symmetry

MDPI

Article

High Area-Efficient Parallel Encoder with Compatible Architecture for 5G LDPC Codes

Yufei Zhu, Zuocheng Xing *, Zerun Li, Yang Zhang and Yifan Hu

School of Computer, National University of Defense Technology, Changsha 410073, China; zhuyufei17@nudt.edu.cn (Y.Z.); lizerun16@nudt.edu.cn (Z.L.); zhangyang@nudt.edu.cn (Y.Z.); huyifan17@nudt.edu.cn (Y.H.)

* Correspondence: zcxing@nudt.edu.cn

Abstract: This paper presents a novel parallel quasi-cyclic low-density parity-check (QC-LDPC) encoding algorithm with low complexity, which is compatible with the 5th generation (5G) new radio (NR). Basing on the algorithm, we propose a high area-efficient parallel encoder with compatible architecture. The proposed encoder has the advantages of parallel encoding and pipelined operations. Furthermore, it is designed as a configurable encoding structure, which is fully compatible with different base graphs of 5G LDPC. Thus, the encoder architecture has flexible adaptability for various 5G LDPC codes. The proposed encoder was synthesized in a 65 nm CMOS technology. According to the encoder architecture, we implemented nine encoders for distributed lifting sizes of two base graphs. The eperimental results show that the encoder has high performance and significant area-efficiency, which is better than related prior art. This work includes a whole set of encoding algorithm and the compatible encoders, which are fully compatible with different base graphs of 5G LDPC codes. Therefore, it has more flexible adaptability for various 5G application scenarios.

Keywords: LDPC codes; encoder; area-efficient; parallel; compatible architecture; 5G

Citation: Zhu, Y.; Xing, Z.; Li, Z.; Zhang, Y.; Hu, Y. High Area-Efficient Parallel Encoder with Compatible Architecture for 5G LDPC Codes. *Symmetry* **2021**, *13*, 700. https://doi.org/10.3390/sym13040700

Academic Editor: Alexander Shelupanov

Received: 5 March 2021
Accepted: 13 April 2021
Published: 16 April 2021

1. Introduction

Low-density parity-check (LDPC) codes have been recognized for their excellent error correction abilities near the Shannon limit [1], and LDPC codes are advantageous in hardware implementation [2]. At present, many communication systems have taken these codes as standards, such as the Digital Video Broadcasting Satellite (DVB-S2/S2X, Europe) [3], the Consultative Committee for Space Data Systems (CCSDS) [4], Wireless Local Area Network (WLAN, IEEE 802.11n) [5], the China digital radio standard [6], and the 5th Generation Mobile Communication Technology (5G) [7,8].

Currently, research on mobile communication systems has entered the 5G phase [9]. Channel encoding is one of the 5G core technologies; it is mainly used to ensure the correct transmission of channel information and to improve communication quality [10]. The Third Generation Partnership Project (3GPP) organization has finally decided to take the LDPC code as the data channel coding scheme for 5G enhanced Mobile Broadband (eMBB) [11,12]. Compared to 4G, 5G has higher requirements in terms of the data transmission rate and information transmission reliability [13,14]. Therefore, it has important significance and application value for exploring a novel LDPC encoding scheme and implementing 5G LDPC codes [15,16]. To achieve scalable data transmission and flexibility, 3GPP has decided to take two kinds of base graphs for 5G channel encoding—*BG*1 and *BG*2 [17].

Presently, some research initiatives have focused on 5G LDPC codes [18,19]. 5G New Radio (NR) has higher performance demands on channel coding solutions [20]; one study discusses the design concept of the new quasi-cyclic low-density parity-check (QC-LDPC) codes that have different structural characteristics and meet the multiple requirements of 5G NR channel coding [21]. Designed as structured LDPC codes, QC-LDPC codes have been research hotspots in the recent past. QC-LDPC codes possess obvious advantages in

circuit implementations, and compared to other kinds of LDPC codes, QC-LDPC codes need fewer hardware resources [22].

Owing to the sparsity of the parity-check matrix, the QC-LDPC encoder can be achieved with a low-complexity design. Despite LDPC codes being defined by a parity check matrix, it is difficult to directly realize a low-complexity LDPC encoder as their generator matrix is usually unknown. Some studies have been conducted to acquire low-complexity LDPC encoding. The direct encoding algorithm was originally proposed by Dr. Gallager [23]. It is a general coding algorithm for linear block codes. The method directly uses C = S × G matrix to obtain the code word C (S represents information bits); its theory is simple, but its coding process is complex. The algorithm utilizes Gaussian elimination to transform the check matrix (H matrix) into a generator matrix (G matrix). Its computation amount and the algorithm complexity are high, and the sparsity of the H matrix is damaged in the process. The algorithm needs to store G matrix information in hardware circuits. The consumption of hardware resources is substantially large, so its hardware implementation is difficult. The LU algorithm [24] uses the H matrix to encode directly; it does not need to convert the H matrix into a G matrix, and the H matrix is split into an information bit matrix H_s and a check bit matrix H_P. However, this method requires LU decomposition on the H_P matrix, that is, H left-multiplies a permutation matrix A, and the determination of A is difficult, so the encoding algorithm is still complex. When the H matrix is a singular matrix, it cannot realize LU decomposition, and the LU algorithm cannot be accomplished. The algorithm is also unsuitable for hardware implementation. The approximate lower triangular matrix encoding algorithm is an effective encoding method named as the RU algorithm [25], which directly applies the H matrix for encoding. The encoding complexity of the RU algorithm is lower than the direct encoding algorithm. The RU algorithm converts the check matrix H into an approximate lower triangular matrix by row and column permutation under the condition of known information bits. The check bits are then obtained by using check equations. Finally, the information bits and check bits are connected in series to form the final codeword. A main disadvantage of the RU coding process is that there is no precise programmable step-by-step encoding algorithm. The multiple matrix computations within the RU algorithm obviously limit the design of a fast flexible encoder.

In 5G communication, in order to meet the needs of diverse communication scenarios, the 3GPP organization has considered the compatibility requirements for the characteristics of different scenarios when formulating the standards for 5G LDPC codes. The 5G LDPC standards contain two different base graphs, BG1 and BG2, which correspond to two different base matrices, H_{BG1} and H_{BG2}. In addition, each base matrix has two sub-matrices B (corresponding to core parity bits), that is, the base matrices of 5G LDPC codes can be further divided into four base matrices.

Although Reference [26] introduces a QC-LDPC encoding structure, its encoding scheme only considers the case of one submatrix B in a single base graph. The paper does not research the significant requirements of encoding compatibility for 5G LDPC codes, which cannot meet the practical application conditions in different scenarios. Reference [6] introduces the encoding approach of CDR LDPC codes. Combining the characteristics of the generation matrix and the check matrix, it designs a hardware-friendly encoding method. In addition, it adopts an optimized control and storage design in implementing the four LDPC codes specified by CDR standard. Reference [22] introduces hardware architectures for encoding QC-LDPC codes, which is based on the features of recursively-constructed QC-LDPC codes. It takes LU decomposition, involved matrices need to be precomputed, compressed and stored in encoding memories. Reference [27] proposes two encoding architectures which can support several code lengths for different applications. The design can realize the requirements for different encoding parameters. Reference [28] proposes a fully parallel LDPC encoder based on reduced complexity XOR trees; it is designed for the IEEE 802.11n standards. Reference [29] introduces a method to improve hardware multiplication based on constant matrices in GF(2); it tries to apply the method

to the QC-LDPC encoding algorithm. Reference [30] describes that the throughput of QC-LDPC codes could be improved by trimming the full-base matrix into the requested matrix size.

For high performance and compatibility of 5G LDPC encoding requirements, this work presents a highly area-efficient parallel QC-LDPC encoder core with compatible architecture, which is compatible with the latest 5G standard. It has high encoding performance and a low hardware cost.

The remaining sections of this paper are organized as follows: Section 2 briefly analyzes the characteristics of 5G LDPC codes. Section 3 proposes a high parallel LDPC encoding algorithm compatible with 5G LDPC codes. Section 4 shows a high area-efficient parallel QC-LDPC encoder with compatible architecture. Section 5 gives experimental results and comparative analysis. Section 6 summarizes our work and provides the conclusions.

2. Analysis of 5G LDPC Codes

LDPC codes are adopted as the data encoding scheme of 5G because its higher encoding throughput and lower latency can better adapt to the data transmission of high-speed services. The main content of 5G LDPC standards is analyzed progressively as follows.

The LDPC codes in 5G standards are QC-LDPC codes. For one QC-LDPC code, the structural characteristics of the check matrix can be denoted by a base graph (BG) or a base graph matrix (H_{BG}), as exampled by the check matrix in Equation (1).

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{1}$$

H_{BG} is a base graph matrix associated with the check matrix H.

$$H_{BG} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \tag{2}$$

In the matrix above, each 1 in H_{BG} represents a 4×4 binary circulant permutation matrix (CPM), and each 0 represents a 4×4 zero matrix; that is, the size Z of the matrix in Equation (1) is 4. The H_{BG} of each QC-LDPC code intuitively indicates the position of the CPM in the check matrix by the distributions of elements '1', providing an important reference in the encoder design of corresponding codes.

