Parallel Encoding Algorithm for LDPC Codes Based on Block-Diagonalization

Takayuki Nozaki
Yamaguchi University, JAPAN
Email: tnozaki@yamaguchi-u.ac.jp

***Abstract*—In this paper, we propose an efficient parallel encoding algorithm for the low-density parity-check (LDPC) codes. The main idea of the proposed encoding algorithm is the block-diagonalization of the parity part of a given parity check matrix by row and column permutation. The numerical examples in this paper show that the proposed encoding algorithm efficiently works in the multi-processor systems.**

I. Introduction

Low-density parity-check (LDPC) codes, invented by Gallager [1], are defined by sparse parity check matrices. Due to the sparseness of the parity check matrices, LDPC codes are decoded by the belief propagation (BP) algorithm, which is a parallel decoding algorithm with complexity $O(N)$, where N is the code-length. On the other hand, LDPC codes are encoded by its generator matrix with complexity $O(N^2)$. In other words, the complexity of encoding is larger than that of decoding for large code-length. Hence, it is important to reduce the encoding complexity of the LDPC codes.

Richardson and Urbanke [2] proposed the first efficient encoding algorithm, which is based on transforming the parity check matrix into an *approximate triangular matrix* by row and column permutation. This algorithm can encode the LDPC codes with complexity $O(N + \delta^2)$, where δ is proportional to N and $\delta \ll N$, and is called *gap*. Kaji [3] proposed an encoding algorithm based on LU-decomposition and showed that the encoding algorithm outperforms Richardson and Urbanke's (RU) encoding algorithm in terms of the complexity for small δ. Recently, encoding algorithms based on block-triangularization were proposed in [4], [5]. However, those encoding algorithms [2], [3], [4], [5] are *sequential* algorithms.

In this paper, we propose an efficient *parallel* encoding algorithm for the LDPC codes to reduce the time of encoding. The main idea of the proposed encoding algorithm is the block-diagonalization of the parity part of a given parity check matrix by row and column permutation. By using the block-diagonalization, encoding of LDPC codes comes down to several small systems of linear equations. By parallelly solving those small systems of linear equations, the proposed parallel algorithm is realized. Moreover, rearranging row and column of diagonal submatrices by the RU algorithm, those small systems of linear equations are efficiently solved. The numerical examples in this paper show that the proposed encoding algorithm efficiently works in the multi-processor systems.

The remainder of the paper is organized as follows. Section II briefly reviews the general framework of encoding algorithm for the LDPC codes and the RU encoding algorithm. Section III introduces hypergraph representation for parity check matrices and hypergraph partition problem. Those will be used for transforming a given parity check matrix into a singly bordered block-diagonal form. Section IV proposes a parallel encoding algorithm. Section V shows some numerical examples of the proposed algorithm and compares the complexity of the proposed algorithm and the RU encoding algorithm.

II. Preliminaries

This section briefly reviews a general framework of encoding for LDPC codes and the RU encoding algorithm.

A. General Framework of Encoding for LDPC codes

In this section, we consider the general framework of the encoding algorithm for the LDPC codes over the finite field $\mathbb{F}_q$ of the order q. We assume that the $M \times N$ parity check matrix $\mathbf{H}$ has full rank, i.e, $\mathrm{rank}(\mathbf{H}) = M$. For a given $\mathbf{H}$, encoding maps a message $\boldsymbol{m}$ of length $N - M$ into a codeword $\boldsymbol{x}$ of length N.

Encoding algorithms of LDPC codes are divided into two steps, called *precoding step* and *encoding step*. In the precoding step, the algorithm transforms the parity check matrix $\mathbf{H}$ into a *systematic form* $\mathbf{H}^{\mathrm{sys}} = (\mathbf{H}^P \quad \mathbf{H}^I)$, where $\mathbf{H}^P$ is a $M \times M$ non-singular matrix and called the parity part of $\mathbf{H}$ and $\mathbf{H}^I$ is called the information part of $\mathbf{H}$. Similarly, each codeword is partitioned as $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{m} \end{pmatrix}$, where $\boldsymbol{p} \in \mathbb{F}_q^M$ (resp. $\boldsymbol{m} \in \mathbb{F}_q^{N-M}$) is called the parity part (resp. information part). Since $\mathbf{H}\boldsymbol{x} = \mathbf{0}$ holds, we have $\mathbf{H}^P\boldsymbol{p} = -\mathbf{H}^I\boldsymbol{m}$. Hence, in the encoding step, the algorithm generates codeword $\boldsymbol{x}$ corresponding to the message $\boldsymbol{m}$ by solving $\mathbf{H}^P\boldsymbol{p} = -\mathbf{H}^I\boldsymbol{m}$. To reduce the encoding complexity, $\mathbf{H}^P$ should have some form which can be efficiently solved.

To keep the decoding performance of an LDPC code and the number of non-zero entries in its parity-check matrix $\mathbf{H}$, the precoding step transforms $\mathbf{H}$ only by row and column permutation, i.e, by using two permutation matrices $\mathbf{P}$ and $\mathbf{Q}$, $\mathbf{H}^{\mathrm{sys}} = \mathbf{PHQ}$. By summarizing above, the precoding step is regarded as an algorithm finding two permutation matrices $\mathbf{P}$ and $\mathbf{Q}$ such that $\mathbf{H}^P\boldsymbol{p} = -\mathbf{H}^I\boldsymbol{m}$ is efficiently solved.

