Design of Raptor-Like LDPC Codes and High Throughput Decoder Towards 100 Gbit/s Throughput

LI Hanwen, BI Ningjing, SHA Jin
(School of Electronic Science & Engineering, Nanjing University, Nanjing 210000, China)

DOI: 10.12142/ZTECOM.202303012
https://kns.cnki.net/kcms/detail/34.1294.TN.20230718.1034.002.html, published online July 19, 2023
Manuscript received: 2023-01-29

Abstract: This paper proposes a raptor-like low-density parity-check (RL-LDPC) code design together with the corresponding decoder hardware architecture aiming at next-generation mobile communication. A new kind of protograph different from the 5G new radio (NR) LDPC basic matrix is presented, and a code construction algorithm is proposed to improve the error-correcting performance. A multi-core layered decoder architecture that supports up to 100 Gbit/s throughput is designed based on the special protograph structure.

Keywords: RL-LDPC; error floor; high throughput; protograph; 5G NR

Citation (Format 1): LI H W, BI N J, SHA J. Design of raptor-like LDPC codes and high throughput decoder towards 100 Gbit/s throughput [J]. ZTE Communications, 2023, 21(3): 86 - 92. DOI: 10.12142/ZTECOM.202303012
Citation (Format 2): H. W. Li, N. J. Bi, and J. Sha, "Design of raptor-like LDPC codes and high throughput decoder towards 100 Gbit/s throughput," *ZTE Communications*, vol. 21, no. 3, pp. 86 - 92, Sept. 2023. doi: 10.12142/ZTECOM.202303012.

1 Introduction

5G is emerging as a promising technology on a global scale. However, with the demand for virtual reality (VR), Metaverse, and other high throughput applications, the transmission speed needs to be improved. As one of the most important technologies in wireless communication, channel coding is of great significance to the improvement of system reliability[1].

Low-density parity-check (LDPC) codes have been applied in many communication systems due to their top performance, which is close to the Shannon limit and suitable for parallel structures. LDPC codes were first proposed by GALLAGER in 1963[2], and in 1996, MACKAY and NEAL proposed the positional degree propagation iterative decoding algorithm for LDPC codes[3], which significantly improved performance.

Code construction is a critical part of LDPC code design because it directly affects error correction performance. The random construction method cannot guarantee the absence of short cycles. Based on this, the progressive edge growth (PEG) algorithm was proposed[4], which tries to avoid short cycles. Some improved PEG algorithms were proposed in Refs. [5 - 6], which failed to guarantee the good performance of the code. The approximate cycle extrinsic (ACE) message degree algorithm was suggested in Ref. [7], which could eliminate the cycles selectively in the LDPC codes. The codes constructed by the ACE algorithm have lower error floors but perform worse in the waterfall region than the PEG algorithm.

Typically, the LDPC code can be obtained by lifting the protograph, which is a structure that governs the macroscopic statistics of the code. The protograph is an important optimization object in LDPC design because it determines the error-correcting performance of LDPC codes. Some wireless communication LDPC codes are designed with this approach, like the 5G NR LDPC code.

The belief propagation (BP) algorithm is the classic decoding algorithm for LDPC codes. The BP algorithm is computationally intensive and not conducive to hardware implementation, so researchers have proposed the min-sum decoding algorithm, which still uses the idea of iterative decoding with probabilistic computation but reduces the hardware resource greatly. It occupies an important position in the implementation of LDPC decoders, and most of the decoders adopt this method as a guiding idea.

The code construction algorithm and the decoding algorithm tend to affect the performance of the code, while the

This work is supported in part by ZTE Industry-University-Institute Cooperation funds under Grant No. 2020ZTE01-03.

hardware implementation of the decoder has more impact on the throughput. Many researchers have proposed different decoder architectures in the hardware implementation aspect.

In Ref. [8], researchers used a fully parallel decoding scheme to partition the global interconnection and weaken the routing pressure by grouping the variable nodes. The architecture achieves a throughput of 92.8 Gbit/s. A partial parallel decoder architecture with multi-frame pipeline decoding is proposed in Ref. [9] to increase the parallelism. It can achieve higher throughput while having high decoding flexibility. In Ref. [10], researchers proposed an iterative unfolding architecture, which fully expanded the hardware architecture used for one iteration so that one iteration could be performed per cycle. This approach lacks flexibility while significantly increasing throughput. The multi-core decoder was implemented in Ref. [11], which could satisfy high throughput while being relatively flexible.

