Reduced Check Node Storage for Hardware-Efficient LDPC Decoder

Bich Ngoc TRAN-THI, Trong Hai LE

Faculty of Aviation Operations, Vietnam Aviation Academy, Ho Chi Minh City, Vietnam

ngocttb@vaa.edu.vn

Submitted September 19, 2025 / Accepted November 27, 2025 / Online first January 20, 2026

Abstract. *This paper proposes a hardware- and memory-efficient architecture for Low-Density Parity-Check (LDPC) decoding, targeting enhanced-performance applications with constrained resources. The design integrates two novel techniques: (i) the Variable Single minimum Min-Sum (VSMS) algorithm, which reduces hardware complexity by identifying the first minimum value and its position during check node processing, while improving error correction through a correction factor applied in variable node updates; and (ii) a memory splitting strategy that exploits the structural properties of LDPC codes to optimize memory usage. Implementation on a Xilinx Kintex UltraScale+ (xcku5p) FPGA demonstrates a reduction in storage requirements by over 46.2% compared to conventional decoders. Furthermore, the proposed decoder achieves a performance gain of up to 0.38 dB at a Bit Error Rate (BER) of 10^{-8}, outperforming traditional Min-Sum-based approaches.*

Keywords

5G NR, error correction, FPGA, LDPC decoder, VSMS algorithm

1. Introduction

Error Correction Codes (ECCs) are commonly used in modern wireless communication systems to enhance reliability and greatly decrease the signal-to-noise ratio (SNR) needed for accurate data transfer over noisy channels. These codes include controlled redundancy, enabling decoders to detect and correct transmission errors. Among various ECC schemes, Low-Density Parity-Check (LDPC) codes are widely used in many communication standards and applications due to their excellent error-correction performance and ability to approach the Shannon limit [1]. Notable examples include second-generation digital satellite video transmission for satellite communications [2], WiMAX [3], WiGig [4], Wi-Fi [5], fiber-optic communication systems [6], [7], data storage systems [8], [9], the Advanced Television Systems Committee (ATSC) 3.0 broadcasting standard [10] and Fifth Generation (5G) New Radio (NR) [11].

For LDPC codes, Belief Propagation (BP) decoding, sometimes referred to as Sum-Product (SP) decoding on factor graphs [12], provides superior error-correction performance among decoding techniques. However, considerable memory needs and computational complexity are the price paid for this performance [13]. To address these issues, by employing max-log approximations for check node messages, the Min-Sum (MS) decoding technique aims to lower the computational complexity of the BP. The MS decoding is easier and more effective for hardware implementation because it simply needs adds and comparisons [14]. However, the standard MS algorithm exhibits suboptimal decoding performance due to the overestimation of check node messages, which leads to degraded error correction capability. To address this limitation, several improved variants have been proposed to balance decoding performance with implementation complexity. Among these, the Normalized Min-Sum (NMS) and Offset Min-Sum (OMS) algorithms are widely recognized [15]. These algorithms mitigate the overestimation problem by incorporating a scaling factor or an offset during the message update process. Building on these approaches, many additional refinements have been introduced to achieve closer to optimal decoding [11], [16–20]. These enhancements are significant because they can provide near-optimal error correction performance while maintaining low computational costs, making them suitable for hardware implementations in resource-constrained systems.

Not only does it offer high throughput and strong correction performance, but the decoding algorithm used in LDPC decoding also requires substantial computational resources and memory. Notably, memory usage contributes significantly to power consumption. Therefore, reducing memory needs is crucial for designing decoders. Many studies have focused on optimizing memory. In [21], two important strategies for reducing memory usage are implemented. First, the authors used the MS algorithm, which minimizes check-node storage by keeping only the first minimum and second-minimum magnitudes along with their signs, the first minimum position index, rather than storing all iterative messages. Second, the iterative messages are compressed using a low bit-width fixed-point representation (for example, 3 bits), which further reduces the amount of required memory. The authors of the study [22]

DOI: 10.13164/re.2026.0056

improved memory utilization by modifying the MS algorithm and compressing extrinsic data. They applied lower-bit quantization to the difference between the first and second minima. Instead of storing entire input messages, they only retained the first minimum value, the difference between the first and second minima, the signs, and the index of the position of the first minimum. Another effective way to optimize hardware utilization is to use an approximation calculation for check node processing. Instead of calculating the two minimum values from the input Variable-To-Check (VTC) messages, solely the first minimum value is focused on computing [16], [17]. The authors [23] reduce memory consumption by eliminating the First-In, First-Out (FIFO) buffer and the dedicated a Posteriori Probability (APP) memory by reusing the variable-to-check magnitude memory as an accumulator. Additionally, instead of saving the first minimum value, they only keep the second minimum and use a threshold-based approach to estimate the first. Together, these techniques significantly lower memory usage and decoder area, making the design more suitable for hardware-constrained implementations. To reduce memory usage, the authors in [18] implemented a split storage method that separates CTV memory based on the type of layer. This approach involves storing partial data for low-degree layers while keeping full data for other layers, which leads to a more efficient memory allocation. Additionally, they utilized layer merging to process orthogonal layers concurrently, significantly decreasing the overall number of clock cycles and the memory depth. The authors [24] proposed the Split-Row decoding approach, which divides each row processor into two simplified, practically independent halves. This reduces connection complexity, minimizes memory requests, and enhances row processing parallelism.

Quasi-cyclic LDPC (QC-LDPC) codes are a class of LDPC codes that are widely used in practice. They are a structured subclass that can adopt irregular degree distributions [25]. They incorporate cyclic submatrices, enabling efficient hardware implementation without degrading decoding performance. QC-LDPC codes are particularly suited for high-reliability applications such as satellite communications and 5G. Their irregular structure enhances error-correction capability and reduces error floors. Notably, 5G QC-LDPC codes demonstrate substantial irregularity in their structural design [26].

Motivated by the strengths and limitations of existing LDPC decoding algorithms, this article focuses on efficiently designing the check node memory unit within a 5G LDPC decoder. It leverages the proposed Variable Single minimum Min-Sum (VSMS) algorithm as the core decoding design. The study introduces an enhanced decoding algorithm derived from the traditional MS method, aimed at improving error correction performance while minimizing hardware complexity. Unlike conventional MS-based approaches that require the identification of two minimum values during Check Node processing, the VSMS algorithm simplifies this step by extracting only the first minimum value and its corresponding index from the VTC input messages. This simplification not only reduces hardware resource usage but also allows for a more compact and efficient memory structure. By employing a split storage strategy that takes advantage of the irregular check node degree distribution characteristic of 5G LDPC codes, the proposed design achieves a memory reduction of over 46.2%, making it highly suitable for resource-constrained Field Programmable Gate Array (FPGA) implementations.

The remainder of this paper is organized as follows: Section 2 introduces the fundamental concepts of LDPC codes and the proposed algorithm. Section 3 presents the architecture of the decoder and the check-node memory block. The simulation results and hardware implementation are presented in Sec. 4. Finally, Section 5 concludes the paper.

