electronics

MDPI

Article

Efficient QC-LDPC Encoder for 5G New Radio

Tram Thi Bao Nguyen and Tuy Nguyen Tan and Hanho Lee *

Department of Information and Communication Engineering, Inha University, Incheon 22212, Korea; baotram137@gmail.com (T.T.B.N.); nguyentantuy@gmail.com (T.N.T.)

* Correspondence: hhlee@inha.ac.kr; Tel.: +82-32-860-7449

Received: 22 May 2019; Accepted: 11 June 2019; Published: 13 June 2019

Abstract: This paper presents a novel efficient encoding method and a high-throughput low-complexity encoder architecture for quasi-cyclic low-density parity-check (QC-LDPC) codes for the 5th-generation (5G) New Radio (NR) standard. By storing the quantized value of the permutation information for each submatrix instead of the whole parity check matrix, the required memory storage size is considerably reduced. In addition, sharing techniques are employed to reduce the hardware complexity. The encoding complexity of the proposed method was analyzed, and indicated a substantial reduction in the required area as well as memory storage when compared with existing state-of-the-art encoding approaches. The proposed method requires only 61% gate area, and 11% ROM storage when compared with a similar LDPC encoder using the Richardson–Urbanke method. Synthesis results on TSMC 65-nm complementary metal-oxide semiconductor (CMOS) technology with different submatrix sizes were carried out, which confirmed that the design methodology is flexible and can be adapted for multiple submatrix sizes. For all the considered submatrix sizes, the throughput ranged from 22.1–202.4 Gbps, which sufficiently meets the throughput requirement for the 5G NR standard.

Keywords: quasi-cyclic LDPC code; channel codes; 5G New Radio; encoding

1. Introduction

Low-density parity-check (LDPC) codes [1], which were first proposed by Gallager in the early 1960s and rediscovered by MacKay and Neal [2] in 1996, have attracted widespread attention thanks to their remarkable error correction capabilities near the Shannon limit, with advancements in very large-scale integration (VLSI). Moreover, LDPC codes are among the most widely used types of forward error correction (FEC) codes in several communications standards such as the wireless local area network (WLAN, IEEE 802.11n), wireless radio access network (WRAN, IEEE 802.22), digital video broadcast (DVB), and the Advanced Television System Committee (ATSC). Recently, the fifth generation (5G) communication has been a hotspot of research and development [3]. More specially, LDPC codes play an important role in 5G communication and have been selected as the coding scheme for the 5G enhanced Mobile Broad Band (eMBB) data channel [4]. To support compatible rate and scalable data transmission, 3rd Generation Partnership Project (3GPP) has agreed to consider two rate-compatible base graphs, BG1 and BG2, for the channel coding [5]. Accordingly, several studies have been conducted on the 5G LDPC codes. In [6], a low-cost and flexible demonstration platform is designed and implemented to evaluate the real-time performance of LDPC over the air interface as defined by 5G New Radio (NR) specifications. An algebra-assisted method for constructing 5G LDPC codes is presented in [7].

Over recent years, research on LDPC codes has been focused on structured LDPC codes known as quasi-cyclic low-density parity-check (QC-LDPC) codes [8–12], which exhibit advantages over other types of LDPC codes with respect to the hardware implementations of encoding and decoding using

simple shift registers and logic circuits. A low-complexity encoder can be realized by using QC-LDPC codes, due to the sparseness of the parity check matrix. However, it is not straightforward to encode with low complexity as LDPC codes are defined by their parity check matrix, and the generator matrix is generally unknown. Various approaches have been suggested to improve the hardware complexity of LDPC encoders [13–21]. One of the most conventional approaches is systematic encoding, in which the generator matrix is derived from the parity check matrix by exploiting Gaussian elimination. The main drawback related to this method is that the storage overhead is dramatically increased for large block sizes, which limits its practical applicability. The Richardson–Urbanke (RU) algorithm is a widely-used LDPC codes encoding scheme developed by Richardson and Urbanke [13]. The underlying principle of the method is the transformation of the parity check matrix into an approximate lower triangular (ALT) form by using only row and column permutations, which preserves the sparseness of the matrix. This method suffers from a long critical path, which could make the LDPC encoder unsuitable for high throughput applications. To overcome the limitations of the previous approaches, the design proposed in this paper, which is referred to as a low-complexity high-throughput LDPC encoder architecture for the 5G standard, requires significantly less area and memory storage while maintaining a high throughput.

This paper targets the design of low-complexity high-throughput QC-LDPC encoders for the 5G NR standard. In LDPC encoders, the memory and interconnecting blocks are considered as the major influencing factors of the overall area, delay, and power performance of the hardware design. Hence, the size of the read only memory (ROM) was decreased by storing the quantized value of the permutation information for each submatrix instead of the entire parity check matrix H. The proposed architecture requires less matrix multiplications than the RU method, by exploiting the characteristics of the 5G NR base matrix. In addition, the proposed algorithm does not require the inverse of the component matrix, which presents a primary advantage over the RU method. Moreover, block-memories are not required to store the generator matrix G, and the number of required components is reduced. The ROM size of the proposed method is 98.2% and 88.9% lower than those of the G matrix method and RU method, respectively.

To assess the benefits of the proposed encoding approach, we further implement and synthesize several QC-LDPC encoder architectures with different submatrix sizes Z = 30, 64, 96, 144, and 352. The application specific integrated circuit (ASIC) post synthesis implementation results on TSMC 65-nm complementary metal-oxide semiconductor (CMOS) technology revealed an area efficiency up to 597 Gbps/mm^2 when the proposed encoding method was implemented. Hence, it can be concluded that a promising encoding architecture design for 5G NR LDPC codes was developed in this study.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of the characteristics of 5G NR QC-LDPC codes. In Section 3, two conventional LDPC encoding algorithms from the literature are outlined. A novel 5G NR QC-LDPC encoding approach and a low-complexity high-throughput QC-LDPC encoder architecture are described in Section 4. Section 5 presents the implementation and comparison results, followed by the conclusions in Section 6.

2. 5G NR QC-LDPC Codes

The NR access technology marks a transition in FEC coding for the 3GPP of cellular technologies [22]. In this section, the QC-LDPC codes are reviewed, and the characteristics of standard 5G QC-LDPC codes are summarized. In addition, procedures are presented for the construction of the parity check matrix of the target LDPC codes.

2.1. Preliminary

Let Z be the size of a circulant permutation matrix and $P_{i,j}$ be the shift value. For any integer value $P_{i,j}$, $0 \leq P_{i,j} \leq Z$, a $Z \times Z$ circulant permutation matrix shifts the $Z \times Z$ identity matrix I to the right by $P_{i,j}$ times for the (i, j)-th non-zero element in a base matrix. This binary circulant permutation matrix is denoted as $Q(P_{i,j})$. Considering $Q(1)$ as an example,

$$Q(1) = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix}. \quad (1)$$

For simple notation, $Q(-1)$ denotes the null matrix (all elements equal to zero) of the same size.

