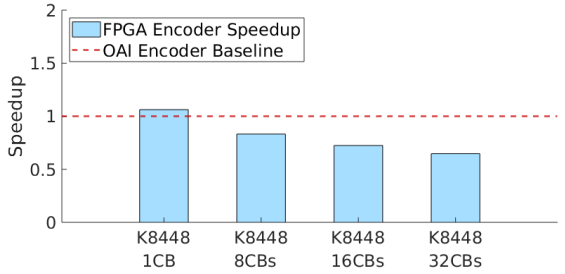
TABLE 4. CRC offloading execution times.

Config.	Platform	Offloading (us)	Post Dec. (us)
(i)	FPGA CRC	91.01	0.14
	SW CRC	80.37	34.16
(ii)	FPGA CRC	180.78	0.19
	SW CRC	171.00	86.57

When the CRC functionalities are performed in software, the FPGA offloading time includes only LDPC decoding while the software post decoding time includes CRC calculation, extraction of decoded CRC and comparison. When the CRC functionalities are performed in the FPGA, the offloading time includes both decoding and CRC calculation and comparison whereas the software post decoding time only includes the extraction of the CRC comparison result.



(a) LDPC Encoding of 1,8,16 and 32 CBs of code block size $K = 1280$.



(b) LDPC Encoding of 1,8,16 and 32 CBs of code block size $K = 8448$.

FIGURE 8. FPGA accelerated LDPC encoding speedup when compared to OAI software encoder. Results are shown for two different code block sizes, $K = 1280, 8448$ and for 1,8,16 and 32 code blocks (CBs). In previous work [32] only the results for 1 CB were extracted.

We observe that the execution times savings in the post decoding time are proportionate to the number of code blocks. Given the small latency penalty in the total offloading time of about $10us$ when offloading CRC to the FPGA, the maximum reduction in execution time for configuration (ii) is $76.8us$.

2) LDPC ENCODER ACCELERATION RESULTS

Our results in [32] for encoding a single code block indicated that offloading might not be favourable, especially for smaller code block sizes. Given our newest additions to our FPGA accelerator where we can offload several code blocks at once, and process them in a pipelined manner, we re-evaluate the execution times and speedup achieved. In Fig. 8, we show the speedup results for our FPGA LDPC encoder, for code block sizes $K = 1280, 8448$ and four different numbers of code blocks when compared to the OAI software encoder.

Our latest results once again validate that offloading smaller code block sizes for LDPC encoding is not efficient. Additionally to our previous work, we now show that even when offloading multiple code blocks of the maximum size, $K = 8448$, the software-based OAI encoder starts outperforming the FPGA accelerator. There are several reasons for this. First, the OAI software LDPC encoder is highly optimized for processing several code blocks at once and this explains the reduction in speedup, for the large code block size $K = 8448$, as the number of code blocks increases. The second reason, is the much less computationally demanding nature of LDPC encoding where the software execution times are considerably lower compared to the LDPC decoding. The above leads to any acceleration on the FPGA being

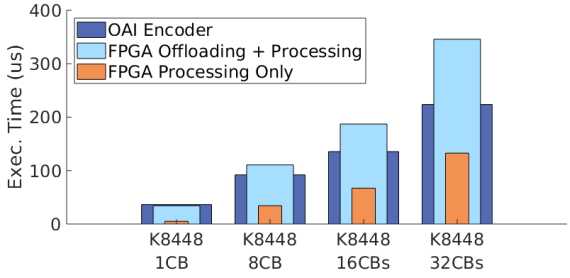


FIGURE 9. OAI LDPC encoder, FPGA encoder offloading and processing total, and FPGA processing only execution times for code block size $K = 8448$ and for 1,8,16 and 32 code blocks.

overshadowed by the PCIe data transfers. This issue is highlighted in Fig. 9 where we present detailed execution time results for both the FPGA and OAI encoder. We observe that from the FPGA offloading plus processing total time, only a relatively small amount is attributed to the FPGA encoder processing. The rest is offloading overheads which can take up to 62% of the total time for encoding 32 code blocks of the maximum size $K = 8448$. The PCIe transfers between the two platforms are identified to having the largest contribution to these overheads. In particular, the FPGA-to-host transfer, is observed to constitute a throughput bottleneck by having, in the worst case, 24 times larger size than the host-to-FPGA transfer due to the inclusion of parity bits and the representation of one encoded bit with an 8-bit word. With the overhead of data transfers being identified as the main obstacle for achieving meaningful acceleration for the LDPC encoding process, we propose that encoding accelerators should implement newer PCIe generations with higher number of lanes, than the Gen 2×8 used in the current work. This will result in more available PCIe bandwidth.

