Received 22 September 2023; revised 6 December 2023 and 17 December 2023; accepted 26 January 2024. Date of publication 6 February 2024; date of current version 8 May 2024.

Digital Object Identifier 10.1109/OJCAS.2024.3363043

GC-Like LDPC Code Construction and Its NN-Aided Decoder Implementation

YU-LUN HSU, LI-WEI LIU (Member, IEEE), YEN-CHIN LIAO (Member, IEEE), AND HSIE-CHIA CHANG (Senior Member, IEEE)

Institute of Electronics, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan
This article was recommended by Associate Editor J. Viraraghavan.

Corresponding author: Y.-L. HSU (e-mail: kimi870928.ee06@nctu.edu.tw)

This work was supported by the Ministry of Science and Technology (MOST) under Grant 111-2622-8-A49-024-SB and Grant 111-2221-E-A49-153.

ABSTRACT The trade-off between decoding performance and hardware costs has been a long-standing challenge in Low-Density Parity Check (LDPC) decoding. Based on model-driven methodology, the Neural Network-Aided Variable Weight Min-Sum (NN-aided vwMS) algorithm is proposed to address this dilemma in this paper. Not only eliminating the second minimum value in the check node update process for reducing hardware complexity, our approach featuring a fast-convergent shuffled scheduling method proposed to enhance convergence speed can also maintain similar decoding performance as compared to the traditional normalized min-sum algorithm. Different from existing model-driven methodologies only suitable for short codes, a Globally-Coupled Like (GC-like) LDPC code construction is presented to enable efficient training with simplified neural networks for longer LDPC codes. To demonstrate the capability of the NN-aided vwMS algorithm with the fast-convergent shuffled scheduling method, a GC-like (9126,8197) LDPC decoder is implemented for NAND flash applications, achieving a 6.56 Gbps throughput with a core area of 0.58 mm^2 under the 40-nm CMOS TSMC process, and average power consumption of 288 mW under the frame error rate of 2.64×10^{-5} at 4.5dB. Our decoder architecture achieves a superior normalized throughput-to-area ratio of 11.31 $Gbps/mm^2$, demonstrating a 2.4x improvement among previous works.

INDEX TERMS ECC, LDPC, decoder, neural network-aided LDPC.

I. INTRODUCTION

LOW-DENSITY Parity Check (LDPC) codes, initially invented by Robert Gallager in the early 1960s [5], are a class of linear block codes that approach the Shannon limit through iterative decoding. Extensive research has been dedicated to their code construction, encoding, and decoding algorithms. LDPC codes are currently one of the most prominent topics in modern coding theory and find applications in various domains, including Ethernet, wireless communications, and solid-state drives.

Over the past few decades, researchers have dedicated considerable effort to finding an algorithm that strikes a balance between hardware complexity and decoding performance. The Pseudo Marginalized Min-Sum (PMMS) Algorithm proposed in [6] combines the first and second minimum values into a single message, effectively reducing the necessary number of interconnections and thereby minimizing hardware complexity. In [7], a novel approximation strategy is proposed that uses only the first minimum in the check update process. This refined approach yields a remarkable reduction in the hardware cost of the sorting unit. However, it's important to note that while these approaches effectively reduce hardware costs, they come at the sacrifice of decoding performance and convergence speed.

Recently, neural networks have gained significant attention from both academia and industry for their ability to learn meaningful features from data, resulting in remarkable real-world applications. This surge in interest extends to the Error Correction Codes (ECC) [8], [9], [10], [11], [12], [13], [14], [15] as well. One such techniques is the integration of neural networks with traditional decoding algorithms, known as model-driven neural networks [16], aimed at enhancing the error correction capabilities of LDPC codes. For example, in [17], the decoding performance of the LDPC decoder is improved by training additive offsets and weights. Alternatively, the approach presented in [18] focuses on training multiplicative correction factors for both the check node and variable node messages.

In contrast to a full neural network decoder, where the entire decoding process is managed exclusively by a neural network without explicit reliance on traditional decoding algorithms, a neural network (NN)-aided decoding algorithm leverages the capabilities of NNs for training parameters used in traditional decoders. Designing such decoders presents a significant challenge because they usually add extra parameters, leading to increased hardware costs, which are crucial factors to consider when designing decoding algorithms. Additionally, current research tends to concentrate on shorter to medium-length LDPC code decoders due to the neural network's complexity scaling with code length. However, it's worth noting that longer LDPC codes typically provide better error correction capabilities.

In this paper, our primary objective is to design a Neural Network-Aided (NN-aided) LDPC decoder. We introduce a novel approach by presenting a Globally-Coupled (GC)-like LDPC code with a more compact structure, maintaining equivalent decoding performance. Compared to the GC-LDPC code proposed in [19], our GC-like LDPC code incorporates not only disjoint local code but also disjoint global code. The special code structure significantly simplifies the neural network, which unleashes code length limitations in training neural networks. Based on this approach, we propose a Neural Network-Aided Variable Weight Min-Sum algorithm (NN-aided vwMS), which estimates the difference between the first minimum and the second minimum values among incoming variable node to check node messages through the neural network. Despite discarding the second minimum, our algorithm achieves decoding performance closely aligned with the normalized Min-Sum algorithm [20]. Furthermore, we improve convergence speed by introducing the fast-convergent shuffled scheduling approach. Hardware implementation results validate that our proposed algorithm effectively balances decoding performance and hardware costs. It's noteworthy that, although we specifically discuss the advantages of the GC-like LDPC code in simplifying the neural network within this paper, the design methodology of the NN-aided decoder is not confined to GC-like LDPC codes alone but also applicable for shorter length LDPC codes.

The remainder of this paper is organized as follows: Section II provides the preliminary concepts. Section III details the GC-like LDPC code construction method and the proposed neural network framework. Section IV elaborates on the shuffled scheduling used in the NN-aided vwMS algorithm and introduces the fast convergence shuffled scheduling. Section V delves into the hardware architecture of the decoder and offers a comparative analysis against the state-of-the-art work. Finally, Section VI concludes this paper.

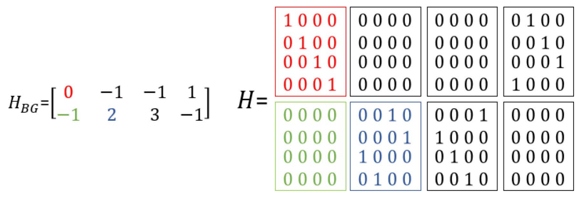


FIGURE 1. An example of QC-LDPC code and its base matrix.

II. BACKGROUND

A. LDPC CODES FUNDAMENTALS

An (n, k) LDPC code is defined by the parity check matrix $\boldsymbol{H}$, a sparse matrix of dimensions $m \times n$. Here, n signifies the code's length, k denotes the information length, and m corresponds to the parity length, expressed as $m = n - k$. LDPC code can also be visually represented as a bipartite graph known as Tanner graph [21]. In Tanner graph, a cycle is defined as a group of interconnected edges that start and end at the same node. A well-constructed parity check matrix adheres to the Row-column (RC) constraint, ensuring that no two columns (or rows) share more than one non-zero element.

