LNOMS-SS: A Low-Complexity LDPC Decoding Algorithm with Hardware Implementation

KaiShi Wang[1,2], Cheng Wang[1,2], Songsong Cai[1,2], Jiewen Jiang[1,2], Xi Gong[1,2], and Weidong Wang[1,2]
1. Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing
2. Information & Electronics Technology Lab, School of Electronic Engineering
Email: wangkaishi@bupt.edu.cn

***Abstract*—With the development of low orbit satellite communication technology, how to improve the communication speed of satellite communication systems and reduce the complexity of system design has become an important research issue. Since the invention of LDPC codes, their excellent decoding performance has been favored by researchers. QC-LDPC codes have become the channel coding scheme for the fifth generation new radio (5G NR) data channel, but at the same time, it has also brought challenges to hardware design. We propose the Layered Normalized Offset Minsum and three layer static scheduling (LNOMS-SS) algorithm. On the basis of the traditional Layered Normalized Minsum algorithm (LNMS) and Layered Offset Minsum algorithm (LOMS), the update order of each layer during hierarchical decoding is determined based on the row weight, column weight, and the number of perforated non-zero elements in the BG matrix. We also implemented the algorithm in hardware and designed an 8-levels LDPC decoder. We proposed innovative designs such as a pre-processing based routing network, a segment approximation based Minsum calculation module, and an improved checksum module. The simulation and implementation results show that our proposed LNOMS-SS algorithm has stronger performance than other algorithms, with lower hardware resource consumption and decoding latency.**

***Index Terms*—Layered Normalized Offset Minsum and improved three layer static scheduling (LNOMS-SS), Low Density Parity Check Code(LDPC), hardware implementation, low complexity.**

I. Introduction

Since LDPC codes were proposed, they have been widely studied and applied because of their excellent error correction ability. QC-LDPC codes have been adopted by 3GPP as one of the channel coding schemes for the fifth generation new radio (5G NR) [1]. With the continuous development of LEO communication technology, due to its global wide-area coverage characteristics, the problem of insufficient 5G coverage in remote areas is expected to be improved, so the number of researchers who are studying the feasibility of deploying 5G on low-orbit satellites is increasing.

Due to the special structure of the check matrix of LDPC codes, the performance of the decoding algorithm is an important factor affecting the performance of LDPC codes. At present, there have been many research results on LDPC decoding algorithms. While proposing the LDPC code, the Bit Flip decoding algorithm (BF) and the Belief Propagation decoding algorithm (BP) were also given in the paper of R.Gallager [2]. The former is a hard-decision decoding algorithm, which is easy to implement in hardware, but has poor error correction performance, while the latter is a soft-decision decoding algorithm, which has good error correction performance, but is difficult to implement in hardware.

The Min-sum (MS) algorithm is proposed in paper [3], [4], which simplifies the BP algorithm by mathematical approximation, greatly reducing hardware implementation complexity at the cost of reduced error correction performance, and remains one of the most widely used LDPC decoding algorithms at present. The papers [5], [6], [7], [8], [9], [10] are all improved algorithms of MS. Researchers have proposed Normalized Min-sum (NMS) or offset Min-sum (OMS), introduced novel algorithms such as deep learning, or improved the traditional MS information transmission method to enhance the error correction performance gap between the MS algorithm and the BP algorithm. Paper [11], [12] demonstrate FPGA implementation based on the aforementioned algorithm.

However, whether it is BP, MS, or improved MS, these are iterative decoding algorithms, which means that the decoder needs to repeatedly pass information between the check node and the variable node during each iteration. When using such algorithms, a large amount of memory space needs to be occupied, and a large amount of address computation needs to be performed, which greatly increases the implementation complexity, decoding latency, and power consumption of the LDPC decoder.