2. Definitions and Preliminaries

2.1 Definitions

Consider a codeword of length N, denoted as $\mathbf{c} = (c_1, c_2, \ldots, c_N)$ which is constructed from K information bits and M parity-check bits, where $M = N - K$. The parity-check bits are used to detect errors and, in some cases, to correct them. The codeword $\mathbf{c}$ is modulated using Binary Phase-Shift Keying (BPSK), producing the signal: $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ which is transmitted over an Additive White Gaussian Noise (AWGN) channel. At the decoder input, the received signal is $\mathbf{y} = \mathbf{x} + \mathbf{z}$ where $\mathbf{z}$ is a Gaussian random variable with zero mean and variance [27]

$$\sigma^2 = N_0 / 2 \tag{1}$$

where N_0 denotes the noise power spectral density. The SNR is expressed as [28]

$$E_\mathrm{s} / N_0 \tag{2}$$

where E_s is the energy per symbol. This ratio can also be represented in terms of the energy per bit as [28]

$$\frac{E_\mathrm{b}}{N_0} = \frac{N}{K} \times \frac{E_\mathrm{s}}{N_0}. \tag{3}$$

LDPC codes are a type of linear error-correcting code represented by using a sparse parity-check matrix, denoted as $\mathbf{H}$, or through a graphical representation called a Tanner graph [29], [30]. In the Tanner graph, there are two types of nodes: variable nodes (VNs) and check nodes (CNs). VNs represent the codeword bits and correspond to the columns of the matrix $\mathbf{H}$. CNs define parity-check equations and are associated with the rows of $\mathbf{H}$. An edge is drawn between a VN and a CN when the corresponding element in $\mathbf{H}$ is non-zero. During decoding, the Tanner graph is used to pass messages iteratively along these edges to improve the estimate of the original codeword. In the matrix $\mathbf{H}$, let d_c denote the number of entries equal to 1 in each row, and d_v denote the number of entries equal to 1 in each column. The values d_c and d_v are called the check node degree and the variable node degree, respectively. An LDPC code is called regular if d_c is the same for all rows and d_v is the same for all columns. LDPC codes are generally categorized into two types: regular and irregular. Regular LDPC codes have uniform node degrees, while irregular LDPC codes allow variable degrees to improve performance.

QC-LDPC codes are structured based on circulant matrices, which allows the use of layered scheduling techniques to improve decoding performance. In the general case, each decoding layer may consist of several consecutive rows of the base matrix **B**, provided that these rows do not overlap. The term non-overlapping means that, within the same column of **B**, there cannot be more than one non-negative entry. Each non-negative element of the matrix **B** is replaced by a circulant matrix. Consequently, in each column of the matrix **H**, there is exactly one non-negative entry within each decoding layer. Therefore, each decoding layer in the layered scheduling technique consists of Z consecutive rows of the matrix **H** (where Z is the expansion factor), corresponding to one row of the matrix **B**. A full decoding iteration is completed once all CNs of the matrix **H** have updated their information.

Notation: $\alpha_{m,n}$ denotes the message passed from VN n to CN m (VTC message); $\beta_{m,n}$ denotes the message passed from CN m to VN n (CTV message). γ_n denotes the channel message. $\tilde{\gamma}_n$ is the updated information. For hardware design, assume the number of bits used to represent $\alpha_{m,n}$, γ_n, $\tilde{\gamma}_n$ is $\tilde{q}$ bits, while the message $\beta_{m,n}$ is represented with q ($q < \tilde{q}$). The set of possible values of a $\tilde{q}$ bit message is denoted by

$$\tilde{M} = (-\tilde{Q}, \ldots, -1, 0, +1, \ldots, +\tilde{Q}) \quad (4)$$

where $\tilde{Q} = 2^{\tilde{q}} - 1$.

Similarly, the set of possible values of a q bit message is denoted by

$$M = (-Q, \ldots, -1, 0, +1, \ldots, +Q) \quad (5)$$

where $Q = 2^q - 1$.

LDPC codes offer several key advantages that make them ideal for contemporary communication networks, particularly for high-speed data transmission and dependable communication. LDPC codes used in 5G (also known as 5G QC-LDPC codes) are defined by two base matrices, BG1 and BG2, with 51 expansion factors designated as Z [26]. The base matrices BG1 and BG2 have the same structural design. The expansion factor Z ranges from 2 to 384. Due to the wide range of expansion values, it can support various information block lengths and different code rates. It is important to note that 5G LDPC codes are significantly irregular in both VN degrees and CN degrees. This study will focus on the BG1 base matrix. In this BG1, CN degrees (d_c) range from 3 to 19, while VN degrees (d_v) vary from 1 to 30. Notably, the first four rows have the highest CN degree of 19. This wide variation in CN degrees considerably impacts hardware design. Additionally, there are 42 columns with variable nodes of degree 1, representing 62% of all variable nodes. These low-degree variable nodes are more prone to errors during decoding because they receive less protection during the parity-check process. The CN degrees of the BG1 5G LDPC code are listed in Tab. 1.

Check Node Degree (d_c)	3	4	5	6	7	8	9	10	19
Number of Rows in BG1	1	5	18	8	5	2	2	1	4

Tab. 1. Statistics of Check Node Degrees in BG1.

Improving LDPC decoding algorithms remains a key research challenge, as it requires balancing hardware complexity, resource usage, and error-correction performance. The choice of algorithm not only affects decoding efficiency but also impacts flexibility, parallelism, and convergence speed—factors directly tied to hardware cost. As mentioned above, the MS algorithm offers a favorable trade-off by reducing hardware complexity compared to BP. However, this comes at the expense of significant performance loss. To address this limitation, recent studies have proposed various modifications to enhance MS decoding, particularly in CN and VN processing [15]. Building on this line of research, the following section presents the MS decoding algorithm and introduces an improved approach developed in this work.

2.2 Decoding Algorithm

Assume the LDPC code is defined by a parity-check matrix of size $M \times N$. H is the Tanner graph of the LDPC code [30]. The set of CN connected to the n-th VN, where $n = 1, 2, \ldots, N$, is denoted by $H(n)$; the set of VNs connected to the m-th CN, where $m = 1, 2, \ldots, M$, is denoted by $H(m)$. The set $H(m)\backslash n$ represents all VNs in $H(m)$ except the n-th VN; similarly, $H(n)\backslash m$ represents all CNs in $H(n)$ except the m-th CN. The messages passed from CN to VN are denoted by $\beta_{m,n}$; the messages $\alpha_{m,n}$ are passed from VN to CN.

The conventional MS algorithm* [15]*:

The iterative MS decoding includes four steps: initialization, CN processing, VN processing, and A-Posteriori (AP) update processing. These operations occur in multiple rounds, known as iterations. The scheduling of the LDPC decoding process determines the order in which the VN and CN nodes are executed, or whether multiple nodes can be processed in parallel. There are several types of scheduling, but the two most common are flooding scheduling [31] and layered scheduling [32]. The iterative layered MS decoding algorithm is described as follows:

Initialization: AP information update $\tilde{\gamma}_n$ and a priori information γ_n are generated for each VN n, and CTV messages $\beta_{m,n}$ transmitted from CN m to VN n are set to zero.

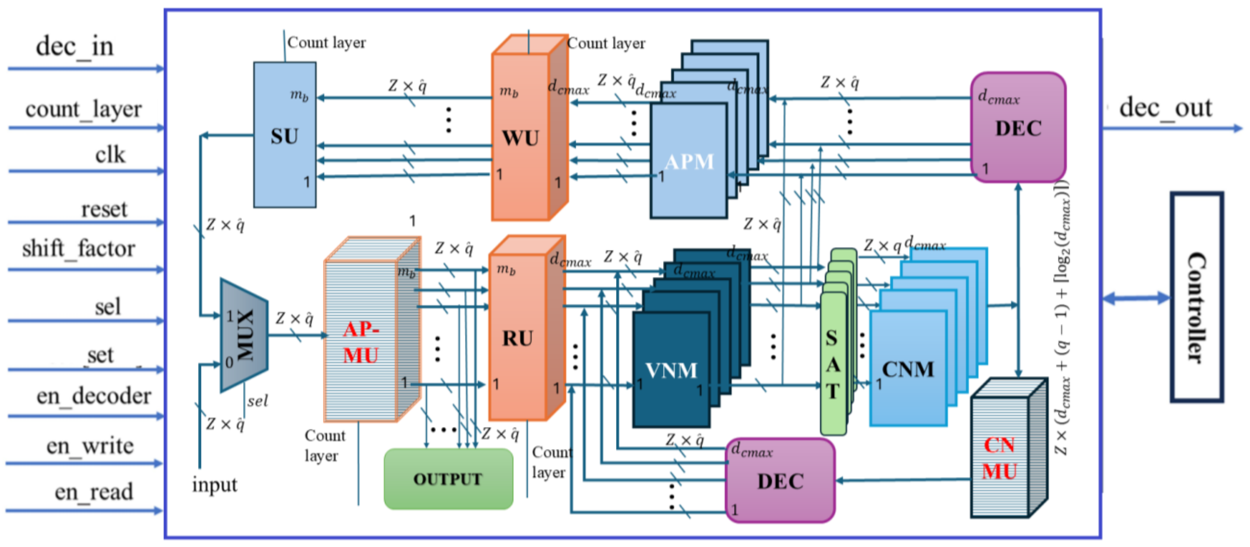


Fig. 1. Structural diagram of the proposed LDPC decoder design.