Quasi-cyclic (QC) LDPC codes typically consist of circulant permutation matrices (CPM) arranged in an $M \times N$ configuration, with each CPM sized $a \times a$. The base matrix is a simplified representation of the QC-LDPC parity check matrix, where each element in the matrix corresponds to a shift coefficient. The parity check matrix and the base matrix are illustrated in Figure 1, where '2' indicates cyclically shifting the CPM two columns to the right, '0' represents the identity matrix, and '-1' signifies an all-zero matrix.

B. VARIABLE WEIGHT MIN-SUM ALGORITHM

Normalized Min-Sum (NMS) algorithm [20] is a simplified approximation of Sum-Product algorithm (SPA). With a properly chosen normalization factor β ($\beta \leq 1$), it is capable of achieving a performance similar to the SPA. For further complexity reduction in the check node update process, the Variable Weight Min-Sum(vwMS) algorithm, as proposed in [22], adopts iteration-dependent weight factors denoted as $w^{(t)}$. This method involves an analysis of the discrepancy between the 1^{st} minimum magnitude and the 2^{nd} minimum magnitude of V2C, allowing us to derive the most probable weight factor for each decoding iteration. Algorithm 1 provides a detailed description of vwMS, and the symbol definitions are provided in Table 1.

C. NEURAL NETWORK-AIDED (NN-AIDED) LDPC DECODER

The utilization of model-driven neural networks for the design of message-passing LDPC decoders, structured according to the edge connections in the Tanner graph derived from the parity check matrix, is an alternative

TABLE 1. Symbol definitions.

Symbol Notation	Description
$L_{v_j \to c_i}$	the message sented from the j^{th} variable node to the i^{th} check node
$L_{c_i \to v_j}$	the message sented from the i^{th} check node to the j^{th} variable node
L_{ch_j}	the channel value corresponding to the j^{th} variable node
L_{v_j}	a posteriori log-likelihood ratio (LLR) of the j^{th} variable node.
x_j	the j^{th} estimated bit
$N(i)$	the neighboring check/variable of the i^{th} variable/check node
t	the iteration counter
T_{max}	the maximum number of iterations
S	the vector of the syndrome, s_i denotes the i^{th} syndrome
$T_i^{(t)}$	the number of $L_{v_j \to c_i}$ that is equal to $M_i^{(t)}$
$M_i^{(t)}$	the absolute minimum of the i^{th} check node in the t^{th} iteration
C2V	the check node to variable node message
V2C	the variable node to check node message

Algorithm 1 Variable Weight Min-Sum (vwMS) Algorithm

```
1:  for $j = 0$ to $n-1$ do                                  ▷ Initialization
2:      $L^{(0)}_{v_j \to c_i} = L_{ch_j}$.
3:  end for
4:  for t=1, ..., $T_{max}$ do
5:      for $i = 0$ to $m-1$ do                              ▷ Check Node Update
6:          if $T_i^{(t)} = 1$ and $L_{v_j \to c_i} = M_i^{(t)}$ then
7:              $Mag^{(t)}_{c_i \to v_j} = M_i^{(t)} + w^{(t)}$
8:          else
9:              $M_i^{(t)}$
10:         end if
11:     end for
12:     for $j = 0$ to $n-1$ do                              ▷ Variable Node Update
13:         $L^{(t)}_{v_j \to c_i} = L_{ch_j} + \sum_{i' \in N(j) \setminus i} L^{(t)}_{c_{i'} \to v_j}$.
14:         $L^{(t)}_{v_j} = L_{ch_j} + \sum_{i' \in N(j)} L^{(t)}_{c_{i'} \to v_j}$
15:         if $L^{(t)}_{v_j} < 0$ then
16:             $x_j = 1$
17:         else
18:             $x_j = 0$
19:         end if
20:     end for
21:     for $i = 0$ to $m-1$ do                              ▷ Syndrome Check
22:         $S_i = \bigoplus_{j \in N(i)} x_j$
23:     end for
24:     Terminate the decoding process if ($S == \mathbf{0}$)
25: end for
```

graphical representation to the message-passing algorithm. In this paradigm, the edges in the Tanner graph are represented as nodes within hidden layers. The neural network comprises the input layer, the hidden layers, and the output layer. The input layer receives channel output, the output layer determines the decoding results, and the hidden layers facilitate message propagation. The dimensions of the input and the output layers are N, corresponding to the number of variable nodes in the Tanner graph. The hidden layers have dimensions E, which is equivalent to the number of edges in the Tanner graph. An example of such a neural network that unfolds two iterations of the prototype decoder, is in Figure 2. The connection from the input layer to the first hidden layer can be thought of as distributing channel

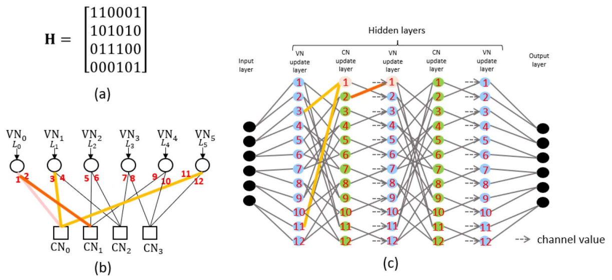


FIGURE 2. (a) Parity check matrix. (b) Tanner graph. (c) An example of the corresponding neural network.

values to the edges of V2C. The subsequent CN update layer can be understood as follows: Assuming the 0th check node needs to update the edge 1, it requires V2C from edges 3 and 11. The next VN update layer can be viewed in this way: Assuming the 0th variable node needs to update edge 1, it requires C2V from edge 2. As the steps continue, a complete neural network can be constructed.

D. GLOBALLY-COUPLED (GC) LDPC CODE

Globally-Coupled Low-Density Parity Check (GC-LDPC) code [19] consists of local codes $\boldsymbol{H}_{Local}$ and global codes $\boldsymbol{H}_{Global}$. $\boldsymbol{H}_{Local}$ comprises s disjoint submatrices positioned along the main diagonal of the upper $s \times s$ submatrices within $\boldsymbol{H}_{GC}$. The remaining sections of the $s \times s$ submatrices within $\boldsymbol{H}_{GC}$ are all-zero matrices. $\boldsymbol{H}_{Gobal}$ resides in the lower section of $\boldsymbol{H}_{GC}$ and serves to connect the local Tanner graphs for exchanging messages between disjoint local codes. The decoding of the received codeword can be achieved partially using disjoint local codes. In case of decoding failure, the global code helps to exchange messages among local codes. The base matrix of the GC-LDPC code is shown in the Equation (1).

$$\boldsymbol{H}_{GC} = \begin{bmatrix} \boldsymbol{H}_{Local,1} & & \\ & \ddots & \\ & & \boldsymbol{H}_{Local,s} \\ \cdots & \cdots & \cdots \\ & \boldsymbol{H}_{Global} & \end{bmatrix} \quad (1)$$