Below are some results of research to address the above issues. The idea of layered decoding is proposed in paper [13], [14], [15], which uses layered algorithm to avoid variable node message storage, thus saving a large amount of memory space; Paper [16] proposes another layered decoding algorithm, which blocks each layer of the parity check matrix, where each block is a cyclic matrix with cyclic weight k, improving the throughput of the decoder. Papers [17], [18], [19] study the effect of message update order of check line on decoding performance, and propose static scheduling and dynamic scheduling message update methods, respectively. Among them, the dynamic scheduling needs to flexibly change the check line message update order according to the code length and modulation coding method, and the decoding performance is better but the implementation complexity is extremely high; Static scheduling method performs check line message updates in some fixed order, which is easy

to implement in hardware, but the decoding performance is inferior to dynamic scheduling method. The essence of these two scheduling algorithms is to update the check rows that provide more iterative probability information in advance to gain the improvement of decoding performance at the cost of increased implementation complexity and memory consumption.

In summary, the main factors limiting the performance of LDPC decoder are complexity, resource utilization and processing latency. In order to improve these issues, the main work of this paper is as follows:

- An novel LDPC decoding algorithm is proposed, which combines LNMS, LOMS and three layer static scheduling algorithm (LNOMS-SS). Our algorithm can reduce the consumption of RAMs and FIFOs, accelerate the convergence rate of decoding. It has the advantages of lower complexity and less resource consumption.
- The above mentioned LDPC decoders are implemented with FPGA, and many innovative designs are adopted to reduce the complexity and latency of the decoders. For example, we design a pre-processing based routing unit, which reduces the memory overhead required by the decoding process; A Minsum calculation module based on segmented approximation is proposed, which reduces a large number of multiplication operations, resource occupation; In addition, we also improve the traditional verification method, reducing the time required for verification, thus reducing the decoding delay.

The structure of this paper is as follows: the second part introduces some preliminary knowledge and the proposed LDPC decoding algorithm; The third part introduces the implementation of the proposed LDPC decoder; The fourth part provides the results of simulation and implementation; The last part concludes.

II. PRELIMINARIES AND LNOMS-SS ALGORITHM

A. PRELIMINARIES ALGORITHM

The layered decoding algorithm of the LDPC codeword, refers to the check node and variable node exchanging messages with other nodes according to the Tanner diagram defined by the check matrix H. In order for the check node to have access to the latest messages during each iteration, researchers usually update the check node information in the order of check matrix layers, and the number of check rows for a layer is determined by the lifting size ZC. After each layer is updated, the updated layers behind can make use of the already updated LLR information, thus the layered decoding algorithm has the advantage of faster convergence compared to the flood decoding algorithm, and can greatly reduce the number of iterations of the decoding.

The normalized min-sum algorithm is illustrated as follows: Let c_n and v_m be the n-th check node and the m-th variable node in the Tanner diagram of the NR-LDPC code, respectively, where $1 < n < M$ and $1 < m < N$, and M, N are the numbers of nodes. Messages from v_m to c_n are recorded as $m_{v_m \to c_n}$, and messages from c_n to v_m as $m_{c_n \to v_m}$. Let $\mathcal{N}(c_n)$ and $\mathcal{N}(v_m)$ denote the set of variable nodes connected to c_n and the set of check nodes connected to v_m, respectively. When we update the c_n, these two messages can be calculated by the formula below:

$$m_{v_m \to c_n} = C_{v_m} - m_{c_n \to v_m} \tag{1}$$

$$m_{c_n \to v_m} = 2 \times \text{atanh} \left(\prod_{v_k \in \mathcal{N}(c_n) \setminus v_m} \tanh \left(\frac{m_{v_k \to c_n}}{2} \right) \right) \tag{2}$$

Where $v_m \in \mathcal{N}(c_i)$. $\mathcal{N}(c_n) \setminus v_m$ is a collection of variable nodes linked to c_n but not including the v_m. The C_{v_m} is the posterity information of the v_m, which is initialised in the first iteration as the LLR of the v_m. However, there are a large number of atanh, $\tanh$ and multiplication operations in the formula, and the computational complexity is extremely high. To reduce the computational complexity, the normalized minimum sum algorithm uses mathematical approximation to simplify the equation $m_{c_n \to v_m}$ into the following equation.