$$\tilde{\gamma}_n = \gamma_n = \log \frac{\Pr(x_n = 0 \mid y_n)}{\Pr(x_n = 1 \mid y_n)}, \quad (6)$$

$$\beta_{m,n} = 0.$$

Iteration: Each decoding iteration is carried out in the following sequence:

VN Processing: The messages sent from the VN to the CN, $\alpha_{m,n}$, are updated as below:

$$\alpha_{m,n} = \gamma_n + \beta_{m,n}\,. \quad (7)$$

CN Processing: The messages sent from the CN to the VN $\beta_{m,n'}$ are updated based on the current VTC messages $\alpha_{m,n'}$

$$\beta_{m,n} = \prod_{n' \in H(m) \setminus n} \operatorname{sgn}(\alpha_{m,n'}) \cdot \min_{n' \in H(m) \setminus n} |\alpha_{m,n'}|\,. \quad (8)$$

From a hardware perspective,

$$\beta_{m,n} = \left(\prod_{n' \in H(m) \setminus n} \operatorname{sgn}(\alpha_{m,n'}) \right) \times \begin{cases} \text{min2}, & \text{if } |\alpha_{m,n}| = \text{min1} \\ \text{min1}, & \text{otherwise} \end{cases} \quad (9)$$

where min1 and min2 are the first two minimum values among all VTC input messages; index_min1 indicates the position of min1. If there is more than one min1 value, index_min1 will be set to the smallest index.

AP-update processing after each iteration: The AP-update information for each code bit $\tilde{\gamma}_n$ is computed based on the updated extrinsic messages (i.e. $\alpha_{m,n}$ and $\beta_{m,n}$).

$$\tilde{\gamma}_n = \alpha_{m,n} + \beta_{m,n}\,. \quad (10)$$

In this work, the decoder stops when it reaches the maximum number of iterations.

The proposed algorithm:

Inspired by the Single-Minimum MS (smMS) and NMS algorithms [15, 16, 17], a modified algorithm, that aims to achieve improved decoding gain and memory efficiency, is presented. This new algorithm builds on the MS algorithm and incorporates modifications in both the CN and VN processes, as outlined below. During the CN processing, only the value of the first minimum (min1) and its corresponding index (index_min1) are determined. The second minimum (min2) is then estimated using the value of min1 and a modified factor, μ, as outlined below:

$$\beta_{m,n} = \left(\prod_{n' \in H(m) \setminus n} \operatorname{sgn}(\alpha_{m,n'}) \right) \times \begin{cases} \text{min1} + \mu, & \text{if } |\alpha_{m,n}| = \text{min1} \\ \text{min1}, & \text{otherwise} \end{cases} \quad (11)$$

To enhance decoding performance, an additional normalized factor δ is applied to the VN processing as detailed below:

$$\alpha_{m,n} = \gamma_n + \delta \cdot \beta_{m,n} \quad (12)$$

where $\mu > 0$; $0 < \delta < 1$.

This proposed algorithm is named the Variable Single minimum Min-Sum (VSMS) algorithm.

The correction factors (μ and δ) are optimized through a joint application of the Density Evolution (DE) method [29] and simulation-based techniques. The optimal values (μ, δ) for 5G LDPC codes are 0.75 and 0.75.

3. Decoder Design

3.1 The Decoder Architecture

The BG1 matrix with specified dimensions $m_b \times n_b = 24 \times 46$ and an expansion factor $Z = 192$, resulting in generated $L = m_b = 24$ layers, is utilized in this design. The parity-check matrix $\mathbf{H}$ will also have specific dimensions $M \times N$, where $M = m_b \times Z = 24 \times 192 = 4608$ and $N = n_b \times Z = 46 \times 192 = 8832$. The block diagram of the LDPC decoder based on the layered MS algorithm is shown in Fig. 1.

The input data γ_n is read sequentially from the transmission channel (channel data) into the AP Memory Unit (AP-MU). Information from the transmission channel γ_n or the updated message from the previous layer $\tilde{\gamma}_n$ is chosen by the MUX block to be stored in the AP-MU. Notably, $\tilde{q} = 6$ bits are used to represent both kinds of information, while γ_n is only used during initialization. Each clock pulse cycle reads $Z \times \tilde{q}$ bits. Read Units (RUs) are responsible for reordering data read from the AP-MU according to layer grouping, to ensure that VN Modules (VNMs) and CN Modules (CNMs) are processed in the correct sequence. In other words, the RU accesses data at corresponding positions in the base matrix, where matrix elements have non-negative values. The RU block also has the task of rearranging the rows of the matrix **H** so that the CN degrees decrease in order. Additionally, these units perform cyclic permutations with the number of steps equal to the non-negative element value of the BG1 matrix. The RU has n_b input ports and d_cmax output ports, where $d_\mathrm{cmax} = 19$ is the maximum CN degree. The Write Unit (WU) performs the opposite function of the RU.

VNMs update VTC information during the VN processing stage. The decoder has a total of d_cmax VNMs. In this study, the VTC information $\alpha_{m,n}$ is calculated using the previous layer's updated CTV information $\beta_{m,n}$, the input information γ_n and the normalized factor (corresponding to (12)). Before sending the data to the CNMs, SAT blocks reduce the number of bits that represent the VTC information from $\tilde{q}$ bits to q bits. The goal is to decrease the Check Node Memory Unit (CNMU)'s storage capacity without sacrificing data dependability. Therefore, the bit width used to represent the CTV information $\beta_{m,n}$ is selected such that $q = 4 < \tilde{q}$. CTV information is updated by CNMs, which are utilized in the CN processing step (11). The CTV message $\beta_{m,n}$ is calculated using the VTC message $\alpha_{m,n}$ of the current layer. Each CNM has d_cmax inputs and one output. Updates to the AP-update $\tilde{\gamma}_n$ are made in the AP Modules (APMs). There are d_cmax APMs in the decoder. To process each layer, the LDPC decoder operates over two consecutive clock cycles. In the first clock cycle, the AP-MU memory is set to “read” mode to provide the codewords corresponding to the layer of the BG1 matrix. Then, the CN and VN processing are carried out in sequence. In the following clock cycle, the decoder performs the update of the codewords. After that, the updated codewords are written back to the AP-MU memory, meaning the AP-MU memory is now in “write” mode. This process continues until the final layer (layer L) is reached, which completes one iteration. In this work, the entire decoding process is performed over 10 iterations. After the 10th iteration is completed, the output of the decoder will contain the decoded codewords. These codewords are stored in the first 22 blocks of the AP-MU, corresponding to $22 \times Z$ bits of information. To optimize memory usage, the CNM outputs data in a compressed format, necessitating a decompression block (DEC) to convert it back to an uncompressed form. Additionally, to minimize hardware resource usage in the CNM, the DEC calculates the second minimum value. In this work, two blocks, VNM and APM, are merged into a single block, and an additional control signal “sel” is used to select whether the block performs the VNM function or the APM function. The controller drives this selection signal so that the VNM mode is chosen in the first clock cycle and the APM mode is chosen in the second clock cycle. The SU distributes the data to prevent data write congestion in AP-MU. The controller’s function is to ensure the decoder operates correctly, meaning that all blocks execute in the correct sequence, perform read/write operations to/from memory properly, and handle other control functions.