H_{BG} can represent the structural characteristics of the check matrix of one QC-LDPC code. However, it cannot reflect the cyclic shift value of each CPM. Therefore, it is essential to define a cyclic shift coefficient matrix P (exponent matrix) to represent the cyclic shift value of the corresponding base graph matrix.

$$P = \begin{bmatrix} P_{1,1} & P_{1,2} & \dots & P_{1,n} \\ P_{2,1} & P_{2,2} & \dots & P_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{m,1} & P_{m,2} & \dots & P_{m,n} \end{bmatrix} \tag{3}$$

There are two values of $P_{m,n}$. When $0 \leq P_{m,n} < Z$, it denotes the cyclic permutation matrix obtained with a $Z \times Z$ submatrix right-shifting by $P_{m,n}$ bits. When $P_{m,n} = -1$, it denotes a $Z \times Z$ zero matrix.

Equation (1) corresponds to the cyclic shift coefficient matrix P, which is shown as follows:

$$P = \begin{bmatrix} 1 & 0 & 2 & -1 \\ -1 & 2 & 1 & 0 \\ 0 & 1 & -1 & 2 \end{bmatrix} \tag{4}$$

Therefore, the check matrix is unique if the lifting size Z and the exponent matrix P of one QC-LDPC code are determined. The description of 5G LDPC codes often adopts this representation method.

In the 5G standard, LDPC codes have two types of base graphs, named BG1 and BG2. Check matrices of BG1 and BG2 both have the characteristic structure shown as Figure 1. These check matrices are termed H matrices.

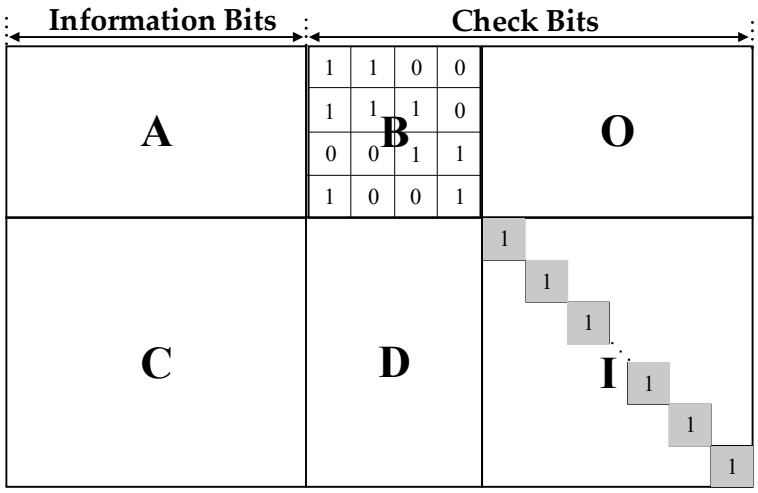

Figure 1. Structure of 5G LDPC check matrix.

The 5G LDPC code has two types of base matrices, namely, H_{BG1} and H_{BG2}. Their information comparison is shown in Table 1.

Table 1. Comparison of H_{BG} matrices.

H_{BG}	Size	M_b	N_b	Element 1
$BG1$	46×68	46	68	316
$BG2$	42×52	42	52	197

BG1 has a total of 316 elements 1 while BG2 has a total of 197 elements 1. The element 1 indicates that the corresponding submatrix is an identity matrix or a cyclic right shift identity matrix. The element 0 indicates that the corresponding submatrix is a zero matrix. Cyclic shift coefficients of submatrices in H are stored in the corresponding coefficient matrix. The coefficient matrix has the same size as the H_{BG} matrix. In the cyclic shift coefficient matrix, the non-negative element i corresponds to the element 1 in the H_{BG}, indicating that the submatrix is the matrix obtained after an identity matrix is cyclically right shifting by i bits. The element -1 corresponds to the element 0 in the H_{BG}, indicating that the corresponding submatrix is a zero matrix.

The size Z of the submatrix in the H is not fixed. The 5G standards specify the values of Z. For BG1 and BG2, the value ranges of Z are the same, as shown in Table 2.

Table 2. Lifting Size Z in 5G LDPC codes.

Z		J							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
a	2	2	4	8	16	32	64	128	256
	3	3	6	12	24	48	96	192	384
	5	5	10	20	40	80	160	320	
	7	7	14	28	56	112	224		
	9	9	18	36	72	144	288		
	11	11	22	44	88	176	352		
	13	13	26	52	104	208			
	15	15	30	60	120	240			

In 5G standards, the cyclic shift coefficients of the submatrices vary in different situations. First, BG1 and BG2 have different coefficient matrices; furthermore, different values of 'a' in Table 2 will result in diverse coefficient matrices even in the same base matrix.

As shown in Table 2, each row of the size Z has the same cyclic shift coefficient matrix, so the size Z can be divided into 8 sets, each of which shares a common coefficient matrix. The Z values after division are shown in Table 3.

Table 3. Sets of Lifting Size Z.

Set 1	$Z = 2 \times 2^j, j = 0,1,2,3,4,5,6,7$
Set 2	$Z = 3 \times 2^j, j = 0,1,2,3,4,5,6,7$
Set 3	$Z = 5 \times 2^j, j = 0,1,2,3,4,5,6$
Set 4	$Z = 7 \times 2^j, j = 0,1,2,3,4,5$
Set 5	$Z = 9 \times 2^j, j = 0,1,2,3,4,5$
Set 6	$Z = 11 \times 2^j, j = 0,1,2,3,4,5$
Set 7	$Z = 13 \times 2^j, j = 0,1,2,3,4$
Set 8	$Z = 15 \times 2^j, j = 0,1,2,3,4$

When a Z value in one set is taken as the size of each submatrix, the coefficient matrix corresponding to the set can be taken to represent the entire H matrix. Equation (5) indicates the final cycle shift coefficients corresponding to different Z values in the same set:

$$P_{ij} = \begin{cases} -1 & if \quad V_{ij} = -1 \\ mod(V_{ij}, Z) & else \end{cases} \tag{5}$$

where V_{ij} denotes the element in the i-th row and the j-th column of the coefficient matrix corresponding to one set. P_{ij} denotes the actual cyclic shift coefficient of the submatrix corresponding to the elements in the i-th row and the j-th column of the H_{BG} for a selected Z in one set.

3. Parallel LDPC Encoding Algorithm Compatible with 5G LDPC Codes

This paper proposes a high parallel QC-LDPC encoding algorithm, which is compatible with 5G LDPC standards. Based on this algorithm, a novel encoder architecture for LDPC codes is designed to satisfy the requirements of 5G LDPC codes mentioned above. There are two base graphs for 5G LDPC codes, BG1 and BG2. These base graphs have different structures, as shown in Figures 2 and 3. Our research is compatible with both BG1 and BG2, this work has wide applicability to the new 5G LDPC standards. Further-

more, we present an integrated solution of the parallel LDPC encoding algorithm and the area-efficient compatible encoder architecture.

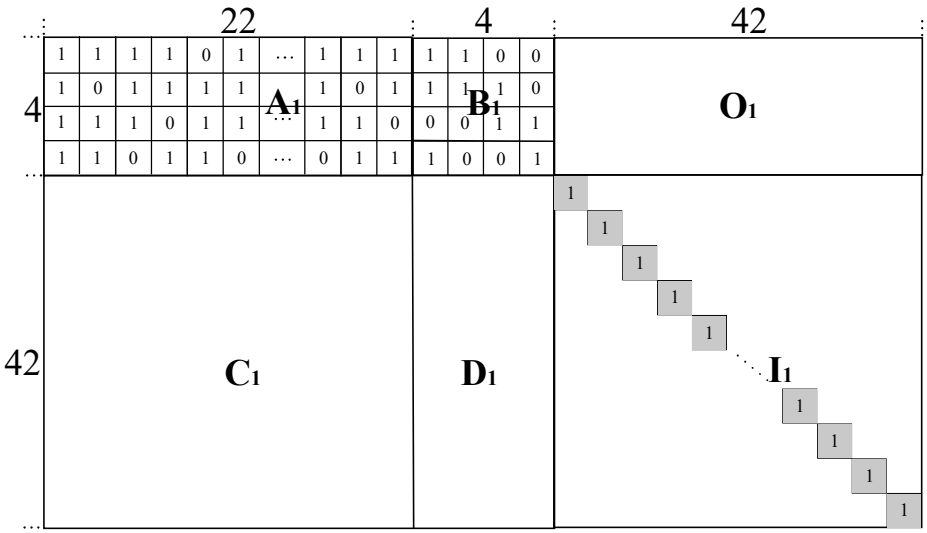

Figure 2. Base matrix structure of *BG*1 (316 Elements 1).

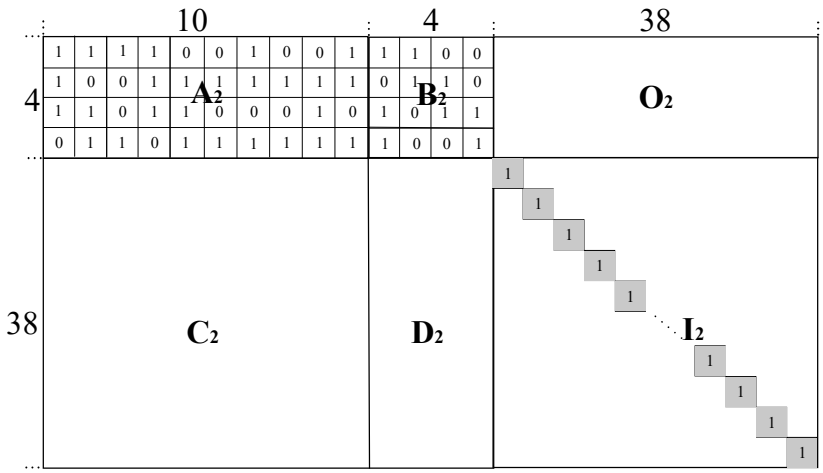

Figure 3. Base matrix structure of *BG*2 (197 Elements 1).

