Spatially Coupled PLDPC-Hadamard Convolutional Codes

Peng-Wei Zhang, Francis C. M. Lau, Fellow, IEEE, and Chiu-Wing Sham, Senior Member, IEEE

Abstract—We propose a new type of ultimate-Shannon-limit-approaching codes called spatially coupled protograph-based low-density parity-check Hadamard convolutional codes (SC-PLDPC-HCCs), which are constructed by spatially coupling PLDPC-Hadamard block codes. We develop an efficient decoding algorithm that combines pipeline decoding and layered scheduling for the decoding of SC-PLDPC-HCCs, and analyze the latency and complexity of the decoder. To estimate the decoding thresholds of SC-PLDPC-HCCs, we first propose a layered protograph extrinsic information transfer (PEXIT) algorithm to evaluate the thresholds of spatially coupled PLDPC-Hadamard terminated codes (SC-PLDPC-TDCs) with a moderate coupling length. With the use of the proposed layered PEXIT method, we develop a genetic algorithm to find good SC-PLDPC-TDCs in a systematic way. Then we extend the coupling length of these SC-PLDPC-TDCs to form good SC-PLDPC-HCCs. Results show that our constructed SC-PLDPC-HCCs can achieve comparable thresholds to the block code counterparts. Simulations illustrate the superiority of the SC-PLDPC-HCCs over the block code counterparts and other state-of-the-art low-rate codes in terms of error performance. For the rate-0.00295 SC-PLDPC-HCC, a bit error rate of $10^{-5}$ is achieved at $E_b/N_0 = -1.465$ dB, which is only $0.125$ dB from the ultimate Shannon limit.

Index Terms—Protograph LDPC code, PLDPC-Hadamard code, PEXIT algorithm, spatially coupled PLDPC codes, spatially coupled PLDPC-Hadamard codes, ultimate Shannon limit.

I. INTRODUCTION

Binary low-density parity-check (LDPC) codes were first proposed in 1963 [1], whose sparse parity-check matrix containing “0” and “1” is randomly constructed, and can be graphically represented by a Tanner graph [2]. A Tanner graph consists of check nodes (CNs), variable nodes (VNs) and the connections between CNs and VNs. After receiving the channel observations, extrinsic information along the connections is iteratively exchanged and calculated at CN processors and VN processors to realize belief propagation (BP) decoding. In [3], a density evolution method is proposed to evaluate the probability density function (PDF) of the extrinsic information and hence to optimize the degree distributions of LDPC codes. Subsequently in [4], an extrinsic information transfer (EXIT) chart technique is proposed to optimize LDPC codes by calculating the mutual information (MI) of the extrinsic information. Through both methods, good-performing LDPC codes are constructed to work close to the Shannon limit under binary erasure channels (BECs), binary symmetric channels (BSCs) and additive-white-Gaussian-noise (AWGN) channels [3]–[5]. Yet, most LDPC codes are designed to achieve good decoding performance at a bit-energy-to-noise-power-spectral-density ratio ($E_b/N_0$) greater than 0 dB. To approach the ultimate Shannon limit, i.e., $E_b/N_0 = -1.59$ dB [6], other low-rate codes are designed. Potential application scenarios of such codes include multiple access wireless systems (e.g., interleave-division multiple-access [7], [8] with a huge number of non-orthogonal users) and deep space communications. In [9] and [10], very low-rate turbo Hadamard codes and zigzag Hadamard codes are proposed. However, their decoders require the use of serial decoding [11]–[13] and cannot make use of parallel decoding as in LDPC decoders. The error performances of turbo Hadamard codes and zigzag Hadamard codes are also not as good as those of the LDPC-Hadamard codes that have been subsequently proposed [14].

In the Tanner graph of an LDPC code, the edges connected to a VN form a repeat code, whereas the edges connected to a CN form a single-parity-check (SPC) code. When the repeat codes and/or SPC codes are replaced with other block codes, a Tanner graph of a generalized LDPC is formed. By replacing SPC codes with Hadamard codes, LDPC-Hadamard codes (LDPC-HCs) have been proposed and their degree distributions have been optimized with the EXIT chart method [14]. The LDPC-HCs not only have thresholds lower than $-1.34$ dB, but also achieve excellent decoding performance at a $E_b/N_0$ lower than $-1.17$ dB. On the other hand, the EXIT chart method cannot effectively analyze degree distributions containing degree-1 and/or punctured VN nodes. The parity-check matrix derived by the progressive-edge-growth (PEG) method [15] according to the degree distributions does not contain any structure and therefore does not facilitate linear encoding and parallel decoding. In [16], a class of structured LDPC-HCs called protograph-based LDPC-HCs (PLDPC-HCs) have been proposed.
Protograph-based LDPC (PLDPC) codes can be described by a small protomatrix or protograph [17]. Using the copy-and-permute operations to lift the protomatrix or protograph, the derived matrix or lifted graph can easily have a quasi-cyclic structure which is conducive to the hardware implementations of encoders and decoders [18]. PLDPC-HCs [16] also inherit such advantages from PLDPC codes. By adding an appropriate amount of degree-1 Hadamard variable nodes (D1H-VNs) to the check nodes in the protograph of a PLDPC code, the SPC constraints are converted into Hadamard constraints and the protograph of a PLDPC-HC is obtained. The protograph-based EXIT (PEXIT) chart method for PLDPC codes [19] has further been modified for analyzing PLDPC-HCs [16]. The modified method not only can produce multiple extrinsic mutual information from the symbol-by-symbol maximum-a-posteriori-probability (symbol-MAP) Hadamard decoder, but also is applicable to analyzing protographs with degree-1 and/or punctured VNs. The structured PLDPC-HCs can achieve good thresholds and comparable error performance as the traditional LDPC-HCs. In [20], a layered decoding algorithm has been proposed to double the convergence rate compared with standard decoding.

While PLDPC-HCs are already working very close to the ultimate Shannon limit, further improvement is possible. In [21], it has been shown that LDPC convolutional codes (LDPC-CCs) can achieve convolutional gains over their block-code counterparts. LDPC-CCs were first proposed in [22] and characterized by the degree distributions of the underlying LDPC block codes (LDPC-BCs). In addition, spatially coupled LDPC (SC-LDPC) codes are constructed by coupling a number of LDPC-BCs. As the number of coupled LDPC-BCs tends to infinity, spatially coupled LDPC convolutional codes (SC-LDPC-CCs) are obtained. In [23] and [24], SC-LDPC-CCs have been shown to achieve capacity over binary memoryless symmetric channels under BP decoding. In [25], SC-LDPC-CCs have been constructed from the perspective of protographs, namely SC-PLDPC-CCs. Through the edge-spreading procedure on protomatrix, the threshold, convergence behavior and error performance of SC-PLDPC ensembles have also been systematically investigated. In particular, PEXIT algorithms have been applied to analyze binary [26] and q-ary SC-LDPC codes [27]. Based on the aforementioned results, spatially coupled PLDPC-Hadamard convolutional codes (SC-PLDPC-HC), which are to be investigated in this paper, have the potential to provide extra gains over their block-code counterparts. A genetic algorithm (GA) will also be proposed to optimize the design of SC-PLDPC-HC.

GA is an optimization algorithm that simulates the evolution of nature, and is widely used in music generation, genetic synthesis and VLSI technology [28]. In the arena of channel codes, GA has been applied to adjust the code rate of turbo codes without puncturing [29]; to construct polar codes that reduce the decoding complexity while maintaining the same decoding performance [30]; together with bit-error-rate (BER) simulations to optimize error performance of short length LDPC codes over both AWGN and Rayleigh fading channels [31]; together with density evolution to optimize the degree distributions of the SC-LDPC codes over BEC channels [32]. GA first forms an offspring generation from the parent generation, and performs operations such as crossover, mutation, and selection on the offspring generation. A similar technique called differential evolution algorithm directly uses the parent generation to perform these operations to achieve the evolution. Differential evolution based optimization methods have been applied to design LDPC and SC-LDPC codes [33], [34]. Compared with the differential evolution algorithm, the genetic algorithm has a stronger global search ability but requires more complex procedures.

In this paper we propose a new type of ultimate-Shannon-limit-approaching codes, namely spatially coupled PLDPC-Hadamard convolutional codes (SC-PLDPC-CCs), and conduct an in-depth investigation into the proposed codes. Our main contributions are as follows.

1) We propose a new type of ultimate-Shannon-limit-approaching codes, namely spatially coupled PLDPC-Hadamard convolutional codes (SC-PLDPC-CCs), which are constructed by spatially coupling PLDPC-Hadamard block codes.

2) We describe the encoding method of SC-PLDPC-CCs. We also develop an efficient decoding algorithm, i.e., a pipeline decoding strategy combined with layered scheduling, for the decoding of SC-PLDPC-CCs, and analyze its latency and complexity.

3) Using the original PEXIT method in [16], we show that the thresholds of different spatially coupled PLDPC-Hadamard terminated codes (SC-PLDPC-TDCs) are distinguishable and improves as the coupling length increases. To improve the convergence rate of the original PEXIT method, we propose a layered PEXIT algorithm to efficiently evaluate the threshold of SC-PLDPC-TDCs with a given coupling length. The thresholds of SC-PLDPC-TDCs with large coupling lengths are then used as estimates for thresholds of SC-PLDPC-CCs.

4) We propose a GA to systematically search for SC-PLDPC-TDCs having good thresholds. Based on the same set of split protomatrices for good SC-PLDPC-TDCs, we extend the coupling length to construct the convolutional codes, i.e., SC-PLDPC-CCs.

5) We have found SC-PLDPC-CCs with comparable thresholds to the underlying PLDPC-Hadamard block codes (PLDPC-BCs). Simulation results show that SC-PLDPC-CCs outperform their PLDPC-BC counterparts and other state-of-the-art low-rate codes in terms of bit error performance. For the rate-0.00295 SC-PLDPC-CC, a BER of $10^{-5}$ is achieved at $E_b/N_0 = -1.465$ dB.