III. PROPOSED GC-LIKE LDPC CODE CONSTRUCTION AND NN-AIDED VWMS ALGORITHM

Previous works primarily focused on short LDPC codes because neural networks are memory-intensive, and the memory requirements for training NN-aided LDPC codes increase quadratically with the number of edges in the Tanner graph. Since the neural network model is not fully connected, where connections between layers are sparse, the connection matrices between layers are used to specify the connectivity pattern. For an LDPC code with E edges in the Tanner graph, the size of the connection matrix of input layer to the first hidden layer and the last hidden layer to the output layer is $N \times E$. The connection matrix between hidden layers are $E \times E$. Therefore, a neural network designed for T^{max}

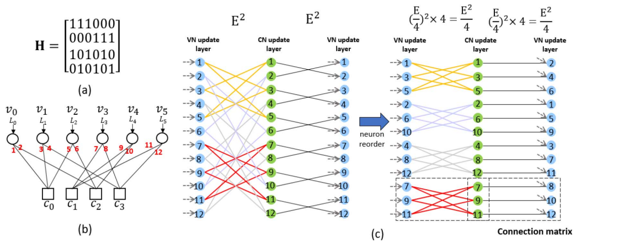


FIGURE 3. (a) Parity check matrix. (b) Tanner graph. (c) An example of edge splitting.

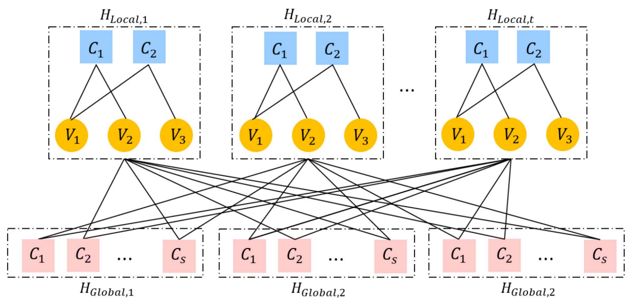


FIGURE 4. Tanner graph of the proposed GC-like LDPC code.

iterations the total size is $2 \times N \times E + T^{max} \times 2 \times E \times E$ for the connection matrix. As code length increases, these edge-specific neural network decoders become impractical because the number of edges grows rapidly.

A. CONSTRUCTION OF GC-LIKE LDPC CODE

In response to the demands of neural networks for long LDPC codes, we aim to simplify the neural network by dividing its edges. In Figure 3, the original neural network is divided into 4 groups; thus, the total size of the connection matrix is reduced from E^2 to $\frac{E^2}{4}$. The methodology highly relies on the LDPC code structure that check nodes and variable nodes should be able to partition into disjoint sets, and this reminds us of the GC-LDPC code.

Several methods have been proposed for the construction of GC-LDPC codes [23], [24]. For example, an algebraic GC-LDPC code can be constructed by the Latin square with superposition. To enable edge splitting in a neural network based on the GC-LDPC code, we construct not only the disjoint local codes but also the disjoint global codes, which means the tanner graph of the row blocks of local code or the global code are not connected between each other. The Tanner graph of the proposed GC-like LDPC code is presented in Figure 4.

The construction of the proposed GC-like LDPC code comprises the generation of the base matrix and the formation of local and global code. Algorithm 2 details the steps to construct the proposed GC-like LDPC codes.

B. PROPOSED NN-AIDED VWMS ALGORITHM

The weight factors utilized in the vwMS algorithm have a significant influence on decoding performance. In the work of [22], weight factors were determined through an exhaustive search. Nonetheless, this method is time-consuming due

Algorithm 2 Construction of the GC-Like LDPC Code

Step I:
The $\boldsymbol{H}_{Local}$ is a QC-LDPC code with l row blocks and k column blocks. The CPM size is denoted as $Z_c \times Z_c$. The shift coefficients are generated by the computer search method and the RC constraint is satisfied. The local code of the GC-LDPC code consists of s copies of $\boldsymbol{H}_{Local}$ on the diagonal.

Step II:
The first row block of $\boldsymbol{H}_{Global}$ is constructed by inserting $s-1$ columns of all-zero submatrices between adjacent column blocks of $\boldsymbol{H}_{Local}$.

Step III:
The remaining $s-1$ row blocks of $\boldsymbol{H}_{Global}$ are constructed by cyclically shifting the row block constructed in Step II by one column block.

Step IV:
Check the RC constraint for the constructed GC-LDPC code. If RC constraint is violated, return to Step I and re-randomize the shift coefficients. Otherwise, construction is done. With $s = 3$, the yellow shaded regions denote non-zero matrices, while the white shaded portions signify all-zero matrices. The entire construction procedure is illustrated in Figure 5.

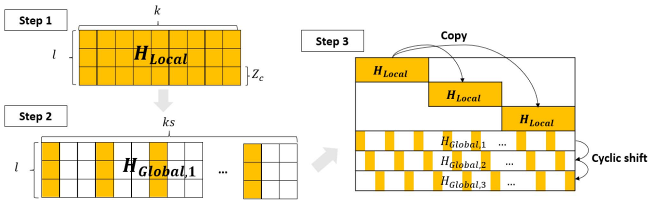


FIGURE 5. The construction flow of the proposed GC-like LDPC code.

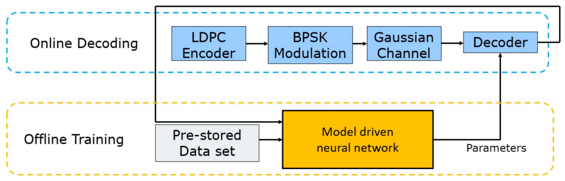


FIGURE 6. Block diagram of neural network-optimized low-resolution decoder.

to the need to assess all possible combinations of weight factors. To overcome this, we propose a novel approach that uses a model-driven neural network to efficiently train optimal weight factors.

The Neural Network-Optimized Low-Resolution Decoder (NOLD), proposed in [25], introduces a structured framework comprising both offline training stages and online decoding stages. As shown in Figure 6, the online decoding stages behave like a typical decoder, applying parameters trained in the offline stages and forwarding the V2C and C2V to the offline training stages; the offline training unfolds the decoding algorithm for a single iteration. Assuming we are training parameters for the t^{th} iteration, the process begins with a pre-stored dataset containing noisy codewords derived from specific channel conditions. The C2V and V2C are

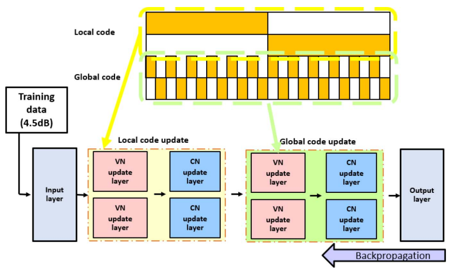


FIGURE 7. **Training diagram of the proposed NN-aided vwMS algorithm.**

obtained from online decoding stages, which has already decoded for $t-1$ iterations. Training continues until the specified maximum iteration count is reached.

