The following publication P. -W. Zhang, F. C. M. Lau and C. -W. Sham, "Protograph-Based LDPC Hadamard Codes," in IEEE Transactions on Communications, vol. 69, no. 8, pp. 4998-5013, Aug. 2021 is available at https://doi.org/10.1109/TCOMM.2021.3077939.

Protograph-Based LDPC Hadamard Codes

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

***Abstract*—In this paper, we propose a new method to design low-density parity-check Hadamard (LDPC-Hadamard) codes — a type of ultimate-Shannon-limit approaching channel codes. The technique is based on applying Hadamard constraints to the check nodes in a generalized protograph-based LDPC code, followed by lifting the generalized protograph. We name the codes formed *protograph-based LDPC Hadamard* (PLDPC-Hadamard) codes. We also propose a modified Protograph Extrinsic Information Transfer (PEXIT) algorithm for analyzing and optimizing PLDPC-Hadamard code designs. The proposed algorithm further allows the analysis of PLDPC-Hadamard codes with degree-1 and/or punctured nodes. We find codes with decoding thresholds ranging from -1.53 dB to -1.42 dB. At a BER of 10^{-5}, the gaps of our codes to the ultimate-Shannon-limit range from 0.40 dB (for rate = 0.0494) to 0.16 dB (for rate = 0.003). Moreover, the error performance of our codes is comparable to that of the traditional LDPC-Hadamard codes. Finally, the BER performances of our codes after puncturing are simulated and compared.**

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

I. Introduction

In 1943, Claude Shannon derived the channel capacity theorem [2], based on which the maximum rate that information can be sent through a channel without errors can be evaluated. In 1993, Berrou *et al*. invented the turbo codes and demonstrated that with a code rate of 0.5, the proposed turbo code and decoder could work within 0.7 dB from the capacity limit at a bit error rate (BER) of 10^{-5} [3], [4]. Besides turbo codes, other well-known capacity-approaching codes are low-density parity-check codes (proposed by Gallager in 1960s [5] and rediscovered by MacKay and Neal in 1990s [6]) and polar codes (proposed by Arikan in 2009 [7]). These capacity-approaching codes have since been used in many wireless communication systems (e.g., 3G/4G/5G, Wifi, satellite communications) and optical communication systems. The progresses of the aforementioned three types of capacity-approaching codes over the past decades can be found in the survey papers [8], [9] and the references therein.

In particular, LDPC codes can be represented by a matrix containing a low density of "1"s and also by its corresponding Tanner graph [10]. In the Tanner graph, there are two sets of nodes, namely variables nodes (VNs) and check nodes (CNs), sparsely connected by links. Messages are updated and passed iteratively along the links during the decoding process. Density evolution (DE) [11] is a kind of analytical method that tracks the probability density function (PDF) of the messages after each iteration. It not only can predict the convergence of the decoder, but also can be used for optimizing LDPC code designs [12]. The extrinsic information transfer (EXIT) chart is another common technique employed to analyze and optimize LDPC codes [13], [14]. An optimal LDPC code design is found when the EXIT curves of the VNs and CNs are "matched" with the smallest bit-energy-to-noise-power-spectral-density ratio (E_b/N_0).

For an LDPC code with given degree distributions and code length, the progressive-edge-growth (PEG) method [15] is commonly used to connect the VNs and CNs with an aim to maximizing the girth (shortest cycle) of the code. The method is simple and the code can achieve good error performance. However, the code has a quadratic encoding complexity with its length because it is un-structured. The hardware implementation of the encoder/decoder also consumes a lot of resources and has high routing complexity.

Subsequently, structured quasi-cyclic (QC) LDPC codes are proposed [16]. QC-LDPC codes have a linear encoding complexity and allow parallel processing in the hardware implementation. Other structured codes, such as the repeat-accumulate (RA) codes and their variants, can be formed by the repeat codes and the accumulators [17], [18]. They belong to a subclass of LDPC codes that have a fast encoder structure and good error performance. Structured LDPC codes can also be constructed by protographs [19]. By expanding the protomatrix (corresponding protograph) with a small size, we can obtain a QC matrix (corresponding lifted graph) that possesses the same properties as the protomatrix. The codes corresponding to the lifted graphs are called protograph-based LDPC (PLDPC) codes. The traditional EXIT chart cannot be used to analyze protographs where degree-1 or punctured variable nodes exist. Subsequently, the protograph EXIT (PEXIT) chart method is developed [20] for analyzing and designing PLDPC codes, and well-designed PLDPC codes are found to achieve performance close to the Shannon limit [9], [21]. When coupled spatially, PLDPC codes can enhance their theoretical thresholds and decoding performance [22], [23], [24]. In the case of block-fading channels, root-protograph LDPC codes are found to achieve near-outage-limit performance [25].

In the Tanner graph of an LDPC code, the VNs are equivalent to repeat codes while CNs correspond to single-parity-check (SPC) codes. If other block codes, such as Hamming codes and BCH codes, are used to replace the repeat codes and/or SPC codes, generalized LDPC (GLDPC) codes are obtained [26], [27], [28]. In [29], [30] and [31], doped-Tanner codes are formed by replacing the SPC component codes in the structured LDPC codes with Hamming codes and

P.W. Zhang and F.C.M. Lau are with the Future Wireless Networks and IoT Focusing Area, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong (e-mail: pengwei.zhang@connect.polyu.hk and francis-cm.lau@polyu.edu.hk).

C.-W. Sham is with the Department of Computer Science, The University of Auckland, New Zealand (e-mail: b.sham@auckland.ac.nz).

The work described in this paper was supported by a grant from the RGC of the Hong Kong SAR, China (Project No. PolyU 152170/18E).

This paper was presented in part at WCNC 2020 [1].

recursive systematic convolutional codes. Ensemble codeword weight enumerators are used to find good GLDPC codes while Hamming codes have been used to design medium-length GLDPC codes with performances approaching the channel capacity (> 0 dB). In [32] and [33], EXIT functions of block codes over binary symmetric channels have been derived and used for analyzing LDPC codes. The same author further demonstrates the use of linear programming algorithm to optimize a rate-$8/9$ GLDPC code from the perspective of degree distribution [34]. To achieve good performance (BER = 10^{-5}) at very low E_b/N_0, say < -1.15 dB, Hadamard codes have been proposed to replace the SPC codes, forming the low-rate (≤ 0.05) LDPC-Hadamard codes [35], [36]. By adjusting the degree distribution of the VNs and using the EXIT chart technique, the EXIT curves of the Hadamard "super CNs" and VNs are matched and excellent error performance at low E_b/N_0 is obtained.

In practice, different channels possess different capacities, depending on factors such as modulation scheme, signal-to-noise ratio and code rate. However, the "ultimate Shannon limit" over an additive-white-Gaussian-noise (AWGN) channel remains at -1.59 dB, i.e., $E_b/N_0 = -1.59$ dB. Scenarios where digital communications may need to work close to the ultimate Shannon limit include space communications, multiple access (e.g. code-division multiple-access [37] and interleave-division multiple-access [38]) with severe inter-user interferences, or embedding low-rate information in a communication link. The most notable channel codes with performance close to this limit are turbo-Hadamard codes [39], [40], [41], [42], concatenated zigzag Hadamard codes [43], [44], and LDPC-Hadamard codes [35], [36]. Both turbo-Hadamard codes and concatenated zigzag Hadamard codes suffer from long decoding latency due to the forward/backward decoding algorithms [39], [43]. The LDPC-Hadamard codes allow parallel processing and hence the decoding latency can be made much shorter [36]. However, in optimizing the threshold of LDPC-Hadamard codes, only the degree distribution of the variable nodes has been found for a given order of the Hadamard code used. Therefore, the method used in optimizing LDPC-Hadamard codes has the following drawbacks.

- For the same variable-node degree distribution, many different code realizations with very diverse bit-error-rate performances can be obtained.
- The code is unstructured, making both encoding and decoding very complex to realize in practice. Take the LDPC-Hadamard code with code rate $R = 0.05$ and Hadamard code order $r = 4$ as an example. For an information length of $65,536$, the degree distributions optimized by [36] indicate that there are $113,426$ Hadamard check nodes and $n = 178,962$ variable nodes. When these large number of nodes are connected by the PEG algorithm, the resultant graph has little structure and is therefore not conducive to parallel encoding/decoding and reduces encoding/decoding efficiency. In the hardware implementation, the unstructured conventional LDPC-Hadamard code further results in high routing complexity and low throughput.
- The degree distribution analysis requires a minimum variable-node degree of 2 because an EXIT curve cannot be produced for degree-1 variable nodes. Moreover, LDPC-Hadamard codes with punctured variable nodes cannot be analyzed.

The concept in [36] has been applied to designing other low-rate generalized LDPC codes [45]. However, the main criterion of those codes is to provide low latency communications and hence their performance is relatively far from the ultimate Shannon limit.

In this paper, we propose a method to design LDPC-Hadamard codes which possesses degree-1 and/or punctured VNs. The technique is based on applying Hadamard constraints to the CNs in a generalized PLDPC code, followed by lifting the generalized protograph. We name the codes formed *protograph-based LDPC Hadamard* (PLDPC-Hadamard) codes [1]. We also propose a modified PEXIT algorithm for analyzing and optimizing PLDPC-Hadamard code designs. Codes with decoding thresholds ranging from -1.53 dB to -1.42 dB have been found, and simulation results show a bit error rate of 10^{-5} can be achieved at $E_b/N_0 = -1.43$ dB. Moreover, the BER performances of these codes after puncturing are simulated and compared. The main contributions of the paper can be summarized as follows:

1) It is the first attempt to use protographs to design codes with performance close to the ultimate Shannon limit. By appending additional degree-1 Hadamard VNs to the CNs of a protograph, the SPC check nodes are converted into more powerful Hadamard constraints, forming the generalized protograph of PLDPC-Hadamard codes. After using the copy-and-permute operations to lift the protograph, the matrix corresponding to the lifted graph is a structured QC matrix which is greatly beneficial to linear encoding, parallel decoding and hardware implementation.
2) To analyze the decoding threshold of a PLDPC-Hadamard code, we propose a modified PEXIT method. We replace the SPC mutual information (MI) updating with our proposed Hadamard MI updating based on Monte Carlo simulations. Different from the EXIT method used in optimizing the degree distribution of VNs in an LDPC-Hadamard code [36], our proposed PEXIT method searches and analyzes protomatrices corresponding to the generalized protograph of the PLDPC-Hadamard codes. The proposed method, moreover, is applicable to analyzing PLDPC-Hadamard codes with degree-1 VNs and/or punctured VNs. Using the analytical technique, we have found PLDPC-Hadamard codes with very low decoding thresholds (< -1.40 dB) under different code rates.
3) Extensive simulations are performed under an AWGN channel. For each case, 100 frame errors are collected before the simulation is terminated. Results show that the PLDPC-Hadamard codes can obtain comparable BER performance to the traditional LDPC-Hadamard codes [36]. At a BER of 10^{-5}, the gaps to the ultimate Shannon limit are 0.40 dB for the rate-0.0494 code, 0.35 dB for

the rate-0.021 code, 0.24 dB for the rate-0.008 code and 0.16 dB for the rate-0.003 code, respectively.

4) Punctured PLDPC-Hadamard codes are studied. Puncturing different VNs in the protograph of a PLDPC-Hadamard code sometimes can produce different BER/FER performance improvement/degradation compared with the unpunctured code. Moreover, when the order of the Hadamard code $r = 5$, puncturing the extra degree-1 Hadamard VNs provided by the non-systematic Hadamard encoding is found to degrade the error performance.

The remainder of the paper is organized as follows. Sect. II reviews the background knowledge of some related codes. Sect. III introduces our proposed PLDPC-Hadamard code, including its structure, encoding and decoding methods, and code rate. In particular, the cases in which the order of the Hadamard code used is even or odd are described and analyzed. A low-complexity PEXIT method for analyzing PLDPC-Hadamard codes is proposed and an optimization algorithm is provided. Sect. IV presents the protomatrices of the PLDPC-Hadamard codes found by the proposed methods, their decoding thresholds and simulation results. The error performance of these codes after puncturing are further evaluated. Sect. V provides some final remarks.