$$m_{c_n \to v_m} = \alpha \times \prod_{v_k \in \mathcal{N}(c_n) \setminus \{v_m\}} \text{sign}(m_{v_k \to c_n}) \times \min_{v_k \in \mathcal{N}(c_n) \setminus \{v_m\}} (|m_{v_k \to c_n}|). \tag{3}$$

The normalization factor α is to compensate for the LLR value calculation error caused by the mathematical approximation. Although it cannot be completely approximated, the decoding effect can be close, while greatly reducing the computational complexity. Therefore, LNMS combines the advantages of the reduced computational complexity of the normalized least sum algorithm and the fast convergence rate of the layered decoding algorithm, and is suitable for scenarios with limited computational resources and time delay sensitivity in the on-board base band.

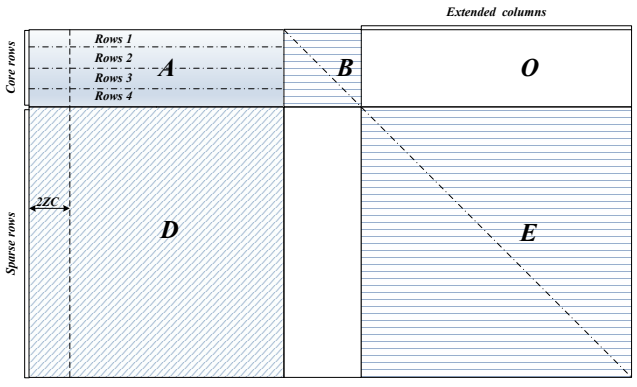

Fig. 1. LDPC BG matrix architecture.

B. LNOMS-SS ALGORITHM

The structure of the BG base diagram is shown in the Fig1. The BG matrix consists of a core matrix (matrix A in the

figure), an identity matrix (matrix E in the figure), and a low-density generative matrix (matrices B, O, D, etc. in the figure). The core matrix has the highest row weight, while the remaining matrix regions are relatively sparse. In order to further improve transmission efficiency, the first two columns of the BG matrix are perforated.

According to the puncturing scheme of NR-LDPC codes in the 3GPP protocol, the code blocks fed into the decoder after block division perform the operation of supplementing 0 of $2Z_C$ length, which makes it possible to have zero, one, or two punctured non-zero elements in each row of the BG base matrix, let O_0, O_1, and O_2 be the row index sets of zero, one, and two punctured elements in the BG matrix, respectively. At the time of layered decoding, if the priority update has two punctured non-zero elements, the calculated minimum sum will all be 0 according to the principle of the minimum sum algorithm, at which point the minimum sum calculation is performed but no substantial check node update is performed, and the later rows cannot take advantage of the newly updated LLR value because the updated LLR value is 0.

So we design an improved three layer static scheduling algorithm for decoding based on the traditional LNMS algorithm. The update order of the first layer is determined by the number of punctured non-zero elements present in each row of the BG base matrix, and the update order of the second layer is determined by the row weights of the BG check rows. The third layer update sequence is random.

Firstly, the first layer static scheduling is performed, and according to the principle of layered decoding algorithm, the layer with fewer punctured non-zero elements is updated first. So the layer i is updated first, $i \in O_0$, and after i is updated, j is updated, $j \in O_1$, and so on. This provides more information on the check node. Then the second layer static scheduling is performed, updating the heavier row layers in BG when the number of punctured non-zero elements is the same, for the same reason as above. Finally, the third layer static scheduling is performed, and when the punctured non-zero elements and BG line weights are both the same, the optimal algorithm should be statistically and sorted to find out which layers in the remaining layers can take advantage of more of the already updated check node information, as this speeds up decoding convergence. But after simulation we found that this kind of algorithm complexity is relatively high, and take a long time, not suitable for the on-board environment. So the third layer static scheduling algorithm designed in this paper will generate the order randomly in the target layer for updating.

The steps of LNOMS-SS algorithm are shown in Alg.1.

Algorithm 1: Proposed LNOMS-SS Algorithm.