3.2 Storage Module

The decoder is designed according to a layered scheduling scheme in which the CTV messages need to be stored for processing in subsequent iterations (in CNMU). Therefore, memory usually occupies a significant portion of the overall chip area and is a major contributor to power consumption in the LDPC decoder [13]. If the code length of the LDPC code is large, the required memory capacity will increase, leading to higher energy consumption. Hence, during the decoding process, it is necessary to apply a method to efficiently save memory usage. In an LDPC decoder, the CNM is responsible for computing messages from CNs to VNs (CTV). The output of the CNM block $[\beta_{m,n_1}, \ldots, \beta_{m,n_{d_\mathrm{cmax}}}]$ (uncompressed format) is stored in the CTV memory (CNMU) where $[n_1, \ldots, n_{d_\mathrm{cmax}}]$ are the variable nodes connected to the m-th check node. Suppose the number of bits representing information is $q = 4$ and the CN degree is $d_\mathrm{cmax} = 19$.

$$m_\mathrm{b} \times Z \times d_\mathrm{cmax} \times q = m_\mathrm{b} \times 76 \times Z \text{ (bits).} \tag{13}$$

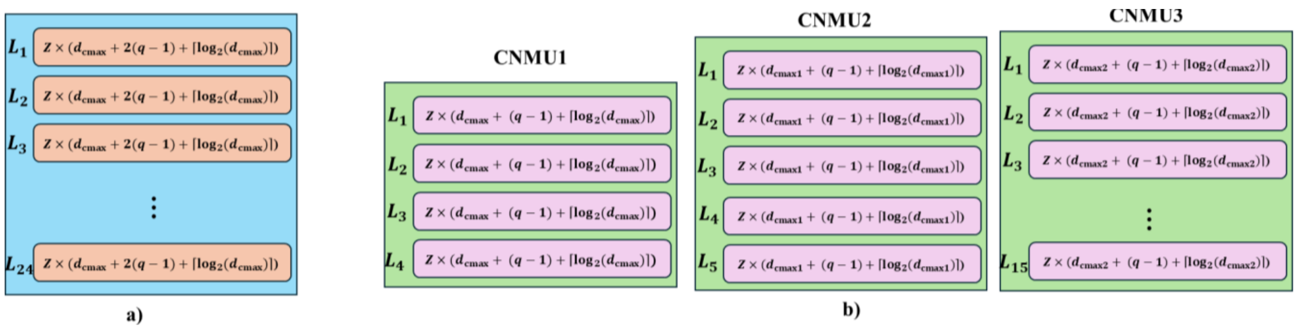
To save memory space, instead of storing all the CN information $[\beta_{m,n_1}, \ldots, \beta_{m,n_{d_\mathrm{cmax}}}]$, the CNMU will only store the first two minimum values of the VTC input information, the position of the first minimum value, and the sign bits [21]. This method is also known as information compression. Consequently, the memory size required for the check nodes becomes:

$$\begin{aligned} & m_\mathrm{b} \times Z \times \left(2 \times (q-1) + d_\mathrm{cmax} + \lceil \log_2 d_\mathrm{cmax} \rceil\right) \\ & \quad = 30 \times Z \times m_\mathrm{b} \text{ (bits).} \end{aligned} \tag{14}$$

Using information compression, the storage space has been reduced by nearly 60.5% compared to conventional storage methods, as seen in (13) and (14). During the CN processing, memory is saved by not determining the second minimum value. The output of the CNM is the first minimum value and its position, which are stored in the CNMU. In this case, the second minimum value does not need to be stored in this memory. Therefore, the memory width of the CNMU is determined by

$$W = Z \times \left(q - 1 + \lceil \log_2(d_\mathrm{cmax}) \rceil + d_\mathrm{cmax}\right). \tag{15}$$

In case : $d_\mathrm{cmax} = 19$, $q = 4$, thus: $W = 27Z$. (16)



L_1 $Z \times (d_{cmax} + 2(q-1) + \lceil \log_2(d_{cmax}) \rceil)$
L_2 $Z \times (d_{cmax} + 2(q-1) + \lceil \log_2(d_{cmax}) \rceil)$
L_3 $Z \times (d_{cmax} + 2(q-1) + \lceil \log_2(d_{cmax}) \rceil)$
L_{24} $Z \times (d_{cmax} + 2(q-1) + \lceil \log_2(d_{cmax}) \rceil)$
a)
CNMU1
L_1–L_4 $Z \times (d_{cmax} + (q-1) + \lceil \log_2(d_{cmax}) \rceil)$
CNMU2
L_1–L_5 $Z \times (d_{cmax1} + (q-1) + \lceil \log_2(d_{cmax1}) \rceil)$
CNMU3
L_1–L_{15} $Z \times (d_{cmax2} + (q-1) + \lceil \log_2(d_{cmax2}) \rceil)$
b)

Fig. 2. The detailed structure of CTV memory for (a) the conventional MS and (b) the proposed decoder.

Since the CTV messages at the output of the CNM from all layers must be stored in memory, the CNMU must contain

$$m_b \times Z \times \left(q - 1 + \lceil \log_2(d_{cmax}) \rceil + d_{cmax}\right). \tag{17}$$

The width of the CNMU depends on the bit width q, the degree of the maximum CN d_{cmax}, and the expansion factor Z. The depth of the CNMU equals the number of decoding layers. In this design, the number of decoding layers is equal to the number of rows in the BG1 matrix: $L = m_b = 24$. From Tab. 1 on the distribution of CN degrees, it is clear that the maximum CN degree, $d_{cmax} = 19$, appears only in the first four decoding layers. Degrees $d_c = 10, 9, 8$ occur in just one or two layers, while in most remaining layers, $d_c < 7$. If all CNMU memory blocks are implemented with the same width as in (15), it can lead to a significant amount of unnecessary memory usage. Therefore, a memory-splitting approach that takes into account variations in CN degrees is suggested. Here, the CNMU is split into three subblocks: CNMU1, CNMU2 and CNMU3 as depicted in Fig. 2.

The CTV messages from the first four layers ($L1 = 4$) are stored in CNMU1, which corresponds to a CN degree of $d_{cmax} = 19$. The CTV messages from the five layers ($L2 = 5$), with CN degrees $d_c = 10, 9, 8$, are stored in CNMU2 (for simplicity, the maximum CN degree is chosen $d_{cmax1} = 10$). The CTV messages from the remaining layers ($L3 = 15$), where the CN degrees are less than 7 ($d_c < 7$), are stored in the third memory CNMU3. To simplify hardware implementation, the maximum CN degree in this case is set to 7 ($d_{cmax2} = 7$).

From (15), the width of the CNMU1 is defined as

$$W1 = 27 \times Z. \tag{18}$$

The depth of CNMU1 is $L1 = 4$. The width of the CNMU2 is defined as

$$W2 = 17 \times Z. \tag{19}$$

The depth of CNMU2 is $L2 = 5$. The width of the CNMU3 is defined as

$$W3 = 13 \times Z. \tag{20}$$

The depth of CNMU3 is $L3 = 15$. The CNMU size of some decoders is listed in Tab. 2.

MS decoder's			VSMS decoder's					
			Before splitting			After splitting		
W	*D*	Total	*W*	*D*	Total	*W*	*D*	Total
30*Z*	24	720*Z*	27*Z*	24	648*Z*	13*Z*	15	401*Z*
						17*Z*	5	
						27*Z*	4	

Tab. 2. Comparison of the CNMU size of various decoders using the splitting technique (*W*-Width; *D*-Depth).

