electronics

MDPI

Article

An FPGA Design with High Memory Efficiency and Decoding Performance for 5G LDPC Decoder

Bich Ngoc Tran-Thi [1,2], Thien Truong Nguyen-Ly [1] and Trang Hoang [1,*]

[1] Department of Electronics, Faculty of Electrical and Electronics Engineering, Ho Chi Minh City University of Technology (HCMUT), VNU-HCM, Ho Chi Minh City 700000, Vietnam
[2] Faculty of Electrical and Electronics Engineering, Vietnam Aviation Academy, Ho Chi Minh City 700000, Vietnam
* Correspondence: hoangtrang@hcmut.edu.vn

Abstract: A hardware-efficient implementation of a Low-Density Parity-Check (LDPC) decoder is presented in this paper. The proposed decoder design is based on the Hybrid Offset Min-Sum (HOMS) algorithm. In the check node processing of this decoder, only the first minimum is computed instead of the first two minimum values among all the variable-to-check message inputs as in the conventional approach. Additionally, taking advantage of the unique structure of 5G LDPC codes, layered scheduling and partially parallel structures are employed to minimize hardware costs. Implementation results on the Xilinx Kintex UltraScale+ FPGA platform show that the proposed decoder can achieve a throughput of 2.82 Gbps for 10 decoding iterations with a 5G LDPC codelength of 8832 bits and a code rate of 1/2. Moreover, it yields a check node memory reduction of 10% with respect to the baseline and provides a hardware usage efficiency of 4.96 hardware resources/layer/Mbps, while providing a decoding performance of 0.15 dB better than some of the existing decoders.

Keywords: 5G new radio; decoder architectures; Field Programmable Gate Array (FPGA); Low-Density Parity-Check (LDPC) codes; Min-Sum algorithm

check for updates

Citation: Tran-Thi, B.N.; Nguyen-Ly, T.T.; Hoang, T. An FPGA Design with High Memory Efficiency and Decoding Performance for 5G LDPC Decoder. *Electronics* **2023**, *12*, 3667. https://doi.org/10.3390/electronics12173667

Academic Editor: Akash Kumar

Received: 31 July 2023
Revised: 26 August 2023
Accepted: 28 August 2023
Published: 30 August 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Gallager first introduced the Low-Density Parity-Check (LDPC) codes as a family of linear Forward Error Correction (FEC) codes in his doctoral thesis in 1962 [1]. LDPC codes were known as near-Shannon limits error correction codes in practice [2]. Since their rediscovery in the middle of the 1990s [3], they have received much attention in both academia and industry. LDPC codes have a lot of advantages such as excellent coding gain, very low error floor, inherently parallelizable decoding schemes, and various optimum decoding architectures. Several industry standards, including WLAN (802.11ay) [4], WiMAX (802.16e) [5], WiFi 6 [6], DVB-S2 [7], and 10GbaseT (802.3an) [8] systems, have actually adopted them. Moreover, they have been selected for the enhanced Mobile Broadband (eMBB) in the 5th generation (5G) mobile communications by 3GPP (3rd Generation Partnership Project) [9,10]. LDPC codes have a variety of desirable characteristics that make them particularly appealing for implementation. The LDPC decoding algorithm may be performed using low-complexity computations, resulting in comparatively acceptable processing hardware design and implementation costs.

LDPC codes can be effectively decoded with the iterative Message-Passing (MP) algorithm. This decoding technique is the standard Belief-Propagation (BP) algorithm [11], which is also called the Sum-Product (SP) [12]. Although the BP algorithm provides excellent decoding performance, its implementation complexity is very high. Therefore, the Min-Sum (MS) algorithm [13] was proposed to reduce the processing of exchanged messages by applying computational approximations. The computations of the MS algorithm required only addition and comparison operations. The MS algorithm carried out the low-cost hardware implementation at the cost of significant error correction performance

degradation. In an effort to improve this performance loss, numerous approaches have been presented in the state-of-the-art, including some well-known algorithms such as the Offset Min-Sum (OMS) and the Normalized Min-Sum (NMS) [14,15]. Moreover, several enhancements have been further proposed in the literature, such as the Improved Offset Min-Sum (IOMS) algorithm [16], the Second Minimum Approximation Min-Sum algorithm (SMA-MSA) [17], the Simplified 2-Dimensional Scaled (S2DS) algorithm [18], the Hybrid Offset Min-Sum (HOMS), and the Variable Offset Min-Sum (VOMS) algorithms [19].

Although the design of individual processing units is relatively basic, the design of a whole LDPC decoder is influenced by a complicated interaction between a number of system properties. These include processing delay, decoding performance, processing throughput, bandwidth efficiency, power consumption, and flexibility. In the case of an area-efficient decoder design, one of the main difficulties is an increase in decoding algorithm complexity, which results in more hardware and memory resource consumption. Many designs have been proposed in recent years to reduce hardware resources. For instance, the authors in [20] proposed a layered decoder architecture that included several single-port memory bank structures. Once a pipeline conflict occurs, the A-Posteriori (AP) update is postponed. The conflicted check node messages contribute to the AP only when a new non-conflicted update happens. In the case that the base graph matrix is dense, which means that it has a high percentage of non-negative values relative to the total number of entries, this postponement of the AP updates might dramatically lower the performance of the layered scheduling. The authors in [21] proposed an LDPC decoder architecture without any stall cycles caused by memory access pipeline conflicts or check node degree change. The design was performed as a hybrid decoding schedule. This means it generally operates as layered scheduling and only shifts to flooded scheduling when a pipeline conflict occurs. The resulting LDPC decoder obtains better hardware efficiency. However, the design produces an increase in the hardware usage of registers. Another hardware-friendly structure was presented in [22]. Based on the 5G LDPC basis matrix's orthogonality property [9], a layer-merging and a split storage methods were utilized to improve efficiency and reduce the check-to-variable messages memory cost, respectively. The message reordering and selective-shift structure approach are applied to optimize the interconnection blocks. However, the space of memory that holds AP messages is slightly expanded due to the selective-shift structure. Recently, the hardware architecture of the QC-LDPC decoder with additional architectural transformations for parallel processing and lower routing complexity was presented in [23]. The decoder kernel was based on the Simplified Offset Min-Sum (SOMS) algorithm. In that work, they suggested a new message grouping technique that reduced the computational complexity of the QC-LDPC decoder. However, the error correction performance was reported to be lower than that of the conventional OMS decoder.

Inspired by the pros and cons of the decoders above, this paper presents an FPGA hardware implementation for the 5G LDPC decoder architecture using the HOMS algorithm [19] as the decoding kernel. This decoder leads to a reduction in hardware cost and memory size, as well as a better decoding performance. In order to achieve this, it is necessary to exploit several techniques, such as pipelining, layered scheduling, appropriate quantization bits, partially parallel structure, and parallel processing datapath units. In particular, unlike the conventional MS decoder, the proposed decoder does not need to determine the first two minimum values of all variable-to-check message inputs during check node processing; instead, it only needs to compute the first minimum value. As a consequence, it is possible to minimize hardware resources and memory at the same time.

The rest of this paper is structured as follows. Section 2 provides some definitions and fundamental information for 5G QC-LDPC codes. The proposed hardware design is described in Section 3 and the results and associated discussions are presented in Section 4. Finally, Section 5 concludes the paper.

2. Related Definitions and Preliminaries

2.1. Definitions

For simplicity, the list of notations used throughout the paper is given in Table 1.

Table 1. The list of mathematical notations and their meaning.

Notation	Meaning
$\mathbf{B}$	Base matrix
$\mathbf{H}$	Parity-Check Matrix (PCM)
Z	The expansion factor
N	The length of the codeword
K	Information bits
R	The code rate
$\mathbf{c}$	A codeword $\mathbf{c} = (c_1, c_2, \ldots, c_N)$
$\mathbf{x}$	The transmitted codeword $\mathbf{x} = (x_1, x_2, \ldots, x_N)$
$\mathbf{y}$	The received information $\mathbf{y} = (y_1, y_2, \ldots, y_N)$
σ^2	Noise variance
n	The coded bit $1 \leq n \leq N$
m	The parity check bit $1 \leq m \leq M$
$H(n)$	The set of Check Nodes (CNs) connected to the Variable Node (VNs) n
$H(n) \backslash m$	The set of $H(n)$ with check node m excluded.
$H(m)$	The set of Variable Nodes (VNs) connected to the Check Node (CNs) m
$H(m) \backslash n$	The set of $H(m)$ with check node n excluded.
d_v	The number of neighbors of a VN is called its degree, $d_v = \lvert H(n) \rvert$
d_c	The number of neighbors of a CN is called its degree, $d_c = \lvert H(m) \rvert$
$\widetilde{\gamma}_n$	A Posteriori (AP) information update
γ_n	A priori information for each variable node n
$\beta_{m,n}$	Check-to-Variable (CTV) messages sent from CN m to VN n
$\alpha_{m,n}$	Variable-to-Check (VTC) messages sent from VN n to CN m
$\widetilde{q}$	The number of bits that represent γ_n, $\alpha_{m,n}$, $\widetilde{\gamma}_n$ messages (in this paper, $\widetilde{q} = 6$ bits)
q	The number of bits that represent $\beta_{m,n}$ messages (in this paper, $q = 4$ bits)
$\varphi(x)$	The $\tilde{q}$-bit quantization map of x, that rounds each element of x to the nearest integer less than or equal to x.