2.2. Introduction to QC-LDPC Codes

A binary QC-LDPC code can be characterized by the null space of an array of sparse circulants of the same size [7,23,24]. Taking into account the implementation, the parity-check matrix H of a QC-LDPC code can be defined by its base graph and shift coefficients $(P_{i,j})$. Elements 1s and 0s in the base graph are replaced by a circulant permutation matrix and a zero matrix of size $Z \times Z$, respectively. For two positive integers m_b and n_b, with $m_b \leq n_b$, consider the QC-LDPC code expressed by the following $m_b \times n_b$ array of $Z \times Z$ circulants over GF(2):

$$H = \begin{bmatrix} Q(P_{1,1}) & Q(P_{1,2}) & \cdots & Q(P_{1,n_b}) \\ Q(P_{2,1}) & Q(P_{2,2}) & \cdots & Q(P_{2,n_b}) \\ \vdots & \vdots & \ddots & \vdots \\ Q(P_{m_b,1}) & Q(P_{m_b,2}) & \cdots & Q(P_{m_b,n_b}) \end{bmatrix}. \quad (2)$$

The exponent matrix of H, which is $E(H)$, has the following form:

$$E(H) = \begin{bmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,n_b} \\ P_{2,1} & P_{2,2} & \cdots & P_{2,n_b} \\ \vdots & \vdots & \ddots & \vdots \\ P_{m_b,1} & P_{m_b,2} & \cdots & P_{m_b,n_b} \end{bmatrix}. \quad (3)$$

Each entry in the matrix E is referred to as a shift value. It should be noted that the parity check matrix H in Equation (2) can be constructed by expanding the $m_b \times n_b$ exponent matrix $E(H)$. This procedure is referred to as protograph construction [25].

2.3. 5G NR QC-LDPC Characteristics

As mentioned above, QC-LDPC codes play an important role in 5G communications and have been accepted as the channel coding scheme for the 5G eMBB data channel in 3GPP standard meeting. Figure 1 illustrates the general structure of the NR QC-LDPC base graph. The columns are divided into three parts: information columns, core parity columns, and extension parity columns. The rows are partitioned into two parts: core check rows and extension check rows. As shown in the figure, the base matrix is composed of five submatrices, namely, A, B, O, C, and I [22]. Submatrix A corresponds to systematic bits. In addition, B corresponds to the first set of parity bits and is a square matrix with a dual-diagonal structure: its first column is of weight 3, whereas the submatrix composed of other columns after the first column has an upper dual-diagonal structure. Submatrix O is an all-zero matrix. For the efficient support of incremental redundancy hybrid automatic repeat request (IR-HARQ), a single parity-check (SPC) based extension is used to support lower rates, as shown in Figure 1. Submatrix C corresponds to SPC rows, and I is an identity matrix that corresponds to the second set of parity bits, i.e., the SPC extension. The combination of A and B is referred to as the kernel, and the other parts (O, C, and I) are referred to as extensions. This code structure is similar to the Raptor-like extension, as described in [26].

The 3GPP agreed to consider two rate-compatible base graphs, denoted by BG1 and BG2, for the channel coding. Base graphs BG1 and BG2 have similar structures. However, BG1 is targeted for

larger block lengths $(500 \leq K \leq 8448)$ and higher rates $(1/3 \leq R \leq 8/9)$, whereas BG2 is targeted for smaller block lengths $(40 \leq K \leq 2560)$ and lower rates $(1/5 \leq R \leq 2/3)$. The actual base graph usage and the definition of the two matrices are detailed in the NR standard specification TS 38.212 [27]. The base graph that supports K_{max} should support the following set of shift sizes Z, where $Z = a \times 2^j$ for $a \in \{2, 3, 5, 7, 9, 11, 13, 15\}$ and $0 \leq j \leq 7$.

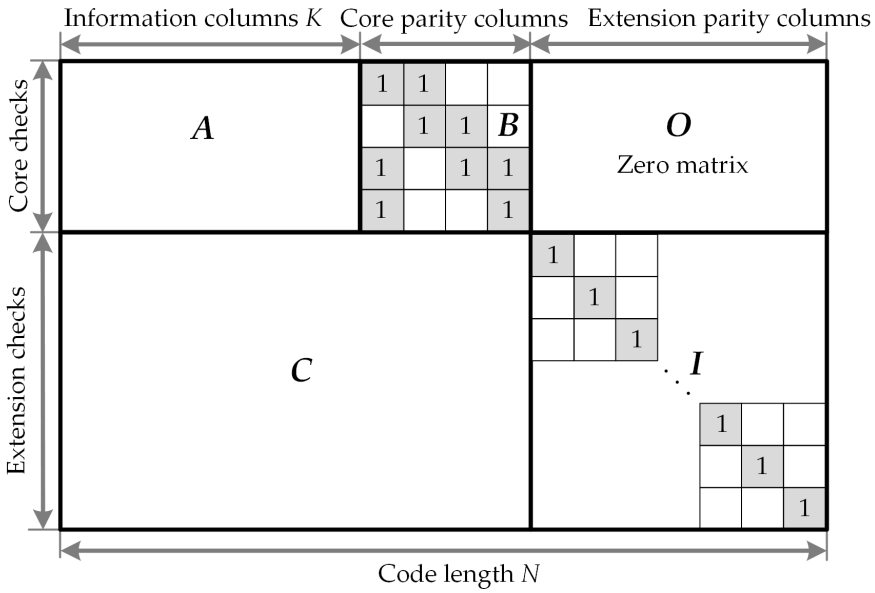

Figure 1. Sketch of base parity check structure for the 5G NR QC-LDPC codes.

For base graphs BG1 and BG2, the number of shift coefficient designs is 8. All lift sizes are divided into eight sets based on parameter a, where a is used for the definition of the lifting-size $a \times 2^j$. The set of shift coefficients are listed in Table 1.

Table 1. Relationship between exponent matrices and sets of lifting size.

Exponent Matrix	Lifting Size Set
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	$Z = 15 \times 2^j, j = 0, 1, 2, 3, 4$

The shift value $P_{i,j}$ can be calculated using the function $P_{i,j} = f(V_{ij}, Z)$, where $V_{i,j}$ is the shift coefficient of the (i, j)-th element in the corresponding shift design. The function f is defined as Equation (4), in which *mod* denotes the modulo arithmetic:

$$P_{i,j} = f(V_{i,j}, z) = \begin{cases} -1, & \text{if } V_{i,j} = -1, \\ mod(V_{i,j}, z), & else. \end{cases} \tag{4}$$

The following procedures are the steps of constructing the parity check matrix of the target (N, K) QC-LDPC code with a given information block size K and code rate $R = K/N$. For a base graph, k_b denotes the number of information circulant columns; thus, if the lifting size is Z, $K = Z \times k_b$ nominally.