In this paper, we design a new kind of RL-LDPC code aiming at next-generation mobile communication and give the construction results of the core matrix. A code construction algorithm is proposed to optimize the performance of the code by eliminating cycles of lengths 6 and 4. In addition, a hardware architecture is proposed to match the above codes using a multi-core row-layered decoding approach to achieve a throughput of 100 Gbit/s. The multi-core decoder mainly includes an input buffer, a controller, decoder cores and their memory blocks, an output buffer, and an output selection module.

The remainder of this paper is organized as follows. Section 2 introduces the code construction method we proposed. Then we describe the hardware architecture in Section 3. In Section 4, the performance and hardware resource consumption of the code are presented. Finally, the conclusions are drawn in Section 5.

2 Codes Construction

2.1 RL-LDPC Codes Construction

LDPC code is determined by an $m \times n$ parity check matrix $\boldsymbol{H}$, where one element in the $\boldsymbol{H}$ matrix is much less than zero, m represents the number of rows, and n represents the number of columns. Each column corresponds to a variable node (VN), and each row corresponds to a check node (CN). The information block length is $k = m - n$. The elements in the parity check matrix $\boldsymbol{H}$ indicate whether there is information exchange between the VNs and the CNs. The tanner diagram can be used to represent the information transmission network that connects the CNs and VNs. As shown in Fig. 1, the circles represent the CNs and the squares represent the VNs. The structure of starting from one node and making a circle along the connection to return to the node is called a "cycle". Short cycles can have a bad effect on performance.

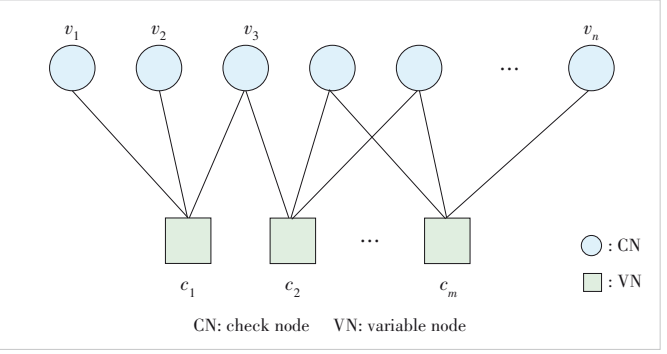
The protograph structure of RL-LDPC is shown in Fig. 2. $\boldsymbol{H}_{\mathrm{HRC}}$ is the core matrix framed in green in Fig. 2, which corresponds to the highest code rate of the LDPC code, and $\boldsymbol{H}_{\mathrm{IRC}}$ is the extension matrix framed in blue in Fig. 2. The sizes are marked in the figure. The size of the extension matrix can be adjusted according to different code rates. $\boldsymbol{H}_{\mathrm{HRC}}$ and $\boldsymbol{H}_{\mathrm{IRC}}$ are constructed and optimized separately[12]. Matrix $\boldsymbol{B}$ is a square matrix with a double diagonal structure and its size is $P \times P$. Matrix $\boldsymbol{I}$ is an identity matrix, which corresponds to the variable node part of RL-LDPC with a degree of 1.

$\boldsymbol{H}_{\mathrm{HRC}}$ consists of a number of elements shown as follows:

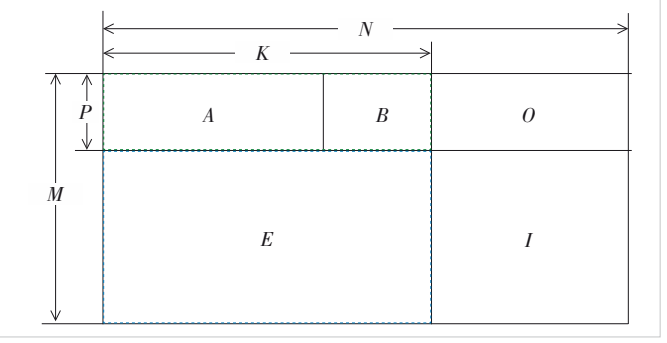
$$\boldsymbol{H}_{\mathrm{HRC}} = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,K+P} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,K+P} \\ \vdots & \vdots & \vdots & \vdots \\ h_{P,1} & h_{P,2} & \cdots & h_{P,K+P} \end{bmatrix}. \quad (1)$$

The parity check matrix is obtained by expanding the base matrix above by a lifting size Z. Therefore, each element of $\boldsymbol{H}_{\mathrm{HRC}}$ is replaced by a square matrix of $Z \times Z$. Elements "−1" are replaced by the all-zero matrix, while $h_{i,j}>0$ elements are replaced by a circulant matrix, obtained by right shifting the identity matrix by $h_{i,j}$ positions. The range of cyclic lifting number $h_{i,j}$ is 0 to $Z - 1$.