Based on Tab. 2, the VSMS's CNMU consists of the CNMU1 of $L1 \times 27 \times Z = 20736$ bits; the CNMU2 of $L2 \times 17 \times Z = 16320$ bits, and the CNMU3 of $L3 \times 13 \times Z = 37440$ bits. The proposed VSMS decoders only need to compute the first minimum value and its position during the CN processing. Therefore, before applying the memory splitting technique, the memory requirement of the VSMS decoder is approximately 10% lower than that of the CNMU in the MS decoder. Based on specific characteristics of 5G LDPC codes, after splitting storage, the CNMU in the proposed decoder achieves a storage reduction of approximately 44.3% compared to the MS decoder, and about 38.12% compared to the original undivided structure.

4. Simulation Results and Hardware Implementation

4.1 Simulation Results

Monte Carlo simulations are conducted using MATLAB R2022b for various algorithms with BG1 5G LDPC codes and code rates of 1/2 and 2/3 and codeword lengths of 8832 and 6720, respectively. Several previous studies on MS–based LDPC decoders have already demonstrated that the decoding performance (in terms of Bit Error Rate (BER) or Frame Error Rate (FER) versus E_b/N_0) improves to near-optimal levels as the maximum number of iterations increases. However, throughput decreases and more hardware resources are required [29, 33, 34]. Additionally, in this work, we focused on the architectural and memory-saving aspects of the decoder rather than on an exhaustive performance evaluation across different iteration counts in our results. Therefore, the iteration versus E_b/N_0 curves are not included. In our simulations and implementation, the LDPC decoder was run with a fixed maximum of 10 iterations. This value was chosen to balance performance and complexity and is consistent with many reported LDPC implementations. The expansion factor is $Z = 192$. The correction factors for the VSMS are $\mu = 0.75$, $\delta = 0.75$. The decoding performance of different algorithms is simulated, including Hybrid Offset Min-Sum (HOMS) ($\beta = 0.5$, $\delta = 0.375$) [11], smMS (with the offset parameter set to 1) [17], NMS ($\beta = 0.75$) and MS [15]. The simulation results are illustrated in Figs. 3 and 4.

In CN processing, both the VSMS and HOMS algorithms only need to compute the first minimum value of the input messages and its position. The proposed algorithms also refine the VN processing to reduce the overestimation of information inherent in the MS algorithm. Due to the specific characteristics of 5G LDPC codes, the extended parity-check part of the base matrix contains many degree-1 VNs that are connected to only a single CN. These VNs are typically weakly protected and prone to errors, which significantly affect decoding performance. The VSMS algorithm employs correction factors in both the VN and CN processing to mitigate the error susceptibility of these VNs. From Figs. 3 and 4, the VSMS and HOMS algorithms achieve the same decoding performance. At BER values of 10^{-4} and 10^{-8}, the error-correction performance of the proposed VSMS and HOMS algorithms outperforms that of the smMS and MS algorithms by up to 0.32 dB and 0.38 dB, respectively. The smMS algorithm exhibits an error floor at a BER of 10^{-4}.

The impact of modulation schemes and channel conditions on the proposed VSMS algorithm is investigated, as shown in Fig. 5. The performance evaluation is conducted under various modulation schemes, including BPSK, QPSK (Quadrature Phase-Shift Keying) and 16-QAM (Quadrature Amplitude Modulation) over two different channel conditions: AWGN and AWGN combined with frequency–nonselective Rayleigh fading (AWGN + Rayleigh fading).

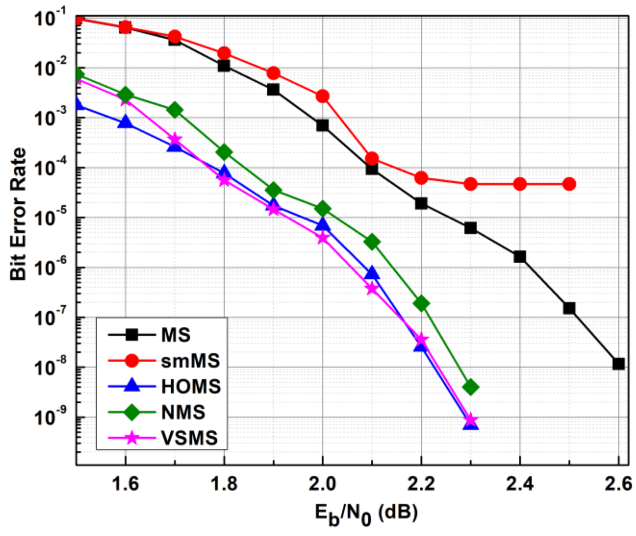


Fig. 3. Bit Error Rate performance of various decoders with block size (8832, 4608) at code rate ½.

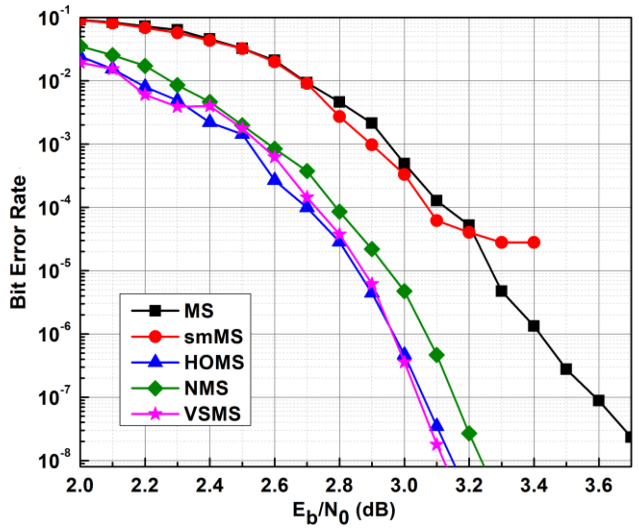


Fig. 4. Bit Error Rate performance of various decoders with block size (6720, 2496) at code rate 2/3.

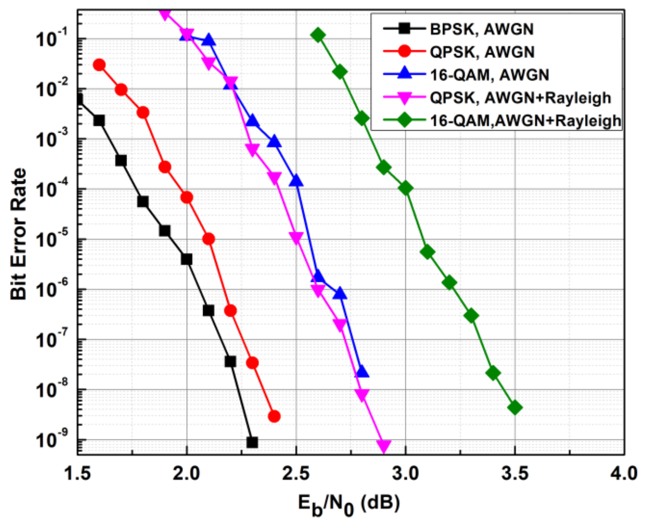


Fig. 5. The decoding performance of the proposed VSMS algorithm with a code length of 8832 and a code rate of 1/2 is evaluated under various modulation schemes: BPSK under AWGN is included as a baseline, while QPSK and 16-QAM are reported over AWGN and AWGN+Rayleigh channels.