Input: Quantized-LLRs $(L) = \{L_1, L_2, L_3, \ldots, L_n\}$;
Output: Decoded bits $(\hat{D}) = \{D_1, D_2, D_3, \ldots, D_n\}$;
Initialization:
$\alpha = [\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_5]$ (α is Normalization Factor);
$\beta = 0.5$ (β is Offset Factor);
$SM - State \rightarrow IDLE$;
for $i = 0$ *to* 67 **do**
  $LLRRAM_i = 0$ (RAMs of Zc LLRs);
  $Addr = [Addr, addr_i]$;
  ($addr_i$ is Layerd Preprocessed LLR address)
end
if $bg = 1, 2$ **then**
  Generate SSM (Static scheduling rows sequence)
  Select α based on BG and ZC
end
Start Decoding iteratively:
for $iteration = 0$ *to* $ite_{max} - 1$ **do**
  $LLRRAM = L$; $PI = 0$;
  $LLRRAM = LLRRAM - PI$;
  for $i = SSM(1)$ *to* $SSM(bgrows)$ **do**
    $R(j) = LLRRAM(addr_i)$;
    for $j = 1$ *to* $len(R(j))$ **do**
      $min1 = \min(|R(j)|$;
      $idx = \zeta(min1, |R(j)|)$;
      $min2 = \min(|R(j)\backslash_{R(j,idx)|})$;
      $Sign = \prod sign(R(j))$;
      $min1 = \max\{0, min1 - \beta\}$;
      $min2 = \max\{0, min2 - \beta\}$;
      PI = $\alpha * Sign * (min1 + min2)$
    end
    $LLRRAM = LLRRAM + PI$
  end
  Check: Improved verification method
  if $iteration = ite_{max} - 1$ *or H' * LLRRAM = 0* **then**
    Break the loop
  end
end
Decision:
$D_i = \begin{cases} 0 & \text{LLRRAM}_i \geq 0 \\ 1 & \text{otherwise} \end{cases}$, $D_i \in \hat{D}$
Output $(\hat{D}) = \{D_1, D_2, D_3, \ldots, D_n\}$;

III. PROPOSED HARDWARE ARCHITECTURE

A. Overall architecture

The overall architecture of our proposed NR-LDPC decoder is shown in the Fig2. The whole decoder contains six main functional units, namely the Central Control Unit (CCU), LLR Ping Pong Buffer Unit (PPBU), Pre-operation Routing Unit (PRU), Check node processing unit (CNU), Variable node processing unit (VNU) and Check and decision unit (CDU).

B. PPBU

Our PPBU was designed as two sets of 68 dual port ram blocks at a depth of 384 and the control logic circuits that controlled them. The depth and number of RAM match the maximum ZC supported by the 3GPP protocol. The benefits of adopting a ping pong design can cache data for successive multiple users, improving decoding efficiency. Using dual port RAM can make the read/write logic independent of each other, and the V2C information output by VNU can be returned to

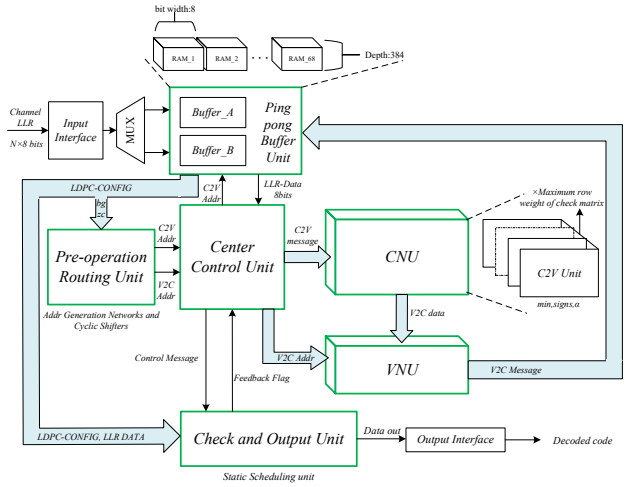

Fig. 2. Proposed LDPC decodeing architecture.

ping pong cache module for immediate processing. Because of the layered decoding characteristics, the different check lines within the layer are strictly orthogonal, so the V2C information update does not affect the readout of C2V information, further increasing the efficiency of decoding.