The method of code construction based on protograph is used in this paper. A protograph with a low iterative decoding threshold needs to be constructed first, and then the lifting numbers are generated to obtain our LDPC code. Following



▲Figure 1. Tanner graph



▲ Figure 2. Structure of raptor-like low-density parity-check (RL-LDPC) code

the below principles when constructing a protograph can ensure the basic performance of the code.

Keeping the variable node degree ⩾ 3 can ensure linear minimum distance growth with the blocklength in the process of constructing $\boldsymbol{H}_{\mathrm{HRC}}$. The linear minimum distance growth property ensures that the performance of the code does not degrade as the block length increases. Adding some variable nodes with the degree −2 or even degree −1 in the code is beneficial to the iterative decoding threshold. A small number of such nodes will not destroy the linear minimum distance growth property, but they need to be added prudently. Adding punctured variable nodes in the $\boldsymbol{H}_{\mathrm{HRC}}$ can improve the threshold of the code[13]. During the protograph construction, almost every check node in $\boldsymbol{H}_{\mathrm{IRC}}$ should be connected to the punctured variable node.

2.2 Protograph Construction Constraint

We present the constraints that need to be followed when constructing a protograph. "1" in Fig. 3 represents a non-zero submatrix, and "0" represents a zero submatrix. The specific constraints can be described as follows. Each protograph row is obtained by shifting the above row cyclically. The first row is shifted by one column to generate the second row, the second row is shifted by one column to generate the third row, and so on. The number of shifts can be flexibly adjusted as needed. Fig. 3 shows the protograph structure with a size of 4×16. The positions of "1" elements in rows are obtained by shifting the above row by one column. Such a construction structure can simplify the design of the reading and writing networks in traditional architecture, which reduces the consumption of hardware resources.

In practical applications, this constraint will not be as regular as shown in the figure above. In our decoder design, the constraint is added only in the first K columns of the protograph. The left columns are directly solidified into a double diagonal matrix to adapt to the invertibility of the matrix and encoding requirements. The corresponding reading and writing networks are consistent with the traditional structure.

Our construction method can directly construct the protograph of the required size randomly: determining the protograph of the first row first, and then shifting to obtain the protograph of other rows. The check part of the protograph is fixed as a double diagonal structure and does not participate in shifting.

The specific steps are as follows:

1	1	0	1	1	0	1	0	1	1	1	0	1	1	1	1
1	1	1	0	1	1	0	1	0	1	1	1	0	1	1	1
1	1	1	1	0	1	1	0	1	0	1	1	1	0	1	1
1	1	1	1	1	0	1	1	0	1	0	1	1	1	0	1

▲**Figure 3. Construction constraint structure**

1) Generate the first row of the protograph randomly;

2) Generate additional rows of the protograph by shifting the first row;

3) Randomly reduce or increase the number of "1" elements in the protograph to improve the threshold.

2.3 Codes Construction Method

In addition to searching for protograph and optimizing the iterative decoding threshold, a code construction algorithm is designed, which outperforms the conventional PEG algorithm. The codes constructed by using PEG algorithms contain a small number of 6 cycles, so we suggest a new algorithm for code construction to guarantee the absence of 6 cycles in the construction.

Starting from the code we randomly construct, our algorithm searches for all 6 cycles and 4 cycles in the code and modifies the submatrix's lifting number, which connects the maximum number of cycles, so that the number of cycles passing through this submatrix is reduced. This process is repeated until the number of cycles cannot be reduced through all submatrices.

Set the parity check matrix of the constructed LDPC code as $\boldsymbol{H}$. P is the corresponding protograph, $h_{i,j}$ is each element of $\boldsymbol{H}$, $p_{i,j}$ is each element of P, and n is the iteration number. Set $g4_{i,j,n}$, $g6_{i,j,n}$ as the number of 4 cycles and 6 cycles associated with $h_{i,j}$. The element with the largest number of associated 4 cycles is $g4_{\max\_i,\max\_j}$. The element with the largest number of associated 6 cycles is $g6_{\max\_i,\max\_j}$. Set the lifting size as Z.