Step 1: Obtain the base graph BG1 or BG2 and determine the value of k_b for the given K and R.

– For BG1: $k_b = 22$.

– For BG2: $k_b = 10$ if $K > 640$; $k_b = 9$ if $560 < K \leq 640$; $k_b = 8$ if $192 < K \leq 560$; and $k_b = 6$ elsewhere.

Step 2: Determine Z by selecting the minimum Z value in Table 2, such that $k_b \times Z \geq K$.

Step 3: After the lifting size Z is determined, the corresponding shift coefficient matrix is then selected from Table 1 {Set 1, Set 2,..., Set 8} according to set Z.

Step 4: Calculate the shifting coefficient value $P_{i,j}$ by the modular Z operation, as discussed in Equation (4).

Step 5: Replace each entry in the final exponent matrix with the corresponding circulant permutation matrix or zero matrix of size $Z \times Z$. The QC-LDPC code construction is completed and a parity check matrix H of size $m_b Z \times n_b Z$ is obtained. In 5G QC-LDPC codes, shortening and puncturing is carried out to obtain the desired information lengths and rate adaption. Figure 2 presents an illustration of the encoding process of these codes

Table 2. Lifting size Z supported by standard 5G QC-LDPC codes.

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

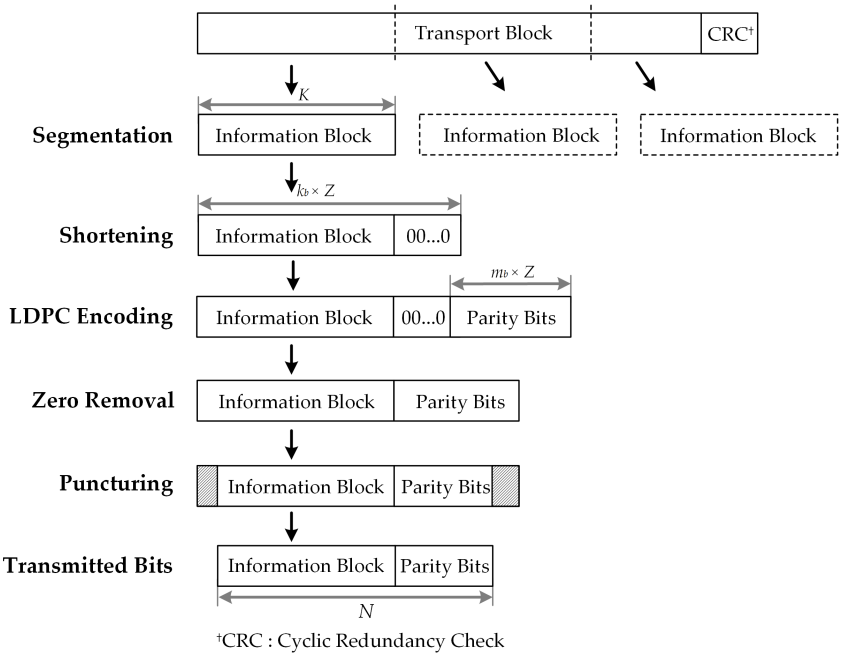

Figure 2. Shortening by zero padding and puncturing of standard 5G QC-LDPC codes.

3. LDPC Encoding Algorithms

Given a parity check matrix H, the objective of LDPC encoding is to solve parity equations:

$$HC^T = 0^T, \tag{5}$$

where C is the systematic codeword, which consists of the information bit vector S and parity code vector P.

This section presents a review on two generic encoding methodologies for the implementation of the LDPC encoder: the Gaussian elimination method and the RU method.

3.1. LDPC Encoding with Gaussian Elinination

The Gaussian elimination is the most conventional method of encoding LDPC codes, which is carried out by the multiplication of the generator matrix G, and contains a complexity quadratic in the block length [19]. The unknown generator matrix G can be derived from the parity check matrix H. A generator matrix for code with a parity check matrix H can be obtained by carrying out Gauss–Jordan elimination on H in the following form:

$$H = \begin{bmatrix} A & I_{N-K} \end{bmatrix}, \tag{6}$$

where A is an $(N-K) \times K$ binary matrix and I_{N-K} is the identity matrix of order $(N-K)$. The generator matrix is as follows:

$$G = \begin{bmatrix} I_K & A^T \end{bmatrix}. \tag{7}$$

The codeword C is then obtained by multiplying the generator matrix G by the systematic bits S as follows:

$$C = SG. \tag{8}$$

The sequential LDPC encoder based on the multiplication of the G matrix requires a ROM to store the generator matrix used to compute the codeword C. The main drawback of this approach is that, unlike parity check matrix H, the corresponding generator matrix G will most likely not be sparse. The complexity of this straightforward encoding algorithm is $O(N^2)$, where N is the number of bits in a codeword. Therefore, the implementation of the matrix multiplication at the encoder results in a very high complexity. For an arbitrary parity check matrix, the construction of G should be avoided and encoding should be carried out using back substitution with H.

3.2. LDPC Encoding with the RU Method

Instead of determining a generator matrix for H, an LDPC code can be directly encoded using the parity check matrix by transforming it into a lower triangular form and applying back substitution. The RU encoding method, which was proposed by Richardson and Urbanke [13], is a linear time encoding method for sparse parity check matrices. The underlying principle is transformation using only row and column permutations, to reformulate a parity check matrix H into a sparse matrix. Therefore, this approach can reduce the complexity more than the G matrix multiplication method. The RU algorithm consists of two steps: a pre-processing step and actual encoding step.