As shown in the two figures, the check matrix H is divided into multiple sub-blocks. BG1 and BG2 have different base graphs. BG1 is mainly used for high-performance encoding scenarios. Taking BG1 as an example, the size of H_{BG1} is 46×68. For the sub-blocks A, B, O, C, D and I, their sizes are 4×22, 4×4, 4×42, 42×22, 42×4, 42×42. Since the lifting size of H_{BG} is Z, the codeword C is uniformly divided by the size Z to match the base matrix H_{BG}. According to the sub-block structure of H_{BG}, C can be denoted as $C = [S_1, \ldots, S_{kb}, P_{a1}, \ldots, P_{a4}, P_{b1}, \ldots, P_{b(Mb-4)}]$. The column number of information sequence S is as same as the column numbers of block A and block C. The column number of the check sequence P_a is as same as those of block B and block D. The column number of the check sequence P_b is as same as those of block O and block I.

The encoding of 5G LDPC codes is defined by $H \times C^T = O^T$, which can be expressed as the following.

$$\begin{bmatrix} A & B & O \\ C & D & I \end{bmatrix} \cdot \begin{bmatrix} S^T \\ P_a^T \\ P_b^T \end{bmatrix} = O \tag{6}$$

(1) First, Equation (6) is decomposed into the following equation set.

$$\begin{cases} A \cdot S^T + B \cdot P_a^T = 0 \\ C \cdot S^T + D \cdot P_a^T + I \cdot P_b^T = 0 \end{cases} \tag{7}$$

(2) Then, P_a and P_b are calculated as follows.

$$\begin{cases} {P_a}^T = B^{-1} \cdot \left(A \cdot S^T\right) \\ P_b^T = \begin{bmatrix} C & D \end{bmatrix} \cdot \begin{bmatrix} S^T \\ {P_a}^T \end{bmatrix} = C \cdot S^T + D \cdot P_a^T \end{cases} \tag{8}$$

Equation (8) shows that there are factors in the $\alpha \cdot \beta^T$ form during the calculation of the check bits P_a and P_b, such as $A \cdot S^T$, $C \cdot S^T$, and $D \cdot P^T$. However, during the calculation of P_a, there is an additional matrix multiplication between B^{-1} and $A \cdot S^T$. Considering the characteristics of the cyclic shift coefficients in the B block, let $Var = A \cdot S^T$, A is the submatrix of H with known coefficients, and S is the bit sequence of input information.

$$\begin{cases} var^T = A \cdot S^T = [var_1, var_2, var_3, var_4]^T \\ P_a = [p_{a1}, p_{a2}, p_{a3}, p_{a4}] \end{cases} \tag{9}$$

For $BG1$ and $BG2$, there are four different cases of circular shift coefficients corresponding to the B submatrices of H_{BG} matrices, which are shown as Figure 4a,b.

$$\begin{bmatrix} 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & -1 \\ -1 & -1 & 0 & 0 \\ 1 & -1 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & -1 & -1 \\ 105 & 0 & 0 & -1 \\ -1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 \end{bmatrix}$$

(**a**)

$$\begin{bmatrix} 0 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 \\ 1 & -1 & -1 & 0 \end{bmatrix}$$

(**b**)

Figure 4. Cyclic shift coefficients of B submatrices for H_{BG1} (**a**) and H_{BG2} (**b**).

In Equation (7), $A \cdot S^T + B \cdot {P_a}^T = 0$, so $[var_1, var_2, var_3, var_4]$ and $[P_{a1}, P_{a2}, P_{a3}, P_{a4}]$ have the following equation relationships.

The computational process of P_a and Var corresponding to the left submatrix of Figure 4a is as follows:

$$\begin{cases} p_{a1}(<<1) + p_{a2} + var_1 = 0 \\ p_{a1} + p_{a2} + p_{a3} + var_2 = 0 \\ p_{a3} + p_{a4} + var_3 = 0 \\ p_{a1}(<<1) + p_{a4} + var_4 = 0 \end{cases} \tag{10}$$

$$\Rightarrow \begin{cases} var_1 + var_2 + var_3 + var_4 = p_{a1} \\ [(var_1 + var_2 + var_3 + var_4) << 1] + var_1 = p_{a2} \\ [(var_1 + var_2 + var_3 + var_4) << 1] + var_3 + var_4 = p_{a3} \\ [(var_1 + var_2 + var_3 + var_4) << 1] + var_4 = p_{a4} \end{cases} \tag{11}$$

The computational process of P_a and Var corresponding to the left submatrix of Figure 4b is as follows:

$$\begin{cases} p_{a1} + p_{a2} + var_1 = 0 \\ p_{a2} + p_{a3} + var_2 = 0 \\ (p_{a1} << 1) + p_{a3} + p_{a4} + var_3 = 0 \\ p_{a1} + p_{a4} + var_4 = 0 \end{cases} \tag{12}$$

$$\Rightarrow \begin{cases} (var_1 + var_2 + var_3 + var_4) >> 1 = p_{a1} \\ var_1 + (var_1 + var_2 + var_3 + var_4) >> 1 = p_{a2} \\ var_1 + var_2 + (var_1 + var_2 + var_3 + var_4) >> 1 = p_{a3} \\ var_4 + (var_1 + var_2 + var_3 + var_4) >> 1 = p_{a4} \end{cases} \tag{13}$$

The computational process of P_a and Var corresponding to the right submatrix of Figure 4a is as follows:

$$\begin{cases} p_{a1} + p_{\mathrm{a}2} + var_1 = 0 \\ (p_{a1} << (105/Z)) + p_{a2} + p_{\mathrm{a}3} + var_2 = 0 \\ p_{a3} + p_{\mathrm{a}4} + var_3 = 0 \\ p_{a1} + p_{\mathrm{a}4} + var_4 = 0 \end{cases} \tag{14}$$

$$\Rightarrow \begin{cases} (var_1 + var_2 + var_3 + var_4) >> (105/Z) = p_{a1} \\ [(var_1 + var_2 + var_3 + var_4) >> (105/Z)] + var_1 = p_{a2} \\ [(var_1 + var_2 + var_3 + var_4) >> (105/Z)] + var_3 + var_4 = p_{a3} \\ [(var_1 + var_2 + var_3 + var_4) >> (105/Z)] + var_4 = p_{a4} \end{cases} \tag{15}$$

The computational process of P_a and Var corresponding to the right submatrix of Figure 4b is as follows:

$$\begin{cases} (p_{a1} << 1) + p_{a2} + var_1 = 0 \\ p_{a2} + p_{a3} + var_2 = 0 \\ p_{a1} + p_{a3} + p_{a4} + var_3 = 0 \\ (p_{a1} << 1) + p_{a4} + var_4 = 0 \end{cases} \tag{16}$$

$$\Rightarrow \begin{cases} var_1 + var_2 + var_3 + var_4 = p_{a1} \\ [(var_1 + var_2 + var_3 + var_4) << 1] + var_1 = p_{a2} \\ [(var_1 + var_2 + var_3 + var_4) << 1] + var_1 + var_2 = p_{a3} \\ [(var_1 + var_2 + var_3 + var_4) << 1] + var_4 = p_{a4} \end{cases} \tag{17}$$

Finally, the check information bits P_a and P_b are obtained, and the encoded codeword C is the output.

$$\Rightarrow \begin{cases} P_a = [p_{a1}, p_{a2}, p_{a3}, p_{a4}] \\ P_b^T = C \cdot S^T + D \cdot P_a^T \\ C = [S, P_a, P_b] \end{cases} \tag{18}$$

For the check matrix H with a special structure in 5G LDPC codes, our scheme elides the matrix inversion operations in encoding. The scheme directly utilizes the linear mathematical relationship between Var and P_a to obtain the check sequence P_a by computing

the intermediate variable (*Var* represents $A \cdot S^T$), which simplifies the encoding process. Through the above equations, the scheme uses *var1*, *var2*, *var3*, and *var4* to solve P_{a1}, P_{a2}, P_{a3}, and P_{a4}. Because P_a has been solved, C and D are both submatrices with known coefficients in H_{BG}, and S is the known information sequence. We can then obtain P_b by ${P_b}^T = C \cdot S^T + D \cdot {P_a}^T$. Finally, we can obtain the encoded codeword $C = [S, P_a, P_b]$ by combining S, P_a, and P_b. With this LDPC encoding scheme, the presented encoder mainly includes $\alpha \cdot \beta^T$ operation units in its hardware implementation, it greatly reduces the hardware complexity of the encoder architecture, laying a foundation for the realization of the high area-efficiency encoder in this paper.