The specific algorithm is shown below:

Algorithm 1: Cycle elimination construction algorithm

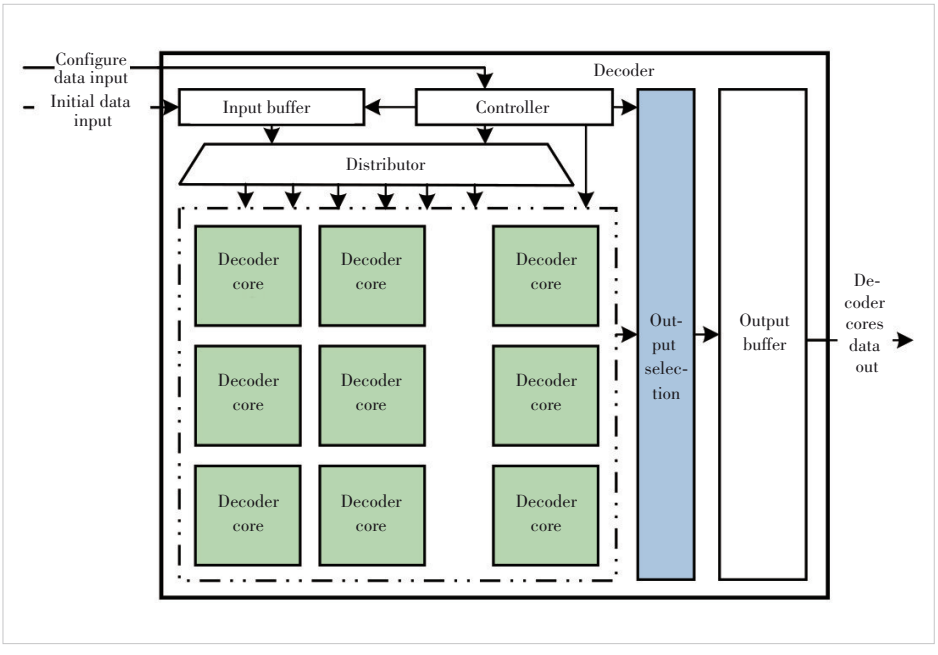
Input: the protograph P, the size of protograph M,N
Output: the parity check matrix $\boldsymbol{H}$
for (i = 0 to M) do
  for (j = 0 to N) do
    if $p_{i,j}$≠0 **then**
      $h_{i,j}$= random(0, Z). // Initializations
      else
      $h_{i,j}$=−1.
    end if
  end for
end for
n = 0.
while (n< Maximum number of iterations)
for (i = 0 to M) do
  for (j =0 to N) do
    if $p_{i,j}$≠0 **then**
      $g4_{i,j,n}$ = cycle4_calculate(H,i,j).
      $g6_{i,j,n}$ = cycle6_calculate(H,i,j). // Cycle calculation
      else
    $g4_{i,j,n}$=0.
      $g6_{i,j,n}$=0.

end if
end for
end for
$g4_{max\_i,max\_j} = MAX(g4_{i,j,n})$.
$h_{max\_i,max\_j} = random(0,Z)$.
$g6_{max\_i,max\_j} = MAX(g6_{i,j,n})$.
$h_{max\_i,max\_j} = random(0,Z)$.// Elemental position calculation.
$n = n + 1$.
if $MAX(g4_{i,j,n}) == MAX(g4_{i,j,n-1})$ and $MAX(g6_{i,j,n}) == MAX(g6_{i,j,n-1})$ **then**
Record current H. // Exit judgment
break.
end if
end while

3 Hardware Design

We design a multi-core decoder architecture to achieve a throughput of 100 Gbit/s. The architecture is shown in Fig. 4.

The multi-core decoder includes an input buffer, controller, decoder core, output buffer, and output selection module. The function of the output selection module is to select the decoded data from multiple decoder cores according to the controller configuration. The input buffer module converts the data fragments into a complete codeword. The input buffer module operates at a different higher frequency clock domain to accommodate the data stream with the internal blocks. The controller receives external configuration information, distributes the complete codeword to the free decoder core, and controls the output of the data frame at the end of the decoding.