First, in the pre-processing step, the parity check matrix H is converted into the approximate lower triangular (ALT) form, as shown in Figure 3. The parity check matrix H is given by the $M \times N$ matrix, where N is the block length of the code and M is the number of parity check equations. Given that the matrix transformation is realized solely by row and column permutations, the H matrix remains a sparse matrix:

$$H_T = \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix}. \tag{9}$$

Here, the matrix T has a lower triangular form with 1s along the diagonal, and all the entries above the diagonal are 0s. By multiplying H from the left by

$$\begin{bmatrix} I & 0 \\ -ET^{-1} & I \end{bmatrix}, \tag{10}$$

the following is obtained:

$$\tilde{H} = \begin{bmatrix} A & B & T \\ \tilde{C} & \tilde{D} & 0 \end{bmatrix}, \tag{11}$$

where

$$\begin{aligned} \tilde{C} &= -ET^{-1}A + C, \\ \tilde{D} &= -ET^{-1}B + D, \\ \tilde{E} &= -ET^{-1}T + E = 0. \end{aligned} \tag{12}$$

The actual encoding step is performed by matrix-multiplication, forward-substitution and vector addition operations. Let the codeword $C = [s \quad p_1 \quad p_2]$ where s represents the information bits, p_1 denotes the first G parity check bits, and p_2 contains the remaining $(M - G)$ parity check bits. The codeword C must satisfy the parity check equation $HC^T = 0^T$. The two equations are then expressed by:

$$\begin{aligned} As^T + Bp_1^T + Tp_2^T &= 0^T, \\ \tilde{C}s^T + \tilde{D}p_1^T + 0p_2^T &= 0^T. \end{aligned} \tag{13}$$

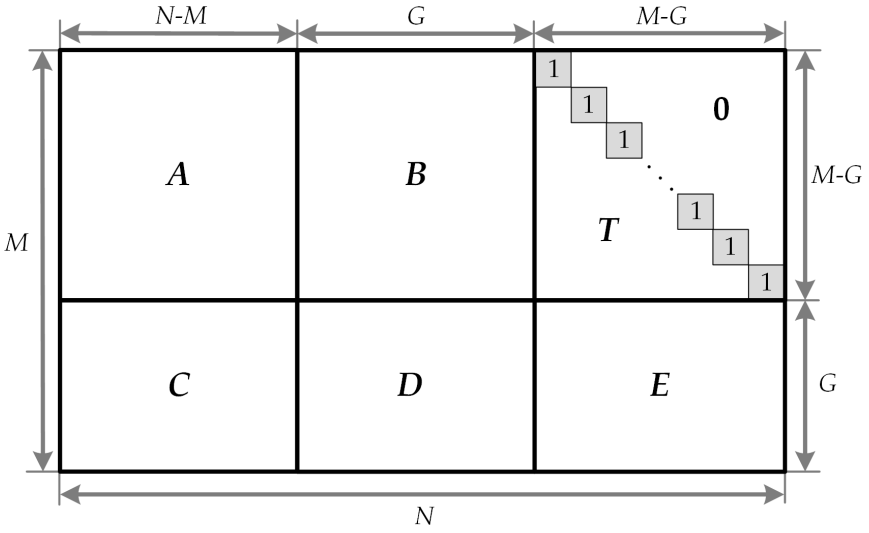

Figure 3. The parity check matrix H in approximate lower triangular form.

Using the RU method, the calculation of the parity bits in the first parity portion p_1 is only dependent on the information bits, given that E was cleared. Hence, it can be calculated independently of the parity bits in p_2. If $\tilde{D}$ is non singular, then p_1^T can be obtained from Equation (13):

$$p_1^T = \tilde{D}^{-1}\tilde{C}s^T. \tag{14}$$

If $\tilde{D}$ is singular in GF(2), then it is necessary to further permute the columns of $\tilde{H}$ to eliminate this singularity. Once p_1 is known, p_2 can be determined using Equation (13):

$$p_2^T = -T^{-1}(As^T + Bp_1^T). \tag{15}$$

Given that T is the lower triangular form, p_2 can be found using back substitution. The complexity of this encoding procedure can be kept low since A, B and T are sparse. Tables 3 and 4 present the complexity of calculation of p_1^T and p_2^T, respectively. The complexity of the RU algorithm is given by $O(N + G^2)$, where N is the block length and G is the gap to linear encoding. The gap is actually the

number of rows of the parity check matrix that cannot be set into a triangular form using only row and column permutations. With a small gap G, the lower encoding complexity for the code is achieved.

Table 3. Complexity analysis of p_1^T calculation.

Operation	Comment	Complexity
As^T	Multiplication by sparse matrix	$O(N)$
$T^{-1}As^T$	Back substitution, T is lower triangular matrix	$O(N)$
$-ET^{-1}[As^T]$	Multiplication by sparse matrix	$O(N)$
Cs^T	Multiplication by sparse matrix	$O(N)$
$\tilde{C} = -ET^{-1}[As^T] + Cs^T$	Addition	$O(N)$
$-\tilde{D}^{-1}\tilde{C}s^T$	Multiplication by $G \times G$ matrix	$O(G^2)$

Table 4. Complexity analysis of p_2^T calculation.

Operation	Comment	Complexity
As^T	Multiplication by sparse matrix	$O(N)$
Bp_1^T	Multiplication by sparse matrix	$O(N)$
$[As^T] + [Bp_1^T]$	Addition	$O(N)$
$-T^{-1}(As^T + Bp_1^T)$	Back substitution, T is lower triangular	$O(N)$

The disadvantage of encoding using the RU method is that there is no exact programmable step-by-step algorithm. The multiple matrix calculations in this algorithm significantly limit the development of a rapid flexible encoder [28]. In addition, the RU method is subjected to a long critical path and odd constraints, which could render the LDPC encoder non-systematic [19].

4. Proposed 5G NR QC-LDPC Encoder Design

4.1. Proposed QC-LDPC Encoding Algorithm

This section presents an efficient scheme developed in this study for the construction of efficient encoders for 5G NR QC-LDPC codes. The proposed encoding method is based on the special characteristics of 5G NR QC-LDPC codes, which are presented in Figure 1. The proposed architectures target low-complexity, while ensuring high-throughput. As reported in the literature review, base graphs BG1 and BG2 have similar structures. In this paper, we focus our description on BG1 with a size of $m_b \times n_b$ (m_b = 46, and n_b = 68), which is the main 5G NR high rate base graph.

Let the codeword $C = [s \quad p_a \quad p_c]$, where s denotes the systematic portion, which is divided into 22 groups of Z bits, since the base graph BG1 has $k_b = n_b - m_b = 22$ information bit columns. Moreover, $s = [s_1, s_2, \ldots, s_{k_b}]$, where each element of s is a vector of length Z. The information messages received by the encoder are stored in registers that are organized by k_b blocks, denoted by s_i $(i = 1, 2, \ldots, k_b)$, which correspond to the systematic blocks, where each consists of Z bits. Given that the encoder was designed to read Z bits per clock cycle, it requires k_b cycles to store all the information blocks. Moreover, the parity sequence can be grouped into sets of Z bits. Suppose that the parity portion of each message p is split into two sub-components as follows: the first $g = 4$ parity bits $p_a = [p_{a_1}, p_{a_2}, \ldots, p_{a_g}]$, and the remaining $(m_b - g) = 42$ parity bits $p_c = [p_{c_1}, p_{c_2}, \ldots, p_{c_{m_b-g}}]$. More precisely, the encoded codeword can be expressed as:

$$C = \left[s_1, s_2, \ldots, s_{k_b}, p_{a_1}, p_{a_2}, \ldots, p_{a_g}, p_{c_1}, p_{c_2}, \ldots, p_{c_{m_b-g}}.\right] \tag{16}$$