Section II of this paper highlights the main differences/improvements of the current work compared with the previously studied PLDPC-HCs. Section III reviews the structures of related block codes and spatially coupled codes. Section IV introduces the structure and encoding process of SC-PLDPC-CCs; describes a pipeline decoding...
strategy combined with layered scheduling for decoding SC-PLDPCH-CCs, and analyzes its latency and complexity. Also, a layered PEXIT chart method is proposed to evaluate the threshold of SC-PLDPCH-TDCs/SC-PLDPCH-CCs efficiently and a GA is proposed to optimize protomatrices for SC-PLDPCH-TDCs/SC-PLDPCH-CCs. Section V presents the thresholds and optimized protomatrices of SC-PLDPCH-CCs with different code rates. It also compares the simulated BER results of the SC-PLDPCH-CCs with those of the underlying protograph-based PLDPC-Hadamard block codes (PLDPCH-BCs) and other state-of-the-art low-rate codes. Finally, Section VI presents some concluding remarks.

II. MAIN DIFFERENCES/IMPROVEMENTS BETWEEN THIS WORK AND PREVIOUS WORKS

The major differences/improvements between the current work and the previously studied PLDPC-HCs [16], [20] are as follows.

1) The PLDPC-HCs in [16] and [20] and the codes being investigated in this work are constructed based on protographs with the SPC check nodes (SPC-CNs) replaced by Hadamard constraints. However, the PLDPC-HCs studied in [16] and [20] are block codes while the codes being investigated in this work are formed by spatially coupling these block codes.

2) Though PEXIT algorithms have been applied for analyzing the codes in [16] and in this work, a layered PEXIT algorithm is proposed in this work to improve the convergence rate of the original PEXIT algorithm in [16]. Also, a more efficient way of evaluating the extrinsic MI of the Hadamard CNs is applied in the algorithm here compared with [16].

3) A GA is proposed in this work to search for spatially-coupled PLDPC-HCs with good theoretical thresholds.

4) We estimate the thresholds of SC-PLDPCH-CCs based on the thresholds of SC-PLDPCH-TDCs with large coupling lengths.

5) The PLDPC-Hadamard sub-decoders in [20] and the processors in this work are of similar structures. All of them apply layered decoding algorithms for decoding the PLDPC-HCs. However, the structure of the PLDPC-Hadamard sub-decoders in [20] is more or less fixed for a given code design; and the number of decoding iterations affects the error performance of the code, the decoding latency and throughput. The pipeline decoder described in this work consists of a series of processors, the number of which can vary; and the number of processors in the decoder affects the error performance of the code, the decoding latency, the hardware complexity and throughput.

6) The SC-PLDPCH-CCs in this work can achieve a better error performance and throughput than the PLDPCH-BCs [16] with a higher hardware requirement.

III. BACKGROUND

A. PLDPC Block Codes

A protograph consists of a set of \( m \) SPC-CNs, a set of \( n \) (\( n > m \)) protograph variable nodes (P-VNs), and a set of edges connecting the SPC-CNs to the P-VNs [17]. The corresponding protomatrix can be denoted by \( B = \{ b(i,j) \} \) in which each row in \( B \) corresponds to a SPC-CN; each column corresponds to a P-VN; and \( b(i,j) \) corresponds to the number of edges connecting the \( i \)-th SPC-CN and the \( j \)-th P-VN. The code rate of a PLDPC block code equals \( R_{\text{PLDPC-BC}} = 1 - \frac{m}{n} \). A two-step lifting method (with factors \( z_1 \) and \( z_2 \)) can be used to lift the protomatrix and hence to construct the parity-check matrix of a PLDPC code [35]. After the first lifting, all entries in the lifted matrix are either “0” or “1”. The second lifting procedure aims to construct a parity-check matrix with a quasi-cyclic structure so as to simplify encoder and decoder designs.

B. LDPC-Hadamard Codes and PLDPC-Hadamard Codes

A Hadamard code with an order \( r \) has a code length of \( q = 2^r \) and can be obtained by a Hadamard matrix \( H_q \) of size \( q \times q \). \( H_q \) can be recursively generated using [9], [14]

\[
\pm h_q = \pm \{ h_j \} = \left[ \pm H_{q/2} \pm H_{q/2} \right] \quad \text{where } H_1 = [\pm 1];
\]

\[
\pm h_j = \pm h_{0,j} \pm h_{1,j} \pm h_{2,j} \cdots \pm h_{2^r-1,j} \quad (j = 0, 1, \ldots , q - 1)
\]

represents the XOR operator. When the Hadamard order \( r \) is an even number, these \( H_q \) matrices for \( r = 2^{2m} \), \( 2^{2m-1} \) for \( m \) even and odd, respectively, can be obtained by \( H_q = H_{2^m} \). A two-step lifting method (with factors \( z_1 \) and \( z_2 \)) can be used to lift the protomatrix and hence to construct the parity-check matrix of a PLDPC code [35]. After the first lifting, all entries in the lifted matrix are either “0” or “1”. The second lifting procedure aims to construct a parity-check matrix with a quasi-cyclic structure so as to simplify encoder and decoder designs.
C. Spatially Coupled LDPC Codes

1) LDPC Convolutional Codes: The parity-check matrix $H_{CC}$ of an LDPC-CC [22] is semi-infinite and structurally repeated. $H_{CC}$ can be written as in (2), as shown at the bottom of the next page, where each $H_i(t)$ $(i = 0, 1, \ldots, m_s)$ is an $M \times N$ component parity-check matrix, $t$ denotes the time index, and $m_s$ is the syndrome former memory. Each codeword $c$ satisfies $cH_{CC}^T = 0$, where $0$ is a semi-infinite zero vector.

2) Spatially Coupled PLDPC Codes: SC-PLDPC codes are constructed based on underlying PLDPC block codes (PLDPC-BCs). We denote $W$ as the coupling width (equivalent to the aforementioned syndrome former memory $m_s$) and $L$ as the coupling length. Based on the $m \times n$ protomatrix $B$ of an underlying PLDPC-BC, an edge-spreading procedure [36] can be first used to obtain $W + 1$ split protomatrices $B_i$ $(i = 0, 1, \ldots, W)$ under the constraint $B = \sum_{i=0}^{W} B_i$. Then $L$ sets of such protomatrices are coupled to construct an SC-PLDPC code [25]. When the $L$ sets of protomatrices are coupled and directly terminated, the resultant protomatrix equals (3), as shown at the bottom of the next page. Such code is called an SC-PLDPC terminated code (SC-PLDPC-TDC) and its code rate equals $R_{SC-PLDPC-TDC} = 1 - \frac{L+W}{L} (1 - R_{PLDPC-BC})$ where $R_{PLDPC-BC} = 1 - \frac{n}{m}$ is the code rate of its underlying block code. When the protograph of a spatially coupled code is terminated with “end-to-end” connections, the corresponding code is called SC-PLDPC tail-biting code (SC-PLDPC-TBC), whose protomatrix can be written as (4), shown at the bottom of the next page. The code rate $R_{SC-PLDPC-TBC}$ of an SC-PLDPC-TBC is the same as that of its underlying block code, i.e., $R_{SC-PLDPC-TBC} = \frac{m_c - m_l}{m_c} = R_{PLDPC-BC}$. By extending the coupling length $L$ of an SC-PLDPC-TDC to infinity, an SC-PLDPC-CC is formed. The semi-infinite protomatrix of an SC-PLDPC-CC is given by (5), as shown at the bottom of the next page.

IV. SPATIALLY COUPLED PLDPC-HADAMARD CONVOLUTIONAL CODES

In this section, we show the details of our proposed SC-PLDPC-CC. First, we show the way of constructing SC-PLDPC codes, including SC-PLDPC tail-biting code (SC-PLDPC-TBC), SC-PLDPC terminated code (SC-PLDPC-TDC) and SC-PLDPC-CC, from its block code counterpart. Second, we briefly explain the encoding process of SC-PLDPC-CCs. Third, we describe an efficient decoding algorithm for SC-PLDPC-CC, which combines the layered decoding used for decoding PLDPC-BC [20] and the pipeline decoding used for decoding SC-PLDPC-CC [38]. Fourth, we propose a layered PEXIT algorithm for evaluating the theoretical threshold of an SC-PLDPC-TDC, which is then used to approximate the threshold of the corresponding SC-PLDPC-CC. Fifth, we propose a GA to optimize the SC-PLDPC-TDC/SC-PLDPC-CC designs based on a given PLDPC-BC.