The information bits are a vector of K bits. It is encoded to produce the codeword $\mathbf{c} = (c_1, c_2, \cdots, c_N)$, which is a vector of N ($N > K$) bits, by adding $M = N - K$ parity check bits to the message. These bits are used to detect and even correct transmission errors. In this work, a codeword $\mathbf{c}$ of the LDPC code is modulated by Binary Phase Shift Keying (BPSK) modulation ($\mathbf{x} = x_1, x_2, \cdots, x_N$) and transmitted over the AWGN (Additive White Gaussian Noise) channel. The information at the receiver can be described by $\mathbf{y} = \mathbf{x} + \mathbf{z}$, where $\mathbf{z}$ is an independent Gaussian random variable with zero mean and noise variance σ^2. The LDPC codes can be presented by a sparse Parity-Check Matrix (PCM) $\mathbf{H}$ with M rows and N columns, where each row corresponds to a Check Node (CN) and each column corresponds to a Variable Node (VN). Moreover, the LDPC codes can be represented using a bipartite graph. This graph is also called a Tanner graph [24]. In the Tanner graph, each VN and each CN represent a code symbol and a parity check equation, respectively. An edge connects a VN i and a CN j if the entry $H(j, i)$ of the PCM $\mathbf{H}$ equals 1.

2.2. 5G Quasi-Cyclic LDPC Codes

Quasi-Cyclic (QC) LDPC codes are one of the typical structured LDPC codes extensively used in practice [25]. These QC-LDPC codes offer a number of advantages, including similar decoding performance compared with random-LDPC codes with a basic and structured design, suitability for hardware implementations with parallelizable architecture, a simple interconnection network, and support for flexible block lengths.

In general, QC-LDPC codes are defined by a base matrix $\mathbf{B}$ of size $L \times C$ with entries $b_{i,j} \geq -1$, where $i \in \{1, 2, \ldots, L\}$ and $j \in \{1, 2, \ldots, C\}$. The PCM $\mathbf{H}$ of a QC-LDPC code is determined by extending its base matrix with an expansion factor Z. Each element in the base matrix is replaced with the sparse matrix of size $Z \times Z$ in the following manner: all zero matrices are extended for elements $b_{i,j} = -1$, and elements $b_{i,j} > -1$ are replaced by a circulant permutation matrix created by right shifting the identity matrix by $b_{i,j}$ positions. As a result, a PCM $\mathbf{H}$ has $M = Z \times L$ rows and $N = Z \times C$ columns. The 5G LDPC code design allows it to support many code sizes and code rates. Moreover, the base matrix of the 5G LDPC code is extensible by the expansion factor up to 384, allowing it to support very high parallelism levels. The expansion factor values can be obtained as $Z = a \times 2^k$, where $a = \{2, 3, 5, 7, 9, 11, 13, 15\}$ and $k = \{0, 1, 2, 3, 4, 5, 6, 7\}$. Consequently, there are all 51 expansion factor Z values listed in Table 2 [9].

Table 2. The set of expansion factors Z.

	Z	**a**							
		2	**3**	**5**	**7**	**9**	**11**	**13**	**15**
k	0	2	3	5	7	9	11	13	15
	1	4	6	10	14	18	22	26	30
	2	8	12	20	28	36	44	52	60
	3	16	24	40	56	72	88	104	120
	4	32	48	80	112	144	176	208	240
	5	64	96	160	224	288	352		
	6	128	192	320					
	7	256	383						

The BG1 and BG2 matrices are considered for 5G New Radio channel coding by 3GPP [9]. While BG1 is constructed for higher rates $(1/3 \leq R \leq 8/9)$ and larger information bits $(500 \leq K \leq 8448)$, BG2 is designed for lower rates $(1/5 \leq R \leq 2/3)$ and smaller information bits $(40 \leq K \leq 2560)$. In this paper, the BG1 matrix is considered. It must be emphasized that the 5G LDPC codes are dramatically irregular for both check nodes and variable nodes. For example, with BG1, the check node degree varies from 3 to 19 and the variable node degree varies from 1 to 30. BG1 can support the maximum information bits of 8448.

QC-LDPC codes can apply layered scheduling due to their construction based on circulant matrices. A decoding layer in layered scheduling typically consists of Z consecutive rows of the parity check matrix $\mathbf{H}$, corresponding to one row of the base matrix $\mathbf{B}$. In general, the decoding layer can consist of a number of consecutive rows of $\mathbf{B}$ such that these rows do not overlap. As mentioned above, since each non-negative element of the base matrix $\mathbf{B}$ is replaced by a circulant matrix (permutation), each column of matrix $\mathbf{H}$ has exactly one non-negative element in each decoding layer. In a decoding layer, only CNs belonging to one (or some) row(s) of the base matrix $\mathbf{B}$ update their messages and the VNs connecting to that CN update their AP information. A decoding iteration is done when all of the CNs of $\mathbf{H}$ update their messages. The algorithm of the suggested decoder with the layered scheduling principle using the HOMS algorithm [19] is summarized in Algorithm 1.

The message transmitted from VN n to neighbor CN m is denoted by $\alpha_{m,n}$ (VTC messages), while the message sent from CN m to VN n is denoted by $\beta_{m,n}$ (CTV messages). The priori information is indicated by γ_n. The PCM $\mathbf{H}$ of size $M \times N$ is divided into L horizontal layers, commonly known as decoding layers. Each decoding layer consists

of M/L consecutive rows of $\mathbf{H}$ such that each variable node is connected only once to each layer. The set of consecutive rows of $\mathbf{H}$ corresponding to layer $l \in \{1, \cdots, L\}$ is denoted as $\mathcal{M}_l$. To perform hardware implementation, we suppose that γ_n, $\alpha_{m,n}$, $\widetilde{\gamma}_n$ messages are quantized on $\widetilde{q}$-bits, while q-bits (where $q < \widetilde{q}$) are used for the quantization of $\beta_{m,n}$ messages. The symbol for the alphabet of $\widetilde{q}$-bit messages is $\widetilde{\mathcal{M}} = (-\widetilde{Q}, \cdots, -1, 0, +1, \cdots, +\widetilde{Q})$, where $\widetilde{Q} = 2^{\widetilde{q}-1} - 1$. The q-bit messages alphabet is expressed similarly by $\mathcal{M} = (-Q, \cdots, -1, 0, +1, \cdots, +Q)$, where $Q = 2^{q-1} - 1$ [26]. More specifically, $\alpha_{m,n}$ messages should be represented in q-bits before being passed to CNs. To this purpose, the saturation functions $s_{\mathcal{M}}(z)$ is applied, which is defined as:

$$s_{\mathcal{M}}(z) = \begin{cases} -Q & \text{if } z < -Q \\ z & \text{if } z \in \mathcal{M} \\ +Q & \text{if } z > +Q \end{cases} \tag{1}$$

Similar definitions apply to the $\tilde{q}$-bit saturation function $s_{\widetilde{\mathcal{M}}}$.

Algorithm 1 Layered HOMS decoding algorithm [19].