B. Richardson and Urbanke's (RU) Encoding Algorithm

In this section, we briefly review the RU encoding algorithm. In the precoding step, the RU encoding algorithm transforms a given parity check matrix $\mathbf{H}$ into an approximate triangular matrix (ATM), which is described as

$$\mathbf{H}^{\mathrm{ATM}} = \mathbf{PHQ} = \begin{pmatrix} \mathbf{T} & \mathbf{C} & \mathbf{H}_u^I \\ \mathbf{D} & \mathbf{E} & \mathbf{H}_l^I \end{pmatrix}, \qquad (1)$$

where $\mathbf{H}^P = \begin{pmatrix} \mathbf{T} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{pmatrix}$, $\mathbf{H}^I = \begin{pmatrix} \mathbf{H}_u^I \\ \mathbf{H}_l^I \end{pmatrix}$, $\mathbf{T}$ is an $(M-\delta) \times (M-\delta)$ upper triangular matrix, $\mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{H}_u^I$ and $\mathbf{H}_l^I$ are $(M-\delta) \times \delta$, $\delta \times (M-\delta)$, $\delta \times \delta$, $(M-\delta) \times (N-M)$ and $\delta \times (N-M)$ matrices, respectively.

Assume that the parity part of a codeword is partitioned as $\boldsymbol{p} = \begin{pmatrix} \boldsymbol{p}_u \\ \boldsymbol{p}_l \end{pmatrix}$, where $\boldsymbol{p}_u$ and $\boldsymbol{p}_l$ are of length $(M-\delta)$ and δ, respectively. Define $\mathbf{\Phi} := (\mathbf{E} - \mathbf{D}\mathbf{T}^{-1}\mathbf{C})$. Then, the encoding step is given as follows:

1) Derive $\boldsymbol{p}_l$ from the following equation
$$\boldsymbol{p}_l = \mathbf{\Phi}^{-1}(\mathbf{D}\mathbf{T}^{-1}\mathbf{H}_u^I - \mathbf{H}_l^I)\boldsymbol{m}.$$
2) Solve $\boldsymbol{p}_u$ by using backward substitution of $\mathbf{T}$
$$\mathbf{T}\boldsymbol{p}_u = -\mathbf{C}\boldsymbol{p}_l - \mathbf{H}_u^I\boldsymbol{m}.$$

Denote the number of non-zero elements in the matrix $\mathbf{A}$, by $\mathrm{wt}(\mathbf{A})$. Let $\mathcal{Z}(\mathbf{A})$ be the number of rows in $\mathbf{A}$ such that there exists at least one non-zero element. Define $\mathcal{S}(\mathbf{A}) := \mathrm{wt}(\mathbf{A}) - \mathcal{Z}(\mathbf{A})$. The numbers of multiplication μ and addition α of the encoding step of the RU algorithm are

$$\mu = \mathrm{wt}(\mathbf{H}^I) + 2\mathrm{wt}(\mathbf{T}) + \mathrm{wt}(\mathbf{C}) + \mathrm{wt}(\mathbf{D}) + \mathrm{wt}(\mathbf{\Phi}^{-1}), \quad (2)$$
$$\alpha = \mathcal{S}(\mathbf{H}^I) + 2\mathcal{S}(\mathbf{T}) + \mathcal{S}(\mathbf{C}) + \mathcal{S}(\mathbf{D}) + \mathcal{S}(\mathbf{\Phi}^{-1}) + M, \quad (3)$$

respectively.

Summarizing the above, the precoding step of the RU algorithm is regarded as an algorithm, whose input is the original parity check matrix $\mathbf{H}$ and output is a four-tuple of matrices $(\mathbf{H}^{\mathrm{ATM}}, \mathbf{P}, \mathbf{Q}, \mathbf{\Phi}^{-1})$. Hence, we denote the precoding step of the RU algorithm, as $\mathsf{ATM}(\mathbf{H}) \to (\mathbf{H}^{\mathrm{ATM}}, \mathbf{P}, \mathbf{Q}, \mathbf{\Phi}^{-1})$. This notation will be used in Section IV-D.

III. Hypergraph representation and Partition Problem

This section defines hypergraphs and introduces the hypergraph representation of matrices [6, Section 3.2] and hypergraph partitioning problem.

A. Hypergraph Representation of Matrix

Definition 1 (Hypergraph): Let $\mathcal{U}$ be a finite set, and let $\mathcal{E}$ be a family of non-empty subset of $\mathcal{U}$. The pair $(\mathcal{U}, \mathcal{E})$ is called *hypergraph* with the set of vertices $\mathcal{U}$ and the set of *nets* (or *hyperedges*) $\mathcal{E}$. If the i-th node $u_i \in \mathcal{U}$ is in the j-th net $e_j \in \mathcal{E}$, i.e, $u_i \in e_j$, we call u_i connected to e_j.