II. Background

A Hadamard code with an order r is a class of linear block codes. A $q \times q$ positive Hadamard matrix $+\boldsymbol{H}_q = \{+\boldsymbol{h}_j, j = 0, 1, \ldots, q-1\}$ can be constructed recursively using

$$+\boldsymbol{H}_q = \begin{bmatrix} +\boldsymbol{H}_{q/2} & +\boldsymbol{H}_{q/2} \\ +\boldsymbol{H}_{q/2} & -\boldsymbol{H}_{q/2} \end{bmatrix}$$

with $q = 2^r$ and $\pm\boldsymbol{H}_1 = [\pm 1]$. Each column $+\boldsymbol{h}_j$ is a Hadamard codeword and $\pm\boldsymbol{H}_q$ contains $2q = 2^{r+1}$ codewords $\pm\boldsymbol{h}_j$. Considering an information sequence $\boldsymbol{u} \in \{0,1\}^{r+1}$, the Hadamard encoder encodes $\boldsymbol{u}$ into a codeword $\boldsymbol{c}^H$ of length q, i.e., $\boldsymbol{c}^H \in \{0,1\}^{2^r} = [c_0^H \ c_1^H \ \ldots \ c_{2^r-1}^H]^T$, where $(\cdot)^T$ represents the transpose operation. Assuming that the $+\boldsymbol{h}_j$ or $-\boldsymbol{h}_j$ corresponding to $\boldsymbol{c}$, i.e., by mapping bit "0" in $\boldsymbol{c}$ to $+1$ and bit "1" to -1, is uniformly transmitted through an AWGN channel with mean 0 and variance σ_{ch}^2, we denote the received signal by $\boldsymbol{y} = [y_0 \ y_1 \ \ldots \ y_{2^r-1}]^T$. We define $\boldsymbol{L}_{ch}^H = [L_{ch}^H(0) \ L_{ch}^H(1) \ \cdots \ L_{ch}^H(2^r-1)]^T$ and $\boldsymbol{L}_{apr}^H = [L_{apr}^H(0) \ L_{apr}^H(1) \ \cdots \ L_{apr}^H(2^r-1)]^T$; where

$$L_{ch}^H(i) = \ln \frac{p(y_i \mid c_i^H = \text{“0”})}{p(y_i \mid c_i^H = \text{“1”})} = \frac{2y_i}{\sigma_{ch}^2}$$

is the channel LLR value of the i-th bit and

$$L_{apr}^H(i) = \ln \frac{\Pr(c_i^H = \text{“0”})}{\Pr(c_i^H = \text{“1”})}$$

is the *a priori* LLR of the i-th bit ($i = 0, 1, ..., 2^r - 1$).

Using a symbol-by-symbol maximum *a posteriori* probability (symbol-MAP) Hadamard decoder [39], the *a posteriori* LLR of the i-th ($i = 0, 1, ..., 2^r - 1$) code bit, which is denoted by $L_{app}^H(i)$, can be computed by

$$\begin{aligned} L_{app}^H(i) &= \ln \frac{\Pr(c_i^H = \text{“0”} \mid \boldsymbol{y})}{\Pr(c_i^H = \text{“1”} \mid \boldsymbol{y})} \\ &= \ln \frac{\sum\limits_{\pm H[i,j]=+1} \Pr(\boldsymbol{c}^H = \pm\boldsymbol{h}_j \mid \boldsymbol{y})}{\sum\limits_{\pm H[i,j]=-1} \Pr(\boldsymbol{c}^H = \pm\boldsymbol{h}_j \mid \boldsymbol{y})} \\ &= \ln \frac{\sum\limits_{\pm H[i,j]=+1} \gamma(\pm\boldsymbol{h}_j)}{\sum\limits_{\pm H[i,j]=-1} \gamma(\pm\boldsymbol{h}_j)} \end{aligned} \tag{1}$$

where

$$\gamma(\pm\boldsymbol{h}_j) = \exp\left(\langle \pm\boldsymbol{h}_j, \boldsymbol{L}_{ch}^H + \boldsymbol{L}_{apr}^H \rangle / 2\right) \tag{2}$$

and $\langle\cdot\rangle$ denotes the inner product. We further define $\boldsymbol{L}_{app}^H = [L_{app}^H(0) \ L_{app}^H(1) \ \cdots \ L_{app}^H(2^r-1)]^T$. Based on the butterfly-like structure of the Hadamard matrix, $\boldsymbol{L}_{app}^H$ can be computed using the fast Hadamard transform (FHT) and the dual FHT (DFHT) [39], [41], [42]. In the case of iterative decoding, the Hadamard decoder subtracts the $L_{apr}^H(i)$ from $L_{app}^H(i)$ and feeds back "new" extrinsic information to other component decoders.

An LDPC code with code length N, information length $k = N - M$ and code rate $R = k/N$ can be represented by a $M \times N$ parity-check matrix $\boldsymbol{H}_{M\times N}$ whose entries only include 0 or 1. The matrix $\boldsymbol{H}_{M\times N}$ can also be represented by a Tanner graph, as shown in Fig. 1. The circles denote VNs corresponding to the columns of the matrix; the squares denote the CNs corresponding to the rows of the matrix; and the edges connecting the VNs and CNs correspond to the "1"s of the matrix. Moreover, a VN with degree-d_j emits $d_j (d_j > 1)$ edges connecting to d_j different CNs and forms a $(d_j, 1)$ repeat code; whereas a CN with degree-d_i emits $d_i (d_i > 1)$ edges connecting d_i different VNs and forms a $(d_i, d_i - 1)$ single-parity-check (SPC) code. A generalized LDPC code is obtained when the repeat code and/or SPC code is/are replaced by other block codes. In [36], the SPC codes of an LDPC code are replaced with Hadamard codes, forming an LDPC-Hadamard code. When an LDPC code contains degree-1 VNs or punctured VNs, the traditional EXIT chart cannot evaluate its decoding performance. However, for LDPC codes constructed based on protographs, their theoretical performance can be estimated by the protograph EXIT (PEXIT) algorithm even if they contain degree-1 VNs or punctured VNs [20].

Fig. 2 illustrates a protograph, and the corresponding protomatrix (also called base matrix) is given by

$$\boldsymbol{B}_{m\times n} = \begin{bmatrix} 1 & 3 & \cdots & 0 & 1 \\ 2 & 1 & \cdots & 2 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & 2 \end{bmatrix}.$$

The entries in $\boldsymbol{B}_{m\times n} = \{b_{i,j} : i = 0, 1, 2 \ldots, m-1; j = 0, 1, 2 \ldots, n-1\}$ are allowed to be larger than 1 and they correspond to the multiple edges connecting the same pair of VN and CN in the protograph. The Tanner graph of a protograph-based LDPC (PLDPC) code can be constructed by duplicating the protograph z times and permuting the edges

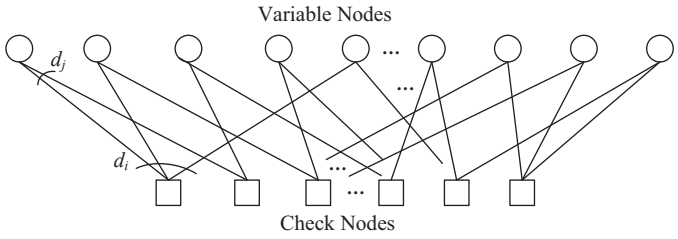


Fig. 1. Representation of an LDPC code by a Tanner graph.

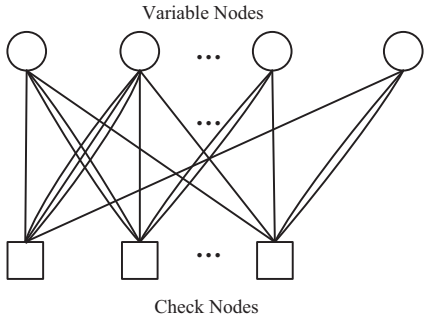


Fig. 2. A protograph.

which connect the same type of VNs and CNs among these duplicated protographs. This expansion process is also called lifting and the parameter z is called the lifting factor. To analyze the decoding performance of a PLDPC code, the PEXIT algorithm can be applied to $\boldsymbol{B}_{m\times n}$. In the PEXIT method, the mutual information (MI) values on all types of edges are updated separately and iteratively [20]. To illustrate the method, different types of MI are first defined as follows:

- $I_{ac}(i,j)$: *a priori* MI from *j*-th VN to *i*-th CN in $\mathbf{B}_{m\times n}$;
- $I_{av}(i,j)$: *a priori* MI from *i*-th CN to *j*-th VN in $\mathbf{B}_{m\times n}$;
- $I_{ev}(i,j)$: extrinsic MI from *j*-th VN to *i*-th CN in $\mathbf{B}_{m\times n}$;
- $I_{ec}(i,j)$: extrinsic MI from *i*-th CN to *j*-th VN in $\mathbf{B}_{m\times n}$;
- $I_{app}(j)$: *a posteriori* MI value of the *j*-th VN;
- I_{ch}: MI from the channel.

Without going into the details, the steps below show how to determine the threshold $(E_b/N_0)_{th}$.

1) Set all MI values to 0. Select a relatively large E_b/N_0.
2) Initialize I_{ch} based on E_b/N_0 and the code rate.
3) Compute $I_{ev}(i,j)\ \forall\ i,j$; set $I_{ac}(i,j) = I_{ev}(i,j)$; compute $I_{ec}(i,j)\ \forall\ i,j$; set $I_{av}(i,j) = I_{ec}(i,j)$.
4) Repeat Step 3) I_{iter} times.
5) Compute $I_{app}(j)$. If $I_{app}(j) = 1\ \forall\ j$, reduce E_b/N_0 and go to Step 2); otherwise set the previous E_b/N_0 that achieves $I_{app}(j) = 1\ \forall\ j$ as the threshold $(E_b/N_0)_{th}$ and stop.

The above analytical process can be regarded as the repeated computation and exchange between the *a priori* MI matrices $\{I_{av}(i,j)\}/\{I_{ac}(i,j)\}$ and extrinsic MI matrices $\{I_{ev}(i,j)\}/\{I_{ec}(i,j)\}$. Moreover, these matrices have the same size as $\boldsymbol{B}_{m\times n}$. Note that the PEXIT algorithm can be used to analyze protographs with degree-1 VNs, i.e., columns in the protomatrix with weight 1. Protographs with punctured VNs will also be analyzed in a similar way, except that the code rate will be changed accordingly and the corresponding I_{ch} will be initialized as 0.

III. Protograph-based LDPC-Hadamard Codes

A. Code Structure

We propose a variation of LDPC-Hadamard code called *protograph-based LDPC Hadamard (PLDPC-Hadamard) code*. The base structure of a PLDPC-Hadamard code is shown in Fig. 3, where each blank circle denotes a protograph variable-node (P-VN); each square with an "H" inside denotes a Hadamard check-node (H-CN); and each filled circle denotes a degree-1 Hadamard variable-node (D1H-VN). We assume

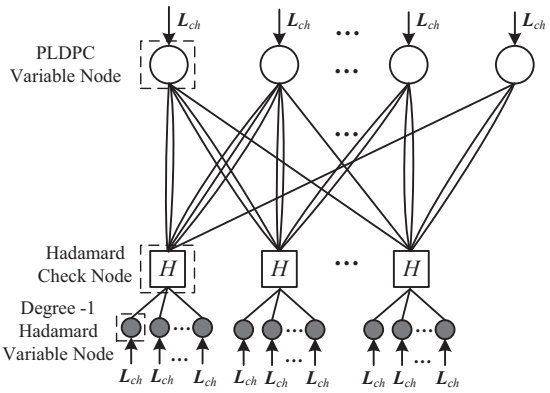


Fig. 3. The protograph of a PLDPC-Hadamard code.

that there are n P-VNs and m H-CNs. The protomatrix of the proposed PLDPC-Hadamard codes is then denoted by $\boldsymbol{B}_{m\times n} = \{b_{i,j}\}$, where $b_{i,j}$ represents the number of edges connecting the i-th H-CN ($i = 0, 1, \ldots, m-1$) and the j-th P-VN ($j = 0, 1, \ldots, n-1$). Moreover, we denote the weight of the i-th row by $d_{c_i} = \sum_{j=0}^{n-1} b_{i,j}$, which represents the total number of edges connecting the i-th H-CN to all P-VNs. For example in Fig. 3, the number of edges connecting each of the three displayed H-CNs to all P-VNs is equal to $d_{c_i} = 6$. These d_{c_i} edges are considered as (input) information bits to the i-th Hadamard code while the connected D1H-VNs represent the corresponding (output) parity bits in the Hadamard code. Recall that an order-r Hadamard code contains 2^{r+1} codewords with each codeword containing $r+1$ information bits. Suppose a Hadamard code of order-$(d_{c_i}-1)$ is used to encode these d_{c_i} inputs and generate $2^{(d_{c_i}-1)} - d_{c_i}$ Hadamard parity-check bits. As these $d_{c_i} = r+1$ bits take part in the same parity-check equation of an LDPC code and need to fulfill the SPC constraint [1], the number of possible combinations of these d_{c_i} bits is only $2^{(d_{c_i}-1)}$ and thus $2^{(d_{c_i}-1)} = 2^r$ Hadamard codewords will be generated. In other words, only half of the 2^{r+1} available Hadamard codewords are used, making the encoding process very inefficient.

Same as in LDPC-Hadamard codes [36], we utilize Hadamard codes with order $r = d_{c_i} - 2$ $(r > 2)$ in the proposed PLDPC-Hadamard codes. With such an arrangement,

[1]If these $d_{c_i} = r+1$ input bits are not required to satisfy a SPC constraint, two other types of LDPC-Hadamard codes can be formed and they are briefly described in Appendix A.