▲**Figure 4. Multi-core decoder architecture**

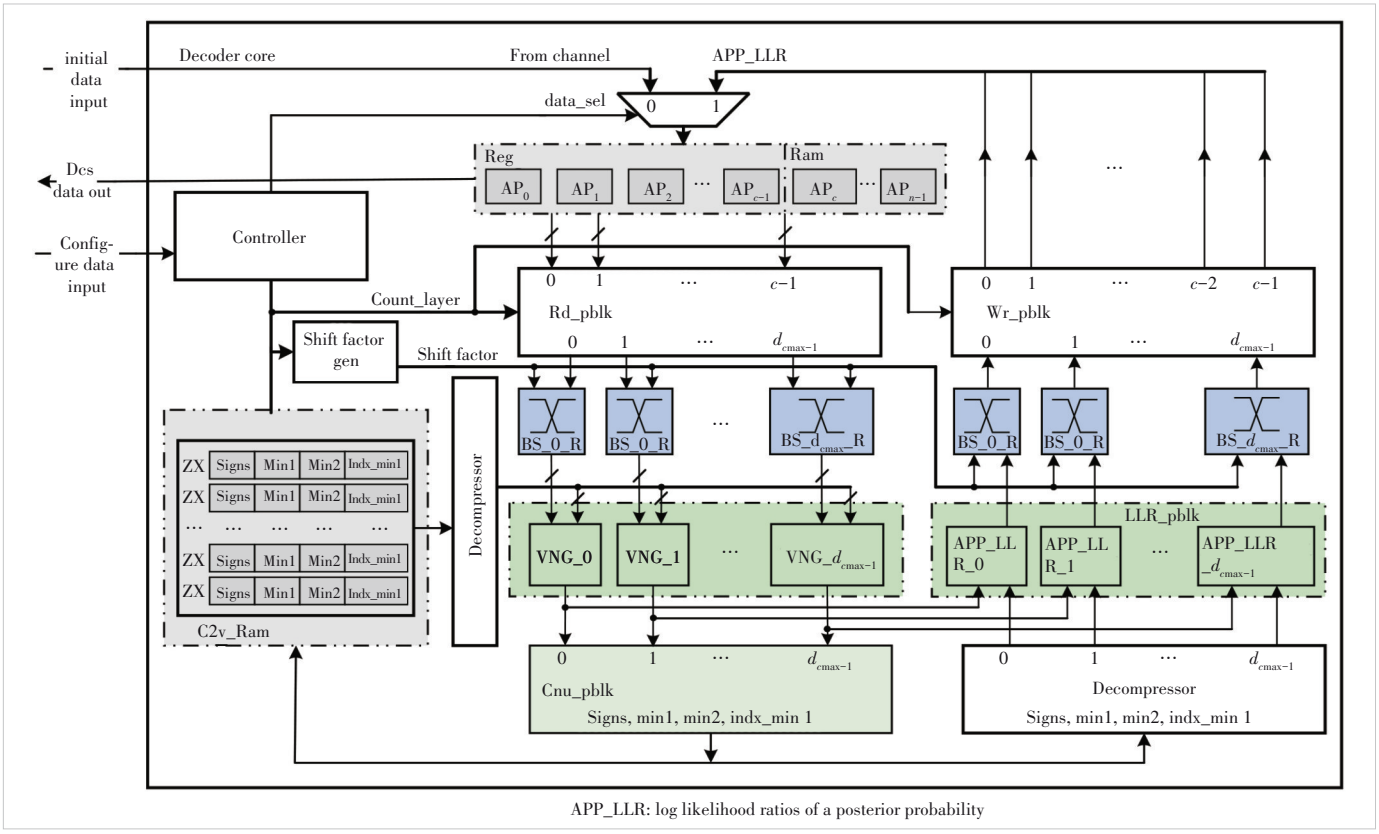
The decoder core, shown in Fig. 5, is the core computing module in the multi-core decoder architecture. It includes the log likelihood ratios of a posterior probability (APP_LLR) information storage module, check-node to variable-node (C2V) message storage module, and other main computing units.

The decoder core[14] receives data and configuration information from the controller. According to the configuration information, iterative decoding is carried out layer by layer. The initial log likelihood ratios (LLR) information or the updated LLR information of the current layer is read from the LLR storage module during the decoding of each layer. The LLR storage module contains two memory components: shift registers and static random-access memory (SRAM). The initial LLR information of each codeword to be decoded or the LLR information that is continuously updated during an iteration is stored in these two pieces of memory. The shift registers receive the initial information and shift according to the current layer index to accommodate the matrix's shift constraints during decoding.