The hypergraph $(\mathcal{U}, \mathcal{E})$ is called graph if $|e_j| \le 2$ for all $e_j \in \mathcal{E}$ [7]. In other words, a hypergraph is a generalization of a graph.

In [6], Çatalyürek and Aykanat gave two hypergraph representation of a sparse matrix, called *column net model* and *row net model*. In this paper, since we only employ the row net model, we refer the row net model as hypergraph representation of a matrix.

For a given $M \times N$ matrix $\mathbf{H} = (h_{i,j})$, the hypergraph representation $\mathcal{H}_{\mathbf{H}} = (\mathcal{U}, \mathcal{E})$ is constructed in the following way. The number of vertices $|\mathcal{U}|$ is m and the number of nets $|\mathcal{E}|$ is n. The node u_i is connected to the edge e_j iff $h_{i,j} \neq 0$, i.e, $u_i \in e_j \iff h_{i,j} \neq 0$. In other words, the i-th net (resp. j-th vertex) corresponds to the i-th column (resp. j-th row).

Remark 1: Consider the Tanner graph $\mathcal{G}_{\mathbf{H}}$ corresponding to the parity check matrix $\mathbf{H}$. If we transform the variable nodes (resp. check nodes) in $\mathcal{G}_{\mathbf{H}}$ to nets (resp. vertices), we can obtain the hypergraph representation for the parity check matrix $\mathbf{H}$.

B. Hypergraph Partitioning Problem

A family $\Pi = \{\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_K\}$ of non-empty subsets of $\mathcal{U}$ is a *K-way hypergraph partition* of $\mathcal{H} = (\mathcal{U}, \mathcal{E})$ if the followings are satisfied:

- Each pair of parts is disjoint, i.e, $\mathcal{U}_i \cap \mathcal{U}_j = \emptyset$ for all $1 \le i < j \le K$.
- Union of K parts is equal to $\mathcal{U}$, i.e, $\bigcup_{i=1}^K \mathcal{U}_i = \mathcal{U}$.

For a fixed partition Π, if a net $e \in \mathcal{E}$ connects to a node u in a part $\mathcal{U}_i$, we call that the net e connects to the part $\mathcal{U}_i$. Denote the set of nets connecting to a part $\mathcal{U}_i$, by $\mathcal{N}(\mathcal{U}_i)$. A net is called *cut* if the net connects to more than one parts. For a fixed partition Π, the set of cuts is called *cut set* and denoted by $\mathcal{X}(\Pi)$. The *cutsize* of Π is given by $|\mathcal{X}(\Pi)|$.

A partition is *balanced* [6] if the following holds:

$$\max_i |\mathcal{U}_i| \le \frac{|\mathcal{U}|}{K}(1+\epsilon),$$

where $\epsilon \ge 0$ represents the predetermined maximum imbalance ratio.

The K-way hypergraph partitioning problem is described as follows: For a given hypergraph $(\mathcal{U}, \mathcal{E})$, derive a balanced K-way partition Π such that $|\mathcal{X}(\Pi)|$ is minimized. The hypergraph partitioning problem is NP-hard [8]. However, there exist several heuristic algorithms. PaToH [9], [10] is a heuristic algorithm solving hypergraph partitioning problem. This paper employs PaToH to solve hypergraph partitioning problem.

IV. Parallel Encoding Algorithm

In this section, we propose a parallel encoding algorithm based on block-diagonalization. The precoding step of the proposed algorithm firstly transforms a given parity check matrix into a singly bordered (SB) block-diagonal form. Secondly, the precoding step of the proposed algorithm rearranges the rows and columns of the diagonal submatrices in the SB block-diagonal matrix.

This section is organized as follows. Section IV-A introduces transformation of a given parity check matrix into a SB block-diagonal form. Section IV-B rearranges the rows and columns of the diagonal submatrices in the SB block-diagonal matrix and constructs a parity part $\mathbf{H}^P$. Section IV-C gives the encoding step of the proposed algorithm. By summarizing Section IV-A and IV-B, Section IV-D presents the precoding step of the proposed encoding algorithm. Finally, we evaluate the complexity of the proposed algorithm in Section IV-E.

A. Singly bordered block-diagonalization

In this section, we define the K-way SB block-diagonal matrix and introduce K-way SB block-diagonalization for matrices [11].

Definition 2 (Block matrix): A block matrix is a matrix whose row and column are partitioned into several sections. A matrix $\mathbf{A}$ of size $M \times N$ with r row partitions and s column partitions is described as

$$\begin{pmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \cdots & \mathbf{A}_{1,s} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{r,1} & \mathbf{A}_{r,2} & \cdots & \mathbf{A}_{r,s} \end{pmatrix},$$

where $\mathbf{A}_{i,j}$ is of size $M_i \times N_j$. Notice that $\sum_{i=1}^{r} M_i = M$ and $\sum_{j=1}^{s} N_j = N$. We refer the submatrix $\mathbf{A}_{i,j}$ as the (i,j)-th block and refer i (resp. j) as block column (resp. block row).