4. Area-Efficient Parallel Pipelined QC-LDPC Encoder with Compatible Architecture

Based on the encoding scheme of this paper, $\alpha \cdot \beta^T$ units are the main operation units of the proposed encoder. $A \cdot S^T$, $C \cdot S^T$, and $D \cdot {P_a}^T$ are all operation units in the form of $\alpha \cdot \beta^T$. According to the quasi-cyclic characteristics of 5G QC-LDPC, one operation unit can process the $\alpha \cdot \beta^T$ operation (Z-bit) in parallel, that is, to complete the computation of a Z-bit sequence in Var^T or ${P_b}^T$. For the H_{BG1}, ${P_a}^T$ and ${P_b}^T$ have up to 46 column sequences, and each sequence is Z-bit, so H_{BG1} needs 46 $\alpha \cdot \beta^T$ operation units. For the H_{BG2}, ${P_a}^T$ and ${P_b}^T$ have up to 42 column sequences, and H_{BG2} then needs 42 $\alpha \cdot \beta^T$ operation units. In order to make the proposed encoder fully support 5G LDPC codes, this encoder sets 46 operation units to compatible with H_{BG1} and H_{BG2}, and the operation units are distributed in *Var* generation module and P_b generation module. Furthermore, the cyclic shift operation and the XOR operation are combined to replace the $\alpha \cdot \beta^T$ computation, avoiding complicated multiplication-accumulation operations required by a direct $\alpha \cdot \beta^T$ operating process. This design not only significantly reduces the complexity of encoder and the hardware costs, but also greatly improves the computing efficiency.

Based on the above analysis, this work has designed a high area-efficient parallel QC-LDPC encoder with compatible architecture, which is shown in Figure 5. The encoder mainly consists of a serial-to-parallel information input buffer, a *Var* generation module, a configurable P_a generation module, a parallel P_b generation module, a cyclic shift coefficient memory module, and an encoding controller.

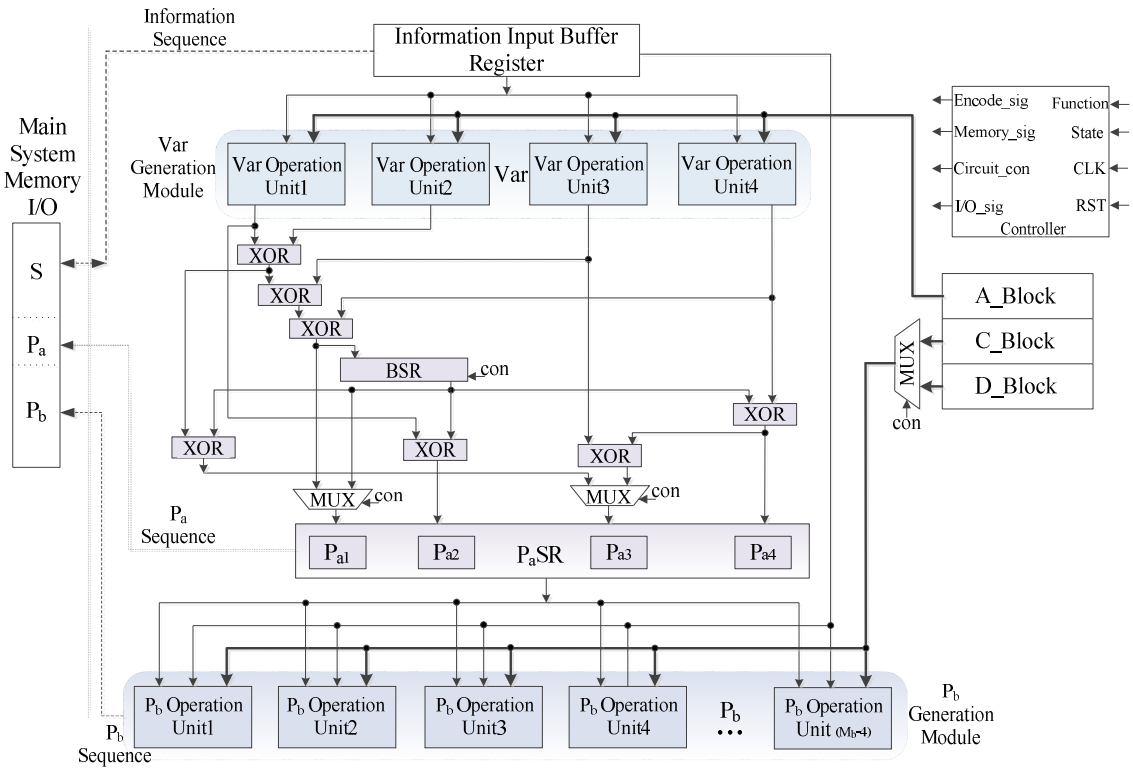

Figure 5. High Area-efficient parallel QC-LDPC encoder with compatible architecture.

The upper and middle encoding modules (*Var* module and P_a module) of the encoder correspond to the high code rate of LDPC encoding used to generate the P_a check bits. The P_b encoding module corresponds to the extended matrix region in H to generate extended check bits. By selecting the number of enabled P_b operation units, the length of P_b check bits can be adjusted to determine the code rate of the encoded codeword. Thus, the encoder architecture can adapt to different code rates of LDPC encoding.

4.1. Information Input Buffer

The information sequence S is input into the buffer, and the buffer sends the Z-bit information sequence to the *Var* operation units in the form of parallel output. The input buffer register is implemented by a register set, which contains two Z-bit registers. Due to the controller signal, when one register supplies the Z-bit S_i sequence to *Var* generation units in parallel, another register can preload the next S_{i+1} sequence. This structure enables the encoder to read in the next information frame to be encoded during the encoding of the present information frame, which saves the information reading time and improves the throughput of the encoder.

4.2. Cyclic Shift Coefficient Memory Module

Based on the structure characteristics of H matrices for 5G LDPC codes, the proposed encoder only needs to store the cyclic shift coefficient values corresponding to the A, C, and D submatrices in the Flash ROM. The memory module for other submatrices can be omitted. The A, C and D submatrices correspond to A_Block, C_Block, and D_Block, respectively. Moreover, the proposed encoder does not need to store the specific content of the H matrix.

The structure of the H_{BG1} is shown in Figure 2. The sizes of the A, C, and D submatrices are 4×22, 42×22, and 42×4, respectively. The cyclic shift coefficients of each row in A or [C, D] are stored in a ROM block, respectively. A total of 46 ROM blocks are required for the encoder. Among them, the coefficients corresponding to the A submatrix are stored in 4 ROMs, and each ROM stores 22 coefficients; the coefficients corresponding to Set [C, D] are stored in 42 ROM blocks, and each block stores 26 coefficient values. The front 22 values denote the coefficients corresponding to the C submatrix while the last 4 values denote the coefficients corresponding to the D submatrix, as shown in Figure 6.

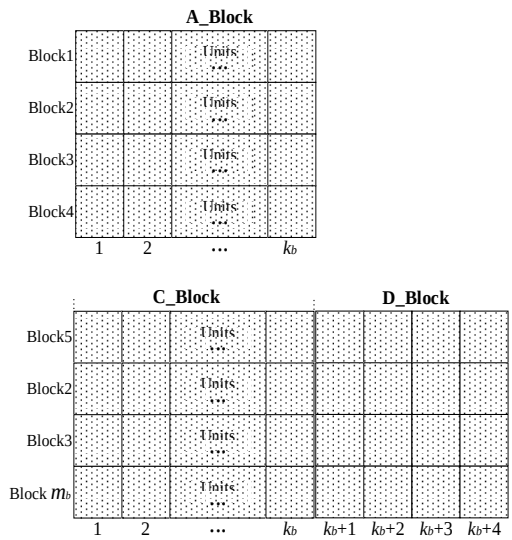

Figure 6. Cyclic Shift Coefficient Memory Module.

The structure of the H_{BG2} is shown as Figure 3. The sizes of the A, C, and D submatrix are 4×10, 38×10, and 38×4, respectively. The coefficients of each row in A or Set [C, D] are respectively stored in a ROM block. A total of 42 ROM blocks are required as

the memory module. Among them, the coefficients corresponding to the A submatrix require 4 ROM blocks for storage, and each block stores 10 coefficients. The coefficients corresponding to Set [C, D] requires 38 ROM blocks for storage, and each block stores 14 coefficient values; the first 10 values denote the coefficients corresponding to the C submatrix while the last 4 values denote the coefficients corresponding to the D submatrix, as shown in Figure 6.

In summary, each coefficient matrix of the H_{BG1} requires 46 ROM blocks for the coefficient memory; each coefficient matrix of the H_{BG2} requires 42 ROM blocks for the coefficient memory. The coefficient memory of the encoder is composed of 46 ROM blocks so that it can be compatible with the two matrices of H_{BG1} and H_{BG1}. The bit width of the coefficients in ROM blocks is determined by the coefficient values of the known encoding algorithm.

4.3. Encoding Operation Unit

As the core operation unit of the encoder, the encoding operation unit is mainly used to realize operations in the $\alpha \cdot \beta^T$ form. The *Var* operation module containing $\alpha \cdot \beta^T$ encoding units is mainly used to execute the $A \cdot S^T$ operation, while the P_b generation module containing $\alpha \cdot \beta^T$ units is mainly used to execute the operation process of $[C\ D] \cdot [S\ P_a]^T$. The circuit structure of the encoding operation unit is shown in Figure 7. The operation unit consists of a Z-bit barrel shift register, a row of XOR gates and a Z-bit state register.