Input: $\mathbf{y} = (y_1, y_1, \ldots, y_N) \in \mathcal{Y}^{\mathcal{N}}$ ($\mathcal{Y}$ is the channel output alphabet) ▷ received word
Output: $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_1, \ldots, \hat{x}_N) \in \{0,1\}^N$ ▷ estimated codeword

Initialization
for all $n = 1, \ldots, N$ **do** $\boldsymbol{L}_n = \log \dfrac{\Pr(x_n = 0 \mid y_n)}{\Pr(x_n = 1 \mid y_n)}$; $\tilde{\gamma}_n = \gamma_n = \varphi(\boldsymbol{L}_n)$
end for
for all $m = 1, \ldots, M$ and $n \in \mathcal{H}(m)$ **do** $\beta_{m,n} = 0$;
end for
Iteration Loop
for all $i = 1, \ldots, n_{iter}$ **do** ▷ Loop over iterations
  for all $l = 1, \ldots, L$ **do** ▷ Loop over horizontal layers
    for all $m = 1, \ldots, \mathcal{M}_l$ and $m \in \mathcal{H}(n)$ **do** ▷ **VN-processing**
      $\alpha_{m,n} = s_{\widetilde{\mathcal{M}}}(\tilde{\gamma}_n - \operatorname{sgn}(\beta_{m,n}) \cdot \max\{|\beta_{m,n}| - \delta, 0\})$
      $\alpha^{sat}_{m,n} = s_{\mathcal{M}}(\alpha_{m,n})$
    end for
    for all $m = 1, \ldots, \mathcal{M}_l$ and $n \in \mathcal{H}(m)$ **do** ▷ **CN-processing**
      $\beta_{m,n} = \begin{cases} A \cdot \min 1 & \text{if position of } \left|\alpha^{sat}_{m,n'}\right| = \text{index_min1} \\ A \cdot \max\{\min 1 - \beta, 0\} & \text{otherwise} \end{cases}$
      $\min 1 = \min_{n \in \mathcal{H}(m)} \left|\alpha^{sat}_{m,n}\right|$
      $A = \prod_{n' \in \mathcal{H}(m) \setminus n} \operatorname{sgn}(\alpha^{sat}_{m,n'})$
    end for
    for all $m = 1, \ldots, \mathcal{M}_l$ and $m \in \mathcal{H}(n)$ **do** ▷ **AP-update**
      $\tilde{\gamma}_n = s_{\widetilde{\mathcal{M}}}(\alpha_{m,n} + \beta_{m,n})$
    end for
  end for
end for
End Iteration Loop

n_{iter} is the maximum number of iterations.
*index_min*1 is the position of *min*1. In cases where there exists more than one *min*1 value, *index_min*1 is determined as the position with the smallest index.
$\delta, \beta > 0$ is the offset factors.
$\alpha^{sat}_{m,n}$ is the value of $\alpha_{m,n}$ saturated to q-bit.

3. The Proposed Decoder Architecture

The layered HOMS algorithm presented in Algorithm 1 is used to implement the suggested QC-LDPC decoder architecture for 5G New Radio (NR) communication systems. In this work, we consider the BG1 matrix of size $m_b \times n_b = 24 \times 46$ and the code rate

of 1/2. The expansion factor $Z = 192$ corresponds to a PCM $\mathbf{H}$ of size $M \times N$, with $M = Z \cdot m_b = 192 \cdot 24 = 4608$ and $N = Z \cdot n_b = 192 \cdot 46 = 8832$. As mentioned before, the 5G LDPC decoder is irregular [9] and the check node degree in the BG1 changes widely from 3 to 19. Let d_{cmax} indicate the maximum check node degree. Since each column of $\mathbf{H}$ contains precisely one non-negative element in each layer, we further define one row of $\mathbf{B}$ within one decoding layer. It should be noted that BG1 is constructed such that $\mathbf{B}$ can be separated into $L = m_b$ horizontal decoding layers. The decoder's parallelism level is equal to the expansion factor Z.

The architecture of the proposed LDPC decoder is depicted in Figure 1. Its architecture consists of a network of multiplexers (MUXes), a large number of Variable Node Units/A Posteriori Blocks (VNUs/APBs), Check Node Units (CNUs), and two memory blocks for AP information ($\tilde{\gamma}_n$) and CTV messages ($\beta_{m,n}$). The MUX block's task is to select between channel intrinsic messages (γ_n) and AP messages ($\tilde{\gamma}_n$) updated from the preceding layer. It is important to understand that intrinsic messages are only used during the initialization stage. The intrinsic signals of $\tilde{q}$-bits quantization are buffered before being stored in the AP Memory Block (AP-MB). The detail of the AP-MB will be discussed in a later subsection. There are n_b simultaneous outputs of the AP-MB, each representing Z values of $\tilde{q}$-bits. The decoded bits are read from the AP-MB when the maximum number of iterations is reached. AP messages are read from the AP-MB and then delivered to the Read Permutation Units (Read-PUs) at each decoding layer. There are n_b parallel Read-PUs that are used to reorganize the data according to the processed layer, ensuring processing by the appropriate following processing units. The Read-PUs select d_{cmax} AP combinations among n_b inputs, which depend on the positions of the non-negative entries of the BG1 matrix's current layer. For example, the first four layers of the BG1 matrix have $d_{\text{cmax}} = 19$ non-negative elements, and the fifth layer has only four non-negative elements, thus, $(d_{\text{cmax}} - 4)$ elements are set to zero. In order to enhance the operating frequency, we apply the pipeline technique at Read-PU outputs. The Write Permutation Units (Write-PUs) perform the reverse operation of Read-PU. One layer is executed in two consecutive clock cycles, and each clock cycle involves the processing of two layers in the base matrix by the two pipeline stages. After pipelining, Read-PU output data are transferred into Read-Shifter Blocks (R-SBs). R-SBs should be used to design the data flow in circular shifts (cyclic) in line with the relevant shift factors. In the proposed architecture in Figure 1, we use d_{cmax} R-SB blocks in parallel, enabling high-processing throughput. Each of them has Z inputs and Z outputs. Figure 2 illustrates the detailed working of right-shifting by the two positions rule of R-SB block with 60 elements of $\tilde{q}$-bits. The Write-Shifter Blocks (W-SBs) perform a reverse function compared to the R-SBs.

VTC messages $\alpha_{m,n}$ are computed or AP values $\tilde{\gamma}_n$ can be updated via Variable Node Units/A Posteriori Blocks (VNUs/APBs). How these blocks work in detail will be mentioned in the next subsection. The output message values of the VNUs/APBs are transmitted in parallel $\tilde{q} \times Z = 1152$ bits updated for the next layer and saturating to q-bits. Here, the saturation values are within the range $[-Q, \cdots, +Q]$. Then these saturated data are sent to the Check Node Units (CNUs). For the HOMS algorithm, the output message of each CNU comprises the first minimum, its index, and the signs of the incoming VTC messages. This message is kept in a compressed format to save memory space, which is referred to as a compressed $\beta-$ message, as seen in Figure 3.

These compressed $\beta-$messages are fed in two directions: they are stored back into the Check Node Memory Blocks (CN-MBs) for the next iteration and directed to the VNUs/APBs for the next decoding layer processing. It is worth noting that the compressed $\beta-$messages from CN-MBs are sent back to VNUs/APBs through Decompress units (DECOMs). Two DECOMs are put into the decoder to convert compressed to uncompressed CTV messages. They create CTV messages for further processing. Similarly, the updated AP messages are rearranged using the Write Permutation Unit (Write-PU) to ensure that they can be placed in the appropriate addresses of AP-MB. The Split block creates the serial signal stream of $Z \times \tilde{q}$ bits that is sent to the AP-MB. One decoding layer is done after

writing them back to the AP-MB. To guarantee that the decoder operates properly, the controller block is required. This block controls the synchronous operation of the other ones. Additionally, it also creates control signals to manage memory access.

Initially, AP-MB storage of all AP messages takes $n_b = 46$ clock cycles. The proposed decoder is designed to operate in two consecutive clock cycles in every layer. In the first clock cycle, the AP-reading signal is in the "enable" condition to provide the AP information messages that correspond to the current BG1 matrix layer. Following one another, operations for VN and CN are updated. In the second clock cycle, the decoder executes the AP processing. After AP information computations, they are written back to AP-MB by activating the AP-writing signal at a "high" level. The processing described above continues until the last layer (i.e., 24th layer) of the BG1 matrix is finished, at the moment one decoding iteration is complete. These procedures are performed in 10 iterations during the decoding process. After the 10th iteration, the LDPC decoder produces information bits, corresponding to the first 22 memory units of the AP-MB (e.g., $22 \times 192 = 4224$ bits).