Definition 3 (Block diagonal matrix): A block diagonal matrix is a square matrix whose non-zero blocks form square submatrices and are only arranged in diagonal blocks. A block diagonal matrix with r row and column partitions is denoted by $\mathrm{diag}[\mathbf{A}_{1,1}, \dots, \mathbf{A}_{r,r}]$.

Definition 4 (K-way SB block-diagonal matrix): A SB block diagonal matrix is a block matrix whose non-zero blocks are only arranged in diagonal blocks and the last block column. The $M \times N$ matrix is a K-way SB block-diagonal matrix if the matrix is described as

$$\begin{pmatrix} \mathbf{A}_1 & O & \cdots & O & \mathbf{B}_1 \\ O & \mathbf{A}_2 & & O & \mathbf{B}_2 \\ \vdots & & \ddots & \vdots & \vdots \\ O & O & \cdots & \mathbf{A}_K & \mathbf{B}_K \\ O & O & \cdots & O & \mathbf{A}_{K+1} \end{pmatrix},$$

where $\mathbf{A}_i$ (resp. $\mathbf{B}_i$) is $M_i \times N_i$ (resp. $M_i \times N_{K+1}$) matrix and $M_{K+1} \geq 0$. In particular, if $M_{K+1} = 0$, then a K-way SB block-diagonal matrix is written in

$$\begin{pmatrix} \mathbf{A}_1 & O & \cdots & O & \mathbf{B}_1 \\ O & \mathbf{A}_2 & & O & \mathbf{B}_2 \\ \vdots & & \ddots & \vdots & \vdots \\ O & O & \cdots & \mathbf{A}_K & \mathbf{B}_K \end{pmatrix}.$$

We refer to transforming matrix H to K-way SB block-diagonal matrix as K-way SB block-diagonalization of $\mathbf{H}$.

For a given K-way hypergraph partition Π, define $\mathcal{U}_{K+1} := \{u \in \mathcal{U} \mid u \notin e \ \ \forall e \in \mathcal{E} \setminus \mathcal{X}(\Pi)\}$. In words, $\mathcal{U}_{K+1}$ represents the set of vertices only connected to $\mathcal{X}(\Pi)$. The following algorithm provides a K-way SB block-diagonalization of a parity check matrix $\mathbf{H}$ by using K-way hypergraph partition.

Algorithm 1 (SB block-diagonalization):
Input: An $M \times N$ parity check matrix $\mathbf{H}$, the number of partitions K
Output: A K-way SB block-diagonalized matrix $\mathbf{H}_K^{\mathrm{SBBD}}$

1) Construct hypergraph representation $\mathcal{H}_{\mathbf{H}}$ for $\mathbf{H}$.
2) By solving the K-way hypergraph partitioning problem for $\mathcal{H}_{\mathbf{H}}$, provide $\Pi = \{\mathcal{U}_i\}_{1 \leq i \leq K}$.
3) (Column-permutation) For $i \in [1, K] := \{1, 2, \dots, K\}$, label the columns corresponding to the nets in $\mathcal{N}(\mathcal{U}_i) \setminus \mathcal{X}(\Pi)$, by i. Label the columns corresponding to the nets in $\mathcal{X}(\Pi)$, by $K+1$. Sort the columns of $\mathbf{H}$ in ascending order of labels.
4) (Row-permutation) For $i \in [1, K]$, label the rows corresponding to the vertices in $\mathcal{U}_i \setminus \mathcal{U}_{K+1}$, by i. Label the rows corresponding to the vertices in $\mathcal{U}_{K+1}$, by $K+1$. Sort the rows of $\mathbf{H}$ in ascending order of labels.
5) Output the resulting matrix as $\mathbf{H}_K^{\mathrm{SBBD}}$

Remark 2: For $i \in [1, K]$, the vertices in $\mathcal{U}_i \setminus \mathcal{U}_{K+1}$ corresponds to the i-th block column of $\mathbf{H}^{\mathrm{SBBD}}$. Similarly, for $j \in [1, K]$, the nets in $\mathcal{N}(\mathcal{U}_i) \setminus \mathcal{X}(\Pi)$ corresponds to the j-th block row of $\mathbf{H}^{\mathrm{SBBD}}$. Since the nets in $\mathcal{N}(\mathcal{U}_i) \setminus \mathcal{X}(\Pi)$ connects only to vertices in $\mathcal{U}_i \setminus \mathcal{U}_{K+1}$, the (i,j)-th block $\mathbf{A}_{i,j}$ is equal to O for $i \neq j$, $i, j \in [1, K]$.

Assume that the parity check matrix $\mathbf{H} \in \mathbb{F}^{M \times N}$ has full rank, i.e, $\mathrm{rank}(\mathbf{H}) = M$. Then the following lemmas and proposition hold.

Lemma 1: If $\mathbf{H}$ has full rank, $M_{K+1} \leq N_{K+1}$ holds.

Proposition 1: If $\mathbf{H}$ has full rank, sub-matrix $\mathbf{A}_{K+1}$ has full rank, i.e, $\mathrm{rank}(\mathbf{A}_{K+1}) = M_{K+1}$.
We can obtain this lemma and proposition in a proof by contradiction.