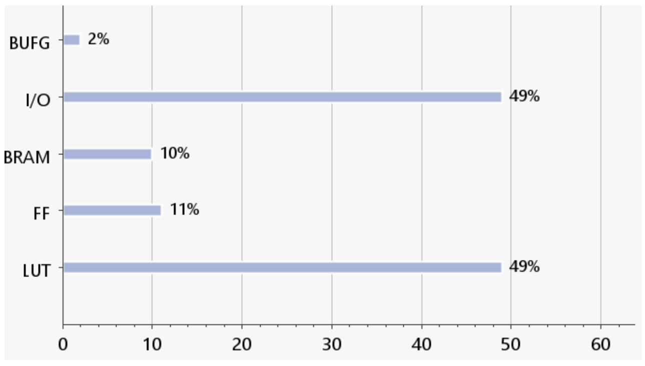


Fig. 6. An overview chart of FPGA resource utilization generated by Vivado after the implementation process.

From Fig. 5, at a BER of 10^{-8}, the proposed VSMS algorithm achieves a decoding gain of 0.075 dB and 0.55 dB when using BPSK modulation compared to QPSK and 16-QAM, respectively. The AWGN channel model provides better decoding performance than the Rayleigh fading channel. This demonstrates that the LDPC decoder employing the VSMS algorithm can effectively utilize QPSK and 16-QAM modulation schemes, together with the AWGN channel, as adopted in 5G NR communication networks [35]. Figure 5 evaluates the decoding performance of the proposed VSMS under different modulation schemes and channel conditions to determine its adaptability to the 5G New Radio network. To keep Fig. 5 readable and to avoid overloading it with curves, the numerous state-of-the-art algorithms are not included in that figure. However, the trends observed in Figs. 3 and 4 suggest that, under the same decoding conditions, the proposed method performs comparably to these state-of-the-art algorithms. Based on those trends, we expect comparable relative performance for the cases illustrated in Fig. 5 when the decoding conditions are the same. Therefore, for decoding performance, the BPSK/AWGN baseline facilitates fair comparison across studies, while the QPSK/16-QAM results establish 5G relevance.

4.2 Implementation Results

The LDPC decoder was implemented using the Verilog HDL (Verilog Hardware Description Language). The hardware results were obtained on a Kintex UltraScale+ FPGA. The results of the FPGA, synthesized and implemented after place and route, are achieved by utilizing Xilinx Vivado 2021.2. A 5G LDPC code with a codeword length of 8832 and a code rate of 1/2 was used. The matrix BG1 is chosen. The maximum number of decoding iterations was set to 10. The message information was quantized with $(q,\tilde{q}) = (4,6)$. The throughput is calculated as [11]:

$$T = \frac{N \times F_\mathrm{m}}{L \times n_\mathrm{i} \times n_\mathrm{c}}\,[\mathrm{Mbps}] \tag{21}$$

where L is the number of decoding layers, n_c is the number of clock cycles required to process one decoding layer, n_i is the maximum number of iterations, N is the codeword length, and F_m [MHz] is the maximum operating frequency.

The resource utilization of the FPGA, as generated by Vivado after the implementation process, is visually depicted in Fig. 6. This figure illustrates that the design heavily depends on Look-Up Tables (LUTs) and Input/Output (I/O) pins, with each utilizing nearly half of the available resources. The usage of flip-flops (FFs) and Block Random Access Memory (BRAM) is moderate, while the consumption of Global Clock Buffers (BUFG) is minimal. Overall, the design is both logic- and I/O-intensive, but it does not place significant strain on memory or clock resources.

Figure 7 illustrates the placement footprint. The light blue vertical bands represent the LUT/CLB (Configurable Logic Block), which is densely packed across most clock regions. This column-wise arrangement is typical of wide, throughput-oriented data paths. In contrast, the darker gaps and narrow colored stripes identify the hard macro/IO columns, including BRAM (Block RAM), DSP (Digital Signal Processing), clock spines, and IO banks. These areas serve as keep-out zones and routing channels, resulting in a sparser region toward the right edge near the IO columns. Despite the asymmetrical distribution of resources and the routing pressure surrounding the hard resources, the implementation successfully achieves timing closure.

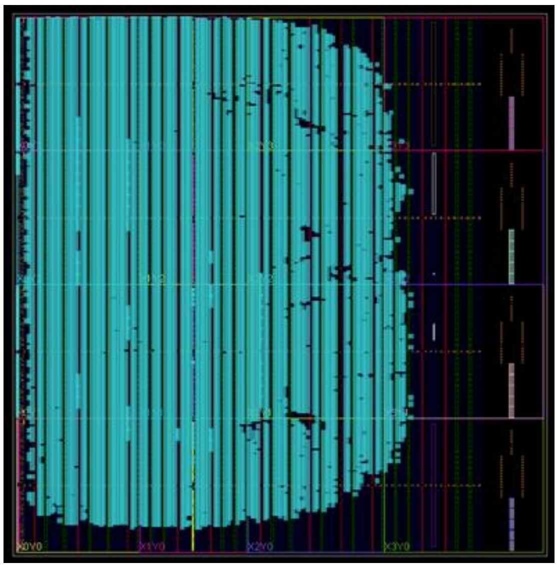
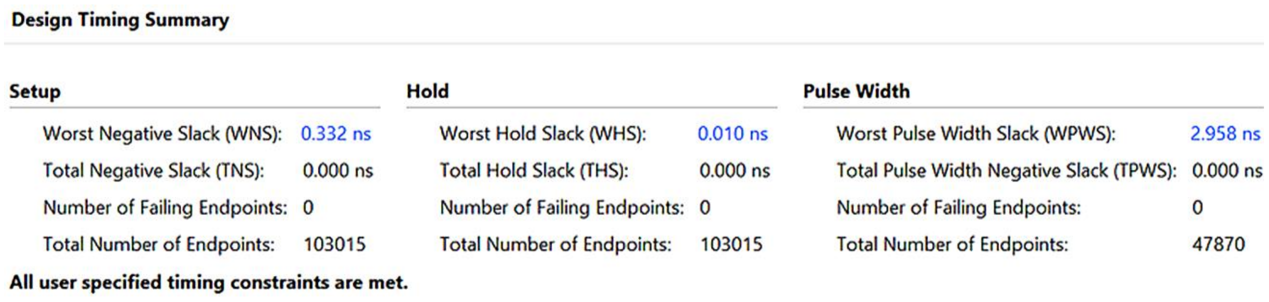


Fig. 7. Graphical visualization of resource utilization by the proposed decoder.


Design Timing Summary

Setup		**Hold**		**Pulse Width**	
Worst Negative Slack (WNS):	0.332 ns	Worst Hold Slack (WHS):	0.010 ns	Worst Pulse Width Slack (WPWS):	2.958 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	103015	Total Number of Endpoints:	103015	Total Number of Endpoints:	47870

All user specified timing constraints are met.

Fig. 8. Timing report of the implemented design in nanoseconds.

Figure 8 displays the timing report for the implemented design. The design successfully meets all user-specified timing constraints at a large scale, featuring 103015 setup and hold endpoints along with 47870 pulse-width endpoints. This indicates a high level of connectivity and significant routing complexity, with no failing endpoints reported. As illustrated in Fig. 8, timing closure at a 7 ns clock period is achieved with a Worst Negative Slack (WNS) of +0.332 ns. This indicates a critical path delay of 6.668 ns and a maximum operating frequency of approximately 150 MHz [36]. After the place-and-route process, hold timing is also met with a small but positive margin; no setup/hold violations remain.

The proposed VSMS decoders employ techniques such as layered scheduling, semi-parallel architecture, improved check-node processing units, and CNMU optimization. The hardware results are compared with several reference LDPC decoders, as summarized in Tab. 3. From Tab. 3, it can be observed that although the proposed VSMS decoder has a throughput 27% lower than [40], it still achieves 2.63 Gbps, which is only 7% less than the decoder in [11], and up to 2.5 times higher than the decoders in [38] and [39]. By utilizing a memory-splitting approach that takes advantage of the irregularity found in 5G LDPC codes, the VSMS decoder requires the least amount of memory compared to the other decoders listed in Tab. 3. The proposed VSMS decoder achieves a reduction in memory usage of roughly 46.2% as compared to the decoder presented by [37].