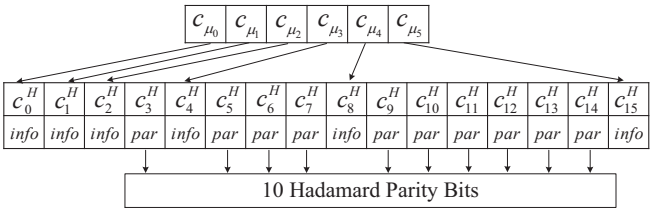


Fig. 4. Example of encoding a length-6 SPC codeword into a length-16 ($r = 4$) Hadamard codeword.

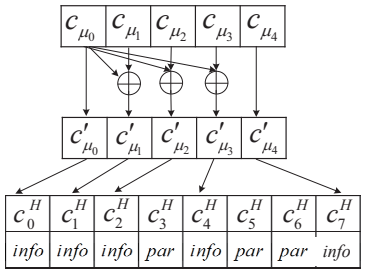


Fig. 5. Example of encoding a length-5 SPC codeword into a length-8 ($r = 3$) Hadamard codeword.

all possible Hadamard codewords, i.e., $2^{(d_{c_i}-1)} = 2^{r+1}$ can be utilized; fewer Hadamard parity bits compared with the case of $r = d_{c_i} - 1$ need to be added (only $(2^{(d_{c_i}-2)} - d_{c_i})$ and $(2^{(d_{c_i}-2)} - 2)$ Hadamard parity-check bits are generated for r is even and odd, respectively); the encoding process becomes most efficient; the overall code rate is increased; and the decoding performance is improved. (Note that a Hadamard code with order $r = 2$ is equivalent to the $(4, 3)$ SPC code. No extra parity-check bits (i.e., D1H-VNs) will be generated if such an Hadamard code is used in the PLDPC-Hadamard code. Thus Hadamard codes with order $r = 2$ are not considered.)

In the following, we consider the cases when r is even and odd separately. It is because systematic Hadamard encoding is possible when r is even and non-systematic Hadamard encoding needs to be used when r is odd.

1) $r = d_{c_i} - 2$ is an even number: We denote a Hadamard codeword by $\boldsymbol{c}^H = \left[c_0^H \; c_1^H \; \dots c_{2^r-1}^H\right]$. For r being an even number, it has been shown that [36]

$$\left[c_0^H \oplus c_1^H \oplus c_2^H \oplus \cdots \oplus \; c_{2^{k-1}}^H \oplus \cdots \oplus \; c_{2^{r-1}}^H\right] \oplus c_{2^r-1}^H = 0. \tag{3}$$

Viewing from another perspective, if there is a length-$(r + 2)$ SPC codeword denoted by $\boldsymbol{c}_\mu = [c_{\mu_0} \; c_{\mu_1} \; \dots \; c_{\mu_r} \; c_{\mu_{r+1}}]$, these bits can be used as inputs to a systematic Hadamard encoder and form a Hadamard codeword where

$$c_0^H = c_{\mu_0}, \; c_1^H = c_{\mu_1}, \cdots, c_{2^{k-1}}^H = c_{\mu_k}, \cdots, \; c_{2^{r-1}}^H = c_{\mu_r}, \; c_{2^r-1}^H = c_{\mu_{r+1}} \tag{4}$$

correspond to $r+2$ P-VNs and the remaining Hadamard parity bits in $\boldsymbol{c}^H$ correspond to $2^r - (r+2)$ D1H-VNs. Fig. 4 shows an example in which a $(6, 5)$ SPC codeword is encoded into a length-16 ($r = 4$) Hadamard codeword.

Referring to Fig. 3, the links connecting the P-VNs to the i-th H-CN always form a SPC. These links can make use of the above mechanism to derive the parity bits of the Hadamard code (denoted as D1H-VNs of the Hadamard check node in Fig. 3) if d_{c_i} is even. In this case, the Hadamard code length equals $2^{d_{c_i}-2}$, and the number of D1H-VNs equals $2^{d_{c_i}-2} - d_{c_i}$. Assuming d_{c_i} is even for all $i = 0, 1, \dots, m-1$, the total number of D1H-VNs is given by $\sum_{i=0}^{m-1} \left(2^{d_{c_i}-2} - d_{c_i}\right)$. When all VNs are sent to the channel, the code rate of the protograph given in Fig. 3 equals

$$R^{\text{even}} = \frac{n-m}{\sum_{i=0}^{m-1} \left(2^{d_{c_i}-2} - d_{c_i}\right) + n}.$$

If we further assume that all rows in $\boldsymbol{B}_{m\times n}$ have the same weight which is equal to d, i.e., $d_{c_i} = d$ for all i, the code rate is simplified to

$$R^{\text{even}}_{d_{c_i}=d} = \frac{n-m}{m\left(2^{d-2} - d\right) + n}.$$

When $n_p (< n)$ P-VNs are punctured, the code rate becomes

$$R^{\text{even}}_{\text{punctured}} = \frac{n-m}{m\left(2^{d-2} - d\right) + n - n_p}.$$

2) $r = d_{c_i} - 2$ is an odd number: For r being an odd number, the 2^r Hadamard codewords in $+\boldsymbol{H}_q$ can satisfy (3) but all the 2^r Hadamard codewords in $-\boldsymbol{H}_q$ cannot. We apply the same non-systematic encoding method in [36] to encode the SPC codeword [2]. Supposing $\boldsymbol{c}_\mu$ is a SPC codeword, we preprocess $\boldsymbol{c}_\mu = \left[c_{\mu_0} \; c_{\mu_1} \dots \; c_{\mu_{r+1}}\right]$ to obtain $\boldsymbol{c}'_\mu = \left[c'_{\mu_0} \; c'_{\mu_1} \dots \; c'_{\mu_{r+1}}\right]$, and then we perform Hadamard encoding for $\boldsymbol{c}'_\mu$ to obtain $\boldsymbol{c}^H = \left[c_0^H \; c_1^H \; \dots c_{2^r-1}^H\right]$, where

$$\begin{aligned}
c_0^H &= c'_{\mu_0} &&= c_{\mu_0} \\
c_1^H &= c'_{\mu_1} &&= c_{\mu_1} \oplus c_{\mu_0} \\
&\vdots \\
c_{2^{k-1}}^H &= c'_{\mu_k} &&= c_{\mu_k} \oplus c_{\mu_0} \\
&\vdots \\
c_{2^{r-1}}^H &= c'_{\mu_r} &&= c_{\mu_r} \oplus c_{\mu_0} \\
c_{2^r-1}^H &= c'_{\mu_{r+1}} &&= c_{\mu_{r+1}}.
\end{aligned} \tag{5}$$

Fig. 5 shows an example in which a (5,4) SPC codeword is encoded into a length-8 ($r = 3$) Hadamard codeword. It can be seen that after the non-systematic encoding, only the first and last code bits are the same as the original information bits, i.e., $c_0^H = c'_{\mu_0} = c_{\mu_0}$ and $c_{2^r-1}^H = c'_{\mu_{r+1}} = c_{\mu_{r+1}}$. Thus we send the remaining code bits, i.e., c_1^H to $c_{2^r-2}^H$, to provide more channel observations for the decoder and the number of D1H-VNs equals $2^{d_{c_i}-2} - 2$. For example, the code bits $\left[c_1^H \; c_2^H \; c_3^H \; c_4^H \; c_5^H \; c_6^H\right]$ shown in Fig. 5 will be sent.

Assuming all the rows in $\boldsymbol{B}_{m\times n}$ have the same weight d, the code rate is given by

$$R^{\text{odd}}_{d_{c_i}=d} = \frac{n-m}{m\left(2^{d-2} - 2\right) + n}.$$

[2]Note that there are other non-systematic encoding methods, e.g., preprocess $\boldsymbol{c}_\mu = \left[c_{\mu_0} \; c_{\mu_1} \dots \; c_{\mu_{r+1}}\right]$ to obtain $\boldsymbol{c}'_\mu = \left[c'_{\mu_0} \; c'_{\mu_1} \dots \; c'_{\mu_{r+1}}\right]$, where $c'_{\mu_i} = c_{\mu_i}$ for $i = 0, 1, 2, \dots, r$; and $c'_{\mu_{r+1}} = c_{\mu_{r+1}} \oplus c_{\mu_0}$.

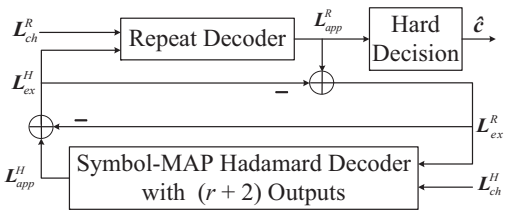


Fig. 6. Block diagram of a PLDPC-Hadamard decoder. The repeat decoder is the same as the variable-node processor used in LDPC decoder. For the symbol-MAP Hadamard decoder, the number of outputs is always $r+2$; the number of inputs is 2^r when r is even; the number of inputs is 2^r+r when r is odd.

If n_p $(<n)$ P-VNs are punctured, the code rate becomes

$$R^{\text{odd}}_{\text{punctured}} = \frac{n-m}{m\,(2^{d-2}-2)+n-n_p}.$$

Note that for $k=1,2,\ldots,r$,

- $c^H_{2^{k-1}} = c'_{\mu_k} = c_{\mu_k} \oplus c_0$ and hence $c_{\mu_k} = c^H_{2^{k-1}} \oplus c_0$;
- c_{μ_k} is transmitted as P-VN; and
- $c^H_{2^{k-1}}$ is transmitted as D1H-VN.

Thus the r information bits c_{μ_k} can have both the *a priori* information provided by the extrinsic information from P-VNs and the channel information of $c^H_{2^{k-1}} = c'_{\mu_k} = c_{\mu_k} \oplus c_0$ from D1H-VNs. However, the two information bits c_{μ_0} and $c_{\mu_{r+1}}$ only have the *a priori* information from P-VNs and the $2^r-(r+2)$ Hadamard parity bits only have the channel information from D1H-VNs. Supposing for every H-CN, n_h $(\le r)$ D1H-VNs corresponding to code bits $c^H_{2^{k-1}}$ $(k=1,2,\ldots,r)$ are also punctured. The code rate further becomes

$$R^{\text{odd}}_{\text{punctured D1H-VN}} = \frac{n-m}{m\,(2^{d-2}-2-n_h)+n-n_p}. \quad (6)$$

B. Decoder of the PLDPC-Hadamard Codes

To evaluate the performance of PLDPC-Hadamard codes, the iterative decoder shown in Fig. 6 is used. It consists of a repeat decoder and a symbol-MAP Hadamard decoder. The repeat decoder is the same as the variable-node processor used in an LDPC decoder and is therefore not described here.

As described in the previous section, each H-CN with an order-r Hadamard constraint is connected to $r+2$ P-VNs in the protograph of a PLDPC-Hadamard code. The symbol-MAP Hadamard decoder of order-r has a total of 2^r or 2^r+r inputs, among which $r+2$ come from the repeat decoder and are updated in each iteration; and the remaining inputs come from the channel LLR information which do not change during the iterative process. Moreover, the symbol-MAP Hadamard decoder will produce $r+2$ extrinsic LLR outputs which are fed back to the repeat decoder. The iterative process between the repeat decoder and symbol-MAP Hadamard decoder continues until the information bits corresponding to all Hadamard codes (after hard decision) become valid SPCs or the maximum number of iterations has been reached. In the following, we show the details of the operations of the symbol-MAP Hadamard decoder.

1) r is an even number: A H-CN has $r+2$ links to P-VNs and is connected to $2^r-(r+2)$ D1H-VNs. Specifically, we denote

- $\boldsymbol{L}^R_{ex} = [L^R_{ex}(0)\ L^R_{ex}(1)\ \cdots\ L^R_{ex}(r+1)]^T$ as the $r+2$ extrinsic LLR information values coming from the repeat decoder (P-VNs),
- $\boldsymbol{L}^H_{apr} = [L^H_{apr}(0)\ L^H_{apr}(1)\ \cdots\ L^H_{apr}(2^r-1)]^T$ as the 2^r *a priori* LLR values of $\boldsymbol{c}^H$,
- $\boldsymbol{y}^H_{ch} = [y^H_{ch}(0)\ y^H_{ch}(1)\ \cdots\ y^H_{ch}(2^r-1)]^T$ as the length-2^r channel observation vector corresponding to $\boldsymbol{c}^H$ and is derived from the D1H-VNs (note that $r+2$ channel observations are zero),
- $\boldsymbol{L}^H_{ch} = [L^H_{ch}(0)\ L^H_{ch}(1)\ \cdots\ L^H_{ch}(2^r-1)]^T$ as the length-2^r channel LLR observations corresponding to $\boldsymbol{c}^H$.