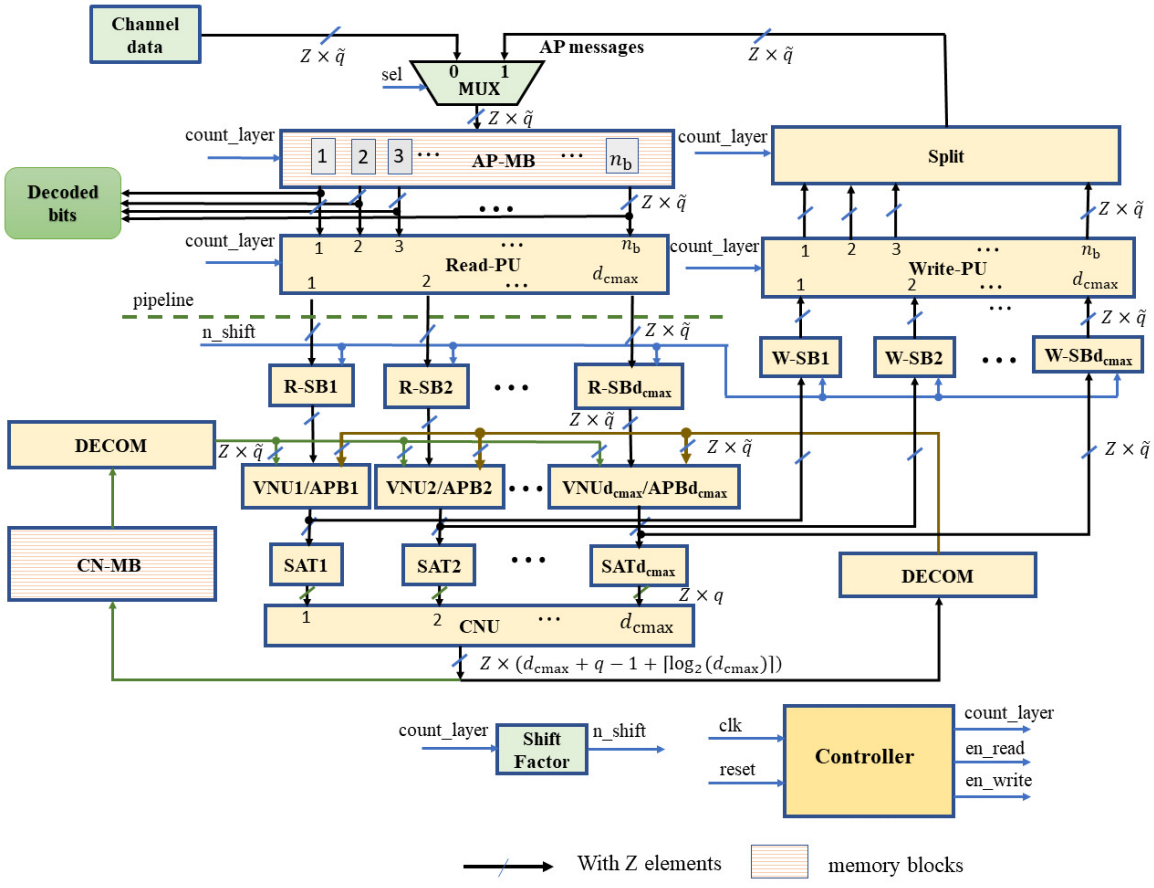


Figure 1. Block diagram of the proposed LDPC decoder architecture.

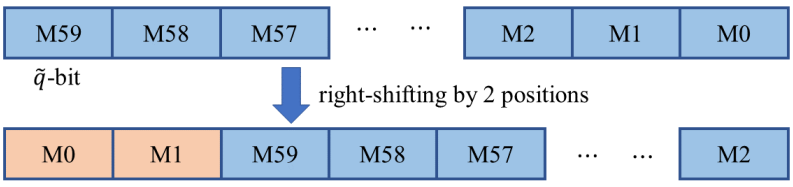


Figure 2. An example of right-shifting by the two positions rule of R-SB block with 60 elements of $\tilde{q}$-bits.

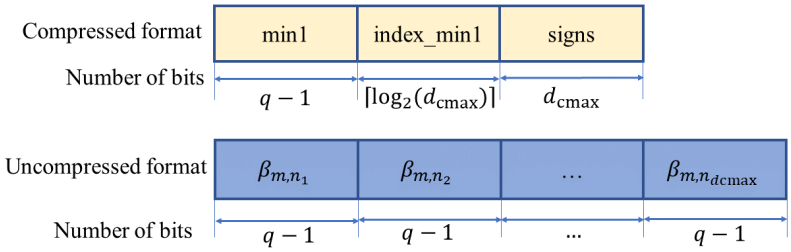


Figure 3. Presentation of the check-to-variable node message in compressed and uncompressed formats.

The following subsections provide a detailed explanation of some main blocks in the proposed decoder architecture.

3.1. Check Node Unit

The main task of the CNU is to calculate the CTV messages ($\beta_{m,n}$). On the hardware aspect, conventional algorithms need to determine the two minimum values of the VTC input signals and the index of the first minimum. In this work, the HOMS method simply requires the first minimum of incoming VTC messages and its index. It is the competitive advantage of the proposed architecture, which may reduce the hardware resources. In addition, to reduce CN-MB storage space, CTV messages are compressed as shown in Figure 3. For a current check node, the CTV messages, named compressed $\beta-$messages, are represented in the format of $q-1+\lceil \log_2(d_{cmax}) \rceil + d_{cmax} = 27$ bits. It is important to note that each conventional β-message includes $2 \cdot (q-1) + \lceil \log_2(d_{cmax}) \rceil + d_{cmax} = 30$ bits.

Each CNU has d_{cmax} inputs, and each one has $Z \times q = 768$ bits. As a result, each CNU block really contains Z computing units that are utilized to compute the Z check nodes within a single row of the BG1 matrix in parallel. Each layer of these d_{cmax} CNUs produces $Z \times (q-1+\lceil \log_2(d_{cmax}) \rceil + d_{cmax}) = 5184$-bit extrinsic CTV messages and delivers to CN-MB.

As illustrated in Figure 4, the proposed CNU architecture consists of d_{cmax} multiplexer units, a Min & Index Finder, a sign unit, and some control signals. In this design, we suppose that all check nodes have the same degree, which is represented by d_{cmax}. To support irregular 5G LDPC codes, some additional control logic (i.e., a set signal) is used. More precisely, in the case of check node degree $d_c < d_{cmax}$, "inactivate" (set = '0') refers to setting the last $(d_{cmax} - d_c)$ to the maximum value ($2^{q-1} - 1 = +7$ including the sign bit). Multiplexers are set up at the inputs to choose the input message based on either the "real" VTC message ($\alpha_{m,n}$) or the maximum value. To find the first minimum ($min1$) and its index ($index_min1$), the CNU is constructed by applying the effective Tree-Structure (TS) technique suggested in [27]. The structure of the CNU is described in more detail in Figure 4. The Most Significant Bits (MSBs) of the d_{cmax} VTC messages are XORed to generate a sign bit, and the remaining $(q-1)$-bits of these are processed by the Min & Index finder to find $min1$ and its index. The d_{cmax} inputs of Min & Index finder can be split down into the sum of 16 and 3 inputs. As a result, the d_{cmax}-mVU (minimum Value Unit) is achieved by combining the corresponding blocks similar to the TS approach used in [27]. The 2-mVU includes one multiplex and one comparator, as illustrated in Figure 5a. The architecture of the 3-mVU consists of two 2-mVU for finding the minimum value and its index, as depicted in Figure 5b. The 16-mVU (or 2^4-mVU) can be constructed from the 2-mVU. The Index Generator (IG) block is used to compute the $index_min1$. This IG block uses the same methodology as in [27].

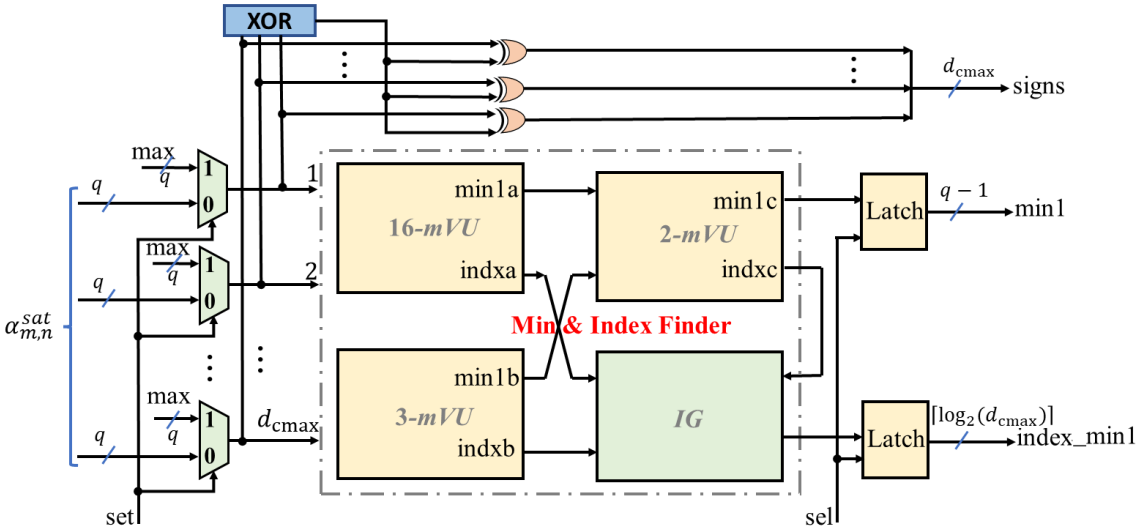


Figure 4. Check Node Unit (CNU) architecture with $d_{\text{cmax}} = 19$ (Min & Index finder architecture is shown in the gray dashed box).