LDPC decoders can be implemented with various features, including codeword length, bit-width representation, decoding algorithm, and the number of iterations. To compare different designs, a standardized metric is necessary. For FPGA designs, the Hardware Usage Efficiency (HUE) metric is defined as the amount of hardware resources needed to process one layer of the base matrix to reach a throughput of 1 Mbps. The formula for calculating HUE is shown below:

$$\text{HUE} = \frac{H_\text{u}}{L \times T} \qquad (22)$$

where H_u is the hardware utilization reflecting how many FPGA resources the LDPC decoder utilizes, specifically the number of F7 MUX, F8 MUX, flip-flops, LUTs, and BRAM measured in bits. L is the number of decoding layers. T [Mbps] is the throughput. The unit of HUE is expressed as hardware resources per layer per Mbps. From this definition, it is evident that a lower HUE value is better. The proposed VSMS decoder achieves a HUE of 4.88 hardware resources/(layer·Mbps), which is comparable to the decoder presented in [11] with a HUE of 4.65 hardware resources/(layer·Mbps). In contrast, the VSMS decoder demonstrates a substantial improvement in hardware efficiency, requiring more than ten times fewer resources compared to the decoders reported in [38] and [40].

5. Conclusion

In this work, an LDPC decoder based on the proposed VSMS decoding algorithm has been designed and analyzed. This approach only requires determining the first minimum value of the input VTC messages and its position during the CN processing, significantly reducing hardware resource consumption compared to reported MS-based decoders. Additionally, by utilizing the irregularity of the BG1 base matrix in 5G LDPC codes, the VSMS decoder incorporates an efficient memory splitting technique that achieves over a 46.2% reduction in storage requirements compared to several reference designs. Simulation results further confirm that the proposed decoder enhances decod-

Features \ Reference Year	This work	[37] 2024	[11] 2023	[38] 2021	[39] 2021	[40] 2020
FPGA	Xilinx Kintex Ultrascale+	Kintex-7 xc7vx980t	Xilinx Kintex Ultrascale+	Xilinx Kintex-7	Virtex 7-XC7VX690T	Xilinx Kintex-7
Quantization Bits $(q,\tilde{q})$	(4,6) bits	--	(4,6) bits	(8,8) bits	(5,5) bits	(5,5) bits
Throughput (Gbps)	2.63	--	2.82	0.391-1.1	2.168	3.6
Codeword Length	8832	13440	8832	3456	6528	155
Expansion Factor Z	192	384	192	384	96	31
Maximum Frequency (MHz)	150	304.5	153.5	160	82	700
Code Rate	1/2	2/3	1/2	1/2-5/6	1/3	--
Number of Layers	24	46	24	5-46	46	3
Number of Iterations	10	6	10	8	10	10
Memory Size (kb)	124.5	231.35	173.25	7146	3456	486
Decoding Algorithm	VSMS	MS	HOMS	OMS	OMS	MS
Hardware Utilization H_u	307999	382786	314968	7438394	--	525764
Memory Type	BRAM	Registers	BRAM	BRAM	BRAM	BRAM
HUE	4.88	--	4.65	147.54	--	48.68

Tab. 3. FPGA implementation results and comparison with reference LDPC decoders.

ing performance by up to 0.38 dB compared to traditional MS-based decoders. These findings demonstrate the effectiveness of the VSMS-based design in achieving both hardware efficiency and error-correction performance, making the decoder well-suited for practical implementation in next-generation communication systems such as 5G NR. In future work, we will study the impact of the proposed approach on power consumption. The power optimization problem can be seen as finding the balance among power consumption, hardware resource usage, and operating frequency or throughput.

Acknowledgments

We acknowledge Vietnam Aviation Academy (VAA) for supporting this study.

Availability of Data and Materials

The datasets and source code generated and/or analyzed during the current study are available from the corresponding author on reasonable request. Please contact Bich Ngoc Tran-Thi at [ngocttb@vaa.edu.vn].

References

[1] CHUNG, S.-Y., FORNEY, G. D., RICHARDSON, T. J., et al. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. *IEEE Communications Letters*, 2001, vol. 5, no. 2, p. 58–60. DOI: 10.1109/4234.905935

[2] KIM, S.-M., PARK, C.-S., HWANG, S.-Y. A novel partially parallel architecture for high-throughput LDPC decoder for DVB-S2. *IEEE Transactions on Consumer Electronics*, 2010, vol. 56, no. 2, p. 820–825. DOI: 10.1109/TCE.2010.5506007

[3] FALCAO, G., SILVA, V., SOUSA, L., et al. High coded data rate and multicodeword WiMAX LDPC decoding on Cell/BE. *Electronics Letters*, 2008, vol. 44, no. 24, p. 1415–1417. DOI: 10.1049/el:20081927

[4] MOTOZUKA, H., YOSOKU, N., SAKAMOTO, T., et al. A 6.16 Gb/s 4.7 pJ/bit/iteration LDPC decoder for IEEE 802.11ad standard in 40nm LP-CMOS. In *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. Orlando (FL, USA), 2015, p. 1289–1292. DOI: 10.1109/GlobalSIP.2015.7418406

[5] JUNG, Y., JUNG, Y., LEE, S., et al. Low-complexity multi-way and reconfigurable cyclic shift network of QC-LDPC decoder for Wi-Fi/WiMAX applications. *IEEE Transactions on Consumer Electronics*, 2013, vol. 59, no. 3, p. 467–475. DOI: 10.1109/TCE.2013.6626226

[6] DJORDJEVIC, I. B. PMD compensation in fiber-optic communication systems with direct detection using LDPC-coded OFDM. *Optics Express*, 2007, vol. 15, no. 7, p. 3692–3701. DOI: 10.1364/OE.15.003692

[7] KHAN, M. N., JAMIL, M., HUSSAIN, M. Adaptation of hybrid FSO/RF communication system using puncturing technique. *Radioengineering*, 2016, vol. 25, no. 4, p. 644–651. DOI: 10.13164/re.2016.0644

[8] KURTAS, E. M., KUZNETSOV, A. V., DJURDJEVIC, I. System perspectives for the application of structured LDPC codes to data storage devices. *IEEE Transactions on Magnetics*, 2006, vol. 42, no. 2, p. 200–207. DOI: 10.1109/TMAG.2005.861751

[9] PALENÍK, T., FARKAS, P., RAKUS, M., et al. Analysis of minimal LDPC decoder system on a chip implementation. *Radioengineering*, 2015, vol. 24, no. 3, p. 783–790. DOI: 10.13164/re.2015.0783

[10] JERJI, F., AKAMINE, C. Advanced ISDB-T and ATSC 3.0 LDPC codes performance and complexity comparison. *IEEE Transactions on Broadcasting*, 2021, vol. 68, no. 1, p. 254–262. DOI: 10.1109/TBC.2021.3136763

[11] 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, vol. 12, p. 1–17. DOI: 10.3390/electronics12173667

[12] KSCHISCHANG, F. R., FREY, B. J., LOELIGER, H.-A. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 2002, vol. 47, no. 2, p. 498–519. DOI: 10.1109/18.910572

[13] MANSOUR, M. M., SHANBHAG, N. R. High-throughput LDPC decoders. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2004, vol. 11, no. 6, p. 976–996. DOI: 10.1109/TVLSI.2003.817545

[14] RICHARDSON, T., URBANKE, R. The renaissance of Gallager’s low-density parity-check codes. *IEEE Communications Magazine*, 2003, vol. 41, no. 8, p. 126–131. DOI: 10.1109/MCOM.2003.1222728