The LLR information to be updated by up to d_{cmax} sets is selected and fed into barrel shifters by the reading network (Rd_pblk). Note that Rd_pblk is implemented by multiplexers in Ref. [15]. But based on our proposed special protograph structure, the Rd_pblk selects the same sets of LLR information from the shift registers without using multiplexers.

The selected information is aligned to the check node through the barrel shifters. In the same clock cycle, the compressed stored check-node to variable-node (C2V) messages are read from the C2V_Ram and distributed to the variable node processing array (VNU_pblk) through the decompressor. The LLR and C2V messages are input to the VNU_pblk to calculate the new variable-node to check-node (V2C) messages. The new V2C messages are input to the check node processing array (CNU_pblk) to calculate C2V messages. At last, the C2V messages are stored in the C2V_Ram.

New V2C messages and new C2V messages are added in the LLR_pblk to obtain new LLR information, which is then realigned to the VNs via barrel shifters. The aligned new LLR information is written back to the registers and RAM by writing network (Wr_pblk) according to the current layer index. Based on the special protograph structure, Wr_pblk only updates LLR information at fixed locations. The decoding of one layer is completed after the LLR information is written back. When the decoding of all layers is complete, the current it-



▲Figure 5. Decoder core architecture

eration is finished. The early termination function in the decoding process is implemented by LLR_pblk, which can calculate the parity check result of the current layer. When all the parity checks are satisfied, the decoding will stop successfully and enter the waiting state for the decision information output.

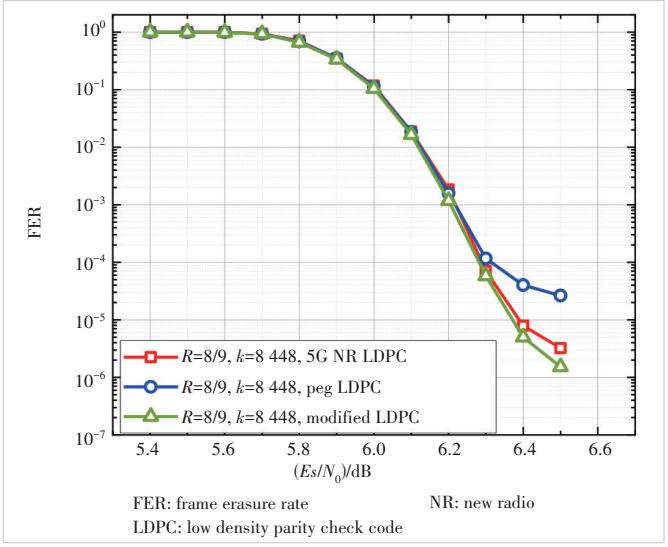
4 Implementation Results

4.1 Results of Improved Error Floor

We build a compute unified device architecture (CUDA)-based code performance testing platform, including a noise generator, a decoder, and a decoding result statistics module. Noise is added directly to the all-zero codeword to improve throughput during testing. And we test the code performance on a single GeForce GTX 1080 Ti GPU.

We evaluate the performance of the proposed LDPC codes in the additive white Gaussian noise (AWGN) channel using quadrature phase shift keying (QPSK) modulation. The belief propagation (BP) flooding decoding schedule is used and the maximum number of iterations is set to 50.

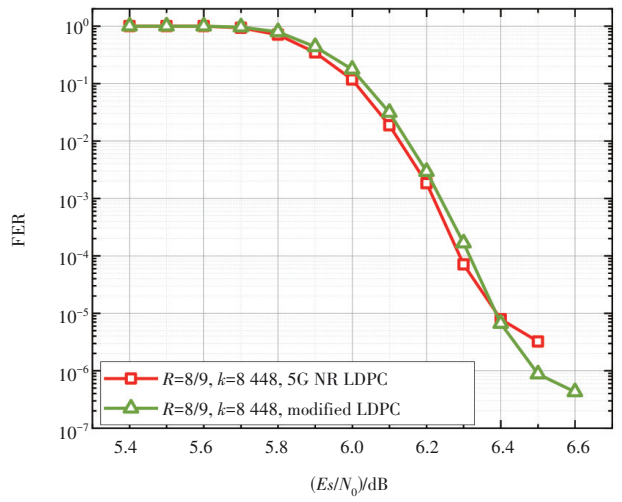
Compared with the 5G NR LDPC code and the PEG code, the code performance constructed by our algorithm has a certain optimization effect, as shown in Fig. 6. As we reduce the number of 6 cycles, the error floor is optimized.