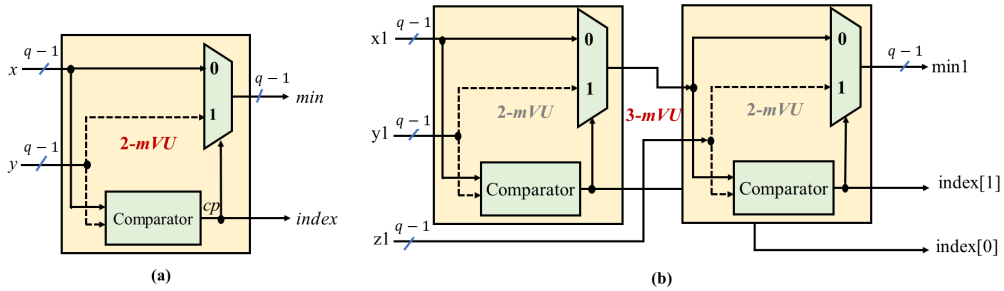


Figure 5. The architecture of (**a**) 2−mVU and (**b**) 3−mVU.

3.2. Variable Node Units and A Posteriori Blocks

As shown in Algorithm 1, VNUs of the proposed decoder calculate $\alpha_{m,n}$ of the VN processing step, while APBs compute $\widetilde{\gamma}_n$ messages of the AP update step. It is worth noting that the main difference between these two blocks is that the VNU uses the subtractor while the APB uses the adder. Hence, we merge their logical functionalities into a VNU/APB, controlled by the "sel" signal. The controller generates this control signal, which selects VNU mode during the first clock and APB mode during the second clock. The detailed architecture of VNUs/APBs is depicted in Figure 6. In the current iteration, during the first clock cycle, the control signal sel = 0, and at the input two multiplexers are selected VNU mode ($\widetilde{\gamma}^{old}_n, \beta^{old}_{m,n}$). The Subtractor/Adder block chooses the subtraction operation, thus, at the output, VTC messages are received. Each VN messages is calculated by $\alpha^{new}_{m,n} = \widetilde{\gamma}^{old}_n - \beta^{old}_{m,n}$, where $\widetilde{\gamma}^{old}_n, \beta^{old}_{m,n}$ are the AP information values and the appropriate CN message values read from the AP-MB and CN-MB, respectively. Moreover, at the first clock cycle, these received $\alpha^{new}_{m,n}$ messages provide to CNUs to create CN messages $\beta^{new}_{m,n}$. During the second clock cycle, the control signal sel = 1, APB mode is executed and AP information update values ($\widetilde{\gamma}^{new}_n$) are updated by $\widetilde{\gamma}^{new}_n = \alpha^{new}_{m,n} + \beta^{new}_{m,n}$.

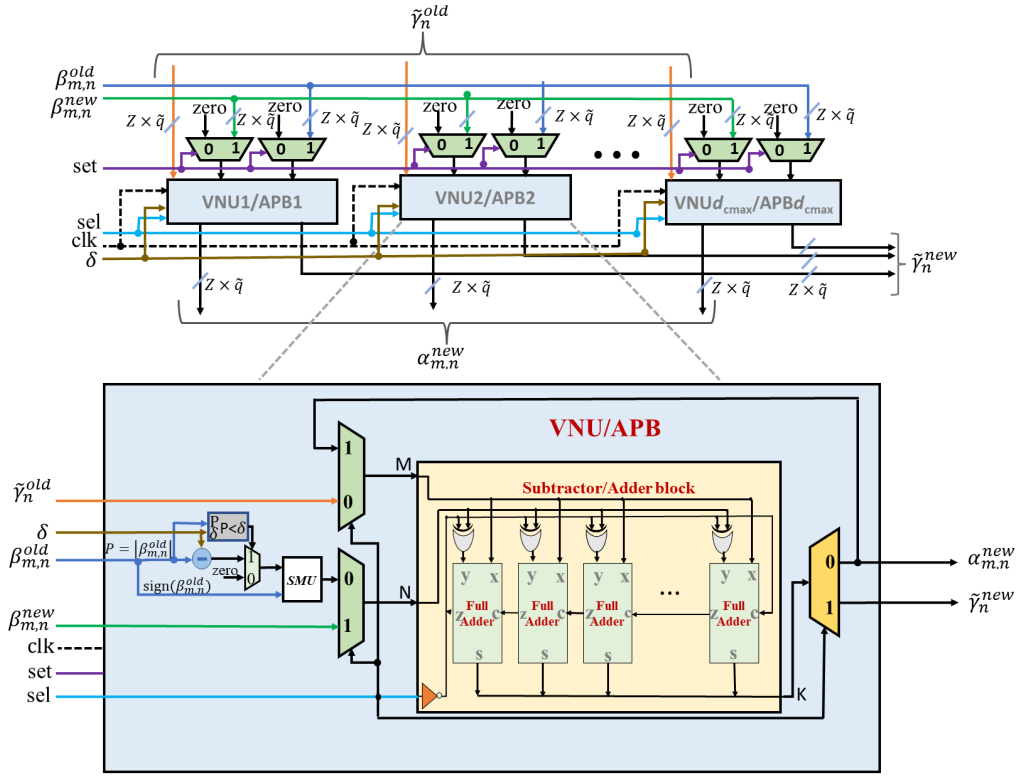


Figure 6. The proposed VNU/APB architecture.

As mentioned above, the check node degree varies dramatically in the 5G LDPC codes. Therefore, control signal "set" is needed to inactivate (i.e., the input values are set to zero) in cases with the check node degree $d_c < d_{\text{cmax}}$. In this case, the last $(d_{\text{cmax}} - d_c)$ CTV inputs are inactivated (set = '0').

It is important to emphasize that for the HOMS algorithm, VN processing is modified by applying an offset factor ($\delta > 0$) to the VTC message. Before sending into the Subtractor/Adder block, the $\beta_{m,n}^{old}$ magnitude is reduced by the offset parameter $\delta > 0$ or set to zero if the resulting value is less than zero. The SMU transforms a sign-magnitude binary number in $\tilde{q}$-bits (i.e., 1-bit sign and $\tilde{q} - 1$ magnitude) to $\tilde{q}$-bit in two's complement format.

3.3. Memory Blocks

LDPC decoder architecture requires two storage options: one for the $\tilde{\gamma}_n$ values (AP-MB) and another for the $\beta_{m,n}$ messages (CN-MB). The number of memory units of the AP-MB equals the number of columns of the base matrix. Each unit contains Z memory locations with a width of $\tilde{q}$-bits. Accordingly, the AP-MB contains n_b memory units that hold $n_b \times Z = 8832$ received-AP information of $\tilde{q}$-bits, for a total of $n_b \times Z \times \tilde{q} = 52{,}992$ bits. The $\tilde{q} \times Z = 1152$-bit AP messages are sent sequentially into AP-MB. That means n_b clock cycles are needed to store all of the AP messages. Such n_b AP groups are stored in n_b memory units and read simultaneously from the AP-MB in a single clock cycle.

In the proposed decoder, the second minimum should not be calculated and stored in the CN-MB, which is the main difference from the other conventional decoders. The CN-MB is used for the storage of the compressed $\beta-$message. Since $\beta-$messages from all layers must be kept, the depth of the CN-MB is equal to the number of layers. Consequently, a total of $m_b \times Z \times (q - 1 + \lceil \log_2(d_{\text{cmax}}) \rceil + d_{\text{cmax}}) = 124{,}416$ bits of the CN-MB is required.

Table 3 shows the size of the proposed and conventional CN-MB. Without storing the second minimum, the size of the proposed CN-MB could be reduced by 10%. These CN-MBs' outputs are sent into the Decompress block. This compressed $\beta-$message consists of four parts: 19 MSBs between 26–8 bit-position representing the signs of 19 CNU input

messages, the next 7–5 bit-positions corresponding to the magnitude of $min1$ and the last 5 Least-Significant-Bits (LSBs) of 4–0 bit-positions representing the index of $min1$. The overall memory of the LDPC decoder is mostly determined by the size of the AP-MB and CN-MB. According to the calculations above, the AP-MB requires 52,992 bits and the CN-MB stores 124,416 bits. Therefore, the total memory is 173.25 kb.