A. Code Construction

SC-PLDPC codes are constructed in a similar way as SC-PLDPC codes. We also denote the coupling width as $W$ and coupling length as $L$ in an SC-PLDPC code. Given a PLDPC-BC with a protomatrix $B$, we apply the edge-spreading procedure to split $B$ into $W + 1$ protomatrices $B_i$ $(i = 0, 1, \ldots, W)$ under the constraint $B = \sum_{i=0}^{W} B_i$. Then we couple $L$ sets of these matrices to construct the protomatrix of an SC-PLDPC code. Similar to the SC-PLDPC codes described in Section III-C, an SC-PLDPC-TDC is formed if the coupled matrices are directly terminated; an SC-PLDPC-TBC is formed if the coupled matrices are connected end-to-end; and an SC-PLDPC-CC is formed if the coupling length $L$ becomes infinite. Since the
constructed protomatrices only represent the connections between P-VNs and H-CNs, SC-PLDPCH codes have protomatrix structures similar to those of SC-PLDPC codes, i.e., (3) for SC-PLDPCH-TDC; (4) for SC-PLDPCH-TBC; and (5) for SC-PLDPCH-CC. Unlike the protographs of SC-PLDPC codes which consist of P-VNs and SPC-CNs, the protographs of SC-PLDPCH codes contain P-VNs and H-CNs connected with some appropriate D1H-VNs. Fig. 1(b) shows the protograph of an SC-PLDPCH-CC derived from the PLDPCH-BC in Fig. 1(a). SC-PLDPCH codes can be constructed from the coupled protographs using the lifting process. Assuming that $B$ has a constant row weight of $d$

$$
H_{CC} = \begin{bmatrix}
H_0(1) & H_1(1) & \cdots & H_m(1) \\
H_0(2) & H_1(2) & \cdots & H_m(2) \\
\vdots & \vdots & \ddots & \vdots \\
H_0(t) & H_1(t) & \cdots & H_m(t)
\end{bmatrix}
$$

(2)

$$
B_{SC-PLDPC-TDC} = \begin{bmatrix}
B_0 \\
B_1 & B_0 \\
\vdots & \ddots & \ddots \\
B_W & \cdots & B_0 \\
B_W & B_1 & \cdots \\
\vdots & \ddots & \ddots \\
B_W & \cdots & B_0 \\
B_W & \cdots & B_0 \\
B_W & \cdots & B_0 \\
B_W & \cdots & B_0
\end{bmatrix}
$$

(3)

$$
B_{SC-PLDPC-TBC} = \begin{bmatrix}
B_0 \\
B_1 & B_0 \\
\vdots & \ddots & \ddots \\
B_W & \cdots & B_0 \\
B_W & B_1 & \cdots \\
\vdots & \ddots & \ddots \\
B_W & \cdots & B_0 \\
B_W & \cdots & B_0 \\
B_W & \cdots & B_0
\end{bmatrix}
$$

(4)

$$
B_{SC-PLDPC-CC} = \begin{bmatrix}
B_0 \\
B_1 & B_0 \\
\vdots & \ddots & \ddots \\
B_W & \cdots & B_0 \\
B_W & B_1 & \cdots \\
\vdots & \ddots & \ddots \\
B_W & \cdots & B_0
\end{bmatrix}
$$

(5)
and hence an order-\(r\) (\(= d - 2\)) Hadamard code is used (recall \(r\) is even), it can be readily shown that the code rates of the SC-PLDPC codes are as follows. For SC-PLDPC-TDCs the code rate equals 
\[ \frac{R_{\text{SC-PLDPC-TDC}}}{R_{\text{PLDPC-Hadamard}}} = \frac{nL-m(L+W)}{n-m+1(\frac{M}{2})} = \frac{n-m+(\frac{M}{2})}{n-m+1(\frac{M}{2})} \]. For SC-PLDPC-TBCs and SC-PLDPC-CCs, their code rates are the same as the block code counterparts, i.e., 
\[ R_{\text{SC-PLDPC-TBC}} = R_{\text{SC-PLDPC-CC}} = \frac{n-m+1(\frac{M}{2})}{n-m+1(\frac{M}{2})}. \]

B. Encoding of SC-PLDPC-CC

From this point forward and unless otherwise stated, we focus our study on SC-PLDPC-CC. We also assume that the row weight of \(B\) equals \(d = r + 2\) and is even. After performing a two-step lifting process on (5), we obtain the semi-infinite parity-check matrix of an SC-PLDPC-CC in Fig. 2. Denoting the two lifting factors by \(z_1\) and \(z_2\), each \(H_t\) (\(i = 0, 1, \ldots, W\)) which is the lifted \(B_t\) has a size of \(M \times N = m z_1 z_2 \times n z_1 z_2\). At time \(t\), \(M - N\) information bits denoted by \(b(t) \in \{0, 1\}^{M-N}\) are input to the SC-PLDPC-CC encoder. The output of the SC-PLDPC-CC encoder contains \(N\) coded bits corresponding to P-VNs, which are denoted by \(P(t)\); and \(M(2^r-r-2)\) Hadamard parity-check bits corresponding to D1H-VNs, which are denoted by \(D(t)\). Referring to Fig. 2, we generate the output bits as follows.

1) \(t = 1\): Given \(b(1)\), \(P(1)\) is generated based on the first block row of \(H_{\text{SC-PLDPC-CC}}\), i.e., \(H_0\). Moreover, 
\[ D(1) = \text{computed based on } [0 \ldots 0 P(1)] \text{ and the structure } [H_W \ldots H_1 H_0], \]

2) \(t = 2\): Given \(b(2)\) and \(P(1)\), \(P(2)\) is generated based on the second block row of \(H_{\text{SC-PLDPC-CC}}\), i.e., \([H_1 H_0]\). Moreover, 
\[ D(2) = \text{computed based on } [0 \ldots 0 P(1) P(2)] \text{ and the structure } [H_W \ldots H_1 H_0]. \]

3) \(t \leq W\): Given \(b(t)\) and \([P(1) P(2) \ldots P(t-1)]\), \(N\) coded bits \(P(t)\) are generated based on the \(t\)-th block row of \(H_{\text{SC-PLDPC-CC}}\), i.e., \([H_{t-1} \ldots H_1 H_0]\).
\[ D(t) \text{ corresponding to the } M(2^r-r-2) \text{ D1H-VNs are computed based on } [0 \ldots 0 P(1) \ldots P(t)] \text{ and the structure } [H_W \ldots H_1 H_0]. \]

4) \(t > W\): Given \(b(t)\) and \([P(t-W) \ldots P(t-1)]\), \(P(t)\) is generated based on the \(t\)-th block row of \(H_{\text{SC-PLDPC-CC}}\), i.e., \([H_W \ldots H_1 H_0]\). Then, \(D(t)\) is computed based on \([P(t-W) \ldots P(t-1) P(t)]\) and the structure \([H_W \ldots H_1 H_0]\). The constraint length of the SC-PLDPC-CC therefore equals \((W+1)N + M(2^r-r-2)\).

Remarks: The values of \(D(t)\) are generated during the encoding corresponding to the \(t\)-th block row. They are not needed for generating other \(D(t')\) where \(t \neq t'\). When \(t \leq W\),

C. Pipeline Decoding

1) Decoding Structure: At the receiving end, we receive channel observations regarding the coded bits \(P(t)\) (corresponding to P-VNs) and Hadamard parity-check bits \(D(t)\) (corresponding to D1H-VNs). We denote the log-likelihood-ratio (LLR) values corresponding to \(P(t)\) by \(L^p_{\text{ch}}(t)\) and the LLR values corresponding to \(D(t)\) by \(L^d_{\text{ch}}(t)\). We consider a pipeline decoder which consists of \(I\) identical message-passing processors [22], [38], [39]. Each processor is a PLDPC-Hadamard block sub-decoder corresponding to \([H_W \ldots H_1 H_0]\). Thus, each processor operates on \((W+1)I\) sets of P-VNs and one set of D1H-VNs each time, i.e., a total of \(N(W+1)\) P-VNs and \(M(2^r-r-2)\) D1H-VNs (when \(r\) is even). Hence the pipeline decoder operates on \((W+1)I\) sets of P-VNs and \(I\) sets of D1H-VNs each time. Each processor (sub-decoder) can apply either the standard decoding algorithm or the layered decoding algorithm to compute/update the a posteriori probability in LLR form (APP-LLR) for the coded bits \(P(t)\) and the related extrinsic LLR information. Here, we apply the layered decoding algorithm [20] in each of these PLDPC-Hadamard block sub-decoders.

We denote the APP-LLR values of the coded bits \(P(t)\) by \(L^p_{\text{app}}(t)\). Referring to Fig. 3, \([L^p_{\text{ch}}(1), L^d_{\text{ch}}(1)]\) is first input to the pipeline decoder and Processor #1 updates the APP-LLR of all P-VNs inside, i.e., \(L^p_{\text{app}}(1)\). Also, extrinsic LLR information is updated and stored in the processor but is not depicted in the figure. Then, \([L^p_{\text{ch}}(2), L^d_{\text{ch}}(2)]\) is input to the pipeline decoder while \([L^p_{\text{app}}(1), L^d_{\text{app}}(1), L^d_{\text{app}}(1)]\) and related extrinsic LLR information are shifted to the left in the decoder. Processor #1 updates the APP-LLRs of all P-VNs inside, i.e., \(L^p_{\text{app}}(2)\) and \(L^p_{\text{app}}(2)\). Again, extrinsic LLR information is updated and stored in the processor but is not depicted. Subsequently, \([L^p_{\text{ch}}(3), L^d_{\text{ch}}(3)]\) \((t = 3, 4, \ldots)\) are input into the decoder one set by one set. Every time, all
sets of LLRs inside the decoder are shifted to the left by one “$H_i$” block, and all APP-LLRs of all P-VNs inside the different $I$ processors are updated. When \( \{L_{ch}^{P}((W+1)I+t'), L_{ch}^{D}((W+1)I+t')\} \) \( (t' = 1, 2, \ldots) \) is input to the decoder, the APP-LLRs \( L_{app}^{P}(t') \) are output and the values of the coded bits \( P(t') \) are determined.

2) Latency and Complexity Analysis: In this section, we compare the latency and complexity of SC-PLDPCH-CC decoder and PLDPCH-BC decoder [20], [40]. We define “latency” as the time interval between (i) the LLR information entering a decoder and (ii) the corresponding decoded P-VNs output from the decoder. We assume a two-step lifting process with the factors $z_1$ and $z_2$ applied to lift the base matrices of SC-PLDPCH-CC and PLDPCH-BC. Moreover, layered decoding is used in both cases. For a given PLDPCH-BC, we denote the time taken to complete one decoding iteration by $T_{\gamma}$, which has been shown to be proportional to the number of layers, i.e., $mz_2$ [40]. Denoting the maximum number of iterations by $I_{BC}$, the time delay (i.e., latency) for decoding one PLDPCH-BC codeword equals $T_{PLDPCH-BC} = I_{BC}T_{\gamma}$.