The parity check matrix H of 5G NR QC-LDPC codes can be partitioned into six matrices and presented in the following form:

$$H = \begin{bmatrix} A & B & 0 \\ C_1 & C_2 & I \end{bmatrix}, \tag{17}$$

where A is $g \times k_b$, B is $g \times g$, C_1 is $(m_b - g) \times k_b$, and C_2 is $(m_b - g) \times g$. Moreover, I is an identity matrix with dimensions of $(m_b - g) \times (m_b - g)$. The encoding of LDPC codes is carried out using the following defining equation:

$$HC^T = 0^T. \tag{18}$$

Equation (18) can also be expressed as:

$$\begin{bmatrix} A & B & 0 \\ C_1 & C_2 & I \end{bmatrix} \begin{bmatrix} s \\ p_a \\ p_c \end{bmatrix} = 0^T. \tag{19}$$

Equation (19) is then naturally split into two equations, as follows:

$$A_s^T + Bp_a^T + 0p_c^T = 0^T, \tag{20}$$

$$C_1 s^T + C_2 p_a^T I p_c^T = 0^T. \tag{21}$$

The proposed algorithm is performed in two steps. In the initial step, the parity bits in the first portion p_a are computed by solving Equation (20). The second step in the encoding process includes the computation of the p_c parity portions using Equation (21).

The first step in the encoder implementation is the determination of the p_a part. Initially, Equation (20) is re-written in block form as follows:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,k_b} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,k_b} \\ a_{3,1} & a_{3,2} & \ddots & a_{3,k_b} \\ a_{4,1} & a_{4,2} & \cdots & a_{4,k_b} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{k_b} \end{bmatrix} + \begin{bmatrix} 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & -1 \\ -1 & -1 & 0 & 0 \\ 1 & -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} p_{a_1} \\ p_{a_2} \\ p_{a_3} \\ p_{a_4} \end{bmatrix} = 0. \tag{22}$$

This can then be expanded into the following set of equations:

$$\sum_{j=1}^{k_b} a_{1,j} s_j + p_{a_1}^{(1)} + p_{a_2} = 0, \tag{23}$$

$$\sum_{j=1}^{k_b} a_{2,j} s_j + p_{a_1} + p_{a_2} + p_{a_3} = 0, \tag{24}$$

$$\sum_{j=1}^{k_b} a_{3,j} s_j + p_{a_3} + p_{a_4} = 0, \tag{25}$$

$$\sum_{j=1}^{k_b} a_{4,j} s_j + p_{a_1}^{(1)} + p_{a_4} = 0, \tag{26}$$

where $p_{a_1}^{(\alpha)}$ denotes the α^{th} (right) cyclic shifted version of p_{a_1} for $0 \leq \alpha \leq Z$. By adding up all the above equations, the following is obtained:

$$p_{a_1} = \sum_{i=1}^{4} \sum_{j=1}^{k_b} a_{i,j} s_j. \tag{27}$$

It should be noted that a straightforward implementation of $a_{i,j}s_j$ can be done with the use of Z-bit cyclic shifters. Since $a_{i,j}s_j$ is a circular right shift of s_j with the shift coefficient defined by $a_{i,j}$, the hardware complexity is trivial. Based on the definition below,

$$\lambda_i = \sum_{j=0}^{k_b} a_{i,j} s_j \text{ for } i = 1, 2, 3, 4, \tag{28}$$

the following can be obtained:

$$p_{a_1} = \sum_{i=1}^{4} \lambda_i, \tag{29}$$

$$p_{a_2} = \lambda_1 + p_{a_1}^{(1)}, \tag{30}$$

$$p_{a_3} = \lambda_3 + p_{a_4}, \tag{31}$$

$$p_{a_4} = \lambda_4 + p_{a_1}^{(1)}. \tag{32}$$

From Equation (28), each λ_i value is computed by accumulating all the $a_{i,j}s_j$ values. In Modulo 2, λ_i is obtained by carrying out XOR operations on all the elements of $a_{i,j}s_j$. The λ_i values can be estimated per clock cycle in $g = 4$ cycles. The first block of the parity bits p_{a_1} is then calculated by accumulating all the λ_i values. The remaining parity bits p_{a_i} can be obtained using a method that can be easily derived from Equations (30)–(32). This process can be done in two clock cycles since there is dependency between p_{a_3} and p_{a_4}. All the parity bits p_a in the first parity portion are stored in registers.

In a second step, the p_c portion can be easily determined based on Equation (21), where matrices C_1 and C_2 are given by

$$C_1 = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,k_b} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,k_b} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m_b-g,1} & c_{m_b-g,2} & \cdots & c_{m_b-g,k_b} \end{bmatrix}; \quad C_2 = \begin{bmatrix} c_{1,k_b+1} & c_{1,k_b+2} & \cdots & c_{1,k_b+g} \\ c_{2,k_b+1} & c_{2,k_b+2} & \cdots & c_{2,k_b+g} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m_b-g,k_b+1} & c_{m_b-g,k_b+2} & \cdots & c_{m_b-g,k_b+g}. \end{bmatrix} \tag{33}$$

Upon the application of Equation (21), the elements of p_c can be computed using the following equations:

$$\begin{aligned} p_{c_1} &= \sum_{j=1}^{k_b} c_{1,j} s_j + \sum_{j=1}^{g} c_{1,k_b+j} p_{a_j}, \\ p_{c_2} &= \sum_{j=1}^{k_b} c_{2,j} s_j + \sum_{j=1}^{g} c_{2,k_b+j} p_{a_j}, \\ &\vdots \\ p_{c_{m_b-g}} &= \sum_{j=1}^{k_b} c_{m_b-g,j} s_j + \sum_{j=1}^{g} c_{m_b-g,k_b+j} p_{a_j}. \end{aligned} \tag{34}$$

Similarly, $c_{i,j}s_j$ represents a circular shift of s_j with the shift coefficient defined by $c_{i,j}$, and $c_{i,k_b+j}p_{a_j}$ represents a circular shift of p_{a_j} with the shift coefficient defined by c_{i,k_b+j}. As soon as $c_{i,j}s_j$ and $c_{i,k_b+j}p_{a_j}$ have been obtained, they can be used to determine the value of the corresponding parity bits in the second parity portion p_c. This step can be performed in a single clock cycle. Hence, all the p_c parity bits can be acquired in $(m_b - g)$ clock cycles. The encoded codeword is then a combination of the original message s and the two calculated parity portions p_a and p_c.