Table 3. Comparison between the proposed and conventional CN-MB.

Conventional CN-MB			Proposed CN-MB		
Width	**Depth**	**Total**	**Width**	**Depth**	**Total**
30Z	24	720Z	27Z	24	648Z

3.4. Decompress Block

It should be noted that the CTV message is stored in CN-MBs as a compressed $\beta-$message. The Decompress block (DECOM) converts the CTV messages from a compressed to an uncompressed format (Figure 3). Moreover, this block is responsible for reducing the input value $min1$ by a negative offset (β). In the case that the difference value ($min1-\beta$) is less than 0, the output of this block is set to 0. The detailed architecture of the DECOM block is depicted in Figure 7. The d_{cmax} outputs of the DECOM represent $d_{\text{cmax}}-1=18$ replications of ($\max\{(\text{min1}-\beta),0\}$) magnitudes and only one $min1$ magnitude (its position equals the $index_min1$). The FCU (Format Conversion Unit) converts a sign-magnitude binary number in q-bits (i.e., 1-bit sign and $q-1$ magnitude) to $\tilde{q}$-bits in two's complement format.

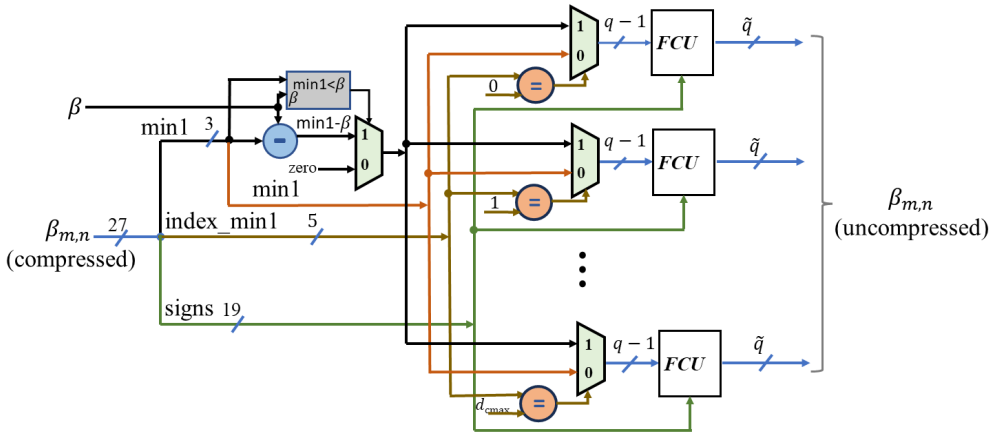


Figure 7. Decompress block architecture.

4. Results and Discussion

In this work, an LDPC decoder was developed in Verilog Hardware Description Language (HDL) instead of using intellectual property (IP) modules. The results of FPGA synthesized and implemented after place and route are achieved by using Xilinx Vivado 2021.2. This section additionally reports on the decoding performance of several 5G LDPC codes.

4.1. Performance Results

We consider the BG1 matrix with code rates of 1/2, 2/3, codeword lengths N of 8832, 6720, and the expansion factor Z of 192. The HOMS is set with $\beta=0.5$ and $\delta=0.375$ [19]. For comparison purposes, the performances of the NMS ($1/\alpha=0.75$) [14], OMS ($\beta=0.5$) [14], IOMS ($\gamma=0.875$ and $\eta=0.5$) [16], SMA-MSA ($\alpha 2=0.25$ and $\gamma=0.75$) [17], S2DS [18], and VOMS ($\beta=0.5$ and $\tau=0.875$) [19] algorithms are also evaluated.

First, to demonstrate the modification of the CTV messages, we have compared the CTV messages among the HOMS, BP, and MS algorithms along with various reported MS-based decoding algorithms, as illustrated in Figure 8.

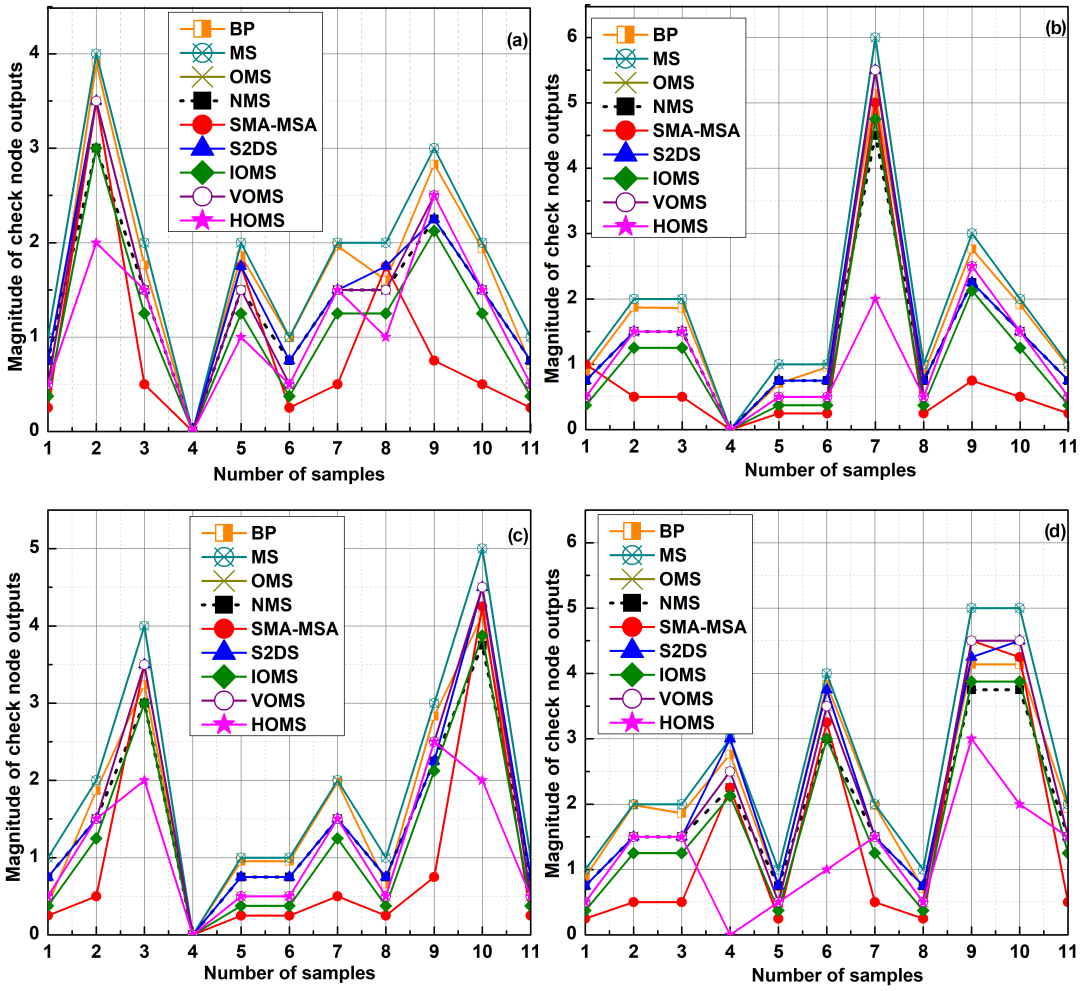


Figure 8. Comparison of the Check Node unit outputs among various LDPC decoding algorithms.

In this demonstration, we calculate four CTV messages with the same data inputs for some algorithms such as the BP, MS, OMS, NMS, IOMS, SMA-MSA, VOMS, and HOMS. These four CTV messages correspond to Figure 8a,b,c and d, respectively. There are 11 tested samples. Looking closely, we can see that the gap between the HOMS and the BP algorithm is large. There are two reasons for this overestimation: first, the HOMS algorithm is based on the MS algorithm that has self-existing errors; second, the updated CTV messages in the HOMS algorithm are calculated only based on the *min*1 value, while the *min*2 is approximated. To overcome this disadvantage, another offset factor is applied to the VTC messages of the HOMS in variable node processing. A solution is offered to modify VNU instead of continuing to modify CNU because the CNU is one of the most complex blocks, therefore, adding a modified offset factor may increase the hardware complexity. Meanwhile, the VNU/APB block is very simple, using only adders/subtractors. Next, to verify the error correction capacity of these algorithms, we proceed with the Monte Carlo simulations for several 5G LDPC codes. Some simulation results are shown in Figures 9 and 10.