The target code is a (9216,8197) regular GC-like LDPC code with a column degree of 4 and a CPM size of 128×128. Since the target code consists of disjoint local and global codes, each local and global update layer can be split into two disjoint parts. The decoder in the online decoding stages is a vwMS decoder, which adopts weight factors trained in the offline training stage. The block diagram of the neural network during the offline training stage is illustrated in Figure 7.

The activation functions for each layer are shown below, and the training goal is to derive the weight factors denoted w for each decoding iteration and minimize the loss.

- **Input Layer:** The j^{th} node in the first layer receives the j^{th} channel value, where $j = 1, 2, 3 \cdots N$.

$$L_j^{in} = L_{ch_j} \tag{2}$$

- **Local Variable Nodes Update Layer:** The local V2C $L_{v_j\to c_i}^{local}$ is updated with the channel value L_{ch_j} and the local C2V $L_{c_{i'}\to v_j}^{local}$ and global C2V $L_{c_{i'}\to v_j}^{global}$.

$$L_{v_j\to c_i}^{local} = L_{ch_j} + \sum_{i'\in N(j)\backslash i} L_{c_{i'}\to v_j}^{local} + \sum_{i'\in N(j)\backslash i} L_{c_{i'}\to v_j}^{global} \tag{3}$$

- **Local Check Nodes Update Layer:** The local C2V $L_{c_i\to v_j}^{local}$ is updated with the incoming V2C $L_{v_{j'}\to c_i}$. The j_{min} is the index of the smallest $L_{v_{j'}\to c_i}$. The weight factor w is the training parameter:

$$L_{c_i\to v_j}^{local} = 0.75 \times \prod_{j'\in N(i)\backslash j} sign(L_{v_{j'}\to c_i}) \times \begin{cases} \min\limits_{j'\in N(i)} (|L_{v_{j'}\to c_i}|) + w, & \text{if } j = j_{min} \\ \min\limits_{j'\in N(i)} (|L_{v_{j'}\to c_i}|), & \text{otherwise.} \end{cases} \tag{4}$$

- **Global Variable Nodes Update Layer:** The global V2C $L_{v_j\to c_i}^{global}$ is updated with the channel value L_{ch_j} and the local C2V $L_{c_{i'}\to v_j}^{local}$ and global C2V $L_{c_{i'}\to v_j}^{global}$.

$$L_{v_j\to c_i}^{global} = L_{ch_j} + \sum_{i'\in N(j)\backslash i} L_{c_{i'}\to v_j}^{local} + \sum_{i'\in N(j)\backslash i} L_{c_{i'}\to v_j}^{global} \tag{5}$$

- **Global Check Nodes Update Layer:** The global C2V $L_{c_i\to v_j}^{global}$ is updated with the incoming V2C $L_{v_{j'}\to c_i}$. The j_{min} is the index of the smallest $L_{v_{j'}\to c_i}$. The weight factor w is the training parameter:

$$L_{c_i\to v_j}^{global} = 0.75 \times \prod_{j'\in N(i)\backslash j} sign(L_{v_{j'}\to c_i}) \times \begin{cases} \min\limits_{j'\in N(i)} (|L_{v_{j'}\to c_i}|) + w, & \text{if } j = j_{min} \\ \min\limits_{j'\in N(i)} (|L_{v_{j'}\to c_i}|), & \text{otherwise.} \end{cases} \tag{6}$$

- **Output Layer:** For the output layer, the j^{th} variable node produces a decision based on

$$L_{v_j} = L_{ch_j} + \sum_{i'\in N(j)} L_{c_{i'}\to v_j}. \tag{7}$$

The soft-sign function is used for bit estimation because the derivative of the sign function is 0 except for 0, which could pose problems during back-propagation.

$$L_j^{output} = \frac{L_{v_j}}{L_{v_j} + 0.01} \tag{8}$$

$$Loss = \sum_{j=0}^{N} \left(L_j^{output} - c_j\right)^2 \tag{9}$$

The loss function can be considered as a criterion to estimate the performance of neural network models. Most neural network models adopt the binary cross-entropy function as the loss function; however, it can only be seen as the approximation of the bit error rate [26]. To exploit the soft bit information, we use the square error that measures the distance between the result L_j^{output} and the code bit c_j as a loss function to train the proposed neural network.

TABLE 2. **Quantized weight factors *w*.**

Iteration	1	2	3	4	5	6
Parameter	1.25	1.25	1.5	2	1.75	2
Iteration	7	8	9	10	11	12
Parameter	1.75	1.75	1.75	1.75	1	1.5

C. NUMERICAL RESULTS

The neural network is built with TensorFlow, and Adaptive Moment Estimation (ADAM) [27] with a learning rate equal to 0.01 is used to optimize the neural network. There are 64000 codewords in the pre-stored data set that are modulated by BPSK and distorted by AWGN generated in 4.7 dB. The batch size is 64.

The training results of weight factors, quantized into 2 integer bits and 2 fractional bits, are shown in Table 2. The validation loss barely decreases after 12 iterations, indicating that a convergence of the decoder is achieved within 12 iterations. The weight factors denoted $w^{(t)}$, where $t = 1, 2, 3, \ldots, 12$, are used for the iteration 1st to 12th. Subsequently, for decoding iterations beyond 12th, $w^{(12)}$ is used.

Figure 8 shows the decoding performance of the NMS decoding algorithm, the NN-aided vwMS algorithm and the

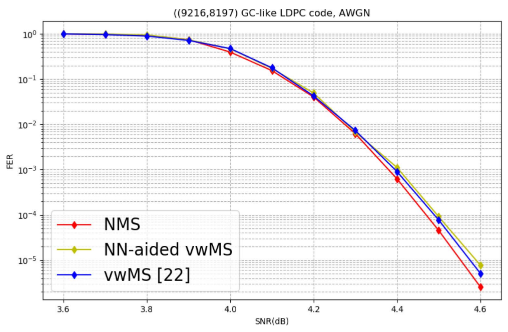


FIGURE 8. **The FER performance of the (9216,8197) GC-like LDPC code using the NMS algorithm and the NN-aided vwMS algorithm over AWGN channel.**

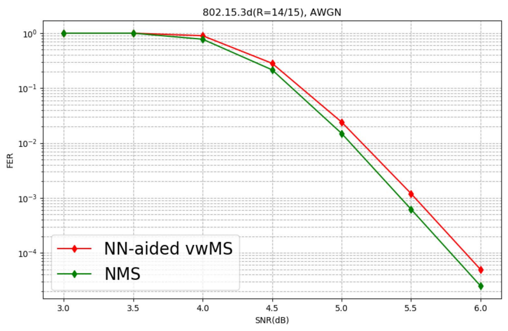


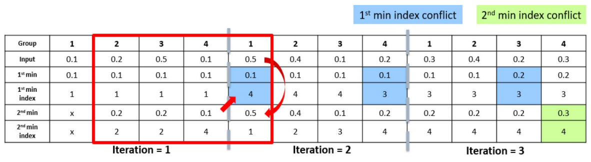
FIGURE 9. **The FER performance of the LDPC code designed for IEEE 802.15.3d using the NMS algorithm and the NN-aided vwMS algorithm over AWGN channel.**

vwMS algorithm proposed in [22]. The scheduling methods of these three algorithms are flooding and T^{max} is 50. While the vwMS algorithm demonstrates comparable performance, the efficiency of an exhaustive search for optimal weight factors is challenging when more accurate numerical values are required. This results in an increased search space, making the exhaustive search less efficient.