Proposition 1 shows that the sub-matrix $\mathbf{A}_{K+1}$ always has full rank. On the other hand, there is a possibility that $\mathrm{rank}(\mathbf{A}_i) < M_i$ for $i \in [1, K]$.

B. Construction of Parity Part $\mathbf{H}^P$

Assume $\mathrm{rank}(\mathbf{A}_i) = M_i$, for all $i \in [1, K]$. As discussed in Section II, by using conventional encoding algorithm, e.g, the RU encoding algorithm, we can obtain a pair of permutation matrix $(\mathbf{P}_i, \mathbf{Q}_i)$ such that $\mathbf{H}_i^P$, given by $\mathbf{P}_i \mathbf{A}_i \mathbf{Q}_i = (\mathbf{H}_i^P \ \ \mathbf{I}_i^I)$, is efficiently solved. In this section, we rearrange the rows and columns of submatrices $\mathbf{A}_i$ by using such $(\mathbf{P}_i, \mathbf{Q}_i)$ and construct the parity part $\mathbf{H}^P$.

1) In the case of $M_{K+1} > 0$: Denote $\mathbf{P}_i \mathbf{B}_i \mathbf{Q}_{K+1} = (\mathbf{B}_i^P \ \ \mathbf{B}_i^I)$ for $i \in [1, K]$, where $\mathbf{B}_i^P$ (resp. $\mathbf{B}_i^I$) is an $M_i \times M_{K+1}$ (resp. $M_i \times (N_{K+1} - M_{K+1})$) matrix. By the column permutation matrix $\mathrm{diag}[\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_{K+1}]$ and the row permutation matrix $\mathrm{diag}[\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_{K+1}]$, the matrix $\mathbf{H}_K^{\mathrm{SBBD}}$ is transformed as:

$$\begin{pmatrix} \mathbf{A}_1^P & \mathbf{A}_1^I & O & O & \cdots & O & O & \mathbf{B}_1^P & \mathbf{B}_1^I \\ O & O & \mathbf{A}_2^P & \mathbf{A}_2^I & & O & O & \mathbf{B}_2^P & \mathbf{B}_2^I \\ \vdots & \vdots & & & \ddots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & \cdots & \mathbf{A}_K^P & \mathbf{A}_K^I & \mathbf{B}_K^P & \mathbf{B}_K^I \\ O & O & O & O & \cdots & O & O & \mathbf{A}_{K+1}^P & \mathbf{A}_{K+1}^I \end{pmatrix}$$

The submatrix $\mathbf{H}^P$ (resp. $\mathbf{H}^I$) is constructed from the odd (resp. even) block columns, i.e,

$$\mathbf{H}^P = \begin{pmatrix} \mathbf{A}_1^P & \cdots & O & \mathbf{B}_1^P \\ \vdots & \ddots & \vdots & \vdots \\ O & \cdots & \mathbf{A}_K^P & \mathbf{B}_K^P \\ O & \cdots & O & \mathbf{A}_{K+1}^P \end{pmatrix}, \tag{4}$$

$$\mathbf{H}^I = \begin{pmatrix} \mathbf{A}_1^I & \cdots & O & \mathbf{B}_1^I \\ \vdots & \ddots & \vdots & \vdots \\ O & \cdots & \mathbf{A}_K^I & \mathbf{B}_K^I \\ O & \cdots & O & \mathbf{A}_{K+1}^I \end{pmatrix} =: \begin{pmatrix} \mathbf{H}_1^I \\ \vdots \\ \mathbf{H}_K^I \\ \mathbf{H}_{K+1}^I \end{pmatrix}. \tag{5}$$

2) In the case of $M_{K+1} = 0$: Define $\mathbf{B}_i^I := \mathbf{P}_i\mathbf{B}_i$. In a similar way to the previous section, by the column permutation matrix $\mathrm{diag}[\mathbf{P}_1, \ldots, \mathbf{P}_K]$ and the row permutation matrix $\mathrm{diag}[\mathbf{Q}_1, \ldots, \mathbf{Q}_K, \mathbf{I}]$, the matrix $\mathbf{H}_K^{\mathrm{SBBD}}$ is transformed as:

$$\begin{pmatrix} \mathbf{A}_1^P & \mathbf{A}_1^I & O & O & \cdots & O & O & \mathbf{B}_1^I \\ O & O & \mathbf{A}_2^P & \mathbf{A}_2^I & & O & O & \mathbf{B}_2^I \\ \vdots & \vdots & & & \ddots & \vdots & \vdots & \vdots \\ O & O & O & O & \cdots & \mathbf{A}_K^P & \mathbf{A}_K^I & \mathbf{B}_K^I \end{pmatrix}.$$

The submatrix $\mathbf{H}^P$ (resp. $\mathbf{H}^I$) is constructed as:

$$\mathbf{H}^P = \begin{pmatrix} \mathbf{A}_1^P & & O \\ & \ddots & \\ O & & \mathbf{A}_K^P \end{pmatrix}, \tag{6}$$

$$\mathbf{H}^I = \begin{pmatrix} \mathbf{A}_1^I & \cdots & O & \mathbf{B}_1 \\ \vdots & \ddots & \vdots & \vdots \\ O & \cdots & \mathbf{A}_K^I & \mathbf{B}_K \end{pmatrix} =: \begin{pmatrix} \mathbf{H}_1^I \\ \vdots \\ \mathbf{H}_K^I \end{pmatrix}. \tag{7}$$

Since $\det(\mathbf{H}^P) = \prod_i \det(\mathbf{A}_i^P) \neq 0$, the parity part $\mathbf{H}^P$ is non-singular. In other words, there is an unique solution of $\mathbf{H}^P\boldsymbol{p} = -\mathbf{H}^I\boldsymbol{m}$ for a given $\boldsymbol{m}$.