4.2. Proposed QC-LDPC Encoder Architecture

Figure 4 details the overall block diagram for the proposed low complexity 5G NR QC-LDPC code encoder. The hardware architectures were designed to conduct the encoding process through steps defined in Equations (29)–(32) and (34). In the first step, the computation of the parity bits in the first portion p_a is carried out. From Equation (28), each λ_i value is computed by accumulating all the cyclic shift results of s_j. Since the information message s consists of k_b blocks of Z bits, a total of $k_b = 22$ barrel shifters of size Z, which are denoted by CS_j, $j = 1, 2, ..., k_b$, are required for the

circular shift process. The vector addition of all the λ_i components is then carried out by the XOR trees. Each intermediate λ_i value corresponding to Equation (28) can be estimated per clock cycle and stored in the λ_memory to be used later. Thus, the value of p_{a_1} can be obtained in $g = 4$ clock cycles when all λ_i values are obtained and stored in memory. The remaining parity bits of p_a can be obtained in 2 clock cycles with the use of XOR gates.The objective of the second step is the calculation of the parity bits in the second portion p_c. According to (34), the parity blocks p_{c_i} can be achieved by the vector addition of $c_{i,j}s_j$ and $c_{i,k_b+j}p_{a_j}$. The value of $c_{i,j}s_j$ is also computed by accumulating all the cyclic shift results of s_j. In this step, the overall hardware complexity can be further decreased by exploiting the sharing technique. More specifically, the barrel shifters and XOR trees are reused for the computation of p_c in this step. Control signals are generated by the controller block. The value of $c_{i,k_b+j}p_{a_j}$ is estimated by accumulating all the cyclic shift results of p_{a_j}. The required number of Z–bit barrel shifters is $g = 4$. The main blocks of the proposed architecture can be described as follows.

(1) Input/ Output Buffer: the input buffer, which is implemented as a number of serial input parallel output shift registers, is exploited to store the input systematic bits s_i received by the encoder. The output buffer is used to store the encoded codeword.

(2) Memory Blocks: two memory blocks are utilized, namely, one for the submatrix permutation values, and the others for the accumulated values λ that correspond to matrix A. In Figure 4, the AROM, C_1ROM,and C_2ROM correspond to the ROMs that store the coefficients of matrix A, matrix C_1, and matrix C_2, respectively. Under the assumption that $\tilde{q} = \lceil log_2 Z \rceil$ bits represent the required word length to store the permutation information for each submatrix: $\tilde{q}gk_b$, $\tilde{q}(m_b - g)k_b$, and $\tilde{q}(m_b - g)g$ bits are required to store matrix A, C_1, and C_2, respectively. A significant portion of the hardware complexity of the LDPC encoder consists of the memory required to store the parity check matrix. Unlike the RU method, the proposed algorithm does not require the inverse of the component matrix, which reflects its primary advantage over the RU method. Compared with the Gaussian method, the proposed architecture does not require for block-memories to store the generator matrix G, which further decreases the number of required components. The λ_memory is implemented as a dual port random access memory (RAM) for storing λ_i messages $(i = 1, 2, \ldots, g)$. Each memory word λ_i consists of Z bits, corresponding to one accumulated message of matrix A. Moreover, a total of $(g \times Z)$ bits of λ_memory are required for the proposed encoder.

(3) Barrel Shifters: barrel shifters are used to implement the cyclic shift permutations, according to the shift values provided by the cyclic shifter controllers. It should be noted that the number of cyclic shifters is equal to the number of message blocks, and the size of the barrel shifters is equal to submatrix size Z.

(4) XOR Trees: in Modulo 2, the addition implementation is obtained by carrying out an XOR operation on all the elements.

(5) Controller: this block generates control signals, such as *data_sel* to indicate the step being processed; and *mem_en*, to enable write access to the λ_memory.

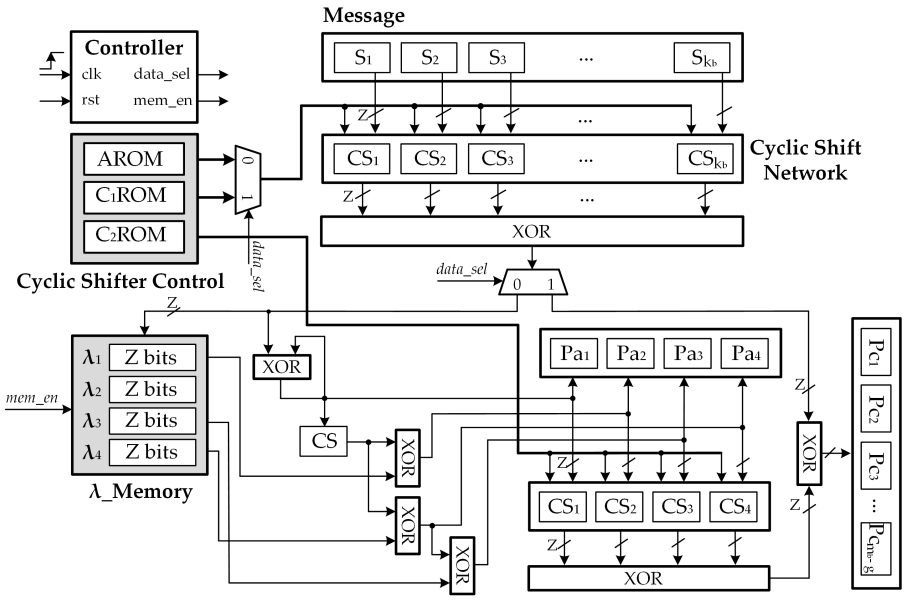

Figure 4. Low-complexity high-throughput encoder architecture for 5G NR QC-LDPC code.

5. Performance Analysis and Comparison

This section reports the implementation results of the proposed LDPC encoder architecture as well as a detailed comparison between the proposed method and other encoder implementations in terms of area and speed for 5G NR standard. First, the design characteristics of different LDPC code encoders are discussed. Thereafter, an analysis of the proposed LDPC encoder, with respect to its implementation on ASIC, is presented.

Table 5 presents a comparison between the area and speed of the proposed encoding method and those of other state-of-the-art approaches. As shown in Table 5, the matrix size was utilized to determine the ROM storage, and the Hamming weights of the matrices were used in computing the gate count. Since all the systematic bits and parity check bits in the first parity portion are stored in registers, the number of flip-flops required was estimated by the bit sizes of K and p_a. In Table 5, the time interval between input frames was exploited in order to compare the processing speed of different encoding methods. The time between two consecutive input frames is based on the total number of clock cycles between the arrivals of the first Z bits of a frame up to the cycle wherein the encoder is ready to receive another frame.