Based on the same protomatrix as the PLDPCH-BC, we use the edge spreading approach to construct an SC-PLDPCH-CC. We exploit the pipeline decoding method in the previous section with $I$ identical processors, each of which applies the layered decoding algorithm. Moreover, it can be readily shown that the design used to implement the aforementioned PLDPCH-BC decoder [40] can be slightly modified to implement these processors. As each processor needs to update $mz_2$ layers of information whenever a new set of \( \{L_{ch}^{P}(t), L_{ch}^{D}(t)\} \) \( (t = 1, 2, \ldots) \) enters the decoder, the time taken is the same as for the PLDPCH-BC decoder to complete one iteration, i.e., $T_{\gamma}$. As shown in Fig. 3 and discussed in the previous section, the APP-LLRs \( L_{app}^{P}(t') \) are output from the decoder when \( \{L_{ch}^{P}((W+1)I+t'), L_{ch}^{D}((W+1)I+t')\} \) is input to the pipeline decoder. Thus, the total time delay (i.e., latency) for decoding one set of input equals $T_{SC-PLDPCH-CC} = (W+1)IT_{\gamma}$. In addition, $(W+1)$ sets of LLRs corresponding to different $t'$s are being processed by the pipeline decoder, as compared to only one set of LLRs being processed by the PLDPCH-BC decoder at any time.

In summary, when a PLDPCH-BC and an SC-PLDPCH-CC are “derived” from the same protomatrix of size $m \times n$ with the same lifting factors $z_1$ and $z_2$, the PLDPCH-BC decoder and each processor in an SC-PLDPCH-CC pipeline decoder processes the $mz_2$ layers in layered decoding with the same time delay. The SC-PLDPCH-CC pipeline decoder consists of $I$ processors, each having a similar structure as a PLDPCH-BC decoder; and requires $(W+1)I$ times memory storage compared with a PLDPCH-BC decoder. The latencies of the SC-PLDPCH-CC pipeline decoder and PLDPCH-BC decoder are $T_{SC-PLDPCH-CC} = (W+1)IT_{\gamma}$ and $T_{PLDPCH-BC} = I_{BC}T_{\gamma}$, respectively. Under the condition that the two latencies are identical, i.e., $T_{SC-PLDPCH-CC} = T_{PLDPCH-BC}$ or $(W+1)I = I_{BC}$, the SC-PLDPCH-CC pipeline decoder achieves a $(W+1)$ times throughput compared with the PLDPCH-BC decoder.

3) SC-PLDPCH-CC and PLDPCH-BC Under the Same Constraint Length/Blocklength: In this section, we further compare the case when the constraint length of a SC-PLDPCH-CC is the same as the blocklength of a PLDPCH-BC. We assume a two-step lifting process with the factors $z_1$ and $z_2$ applied to lift the $m \times n$ base matrices of SC-PLDPCH-CC and, $z'_1$ and $z'_2$ to lift the same size base matrix of PLDPCH-BC. The constraint length of the SC-PLDPCH-CC is $L_{CC} - BL = (W+1)N + M(2^r - r - 2) = (W+1)z_1z_2n + z_1z_2m(2^r - r - 2)$ and the blocklength of the PLDPCH-BC is $L_{BC} - BL = z'_1z'_2n + z'_1z'_2m(2^r - r - 2)$. For simplicity, we let $z_1 = z'_1$. When $L_{CC} - BL = L_{BC} - BL$, we
have \( \delta_{z_2} \triangleq z_2'/z_2 = 1 + \frac{W}{1+(2 r - 2)\sigma_0^2} \). The result implies that \( z_2' \) is strictly larger than \( z_2 \). For example, when \( W = 1, m = 7, n = 11, r = 4 \), we obtain \( \delta_{z_2} = z_2'/z_2 = 1.136 \). When the number of H-CN’s in one layer is increased by a factor of \( \delta_{z_2} \), the time taken to complete one decoding iteration is increased by the same factor. In order to maintain the same decoding latency, the maximum number of iterations for decoding one PLDPCH-BC codeword should be reduced by the same factor, i.e., reduced from \( I_{BC} \) (see above section) to \( I'_{BC} = I_{BC}/\delta_{z_2} \).

D. Layered EXIT Algorithm

In [16], a low-complexity EXIT chart technique has been proposed to evaluate the theoretical threshold of PLDPCH-BCs. The thresholds of SC-PLDPCH-CCs are expected to be comparable to those of their underlying block codes. However, direct analysis of SC-PLDPCH-CCs with infinite length is very complicated and time-consuming, which is not conducive to the optimal design of the codes in the next section, i.e., Section IV-E.

As mentioned in Section IV-A, under the same set of split protomatrices \( \{B_0, B_1, \ldots, B_W\} \), SC-PLDPCH-CCs can be obtained by extending the coupling length \( L \) of SC-PLDPCH-TDCs to infinity. An SC-PLDPCH-TDC can be treated as a PLDPCH-BC with a large size and hence its threshold can be evaluated using the original EXIT chart method in [16]. For SC-PLDPCH-TDCs constructed with different split protomatrices, the original EXIT chart method will generate different (i.e., distinguishable) thresholds.\(^3\) For example, Fig. 4 shows the thresholds for two different SC-PLDPCH-TDCs using the original EXIT chart method. We set \( W = 1 \). Based on the optimal protomatrix \( B \) in [16], the split protomatrices of SC-PLDPCH-TDC \( #1 \) are shown as (6), at the bottom of the next page, and the split protomatrices of SC-PLDPCH-TDC \( #2 \) are shown as (7), at the bottom of the next page, where \( B_0 + B_1 = B \). From Fig. 4, we can observe that the thresholds of two SC-PLDPCH-TDCs are distinguishable and improved (reduced) as the coupling length \( L \) increases. Moreover, the improvement diminishes as \( L \) becomes large. Thus, threshold saturation is observed. Subsequently, we select the protomatrices with good thresholds (i.e., low \( E_b/N_0 \)) to construct the corresponding SC-PLDPCH-CCs.

To improve the convergence rate of the original EXIT chart method, we propose a layeredEXIT method to analyze the SC-PLDPCH-TDCs. We first define \( I_{av}(i,j) \) as the \( a \) priori mutual information (MI) from the \( i \)-th P-VN to the \( j \)-th P-\( \text{HCN} \); \( I_{cv}(i,j) \) as the extrinsic MI from the \( i \)-th P-\( \text{VN} \) to the \( j \)-th P-\( \text{HCN} \); \( I_{ah}(k) \) as the \( a \) priori MI of the \( k \)-th information bit in the \( i \)-th P-\( \text{HCN} \); \( I_{ch}(k) \) as the extrinsic MI of the \( k \)-th information bit in the \( i \)-th P-\( \text{HCN} \); \( I_{app}(j) \) as the \( a \) posteriori MI of the \( j \)-th P-\( \text{VN} \); \( \sigma_{app}(j) \) as the \( a \) posteriori information of the \( j \)-th P-\( \text{VN} \); and \( \sigma_{temp}(j) \) as the temporary value of \( \sigma_{app}(j) \). We also assume that the channel LLR value \( L_{ch} \) follows a normal distribution \( \mathcal{N}(\sigma_{ch}^2/2, \sigma^2) \), where \( \sigma_{ch}^2 = 8R \cdot E_b/N_0 \). \( R \) is the code rate of SC-PLDPCH-TDC, and \( E_b/N_0 \) is the bit-energy-to-noise-power-spectral-density ratio. When the output MI from P-\( \text{VN} \) or P-\( \text{HCN} \) processors is \( I_{MI} \), we suppose that the corresponding LLR values of the extrinsic information obey a normal distribution \( \mathcal{N}(\pm \sigma^2/2, \sigma^2) \). The relationship between \( I_{MI} \) and \( \sigma \) can be approximately computed by the functions \( I_{MI} = J(\sigma) \) and \( \sigma = J^{-1}(I_{MI}) \) in [41] and [42].

Given a coupling width \( W \) and a set of protomatrices \( \{B_0, B_1, \ldots, B_W\} \) each of size \( m \times n \), we use these protomatrices to construct an SC-PLDPCH-TDC with the coupling width \( W \) and an appropriate coupling length \( L \). The protomatrix of the SC-PLDPCH-TDC \( \{B_{SCPLDPCH-TDC} \} \) therefore has a size of \( m(L + W) \times nL \). In (8), as shown at the bottom of the next page, the protomatrix \( B \) of size \( m \times n = 3 \times 4 \) is split into \( B_0 \) and \( B_1 \) of the same size assuming \( W = 1 \). Based on \( B_0 \) and \( B_1 \), an SC-PLDPCH-TDC with \( W = 1 \) and \( L = 2 \) is constructed in (9), as shown at the bottom of the next page, and has a size of \( m(L + W) \times nL = 9 \times 8 \). Note that in computing the threshold of the SC-PLDPCH-TDC, a larger \( L \), e.g., \( L = 10 \), will be used for threshold evaluation such that \( nL > m(L + W) \).

Similar to the decoding strategy in the layered decoder [20], the layered EXIT method proceeds as follows.

1) Set the initial \( E_b/N_0 \) in dB (i.e., \( E_b/N_0 \text{(dB)} \)).\(^4\)
2) Set the maximum number of iterations \( N_{\text{max}} = 150 \).
3) Compute \( \sigma_{L_{ch}} = (8R \cdot 10^{E_b/N_0 \text{(dB)}}/10)^{1/2} \) for \( L_{ch} \), and set \( \sigma_{app}(j) = \sigma_{L_{ch}} \) for \( j = 1, 2, \ldots, nL \).

\(^3\)Note that when \( r \) is odd, \( \delta_{z_2} = 1 + \frac{W}{1+(2 r - 2)\sigma_0^2} \).