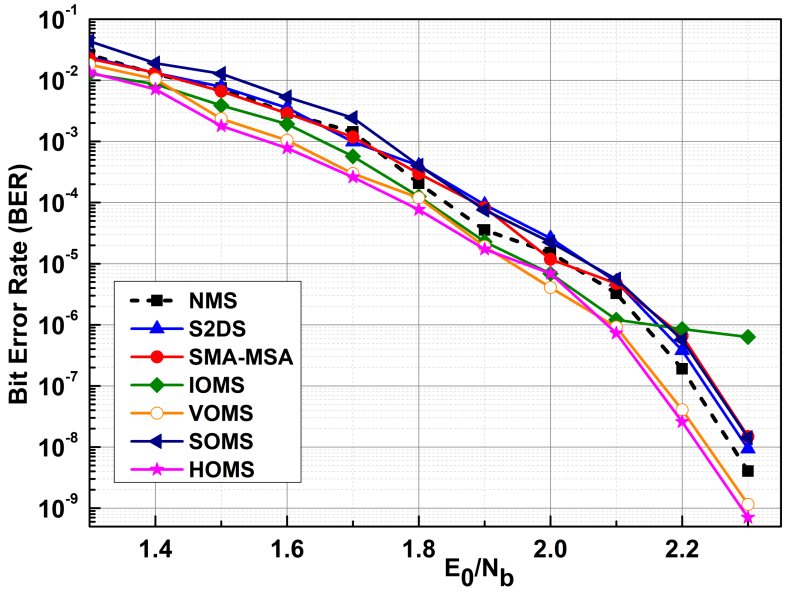


Figure 9. BER performance of various LDPC decoders for the (8832, 4608), code rate 1/2.

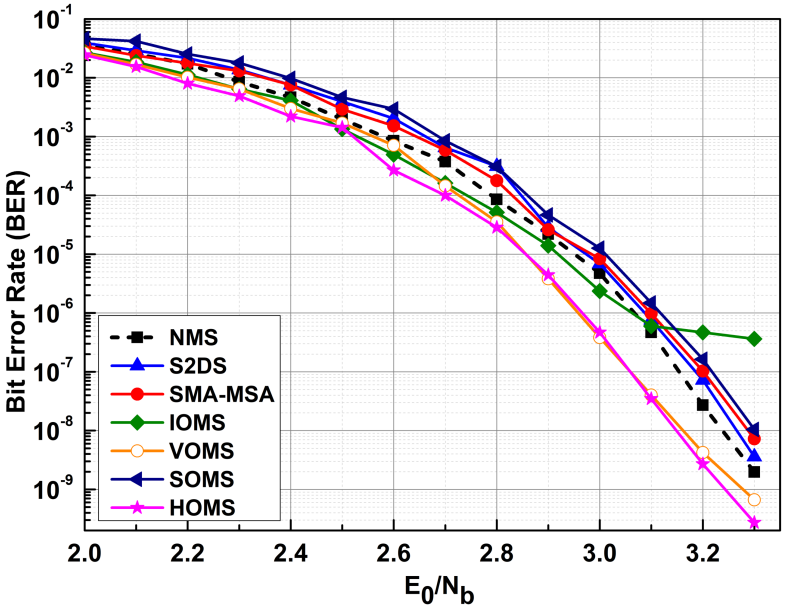


Figure 10. BER performance of various LDPC decoders for the (6720, 2496), code rate 2/3.

It can be seen that at the target BER of 10^{-8}, the HOMS algorithms improve the decoding gain up to 0.15 dB compared to the NMS, IOMS, SMA-MSA, S2DS, and SOMS algorithms.

4.2. Implementation Results

In this section, we evaluate the hardware resources of the proposed LDPC decoder on the FPGA platform. The design is synthesized and implemented on the Xilinx Kintex UltraScale+ FPGA by using Vivado software. We investigate a 5G LDPC code with a codeword length of 8832 bits and a code rate of 1/2. The decoding iteration number is set to 10, and the quantization bits $(q, \tilde{q})$ are (4, 6). The throughput computed is given by the formula [23]:

$$T = \frac{N \times F_{\max}}{L \times n_{iter} \times n_{cyc}} [\text{Mbps}] \quad (2)$$

where L is the number of decoding layers, n_{cyc} is the number of clock cycles necessary to complete one layer, n_{iter} is the maximum number of iterations, N is the codeword length, and $F_{\max}$ [MHz] is the maximum operating frequency.

The proposed 5G LDPC decoder achieves a throughput of 2.82 Gbps while decoding a code with code length $N = 8832$ bits, $L = 24$ layers at a code rate $1/2$. The decoding iteration number is set to 10 and the number of clock cycles per layer is 2. Since LDPC decoder designs have varying implementation characteristics such as code length, quantization bits, algorithm, and iteration number, the performance normalization is required for a fair comparison. Therefore, the Hardware Usage Efficiency (HUE) parameter is used. The HUE metric is defined as the hardware resources needed for each layer of the base matrix to produce a throughput of 1 Mbps. It is measured in hardware resources/layer/Mbps. A lower HUE value implies a better hardware-efficient design. The HUE metric is calculated as follows [23]:

$$\text{HUE} = \frac{H_u}{L \times T} \tag{3}$$

where L is the number of decoding layers, T [Mbps] is the throughput, and H_u reflects the FPGA hardware utilization, which is the number of F7 MUXes, F8 MUXes, Flip-flops, LUTs, and BRAMs in bits used by the LDPC decoder. Note that, for BRAM, we need to calculate its size in bits. This means each bit of BRAM is considered to be the same weight as one flip-flop or one MUX.

In Table 4, the implementation results are compared with the other related 5G designs: [21,23,28,29].

Table 4. Implementation results and comparison with related works.

Specifications	**This Work**	**2023 [23]**	**2022 [29]**	**2021 [28]**	**2020 [21]**
FPGA Board	Xinlinx Kintex Ultrascale+	Xinlinx Zynq Ultrascale+	Xinlinx Virtex-7	Xinlinx Kintex-7	Xinlinx Kintex Ultrascale+
Quantization Bits	(4,6)-bit	7-bit	8-bit	8-bit	8-bit
Throughput (Gbps)	2.82	3.6–13.31	2.85	0.391–1.1	3.2–4.9
Code Length	8832	10,368–26,112	10,368	3456	10,368–26,112
Expansion Factor	192	384	384	384	384
Clock Frequency (MHz)	153.5	128.36	255	160	404.8
Code Rate	1/2	1/3–8/9	22/27	1/2–5/6	22/68–22/27
Number of Layers	24	5–46	7	5–46	7–46
Number of Iterations	10	10	11	8	10
Pipeline Stages	2	3	13	-	13
Memory Size (kb)	173.25	765.8	3888	7146	4914
Decoding Algorithm	HOMS	SOMS	OMS	OMS	OMS
Hardware Utilization	314,968	1,200,394	4,204,122	7,438,394	5,233,977
Memory Type	BRAM	Registers	BRAM	BRAM	BRAM
Permutation Network	Barel Shifter	Bit Rotational Mapping	Barel Shifter	Barel Shifter	Barel Shifter
HUE *	4.65	1.96	20.85	147.54	23.12

* Hardware utilization efficiency (HUE) in hardware resources/layer/Mbps.

It can be seen that the suggested decoder's throughput is 2.82 Gbps, which is almost equivalent to the throughput of the decoder in [29] and around 2.5 times higher than the decoder in [28]. Concerning hardware utilization efficiency, the proposed design provides a HUE value of 4.65 hardware resources/layer/Mbps. It shows a hardware utilization efficiency of approximately 4.5–5 times lower than the decoders in [29] or [21]. Although compared to [23], the proposed decoder has 1.7 times lower throughput and about 2.4 times higher HUE, it provides better decoding performance and memory efficiency. More precisely, the memory capacity of the proposed decoder is 4.4 times less than the decoder in [23] and at a BER of 10^{-8}, it shows a gain of 0.15 dB.

5. Conclusions

This paper presented the area-efficient architecture and enhanced decoding performance of a 5G LDPC decoder. The suggested decoder was based on the HOMS algorithm. To accomplish this architecture, various techniques such as layer scheduling, partial-parallel construction, and a low number of quantization bits were used. Particularly, the proposed check node processing only requires computing the first minimum of all input messages. As a result, in the proposed decoder, the storage size for check node messages was decreased by around 10% when compared to the conventional decoders. Implemented decoders demonstrated a gain of 0.15 dB over the SOMS decoder and provided throughputs of 2.82 Gbps, which met the 5G throughput standards.