Based on (4) and the transmission mechanism, *a priori* LLR values exist only for the $r+2$ information bits in $\boldsymbol{c}^H$ and they are equal to the extrinsic LLR values $\boldsymbol{L}^R_{ex}$ from the repeat decoder. Correspondingly, channel LLR values only exist for the 2^r-r-2 Hadamard parity bits in $\boldsymbol{c}^H$ and they are obtained from the received channel observations $\boldsymbol{y}^H_{ch}$. In other words, only 2^r-r-2 entries in $\boldsymbol{y}^H_{ch}$ and also $\boldsymbol{L}^H_{ch}$ are non-zero. Thus the entries of $\boldsymbol{L}^H_{ch}$ and $\boldsymbol{L}^H_{apr}$ are assigned as

$$\begin{cases} L^H_{apr}(k) = L^R_{ex}(0) \\ L^H_{ch}(k) = \frac{2y^H_{ch}(0)}{\sigma^2_{ch}} = 0 \end{cases} \text{for } k=0;$$

$$\begin{cases} L^H_{apr}(k) = L^R_{ex}(i) \\ L^H_{ch}(k) = \frac{2y^H_{ch}(k)}{\sigma^2_{ch}} = 0 \end{cases} \text{for } k=1,2,\cdots,2^{i-1},\cdots,2^{r-1};$$

$$\begin{cases} L^H_{apr}(k) = L^R_{ex}(r+1) \\ L^H_{ch}(k) = \frac{2y^H_{ch}(k)}{\sigma^2_{ch}} = 0 \end{cases} \text{for } k=2^r-1;$$

$$\begin{cases} L^H_{apr}(k) = 0 \\ L^H_{ch}(k) = \frac{2y^H_{ch}(k)}{\sigma^2_{ch}} \end{cases} \text{for the } 2^r-r-2 \text{ remaining } k. \quad (7)$$

The symbol-MAP Hadamard decoder then computes the *a posteriori* LLR ($\boldsymbol{L}^H_{app}$) of the code bits using (1) and (2). By subtracting the *a priori* LLR values from the *a posteriori* LLR values, the extrinsic LLR values ($\boldsymbol{L}^H_{ex}$) can be obtained. Fig. 7 illustrates the flow of the computation of $\boldsymbol{L}^H_{app}$ and hence $\boldsymbol{L}^H_{ex}$ for $r=4$, which corresponds to $r+2=6$ information bits (and $2^r-(r+4)=10$ Hadamard parity bits).

2) r is an odd number: A H-CN is connected to $r+2$ P-VNs and 2^r-2 D1H-VNs, and the bits corresponding to the $r+2$ P-VNs form a SPC codeword $\boldsymbol{c}_\mu$. We use the same notations as in the "r is an even number" case. However, for $\boldsymbol{y}^H_{ch}$, only the first and the last channel observations are zero. Since non-systematic Hadamard code is used, $\boldsymbol{c}_\mu$ does not represent the information bits in $\boldsymbol{c}^H$ for $c_{\mu_0}=$ "1". Thus, we cannot directly apply (1) to obtain the *a posteriori* LLR of $\boldsymbol{c}_\mu$. Here, we present the decoding steps when r is odd.

Referring to Sect. III-A2, the assignment of $\boldsymbol{L}^H_{apr}$ depends on c_{μ_0}. For convenience of explanation, we denote $\boldsymbol{L}^{+H}_{apr}/\boldsymbol{L}^{-H}_{apr}$ as the assignment of $\boldsymbol{L}^H_{apr}$ for $c_{\mu_0}=$ "0"/"1", respectively;

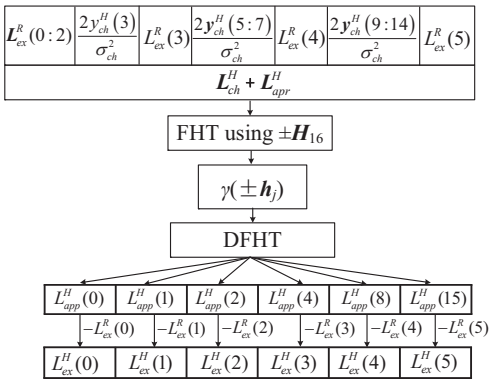


Fig. 7. Operations in the symbol-MAP Hadamard decoder for $r = 4$, i.e., 16 LLR inputs and 6 output LLR values for the information bits.

and we assign $\boldsymbol{L}_{apr}^{\pm H}$ using

$$\begin{cases} L_{apr}^{\pm H}(k) = L_{ex}^R(0) & \text{for } k = 0; \\ \left.\begin{array}{l} L_{apr}^{+H}(k) = L_{ex}^R(i) \\ L_{apr}^{-H}(k) = -L_{ex}^R(i) \end{array}\right. & \text{for } k = 1, 2, \cdots, 2^{i-1}, \cdots, 2^{r-1}; \\ L_{apr}^{\pm H}(k) = L_{ex}^R(r+1) & \text{for } k = 2^r - 1; \\ L_{apr}^{\pm H}(k) = 0 & \text{for the } 2^r - r - 2 \text{ remaining } k. \end{cases} \tag{8}$$

Moreover, we assign $\boldsymbol{L}_{ch}^H$ using

$$\begin{cases} L_{ch}^H(k) = \frac{2y_{ch}^H(k)}{\sigma_{ch}^2} & \text{for } k = 1, 2, \cdots, 2^r - 2; \\ L_{ch}^H(k) = \frac{2y_{ch}^H(k)}{\sigma_{ch}^2} = 0 & \text{for } k = 0, 2^r - 1. \end{cases} \tag{9}$$

Since the first bit in all $+\boldsymbol{h}_j/-\boldsymbol{h}_j$ is "0"/"1" (+1 mapped to "0" and -1 to "1"), we apply $\boldsymbol{L}_{apr}^{\pm H}$ and $\boldsymbol{L}_{ch}^H$ to compute $\gamma(\pm\boldsymbol{h}_j)$, i.e., $\gamma(\pm\boldsymbol{h}_j) = \exp\left(\langle \pm\boldsymbol{h}_j, \boldsymbol{L}_{ch}^H + \boldsymbol{L}_{apr}^{\pm H}\rangle/2\right)$. We also define the $r+2$ *a posteriori* LLR values ($\boldsymbol{L}_{app}^H$) of the original bits $\boldsymbol{c}_\mu$ by

$$\boldsymbol{L}_{app}^H = [L_{app}^H(0)\ L_{app}^H(1)\ \cdots\ L_{app}^H(2^{i-1})\ \cdots\ L_{app}^H(2^{r-1})\ L_{app}^H(2^r-1)]^T. \tag{10}$$

Then we compute $L_{app}^H(0)$ and $L_{app}^H(2^r-1)$ using (1) and (2); and compute $L_{app}^H(2^{i-1})$, $i = 1, 2, \cdots, r$, using DFHT and

$$L_{app}^H(2^{i-1}) = \ln \frac{\sum\limits_{+H[2^{i-1},j]=+1} \gamma(+\boldsymbol{h}_j) + \sum\limits_{-H[2^{i-1},j]=+1} \gamma'(-\boldsymbol{h}_j)}{\sum\limits_{+H[2^{i-1},j]=-1} \gamma(+\boldsymbol{h}_j) + \sum\limits_{-H[2^{i-1},j]=-1} \gamma'(-\boldsymbol{h}_j)} \tag{11}$$

where $\gamma'(-\boldsymbol{h}_j) = \gamma(-\boldsymbol{h}_{2^r-1-j})$, $j = 0, 1, \ldots, 2^r - 1$. Then $\boldsymbol{L}_{ex}^H = \boldsymbol{L}_{app}^H - \boldsymbol{L}_{ex}^R$ of length $r+2$ is computed and fed back to the repeat decoder. The steps to compute $\boldsymbol{L}_{ex}^H$ for the case $r = 3$ is shown in Fig. 8.

Remark: If some more bits in $c_{2^{k-1}}^H$ for $k = 1, 2, \ldots, r$ are punctured, the corresponding channel observation $y_{ch}^H(2^{k-1})$ and LLR values of $L_{ch}^H(2^{k-1})$ are set to 0 and the overall code rate will slightly increase.

C. Code Design Optimization

We propose a low-complexity PEXIT algorithm for analyzing PLDPC-Hadamard codes. Our low-complexity PEXIT algorithm uses the same MI updating method as the original PEXIT algorithm [20] for the P-VNs. However, our algorithm computes extrinsic MI for the symbol-MAP Hadamard decoder whereas the original PEXIT algorithm computes extrinsic MI for the SPC decoder. We use Monte Carlo method in obtaining the extrinsic MI values of the symbol-MAP Hadamard decoder. The algorithm not only has a low complexity, but also is generic and applicable to analyzing both systematic and non-systematic Hadamard codes.

We define the following symbols.

- $I_{av}(i,j)$: the *a priori* mutual information (MI) from the i-th H-CN to the j-th P-VN;
- $I_{ev}(i,j)$: extrinsic MI from the j-th P-VN to the i-th H-CN;
- $I_{ah}(i,k)$: the *a priori* MI of the k-th information bit in the i-th H-CN;
- $I_{eh}(i,k)$: extrinsic MI of the k-th information bit in the i-th H-CN;
- $I_{app}(j)$ the *a posteriori* MI of the j-th P-VN.

Referring to Fig. 3, the channel LLR value L_{ch} follows a normal distribution $\mathcal{N}(\sigma_{L_{ch}}^2/2, \sigma_{L_{ch}}^2)$ where $\sigma_{L_{ch}}^2 = 8R \cdot E_b/N_0$ and R is the code rate. When the output MI of a decoder is I, the corresponding LLR values of the extrinsic information obey a normal distribution $\mathcal{N}(\pm\sigma^2/2, \sigma^2)$. The relationship between I and σ can be approximately computed by the functions $I = J(\sigma)$ and $\sigma = J^{-1}(I)$ in [9], [13].

1) Modified PEXIT Algorithm: To generate the PEXIT curves for the repeat decoder and symbol-MAP Hadamard decoder, we apply the following steps for a given set of protomatrix $\boldsymbol{B}_{m\times n}$ (e.g., (12)), code rate R and E_b/N_0 in dB (denoted as $E_b/N_0(\text{dB})$).

i) Compute $\sigma_{L_{ch}} = \left(8 \cdot R \cdot 10^{(E_b/N_0(\text{dB}))/10}\right)^{1/2}$ for L_{ch}.
ii) For $i = 0, 1, \ldots, m-1$ and $j = 0, 1, \ldots, n-1$, set $I_{av}(i,j) = 0$.
iii) For $i = 0, 1, \ldots, m-1$ and $j = 0, 1, \ldots, n-1$, compute (13) if $b_{i,j} > 0$; else set $I_{ev}(i,j) = 0$. Taking the 3×4 protomatrix in (12) as an example, the weight of each row is $d = 6$ and hence $r + 2 = 6 \Rightarrow r = 4$. After analyzing the MI of the P-VNs, the corresponding 3×4 $\{I_{ev}(i,j)\}$ MI matrix can be represented by (14).
iv) Convert the $m \times n$ $\{I_{ev}(i,j)\}$ MI matrix into an $m \times d$ $\{I_{ah}(i,k)\}$ MI matrix by eliminating the 0 entries and repeating $\{I_{ev}(i,j)\}$ $b_{i,j}(\geq 1)$ times in the same row. Using the previous example, the 3×4 $\{I_{ev}(i,j)\}$ MI matrix is converted into the 3×6 $\{I_{ah}(i,k)\}$ MI matrix shown in (15).
v) For $i = 0, 1, \ldots, m-1$, using the d entries in the i-th row of I_{ah} and $\sigma_{L_{ch}}^2$, generate a large number of sets of LLR values as inputs to the symbol-MAP Hadamard decoder and record the output extrinsic LLR values of the k-th information bit ($k = 0, 1, \ldots, d-1$). Compute the extrinsic MI of the information bit using (16), where $p_e(\xi|X = x)$ denotes the PDF of the LLR values given the bit x being "0" or "1". Form the extrinsic MI matrix

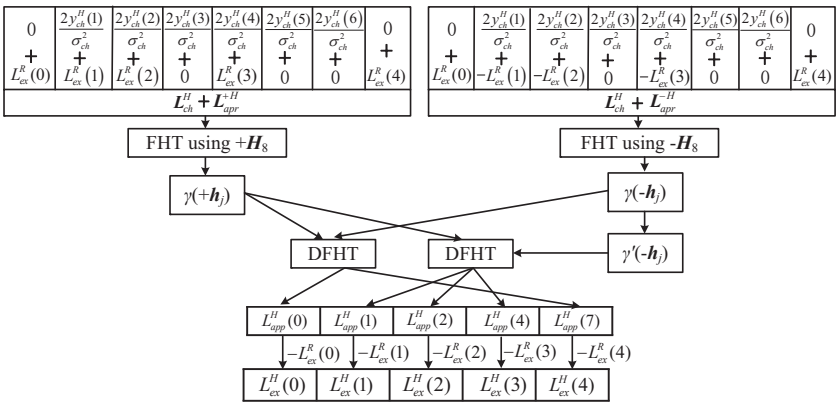


Fig. 8. Operations in the symbol-MAP Hadamard decoder for $r = 3$, i.e., 11 LLR inputs and 5 output LLR values for the information bits.