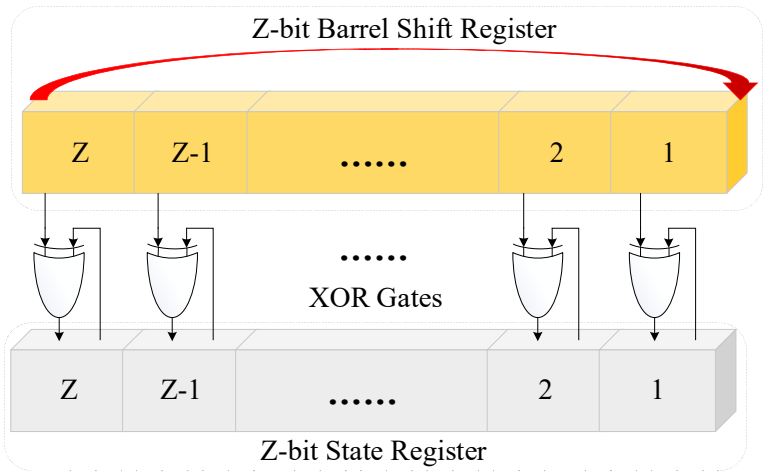

Figure 7. Encoding Operation Unit.

4.4. Var Generation Module

The module integrates four encoding units, which are used to realize the operation of $A \cdot S^T$ in Equation (9). The four encoding units consist of four $\alpha \cdot \beta^T$ operation units (Z-bit granularity), which correspond to *var1*, *var2*, *var3* and *var4* in turn. When the *Var* generation module receives the Z-bit data sequence from the information input buffer, the input buffer first transmits the Z-bit data sequence to the Z-bit barrel shift registers in *var1*, *var2*, *var3*, and *var4* synchronously, by way of corresponding data bits transmission. At the same time, the barrel shift registers read the coefficients in the corresponding position of the A_ROM, and each barrel shift register then shifts the corresponding bits of the data sequence. This means that the $A_{ij} \cdot S_j^T$ operation of a column of Z-bit data is completed, that is, the four operations of $A_{1j} \cdot S_j^T$, $A_{2j} \cdot S_j^T$, $A_{3j} \cdot S_j^T$, and $A_{4j} \cdot S_j^T$ are computed in parallel (i denotes the row of the A submatrix; j denotes the column of the A submatrix). Each result data of the $A_{ij} \cdot S_j^T$ operation takes an XOR operation with the current value of the Z-bit state register (equivalent to a binary addition operation), and one var replaces the value in its state register with the new XOR result, which represents the execution of an $A_{ij} \cdot S_j^T + A_{i(j+1)} \cdot S_{(j+1)}^T$ operation. The four *var1*, *var2*, *var3*, *var4* operation units execute $A_{ij} \cdot S_j^T +$

$A_{(i+1)(j+1)} \cdot S_{(j+1)}{}^T$ operations in parallel, namely, four expressions ($A_{11} \cdot S_1{}^T + A_{12} \cdot S_2{}^T$, $A_{21} \cdot S_1{}^T + A_{22} \cdot S_2{}^T$, $A_{31} \cdot S_1{}^T + A_{32} \cdot S_2{}^T$ and $A_{41} \cdot S_1{}^T + A_{42} \cdot S_2{}^T$) are achieved in parallel, and the initial value in each state register has been set to 0. The *Var* generation module continues to repeat the above operation process, until the completion of four-way $\sum A_{ij} \cdot S_j{}^T$ computation, which would realize the computation processes of the four equations $\sum A_{1j} \cdot S_j{}^T$, $\sum A_{2j} \cdot S_j{}^T$, $\sum A_{3j} \cdot S_j{}^T$, and $\sum A_{4j} \cdot S_j{}^T$.

4.5. Configurable P_a Generation Module

As shown in Figure 5, the *Var* generation module can generate four output results of *var1*, *var2*, *var3*, and *var4* after computing. By inputting the results to the corresponding interfaces of the configurable P_a computation network, it can generate four check information blocks, that is, P_{a1}, P_{a2}, P_{a3}, and P_{a4}.

The new 5G LDPC standards correspond to two sets of base graphs, $BG1$ and $BG2$. As shown in Figure 4, each base matrix (H_{BG}) has two kinds of B submatrices. In other words, the H_{BG} of 5G standards can be further divided into four base matrices. The P_a computation network is innovatively designed as a configurable circuit structure in line with the specific parameters of the four kinds of B submatrices, so that the proposed encoder can be fully compatible with the four base matrices. The four submatrices correspond to four different P_a computation processes. To implement the compatible encoder, the configurable P_a computation network is designed after a detailed analysis of the computational processes and path characteristics. The computation network consists of XOR units, configurable circular shift registers, data multiplexers, and a configurable circuit network. The circuit structure is shown in Figure 2 above. It can be compatible with the computational requirements of the four B submatrices, which means that the computation network can flexibly adapt to $BG1$ and $BG2$. Based on the configurable P_a computation network and the intermediate result *Var*, this encoder can flexibly implement the following four different computation processes to obtain the P_a sequence.

The four computation processes have been listed in the proposed algorithm, which will not be repeated here. The circuit paths of P_a computation network are as follows:

The computing paths of P_a corresponding to the left submatrix of Figure 4a are as follows:

$$\begin{cases} p_{a1} = \sum\limits_{i=1}^{4} var_i \\ p_{a2} = var_1 + \left(\sum\limits_{i=1}^{4} var_i \right) << 1 \\ p_{a2} = var_3 + var_4 + \left(\sum\limits_{i=1}^{4} var_i \right) << 1 \\ p_{a2} = var_4 + \left(\sum\limits_{i=1}^{4} var_i \right) << 1 \end{cases} \tag{19}$$

The computing paths of P_a corresponding to the left submatrix of Figure 4b are as follows:

$$\begin{cases} p_{a1} = \left(\sum\limits_{i=1}^{4} var_i \right) >> 1 \\ p_{a2} = var_1 + \left(\sum\limits_{i=1}^{4} var_i \right) >> 1 \\ p_{a2} = var_1 + var_2 + \left(\sum\limits_{i=1}^{4} var_i \right) >> 1 \\ p_{a2} = var_4 + \left(\sum\limits_{i=1}^{4} var_i \right) >> 1 \end{cases} \tag{20}$$

The computing paths of P_a corresponding to the right submatrix of Figure 4a are as follows:

$$\begin{cases} p_{a1} = \left(\sum_{i=1}^{4} var_i\right) >> (105/Z) \\ p_{a2} = var_1 + \left(\sum_{i=1}^{4} var_i\right) >> (105/Z) \\ p_{a2} = var_3 + var_4 + \left(\sum_{i=1}^{4} var_i\right) >> (105/Z) \\ p_{a2} = var_4 + \left(\sum_{i=1}^{4} var_i\right) >> (105/Z) \end{cases} \tag{21}$$

The computing paths of P_a corresponding to the right submatrix of Figure 4b are as follows:

$$\begin{cases} p_{a1} = \sum_{i=1}^{4} var_i \\ p_{a2} = var_1 + \left(\sum_{i=1}^{4} var_i\right) << 1 \\ p_{a2} = var_1 + var_2 + \left(\sum_{i=1}^{4} var_i\right) << 1 \\ p_{a2} = var_4 + \left(\sum_{i=1}^{4} var_i\right) << 1 \end{cases} \tag{22}$$

P_a sequence register (P_aSR): P_aSR accepts the P_a check blocks from the configurable P_a computation network in parallel, and then registers P_a check blocks into a register set composed of 4 dual-port RAMs. According to the output signal, P_aSR outputs the four check information blocks (P_{a1}, P_{a2}, P_{a3}, and P_{a4}) to the output port of the encoder, which will be stored in the corresponding positions of the encoding memory, belonged to the peripheral main system.

4.6. Parallel P_b Generation Module

The P_b generation module is mainly composed of ($M_b - 4$) encoding operation units, M_b represents the number of rows corresponding to the H_{BG}. The module is used to implement the operations of $P_b{}^T = C{\cdot}S^T + D{\cdot}P_a{}^T$ in Equation (18), including ($M_b - 4$) $\alpha{\cdot}\beta^T$ units with the Z-bit granularity. The structure of the encoding units in the P_b generation module is similar to that of the *Var* generation module. During the generation of P_b check sequences, the computing process of the P_b generation module can be mainly divided into two steps:

(1) The first step is used to complete the computation of $C{\cdot}S^T$, which is executed synchronously with the computation process of the *Var* generation module. The P_b generation module receives the Z-bit data from the information input buffer and transmits to ($M_b - 4$) operation units in parallel. At the same time, due to the control signal, the cyclic shift coefficients are transmitted to the P_b generation module, and these are obtained from the corresponding positions in the C_ROM. The ($M_b - 4$) operation units compute the data sequence in parallel with corresponding coefficients to complete the $C_{ij}{\cdot}S_j{}^T$ process (i denotes the row of the C submatrix, and j denotes the column of the C submatrix). Namely, the units have completed the parallel computation of $C_{1j}{\cdot}S_j{}^T$, $C_{2j}{\cdot}S_j{}^T$, $C_{3j}{\cdot}S_j{}^T$, . . . , $C_{(Mb-4)j}{\cdot}S_j{}^T$. The obtained values of all $C_{ij}{\cdot}S_j{}^T$ will be taken XOR operation with the current values of the state registers in the corresponding operation units, thus realizing the accumulation operation of each $C_{ij}{\cdot}S_j{}^T$ value. The result is as follows:

$$C_{ij}{\cdot}S_j{}^T + \sum_{J=1}^{j-1} C_{iJ}{\cdot}S_J{}^T \tag{23}$$

The $(M_b - 4)$ operation units will execute the operation of Equation (23) in parallel:

$$\begin{gathered} C_{1j}\cdot S_j{}^T + \sum_{J=1}^{j-1} C_{1J}\cdot S_J{}^T \\ C_{2j}\cdot S_j{}^T + \sum_{J=1}^{j-1} C_{2J}\cdot S_J{}^T \\ C_{3j}\cdot S_j{}^T + \sum_{J=1}^{j-1} C_{3J}\cdot S_J{}^T \\ \cdots \\ C_{(M_b\ -\ 4)j}\cdot S_j{}^T + \sum_{J=1}^{j-1} C_{(M_b\ -\ 4)J}\cdot S_J{}^T \end{gathered} \tag{24}$$