[15] DECLERCQ, D., FOSSORIER, M., BIGLIERI, E. *Channel Coding: Theory, Algorithms, and Applications*: *Academic Press Library in Mobile and Wireless Communications*. 1st ed. Academic Press*:* 2016. ISBN: 9780123972231

[16] ANGARITA, F., VALLS, J., ALMENAR, V., et al. Reduced-complexity min-sum algorithm for decoding LDPC codes with low error-floor. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2014, vol. 61, no. 7, p. 2150–2158. DOI: 10.1109/TCSI.2014.2304660

[17] DARABIHA, A., CARUSONE, A. C., KSCHISCHANG, F. R. A bit-serial approximate min-sum LDPC decoder and FPGA implementation. In *2006 IEEE International Symposium on Circuits and Systems*. Kos (Greece), 2006, p. 149–152. DOI: 10.1109/ISCAS.2006.1692544

[18] CUI, H., GHAFFARI, F., LE, K., et al. Design of high-performance and area-efficient decoder for 5G LDPC codes. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2020, vol. 68, no. 2, p. 879–891. DOI: 10.1109/TCSI.2020.3038887

[19] PETROVIĆ, V. L., MARKOVIĆ, M. M., EL MEZENI, D. M., et al. Flexible high throughput QC-LDPC decoder with perfect pipeline conflicts resolution and efficient hardware utilization. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2020, vol. 67, no. 12, p. 5454–5467. DOI: 10.1109/TCSI.2020.3018048

[20] TRAN-THI, B. N., NGUYEN-LY, T. T., HOANG, T. Further improvements in decoding performance for 5G LDPC codes based on modified check-node unit. *Radioengineering*, 2023, vol. 32, no. 2, p. 226–235. DOI: 10.13164/re.2023.0226

[21] ZHANG, K., HUANG, X., WANG, Z. An area-efficient LDPC decoder architecture and implementation for CMMB systems. In *2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors*. Boston (MA, USA), 2009, p. 235–238. DOI: 10.1109/ASAP.2009.34

[22] XIANG, B., SHEN, R., PAN, A., et al. An area-efficient and low-power multirate decoder for quasi-cyclic low-density parity-check

codes. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2009, vol. 18, no. 10, p. 1447–1460. DOI: 10.1109/TVLSI.2009.2025169

[23] LEE, H.-C., LI, M.-R., HU, J.-K., et al. Optimization techniques for the efficient implementation of high-rate layered QC-LDPC decoders. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2016, vol. 64, no. 2, p. 457–470. DOI: 10.1109/TCSI.2016.2612655

[24] MOHSENIN, T., BAAS, B. M. Split-row: A reduced complexity, high throughput LDPC decoder architecture. In *2006 International Conference on Computer Design*. San Jose (CA, USA), 2006, p. 320–325. DOI: 10.1109/ICCD.2006.4380835

[25] FOSSORIER, M. P. Quasicyclic low-density parity-check codes from circulant permutation matrices. *IEEE Transactions on Information Theory*, 2004, vol. 50, no. 8, p. 1788–1793. DOI: 10.1109/TIT.2004.831841

[26] RICHARDSON, T., KUDEKAR, S. Design of low-density parity-check codes for 5G New Radio. *IEEE Communications Magazine*, 2018, vol. 56, no. 3, p. 28–34. DOI: 10.1109/MCOM.2018.1700839

[27] GALLAGER, R. G. *Principles of Digital Communication*. Cambridge, UK: Cambridge University Press, 2012. DOI: 10.1017/CBO9780511813498

[28] PROAKIS, J. G., SALEHI, M. *Digital Communications*. New York: McGraw-Hill, 2001. ISBN: 9780072957167

[29] RICHARDSON, T. J., URBANKE, R. L. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 2001, vol. 47, no. 2, p. 599 to 618. DOI: 10.1109/18.910577

[30] TANNER, R. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 1981, vol. 27, no. 5, p. 533 to 547. DOI: 10.1109/TIT.1981.1056404

[31] SHARON, E., LITSYN, S., GOLDBERGER, J. Efficient serial message-passing schedules for LDPC decoding. *IEEE Transactions on Information Theory*, 2007, vol. 53, no. 11, p. 4076–4091. DOI: 10.1109/TIT.2007.907507

[32] KANG, P., CAI, K., HE, X., et al. Generalized mutual information-maximizing quantized decoding of LDPC codes with layered scheduling. *IEEE Transactions on Vehicular Technology*, 2022, vol. 71, no. 7, p. 7258–7273. DOI: 10.1109/TVT.2022.3162579

[33] HAILES, P., XU, L., MAUNDER, R. G., et al. A survey of FPGA-based LDPC decoders. *IEEE Communications Surveys & Tutorials*, 2015, vol. 18, no. 2, p. 1098–1122. DOI: 10.1109/COMST.2015.2510381

[34] TRAN-THI, B. N., NGUYEN-LY, T. T., HONG, H. N., et al. An improved offset min-sum LDPC decoding algorithm for 5G new radio. In *2021 International Symposium on Electrical and Electronics Engineering (ISEE)*. Ho Chi Minh City (Vietnam), 2021. p. 106–109. DOI: 10.1109/ISEE51682.2021.9418782

[35] 3RD GENERATION PARTNERSHIP PROJECT (3GPP). *NR; Physical Channels and Modulation (3GPP TS 38.211). V15.10.0 (Release 15)*. Sophia Antipolis: ETSI, 2022. [Online] Cited 2025-09-10. Available at: https://www.etsi.org/deliver/etsi_ts/138200_138299/138211/15.10.00_60/ts_138211v151000p.pdf

[36] ADVANCED MICRO DEVICES (AMD). *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906).* Section: “Timing Path Summary — Header Information.” Version 2025.1. AMD, 2025. [Online] Cited 2025-11-10. Available at: https://docs.amd.com/r/en-US/ug906-vivado-design-analysis/Timing-Path-Summary-Header-Information

[37] MEJMAA, B., AKHARRAZ, I., AHAITOUF, A. A field-programmable gate array-based quasi-cyclic low-density parity-check decoder with high throughput and excellent decoding performance for 5G new-radio standards. *Technologies*, 2024, vol. 12, p. 1–16. DOI: 10.3390/technologies12110215

[38] NADAL, J., BAGHDADI, A. Parallel and flexible 5G LDPC decoder architecture targeting FPGA. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2021, vol. 29, no. 6, p. 1141–1151. DOI: 10.1109/TVLSI.2021.3072866

[39] POURJABAR, S., CHOI, G. S. A high-throughput multi-mode LDPC decoder for 5G NR. *International Journal of Circuit Theory and Applications*, 2021, vol. 50, no. 4, p. 1365–1374. DOI: 10.48550/arXiv.2102.13228

[40] LIU, Y., TANG, W., MITCHELL, D. G. Efficient implementation of a threshold modified min-sum algorithm for LDPC decoders. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2020, vol. 67, no. 9, p. 1599–1603. DOI: 10.1109/TCSII.2020.3001601

About the Authors ...

Bich Ngoc TRAN-THI was born in Phu Tho province, Vietnam. She earned her BS and MS in Physics (communication systems) from Southern Federal University (SFU) in Rostov-on-Don, Russia, in 2002 and 2004, respectively. She obtained her Ph.D. from the Faculty of Electrical and Electronics Engineering at Ho Chi Minh City University of Technology in 2024. Currently, she works as a lecturer at the Faculty of Aviation Operations at Vietnam Aviation Academy. Her research interests include antennas, wireless communications, and Low-Density Parity-Check decoder architectures on FPGA platforms.

Trong Hai LE was born in Thanh Hoa Province, Vietnam. He received his BE in Air Traffic Management from Vietnam Aviation Academy in 2023. Currently, he works as a teaching assistant at the Faculty of Aviation Operations at Vietnam Aviation Academy. His research interests include wireless communications, flight management and communication, navigation, and surveillance (CNS) technologies.