3) COMPARISON TO RELATED WORK

In order to provide additional context to our work, in this subsection, we compare the latency and throughput results of our FPGA-based accelerator with selected, notable related publications that target different hardware platforms for 5G NR LDPC decoding. Some of the comparisons are only indicative and serve the purpose of just showcasing the current state-of-the-art results in LDPC decoding. Direct comparisons for drawing hard conclusions on a fair basis between different implementations in the literature can be a challenging task to perform for several reasons:

- Most related works perform standalone evaluation of their LDPC architectures. Our work takes into account overheads from integration of our accelerator into a complete, software-based 5G NR platform and evaluates the execution time of the entire offloading process.
- Available works in the literature often provide limited results in terms of different LDPC configurations (code block size, code rate, etc.) or even omit mentioning their configurations at all.
- Benchmarking methodology and measurements are not always suitable for performing comparisons or are not described in enough detail.

TABLE 5. LDPC decoders comparison.

	Platform	Config.	Latency(us)	Throughput(Gbps)
This Work	Xilinx VC707	# CBs = 1, # Iter. = 5 R=1/3, K=8448	61.65	0.137
This Work	Xilinx VC707	# CBs = 25, # Iter. = 5 R=1/3, K=8448	608.40	0.347
This Work	Xilinx VC707	# CBs = 34, # Iter. = 2 R=8/9, K=8448	174.00	1.650
[22]	NVidia Titan RTX	# CBs = 1, # Iter. = 5 R=1/3, K=8448	87.00	0.290
[22]	NVidia Titan RTX	# CBs = 20, # Iter. = 5 R=1/3, K=8448	700.00	3.964
[19]	TSMC 40 nm CMOS ASIC	# CBs = N/A, # Iter. = 6 R=11/13, K=1536	N/A	10.860
[20]	TSMC 65 nm CMOS ASIC	# CBs = N/A, # Iter. = 10 R=1/3, K=1232	N/A	3.040
[12]	Xilinx XCKU060	# CBs = N/A, # Iter. = 10 R=1/5, K=3840	N/A	0.071

- Different platforms offer different trade-offs such as power consumption, resources utilization, etc., that are not highlighted when performing an execution time and throughput only comparison.

With the above considerations in mind, in Table 5, we present the decoding latency and throughput results of our work along with indicative, related ones for different implementation platforms. As many LDPC parameters as possible are included for each table entry. In some cases, missing parameters have been inferred from the available parameters of each publication, to the best of our knowledge.

In [22], a GPU LDPC decoding accelerator is presented. Similarly to our work, the accelerator is fully 5G NR compatible and integrated with the OAI software. We observe that for decoding 1 CB with the exact same configuration, the decoding latency measurement that the GPU provides is 87*us* while the FPGA provides 61.65*us*. The FPGA also has an execution time advantage when decoding 25 CBs in 608.40*us* compared to the GPU that requires 700*us* for decoding 20 CBs (the closest available number of CBs). However, the FPGA decoder lacks in throughput in the above two cases as well as in the maximum throughput of 1.6*Gbps* achieved. Still, it is not clear if the measured throughput for the GPU refers to encoded or decoded throughput. Our FPGA accelerator throughput is calculated as the decoded throughput, taking into account only the decoded information bits. Finally, GPUs are known to consume more power than FPGAs in many applications but the authors do not provide an exact measurement to verify this. In [19] and [20], the ASICs that are proposed achieve a maximum throughput of 10.860 and 3.040*Gbps* correspondingly but without any latency measurement. ASICs are known to operate at higher frequencies and consume less power than FPGAs, at the cost of no flexibility and reprogrammability that FPGAs offer. In contrast to our work, the above two ASIC implementations do not support all the 5G NR LDPC configurations and their throughput is evaluated standalone without integration in a complete system. Finally, in [12] an FPGA architecture is proposed for LDPC decoding, targeting Xilinx Kintex Ultrascale+ devices. The authors provide a measurement of 71*Mbps* of decoded throughput for one particular LDPC configuration and for a standalone evaluation of their design.