$$\boldsymbol{B}_{3\times 4} = \begin{bmatrix} 2 & 0 & 2 & 2 \\ 0 & 2 & 2 & 2 \\ 3 & 2 & 0 & 1 \end{bmatrix} \tag{12}$$

$$I_{ev}(i,j) = J\left(\sqrt{\sum_{s\neq i} b_{s,j}\left(J^{-1}\left(I_{av}\left(s,j\right)\right)\right)^2 + \left(b_{i,j}-1\right)\cdot\left(J^{-1}\left(I_{av}\left(i,j\right)\right)\right)^2 + \sigma^2_{L_{ch}}}\right) \tag{13}$$

$$I_{ev} = \begin{bmatrix} I_{ev}\left(0,0\right) & 0 & I_{ev}\left(0,2\right) & I_{ev}\left(0,3\right) \\ 0 & I_{ev}\left(1,1\right) & I_{ev}\left(1,2\right) & I_{ev}\left(1,3\right) \\ I_{ev}\left(2,0\right) & I_{ev}\left(2,1\right) & 0 & I_{ev}\left(2,3\right) \end{bmatrix} \tag{14}$$

$$\begin{aligned} I_{ah} &= \begin{bmatrix} I_{ah}\left(0,0\right) & I_{ah}\left(0,1\right) & I_{ah}\left(0,2\right) & I_{ah}\left(0,3\right) & I_{ah}\left(0,4\right) & I_{ah}\left(0,5\right) \\ I_{ah}\left(1,0\right) & I_{ah}\left(1,1\right) & I_{ah}\left(1,2\right) & I_{ah}\left(1,3\right) & I_{ah}\left(1,4\right) & I_{ah}\left(1,5\right) \\ I_{ah}\left(2,0\right) & I_{ah}\left(2,1\right) & I_{ah}\left(2,2\right) & I_{ah}\left(2,3\right) & I_{ah}\left(2,4\right) & I_{ah}\left(2,5\right) \end{bmatrix} \\ &= \begin{bmatrix} I_{ev}\left(0,0\right) & I_{ev}\left(0,0\right) & I_{ev}\left(0,2\right) & I_{ev}\left(0,2\right) & I_{ev}\left(0,3\right) & I_{ev}\left(0,3\right) \\ I_{ev}\left(1,1\right) & I_{ev}\left(1,1\right) & I_{ev}\left(1,2\right) & I_{ev}\left(1,2\right) & I_{ev}\left(1,3\right) & I_{ev}\left(1,3\right) \\ I_{ev}\left(2,0\right) & I_{ev}\left(2,0\right) & I_{ev}\left(2,0\right) & I_{ev}\left(2,1\right) & I_{ev}\left(2,1\right) & I_{ev}\left(2,3\right) \end{bmatrix} \end{aligned} \tag{15}$$

$$I_E = \frac{1}{2}\sum_{x\in\{0,1\}}\int_{-\infty}^{\infty} p_e\left(\xi|X=x\right)\log_2\frac{2\cdot p_e\left(\xi|X=x\right)}{p_e\left(\xi|X=\text{“0”}\right)+p_e\left(\xi|X=\text{“1”}\right)}d\xi \tag{16}$$

$$I_{eh} = \begin{bmatrix} I_{eh}\left(0,0\right) & I_{eh}\left(0,1\right) & I_{eh}\left(0,2\right) & I_{eh}\left(0,3\right) & I_{eh}\left(0,4\right) & I_{eh}\left(0,5\right) \\ I_{eh}\left(1,0\right) & I_{eh}\left(1,1\right) & I_{eh}\left(1,2\right) & I_{eh}\left(1,3\right) & I_{eh}\left(1,4\right) & I_{eh}\left(1,5\right) \\ I_{eh}\left(2,0\right) & I_{eh}\left(2,1\right) & I_{eh}\left(2,2\right) & I_{eh}\left(2,3\right) & I_{eh}\left(2,4\right) & I_{eh}\left(2,5\right) \end{bmatrix} \tag{17}$$

$$\begin{aligned} I_{av} &= \begin{bmatrix} I_{av}\left(0,0\right) & 0 & I_{av}\left(0,2\right) & I_{av}\left(0,3\right) \\ 0 & I_{av}\left(1,1\right) & I_{av}\left(1,2\right) & I_{av}\left(1,3\right) \\ I_{av}\left(2,0\right) & I_{av}\left(2,1\right) & 0 & I_{av}\left(2,3\right) \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{2}\sum_{k=0}^{1} I_{eh}(0,k) & 0 & \frac{1}{2}\sum_{k=2}^{3} I_{eh}(0,k) & \frac{1}{2}\sum_{k=4}^{5} I_{eh}(0,k) \\ 0 & \frac{1}{2}\sum_{k=0}^{1} I_{eh}(1,k) & \frac{1}{2}\sum_{k=2}^{3} I_{eh}(1,k) & \frac{1}{2}\sum_{k=4}^{5} I_{eh}(1,k) \\ \frac{1}{3}\sum_{k=0}^{2} I_{eh}(2,k) & \frac{1}{2}\sum_{k=3}^{4} I_{eh}(2,k) & 0 & I_{eh}(2,5) \end{bmatrix} \end{aligned} \tag{18}$$

$\{I_{eh}(i,k)\}$ of size $m\times d$. (Details of the method is shown in Appendix B.) Using the previous example, the matrix is represented by (17).

Remark: Our technique makes use of multiple *a priori* MI values ($\{I_{ah}(i,k)\}$) as well as channel information $\sigma_{L_{ch}}$ and produces multiple extrinsic MI values ($\{I_{eh}(i,k)\}$). In [46], an EXIT function of symbol-MAP Hadamard decoder under the AWGN channel is obtained. However, the function involves very high computational complexity, which increases rapidly with an increase of the Hadamard order r. The function also cannot be used for analyzing non-systematic Hadamard codes. In [36], simulation is used to characterize the symbol-MAP Hadamard decoder but the method is based on a single *a priori* MI value as well as channel information and produces only one output extrinsic MI.

vi) Convert the $m\times d$ $\{I_{eh}(i,k)\}$ MI matrix into an $m\times n$ $\{I_{av}(i,j)\}$ MI matrix. For $i = 0,1,\ldots,m-1$ and $j = 0,1,\ldots,n-1$; if $b_{i,j} > 0$, set the value of $I_{av}(i,j)$ as the average of the corresponding $b_{i,j}$ MI values in the i-th row of $\{I_{eh}(i,k)\}$; else set $I_{av}(i,j) = 0$. In the above example, $\{I_{av}(i,j)\}$ becomes (18).

Algorithm 1: Searching $\boldsymbol{B}_{m\times n}$ with a low threshold

```
Generate a random protomatrix B_{m×n} according to the corresponding constraints;
E_b/N_0(dB) = −1.40 dB;
while E_b/N_0(dB) > −1.59 dB do
    σ_{L_ch} = sqrt(8 · R · 10^{(E_b/N_0(dB))/10});
    I_ch(j) = J(σ_{L_ch}) for ∀ j;
    I_av(i, j) = 0 for ∀ i, j;
    It = 0;
    while It < 300 do
        Use the proposed PEXIT algorithm to analyze B_{m×n} and obtain I_app(j) for j = 0, 1, ..., n − 1;
        if I_app(j) = 1 for ∀ j then
            E_b/N_0(dB) = E_b/N_0(dB) − 0.01 dB; Goto line 3;
        It = It + 1;
    Break;
Threshold equals E_b/N_0(dB) + 0.01 dB.
```

vii) Repeat Steps iii) to vi) until the maximum number of iterations is reached; or when $I_{app}(j) = 1$ for all $j = 0, 1, \ldots, n-1$ where $I_{app}(j) = J\left(\sqrt{\sum_{i=0}^{m-1} b_{i,j}(J^{-1}\left(I_{av}\left(i,j\right)\right))^2 + \sigma_{L_{ch}}^2}\right)$.

Note that our PEXIT algorithm can be used to analyze PLDPC-Hadamard designs with degree-1 and/or punctured VNs. In case of puncturing, the corresponding channel LLR values in the analysis will be set to zero.

2) Optimization Criterion: For a given code rate, our objective is to find a protograph of the PLDPC-Hadamard code such that it achieves $I_{app}(j) = 1\ \forall\ j$ within a fixed number of iterations and with the lowest threshold E_b/N_0. To reduce the search space, we impose the following constraints: the weights of all rows in the protomatrix are fixed at d; the maximum column weight, the minimum column weight, and the maximum value of each entry in protomatrix are preset according to the code rate and order of the Hadamard code; the maximum number of iterations used in the PEXIT algorithm is set to 300; and a target threshold is set to below -1.40 dB.

Algorithm 1 shows the steps to find a protomatrix with a low threshold. A protomatrix is first randomly generated according to the constraints above. Then it is iteratively analyzed by the PEXIT algorithm to see if the corresponding PEXIT curves converge under the current E_b/N_0 (dB). If the protomatrix is found satisfying $I_{app}(j) = 1$ for all j, E_b/N_0 (dB) is reduced by 0.01 dB and the protomatrix is analyzed again. If the number of iterations reaches 300 and the condition $I_{app}(j) = 1$ for all j is not satisfied, the analysis is terminated and the E_b/N_0 threshold is determined. The process is repeated until a protomatrix with a satisfactory E_b/N_0 threshold is found. (On average, the PEXIT algorithm takes 35s (for $r = 4$) to 120s (for $r = 10$) to determine the threshold of a protomatrix. Using annealing approaches or genetic algorithms to generate the protomatrices would speed up the search and is part of our on-going research effort.)

IV. Simulation Results

When a protomatrix $\boldsymbol{B}_{m\times n} = \{b(i,j)\}$ with low threshold is found, we use a two-step lifting mechanism together with the progressive edge-growth (PEG) method [15] to construct an LDPC code. In the first step, we "lift" a base matrix $\{b(i,j)\}$ by replacing each non-zero entry $b(i,j)$ with a summation of $b(i,j)$ different $z_1 \times z_1$ permutation matrices and replacing each zero entry with the $z_1 \times z_1$ zero matrix. After the first lifting process, all entries in the lifted matrix are either "0" or "1". In the second step, we lift the resultant matrix again by replacing each entry "1" with a $z_2 \times z_2$ circulant permutation matrix (CPM), and replacing each entry "0" with the $z_2 \times z_2$ zero matrix. As can be seen, the final connection matrix can be easily represented by a series of CPMs. Note that in each lifting step, the permutation matrices and CPMs are selected using the PEG algorithm [15] such that the girth (shortest cycle) in the resultant matrix can be maximized. Subsequently, each CN will be replaced by a Hadamard CN connected to an appropriate number of D1H-VNs. Without loss of generality, we transmit all-zero codewords. Moreover, the code bits are modulated using binary phase-shift-keying and sent through an AWGN channel. The maximum number of iterations performed by the decoder is 300. At a particular E_b/N_0, we run the simulation until 100 frame errors are collected.

A. Unpunctured PLDPC-Hadamard Codes

1) $r = 4$ and $d = r+2 = 6$: We attempt to find a PLDPC-Hadamard code with a target code rate of approximately 0.05. We select a protomatrix $\boldsymbol{B}_{7\times 11}$ of size 7×11, i.e., $m = 7$ and $n = 11$, and hence the code rate equals $R = 0.0494$. Moreover, we set the minimum column weight to 1, maximum column weight to 9, and maximum entry value to 3. Using the proposed analytical method under the constraints above, we find the following protomatrix which has a theoretical threshold of -1.42 dB.

$$\boldsymbol{B}_{7\times 11} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 0 & 1 \\ 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 2 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 0 \\ 3 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 2 & 0 \end{bmatrix} \tag{19}$$

Fig. 9 plots the PEXIT curves of the repeat decoder and the symbol-MAP Hadamard decoder under $E_b/N_0 = -1.42$ dB. It can be observed that the two curves are matched. By lifting the protomatrix with factors $z_1 = 32$ and $z_2 = 512$, we obtain a PLDPC-Hadamard code with information length $k = z_1 z_2 (n - m) = 65,536$ and code length $N_{total} = z_1 z_2 [m(2^{d-2} - d) + n] = 1,327,104$. (See Appendix C in [47] for details of the code structure after the lifting process.)

The BER and FER results of the PLDPC-Hadamard code are plotted in Fig. 10. Our code achieves a BER of 10^{-5} at $E_b/N_0 = -1.19$ dB, which is 0.23 dB from the threshold. At a BER of 10^{-5}, the gaps of our rate-0.0494 PLDPC-Hadamard code to the Shannon capacity for $R = 0.05$ and to the ultimate Shannon limit are 0.25 dB and 0.40 dB, respectively. Compared with the LDPC-Hadamard code in [36] which uses

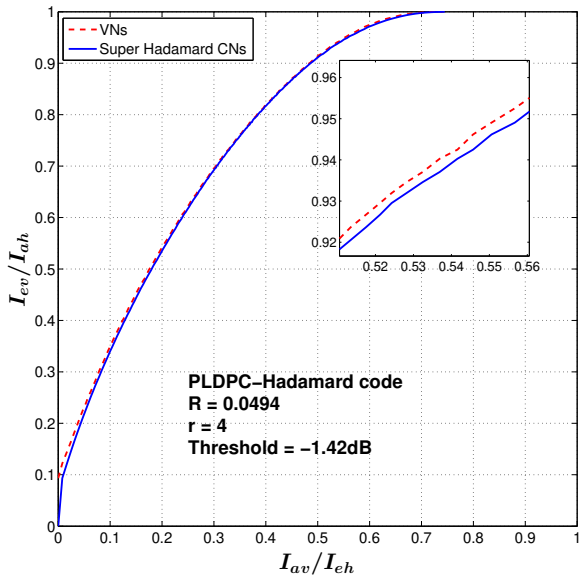


Fig. 9. The PEXIT chart of the PLDPC-Hadamard code given in (19) with $R = 0.0494$ and $r = 4$.