3) Construction Algorithm: Let $I\{\cdot\}$ be the indicator function which is equal to 1 if the condition inside the braces is fulfilled and 0 otherwise. The construction of parity part $\mathbf{H}^P$ and information part $\mathbf{H}^I$ is summarized as the following algorithm.

Algorithm 2 (Construction of $\mathbf{H}^P$ and $\mathbf{H}^I$):
Input: A K-way SB block diagonal matrix $\mathbf{H}_K^{\mathrm{SBBD}}$ with $\mathrm{rank}(\mathbf{A}_i) = M_i$ for all $i \in [1, K+1]$
Output: Parity part $\mathbf{H}^P$ and information part $\mathbf{H}^I$

1) By using $\mathsf{ATM}(\mathbf{A}_i) \to (\mathbf{A}_i^{\mathrm{ATM}}, \mathbf{P}_i, \mathbf{Q}_i, \mathbf{\Phi}_i^{-1})$, get $(\mathbf{P}_i, \mathbf{Q}_i)$ for all $i \in [1, K + I\{M_{K+1} > 0\}]$.
2) If $M_{K+1} > 0$ holds, then construct $\mathbf{H}^I$ and $\mathbf{H}^P$ as in Section IV-B1. Otherwise, constrict $\mathbf{H}^I$ and $\mathbf{H}^P$ as in Section IV-B2.

Remark 3: In Step 1, Algorithm 2 employs the RU encoding algorithm to get permutation matrices $(\mathbf{P}_i, \mathbf{Q}_i)$. Naturally, we can employ other encoding algorithms to get permutation matrices $(\mathbf{P}_i, \mathbf{Q}_i)$. In other words, we can replace the RU algorithm used in Step 1 with other encoding algorithm which gives $(\mathbf{P}_i, \mathbf{Q}_i)$.

C. Encoding Step

Let $\boldsymbol{p}$ (resp. $\boldsymbol{m}$) be partitioned into $K + 1$ parts $\boldsymbol{p}_i$ (resp. $\boldsymbol{m}_i$) of length M_i (resp. $N_i - M_i$). Then, the encoding step of the proposed algorithm is the following.

1) In the case of $M_{K+1} > 0$: From (4) and (5), we have

$$\mathbf{A}_i^P\boldsymbol{p}_i + \mathbf{B}_i^P\boldsymbol{p}_{K+1} = -\mathbf{A}_i^I\boldsymbol{m}_i - \mathbf{B}_i^I\boldsymbol{m}_{K+1}, \ \forall i \in [1, K], \tag{8}$$

$$\mathbf{A}_{K+1}^P\boldsymbol{p}_{K+1} = -\mathbf{A}_{K+1}^I\boldsymbol{m}_{K+1}. \tag{9}$$

Notice that from (8), $\boldsymbol{p}_i$ for $i \in [1, K]$ is solved if $\boldsymbol{p}_{K+1}$ is given. Hence, the encoding algorithm is given as follows:

1) Solving (9), derive $\boldsymbol{p}_{K+1}$.
2) Parallelly solving (8), derive $\boldsymbol{p}_i$ for all $i \in [1, K]$.

Notice that $\mathbf{A}_i^P\boldsymbol{p}_i = \boldsymbol{b}$ is efficiently solved by some conventional methods, e.g, the RU algorithm.

2) In the case of $M_{K+1} = 0$: From (6) and (7), we have

$$\mathbf{A}_i^P\boldsymbol{p}_i = -\mathbf{A}_i^I\boldsymbol{m}_i - \mathbf{B}_i^I\boldsymbol{m}_{K+1}, \quad \forall i \in [1, K]. \tag{10}$$

Notice that $\boldsymbol{p}_i$ is derived independently each other, from (10). Hence the encoding algorithm is given as follows:

1) Parallelly solving (10), derive $\boldsymbol{p}_i$ for all $i \in [1, K]$.

Therefore, the proposed encoding algorithm efficiently works in the K-processor system if $M_{K+1} = 0$ holds.

D. Precoding Step

By summarizing Section IV-A and IV-B, this section gives the precoding step of the proposed algorithm.