V. FUTURE WORK

Based on the results that we presented in the previous section, we discuss on potential future improvements of our FPGA-based LDPC accelerator system. First, due to the increased interest in hardware acceleration for LDPC, more specialized hardware is becoming available. The Xilinx Zynq Ultrascale+ FPGAs include the Soft Decision Forward Error Correction (SD-FEC) [36] hardened blocks that are dedicated only for LDPC and Turbo codes processing at 667MHz. Such an FPGA is integrated into the Xilinx T1 Telco Accelerator card [37] that also supports newer PCIe generations and increased number of lanes. Porting our design in such state-of-the-art boards has the potential to further improve latency and throughput results. Another way forward with regards to better execution time results, would be to offload consecutive functionalities of the 5G NR specified coding chain [33] to the FPGA. Such functionalities include (de)segmentation, rate-(de)matching and (de)interleaving and offloading them could result in significant acceleration as profiling results of our previous work show that they contribute a non trivial amount of the total physical layer processing time [32]. Given the wide acceptance that the O-RAN Alliance specifications have been gaining, it could be beneficial to claim such compliance. For interfacing with hardware accelerators, the O-RAN Alliance is considering adopting the Wireless Baseband Device (BBDEV) library. Adding support for BBDEV interfacing to the FPGA accelerator would require development of additional VHDL components to handle all the operations that BBDEV defines, other than just transmitting/receiving to/from a queue. Finally, given that modern general purpose machines can include several tenths of cores it would be worth performing a trade-off analysis that takes this into account. Multiple LDPC code blocks can be processed by different software threads and thus the total LDPC decoding execution time can potentially be reduced down to

the decoding of a single block. Thus, given the number of cores that can be allocated for performing LDPC decoding, the LDPC configurations that favour FPGA offloading might differ.

VI. CONCLUSION

In the current paper we showcased, for the first time, a high throughput, FPGA-based accelerator for LDPC processing that is fully integrated with a research oriented, software-based, and 5G NR compliant platform. Drawing from our previous experience, we have presented a highly efficient FPGA design, based on a Xilinx IP Core, for LDPC encoding, decoding and CRC calculations of the 5G NR coding chain. When integrated with OAI, the accelerator can achieve an LDPC decoding throughput of up to 1.6*Gbps* and a speedup of up to 13 times when compared to the single core implementation of the OAI LDPC decoder. Moreover, the LDPC encoding process is of much less computational complexity and thus acceleration by hardware offloading is more challenging to achieve. Finally, we indicate potential future improvements to our accelerator system for achieving better performance and wider interoperability as the Open RAN concept requires.

REFERENCES

[1] Q. Yang, X. Li, H. Yao, J. Fang, K. Tan, W. Hu, J. Zhang, and Y. Zhang, "BigStation: Enabling scalable real-time signal processing in large MU-MIMO systems," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, 2013, pp. 399–410.

[2] J. Ding, R. Doost-Mohammady, A. Kalia, and L. Zhong, "Agora: Real-time massive MIMO baseband processing in software," in *Proc. 16th Int. Conf. Emerg. Netw. Exp. Technol.*, 2020, pp. 232–244.

[3] G. Georgis, M. Filo, A. Thanos, C. Husmann, J. C. De Luna Ducoing, R. Tafazolli, and K. Nikitopoulos, "SWORD: Towards a soft and open radio design for rapid development, profiling, validation and testing," *IEEE Access*, vol. 7, pp. 186017–186040, 2019.

[4] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, "OpenAirInterface: Democratizing innovation in the 5G era," *Comput. Netw.*, vol. 176, Jul. 2020, Art. no. 107284. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128619314410

[5] Intel. *An Overview of FlexRAN Software Wireless Access Solutions*. Accessed: Jun. 19, 2021. [Online]. Available: https://software.intel.com/content/www/us/en/develop/videos/an-overview-of-flexran-sw-wireless-access-solutions.html

[6] Amarisoft. *Amarisoft LTE and NR Network Software Suit*. Accessed: Jun. 19, 2021. [Online]. Available: https://www.amarisoft.com/products/sovereign-5g/

[7] O-RAN Alliance. (2020). *O-RAN O-DU Low User Guide*. [Online]. Available: https://docs.o-ran-sc.org/projects/o-ran-sc-o-du-phy/en/latest/overview1.html

[8] H. Wu and H. Wang, "A high throughput implementation of QC-LDPC codes for 5G NR," *IEEE Access*, vol. 7, pp. 185373–185384, 2019.