Author Contributions: B.N.T.-T. proposed the idea, performed simulations, collected and analyzed data and prepared the original manuscript. T.T.N.-L. reviewed, supervised and revised the manuscript. T.H. supervised and revised the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data sharing not applicable.

Acknowledgments: We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

AP	A Posteriori
APB	A Posteriori Block
AP-MB	A Posteriori Memory Block
AWGN	Additive White Gaussian Noise
BP	Belief-Propagation
BPSK	Binary Phase Shift Keying
CN	Check Node
CN-MB	Check Node Memory Block
CNU	Check Node Unit
CTV	Check-to-Variable
DECOM	Decompress unit
FEC	Forward Error Correction
FPGA	Field Programmable Gate Array
HOMS	Hybrid Offset Min-Sum
HUE	Hardware Usage Efficiency
IG	Index Generator
IOMS	Improved Offset Min-Sum
LDPC	Low-Density Parity-Check
LSB	Least-SignificantBits

MP	Message-Passing
MS	Min-Sum
MSB	Most Significant Bit
MUX	Multiplexer
mVU	minimum Value Unit
NMS	Normalized Min-Sum
NR	New Radio
OMS	Offset Min-Sum
PCM	Parity-Check Matrix
QC-LDPC	Quasi-Cyclic LDPC
Read-PU	Read Permutation Unit
R-SB	Read-Shifter Blocks
S2DS	Simplified 2-Dimensional Scaled
SMA-MSA	Second Minimum Approximation Min-Sum algorithm
SOMS	Simplified Offset Min-Sum
SP	Sum-Product
TS	Tree-Structure
VN	Variable Node
VNU	Variable Node Unit
VOMS	Variable Offset Min-Sum
VTC	Variable-to-Check
Write-PU	Write Permutation Unit

References

1. Gallager, R. Low-density parity-check codes. *IRE Trans. Inf. Theory* **1962**, *8*, 21–28. [CrossRef]
2. Chung, S.Y.; Forney, G.D.; Richardson, T.J.; Urbanke, R. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. *IEEE Commun. Lett.* **2001**, *5*, 58–60. [CrossRef]
3. MacKay, D.J.; Neal, R.M. Near Shannon limit performance of low-density parity check codes. *Electron. Lett.* **1997**, *33*, 457–458. [CrossRef]
4. Lin, C.H.; Su, H.H.; Chen, T.S.; Lu, C.K. Reconfigurable Low-Density Parity-Check (LDPC) Decoder for Multi-Standard 60 GHz Wireless Local Area Networks. *Electronics* **2022**, *11*, 733. [CrossRef]
5. Liu, C.H.; Yen, S.W.; Chen, C.L.; Chang, H.C.; Lee, C.Y.; Hsu, Y.S.; Jou, S.J. An LDPC decoder chip based on self-routing network for IEEE 802.16e applications. *IEEE J. Solid-State Circuits* **2008**, *43*, 684–694. [CrossRef]
6. Wu, Y.; Wu, B.; Zhou, X. High-Performance QC-LDPC Code Co-Processing Approach and VLSI Architecture for Wi-Fi 6. *Electronics* **2023**, *12*, 1210. [CrossRef]
7. Kim, S.M.; Park, C.S.; Hwang, S.Y. A novel partially parallel architecture for high-throughput LDPC decoder for DVB-S2. *IEEE Trans. Consum. Electron.* **2010**, *56*, 820–825. [CrossRef]
8. Cohen, A.E.; Parhi, K.K. A low-complexity hybrid LDPC code encoder for IEEE802.3an (10GBase-T) ethernet. *IEEE Trans. Signal Process.* **2009**, *57*, 4085–4094. [CrossRef]
9. Richardson, T.; Kudekar, S. Design of low-density parity-check codes for 5G new radio. *IEEE Commun. Mag.* **2018**, *56*, 28–34. [CrossRef]
10. Thi Bao Nguyen, T.; Nguyen Tan, T.; Lee, H. Low-complexity high-throughput QC-LDPC decoder for 5G new radio wireless communication. *Electronics* **2021**, *10*, 516. [CrossRef]
11. Richardson, T.J.; Urbanke, R.L. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inf. Theory* **2001**, *47*, 599–618. [CrossRef]
12. Kschischang, F.R.; Frey, B.J.; Loeliger, H.A. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory* **2001**, *47*, 498–519. [CrossRef]
13. Fossorier, M.P.; Mihaljevic, M.; Imai, H. Reduced complexity iterative decoding of low-density parity-check codes based on belief propagation. *IEEE Trans. Commun.* **1999**, *47*, 673–680. [CrossRef]
14. Chen, J.; Dholakia, A.; Eleftheriou, E.; Fossorier, M.P.; Hu, X.Y. Reduced-complexity decoding of LDPC codes. *IEEE Trans. Commun.* **2005**, *53*, 1288–1299. [CrossRef]
15. Chen, J.; Fossorier, M.P. Near optimum universal belief propagation based decoding of low-density parity check codes. *IEEE Trans. Commun.* **2002**, *50*, 406–414. [CrossRef]
16. Tran-Thi, B.N.; Nguyen-Ly, T.T.; Hong, H.N.; Hoang, T. An improved offset min-sum LDPC decoding algorithm for 5G new radio. In Proceedings of the 2021 International Symposium on Electrical and Electronics Engineering (ISEE), Ho Chi Minh, Vietnam, 15–16 April 2021.
17. Català-Pérez, J.M.; Lacruz, J.O.; Garcia-Herrero, F.; Valls, J.; Declercq, D. Second minimum approximation for min-sum decoders suitable for high-rate LDPC codes. *Circuits, Syst. Signal Process.* **2019**, *38*, 5068–5080. [CrossRef]

18. Cho, K.; Lee, W.H.; Chung, K.S. Simplified 2-dimensional scaled min-sum algorithm for LDPC decoder. *J. Electr. Eng. Technol.* **2017**, *12*, 3, 1262–1270. [CrossRef]
19. Tran-Thi, B.N.; Nguyen-Ly, T.T.; Hoang, T. High-performance and low complexity decoding algorithms for 5G Low-Density Parity-Check codes. *J. Commun.* **2022**, *17*, 358–364. [CrossRef]
20. Boncalo, O.; Kolumban-Antal, G.; Amaricai, A.; Savin, V.; Declercq, D. Layered LDPC decoders with efficient memory access scheduling and mapping and built-in support for pipeline hazards mitigation. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2018**, *66*, 1643–1656. [CrossRef]
21. Petrović, V.L.; Marković, M.M.; El Mezeni, D.M., Saranovac, L.V.; Radošević, A. Flexible high throughput QC-LDPC decoder with perfect pipeline conflicts resolution and efficient hardware utilization. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2020**, *67*, 5454–5467. [CrossRef]
22. Cui, H.; Ghaffari, F.; Le, K.; Declercq, D.; Lin, J.; Wang, Z. Design of high performance and area-efficient decoder for 5G LDPC codes. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2020**, *68*, 879–891. [CrossRef]
23. Verma, A.; Shrestha, R. Low computational-complexity SOMS-algorithm and high-throughput decoder architecture for QC-LDPC codes. *IEEE Trans. Veh. Technol.* **2023**, *72*, 66–80. [CrossRef]
24. Tanner, R. A recursive approach to low complexity codes. *IEEE Trans. Inf. Theory* **1981**, *27*, 533–547. [CrossRef]
25. Fossorier, M.P. Quasicyclic low-density parity-check codes from circulant permutation matrices. *IEEE Trans. Inf. Theory* **2004**, *50*, 1788–1793. [CrossRef]
26. Nguyen-Ly, T.T.; Savin, V.; Le, K.; Declercq, D.; Ghaffari, F.; Boncalo, O. Analysis and design of cost-effective, high-throughput LDPC decoders. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **2017**, *26*, 508–521. [CrossRef]
27. Wey, C.L.; Shieh, M.D.; Lin, S.Y. Algorithms of finding the first two minimum values and their hardware implementation. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2008**, *55*, 3430–3437.
28. Nadal, J.; Baghdadi, A. Parallel and flexible 5G LDPC decoder architecture targeting FPGA. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **2021**, *29*, 1141–1151. [CrossRef]
29. Li, Y.; Li, Y.; Ye, N.; Chen, T.; Wang, Z.; Zhang, J. High Throughput Priority-Based Layered QC-LDPC Decoder with Double Update Queues for Mitigating Pipeline Conflicts. *Sensors* **2022**, *22*, 3508. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.