C. PRU

The function of the routing unit (RU) of traditional LDPC decoders is to adjust the output order of a posterity message. The essence of RU is to continuously shift Zc a posterity message in the LLR cache module to the left circularly, generate an address via the address generation module, and extract the LLR value needed by the check node update. Completing such operations generally requires opening up storage space for a posterity beyond the LLR storage space, and designing a barrel shifter, Banyan or QSN, to do so, with high memory overhead.

To reduce hardware memory overhead for satellite communications scenarios, we designed the transmission bit rate from low to high for a total of 8 gears, and a pre-processing based routing unit was designed for this. According to the definition of the LDPC decoding base matrix by the NR protocol, BG1 and BG2 are expandable matrices of 46 * 68 and 42 * 52, respectively, with 4 rows and 22 columns at the top of BG1 as the core matrix and 4 rows and 10 columns at the top of BG2 as the core matrix. We pre compute the initial address in the Ping Pong cache module corresponding to each layer in the layered decoding of different bit rates and store it in the ROM. This requires only 68 sets of read/write address lines, while the cyclic addition and subtraction operation of the address can be done by simple decision logic without the need for complex shift networks and additional LLR memory space.

Similarly, to implement the improved three layer static scheduling algorithm, we pre compute the decoding priorities of different bGs by simulation and store them in the ROM memory. When a new layer of decoding is turned on, the routing unit extracts the layer serial number in the static scheduling ROM in advance and the corresponding LLR address in the address ROM from that serial number. Because the three layer static scheduling algorithm of the two kinds of bg only needs to store a small number of layer numbers additionally, the added resource consumption is low.

D. CNU & VNU

The role of the CNU is to calculate the Min-sum and complete the C2V information update. We designed the control module CNU-TOP and the CNU sub module designed for different row weights of the check row. Decoding begins when the CCU goes to control the PPBU based on the adder from the PRU output, with the aim of extracting the LLR value and passing it to the CNU-TOP module, which activates the corresponding CNU sub module based on the CONFIG of the input. These sub modules use a parallel tree comparison structure to find the minimum, followed by a min-sum calculation. The advantage of this structure is that the computation latency is greatly reduced compared to the serial structure.

The structure of the CNU and VNU is shown on the left side of the Fig3, and the work flow diagram of the CNU and VNU is shown on the right side.

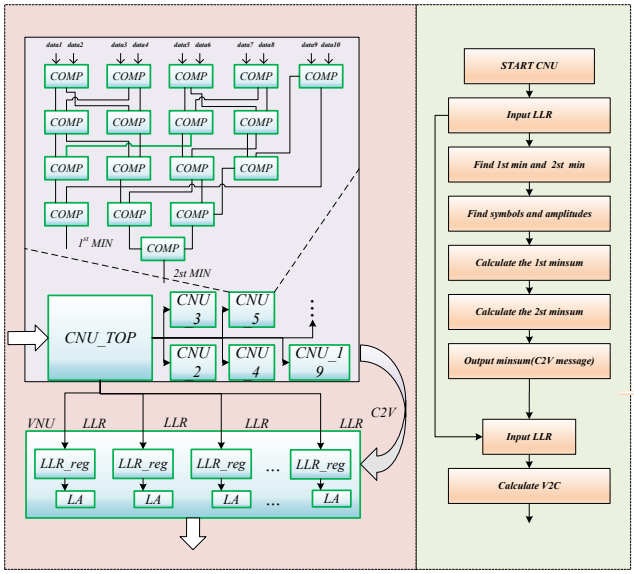

Fig. 3. CNU & VNU

The VNU's role is to complete the calculation of the V2C and transmit it to the PPBU for updates. Due to the existence of the PRU, the VNU structure we designed was simpler, storing the initial LLR while the CNU was working, plus or minus and limiting the LLR value and C2V after the CNU output the C2V information, and sending back the memory unit for update after generating the V2C message.