To enhance the robustness of our training framework, we conducted experiments focusing on an LDPC code designed for IEEE 802.15.3d [28], featuring a code rate of 14/15 and a code length of 1440. As shown in Figure 9, the decoding performance of the NMS decoding algorithm and the NN-aided vwMS algorithm is closed under flooding scheduling, with T^{max} set at 50. This result highlights the effectiveness of the proposed training methodology.

IV. NN-AIDED VWMS ALGORITHM WITH FAST CONVERGENCE

Column-based shuffled scheduling is a hardware-friendly implementation of the LDPC decoder, and the ordered set is proposed to simplify the complexity of the decoder [29]. By carefully considering the replacement strategy, it's feasible to maintain comparable decoding performance with only the two smallest V2C. This replacement should occur not only when the magnitude of the input V2C is smaller than one of the magnitude of the 1^{st} and 2^{nd} minimum, but also when there's an index conflict that arises when the index of the 1^{st} minimum or the 2^{nd} minimum equals the index of the current processed variable node. Figure 10 illustrates an example of


1st min index conflict
2nd min index conflict
Iteration = 1
Iteration = 2
Iteration = 3

Group	1	2	3	4	1	2	3	4	1	2	3	4
Input	0.1	0.2	0.5	0.1	0.5	0.4	0.1	0.2	0.3	0.4	0.2	0.3
1st min	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2
1st min index	1	1	1	1	4	4	4	3	3	3	3	3
2nd min	x	0.2	0.2	0.1	0.5	0.4	0.1	0.2	0.2	0.2	0.2	0.3
2nd min index	x	2	2	4	1	2	3	4	4	4	4	4

FIGURE 10. **An example of the 1^{st} minimum and the 2^{nd} minimum conflict in the shuffled NMS algorithm.**

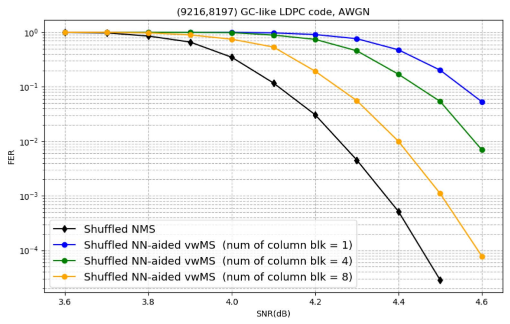


FIGURE 11. **The FER performance of the (9216,8197) GC-like LDPC code using shuffled NN-aided vwMS and shuffled NMS over AWGN channel.**

the index conflict in the shuffled NMS algorithm. In this example, the 1^{st} minimum is outdated in the 1^{st} group of the 2^{nd} iteration, since the magnitude of the input V2C is larger than the magnitude of the 1^{st} minimum. The 1^{st} minimum is replaced by the 2^{nd} minimum.

For the NN-aided vwMS algorithm with shuffled scheduling, the magnitude of the input V2C directly replaces the magnitude of the 1^{st} minimum when a conflict arises, as there is no stored information about 2^{nd} minimum. However, as illustrated in Figure 11, increasing the number of column blocks updated in a single sub-iteration does enhance the Frame Error Rate (FER), the overall performance remains undesirable. This is primarily because the pseudo 1^{st} minimum, obtained by sorting the magnitude of the incoming V2C to replace the outdated 1^{st} minimum in case of a conflict, is often larger than the actual 1^{st} minimum.

To ensure the decoding performance, two scenarios need to be taken into consideration when determining the replacement value for the 1^{st} minimum during a conflict:

1) The replacement value shouldn't be too large, leading to performance degradation due to erroneous C2V.
2) The replacement value shouldn't be too small, risking the chance of missing the opportunity to replace the 1^{st} minimum in successively sub-iterations.

The proposed method compares the magnitude of V2C with the magnitude of the outdated 1^{st} minimum plus an *increment* in the event of a conflict. It then updates the 1^{st} minimum with the smaller one. Taking into account the aforementioned considerations, Figure 12 shows the performance on various *increment* values ranging from 1 to 3.5. The best performance is observed when the *increment* is set to 1, which is subsequently adopted.

In addition to the decoding performance, we also examine the average number of iterations, which is detailed in

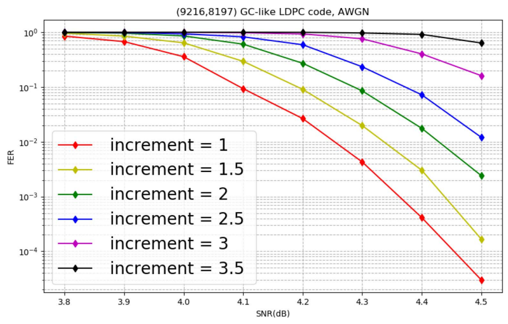
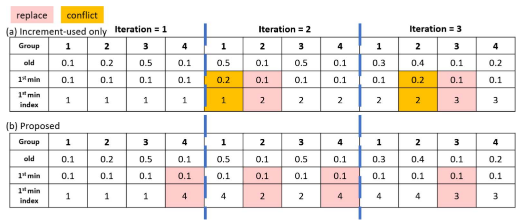


FIGURE 12. The FER performance of the proposed (9216,8197) GC-like LDPC code with the increment between 1 and 3.5.

Algorithm 3 The 1^{st} Minimum Update Method of Proposed Fast-Convergent Shuffled Scheduling

```
    Input: V2C magnitude, VN index, old $1^{st}$ min magnitude, old $1^{st}$ min index
    Output: new $1^{st}$ min magnitude, new $1^{st}$ min index
 1: if VN index == old $1^{st}$ min index then
 2:     if V2C magnitude $>$ old $1^{st}$ min magnitude $+$ increment then
 3:         new $1^{st}$ min magnitude $=$ old $1^{st}$ min magnitude $+$ increment
 4:         new $1^{st}$ min index $=$ old $1^{st}$ min index
 5:     else
 6:         new $1^{st}$ min magnitude $=$ V2C magnitude
 7:         new $1^{st}$ min index $=$ VN index
 8:     end if
 9: else if V2C magnitude $<=$ old $1^{st}$ min magnitude then
10:     new $1^{st}$ min magnitude $=$ V2C magnitude
11:     new $1^{st}$ min index $=$ VN index
12: else
13:     new $1^{st}$ min magnitude $=$ old $1^{st}$ min magnitude
14:     new $1^{st}$ min index $=$ old $1^{st}$ min index
15: end if
```


replace conflict

(a) Increment-used only

	Iteration = 1				Iteration = 2				Iteration = 3			
Group	1	2	3	4	1	2	3	4	1	2	3	4
old	0.1	0.2	0.5	0.1	0.5	0.1	0.5	0.1	0.3	0.4	0.1	0.2
1st min	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.1	0.2	0.1	0.1
1st min index	1	1	1	1	1	2	2	2	2	2	3	3

(b) Proposed

Group	1	2	3	4	1	2	3	4	1	2	3	4
old	0.1	0.2	0.5	0.1	0.5	0.1	0.5	0.1	0.3	0.4	0.1	0.2
1st min	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
1st min index	1	1	1	4	4	2	2	4	4	4	3	3

FIGURE 13. Illustration of the 1^{st} minimum update of NN-aided vwMS with proposed fast-convergent shuffled scheduling.