Algorithm 3 (Precoding step):
Input: An $M \times N$ full-rank parity check matrix $\mathbf{H}$, the maximum number of partitions $K_{\max}$
Output: Parity part $\mathbf{H}^P$ and information part $\mathbf{H}^I$

1) Set $K \leftarrow K_{\max}$.
2) Inputting $\mathbf{H}$ and K to Algorithm 1, get $\mathbf{H}_K^{\mathrm{SBBD}}$.
3) If $\mathrm{rank}(\mathbf{A}_i) = m_i$ for all $i \in [1, K]$, then go to Step 5. Otherwise, set $K \leftarrow K - 1$ and go to the next step.
4) If $K > 1$, then go to Step 2. Otherwise, go to the next step.
5) Construct $\mathbf{H}^P$ and $\mathbf{H}^I$, by using Algorithm 2 with input $\mathbf{H}_K^{\mathrm{SBBD}}$.

Remark 4: As discussed in Section IV-C, $M_{K+1} = 0$ is desired since the encoding algorithm parallelly works. Hence, Step 2 should be repeated until $M_{K+1} = 0$ holds.

E. Complexity

In this section, we evaluate the encoding complexity in the case of employing the RU algorithm to derive permutation matrices $(\mathbf{P}_i, \mathbf{Q}_i)$. More precisely, we compute the number of multiplication and addition of the proposed algorithm. Then, matrix $\mathbf{A}_i^P$ is described as $\begin{pmatrix} \mathbf{T}_i & \mathbf{C}_i \\ \mathbf{D}_i & \mathbf{E}_i \end{pmatrix}$ and $\mathbf{\Phi}_i := \mathbf{E}_i - \mathbf{D}_i\mathbf{T}_i^{-1}\mathbf{C}_i$.

In the case of $M_{K+1} = 0$, from (10), the numbers of multiplication μ_i and addition α_i for deriving $\boldsymbol{p}_i$ are given in a similar way of Section II-B

$$\mu_i = \mathrm{wt}(\mathbf{H}_i^I) + 2\mathrm{wt}(\mathbf{T}_i) + \mathrm{wt}(\mathbf{C}_i) + \mathrm{wt}(\mathbf{D}_i) + \mathrm{wt}(\mathbf{\Phi}_i^{-1}),$$
$$\alpha_i = \mathcal{S}(\mathbf{H}_i^I) + 2\mathcal{S}(\mathbf{T}_i) + \mathcal{S}(\mathbf{C}_i) + \mathcal{S}(\mathbf{D}_i) + \mathcal{S}(\mathbf{\Phi}_i^{-1}) + M_i.$$

TABLE I
THE BLOCK SIZE OF $\mathbf{H}_2^{\mathrm{SBBD}}$ FOR SOME CODES IN [12]

Name	(N, M)	(N_1, M_1)	(N_2, M_2)	(N_3, M_3)
PEGReg504x1008	(1008, 504)	(277,251)	(272,253)	(459,0)
PEGReg252x504	(504, 252)	(128,126)	(141,126)	(235,0)
PEGirReg504x1008	(1008, 504)	(302,252)	(313,252)	(393,0)
PEGirReg252x504	(504, 252)	(149,126)	(154,126)	(201,0)
32000.2240.3.105	(32000, 2240)	(5656,1000)	(6116,1140)	(20228,0)
16383.2130.3.103	(16383, 2130)	(3365,1055)	(3471,1075)	(9547,0)
4095.737.3.101	(4095, 737)	(893,369)	(902,368)	(2300,0)
10000.10000.3.631	(20000,10000)	(5764,4904)	(6166,5096)	(8120,0)
8000.4000.3.483	(8000, 4000)	(2373,1990)	(2400,2010)	(3227,0)
4000.2000.3.243	(4000, 2000)	(1158,989)	(1209,1011)	(1633,0)
504.504.3.504	(1008, 504)	(301,253)	(297,251)	(410,0)

TABLE II
COMPARISON OF ENCODING COMPLEXITY FOR THE PROPOSED ALGORITHM WITH THE RU ALGORITHM

Name	RU Algorithm		Proposed Algorithm			
	μ/α	(δ)	μ_1/α_1	(δ_1)	μ_2/α_2	(δ_2)
PEGReg504x1008	4599/3542	(21)	2369/1802	(23)	2384/1819	(23)
PEGReg252x504	2283/1739	(16)	1125/839	(13)	1124/843	(13)
PEGirReg504x1008	5068/4058	(1)	2587/2079	(2)	2599/2093	(1)
PEGirReg252x504	2560/2054	(1)	1284/1028	(2)	1289/1034	(1)
32000.2240.3.105	102659/98159	(7)	50366/48136	(10)	52180/49871	(10)
16383.2130.3.103	55472/51164	(16)	27453/25298	(15)	28055/25848	(19)
4095.737.3.101	14418/12915	(10)	7192/6428	(9)	7174/6412	(9)
10000.10000.3.631	144200/123378	(337)	87501/76950	(302)	96271/85278	(325)
8000.4000.3.483	45006/36709	(141)	23913/19638	(117)	25204/20868	(127)
4000.2000.3.243	20240/16075	(74)	10417/8279	(64)	10475/8298	(61)
504.504.3.504	4572/3517	(19)	2262/1721	(14)	2273/1729	(16)