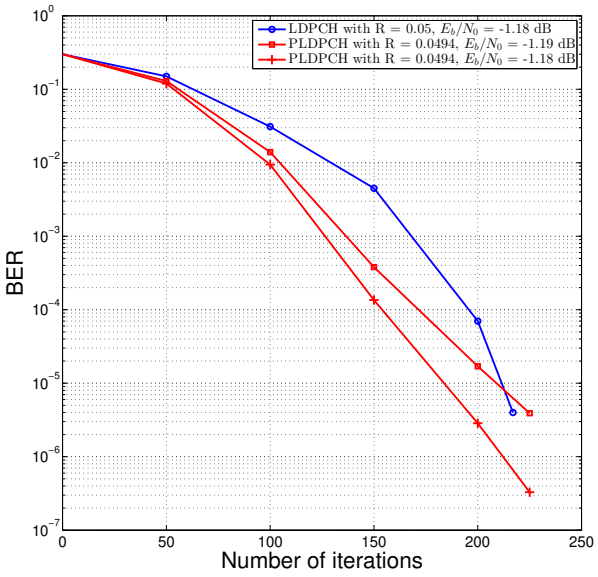


Fig. 11. BER performance versus number of iterations for the LDPC-Hadamard code in [36] ($E_b/N_0 = -1.18$ dB) and PLDPC-Hadamard code ($E_b/N_0 = -1.18$ and -1.19 dB). $r = 4$ and $k = 65,536$.

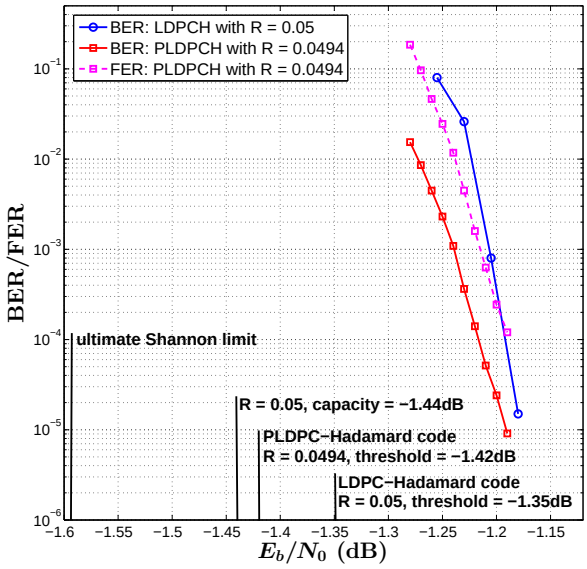


Fig. 10. BER (red curve) and FER (pink curve) performance of the proposed PLDPC-Hadamard code compared with the BER of the LDPC-Hadamard code (blue curve) in [36]. $r = 4$ and $k = 65,536$.

$R = 0.05$ and $r = 4$, our proposed PLDPC-Hadamard code has a slight performance improvement.

In Fig. 11, we further compare the BER results of the rate-0.05 LDPC-Hadamard code in [36] at $E_b/N_0 = -1.18$ dB and our rate-0.0494 PLDPC-Hadamard code at $E_b/N_0 = -1.18$ dB and -1.19 dB under different number of iterations. Note that the result of the LDPC-Hadamard code is the average from 20 simulations [36], whereas our result is the average from $10,000$ simulations. In other words, our simulation results are statistically very accurate due to the large number of simulations involved. For the same number of iterations and at $E_b/N_0 = -1.18$ dB, our PLDPC-Hadamard code produces a lower BER compared with the LDPC-Hadamard code in [36]. When our proposed PLDPC-Hadamard code operates at a slightly lower E_b/N_0, i.e., -1.19 dB, the BER of the proposed code still outperforms the conventional code except for iteration numbers beyond 200. Thus, we conclude that the proposed code achieves a faster convergence rate compared with the conventional code. In particular, our results are more precise because $10,000$ simulations are used for our code compared with only 20 simulations used for the conventional code in [36].

2) $r = 5$ and $d = 7$: We attempt to search a PLDPC-Hadamard code with a target code rate of approximately $R \approx 0.02$. We select $m = 6$ and $n = 10$, and obtain a code rate of $R = 0.021$. Other constraints of the protomatrix are the same as in the "$r = 4$" case. The following protomatrix with a threshold of -1.51 dB is found.

$$\boldsymbol{B}_{6\times 10} = \begin{bmatrix} 3 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 0 & 2 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 1 & 0 & 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 3 \\ 0 & 0 & 0 & 2 & 0 & 0 & 1 & 2 & 0 & 2 \\ 2 & 0 & 1 & 1 & 0 & 0 & 0 & 2 & 0 & 1 \end{bmatrix}. \quad (20)$$

The same lifting factors $z_1 = 32$ and $z_2 = 512$ are used to expand $\boldsymbol{B}_{6\times 10}$. The rate-0.021 PLDPC-Hadamard code has an information length of $k = z_1 z_2 (n - m) = 65,536$ and a code length of $N_{total} = z_1 z_2 [m(2^{d-2} - 2) + n] = 3,112,960$. Fig. 12 shows the PEXIT chart of the code at $E_b/N_0 = -1.51$ dB. We can observe that the two curves do not crossed and are matched.

Fig. 13 show that the code achieves a BER of 1.4×10^{-5} and a FER of 1.3×10^{-4} at $E_b/N_0 = -1.24$ dB (red curve), which is 0.27 dB away from the designed threshold.

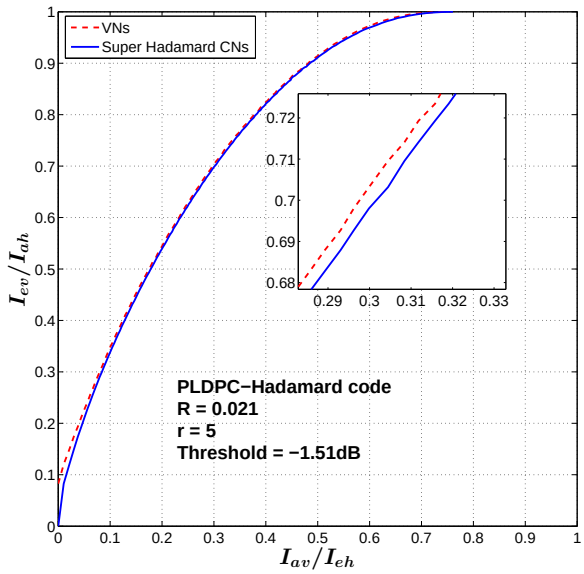


Fig. 12. The PEXIT chart of the PLDPC-Hadamard code given in (20) with $R = 0.021$ and $r = 5$.

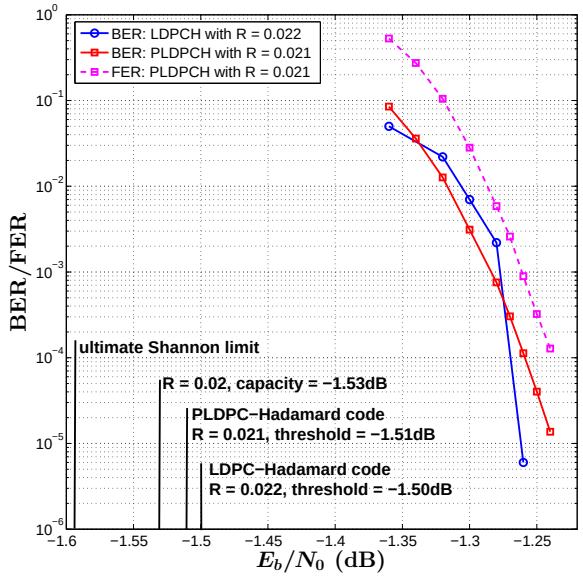


Fig. 13. BER (red curve) and FER (pink curve) performance of the proposed PLDPC-Hadamard code compared with the BER of the LDPC-Hadamard code (blue curve) in [36]. $r = 5$ and $k = 65,536$.

Compared with the BER curve (blue curve) of the rate-0.022 LDPC-Hadamard code in [36], our PLDPC-Hadamard code can achieve comparable results. At a BER of 10^{-5}, the gaps to the Shannon capacity of $R = 0.020$ and to the ultimate Shannon limit are 0.29 dB and 0.35 dB, respectively.

3) $r = 8$ and $d = 10$: A rate-0.008 PLDPC-Hadamard code is constructed using $m = 5$ and $n = 15$. Compared with the previous cases, the maximum column weight is now increased to 11. The aim is to allow the PEXIT curves to be matched at a lower E_b/N_0. The following protomatrix is

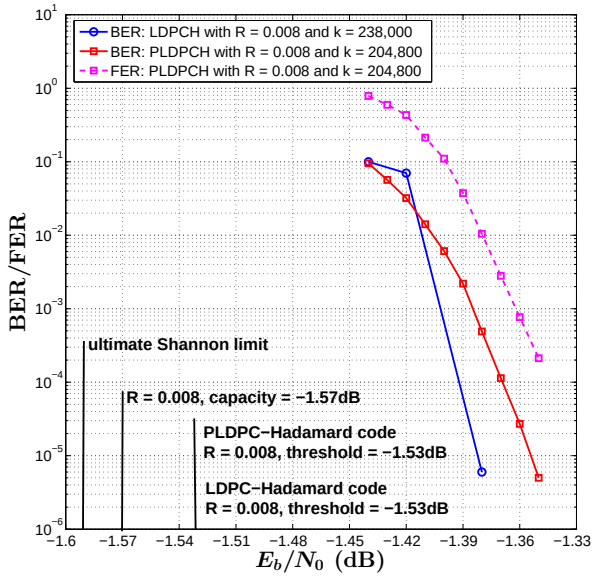


Fig. 14. BER (red curve) and FER (pink curve) performance of the proposed PLDPC-Hadamard code compared with the BER of the LDPC-Hadamard code (blue curve) in [36]. $r = 8$.

found with a threshold of -1.53 dB, which is the same as the rate-0.008 LDPC-Hadamard code in [36].

$$\boldsymbol{B}_{5\times 15} = \begin{bmatrix} 2 & 0 & 1 & 0 & 0 & 0 & 0 & 3 & 2 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 2 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 3 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 & 2 & 0 & 0 & 1 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 2 & 3 & 0 & 0 \end{bmatrix} \tag{21}$$

We use the lifting factors $z_1 = 16$ and $z_2 = 1280$. Fig. 14 shows that at $E_b/N_0 = -1.35$ dB, the PLDPC-Hadamard code achieves a FER of 2.1×10^{-4} and a BER of 3.8×10^{-6}, which is 0.18 dB away from the designed threshold. Compared with the BER curve in [36], there is a performance gap of about 0.03 dB at a BER of 10^{-5}. At the same BER, the gaps of our code to the rate-0.008 Shannon limit and to the ultimate Shannon limit are 0.22 dB and 0.24 dB, respectively.

4) $r = 10$ and $d = 12$: A rate-0.00295 PLDPC-Hadamard code shown in (22) is constructed using $m = 6$ and $n = 24$. (The target rate is approximately 0.003.) Compared with the case "$r = 8$", the maximum entry value in the protomatrix is increased to 4. The PLDPC-Hadamard code has a theoretical threshold of -1.53 dB, which is slightly higher (0.02 dB) than that of the LDPC-Hadamard code in [36]. We use the lifting factors $z_1 = 20$ and $z_2 = 1280$. Fig. 15 shows that our PLDPC-Hadamard code achieves a BER of 2.8×10^{-6} at $E_b/N_0 = -1.43$ dB, which is 0.01 dB higher compared with the LDPC-Hadamard code in [36]. However, our code is 29.11% shorter compared with the code in [36]. This performance has a 0.10 dB gap from the designed threshold. The gaps of our code to the rate-0.003 Shannon limit and to the

$$\boldsymbol{B}_{6\times 24} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 & 2 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 3 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 0 & 1 & \\ 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 3 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 3 & 2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{22}$$

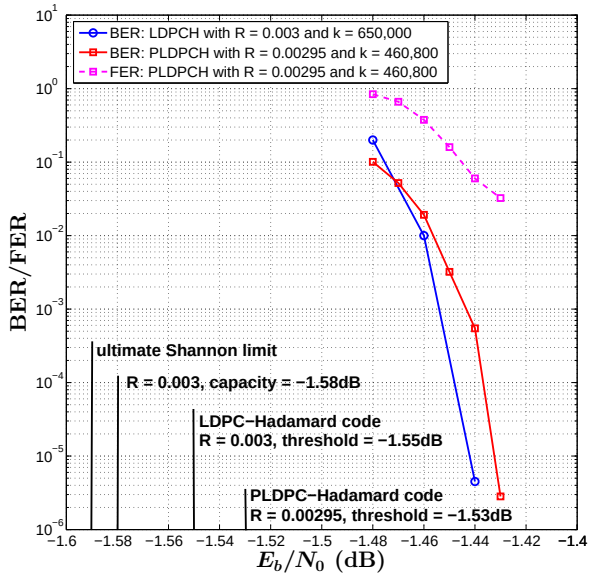


Fig. 15. BER (red curve) and FER (pink curve) performance of the proposed PLDPC-Hadamard code compared with the BER of the LDPC-Hadamard code (blue curve) in [36]. $r = 10$.