Table 3. We observe that the NN-aided vwMS algorithm with increment-used shuffled scheduling converges slower than the shuffled NMS. To enhance convergence speed, we have implemented a strategy to reduce conflict frequency. Specifically, we keep track of the most recent 1^{st} minimum index, updating it when the input V2C equals the 1^{st} minimum as outlined in line 9 of Algorithm 3. The algorithmic details are provided in Algorithm 3. In this algorithm, 'old' refers to the value before an update, and 'new' refers to the value after the update.

Figure 13 is an example of Algorithm 3. Through frequent replacement of the 1^{st} minimum index, conflicts are postponed or potentially precluded, thereby enhancing the algorithm's convergence speed.

TABLE 3. The comparison table of the average number of iterations.

SNR(dB)	4.1	4.2	4.3	4.4	4.5	4.6
Shuffled NMS	15.26	10.53	7.89	6.71	6.02	5.54
Increment-used	18.27	12.56	9.30	7.69	6.77	6.17
Proposed	16.75	10.92	8.33	6.91	6.10	5.56

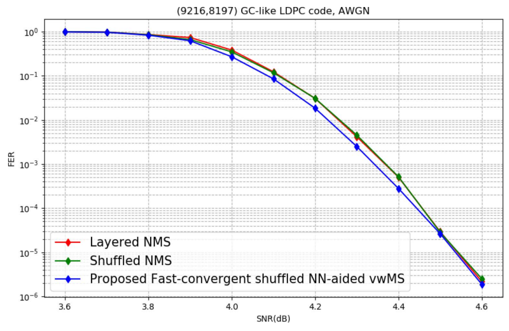


FIGURE 14. The FER performance of the (9216,8197) GC-like LDPC code using layered NMS, shuffled NMS and proposed fast-convergent shuffled NN-aided vwMS over AWGN channel.

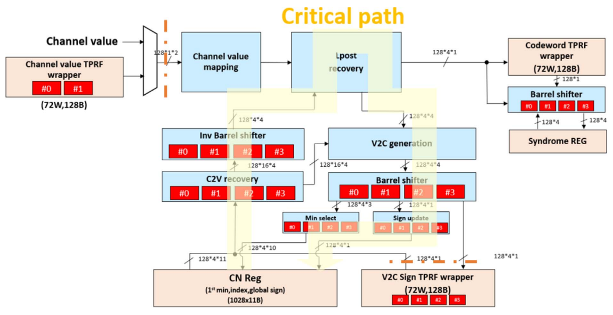


FIGURE 15. Proposed decoder architecture.

The FER performance over AWGN channel using BPSK signaling and decoded with 50 iterations of the layered NMS, shuffled NMS and the proposed fast-convergent shuffled NN-aided vwMS algorithm is shown in Figure 14. The average decoding iteration using the proposed fast-convergent NN-aided vwMS is detailed in Table 3. The proposed algorithm demonstrates a 9.9% improvement in convergence speed at an SNR of 4.5 dB.

V. DECODER ARCHITECTURE AND IMPLEMENTATION RESULTS

A. DECODER ARCHITECTURE

The top-level architecture is shown in Figure 15. The critical path starts and ends at the registers containing check node information. Processing of a column block containing four non-zero CPMs, the requirement is 4×128 check node information to recover the C2V and a posterior probability. In each cycle, the V2C generation units sum the channel value and the C2V to compute the V2C. Subsequently, the 1^{st} minimum and the global sign will be updated and stored in the check node information registers.

TABLE 4. Comparison table of the 1K-Byte LDPC decoders.

	This work	C Zhang [26]	KC Ho [27]	MH Chung [28]	H Zhong [29]
Code (N,K)	LDPC (9216, 8197)	LDPC (9216,8192)	LDPC (9376,8204)	LDPC (9280,8315)	LDPC (9376,8192)
Coderate	0.889	0.889	0.875	0.896	0.889
Scheduling	Shuffled	Layered	Shuffled	Flooding	Shuffled
Iteration	Avg. 6.18	Max. 4	Avg. 8	Max. 4	Max. 16
Status	Post-Layout	Synthesis	Synthesis	Synthesis	Synthesis
Process	40 nm	28 nm	90 nm	90 nm	65 nm
Frequency (MHz)	313	126 $\sim$ 272	200	138.8	300
Area (mm^2)	0.58	0.26	1.4	2.52	2.32
[a] **Throughput (Gbps)**	6.56	1.59	0.52	0.393	2.1
[b] **TAR (Gbps/mm^2)**	11.31	6.11	0.37	0.16	1.1
[c] **NTAR (Gbps/mm^2)**	11.31	2.1	4.2	1.8	4.7
Power (mW)	288	50.5	N/A	N/A	N/A

[a] Throughput = $\frac{CodewordLength}{(AverageCycleCount \times ClockPeriod)}$

[b] TAR = $\frac{Throughput}{area}$

[c] NTAR = $\frac{Throughput}{Area} \times (\frac{Process}{40})^3$

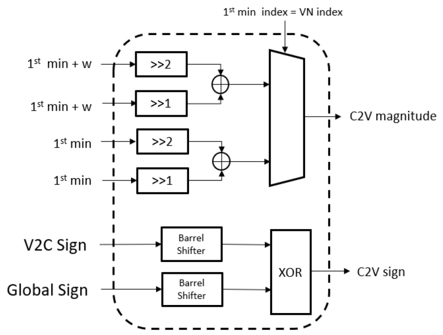


FIGURE 16. Check node to variable node messages recovery unit.

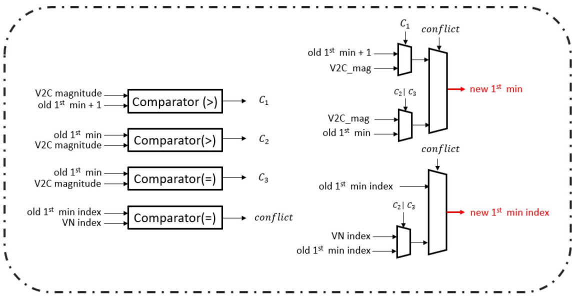


FIGURE 17. Architecture of the minimum selection unit of the proposed decoder.