E. Implementation of Normalized factor

The C2V updating process involves the selection of α and the twice multiplication operation. The selection scheme

of the normalization factor is divided into two schemes, fixed and curve fitting. The latter needs pre-simulation, establishing a large LUT, listing the best normalization factors for different bit rates and modulation coding modes. The former is convenient, but the error rate is not good. Therefore, we combined two ways to calculate a suitable normalization factor based on the 8 bit rate pertinency of our design, and established a LUT in advance.

In the aspect of multiplication optimization of normalized factors, we propose an approximate optimization method because of using multipliers directly with more resources and difficult timing control. That is, the commonly used normalization factors are approximated as 3/8, 4/8, 5/8, 6/8, 7/8. In this way, the multiplication operation can be replaced with the FPGA implementation, namely, the easy addition and shift operation, greatly reducing the resource consumption of the multiplier, and the loss of decoding performance is minimal. For example, assuming that the LLR value as a prior probability is represented in 8-bit binary, the register as an intermediate variable needs to be expanded to 11 bits to prevent the LLR value from overflowing during the operation. From the pre fit curve, alpha closer to 0.375 would be approximated to 0.375, and so on, several different alphas would be written to LUT. The LLR finds the Minsum via the CNU, and is written to an 11 bit temporary register whose operation formula is shown in the following table I.

TABLE I
NORMALIZETION FACTORS SIMPLIFIED APPROXIMATION FORMULA

α	**Simplified approximation formula**
0.375	$((LLR << 1) + LLR) >> 3$
0.5	$(LLR >> LLR)$
0.625	$((LLR << 2) + LLR) >> 3$
0.75	$((LLR << 1) + LLR) >> 2$
0.875	$((LLR << 1 + LLR << 2) + LLR) >> 3$

F. CU

The check unit (CU) is used to determine whether the translated codeword passes the check of the check equation. Due to the sparsity of the LDPC codewords, there are a large number of complementary 0 elements in the LDPC's check matrix, that is, there are many check lines with very small line weights covering very little information bits. To simplify the design of the checkout section and reduce the latency of on-board checkout, we choose to use only a portion of the checkout equation to complete the checkout. The number of degrees of the check nodes corresponding to these check equations should be as large as possible, thus ensuring more information bits are covered. Therefore, the check unit designed in this paper only uses the check equation corresponding to the first 4 lines of the basic diagram.

IV. RESULTS OF SIMULATION AND IMPLEMENTATION

A. Results of Simulation Analysis

In this section, we have designed an FPGA-based LDPC decoder for using the proposed LNOMS-SS algorithm. Table II summarizes key design parameters of our decoder including modulation schemes, modulation orders, code rates, and spectral efficiencies. The LDPC codes for simulation adhere to the 5G standard with an additive white Gaussian noise (AWGN) underlying channel. Taking level 2 in Table II as an example: QPSK modulation is employed with an input block length of 8424, base graph 1 (bg1), and a code rate of 0.975.

Fig.4 compares simulation results of the proposed LNOMS-SS algorithm with conventional LDPC decoding algorithms under these conditions. The benchmark algorithms are NMS with a normalization factor of 0.75 and OMS with an offset factor of 0.5. Each algorithm undergo 15 iterations. Additional simulations evaluate distinct stages of LNOMS-SS static scheduling (e.g., LNOMS-SS layer1 denotes the first scheduling layer). The left subfigure of Fig. 4 presents the bit error rate (BER) versus SNR curves. Results demonstrate that the proposed method achieves an average 0.1 dB coding gain over conventional NMS and OMS in block error rate (BLER) performance, with this advantage becoming more pronounced at higher SNR. The three scheduling layers contribute differently to BLER reduction: layers 1, 2, and 3 account for 20%, 70%, and 10% of improvement respectively. The dominant contribution of the first two layers results from prioritizing rows with higher row weights among those sharing identical punctured non-zero element counts, thereby utilizing more parity information earlier in decoding. The right subfigure of Fig. 4 shows the average iterations versus SNR curves. LNOMS-SS reduces average iterations by approximately 1 compared to OMS and NMS, and by 0.4 compared to LMS in medium-to-high SNR regions.