\(^4\)The initial values will be different according to different codes (corresponding to different Hadamard order \( r \)). In our optimization design, given \( W = 1 \) and \( L = 10 \), initial \( E_b/N_0 \) is \( -0.30 \) dB for \( r = 4 \), \(-0.40 \) dB for \( r = 5 \), \(-0.80 \) dB for \( r = 8 \) and \(-0.85 \) dB for \( r = 10 \), respectively.
4) For \( i = 1, 2, \ldots, m(L + W) \) and \( j = 1, 2, \ldots, nL \), set \( I_{av}(i, j) = 0. \)
5) Set the iteration number \( N_{it} = 1. \)
6) Set \( it = 1. \)
7) For \( j = 1, 2, \ldots, nL \), subtract \( b_{TDC}(i, j) \) from \( (\sigma_{app}^{-1}(J) I_{av}(i, j)) \) and then compute \( \sigma_{temp}(j) \) using (10), as shown at the bottom of the next page.
8) For \( j = 1, 2, \ldots, nL \), compute (11), as shown at the bottom of the next page, if \( b_{TDC}(i, j) > 0 \); otherwise set \( I_{ev}(i, j) = 0. \)

Taking the first row of the \( 9 \times 8 \) protomatrix \( B_{SC-PLDPCH-TDC} \) in (9) as an example, we obtain the \( 1 \times 8 \) vector \( b_{TDC}(1, :) \) shown in (12), at the bottom of the next page. After analyzing the MI of the P-VNs, the corresponding \( 1 \times 8 \) MI vector \( I_{ev}(1, :) \) is shown in (13), at the bottom of the next page.

9) Convert the \( 1 \times nL I_{ev}(i, :) \) MI vector into a \( 1 \times d I_{ach} \) MI vector by eliminating the 0 entries and repeating \( b_{TDC}(i, j) \) times the entry \( I_{ev}(i, j) \).

**Remark:** Each row weight of a protomatrix \( B \) for the underlying PLDPCH-BC equals \( d = r + 2 \) [16] and hence each row corresponds to a \( r = d - 2 \) Hadamard code. However, due to the structure of SC-PLDPCH-TDCs, the first and last \( Wm \) rows in \( B_{SC-PLDPCH-TDC} \) contain only part of the structure \( [B_W \ldots B_1 B_0] \), where \( B = \sum_{i=0}^{W} B_i \) and thus the row weight could be less than \( d \). For simplicity and uniformity, we compute the MI values of \( r = d - 2 \) Hadamard code for each row (i.e., each H-CN). For the first/last \( Wm \) rows in \( B_{SC-PLDPCH-TDC} \), when their row weight \( d_1 \) is less than \( d \), the first/last \( d - d_1 \) MI values in \( I_{ach} \) are set to 1.5 For example, the row weight of the underlying protomatrix \( B \) (8) equals \( d = 6 \) while the row weight of the \( 1 \times 8 \) vector \( b_{TDC}(1, :) \) equals \( d_1 = 3 < d \). Hence, when converting \( 1 \times 8 I_{ev}(1, :) \) into the \( 1 \times 6 I_{ach} \) MI vector, the first \( d - d_1 \) values, i.e., \( [I_{ach}(1) I_{ach}(2) I_{ach}(3)] \) are set to 1, as shown in (14), at the bottom of the next page.

10) Based on \( \sigma_{ach} \) and the \( d \) entries in \( I_{ach} \), we use the Monte Carlo method in [16] to generate \( d \) extrinsic MI values, i.e., a \( 1 \times d \) MI vector \( I_{ach}. \) For more details of the method, please refer to the Appendices in [16]. We use the same symbol definitions as in [16]. Hence \( \nu(x) \) is shown in (15), as shown at the bottom of the next page.

\[ B_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}; \quad B_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 2 & 0 \end{bmatrix} \]

\[ B_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}; \quad B_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 2 \end{bmatrix} \]

\[ B = \begin{bmatrix} 2 & 0 & 2 & 2 \\ 0 & 2 & 2 & 2 \\ 3 & 2 & 0 & 1 \end{bmatrix}, \quad B_0 = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 2 & 0 & 1 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 1 & 1 \\ 2 & 0 & 0 & 0 \end{bmatrix} \]

\[ B_{SC-PLDPCH-TDC} = \begin{bmatrix} B_0 & 0 \\ B_1 & B_0 \\ 0 & B_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 1 & 0 & 0 & 2 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 2 & 0 & 0 & 0 & 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix} \]

5MI value equal to 1 means that the corresponding MI is known and does not provide any new information in the analysis.
6When MI value equals 1, Monte Carlo method will generate the corresponding LLL values with large absolute values. In our analysis, we set them as \( \pm 100. \)
Convert the MI vector $I_{eh}$ is shown in (17), at the bottom of the page.

11) Convert the $1 \times d$ MI vector into a $1 \times nL$ MI vector. For $j = 1, 2, \ldots, nL$, if $b_{TDC}(i, j) > 0$, set the value of $I_{av}(i, j)$ as the average of the corresponding $b_{TDC}(i, j)$ MI values in $I_{eh}$; otherwise set $I_{av}(i, j) = 0$.

In the above example, we omit the first $d - d_1 = 3$ MI values in $I_{eh}$, i.e., $[I_{eh}(1) I_{eh}(2) I_{eh}(3)]$, and use the remaining $d_1 = 3$ MI values, i.e., $[I_{av}(4) I_{av}(5) I_{av}(6)]$ to compute the $I_{av}(1, :)$. MI vector given in (18), as shown at the bottom of the page.

12) For $j = 1, 2, \ldots, nL$, use the “new” extrinsic information $I_{av}(i, :)$ to update the a posteriori information $\sigma_{app}(j)$ by (19), as shown at the bottom of the page.

![Image](59x237 to 475x238)

\[
\sigma_{temp}(j) = \sqrt{(\sigma_{app}(j))^2 - b_{TDC}(i, j) \times (J^{-1}(I_{av}(i, j)))^2} \forall j
\]  

\[
I_{ev}(i, j) = J \left( \sqrt{(\sigma_{app}(j))^2 - (J^{-1}(I_{av}(i, j)))^2} \right) \forall j
\]  

\[
b_{TDC}(1, :) = [1 \ 0 \ 0 \ 2 \ 0 \ 0 \ 0]
\]  

\[
I_{ev}(1, :) = [I_{ev}(1, 1) \ 0 \ 0 \ I_{ev}(1, 4) \ 0 \ 0 \ 0]
\]  

\[
I_{ah} = [I_{ah}(1) \ I_{ah}(2) \ I_{ah}(3) \ I_{ah}(4) \ I_{ah}(5) \ I_{ah}(6)] = [1 \ 1 \ 1 \ I_{ev}(1, 1) \ I_{ev}(1, 4) \ I_{ev}(1, 4)]
\]  

\[
I_{eh}(k) = \frac{1}{2} \sum_{x \in \{0, 1\}} \int_{-\infty}^{\infty} p_e(\xi | X = x) \log_2 2 \cdot p_e(\xi | X = x^0) + p_e(\xi | X = x^1) d\xi
\]  

\[
\approx 1 - \frac{1}{w} \sum_{a=1}^{w} \log_2 \left( 1 + e^{-(1 - 2U(a, k)) \times V(a, k)} \right)
\]  

\[
I_{eh} = [I_{eh}(1) \ I_{eh}(2) \ I_{eh}(3) \ I_{eh}(4) \ I_{eh}(5) \ I_{eh}(6)]
\]  

\[
I_{av}(1, :) = [I_{av}(1, 1) \ 0 \ I_{av}(1, 3) \ 0 \ 0 \ 0 \ 0]
\]  

\[
= [I_{eh}(4) \ 0 \ \frac{1}{2} \sum_{k=0}^{6} I_{eh}(k) \ 0 \ 0 \ 0 \ 0]
\]  

\[
\sigma_{app}(j) = \sqrt{(\sigma_{temp}(j))^2 + b_{TDC}(i, j) \times (J^{-1}(I_{av}(i, j)))^2} \forall j
\]
algorithms do not converge at \( E_b/N_0 = -0.45 \) dB when the maximum number of iterations is set to \( N_{\text{max}} = 150 \) for the layered PEXIT chart method, and is set to \( 2N_{\text{max}} = 300 \) for original PEXIT chart method. Thus both algorithms arrive at the same decoding threshold, i.e., \( (E_b/N_0)^* = -0.40 \) dB. The proposed layered PEXIT algorithm can therefore speed up the convergence rate when calculating the threshold of SC-PLDPCH-TDCs.

Remark: The difference between our proposed layered PEXIT algorithm and the shuffled EXIT algorithm \([43]\) are as follows.

- Given a protomatrix, our algorithm performs the analysis row by row, while the shuffled EXIT algorithm \([43]\) performs the analysis column by column.
- When analyzing SC-PLDPCH-TDCs, our algorithm computes MI values for Hadamard check nodes, while \([43]\) computes MI values for SPC-CNs.

### E. Optimizing Protomatrices Using Genetic Algorithm

To solve a problem based on GA \([44]\), a generation group is first created or initialized and a fitness function is developed to calculate the fitness value of each individual in the group. Based on the fitness values, some individuals are selected from the “parent” generation group to form an “offspring” generation group. To facilitate obtaining good solutions, the individuals having the best fitness values in the parent generation group will be kept in the offspring generation group. Crossover and mutation operations are further performed in the offspring generation group. Subsequently, the “offspring” generation group becomes the “parent” generation group. By repeating the generation cycles, GA has a high probability of arriving at the global optimal solution to the problem \([28]\).