To make it clear, a target LDPC code with a base graph BG1 and submatrix size $Z = 16$ was considered in Table 6. As can be observed from Table 6, the proposed encoder gains a significant reduction in the storage overhead. In the Gaussian elimination method, the entire generator matrix G is stored in the memory. In the RU method, the location of the edges (ones) of each row is stored, with an extra bit indicating the end of a row. By only storing the values of shift coefficients for each submatrix, the proposed method dramatically reduces the ROM size by 98.2% and 88.9% when compared with the G matrix method and RU method, respectively. Moreover, the proposed encoder reduces the number of XOR logic gate counts by 1.65 times compared with the RU method. This leads to a significant reduction in the hardware complexity for the proposed encoder as these components are the main contributors of logic resources in the encoder architecture. Hence, the proposed encoding structure shows a significant advantage over other LDPC encoding methods with respect to hardware complexity. As can be seen from Table 6, the Gaussian elimination approach requires only 23 clock cycles to generate the encoded codeword for a given LDPC code. However, this method suffers from a significant storage overhead which makes it less of an idea for implementation. From the analysis of the RU design, it was found to require 471 clock cycles per codeword. This is significantly higher than that of the proposed encoder, which only requires 70 clock cycles. From Table 6, it can be observed that

the number of clock cycles required per codeword for the encoding of the proposed encoder design decreased to 14.8% of that of RU method.

Table 5. Comparison between Gaussian method, RU method, and proposed method.

		Gaussian	RU	Proposed
Area	Flip-flops	k_bZ	$(k_b+g)Z$	$(k_b+g)Z$
	XOR gates	$(k_bZ-1)m_bZ$	$2m_b+(m_b-g)Z$	$(k_b+2g-1)Z$
	AND gates	$k_bm_bZ^2$	–	–
	Barrel shifter (Z bits)	–	–	k_b+g+1
	Memory (bits)	ROM = $k_bm_bZ^2$	ROM = $(245x+29y+274)Z$	ROM = $\tilde{q}[m_b(k_b+g)-g^2]$ λ_mem = gZ
Speed (clock cycles)		k_b+1	$28Z+k_b+1$	n_b+2

Where $\tilde{q}=\lceil log_2 Z\rceil$; $x=\lceil \log_2(k_bZ)\rceil$; $y=\lceil \log_2(gZ)\rceil$.

Table 6. Comparison between Gaussian method, RU method, and proposed method for submatrix size $Z=16$.

		Gaussian	RU	Proposed
Area	Flip-flops	352	416	416
	XOR gates	258,336	764	464
	AND gates	259,072	–	–
	Barrel shifter (Z bits)	–	–	27
	Memory (bits)	ROM = 259,072	ROM = 42,488	ROM = 4720 λ_mem = 64
Speed (clock cycles)		23	471	70

The ASIC post synthesis implementation results on TSMC 65–nm CMOS technology are shown in Table 7, for various QC-LDPC encoders with expansion factors Z = 30, 64, 96, 144, and 352, which are indicated in the table as BG1-Z30, BG1-Z64, BG1-Z96, BG1-Z144, and BG1-Z352, respectively. In Table 7, $\tilde{q}$ size denotes the word length required to store the shift sizes while CPC stands for the number of clock cycles required per codeword for encoding. Note that all input data bits were assumed to be available for encoding, and the serialization factors are not included in the results. In the proposed design, the CPC is equal to the maximum number of clock cycles required for the calculation of the p_a and p_c parity check bits. The computation of p_a requires $(g+2)$ clock cycles, in which g clock cycles are used to compute all the λ values and p_{a_1}, and two extra clock cycles are required for estimation of the remaining parity bits in the p_a portion. The computation of p_c requires (m_b-g) clock cycles. Hence, this method requires (m_b+2) clock cycles in total. The information throughput reported in Table 7 is given by the formula

$$Throughput=\frac{m_b\times Z\times f_{max}}{CPC}, \tag{35}$$

where f_{max} is the maximum operating frequency (post synthesis). For different submatrix sizes, the throughput varied from 22.1–202.4 Gbps. In Table 7, the occupied areas are also reported. It should be noted that there is a significant increase in the core area when processing higher submatrix sizes. Since encoder architecture of a higher submatrix size Z requires a higher $\tilde{q}$ size, additional memory and hardware components are required. It is shown that the encoding complexity of the proposed design is linearly proportional to the submatrix size Z of the code. To keep the throughput comparison on equal basis, the throughput-to-area ratio metric was further defined as

$TAR = Throughput/Area$ (Gbps/mm^2). For all the considered submatrix sizes in Table 7, the TAR ranged from 520–597 Gbps/mm^2.

Based on the implementation results presented above, it is clear that the design methodology is applicable to different submatrix sizes and offers a significantly high area efficiency and high information throughput, which is more than enough to satisfy the throughput requirement for the 5G NR standard.

Table 7. ASIC implementation results of LDPC encoders for different lifting sizes $Z = 30$, 64, 96, 144, and 352.

Encoder	BG1-Z30	BG1-Z64	BG1-Z96	BG1-Z144	BG1-Z352
CMOS technology	65-nm	65-nm	65-nm	65-nm	65-nm
Base graph	BG1	BG1	BG1	BG1	BG1
Subset	8	1	2	5	6
Submatrix size Z	30	64	96	144	352
$\tilde{q}$ size (bits)	5	6	7	8	9
CPC (clock cycles)	48	48	48	48	48
Max. frequency (MHz)	769	714	667	645	600
Throughput (Gbps)	22.1	43.8	61.4	89	202.4
Area (mm^2)	0.037	0.077	0.117	0.171	0.389
Gate counts	45.9 K	96 K	146.3 K	214 K	486.4 K
TAR † (Gbps/mm^2)	597	569	525	520	520

† $TAR = Throughput/Area$.

6. Conclusions

In this paper, a novel low-complexity high-throughput encoder approach for the 5G NR standard is proposed. Based on the proposed encoding algorithm, five encoder architectures with different submatrix sizes were implemented. The derived architecture exhibited a significantly lower hardware complexity, as it decreased the memory and logic component requirements. The proposed design demonstrates a superior performance to the alternative methods. Moreover, the synthesis results revealed that the proposed design is appropriate for the high throughput 5G standard.

Author Contributions: T.T.B.N. conceptualized the idea of this research, conducted experiments, collected data, and prepared the original version. T.N.T. reviewed, analyzed data, and updated the manuscript. H.L. supervised, validated, reviewed, and supported the research with funding.

Funding: This work was supported by the INHA UNIVERSITY Research Grant.