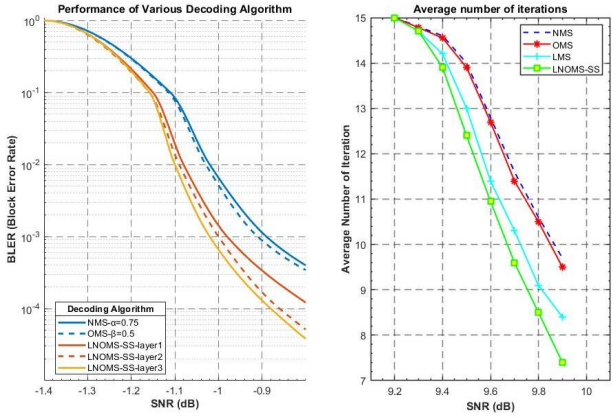

Fig. 4. Performance & Average number of iterations of Various Decoding Algorithm

B. Hardware Complexity Analysis

The resource consumption of several decoders is given in Table III. Our architecture is based on the XC7VX-690T implementation, which has 433200 LUTs, 174200 RAMs, and 866400 FIFOs. From the data in the table, it can be seen that due to the use of a pre-processing based routing unit in this decoder, our designed decoder LUT has improved compared to the architecture in the literature, but the BRAM

TABLE II
EIGHT LEVELS OF DESIGN

$level$	MOD	$Order$	$Rate * 1024$	SE
0	$\pi/2-BPSK$	1	157	0.153
1	$\pi/2-BPSK$	1	555	0.542
2	QPSK	2	499	0.975
3	QPSK	2	696	1.359
4	8PSK	3	500	1.465
5	8PSK	3	600	1.758
6	16APSK	4	642	2.508
7	16APSK	4	741	2.8955

TABLE III
COMPARISON BETWEEN PROPOSED DECODER AND OTHER EXISTING LDPC DECODERS.

	Proposed	**A. Katyushnyj [12]**
Code Length	576−8424	576−8424
Frequency	200MHz	204.8MHz
LUTs	41364	32896
Block RAM&FIFO	28028	24543
Decoding cycles(μs,10 iterations)	2860	3840
Throughput (Mbps)	625	369

and register consumption have been reduced. Due to our improved verification method, the verification time will be significantly reduced, with an average decoding cycle time decrease of 25% after 10 iterations. Therefore, from the perspective of overall latency and implementation complexity, our LNOMS-SS decoder has lower complexity and shorter latency. Meanwhile, our throughput is higher.

V. CONCLUSION

This article proposes the LNOMS-SS algorithm based on the traditional LNOMS algorithm and implements it in hardware. An 8-levels LDPC decoder suitable for low orbit satellite scenarios is designed. The simulation results show that the error performance of LNOMS-SS algorithm is 0.5dB stronger than that of traditional LNOMS algorithm. Although it is not as good as BP algorithm, its advantages are low hardware implementation resource consumption and low complexity. We have designed a pre-processing based routing network for the decoder, specifically designed for satellite communication scenarios to reduce hardware resource consumption; We optimized the calculation process of Minsum and used an approximate method to reduce the use of a large number of multipliers; We have also optimized the verification process, greatly reducing the verification time. The implementation results show that our design reduces the consumption of BRAM and registers, lowers the time required for verification, and improves the overall decoding performance compared to traditional designs. The algorithm and implementation results proposed in this article can provide some reference for the design of LDPC decoders in low orbit satellite scenarios.

ACKNOWLEDGMENT

This work was supported by the National Natural ScienceFoundation of China (NSFC) under the Grant No.62371054.

REFERENCES

[1] J. Nadal and A. Baghdadi, "Parallel and Flexible 5G LDPC Decoder Architecture Targeting FPGA," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 6, pp. 1141-1151, June 2021.

[2] R. G. Gallager, "Low-density parity-check codes," IRE Trans. Inform.Theory, vol. IT-8, pp. 21-28, Jan. 1968.

[3] M. P. C. Fossorier, M. Mihaljevic and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," in IEEE Transactions on Communications, vol. 47, no. 5, pp. 673-680, May 1999.