[9] A. Li, Q. Jiang, K. Xie, M. Wang, L. Li, and W. Luo, "Low latency LDPC hard-decision algorithm for 5G NR," *IET Circuits, Devices Syst.*, vol. 14, no. 2, pp. 229–234, Mar. 2020. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-cds.2019.0160

[10] Q. Lu, Z. Shen, C.-W. Sham, and F. C. M. Lau, "A parallel-routing network for reliability inferences of single-parity-check decoder," in *Proc. Int. Conf. Adv. Technol. Commun. (ATC)*, Oct. 2015, pp. 127–132.

[11] C.-W. Sham, X. Chen, W. M. Tam, Y. Zhao, and F. C. M. Lau, "A layered QC-LDPC decoder architecture for high speed communication system," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, Dec. 2012, pp. 475–478.

[12] A. Katyushnyj, A. Krylov, A. Rashich, C. Zhang, and K. Peng, "FPGA implementation of LDPC decoder for 5G NR with parallel layered architecture and adaptive normalization," in *Proc. IEEE Int. Conf. Electr. Eng. Photon. (EExPolytech)*, Oct. 2020, pp. 34–37.

[13] A. Verma and R. Shrestha, "A new partially-parallel VLSI-architecture of quasi-cyclic LDPC decoder for 5G new-radio," in *Proc. 33rd Int. Conf. VLSI Design 19th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2020, pp. 1–6.

[14] J.-C. Liu, H.-C. Wang, C.-A. Shen, and J.-W. Lee, "Low-complexity LDPC decoder for 5G URLLC," in *Proc. IEEE Asia Pacific Conf. Postgraduate Res. Microelectron. Electron. (PrimeAsia)*, Oct. 2018, pp. 43–46.

[15] Y. Delomier, B. L. Gal, J. Crenne, and C. Jego, "Model-based design of flexible and efficient LDPC decoders on FPGA devices," *J. Signal Process. Syst.*, vol. 92, no. 7, pp. 727–745, Jul. 2020.

[16] A. Aronov, L. Kazakevich, J. Mack, F. Schreider, and S. Newton, "5G NR LDPC decoding performance comparison between GPU & FPGA platforms," in *Proc. IEEE Long Island Syst., Appl. Technol. Conf. (LISAT)*, May 2019, pp. 1–6.

[17] J. Ling and P. Cautereels, "Fast LDPC GPU decoder for cloud RAN," *IEEE Embedded Syst. Lett.*, early access, Jan. 19, 2021, doi: 10.1109/LES.2021.3052714.

[18] M. K. Roberts, "Simulation and implementation design of multi-mode decoder for Wimax and WLAN applications," *Measurement*, vol. 131, pp. 28–34, Jan. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224118307905

[19] C.-H. Lin, C.-X. Wang, and C.-K. Lu, "LDPC decoder design using compensation scheme of group comparison for 5G communication systems," *Electronics*, vol. 10, no. 16, p. 2010, Aug. 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/16/2010

[20] T. T. B. Nguyen, T. N. Tan, and H. Lee, "Low-complexity high-throughput QC-LDPC decoder for 5G new radio wireless communication," *Electronics*, vol. 10, no. 4, p. 516, Feb. 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/4/516

[21] T. T. B. Nguyen, T. N. Tan, and H. Lee, "Efficient QC-LDPC encoder for 5G new radio," *Electronics*, vol. 8, no. 6, p. 668, Jun. 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/6/668

[22] C. Tarver, M. Tonnemacher, H. Chen, J. Zhang, and J. R. Cavallaro, "GPU-based, LDPC decoding for 5G and beyond," *IEEE Open J. Circuits Syst.*, vol. 2, pp. 278–290, 2021.

[23] Intel. (Mar. 2021). *5G LDPC Intel FPGA IP User Guide*. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_5g_ldpc.pdf

[24] Xilinx. (Feb. 2021). *LDPC Encoder/Decoder v2.0 LogiCORE IP Product Brief*. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/ldpc/v2_0/pb052-ldpc.pdf

[25] 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

[26] Accelercomm. (Feb. 2021). *5G Virtual Baseband Using L1 Acceleration*. [Online]. Available: https://www.accelercomm.com/news/high-performance-accelerator-cards

[27] NVIDIA. (Sep. 2020). *GPU-Accelerated Signal Processing Technical Overview*. [Online]. Available: https://developer.download.nvidia.com/aerial/telco-signal-processing-technical-overview-us-1397003-r4-web.pdf

[28] G. Georgis, A. Thanos, M. Filo, and K. Nikitopoulos, "A DSP acceleration framework for software-defined radios on X86 64," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 1648–1652.