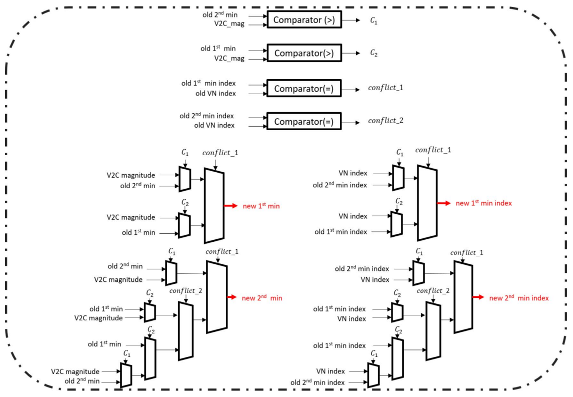


FIGURE 18. Architecture of the minimum selection unit of shuffled NMS decoders.

B. CHECK NODE UNIT

The check node unit consists of two parts: the C2V recovery unit and the minimum selection unit.

The architecture of the C2V recovery unit is shown in Figure 16. Multiplication of the scaling factor 0.75 is achieved through shift and addition operations. Additionally, the weight factor w is added to the magnitude of the 1st minimum. Compared to the shuffled NMS decoder, the proposed decoder requires a higher hardware cost for this module due to the requirement of additional computations to determine the magnitude of the 2nd minimum.

The hardware cost of the check node registers is reduced by 48% compared to that of the shuffled NMS decoder. This arises from the fact that the shuffled NMS decoder stores the magnitude of the 1st and the 2nd minimum, the index of the 1st and 2nd minimum, and the global signs. In contrast, the proposed decoder solely requires storage for the magnitude of the 1st minimum, the magnitude of the 1st minimum index, and the global signs.

The architecture of the minimum selection unit is shown in Figure 17. Compared to the same unit in the shuffled NMS decoders, as illustrated in Figure 18, the number of multiplexers in this work has been reduced from 18 to 5, resulting in a hardware cost reduction of 46%.

C. IMPLEMENTATION RESULTS AND COMPARISONS

The synthesis results under TSMC 40-nm SS corner (125°C, 0.81V) are presented in Table 5. Benefiting from eliminating the need for the 2nd minimum, the clock period in this work has been reduced to 2.5 ns. Furthermore, the total gate counts are reduced by 17 %. Although the average iteration of this work is 0.08 higher than that of the shuffled NMS algorithm at an SNR of 4.5 dB, the throughput still outperforms the shuffled NMS algorithm by 6.5% due to the shortened critical path.

The implementation results of the proposed (9216, 8197) GC-like LDPC code decoder, along with the state-of-the-art 1K-Byte LDPC codes, are presented in Table 4. Operating at

TABLE 5. Implementation result of proposed decoder and shuffled NMS decoder.

	Proposed	Shuffled NMS
Status	Synthesis	
Process (nm)	40	
LDPC (N,K)	(9216,8197)	
Base Graph	(8,72)	
CPM Size	128	
Clock Period (ns)	2.5	2.7
Frequency (MHz)	400	357
Average Iterations	6.10	6.02
Total Area (mm^2)	0.40	0.48
Total Gate Count(K)	582	706
Throughput (Gbps)	8.39	7.88

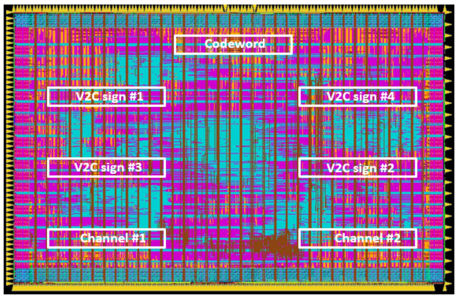


FIGURE 19. Layout of the proposed decoder under TSMC 40 nm process.

a frequency of 313MHz, our decoder achieves a throughput of 6.56 Gbps at an SNR of 4.5 dB. The average power consumption is 288 mW measured by Synopsys Prime Time. Considering the variations in process technologies in the designs, we apply the normalized throughput-to-area ratio (NTAR) [30] as performance normalization. The normalized results are presented in Table 4. Because the NN-aided vwMS algorithm is a simplified version of the NMS algorithm, and the fast-convergent shuffled scheduling helps accelerate the convergence speed, this work achieves the highest NTAR compared to other works.

VI. CONCLUSION

In this paper, an NN-aided vwMS algorithm with fast-convergent shuffled scheduling is proposed. The hardware costs of the proposed decoder are significantly reduced by utilizing the weight factors obtained from the neural network. In addition, a new GC-like LDPC code construction is proposed to address the challenges faced by model-driven neural networks. Implemented under the 40 nm CMOS TSMC process, the proposed (9216,8197) LDPC decoder achieves a throughput of 6.48 Gbps at a clock rate of 313 MHz, with a core area of 0.58 mm^2 and an average power consumption of 288 mW. We believe that our approach, embodied by the NN-aided vwMS algorithm and the GC-like LDPC code construction, has the potential to open new avenues for achieving greater efficiency in future LDPC decoder designs.

REFERENCES

[1] C. Zhang, J. Mo, Z. Lian, and W. He, "High energy-efficient LDPC decoder with AVFS system for NAND flash memory," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2021, pp. 1–4.

[2] K.-C. Ho, P.-C. Fang, H.-P. Li, C.-Y. M. Wang, and H.-C. Chang, "A 45nm 6b/cell charge-trapping flash memory using LDPC-based ECC and drift-immune soft-sensing engine," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Techn. Papers*, 2013, pp. 222–223.

[3] M.-H. Chung, Y.-M. Lin, C.-Z. Zhan, and A. Y. Wu, "Cost-effective scalable QC-LDPC decoder designs for non-volatile memory systems," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 2625–2628.

[4] H. Zhong, W. Xu, N. Xie, and T. Zhang, "Area-efficient min-sum decoder design for high-rate quasi-cyclic low-density parity-check codes in magnetic recording," *IEEE Trans. Mag.*, vol. 43, no. 12, pp. 4117–4122, Dec. 2007.

[5] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[6] H. L. Davila, T.-H. Wu, S.-J. J. Jou, S.-G. Chen, and P.-Y. Tsai, "Low routing complexity multiframe pipelined LDPC decoder based on a novel pseudo marginalized min-sum algorithm for high throughput applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 31, no. 1, pp. 29–42, Jan. 2023.

[7] A. Darabiha, A. C. Carusone, and F. R. Kschischang, "A bit-serial approximate min-sum LDPC decoder and FPGA implementation," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2006, pp. 149–152.

[8] S. M. Berber and V. Kecman, "Convolutional decoders based on artificial neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2, 2004, pp. 1551–1556.

[9] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Proc. 54th Annu. Allerton Conf. Commun., Control, Comput.*, 2016, pp. 341–346.