The computation process of Equation (24) is executed in parallel. Taking $BG1$ as an example, the size of the submatrix C is 42×22:

$$\begin{cases} C_{11}\cdot S_1{}^T + C_{12}\cdot S_2{}^T + C_{13}\cdot S_3{}^T + \ldots + C_{1,22}\cdot S_{22}{}^T = \sum_{J=1}^{22} C_{1J}\cdot S_J{}^T \\ C_{21}\cdot S_1{}^T + C_{22}\cdot S_2{}^T + C_{23}\cdot S_3{}^T + \ldots + C_{2,22}\cdot S_{22}{}^T = \sum_{J=1}^{22} C_{2J}\cdot S_J{}^T \\ C_{31}\cdot S_1{}^T + C_{32}\cdot S_2{}^T + C_{33}\cdot S_3{}^T + \ldots + C_{3,22}\cdot S_{22}{}^T = \sum_{J=1}^{22} C_{3J}\cdot S_J{}^T \\ \vdots \\ C_{(Mb-4)1}\cdot S_1{}^T + C_{(Mb-4)2}\cdot S_2{}^T + C_{(Mb-4)3}\cdot S_3{}^T + \ldots + C_{(Mb-4),22}\cdot S_{22}{}^T = \sum_{J=1}^{22} C_{(Mb-4)J}\cdot S_J{}^T \end{cases} \tag{25}$$

Lastly, the computational results of the $(M_b - 4)$ operation units are obtained. In this way, the $C\cdot S^T$ operations of a data sequence $S = \{S_1, S_2, S_3, \cdots, S_{22}\}$ is completed in parallel. The length of a data unit S_j is Z bits.

(2) The second step is used to execute the $D\cdot P_a{}^T$ operation and complete the computation of $P_b{}^T = C\cdot S^T + D\cdot P_a{}^T$. The P_b operation units from 1 to $(M_b - 4)$ receive the check bits $P_{a(j)}$ generated by the P_a generation module in parallel. At the same time, the cyclic shift coefficients at the corresponding positions of the D_ROM are sent to the P_b operation units. The cyclic shift coefficients of one column in the D_ROM are read each time (the coefficients' number of one column corresponds to the number of P_b operation units). The coefficients of the column are then sent to each P_b operation unit synchronously. The operation units will accurately execute cyclic-shift operations to the P_a check bits (Z-bit) immediately. Such operations are used to replace the multiplication of $D\cdot P_a{}^T$ to obtain the $D_{1j}\cdot P_{a(j)}, D_{2j}\cdot P_{a(j)}, \cdots, D_{(R\text{-}4)j}\cdot P_{a(j)}$. Then, the $(M_b - 4)$ sequence values will be performed XOR operations in parallel with the current values of the state register in each P_b operation unit. The completion of $P_b{}^{\mathrm{T}} = C\cdot S^T + D\cdot P_a{}^T$ only requires 4 clock cycles after the operations in the first step are finished.

$$\begin{cases} P_{b1}{}^T = D_{11}\cdot P_{a1}{}^T + D_{12}\cdot P_{a2}^T + D_{13}\cdot P_{a3}^T + D_{14}\cdot P_{a4}^T + \sum_{J=1}^{22} C_{1J}\cdot S_J{}^T \\ P_{b2}{}^T = D_{21}\cdot P_{a1}{}^T + D_{22}\cdot P_{a2}^T + D_{23}\cdot P_{a3}{}^T + D_{24}\cdot P_{a4}{}^T + \sum_{J=1}^{22} C_{2J}\cdot S_J{}^T \\ P_{b3}{}^T = D_{31}\cdot P_{a1}{}^T + D_{32}\cdot P_{a2}{}^T + D_{33}\cdot P_{a3}{}^T + D_{34}\cdot P_{a4}{}^T + \sum_{J=1}^{22} C_{3J}\cdot S_J{}^T \\ \vdots \\ P_{(Mb-4)}{}^T = D_{(Mb-4)1}\cdot P_{a1}{}^T + D_{(Mb-4)2}\cdot P_{a2}{}^T + D_{(Mb-4)3}\cdot P_{a3}{}^T + D_{(Mb-4)4}\cdot P_{a4}{}^T + \sum_{J=1}^{22} C_{(Mb-4)J}\cdot S_J{}^T \end{cases} \tag{26}$$

Finally, all check bits of P_b, namely, the check sequences of $P_b = \{P_{b1}, P_{b2}, P_{b3}, \cdots, P_{b(Mb-4)}\}$ are obtained in parallel. The P_b generation module transmits P_b to the output port which is then output to the encoding memory of the peripheral main system.

In the encoding memory, an encoded codeword consists of the information bits (S), the check bits (P_a) and (P_b), and the codewords will be transmitted in the form of $\{S, P_a, P_b\}$ in a batch manner.

4.7. Controller Module

The controller module is responsible for the control function of the encoder. It generates the corresponding signals to control function modules of the encoder to execute the relevant encoding works correctly. Its main signals include encoding control signal, memory control signal, input/output control signal, and circuit configuration signal.

5. Experimental Results and Comparative Analysis

5.1. Comparison of Encoding Methods

The proposed encoder architecture can be fully compatible with $BG1$ and $BG2$. It can adapt to LDPC codes with different lifting sizes Z in these two base graphs. This work implements $BG1$ and $BG2$ with two sets of lifting sizes Z, which fully verifies the performance indicators of the new encoder architecture, as well as the area efficiency of each encoder. The ASIC post synthesis implementation results on 65 nm CMOS technology are shown in the Section 5.2.

Firstly, the proposed encoders have been compared with other LDPC encoding schemes. Table 4 quantitatively compares the proposed encoder to related prior art. We normalised the processes of the ASIC implementations to 65 nm. The comparative results of the ASIC implementations are based on normalized data.

Reference [4] introduces a new structure for the multiply operation of a bit vector with a dense QC matrix, which is the basic operation for LDPC encoding. According to a improved scheduling, the encoding architecture utilizes the parallelism of the LDPC codes by processing multiple bits concurrently. Based on the design, it proposes encoder architecture for CCSDS codes with the applicable encoding methods. It is pending further research on the compatibility of different CCSDS codes, which is expected to improve the flexibility and reduce the cost. Compared with the architecture, this wok has about 2–30 times throughput. Its implenentation occupies 8945 LUTs and 12,420 Flip-Flops of the FPGA (Virtex-7). In terms of its occupied resources, its resource efficiency is also lower than this work.

The paper introduces a encoder scheme of LDPC generator matrix in frequency modulation-China digital radio (CDR) [6]. It utilizes the feature of the parity matrix to parallelize encoding operations for rows and columns. An approach is introduced to control memories; it can be applied to the code with some rates to improve the utilization of circuit resources. Its implenentation occupies 32,479 LUTs, 32,313 Registers, and 36 Block RAM of the FPGA (Spartan-6). Compared to the encoder, this work has obvious promotion in the throughput and the resource efficiency.

Although Reference [26] proposes a QC-LDPC encoder structure for 5G NR, its encoding scheme only considers the case of one submatrix B in a single base graph. The scheme has not researched the obvious requirements of encoding compatibility for practical applications of 5G LDPC. However, for 5G or B5G communication, 3GPP and major corporations have focused on the compatibility for the requirements of diverse applications and integrate the compatibility into the formulation of the 5G LDPC standard to serve the needs of various scenarios. Two kinds of base graphs are involved in 5G LDPC codes; H_{BG1} and H_{BG2} have significant difference, which can be further divided into four base matrices, considering four different submatrices B. This work includes a whole set of the high-compatible algorithm, and the proposed encoder also has wide compatibility, which is fully compatible with different base graphs of 5G LDPC codes. Thus, this work has more flexible adaptability for various 5G application scenarios. Besides, this work has

obvious advantages of higher performance and higher area-efficiency. Compared with the scheme, this wok has about 1.6–2.8 times throughput and 1.4–2.5 times area efficiency, when implementing the proposed encoders in different sizes. These advantages represent lower latency and lower application cost.

Table 4. Comparison information.

Work	[4]	[6]	[26]	[27]	[31]	[32]	[33]	**This Work**
Type	CCSDS	CDR	5G NR	IRA	Gauss	RU	GF(2^2) QC	5G
Implemented Codes	QC-LDPC	LDPC	QC-LDPC (B1 of *BG*1)	LDPC	LDPC	QC-LDPC	QC-LDPC	QC-LDPC (Fully Compatible with *BG*1 and *BG*2)
Process (nm)	28	45	65	28	90	90	28	65
Comparison Platform	FPGA	FPGA	ASIC	ASIC	ASIC	FPGA	ASIC	ASIC
Frequency (MHz)	495	200	600 (Z352)	—	154 (213) *	200	400 (172) *	575 (*BG*1-Z384) 725 (*BG*2-Z64)
Area (Resource) (mm^2)	35 × 35 8945 LUTs + 12,420 FFs	23 × 23 32,479 LUTs + 36 BRAM	0.389	58.5 K Gates+ (512 × 64) SRAM	0.032 (0.017) *	529 60% memory 4% logic	8.66 k Gates (0.007) *	0.511 (Z384) 0.084 (Z64)
Throughput (Gbps)	15.84	0.4	202.4	3.57 (1.54) *	2.31 (3.20) *	0.069	6.3 (2.71) *	362.7 (*BG*1) 121.8 (*BG*2)
Throughput/ Area ($Gbps/mm^2$)	Low	Low	520	Low	188 *	Low	387 *	710 (*BG*1) 1450 (*BG*2)

* Areas are scaled as λ^2, frequencies are scaled as $1/\lambda$, where λ is the process-size ratio for 65 nm.