[29] M. Filo, J. C. De Luna Ducoing, C. Jayawardena, C. Husmann, R. Tafazolli, and K. Nikitopoulos, "Evaluating non-linear beamforming in a 3GPP-compliant framework using the SWORD platform," in *Proc. IEEE 31st Annu. Int. Symp. Pers., Indoor Mobile Radio Commun.*, Sep. 2020, pp. 1–6.

[30] K. Nikitopoulos, M. Filo, C. Jayawardena, J. C. De Luna Ducoing, and R. Tafazolli, "Non-linear base-station processing within a 3GPP compliant framework," *IEEE Access*, vol. 9, pp. 72066–72077, 2021.

[31] M. Filo, Y. Xia, and K. Nikitopoulos, "SACCESS: Towards a software acceleration framework for 5G radio access networks," in *Proc. IEEE Int. Medit. Conf. Commun. Netw. (MeditCom)*, Sep. 2021, pp. 337–342.

[32] E. A. Papatheofanous, D. Reisis, and K. Nikitopoulos, "The LDPC challenge in software-based 5G new radio physical layer processing," in *Proc. IEEE Int. Medit. Conf. Commun. Netw. (MeditCom)*, Sep. 2021, pp. 331–336.

[33] *3GPP TS 38.212: NR; Multiplexing and Channel Coding*. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.02.00_60/ts_138212v150200p.pdf

[34] M. Jacobsen, D. Richmond, M. Hogains, and R. Kastner, "RIFFA 2.1: A reusable integration framework for FPGA accelerators," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 8, no. 4, pp. 1–23, Sep. 2015, doi: 10.1145/2815631.

[35] S. Wagner. (2019). *NR LDPC Decoder*. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/openair1/PHY/CODING/nrLDPC_decoder/doc/nrLDPC/nrLDPC.pdf
[36] *Soft-Decision FEC Integrated Block*. Accessed: Jul. 3, 2021. [Online]. Available: https://www.xilinx.com/products/intellectual-property/sd-fec.html
[37] *Xilinx T1 Telco Accelerator Card Product Brief*. Accessed: Jul. 3, 2021. [Online]. Available: https://www.xilinx.com/publications/product-briefs/xilinx-t1-product-brief.pdf

ELISSAIOS ALEXIOS PAPATHEOFANOUS received the B.Sc. degree in physics and the M.Sc. degree in control and computing from the National and Kapodistrian University of Athens (NKUA), Greece, in 2017 and 2019, respectively. He is currently working as a Research Fellow in advanced physical layer implementations with the Institute for Communication Systems, Department of Electrical and Electronic Engineering, University of Surrey. His experience includes Ph.D. studies and research projects with the Digital Systems Team (DST) research group of the Department of Physics, NKUA. His interests focus on parallel processing architectures for wireless communications, signal processing, and machine learning.

DIONYSIOS REISIS received the Ptychion (B.Sc.) degree in electrical engineering from the University of Patras, Greece, in 1984, and M.Sc. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, USA. He is currently a Professor with the Electronics Lab, Department of Physics, National and Kapodistrian University of Athens (NKUA), Greece. In 1991, he joined NKUA, where he is also the Head of the NKUA's M.Sc. Programme "Electronics and Radioelectrology" and "Control & Computing." He has publications in scientific journals, such as IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, *CAS*, and *Journal of Parallel Computing*, and conferences, such as IPDPS, CVPR, and IEEE ICECS. His research interests focus on parallel processing architectures and algorithms for applications in signal and image processing, computer vision, and telecommunication systems. He has been involved in research and development projects funded by EU (RACE, ACTS, and Horizon 2020), European Space Agency, and Greek General Secretariat for Research and Technology.

KONSTANTINOS NIKITOPOULOS (Senior Member, IEEE) is currently a Reader in signal processing for communication systems with the Department of Electrical and Electronic Engineering, Institute for Communication Systems, University of Surrey, Guildford, U.K. He is a member of the 5G and 6G Innovation Centre, where he leads the "Theory and Practice of Advanced Concepts in Wireless Communications" work area, and he is the Director of the Wireless Systems Laboratory, Institute for Communication Systems. He has held research positions with RWTH Aachen University, Germany; the University of California at Irvine, USA; and University College London, U.K. He was a recipient of the prestigious First Grant of the U.K.'s Engineering and Physical Sciences Research Council and the Principal Investigator of several research projects, including the AutoAir I and II projects, of the U.K.'s 5G Testbeds and Trials Programme.