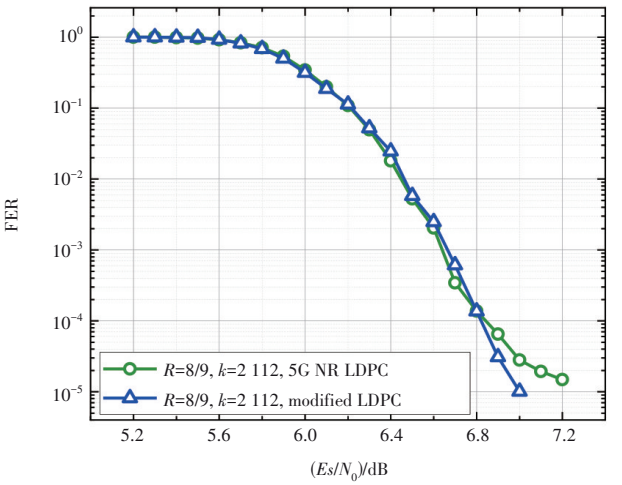
▲Figure 6. Performance comparison after optimizing the number of 6-cycle

4.2 Optimized LDPC Code Performance

After applying the code constraints, the code is constructed using the construction algorithm described above, and performance curves with an information blocklength of k=8 448 bits are shown in Fig. 7. Although the code we constructed still has a gap of less than 0.05 dB compared with the 5G NR



(a) Information blocklength is 8 848 bits



(b) Information blocklength is 2 112 bits

FER: frame erasure rate NR: new radio
LDPC: low density parity check code

▲**Figure 7. Performance comparison under rate of 8/9**

LDPC in the waterfall region, it has an improvement in the error floor region. The size of the parity check matrix we used during the test is 5×29, and the lifting size is 352.

The performance of the matrix with a shorter information blocklength of k=2 112 bits is shown in Fig. 7. The lifting size is 88. The code we constructed also has the advantage in the error floor region.

4.3 Hardware Implementation Results

For LDPC codes with a lifting size of 352, we design the corresponding decoder prototype to evaluate and compare the hardware resources accurately before and after adding the protograph construction constraints. The decoder consists of buffers, a controller, several decoder cores, etc. We use a quantization length of 6 bits for the LLR information and 4 bits for the internal messages.

The implementation results, as well as the corresponding comparisons, are reported in this section. The formula for calculating throughput is as follows:

$$\text{Throughput} = R \times \frac{N_{\text{cores}} \times n}{N_{\text{iter}} \times M} \times f_{\text{clk}}, \quad (2)$$

where R is the code rate, N_{cores} is the number of decoder cores, n is the length of the codeword, M is the number of basic matrix rows used for decoding, and N_{iter} is the average number of iterations. To achieve the desired throughput with the code, the information blocklength of which is 8 448 bits, 3 cores are required.

In order to evaluate the effectiveness of our optimization methods, the synthesis results of decoders with and without applying the optimizations proposed in Sections 2 and 3 are listed. We use the architecture from Ref. [15] as the baseline. Table 1 summarizes the hardware resources of the decoder core on the Xilinx Virtex UltraScale XCVU440 field programmable gate array (FPGA). The decoder core occupies about 34.0% of the lookup tables and 1.2% of the flip-flops available on the device. Compared with the architecture used in Ref. [15], the proposed decoder reduces the resources of the lookup table by about 15%.

▼**Table 1. Implementation results**

Resources	Implementation in Ref. [15]	Proposed
LUTs	1 015 935	861 574
FFs	62 460	62 460
Brams	66	66

Brams: block-RAMs FFs: flip-flops LUTs: lookup tables

5 Conclusions

This paper proposes an RL-LDPC code aiming at next-generation mobile communication, with a corresponding hardware architecture with a throughout up to 100 Gbit/s. In this code, the designed structure of the protograph is different from the conventional 5G NR LDPC basic matrix, and an improved code construction algorithm is investigated for better performance. The decoder implemented in our work can achieve a throughput higher than 100 Gbit/s, and compared with the traditional architecture, the resources of the lookup table are reduced by about 15% on Xilinx Virtex UltraScale XCVU440 FPGA.