Reference [27] proposes two kind encoding hardware designs for Irregular Repeat Accumulate (IRA) LDPC codes, which can be used in communication and memory systems. One proposed architecture is a reconfigurable architecture; it is suitable for applications requiring the transition among finite codes frequently. The second proposed architecture utilizes the sparse feature of the parity-check matrices, and reduces the circuit cost by storing its matrix in memory in the sparse form. Compared with the architectures, this wok has more than 9 times throughput and obvious resource efficiency.

Reference [31] takes the Gauss Elimination method to design the check matrix; the encoded codeword is obtained by matrix multiplication based on the generator matrix and the codeword. It proposed a regular LDPC encoder with pipelining structures to obtain a compact encoding process. However, matrix multiplication causes relative complexity in terms of a small block size. The memory overhead will be rapidly raised for larger blocks and for multiple data frames. Compared with the method, this work has about 10–100 times throughput and 4–8 times area-efficiency (Throughput/Area), when implementing the LDPC encoders.

A LDPC encoder is presented for CMMB based on RU algorithm [32]. The RU method takes an modified greedy algorithm for the sparse marix with approximate triangulation. This optimized algorithm can reduce the complexity of encoding. But the implementation of the RU method requires a set of calculations, where data dependence exists in computation steps, limiting the parallelism. Besides, the method has a long critical path, that would cause the encoder implementation unsuitable for high performance scenarios. The LDPC encoder is implemented with Stratix II FPGA, and it consumes 60% the memory resource and 4% the logic resource of the chip. Compared to the design, this work has significant advantages in the throughput and the resource efficiency than its implementation results.

Reference [33] proposes a nonbinary QC-LDPC encoding architecture; it introduces two methods taking advantage of finite Fourier transform to reduce the hardware complexity. In the paper, a GF(2^2) QC-LDPC encoder is implemented, and the relevant parameters

of the result are normalized to 65 nm process. Compared with the scheme, this wok has more than 12.6 times throughput and 1.8–3.7 times area efficiency.

5.2. Implementation Comparison of BG1 and BG2

As the core of one LDPC code, *BG* determines the macro characteristics and encoding performance of the LDPC code. There are two sets of base graphs in the 5G NR standard, namely, *BG*1 and *BG*2. In order to satisfy the needs of different communication scenarios, 5G LDPC codes should be able to flexibly support different encoding parameters. Considering the diversity of future communication scenarios, new encoders should be compatible with *BG*1 and *BG*2.

For the two base graphs, we choose multiple lifting sizes Z, which are uniformly distributed from small to large. For *BG*1, the proposed encoders choose lifting sizes Z as 28, 60, 128, 240, 384, which are respectively recorded as $BG_1(28)$, $BG_1(60)$, $BG_1(128)$, $BG_1(240)$, $BG_1(384)$ in Table 5. For *BG*2, the proposed encoders choose submatrix sizes Z as 64, 120, 224, 352, which are respectively recorded as $BG_2(64)$, $BG_2(120)$, $BG_2(224)$, $BG_2(352)$ in Table 6.

In Tables 5 and 6, Length denotes the word length bits required to match lifting sizes Z. ECC denotes the clock cycles required by encoders to complete the encoding process of a codeword. By default, all data bits of input information need to be encoded. As for the proposed encoder, ECC is equal to the total number of clock cycles required to generate the P_a and P_b check bits corresponding to a codeword.

Table 5. Result comparison of ASIC implementation for lifting size Z = 28, 60, 128, 240, 384.

Proposed Encoder [1]	$BG_1(28)$	$BG_1(60)$	$BG_1(128)$	$BG_1(240)$	$BG_1(384)$
Matrix Set	4	8	1	8	2
Lifting size Z	28	60	128	240	384
Length	5	6	7	8	9
ECC (clock cycles)	28	28	28	28	28
Frequency (MHz)	746	709	658	610	575
Area (mm^2)	0.039	0.086	0.182	0.328	0.511
Equivalent Gates	48.5 K	107.1 K	226.9 K	410.3 K	639.5 K
T-P (Gbps)	34.3	69.9	138.4	240.5	362.7
AE-P (Gbps/mm^2)	879	813	760	733	710
T-S (Gbps)	16.4	33.4	66.2	115.0	173.5
AE-S (Gbps/mm^2)	421	388	364	351	339

[1] ASIC post synthesis implementation results on 65 nm CMOS technology.

Table 6. Result comparison of ASIC implementation for lifting size Z = 64, 120, 224, 352.

Proposed Encoder [1]	$BG_2(64)$	$BG_2(120)$	$BG_2(224)$	$BG_2(352)$
Matrix Set	1	8	4	6
Lifting size Z	64	120	224	352
Length	6	7	8	9
ECC (clock cycles)	16	16	16	16
Frequency (MHz)	725	672	631	586
Area (mm^2)	0.084	0.156	0.282	0.435
Equivalent Gates	104.6 K	195.1 K	352.9 K	545.2 K
T-P (Gbps)	121.8	211.7	371.0	541.5
AE-P (Gbps/mm^2)	1450	1357	1316	1245
T-S (Gbps)	29.0	50.4	88.3	128.9
AE-S (Gbps/mm^2)	345	323	313	296

[1] ASIC post synthesis implementation results on 65 nm CMOS technology.

Based on the parallel pipelined encoding architecture, the proposed encoder needs a total of k_b + 4 clock cycles to generate check sequences (P_a and P_b). In addition, it needs another 2 clock cycles to input the information sequence and output the encoded codeword. Therefore, the proposed encoder needs a total of k_b + 6 clock cycles to complete the encoding of an information sequence. The throughput rate of check bits is an important index, which represents the encoder performance. The throughput rates of different Z sizes are shown in Tables 5 and 6. Since the actual output sequences of the encoder are check sequences, the throughputs of check sequences are recorded as T-P in the tables, and the throughputs of corresponding information sequences are recorded as T-S. Their computation equations for the proposed encoder are as follows:

$$T_P = \frac{Z \times f \times M_b}{ECC} \tag{27}$$

where M_b denotes the number of rows corresponding to the base matrix, Z denotes the expansion size, and f denotes the work frequency of the encoder. ECC denotes clock cycles required to complete the encoding of a codeword, and $ECC = k_b + 6$. For $BG1$, T-P ranges from 34.3 Gbps to 362.7 Gbps for different Z sizes. For $BG2$, T-P ranges from 121.8 Gbps to 541.5 Gbps for different Z sizes. The comparison shows that when the Z sizes are similar, T-$P_{(BG2)}$ is significantly higher than T-$P_{(BG1)}$. This is mainly because in the case of $BG1$, the encoders require $k_b + 6 = 28$ clock cycles to complete the encoding of a codeword. In the case of $BG2$, the encoder only needs $k_b + 6 = 16$ clock cycles to complete the encoding codeword, that is, $ECC_{(BG1)}$ is greater than ECC $_{(BG2)}$. Therefore, for the same Z size, T-$P_{(BG2)}$ is higher than T-$P_{(BG1)}$.

The computational equation of the corresponding data information is as follows:

$$T_S = \frac{Z \times f \times k_b}{ECC} \tag{28}$$

where k_b is equal to the number of columns in the submatrix A, and the submatrix A corresponds to S sequences. Z and f have the same meanings as defined in T-P. For $BG1$, T-S ranges from 16.4 Gbps to 173.5 Gbps for different Z sizes. For $BG2$, T-S ranges from 29.0 Gbps to 128.9 Gbps for different Z sizes. The comparison shows that when the Z sizes are similar, T-$S_{(BG1)}$ is larger than T-$S_{(BG2)}$. This is because $k_{b(BG1)}$ is significantly larger than $k_{b(BG2)}$, resulting in a gap in information encoding performance between them. It is also in line with the reason why 5G LDPC standards formulate two sets of base graphs. $BG1$ is mainly used for scenarios that require high data throughput; $BG2$ is mainly used for scenarios with lower requirements for data throughput.

The two tables show the synthesized areas of the encoder architecture in different Z sizes. It can be found that the encoders' areas gradually become larger as the increase of Z sizes. This is because the codeword length of the encoder increases with the Z sizes synchronously, making the hardware cost of one encoder needs larger memory modules and more logic units. The comparison shows that the complexity of the proposed encoder is positively related to the lifting size Z. In order to fairly compare the area efficiency of the proposed encoder in different Z sizes, Tables 5 and 6 use AE to represent the area efficiency, which is expressed as follows:

$$\text{AE} = \frac{\text{Throughput}}{\text{Area}} \tag{29}$$

In addition, AE-P denotes the area efficiency of the check bits generated by the encoder. AE-S denotes the area efficiency of the information data bits encoded by the encoder. For $BG1$, AE-P ranges from 879 Gbps/mm^2 to 710 Gbps/mm^2. For $BG2$, AE-P ranges from 1450 Gbps/mm^2 to 1245 Gbps/mm^2. It can be known that when the Z sizes are similar, AE-$P_{(BG2)}$ is higher than AE-$P_{(BG1)}$, which is mainly because T-$P_{(BG2)}$ is higher than T-$P_{(BG1)}$. For $BG1$, AE-S ranges from 421 Gbps/mm^2 to 339 Gbps/mm^2. For $BG2$, AE-S ranges from 345 Gbps/mm^2 to 296 Gbps/mm^2. It is clear that AE-$S_{(BG1)}$ is higher than AE-$S_{(BG2)}$, which is due to that T-$S_{(BG1)}$ is higher than T-$S_{(BG2)}$.