In the case of $M_{K+1} > 0$, the numbers of multiplication μ_i^* and addition α_i^* for deriving $\boldsymbol{p}_i$ are given in a similar way of Section II-B

$$\begin{aligned}\mu_i^* =&\mathrm{wt}(\mathbf{H}_i^I) + 2\mathrm{wt}(\mathbf{T}_i) + \mathrm{wt}(\mathbf{C}_i) + \mathrm{wt}(\mathbf{D}_i) + \mathrm{wt}(\mathbf{\Phi}_i^{-1})\\ &+ \mathrm{wt}(\mathbf{B}_i^P),\\ \alpha_i^* =&\mathcal{S}(\mathbf{H}_i^I) + 2\mathcal{S}(\mathbf{T}_i) + \mathcal{S}(\mathbf{C}_i) + \mathcal{S}(\mathbf{D}_i) + \mathcal{S}(\mathbf{\Phi}_i^{-1}) + 2M_i\\ &+ \mathcal{S}(\mathbf{B}_i^P).\end{aligned}$$

V. NUMERICAL EXAMPLES

In this section, we give some numerical examples of the proposed encoding algorithm and compare the complexity of the proposed encoding algorithm with the RU encoding algorithm.

A. Examples of SB Block-Diagonalization

In this section, we transform some parity check matrices given in [12] into 2-way SB block-diagonal matrices by Algorithm 1. From Table I, we see that Algorithm 1 gives 2-way SB block diagonal matrices with $M_3 = 0$ for those examples. In other words, the parity part $\boldsymbol{p}_1, \boldsymbol{p}_2$ can be parallelly and simultaneously solved. Moreover, we see that M_1 is nearly equal to M_2. In other words, the size of diagonal blocks are nearly equal.

B. Comparison of Encoding Complexity

In this section, we calculate the number of operations of the proposed encoding algorithm and compare with that of the RU algorithm. In those numerical examples, at first, we construct $\mathbf{H}^P$ and $\mathbf{H}^I$ by using Algorithm 2, where the inputs of Algorithm 2 are 2-way SB block-diagonal matrix given in Table I. Next, we calculate the number of operations for the $\mathbf{H}^P$ and $\mathbf{H}^I$ from equations in Section IV-E. The number of operations are given in Table II.

Table II shows the numbers of multiplication μ and addition α for the RU algorithm and the proposed algorithm. In the first column of Table II gives the name of codes given in [12]. The second column of Table II gives the number of operations μ, α and the gap δ of the approximate triangular matrix. The third (resp. fourth) column gives the number of operations and the gap for deriving $\boldsymbol{p}_1$ (resp. $\boldsymbol{p}_2$). Since the derivation of $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are carried out two processors, μ_1 and α_1 (resp. μ_2 and α_2) show the numbers of operations of the first processor (resp. second processor).

From Table I, we see that, the number of operations to derive $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are nearly equal. In other words, the proposed algorithm adequally divides the operation for encoding to two processors.

VI. CONCLUSION AND FUTURE WORKS

In this paper, we have proposed a parallel encoding algorithm based on block-diagonalization of the parity part of the parity check matrix. Numerical examples in the paper have shown that the proposed algorithm adequally divides the operation for encoding to processors. As a future work, we will extend this algorithm for the spatially coupled codes.

ACKNOWLEDGMENT

This work was partially supported by JSPS KAKENHI Grant Number 25889061.

REFERENCES

[1] R. G. Gallager, *Low Density Parity Check Codes.* in Research Monograph series, MIT Press, Cambridge, 1963.

[2] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.

[3] Y. Kaji, "Encoding LDPC codes using the triangular factorization," *IEICE Trans. Fundamentals*, vol. 89, no. 10, pp. 2510–2518, 2006.

[4] T. Shibuya, "Block-triangularization of parity check matrices for efficient encoding of linear codes," in *Proc. 2011 IEEE Int. Symp. Inf. Theory (ISIT)*, July 2011, pp. 533–537.

[5] T. Shibuya and K. Kobayashi, "Efficient linear time encoding for LDPC codes," *IEICE Trans. Fundamentals*, vol. 97, no. 7, pp. 1556–1567, 2014.

[6] U. V. Çatalyürek and C. Aykanat, "Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 7, pp. 673–693, Jul 1999.

[7] C. Berge, *Hypergraphs*, ser. North-Holland mathematical Library. Elsevier, 1989.

[8] T. Lengauer, *Combinatorial algorithms for integrated circuit layout.* John Wiley & Sons, Inc., 1990.

[9] Ü. V. Çatalyürek, "Hypergraph models for sparse matrix partitioning and reordering," Ph.D. dissertation, Computer Engineering and Information Science Bilkent University, 1999.

[10] Ü. Çatalyürek and C. Aykanat, "PaToH (partitioning tool for hypergraphs)," in *Encyclopedia of Parallel Computing*. Springer, 2011, pp. 1479–1487.

[11] C. Aykanat, A. Pinar, and U. V. Çatalyürek, "Permuting sparse rectangular matrices into block-diagonal form," *SIAM J. Sci. Comput.*, vol. 25, no. 6, pp. 1860–1879, Jun. 2004.

[12] D. Mackay, "Encyclopedia of sparse graph codes," http://wol.ra.phy.cam.ac.uk/mackay/codes/data.html.