Given a protomatrix \( B \) corresponding to a PLDPCH-BC, we propose a GA to systematically search for optimized sets of protomatrices \( \{B_0, B_1, \ldots, B_W\} \) (where \( B = \sum_{i=0}^W B_i \)) for the corresponding SC-PLDPCH-TDC. By selecting SC-PLDPCH-TDCs with good thresholds, our final objective is to design optimal SC-PLDPCH-CCs based on the same set of protomatrices. We denote

- \( K \) parent individuals \( \Phi^k \) \((k = 1, 2, \ldots, K)\) as \( K \) sets of \( W + 1 \) protomatrices, i.e., \( \Phi^k = \{B^k_0, B^k_1, \ldots, B^k_W\} \), each of which satisfies \( B = \sum_{i=0}^W B^k_i \);
- the fitness function as \( \psi(\cdot) \) and the fitness value of the \( k \)-th parent individual as \( f_k = \psi(\Phi^k) \);
- the probability of selecting the \( k \)-th parent individual as \( p_{sk} \);
- the probability of crossover as \( p_c \) and the probability of mutation as \( p_m \);
- \( K \) offspring individuals \( Y_k \) \((k = 1, 2, \ldots, K)\) as \( K \) sets of \( W + 1 \) protomatrices, i.e., \( Y_k = \{S^k_0, S^k_1, \ldots, S^k_W\} \), each of which satisfies \( B = \sum_{i=0}^W S^k_i \);
- \( m \times n \) as the size of \( B \), and hence also the size of \( B^k_i \) and \( S^k_i \) for all \( i \) and \( k \);
- \( N_g \) as the number of individuals having good fitness values.

Our proposed GA algorithm is described as follows.

1) Initialization: Set coupling width \( W_i \), coupling length \( L \), and the number of individuals \( K \) in each generation. Randomly generate \( K \) sets of \( \Phi^k = \{B^k_0, B^k_1, \ldots, B^k_W\}, k = 1, 2, \ldots, K \).

2) Computation of fitness values: For each \( \Phi^k \), we construct the corresponding SC-PLDPCH-TDC by coupling \( L \) sets of \( \Phi^k \). Then we apply our proposed layered PEXIT chart method in Section IV-D to analyze the threshold and convergence behavior of the SC-PLDPCH-TDC. Recall that for a specific \( E_b/N_0 \) value (in dB), the iteration number required to converge is given by \( N_0 \) and the maximum number of iterations is given by \( N_{\text{max}} \). Our proposed fitness function \( \psi(\cdot) \) takes both the number of successful convergences, denoted by \( N_c \), and the convergence rate \( N_{\text{it}} \) into consideration. We define \( \psi(\cdot) \) as

\[
\psi(\Phi^k) = f_k = \sum_{i=1}^{N_c} \left[ N_{\text{max}} - N_{\text{it},i}^k \right]
= N_c N_{\text{max}} - \sum_{i=1}^{N_c} N_{\text{it},i}^k
\]

where \( N_{\text{it},i}^k \) represents the number of iterations for the SC-PLDPCH-TDC corresponding to \( \Phi^k \) to converge in the \( i \)-th successful convergence \((i = 1, 2, \ldots, N_c)\). Note that \( \psi(\Phi^k) \) should give a larger value if the corresponding SC-PLDPCH-TDC converges faster and more times in the layered PEXIT algorithm.

In the example given in Table I, the layered PEXIT algorithm has converged at \(-0.30 \) dB, \(-0.35 \) dB and \(-0.40 \) dB, and hence \( N_c = 3 \). Using (20), the fitness value of \( \{B_0, B_1\} \) defined in (6) is therefore given by

\[
\psi(\{B_0, B_1\}) = 3 \times 150 - (80 + 104 + 121) = 145.
\]

3) Selection: Compute the fitness values \( f_k = \psi(\Phi^k) \) for the \( K \) parent individuals \( \Phi^k \) \((k = 1, 2, \ldots, K)\). Subsequently, we normalize \( f_k = (k = 1, 2, \ldots, K) \) to form \( k = 1, 2, \ldots, K \). Then the \( K \) offspring individuals \( Y_k \) \((k' = 1, 2, \ldots, K)\) are chosen from the parent generation group \( \Phi^k \) \((k = 1, 2, \ldots, K)\) as follows. First, the \( N_g \) individuals with the highest fitness values in the parent group are passed to the offspring group directly to fill \( Y_k \) \((k' = 1, 2, \ldots, N_g)\). Then to fill each of the remaining \( K - N_g \) offspring slots, i.e., \( Y_k \) \((k' = N_g + 1, N_g + 2, \ldots, K)\), a random parent individual \( \Phi^k \) is selected according to the probability \( p_{sk} \). These
We use binary phase-shift-keying (BPSK) modulation over an AWGN channel. Moreover, we apply the pipeline decoding with different number of processors to evaluate the error performance of the SC-PLDPCH-CCs found.

For each \( E_b/N_0 \) value, simulations are performed for at least 1000 sub-blocks with at least 100 sub-block errors. Then the corresponding BER is evaluated. Under comparable information lengths and code rates, we compare the BER performance of our proposed codes not only with that of the underlying PLDPCH-BC, but also with those of state-of-the-art codes, namely LDPC-Hadamard block code (LDPC-BC) [14], turbo-Hadamard code (THC) [9], irregular zigzag-Hadamard code (IRZHC) [45, 46] (whenever BER results are available). These long codes with good error performance under very low \( E_b/N_0 \) values can potentially be applied to deep space communications where the signal

V. SIMULATION RESULTS

Based on the optimized PLDPCH-BCs found in [16], we search for good SC-PLDPCH-CCs using the layered PEXIT algorithm and the GA proposed in Section IV-D and Section IV-E. We assume \( W = 1 \), and use \( L = 10 \) and \( N_{\text{max}} = 150 \) in the layered PEXIT method. We also set \( K = 30 \), \( N_g = 4 \), \( p_c = 0.8 \) and \( p_m = 0.6 \) in the GA. We use binary phase-shift-keying (BPSK) modulation over an
from the space probe to Earth is extremely weak; and to embed/transmit hidden information over an ordinary wireless communication link. For $r = 4$ and $r = 5$, we also compare the BER performance of PLDPCH-BCs and SC-PLDPCH-CCs under the same blocklength/constraint length.

### A. Rate-0.0494 and $r = 4$

Based on the $7 \times 11$ protomatrix $B$ of the optimized rate-0.0494 PLDPCH-BC [16], we apply GA and find two $7 \times 11$ protomatrices shown as (21), at the bottom of the next page, after 13 generations. The corresponding fitness value equals 554. Then we increase $L$ to 500 and the code rate of the SC-PLDPCH-TDC constructed by (21) is increased to about 0.0491, approaching that of underlying PLDPCH-BC [16]. Using the proposed layered PEXIT method with $N'_{\max} = 1000$ iterations, Table II lists the thresholds for the $r = 4$ SC-PLDPCH-TDC with $L = 500$, and the $r = 4$ PLDPCH-BC\(^7\) [16], respectively. The theoretical threshold of the terminated code with $L = 500$ is found to be $-1.35$ dB, which is approximated as the threshold of the SC-PLDPCH-CC constructed by (21). Note that the threshold is slightly lower (i.e., slightly better) than that of the PLDPCH-BC.

We use the lifting factors $z_1 = 32$ and $z_2 = 512$ to expand the protomatrix such that the sub-block length of the SC-PLDPCH-CC equals 1,327,104, which is identical to the code length of the PLDPCH-BC with $z_1 = 32$ and $z_2 = 512$, i.e., $N + M(2^r - r - 2) = 1,327,104$. Moreover, the information length in each sub-block/block is 65536 for both codes. The BER performance of the SC-PLDPCH-CC with different number of processors $I$ used in pipeline decoding is shown in Fig. 6(a). We observe that the decoder with $I = 80$ processors in pipeline decoding achieves a BER of $10^{-5}$ at about $E_b/N_0 = -1.24$ dB, which outperforms that with (i) $I = 75$ by about 0.015 dB, (ii) $I = 70$ by about 0.03 dB, and (iii) $I = 60$ by about 0.06 dB. The gaps of the SC-PLDPCH-CC ($I = 80$ and BER of $10^{-5}$) to the PEXIT threshold ($-1.35$ dB) and to the ultimate Shannon limit ($-1.59$ dB) are about 0.11 dB and 0.35 dB, respectively.

\(^7\)In [16], (15) has been used to compute $I_{\text{req}}$. It is not very efficient because of the need to evaluate the PDF of the LLR values. In this paper, (16) is used instead to compute $I_{\text{req}}$, because it can be evaluated much more efficiently with graphics processing units. The computed thresholds are found to be slightly different and larger than those reported in [16].

In the same figure, we plot the BER results of the underlying PLDPCH-BC using $I_{\max} = 300$ standard decoding iterations, which should achieve almost the same BER performance as $I_{\max}/2 = 150 = I_{\text{BC}}$ layered decoding iterations [20]. We recall in Sect. IV-C.2 that when $(W + 1)I = I_{\text{BC}}$, the latencies of both SC-PLDPCH-CC pipeline decoder and PLDPCH-BC decoder become identical. Thus when $I_{\text{BC}}$ is $150$ layered decoding iterations are used in the PLDPCH-BC decoder, the number of processors used in the SC-PLDPCH-CC pipeline decoder equals $I = 150/(W + 1) = 75$. Comparing the corresponding BER curves in the figure shows that the SC-PLDPCH-CC outperforms the PLDPCH-BC by about 0.04 dB at a BER of $10^{-5}$. Next, we increase the second lifting factor of the PLDPCH-BC by a factor of $\delta_{z_2} = 1 + \frac{W}{1 + (2^r - r - 2)I_{\max}} = 1.136$ such that the blocklength of the PLDPCH-BC is the same as constraint length of the SC-PLDPCH-CC. Thus the second lifting factor of the PLDPCH-BC becomes $z_2' = 512 \times 1.136 \approx 582$. To maintain the same decoding latency, the number of layered iterations is reduced from $I_{\text{BC}} = 150$ to $I'_{\text{BC}} = I_{\text{BC}}/\delta_{z_2} = 150/1.136 = 132$. In other words, $I_{\max}$ is reduced to $I'_{\max} = 300/1.136 \approx 264$. Comparing the BER curves in the figure shows that under the same constraint length/blocklength and the same decoding latency, the SC-PLDPCH-CC outperforms the PLDPCH-BC by about 0.03 dB at a BER of $10^{-5}$.