By analyzing the experimental data, it can be concluded that the new architecture encoder is compatible with two sets of base graphs, $BG1$ and $BG2$. The implemented encoders can flexibly adapt to submatrix sizes with various granularities. Their performance and area-efficiency are significantly high. The encoder architecture can not only meet the encoding requirements of 5G NR, but also achieve higher encoding performance.

6. Conclusions

This paper presents a parallel LDPC encoding algorithm with high compatibility, which is compatible with 5G LDPC standards, and this work has implemented the high area-efficient parallel encoder with compatible architecture for 5G LDPC codes based on the proposed algorithm. The proposed encoder has the advantages of parallel encoding and pipeline operation, and it takes a configurable encoding structure. Therefore, the encoder architecture has flexible adaptability with 5G LDPC codes. Based on the encoder architecture, we implemented nine encoders for different Z sizes distributed in two base graphs. The experimental results show that the proposed encoder has high performance and significant area-efficiency. It is better than the related prior art. These indicate that the encoding scheme can satisfy the requirements of current 5G LDPC codes, and it can be further applied to future communication scenarios with higher encoding requirements.

Author Contributions: Conceptualization, Y.Z. (Yufei Zhu); Funding acquisition, Z.X.; Investigation, Y.Z. (Yufei Zhu); Methodology, Y.Z. (Yufei Zhu) and Z.X.; Project administration, Z.X.; Resources, Y.Z. (Yang Zhang) and Y.H.; Software, Y.Z. (Yufei Zhu) and Z.L.; Writing—original draft, Y.Z. (Yufei Zhu); Writing—review & editing, Y.Z. (Yang Zhang) and Y.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Natural Science Foundation of China, grant number 61874140, and it is funded by Hunan Science Project Foundation, grant number 2018xk2102.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. MacKay, A.D. Good error-correcting codes based on very sparse matrices. *IEEE Trans. Inf. Theory* **1999**, *45*, 399–431. [CrossRef]
2. Tsatsaragkos, I.; Paliouras, V. A reconfigurable LDPC decoder optimized for 802.11n/ac applications. *IEEE Trans. Very Large Scale Integr. Syst.* **2018**, *26*, 182–195. [CrossRef]
3. Lazarenko, A.V. FPGA design and implementation of DVB-S2/S2X LDPC encoder. In Proceedings of the 2019 IEEE International Conference on Electrical Engineering and Photonics, St. Petersburg, Russia, 17–18 October 2019; pp. 98–102.
4. Theodoropoulos, D.; Kranitis, N.; Tsigkanos, A. Efficient architectures for multigigabit CCSDS LDPC encoders. *IEEE Trans. Very Large Scale Integr. Syst.* **2020**, *28*, 1118–1127. [CrossRef]
5. Patil, P.; Patil, M.; Itraj, S. IEEE 802.11n: Joint modulation-coding and guard interval adaptation scheme for throughput enhancement. *Int. J. Commun. Syst.* **2020**, *33*, 1–14. [CrossRef]
6. Chen, D.; Chen, P.; Fang, Y. Low-complexity high-performance lowdensity parity-check encoder design for China digital radio standard. *IEEE Access* **2017**, *5*, 20880–20886.
7. 5G PPP Architecture Working Group. *View on 5G Architecture*; White Paper, v 1.0; European Union: Luxembourg, 2016.
8. Fuentes, M.; Carcel, J.L.; Dietrich, C. 5G new radio evaluation against IMT-2020 key performance indicators. *IEEE Access* **2020**, *8*, 110880–110896. [CrossRef]
9. Gour, R.; Ishigaki, G.; Kong, J. Availability-guaranteed slice composition for service function chains in 5G transport networks. *J. Opt. Commun. Netw.* **2021**, *13*, 14–24. [CrossRef]
10. Komal, A.; Jaswinder, S.; Yogeshwar, S. A survey on channel coding techniques for 5G wireless networks. *Telecommun. Syst.* **2020**, *73*, 637–663.
11. Multiplexing and Channel Coding. Document TS 38.212 V15.0.0, 3GPP. 2017. Available online: https://www.3gpp.org/ftp/Specs/archive/38_series/38.212/ (accessed on 3 January 2018).
12. Hui, D.; Sandbeg, S.; Blank, Y. Channel coding in 5G new radio: A tutorial overview and performance comparison with 4G LTE. *IEEE Veh. Technol. Mag.* **2018**, *13*, 60–69. [CrossRef]
13. Richardson, T.; Kudekar, S. Design of low-density parity check codes for 5G new radio. *IEEE Commun. Mag.* **2018**, *56*, 28–34. [CrossRef]

14. Huo, Y.; Dong, X.; Xu, W. 5G cellular user equipment: From theory to practical hardware design. *IEEE Access* **2017**, *5*, 13992–14009. [CrossRef]
15. Zhou, W.; Lentmaier, M. Generalized two-magnitude check node updating with self correction for 5G LDPC codes decoding. In Proceedings of the 12th ITG Conference, Rostock, Germany, 11–14 February 2019; pp. 1–6.
16. Li, J.; Lin, S.; Ab, K. *LDPC Code Designs, Constructions, and Unification*; Cambridge University Press: Cambridge, UK, 2017.
17. Multiplexing and Channel Coding (Release 15). Document TS 38.212, V15.4.0, 3GPP. December 2018. Available online: https://www.3gpp.org/ftp/Specs/archive/38_series/38.212/ (accessed on 11 January 2019).
18. Li, H.; Bai, B.; Mu, X. Algebra-assisted construction of quasi-cyclic LDPC codes for 5G new radio. *IEEE Access* **2018**, *6*, 50229–50244. [CrossRef]
19. Zhang, Y.; Peng, K.; Wang, X. Performance analysis and code optimization of IDMA with 5G new radio LDPC code. *IEEE Commun. Lett.* **2018**, *8*, 1552–1555. [CrossRef]
20. NR; Physical Layer Procedures for Data (Release 15). Document TS 38.214, 3GPP. 2018. Available online: https://www.3gpp.org/ftp/Specs/archive/38_series/38.214/ (accessed on 9 April 2018).
21. Bae, J.H.; Abotabl, A.; Lin, H.P. An overview of channel coding for 5G NR cellular communications. *APSIPA Trans. Signal Inf. Process.* **2019**, *8*, 1–14. [CrossRef]
22. Mahdi, A.; Paliouras, V. A low complexity-high throughput QC-LDPC encoder. *IEEE Trans. Signal Process.* **2014**, *62*, 2696–2708. [CrossRef]
23. Heyun, H.E. *Principle and Application of LDPC*; Post & Telecom Press: Beijing, China, 2009; pp. 147–151.
24. Mathur, J.P.S.; Pandey, A. FER performance analysis and optimization of diagonal structure based QC-LDPC codes with girth 12 using LU decomposition. *J. Electr. Eng. Technol.* **2020**, *15*, 1405–1412. [CrossRef]
25. Richardson, T.J.; Urbanke, R.L. Efficient encoding of low-density parity-check codes. *IEEE Trans. Inf. Theory* **2001**, *47*, 638–656. [CrossRef]
26. Tram, T.B.; Tuy, N.T.; Hanho, L. Efficient QC-LDPC encoder for 5G new radio. *Electronics* **2019**, *8*, 1–15.
27. Talati, N.; Wang, Z.; Kvatinsky, S. Rate-compatible and high-throughput architecture designs for encoding LDPC Codes. In Proceedings of the IEEE International Symposyum Circuits Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; pp. 1–4.
28. Mahdi, N.K.; Paliouras, V. A multirate fully parallel LDPC encoder for the IEEE 802.11n/ac/ax QC-LDPC codes based on reduced complexity XOR trees. *IEEE Trans. Very Large Scale Integr. Syst.* **2021**, *29*, 51–64. [CrossRef]
29. Tzimpragos, G.; Kachris, C.; Soudris, D.; Tomkos, I. A low-complexity implementation of QC-LDPC encoder in reconfigurable logic. In Proceedings of the 2013 23rd International Conference on Field Programmable Logic and Applications, IEEE, Porto, Portugal, 2–4 September 2013; pp. 1–4.
30. Wu, H.; Wang, H. A High Throughput Implementation of QC-LDPC codes for 5G NR. *IEEE Access* **2019**, *7*, 185373–185384. [CrossRef]
31. Nandalal, V.; Kumar, V.A. Design and analysis of (5, 10) regular LDPC encoder using MRP technique. *Wirel. Pers. Commun.* **2021**, *1*, 1–17.
32. Zhibin, Z.; Xiangran, S. A new LDPC encoder for CMMB based on RU algorithm. International conference on applied physics and industrial engineering (ICAPIE). *Phys. Proc. China* **2012**, *24*, 864–870.
33. Zhang, X.; Tai, Y. Low-complexity transformed encoder architectures for quasi-cyclic nonbinary LDPC codes over subfields. *IEEE Trans. Very Large Scale Integr. Syst.* **2017**, *25*, 1342–1351. [CrossRef]