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

References

1. Gallager, R.G. *Low-Density Parity-Check Codes*; MIT Press: Cambridge, MA, USA, 1963.
2. MacKay, D.J.C.; Neal, R.M. Near Shannon Limit Performance of Low-Density Parity-Check Codes. *Electron. Lett.* **1996**, *32*, 1645–1646. [CrossRef]
3. Huo, Y.; Dong, X.; Xu, W. 5G Cellular User Equipment: From Theory to Practical Hardware Design. *IEEE Access* **2017**, *5*, 13992–14010. [CrossRef]
4. Session Chairman (Nokia). Chairman's Notes of Agenda Item 7.1.5 Channel Coding and Modulation, 3GPP TSG RAN WG1 Meeting No. 87, R1-1613710 (2016). Available Online: https://portal.3gpp.org/ngppapp/CreateTdoc.aspx?mode=view&contributionId=752413 (accessed on 22 May 2019).
5. Richardson, T.; Kudekar, S. Design of Low-Density Parity Check Codes for 5G New Radio. *IEEE Commun. Mag.* **2018**, *56*, 28–34. [CrossRef]
6. Ji, W.; Wu, Z.; Zheng, K.; Zhao, L.; Liu, Y. Design and Implementation of a 5G NR System Based on LDPC in Open Source SDR. In Proceedings of the 2018 IEEE Globecom Workshops (GC Wkshps), Abu Dhabi, UAE, 9–13 December 2018; pp. 1–6.

7. Tang, H.; Xu, J.; Kou, Y.; Lin, S.; Abdel-Ghaffar, K. On Algebraic Construction of Gallager and Circulant Low-Density Parity-Check Codes. *IEEE Trans. Inf. Theory* **2004**, *50*, 1269–1279. [CrossRef]
8. Ajaz, S.; Nguyen, T.T.B; Lee, H. An Area-Efficient Half-Row Pipelined Layered LDPC Decoder Architecture. *J. Semicond. Technol. Sci.* **2017**, *17*, 845–853. [CrossRef]
9. Nguyen, T.T.B; Lee, H. Low-Complexity Multi-mode Multi-way Split-row Layered LDPC Decoder for Gigabit Wireless Communications. *Integration* **2018**. [CrossRef]
10. Ajaz, S.; Lee, H. Efficient multi-Gb/s multi-mode LDPC decoder architecture for IEEE 802.11ad applications. *Integration* **2015**, *51*, 21–36. [CrossRef]
11. Ajaz, S.; Lee, H. An efficient radix-4 Quasi-cyclic shift network for QC-LDPC decoders. *IEICE Electron. Express* **2014**, *11*, 1–6. [CrossRef]
12. Ajaz, S.; Lee, H. Reduced-complexity Local Switch Based Multi-mode QC-LDPC Decoder Architecture for Gigabit Wireless Communications. *IET Electron. Lett.* **2013**, *49*, 1246–1248. [CrossRef]
13. Richardson, T.J.; Urbanke, R.L. Efficient Encoding of Low-density Parity-check Codes. *IEEE Trans. Inf. Theory* **2001**, *47*, 638–656. [CrossRef]
14. Khodaiemehr, H.; Kiani, D. Construction and Encoding of QC-LDPC Codes Using Group Rings. *IEEE Trans. Inf. Theory* **2017**, *63*, 2039–2060. [CrossRef]
15. Huang, Q.; Tang, L.; He, S.; Xiong, Z.; Wang, Z. Low-Complexity Encoding of Quasi-Cyclic Codes Based on Galois Fourier Transform. *IEEE Trans. Commun.* **2014**, *62*, 1757–1767. [CrossRef]
16. Li, Z.; Chen, L.; Zeng, L.; Lin, S.; Fong, W. Efficient Encoding of Quasi-cyclic Low-density Parity-check Codes. *IEEE Trans. Commun.* **2006**, *54*, 71–81. [CrossRef]
17. Ilani, I. Designing and Encoding QC-LDPC Codes Using Matrices over Commutative Rings. In Proceedings of the 2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE), Eilat, Israel, 16–18 November 2016; pp. 1–5.
18. Jung, Y.; Chung, C.; Jung, Y.; Kim, J. 7.7 Gbps Encoder Design for IEEE 802.11ac QC-LDPC Codes. *J. Semicond. Technol. Sci.* **2014**, *14*, 419–425. [CrossRef]
19. Cohen, A.E.; Parhi, K.K. A Low-Complexity Hybrid LDPC Code Encoder for IEEE 802.3an (10GBase-T) Ethernet. *IEEE Trans. Signal Process.* **2009**, *57*, 4085–4094. [CrossRef]
20. Zhang, P.; Liu, C.; Jiang, L. Efficient Encoding of QC-LDPC Codes Based on Rotate-left-accumulator Circuits. *Electron. Lett.* **2013**, *49*, 810–812. [CrossRef]
21. Jung, Y.; Kim, J. Memory-efficient and High-speed LDPC Encoder. *Electron. Lett.* **2010**, *46*, 1035–1036. [CrossRef]
22. Li, H.; Bai, B.; Mu, X.; Zhang, J.; Xu, H. Algebra-Assisted Construction of Quasi-Cyclic LDPC Codes for 5G New Radio. *IEEE Access* **2018**, *6*, 50229–50244. [CrossRef]
23. Chen, L.; Xu, J.; Djurdjevic, I.; Lin, S. Near-Shannon-limit Quasi- cyclic Low-density Parity-check Codes. *IEEE Trans. Commun.* **2004**, *52*, 1038–1042. [CrossRef]
24. Chen, L.; Lan, L.; Djurdjevic, I.; Lin, S.; Abdel-Ghaffar, K. An Algebraic Method for Constructing Quasi-cyclic LDPC Codes. In Proceedings of the International Symposium on Information Theory and Its Applications, Parma, Italy, 10–13 October 2004; pp. 535–539.
25. Li, J.; Lin, S.; Abdel-Ghaffar, K.; Ryan, W.; Costello, D.J., Jr. *LDPC Code Designs, Constructions, and Unification*; Cambridge University Press: Cambridge, UK, 2017.
26. Chen, T.; Vakilinia, K.; Divsalar, D.; Wesel, R.D. Protograph Based Raptor-like LDPC Codes. *IEEE Trans. Commun.* **2015**, *63*, 1522–1532. [CrossRef]
27. Ad-Hoc chair (Nokia). Chairman's Notes of Agenda Item 7.1.4. Channel Coding, 3GPP TSG RAN WG1 Meeting AH 2, R1-1711982 (2017). Available Online: https://portal.3gpp.org/ngppapp/CreateTdoc.aspx?mode=view&contributionId=805088 (accessed on 22 May 2019).
28. Yasotharan, H.; Carusone, A.C. A Flexible Hardware Encoder for Systematic Low-density Parity-check Codes. In Proceedings of the 52nd IEEE International Midwest Symposium on Circuits and Systems, Cancun, Mexico, 2–5 August 2009; pp. 54–57.