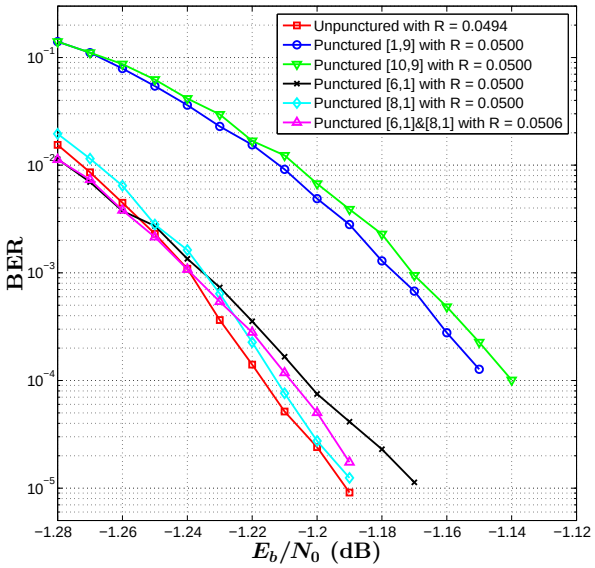


Fig. 16. BER performance of unpunctured/punctured PLDPC-Hadamard codes. One or two P-VNs is/are punctured. $r = 4$ and $k = 65,536$.

ultimate Shannon limit are 0.15 dB and 0.16 dB, respectively.

Remark: For cases with $r = 5, 8$ and 10 (Figs. 13, 14 and 15), the BER results may appear that our proposed PLDPC-Hadamard codes are slightly outperformed by the LDPC-Hadamard codes in [36] at the high E_b/N_0 region. For our codes, we keep running the simulations until 100 block errors are recorded. Thus our reported results have a high degree of accuracy. However, the stopping criterion of the LDPC-Hadamard code simulation in [36] is not known. If an inadequate number of simulations are performed, there could be some statistical difference between the actual error performance and the reported results.

B. Punctured PLDPC-Hadamard Codes

When a code is punctured, the code rate increases. The signals corresponding to the punctured variable nodes are not sent to the receiver and hence their channel LLR values are initialized to zero. In this section, we evaluate the performance of the PLDPC-Hadamard codes designed in the previous section when the codes are punctured. We use α to denote a column number in a protomatrix and β to denote the weight of a column. For example in the protomatrix shown in (19), $[4, 6]$ refers to the 4-th column $[0\ 0\ 0\ 3\ 0\ 2\ 1]^T$ which has a column weight of 6. Thus we use "punctured $[\alpha, \beta]$" to denote a PLDPC-Hadamard code in which the P-VN corresponding to the α-th column in the protomatrix is punctured. Also, the punctured P-VN has a degree of β.

1) $r = 4$: We puncture one P-VN with the largest degree (i.e., 9) or lowest degree (i.e., 1) in (19). Four cases are considered, i.e., $[1, 9]$, $[10, 9]$, $[6, 1]$ and $[8, 1]$. After puncturing, all codes have a rate of 0.0500. Fig. 16 shows that at a BER of 10^{-4}, punctured $[10, 9]$, $[1, 9]$, $[6, 1]$ and $[8, 1]$ have performance losses of about 0.075 dB, 0.065 dB, 0.012 dB and 0.004 dB, respectively, compared with the unpunctured code. Fig. 17 plots the FER of the unpunctured/punctured codes and it shows a similar relative error performance. We also simulate the code when both $[6, 1]$ and $[8, 1]$ P-VNs are punctured. The error performance of the code, as shown in Figs. 16 and Fig. 17, is found to be between punctured $[6, 1]$ and $[8, 1]$. The aforementioned results conclude that punctured $[8, 1]$ outperforms other punctured codes being considered and has a very similar performance as the unpunctured code.

2) $r = 5$: We puncture $[8, 9]$ (largest degree) and $[9, 1]$ (lowest degree) in (20), respectively. After puncturing, both codes have a rate of 0.02116. Fig. 18 shows that punctured $[9, 1]$ achieves the lowest BER while punctured $[8, 9]$ achieves the lowest FER.

We further consider puncturing D1H-VNs corresponding to code bits $c^H_{2^{k-1}}$ $(k = 1, 2, \ldots, r)$ for every H-CN. The rate of such punctured codes is computed using (6). We use $[c^H_1\ c^H_2\ \cdots c^H_{2^{k-1}}]$ $(1 \le k \le r)$ to denote the set of bits being punctured. Three sets of punctured bits are being considered.

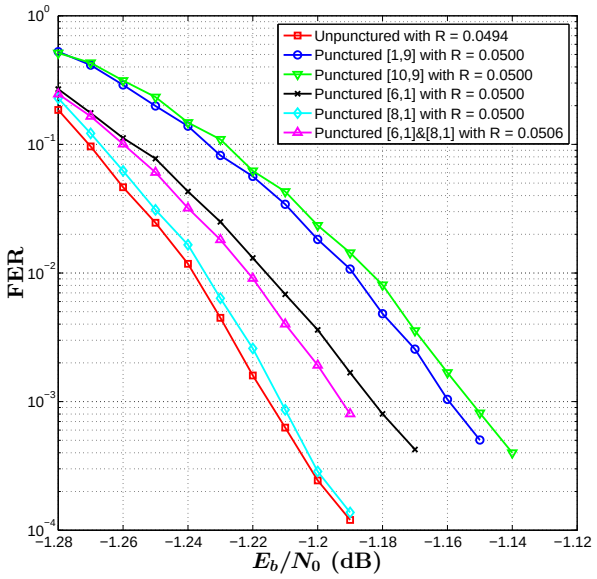


Fig. 17. FER performance of unpunctured/punctured PLDPC-Hadamard codes. One or two P-VNs is/are punctured. $r = 4$ and $k = 65,536$.

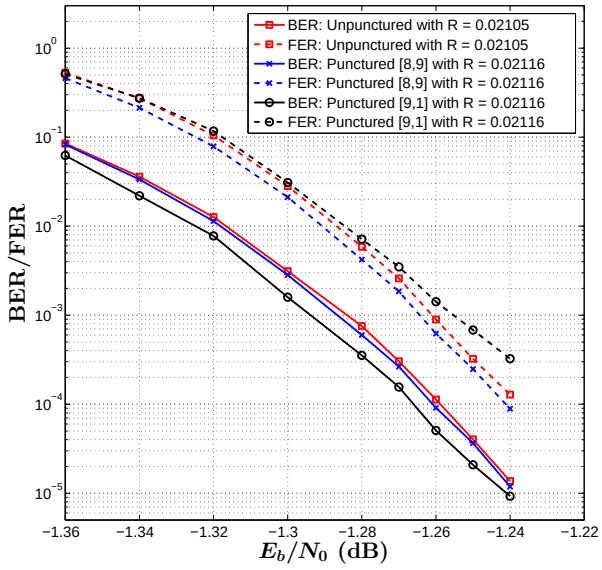


Fig. 18. BER/FER performance of unpunctured/punctured PLDPC-Hadamard codes. One P-VN is punctured. $r = 5$ and $k = 65,536$.

They are $[c_8^H \ c_{16}^H]$, $[c_2^H \ c_4^H \ c_8^H \ c_{16}^H]$ and $[c_1^H \ c_2^H \ c_4^H \ c_8^H \ c_{16}^H]$; and their corresponding rates are 0.022, 0.024 and 0.025, respectively. Fig. 19 shows that in terms of BER and FER, all the punctured codes are degraded compared with the unpunctured rate-0.021 PLDPC Hadamard code. Particularly, punctured $[c_8^H \ c_{16}^H]$ has a 0.02 dB performance loss at a BER of 3.6×10^{-5}; punctured $[c_2^H \ c_4^H \ c_8^H \ c_{16}^H]$ has a 0.03 dB performance loss at a BER of 4.7×10^{-5}; and punctured $[c_1^H \ c_2^H \ c_4^H \ c_8^H \ c_{16}^H]$ has a 0.04 dB performance loss at a BER of 1.4×10^{-5}. The BER/FER results indicate that the channel

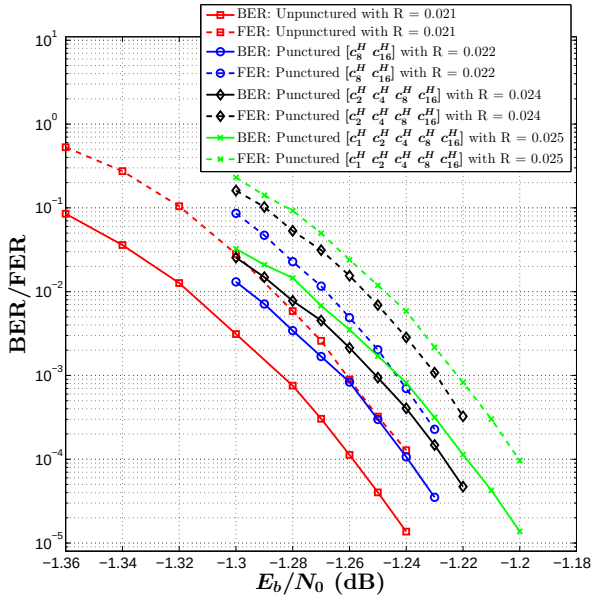


Fig. 19. BER/FER performance of unpunctured/punctured PLDPC-Hadamard codes. Two, four and five D1H-VNs are punctured. $r = 5$ and $k = 65,536$.

observations corresponding to these D1H-VNs provide very useful information for the decoder to decode successfully.

3) $r = 8$ and $r = 10$: For each of the PLDPC-Hadamard codes shown in (21) and (22), we puncture the VN with the largest degree and lowest degree, respectively. Compared with the unpunctured codes, the punctured ones are degraded only very slightly in terms of BER/FER.

V. Conclusion

In this paper, we have proposed an alternate method of designing ultimate-Shannon-limit-approaching LDPC-Hadamard codes —- protograph-based LDPC-Hadamard (PLDPC-Hadamard) codes. By appending degree-1 Hadamard variable nodes (D1H-VNs) to the protograph of LDPC codes, a generalized protograph can be formed to characterize the structure of PLDPC-Hadamard codes. We have also proposed a low-complexity PEXIT algorithm to analyze the threshold of the codes, which is valid for PLDPC-Hadamard protographs with degree-1 variable nodes and/or punctured variable nodes/D1H-VNs. Based on the proposed analysis method, we have found good PLDPC-Hadamard codes with different code rates and have provided the corresponding protomatrices with very low thresholds (< -1.40 dB).

Reliable BER, FER and average number of decoding iterations are derived by running simulations until 100 frame errors are obtained. At a BER of 10^{-5}, the gaps of our codes to the ultimate-Shannon-limit range from 0.40 dB (for rate = 0.0494) to 0.16 dB (for rate = 0.003). Moreover, the error performance of our codes is comparable to that of the traditional LDPC-Hadamard codes. We have also investigated punctured PLDPC-Hadamard codes. When the order of the Hadamard code $r = 4$, puncturing different variable nodes in the protograph produces quite different BER/FER performance degradations compared with the unpunctured code. When

$r = 5$, puncturing one VN can actually improve the BER/FER performance slightly. When $r = 8$ or 10, puncturing one VN does not seem to have any effect. Moreover, we conclude that when $r = 5$, puncturing the extra D1H-VNs provided by the non-systematic Hadamard code degrades the error performance quite significantly.

Finally, we have made use of our proposed analytical technique to find optimal PLDPC-Hadamard code designs. By puncturing these PLDPC-Hadamard codes, punctured codes are obtained and simulated. However, these punctured codes, strictly speaking, are not optimized. In the future, we plan to apply the proposed analytical technique to find optimal PLDPC-Hadamard codes with punctured VNs. We will also investigate annealing approaches or genetic algorithms to speed up the search for optimal protomatrices under some given constraints, and consider spatially-coupled PLDPC-Hadamard codes.

APPENDIX A
TWO OTHER TYPES OF LDPC-HADAMARD CODES

As mentioned in III-A, $d_{c_i} = r+1$ bits from P-VNs need to fulfill the SPC constraint. However, if the inputs to the H-CNs are not required to satisfy the SPC constraint, two other types of codes can be formed.