[10] E. Nachmani, E. Marciano, D. Burshtein, and Y. Be'ery, "RNN decoding of linear block codes," 2017, *arXiv:1702.07560*.

[11] T. Gruber, S. Cammerer, J. Hoydis, and S. T. Brink, "On deep learning-based channel decoding," in *Proc. 51st Annu. Conf. Inf. Sci. Syst. (CISS)*, 2017, pp. 1–6.

[12] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 1361–1365.

[13] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119–131, Feb. 2018.

[14] X. An, Y. Liang, and W. Zhang, "High-efficient reed-solomon decoder based on deep learning," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2020, pp. 1–5.

[15] Z. Ibrahim and Y. Fahmy, "Enhanced learning for recurrent neural network-based polar decoder," in *Proc. 13th Int. Conf. Electr. Eng. (ICEENG)*, 2022, pp. 105–109.

[16] Z. Xu and J. Sun, "Model-driven deep-learning," *Nat. Sci. Rev.*, vol. 5, no. 1, pp. 22–24, 2017.

[17] K. Andreev, A. Frolov, G. Svistunov, K. Wu, and J. Liang, "Deep neural network based decoding of short 5G LDPC codes," in *Proc. 18th Int. Symp. "Probl. Redundancy Inf. Control Syst." (REDUNDANCY)*, 2021, pp. 155–160.

[18] Q. Wang et al., "Normalized min-sum neural network for LDPC decoding," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 1, pp. 70–81, Feb. 2023.

[19] J. Li, S. Lin, K. Abdel-Ghaffar, W. E. Ryan, and D. J. Costello, "Globally coupled LDPC codes," in *Proc. Inf. Theory Appl. Workshop (ITA)*, 2016, pp. 1–10.

[20] J. Chen and M. P. C. Fossorier, "Decoding low-density parity check codes with normalized APP-based algorithm," in *Proc. IEEE Global Telecommun. Conf. (Cat. No. 01CH37270)*, vol. 2, 2001, pp. 1026–1030.

[21] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.

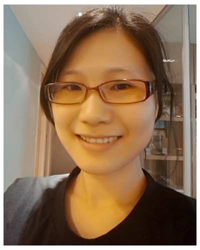
[22] F. Angarita, J. Valls, V. Almenar, and V. Torres, "Reduced-complexity min-sum algorithm for decoding LDPC codes with low error-floor," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 7, pp. 2150–2158, Jul. 2014.

[23] Y.-C. Liao, C. Lin, H.-C. Chang, and S. Lin, "A (21150, 19050) GC-LDPC decoder for NAND flash applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 3, pp. 1219–1230, Mar. 2019.

[24] K. Zhu and H. Yang, "Constructing grith eight GC-LDPC codes based on the GCD FLRM matrix with a new lower bound," *Sensors*, vol. 22, no. 19, pp. 7335–15, 2022.
[25] L. Chu, H. He, L. Pei, and R. C. Qiu, "NOLD: A neural-network optimized low-resolution decoder for LDPC codes," *J. Commun. Netw.*, vol. 23, no. 3, pp. 159–170, Jun. 2021.
[26] X. Xiao, "Neural-net decoding of quantum LDPC codes with straight-through estimators," in *Proc. Inf. Theory Appl. Workshop*, 2019, pp. 1–6.
[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980.*
[28] V. Petrov, T. Kurner, and I. Hosako, "IEEE 802.15. 3D: First standardization efforts for sub-terahertz band communications toward 6G," *IEEE Wireless Commun. Mag.*, vol. 58, no. 11, pp. 28–33, Nov. 2020.
[29] Y.-L. Ueng, C.-J. Yang, K.-C. Wang, and C.-J. Chen, "A multi-mode shuffled iterative decoder architecture for high-rate RS-LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 10, pp. 2790–2803, Oct. 2010.
[30] T. T. B. Nguyen and H. Lee, "Low-complexity multi-mode multi-way split-row layered LDPC decoder for gigabit wireless communications," *Integration*, vol. 65, pp. 189–200, Mar. 2019.

YU-LUN HSU received the B.S. and M.S. degrees in electric and electronic engineering from National Yang Ming Chiao Tung University, Hsinchu, Taiwan, in 2021 and 2023, respectively. She is currently working as an ASIC Design Engineer with Synopsys.

LI-WEI LIU (Member, IEEE) received the B.S. and Ph.D. degrees in electronics engineering from National Chiao Tung University (currently, National Yang Ming Chiao Tung University), Hsinchu, Taiwan, in 2018 and 2022, respectively. His research primarily focuses on error correction coding, machine learning, and VLSI architecture, with a particular emphasis on applications in storage and communication systems. His work aims to develop innovative solutions for enhancing data reliability and system performance.



YEN-CHIN LIAO (Member, IEEE) received the B.S. and M.S. degrees in communication engineering and the Ph.D. degree in electronics engineering from National Chiao Tung University (currently, National Yang Ming Chiao Tung University), Hsinchu, Taiwan, in 2000, 2002, and 2008, respectively.

She was with ISSC Technology from 2002 to 2003, where she was involved in developing the IEEE802.11a/g products. From 2008 to 2010, she was with Ralink Technology, where she was working on the IEEE802.11n enhancement projects and submitting contributions to the IEEE802.11ac standard. From 2010 to 2012, she was recruited by MDV Technology, where she participated in the IEEE802.11d/e product development. She joined the Department of Electronics Engineering, National Chiao Tung University (currently, National Yang Ming Chiao Tung University), as a Postdoctoral Researcher in 2012. Her research interests include error correction coding, machine learning, signal processing, and communication systems.

HSIE-CHIA CHANG (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the Electronics Engineering Department, National Chiao Tung University (currently, National Yang Ming Chiao Tung University), Hsinchu, Taiwan, in 1995, 1997, and 2002, respectively.

He was with OSP/DE1, MediaTek Corporation from 2002 to 2003, where he was involved in decoding architectures for combo single chip. In 2003, he joined the Electronics Engineering Department, National Chiao Tung University (currently, National Yang Ming Chiao Tung University), as a Faculty Member, where he has been a Professor since 2010. He has served as the Deputy Director General with Chip Implementation Center, Taiwan, from 2014 to 2017. He has authored a book, over 120 IEEE journal/conference papers, and over 70 U.S./China/Taiwan patents. His research interests include algorithms and VLSI architectures in signal processing, in particular, error control codes, and cryptosystems.

Dr. Chang was a recipient of the Outstanding Youth Electrical Engineer Award from the Chinese Institute of Electrical Engineering in 2010, and the Outstanding Youth Researcher Award from the Taiwan IC Design Society in 2011. He has also been serving as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: REGULAR PAPERS since 2012, the Circuits and System Society Chair in IEEE Taipei Section from 2019 to 2020, as well as a Technical Program Committee Member of the IEEE Asian Solid-State Circuits Conference from 2011 to 2013, and the International Solid-State Circuits Conference from 2018 to 2021.