[4] J. Chen, A. Dholakia, E. Eleftheriou, M.P.C. Fossorier, and X. Hu, "Reduced-complexity decoding of LDPC codes," IEEE Trans. Comms., vol. 53, no. 8, pp. 1288-1299, Aug. 2005.

[5] B. N. Tran-Thi, T. T. Nguyen-Ly, H. N. Hong and T. Hoang, "An Improved Offset Min-Sum LDPC Decoding Algorithm for 5G New Radio," 2021 International Symposium on Electrical and Electronics Engineering (ISEE), Ho Chi Minh, Vietnam, 2021.

[6] Declercq, D., M. Fossorier, and E. Biglieri, Channel Coding: Theory, Algorithms, and Applications: Academic Press Library in Mobile and Wireless Communications. Academic Press. 2014.

[7] Le Trung, K., F. Ghaffari, and D. Declercq., An Adaptation of Min-Sum Decoder for 5G Low-Density Parity-Check Codes. in 2019 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE. 2019.

[8] Y. Liang, C. -T. Lam and B. K. Ng, "A Low-Complexity Neural Normalized Min-Sum LDPC Decoding Algorithm Using Tensor-Train Decomposition," in IEEE Communications Letters, vol. 26, no. 12, pp. 2914-2918, Dec. 2022.

[9] Q. Wang et al., "Normalized Min-Sum Neural Network for LDPC Decoding," in IEEE Transactions on Cognitive Communications and Networking, vol. 9, no. 1, pp. 70-81, Feb. 2023.

[10] D. Oh and K. K. Parhi, "Min-sum decoder architectures with reduced word length for LDPC codes," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 1, pp. 105–115, Jan. 2010.

[11] R. V. Devi and S. Rajaram, "An Efficient FPGA Implementation of LDPC Decoder for 5G New Radio," 2023 International Conference on Recent Advances in Electrical, Electronics, Ubiquitous Communication, and Computational Intelligence (RAEEUCCI), Chennai, India, 2023, pp. 1-5.

[12] A. Katyushnyj, A. Krylov, A. Rashich, C. Zhang and K. Peng, "FPGA implementation of LDPC decoder for 5G NR with parallel layered architecture and adaptive normalization," 2020 IEEE International Conference on Electrical Engineering and Photonics (EExPolytech), St. Petersburg, Russia, 2020, pp. 34-37.

[13] K. Gunnam, G. Choi, M. Yeary, and M. Atiquzzaman, "VLSI architectures for layered decoding for irregular LDPC codes of WiMax," in Proc. IEEE Int. Conf. Commun. (ICC), June 2007, pp. 4542–4547.

[14] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in Proc. IEEE Workshop on Signal Process. Syst. (SiPS), Oct. 2004, pp. 107–112.

[15] K. Zhang, X. Huang and Z. Wang, "High-throughput layered decoder implementation for quasi-cyclic LDPC codes," in IEEE Journal on Selected Areas in Communications, vol. 27, no. 6, pp. 985-994, August 2009.

[16] Y. Sun and J. R. Cavallaro, "VLSI Architecture for Layered Decoding of QC-LDPC Codes With High Circulant Weight," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 10, pp. 1960-1964, Oct. 2013.

[17] K. Tian and H. Wang, "A Novel Base Graph Based Static Scheduling Scheme for Layered Decoding of 5G LDPC Codes," in IEEE Communications Letters, vol. 26, no. 7, pp. 1450-1453, July 2022.

[18] A. I. V. Casado, M. Griot, and R. D. Wesel, "Informed dynamic scheduling for belief-propagation decoding of LDPC codes," in Proc. IEEE Int. Conf. Commun., Jun. 2007.

[19] T. C.-Y. Chang, P.-H. Wang, J.-J. Weng, I.-H. Lee, and Y. T. Su, "Belief-propagation decoding of LDPC codes with variable node–centric dynamic schedules," IEEE Trans. Commun., vol. 69, no. 8, pp. 5014–5027, Aug. 2021.