References

[1] VENUGOPAL T, RADHIKA S. A survey on channel coding in wireless networks [C]//Proceedings of 2020 International Conference on Communication and Signal Processing (ICCSP). IEEE, 2020: 784 – 789. DOI: 10.1109/ICCSP48568.2020.9182213
[2] GALLAGER R. Low-density parity-check codes [J]. IRE transactions on infor-

mation theory, 1962, 8(1): 21 – 28. DOI: 10.1109/TIT.1962.1057683
[3] MACKAY D J C, NEAL R M. Near Shannon limit performance of low density parity check codes [J]. Electronics letters, 1996, 32(18): 1645. DOI: 10.1049/el: 19961141
[4] HU X Y, ELEFTHERIOU E, ARNOLD D M. Regular and irregular progressive edge-growth tanner graphs [J]. IEEE transactions on information theory, 2005, 51(1): 386 – 398. DOI: 10.1109/TIT.2004.839541
[5] DIOUF M, DECLERCQ D, FOSSORIER M, et al. Improved PEG construction of large girth QC-LDPC codes [C]//The 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC). IEEE, 2016: 146 – 150. DOI: 10.1109/ISTC.2016.7593094
[6] LIU Y H, ZHANG M L, NIU X L. Quasi-cyclic low-density parity-check codes based on progressive cycle growth algorithm [C]//The sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC). IEEE, 2016: 854 – 857. DOI: 10.1109/IMCCC.2016.167
[7] TIAN T, JONES C R, VILLASENOR J D, et al. Selective avoidance of cycles in irregular LDPC code construction [J]. IEEE transactions on communications, 2004, 52(8): 1242 – 1247. DOI: 10.1109/TCOMM.2004.833048
[8] CHENG C C, YANG J D, LEE H C, et al. A fully parallel LDPC decoder architecture using probabilistic Min-sum algorithm for high-throughput applications [J]. IEEE transactions on circuits and systems I: regular papers, 2014, 61(9): 2738 – 2746. DOI: 10.1109/TCSI.2014.2312479
[9] LI M, DERUDDER V, DESSET C, et al. A 100 gbps LDPC decoder for the IEEE 802.11ay standard [C]//The 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC). IEEE, 2019: 1 – 5. DOI: 10.1109/ISTC.2018.8625286
[10] BONCALO O, AMARICAI A. Ultra high throughput unrolled layered architecture for QC-LDPC decoders [C]//IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, 2017: 225 – 230. DOI: 10.1109/ISVLSI.2017.47
[11] LI M, DERUDDER V, BERTRAND K, et al. High-speed LDPC decoders towards 1 Tb/S [J]. IEEE transactions on circuits and systems I: regular papers, 2021, 68(5): 2224 – 2233. DOI: 10.1109/TCSI.2021.3060880
[12] FANG Y, BI G A, GUAN Y L, et al. A survey on protograph LDPC codes and their applications [J]. IEEE communications surveys & tutorials, 2015, 17(4): 1989 – 2016. DOI: 10.1109/COMST.2015.2436705
[13] CHEN T Y, DIVSALAR D, WESEL R D. Protograph-based Raptor-like LDPC codes with low thresholds [C]//IEEE International Conference on Communications (ICC). IEEE, 2012: 2161 – 2165. DOI: 10.1109/ICC.2012.6363996
[14] NGUYEN-LY T T, GUPTA T, PEZZIN M, et al. Flexible, cost-efficient, high-throughput architecture for layered LDPC decoders with fully-parallel processing units [C]//Euromicro Conference on Digital System Design (DSD). IEEE, 2016: 230 – 237. DOI: 10.1109/DSD.2016.33
[15] CUI H X, GHAFFARI F, LE K, et al. Design of high-performance and area-efficient decoder for 5G LDPC codes [J]. IEEE transactions on circuits and systems I: regular papers, 2021, 68(2): 879 – 891. DOI: 10.1109/TCSI.2020.3038887

Biographies

LI Hanwen received his BS degree from Fuzhou University, China in 2020, and MS degree from Nanjing University, China in 2023. His research interests include very-large scale integration design for digital signal processing and error control coding.

BI Ningjing received her BS degree and MS degree in electrical engineering from Nanjing University, China, in 2020 and 2023, respectively. Her research interest focuses on digital very-large scale integration design.

SHA Jin (shajin@nju.edu.cn) received his BS degree in physics and PhD degree in electrical engineering from Nanjing University, China in 2002 and 2007, respectively. From 2012 to 2013, he was a visiting professor with Stanford University, USA. He is currently an associate professor with School of Electronic Science and Engineering, Nanjing University. His current research interests include very-large scale integration design for digital signal processing, error control coding, flash memory error control, and heterogeneous computing systems.