We further compare our proposed code with the rate-0.0453 irregular zigzag-Hadamard code (IRZHC) [45], the rate-0.05 LDPC-Hadamard block code (LDPCH-BC) [14] and the rate-0.033 turbo-Hadamard code (THC) [9], [14]. To have a fair comparison, the information lengths in each block are about 65536 for all codes. First, it is given in [14], Fig. 10] that the rate-0.033 THC (with 80 iterations) achieves a BER of $10^{-5}$ at $E_b/N_0 = -0.95$ dB. In Fig. 6, we further redraw the BER curves of IRZHC ([45, Fig. 12]) and LDPCH-BC ([14, Fig. 10]). As can be observed, IRZHC and LDPCH-BC (with $I_{\max} = 400$ standard decoding iterations) achieve a BER of $10^{-5}$ at $E_b/N_0 = -1.17$ dB and $-1.18$ dB, respectively. However, our SC-PLDPCH-CC (using pipeline decoder with 80 processors) can attain the same BER at $E_b/N_0 = -1.238$ dB. Finally, we use the lifting factors $z_1 = 32$ and $z_2 = 4$ to expand the protomatrix such that the sub-block length of the SC-PLDPCH-CC equals 10368. Fig. 6(b) plots the BER results of the code. At $I = 80$ and a BER of $10^{-5}$, the SC-PLDPCH-CC degrades from $-1.24$ dB to $-0.85$ dB when the sub-block length is reduced from 1,327,104 to 10368.

### B. Rate-0.021 and $r = 5$

Based on the $6 \times 10$ protomatrix of the optimized rate-0.021 PLDPCH-BC [16], we apply GA and find the $6 \times 10$ protomatrices shown as (22), at the bottom of the next page, after 50 generations. The corresponding fitness value equals 609. Using the proposed layered PEXIT method, Table II lists the thresholds for the $r = 5$ SC-PLDPCH-TDC constructed by (22) and for the $r = 5$ PLDPCH-BC [16]. The theoretical threshold of the SC-PLDPCH-TDC with $L = 300$ is estimated to be $-1.37$ dB, which is the same with that of the

| Code | TDC | BC [16] |
|------|-----|---------|
| $r = 4$ | Code rate $R$ | Threshold in dB |
| | 0.0491 | -1.35 |
| $r = 5$ | Code rate $R$ | Threshold in dB |
| | 0.021 | -1.37 |
| $r = 8$ | Code rate $R$ | Threshold in dB |
| | 0.008 | -1.44 |
| $r = 10$ | Code rate $R$ | Threshold in dB |
| | 0.00291 | -1.48 |

\(N'_{\max} = 1000, W = 1, L = 500\) and all other sub-blocks have length $N_{\max} = 1000$.
PLDPCH-BC [16]. Hence, we consider that the \( r = 5 \) SC-PLDPCH-CC constructed by (22) has a threshold of about \(-1.37 \) dB. We use the same lifting factors, i.e., \( z_1 = 32 \) and \( z_2 = 512 \), as those used in the PLDPCH-BC to expand the protomatrix such that the sub-block length of the SC-PLDPCH-CC equals 3,112,960. The information length in each sub-block/block is 65536 for both codes.

The BER performance of the SC-PLDPCH-CC with different \( I \) is shown in Fig. 7. The pipeline decoder with \( I = 80 \) processors achieves a BER of \( 10^{-5} \) at about \( E_b/N_0 = -1.30 \) dB, which outperforms that with (i) \( I = 75 \) by about 0.01 dB, (ii) \( I = 70 \) by about 0.02 dB, and (iii) \( I = 60 \) by about 0.05 dB. The gaps of the SC-PLDPCH-CC (with \( I = 80 \) and BER of \( 10^{-5} \)) to the PEXIT threshold (\(-1.37 \) dB) and to the ultimate Shannon limit (\(-1.59 \) dB) are about 0.07 dB and 0.29 dB, respectively. In the same figure, we can also observe that under the same latency, SC-PLDPCH-CC using \( I = 75 \) processors outperforms PLDPCH-BC using \( I_{BC} = 150 \) layered decoding iterations (equivalent to \( I_{max} = 300 \) standard decoding iterations, i.e., the blue curve with symbol “○”) by about 0.05 dB at a BER of \( 10^{-5} \). Next, we increase the second lifting factor of the PLDPCH-BC by a factor of \( \delta_z = 1 + \frac{W}{1+(2^{-i})^{2}} \approx 1.053 \) such that the block-length of the PLDPCH-BC is the same as constraint length of the SC-PLDPCH-CC. Thus the second lifting factor of the PLDPCH-BC equals \( z_{BC} = 512 \times \delta_z = 512 \times 1.053 = 539 \). To maintain the same decoding latency, \( I'_{BC} = I_{BC}/\delta_z = 150/1.053 \approx 142 \) and \( I'_{max} = I_{max}/1.053 \approx 284 \). Comparing

\[
B_0 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0
\end{bmatrix} \quad B_1 = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\]

(21)

\[
B_0 = \begin{bmatrix}
1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 2 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1
\end{bmatrix} \quad B_1 = \begin{bmatrix}
2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
\]

(22)

\[
B_0 = \begin{bmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 2 \\
0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 3 \\
0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1
\end{bmatrix} \quad B_1 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 2 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

(23)

\[
B_0 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix} \quad B_1 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{bmatrix}
\]

(25)
the BER curves in the figure shows that under the same constraint length/blocklength and the same decoding latency, the SC-PLDPCH-CC outperforms the PLDPCH-BC by more than 0.05 dB at a BER of $10^{-5}$.

In Fig. 7, we further redraw the BER curves of rate-0.0189 THC [9, Fig. 11], rate-0.018 IRZHC [46, Fig. 14] and rate-0.022 LDPCH-BC [14, Fig. 12]. To have a fair comparison, the information lengths in each block are about 65536 for all codes. At a BER of $10^{-5}$, THC, IRZHC (with $I_{max} = 150$ iterations) and LDPCH-BC (with $I_{max} = 400$ iterations) require $E_b/N_0$ values of $-1.17$ dB, $-1.24$ dB and $-1.26$ dB, respectively; while our rate-0.021 SC-PLDPCH-CC using the pipeline decoding with $I = 80$ processors requires only $E_b/N_0 = -1.30$ dB.

C. Rate-0.008 and $r = 8$, Rate-0.00295 and $r = 10$

Based on the $5 \times 15$ protomatrix $B$ of the optimized rate-0.008 PLDPCH-BC [16], we apply our proposed GA and find the protomatrices shown as (22) and (23), at the bottom of the previous page. The protomatrices are found after 59 generations and the corresponding fitness value equals 457. We increase $L$ to 300 such that the SC-PLDPCH-TDC has the same code rate with its underlying block code [16]. Using our proposed PEXIT method, the terminated code with $L = 300$ has a threshold of $-1.45$ dB as shown in Table II. Hence, the theoretical threshold of the SC-PLDPCH-CC constructed by (22) and (23) approximately equals $-1.45$ dB, which is slightly lower (i.e., slightly better) than that of the PLDPCH-BC [16]. We lift the SC-PLDPCH-CC with factors $z_1 = 16$ and $z_2 = 1280$ such that its sub-block length is the same as that of the PLDPCH-BC in [16]. Moreover, the information lengths in each sub-block/block equal 204800 for both codes.

Fig. 8(a) shows the BER performance of the two codes. An SC-PLDPCH-CC pipeline decoder with $I = 100$ processors achieves a BER of $10^{-5}$ at about $E_b/N_0 = -1.40$ dB, which outperforms that with (i) $I = 90$ by about 0.01 dB, (ii) $I = 80$ by about 0.025 dB, (iii) and $I = 75$ by about 0.035 dB. The gaps of the SC-PLDPCH-CC (with $I = 100$...
and BER of $10^{-5}$) to the PEXIT threshold ($-1.45$ dB) and to the ultimate Shannon limit ($-1.59$ dB) are 0.05 dB and 0.19 dB, respectively. We can also observe that under the same latency, SC-PLDPCH-CC using $I = 75$ processors outperforms PLDPCH-BC using $I_{BC} = 150$ layered decoding iterations (equivalent to $I_{max} = 300$ standard decoding iterations, i.e., the blue curve with symbol “o”) by about 0.01 dB at a BER of $10^{-5}$. In Fig. 8(a), we re-draw the BER curve of rate-0.008 LDPCH-BC [14, Fig. 13] which has an information length of 238000. We see that at a BER of $10^{-5}$, the required $E_b/N_0$ for the rate-0.008 LDPCH-BC (with $I_{max} = 400$ iterations) is $-1.38$ dB and that for our rate-0.008 SC-PLDPCH-CC (pipeline decoder containing 100 processors) is $-1.40$ dB.

Based on the $6 \times 24$ protomatrix $B$ of the optimized rate-0.00295 PLDPCH-BC [16], the two protomatrices $B_0$ and $B_1$ obtained by our proposed GA are shown in (25), at the bottom of page 14, and (V), respectively. The protomatrices (25) and (V) are found after 48 generations, and the corresponding fitness value equals 516. Using the proposed PEXIT method, Table II lists the thresholds for SC-PLDPCH-TDC with $L = 100$ and the underlying PLDPCH-BC [16]. The SC-PLDPCH-TDC constructed by $L = 100$ sets of protomatrices (25) and (V) has almost the same code rate with its underlying block code, and its threshold is estimated to be $-1.48$ dB. Hence, the theoretical threshold of the SC-PLDPCH-CC constructed by (25) and (V) also equals $-1.48$ dB, which is slightly greater than that of the PLDPCH-BC [16]. We lift the SC-PLDPCH-CC with factors $z_1 = 20$ and $z_2 = 1280$ such that its sub-block length is the same as that of the PLDPCH-BC [16]. Fig. 8(b) shows the BER performance of the two codes. SC-PLDPCH-CC decoder with $I = 140$ processors achieves a BER of $10^{-5}$ at about $E_b/N_0 = -1.465$ dB, which outperforms that with (i) $I = 120$ by about 0.01 dB, (ii) $I = 100$ by about 0.03 dB, and (iii) $I = 75$ by about 0.07 dB. At a BER of $10^{-5}$, the gaps (for the SC-PLDPCH-CC with $I = 140$ iterations) to the PEXIT threshold ($-1.48$ dB) and to the ultimate Shannon limit ($-1.59$ dB) are 0.015 dB and 0.125 dB, respectively. Under the same latency, SC-PLDPCH-CC using $I = 75$ processors is outperformed by PLDPCH-BC using $I_{BC} = 150$ layered decoding iterations (equivalent to $I_{max} = 300$ standard decoding iterations) by about 0.04 dB at a BER of $10^{-5}$.

In Fig. 8(b), we re-draw the BER curve of rate-0.003 LDPCH-BC [14, Fig. 13] which has an information length of 650,000. Results show that the rate-0.003 LDPCH-BC (with $I_{max} = 400$ iterations) achieves a BER of $10^{-5}$ at $E_b/N_0 = -1.44$ dB and the rate-0.00295 SC-PLDPCH-CC (containing $I = 140$ processors) achieves the same error performance at $E_b/N_0 = -1.465$ dB.

VI. CONCLUSION

We have proposed a new type of ultimate-Shannon-limit-approaching code called SC-PLDPCH-CCs, which are formed by spatially coupling PLDPCH-BCs. We have developed a pipelined decoding with layered scheduling algorithm to efficiently and effectively decode SC-PLDPCH-CCs, and have proposed a layered PEXIT method to evaluate the threshold of SC-PLDPCH-TDCs. Based on the protomatrix of a PLDPCH-BC, we have proposed a genetic algorithm to systematically search for the protomatrices of good SC-PLDPCH-TDCs. We extend the coupling length of these SC-PLDPCH-TDCs with good thresholds to form SC-PLDPCH-CCs. Using the proposed methods, we have found SC-PLDPCH-CCs which
are comparable to or superior to their block code counterparts in terms of theoretical threshold and error performance. Their simulated error performances outperform state-of-the-art low-rate codes with comparable rates and information lengths. At a BER of $10^{-5}$, the SC-PLDPC-CCs of rates 0.0494, 0.021, 0.008 and 0.00295 are only 0.352 dB, 0.29 dB, 0.19 dB and 0.125 dB from the ultimate Shannon limit.

REFERENCES

[1] R. G. Gallager, “Low-density parity-check codes,” Ph.D. thesis, Dept. Elect. Eng., Massachusetts Inst. Technol., Cambridge, MA, USA, 1963.
[2] R. M. Tanner, “A recursive approach to low complexity codes,” IEEE Trans. Inf. Theory, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.
[3] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” IEEE Trans. Inf. Theory, vol. 47, no. 2, pp. 619–637, Feb. 2001.
[4] S. T. Brink, G. Kramer, and A. Ashikhmin, “Design of low-density parity-check codes for modulation and detection,” IEEE Trans. Commun., vol. 52, no. 4, pp. 670–678, Apr. 2004.
[5] E. Sharon, A. Ashikhmin, and S. Litsyn, “Analysis of low-density parity-check codes based on EXIT functions,” IEEE Trans. Commun., vol. 54, no. 8, pp. 1407–1414, Aug. 2006.
[6] D. J. Costello and G. D. Forney, “Channel coding: The road to channel capability,” Proc. IEEE, vol. 95, no. 6, pp. 1150–1177, Jun. 2007.
[7] L. Ping, L. Liu, K. Y. Wu, and W. K. Leung, “Approaching the capacity of multiple access channels using interleaved low-rate codes,” IEEE Commun. Lett., vol. 8, no. 1, pp. 4–6, Jan. 2004.
[8] L. Ping, L. Liu, K. Wu, and W. K. Leung, “Interleave division multiple-access,” IEEE Trans. Wireless Commun., vol. 5, no. 4, pp. 938–947, Apr. 2006.
[9] L. Ping, W. K. Leung, and K. Y. Wu, “Low-rate turbo-Hadamard codes,” IEEE Trans. Inf. Theory, vol. 49, no. 12, pp. 3213–3224, Dec. 2003.
[10] W. K. Leung, G. Yue, L. Ping, and X. Wang, “Concatenated zigzag Hadamard codes,” IEEE Trans. Inf. Theory, vol. 52, no. 4, pp. 1711–1723, Apr. 2006.
[11] S. Jiang, P. W. Zhang, F. C. M. Lau, C.-W. Sham, and K. Huang, “A turbo-Hadamard encoder/decoder system with hundreds of Mb/s throughput,” in Proc. IEEE 10th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC), Dec. 2018, pp. 1–5.
[12] S. Jiang, P. W. Zhang, F. C. M. Lau, and C.-W. Sham, “An ultimate-Shannon-limit-approaching Gbps throughput encoder/decoder system,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 67, no. 10, pp. 2169–2173, Oct. 2020.
[13] S. Jiang, F. C. M. Lau, and C.-W. Sham, “Hardware design of concatenated zigzag Hadamard encoder/decoder system with high throughput,” IEEE Access, vol. 8, pp. 165298–165306, 2020.
[14] G. Yue, L. Ping, and X. Wang, “Generalized low-density parity-check codes based on Hadamard constraints,” IEEE Trans. Inf. Theory, vol. 53, no. 3, pp. 1058–1079, Mar. 2007.
[15] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, “Regular and irregular progressive edge-growth Tanner graphs,” IEEE Trans. Inf. Theory, vol. 51, no. 1, pp. 386–398, Jan. 2005.
[16] P.-W. Zhang, F. C. M. Lau, and C.-W. Sham, “Protograph-based LDPC Hadamard codes,” IEEE Trans. Commun., vol. 69, no. 8, pp. 4998–5013, Aug. 2021.
[17] J. Thorpe, “Low-density parity-check (LDPC) codes constructed from protographs,” in Proc. Ipn. IPC Comm. Conf., Aug. 2003, pp. 1–7.
[18] M. P. C. Fossorier, “Quasi-cyclic low-density parity-check codes from circulant permutation matrices,” IEEE Trans. Inf. Theory, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.
[19] G. Liva and M. Chiani, “Protograph LDPC codes design based on EXIT analysis,” in Proc. IEEE GLOBECOM, Nov. 2007, pp. 3250–3254.
[20] P. W. Zhang, F. C. M. Lau, and C.-W. Sham, “Layered decoding for protograph-based low-density parity-check Hadamard codes,” IEEE Commun. Lett., vol. 25, no. 6, pp. 1776–1780, Jun. 2021.
[21] A. R. Iyengar, M. Papaleo, P. H. Siegel, J. K. Wolf, A. Vanelli-Coralli, and G. E. Corazza, “Windowed decoding of protograph-based LDPC convolutional codes over erasure channels,” IEEE Trans. Inf. Theory, vol. 58, no. 12, pp. 7193–7202, Apr. 2012.
[22] A. J. Felstrom and K. S. Zigangirov, “Time-varying periodic convolutional codes with low-density parity-check matrix,” IEEE Trans. Inf. Theory, vol. 45, no. 6, pp. 2181–2191, Sep. 1999.
Peng-Wei Zhang received the B.E. degree in electronics and information engineering and the M.E. degree in electronics and communication engineering from the Chongqing University of Posts and Telecommunications, China, in 2013 and 2016, respectively, and the Ph.D. degree from the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong, SAR, China, in 2021. He is currently with Huawei Technologies Ltd., Chengdu, China.

Francis C. M. Lau (Fellow, IEEE) received the B.Eng. degree (Hons.) in electrical and electronic engineering and the Ph.D. degree from the King’s College London, University of London, U.K. He is a Professor with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong. He is a co-author of two research monographs. He is also a co-holder of six U.S. patents. He has published more than 330 papers. His main research interests include channel coding, cooperative networks, wireless sensor networks, chaos-based digital communications, applications of complex-network theories, and wireless communications.

Dr. Lau is a fellow of IET. He was a co-recipient of one the Natural Science Award from the Guangdong Provincial Government, China; eight the Best/Outstanding Conference Paper Awards; one the Technology Transfer Award; two the Young Scientist Awards from International Union of Radio Science; and one the FPGA Design Competition Award. He was the General Co-Chair of International Symposium on Turbo Codes and Iterative Information Processing in 2018 and the Chair of the Technical Committee on Non-linear Circuits and Systems, IEEE Circuits and Systems Society from 2012 to 2013. He was an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, since 2017.

Chiu-Wing Sham (Senior Member, IEEE) received the bachelor’s degree in computer engineering and the M.Phil. and Ph.D. degrees from The Chinese University of Hong Kong, in 2000, 2002, and 2006, respectively. He was an Electronic Engineer of the FPGA applications of the motion-control system and system security with cryptography at ASM Pacific Technology Ltd., Hong Kong. During the years at The Hong Kong Polytechnic University, he engaged in various University projects for the commercialization of technology, in particular, a few optical communication projects which were in collaboration with Huawei. He also worked on the physical design of VLSI design automation. He was invited to work at Synopsys, Inc., Shanghai, in the summer of 2005 as a Visiting Research Engineer. He is currently with The University of Auckland as a Senior Lecturer. He has been an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, since 2017.