Fig. 20 shows the first type, in which the information bits (information VNs) are first encoded into an LDPC codeword (with the generation of the parity-check VNs) based on the SPC constraints (SPC-CNs). Subsequently, these VNs (including both information VNs and parity-check VNs) are repeated and interleaved. Then they are used as inputs to the Hadamard check nodes (H-CNs) and to generate the Hadamard parity-check bits (D1H-VNs). Suppose the order of the Hadamard codes used is r and hence there are 2^{r+1} possible Hadamard codewords. As the inputs to the Hadamard check nodes may not satisfy the SPC constraint, the number of inputs would be $r+1$ (instead of $r+2$ in our PLDPC-Hadamard code) and the number of Hadamard parity-check bits (D1H-VNs) generated in each H-CN equals $2^r - (r+1)$ (instead of $2^r - (r+2)$ in our PLDPC-Hadamard code when r is even). Compared with our PLDPC-Hadamard code, the code in Fig. 20 will have a lower code rate when r is even. The decoder structure of the code in Fig. 20 will also be different from ours. The decoder structure of the code in Fig. 20 will consist of a traditional LDPC decoder and a Hadamard decoder, which will iteratively exchange the extrinsic information of the variable nodes (i.e., the VNs shown in the middle layer of Fig. 20). There will be also two interleavers in the code in Fig. 20, as opposed to only one interleaver in our PLDPC-Hadamard code. Thus the decoder is more complicated compared with ours.

Fig. 21 depicts the second type of code in which the SPC constraints are not required. In this case, the information bits (shown as VNs at the top) are repeated and interleaved. Then they are used as inputs to the Hadamard check nodes (H-CNs) and to generate the Hadamard parity-check bits (D1H-VNs). The code can be viewed as a concatenation of repeat codes and Hadamard codes, and the code structure is very different from our PLDPC-Hadamard code.

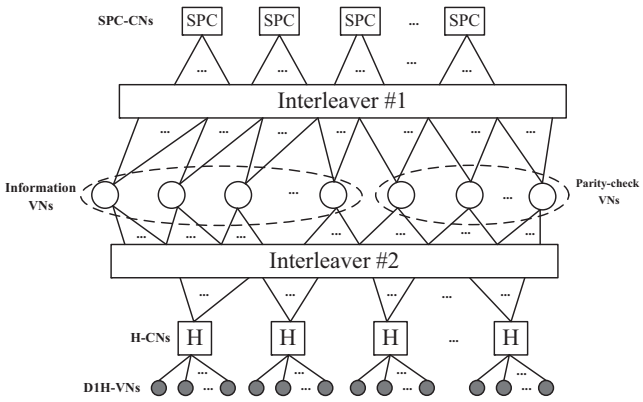


Fig. 20. First type of code in which the inputs to the H-CNs do not need to satisfy the SPC constraint.

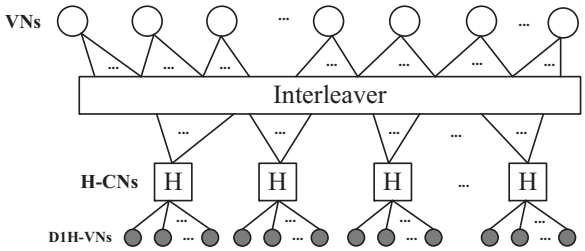


Fig. 21. Second type of code in which the inputs to the H-CNs do not need to satisfy the SPC constraint.

APPENDIX B
MONTE CARLO METHOD FOR FORMING THE $m \times d$ MI MATRIX $\{I_{eh}(i,k)\}$

We define the following symbols:

- $\boldsymbol{\sigma}_\mu = [\sigma_{\mu_0}\ \sigma_{\mu_1}\ \ldots\ \sigma_{\mu_{d-1}}]$: d $(= r+2)$ noise standard deviations;
- $\boldsymbol{c}_\mu = [c_{\mu_0}\ c_{\mu_1}\ \ldots\ c_{\mu_{d-1}}]$: a length-$d$ SPC codeword;
- $\boldsymbol{c}_p = [c_{p_0}\ c_{p_1}\ \ldots\ c_{p_{g-1}}]$: g Hadamard parity bits generated based on the SPC $\boldsymbol{c}_\mu$; $g = 2^r - d$ and $g = 2^r - 2$, respectively, for systematic (r =even) and non-systematic coding (r =odd);
- $\boldsymbol{n}_\mu = [n_{\mu_0}\ n_{\mu_1}\ \ldots\ n_{\mu_{d-1}}]$: d samples following a normal distribution;
- $\boldsymbol{n}_p = [n_{p_0}\ n_{p_1}\ \ldots\ n_{p_{g-1}}]$: g amples following a normal distribution;
- $\boldsymbol{L}_\mu = [L_{\mu_0}\ L_{\mu_1}\ \ldots\ L_{\mu_{d-1}}]$: d LLR values corresponding to the SPC codeword $\boldsymbol{c}_\mu$;
- $\boldsymbol{L}_p = [L_{p_0}\ L_{p_1}\ \ldots\ L_{p_{g-1}}]$: g channel LLR values corresponding to the Hadamard parity bits $\boldsymbol{c}_p$;
- $\boldsymbol{L}_e = [L_{e_0}\ L_{e_1}\ \ldots\ L_{e_{d-1}}]$: d extrinsic LLR values generated by the Hadamard decoder;
- $\boldsymbol{U}$: a $w \times d$ matrix in which each row represents a length-d SPC codeword; and the k-th column ($k = 0, 1, \ldots, d-1$) corresponds to the k-th bit (c_{μ_k}) of the SPC codeword;
- $\boldsymbol{V}$: a $w \times d$ matrix in which each row represents a set of (d) extrinsic LLR values generated by the Hadamard decoder; and the k-th column ($k = 0, 1, \ldots, d-1$) corresponds to the extrinsic LLR value for the k-th bit (c_{μ_k}) of the SPC codeword;

- $\boldsymbol{p}_{e_0} = [p_e(\xi|c_{\mu_0} = \text{“0”}) \quad p_e(\xi|c_{\mu_1} = \text{“0”}) \quad \cdots \quad p_e(\xi|c_{\mu_{d-1}} = \text{“0”})]$: PDFs for $c_{\mu_k} = \text{“0”}$ ($k = 0, 1, \ldots, d-1$);
- $\boldsymbol{p}_{e_1} = [p_e(\xi|c_{\mu_0} = \text{“1”}) \quad p_e(\xi|c_{\mu_1} = \text{“1”}) \quad \cdots \quad p_e(\xi|c_{\mu_{d-1}} = \text{“1”})]$: PDFs for $c_{\mu_k} = \text{“1”}$ ($k = 0, 1, \ldots, d-1$).

The $m \times d$ MI matrix $\{I_{eh}(i,k)\}$ is updated with following steps.

i) Given the standard deviation $\sigma_{L_{ch}}$.
ii) Set $i = 0$.
iii) For the i-th row in the MI matrix $\{I_{ah}(i,k)\}$, use the J-function in [13] to compute the standard deviation $\sigma_{\mu_k} = J^{-1}(I_{ah}(i,k))$ for $k = 0, 1, \ldots, d-1$.
iv) Set $j = 0$.
v) Randomly generate a length-d SPC codeword $\boldsymbol{c}_\mu$; further encode $\boldsymbol{c}_\mu$ into a Hadamard codeword using systematic (when $r = d-2$ is even) or non-systematic (when r is odd) coding and generate the g Hadamard parity bits $\boldsymbol{c}_p$.
vi) Randomly generate a sample vector $\boldsymbol{n}_\mu$ where each n_{μ_k} ($k = 0, 1, \ldots, d-1$) follows a different normal distribution $\mathcal{N}(\sigma_{\mu_k}^2/2, \sigma_{\mu_k}^2)$.
vii) Randomly generate a sample vector $\boldsymbol{n}_p$ where all $n_{p_{k'}}$'s ($k' = 0, 1, \ldots, g-1$) follow the same normal distribution $\mathcal{N}(\sigma_{L_{ch}}^2/2, \sigma_{L_{ch}}^2)$.
viii) For $k = 0, 1, \ldots, d-1$, set $L_{\mu_k} = +n_{\mu_k}$ if $c_{\mu_k} = \text{“0”}$; otherwise set $L_{\mu_k} = -n_{\mu_k}$ if $c_{\mu_k} = \text{“1”}$.
ix) For $k' = 0, 1, \ldots, g-1$, set $L_{p_{k'}} = +n_{p_{k'}}$ if $c_{p_{k'}} = \text{“0”}$; otherwise set $L_{p_{k'}} = -n_{p_{k'}}$ if $c_{p_{k'}} = \text{“1”}$.
x) Input $\boldsymbol{L}_\mu$ and $\boldsymbol{L}_p$, respectively, as the *a priori* and channel LLRs to the Hadamard decoder. Use the decoding algorithm described in Sect. III-B to compute the d output extrinsic LLR values $\boldsymbol{L}_e$.
xi) Assign $\boldsymbol{c}_\mu$ to the j-th row of $\boldsymbol{U}$ and assign $\boldsymbol{L}_e$ to the j-th row of $\boldsymbol{V}$.
xii) Set $j = j + 1$. If $j < w$, go to Step v). (We set $w = 10,000$.)
xiii) The k-th columns ($k = 0, 1, \ldots, d-1$) of both $\boldsymbol{U}$ and $\boldsymbol{V}$ correspond to bit c_{μ_k}. Obtain the PDFs $p_e(\xi|c_{\mu_k} = \text{“0”})$ and $p_e(\xi|c_{\mu_k} = \text{“1”})$ ($k = 0, 1, \ldots, d-1$) based on $\boldsymbol{U}$ and $\boldsymbol{V}$.
xiv) Use $p_e(\xi|c_{\mu_k} = \text{“0”})$ and $p_e(\xi|c_{\mu_k} = \text{“1”})$ to compute (16) and hence $I_{eh}(i,k)$ ($k = 0, 1, \ldots, d-1$).
xv) Set $i = i + 1$. If $i < m$, go to step iii).

REFERENCES

[1] P. W. Zhang, F. C. M. Lau, and C.-W. Sham, “Protograph-based LDPC-Hadamard Codes,” in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2020.

[2] C. E. Shannon, “A Mathematical Theory of Communication,” *Bell Syst. Tech. J*, vol. 27, pp. 379–423, Jul. 1948.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes,” *in Proc. IEEE ICC*, vol. 2, pp. 1064–1070, May 1993.

[4] C. Berrou and A. Glavieux, “Near optimum error correcting coding and decoding: turbo-codes,” *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261–1271, 1996.

[5] R. G. Gallager, *Low-Density Parity-Check Codes*. PhD thesis, Cambridge, MA, USA, 1963.

[6] D. J. MacKay and R. M. Neal, “Good Codes Based on Very Sparse Matrices,” in *Proc. Cryptography Coding. 5th IMA Conf., Number 1025 Lecture Notes Comput. Sci.*, pp. 100–111, Oct. 1995.

[7] E. Arikan, “Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels,” *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[8] S. Shao, P. Hailes, T. Wang, J. Wu, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, “Survey of Turbo, LDPC, and Polar Decoder ASIC Implementations,” *IEEE Commun. Surv. Tut.*, vol. 21, no. 3, pp. 2309–2333, 2019.

[9] Y. Fang, G. A. Bi, Y. L. Guan, and F. C. M. Lau, “A Survey on Protograph LDPC Codes and Their Applications,” *IEEE Commun. Surv. Tut.*, vol. 17, no. 4, pp. 1989–2016, 2015.

[10] R. M. Tanner, “A Recursive Approach to Low Complexity Codes,” *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.

[11] 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.

[12] T. J. Richardson and R. L. Urbanke, “Capacity of Low-Density Parity-Check Codes under Message Passing Decoding,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[13] S. T. Brink, “Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes,” *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[14] 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.

[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] M. 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.

[17] D. Divsalar, H. Jin, and R. McEliece, “Coding Theorems for Turbo-Like Codes,” in *Proc. Allerton Conf.*, pp. 201–210, 1998.

[18] H. Jin, A. Khandekar, and R. McEliece, “Irregular Repeat-Accumulate Codes,” in *Proc. 2nd Int. Symp. Turbo Codes*, pp. 1–8, 2000.

[19] J. Thorpe, “Low-Density Parity-Check (LDPC) Codes Constructed From Protographs,” in *Proc. IPN Progr. Rep.*, pp. 1–7, Aug. 2003.

[20] G. Liva and M. Chiani, “Protograph LDPC Codes Design Based on EXIT Analysis,” *IEEE GLOBECOM 2007*, pp. 3250–3254, 2007.

[21] D. Divsalar, S. Dolinar, and C. Jones, “Low-rate LDPC codes with simple protograph structure,” in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, pp. 1622–1626, 2005.

[22] M. Stinner and P. M. Olmos, “Finite-length performance of multi-edge protograph-based spatially coupled LDPC codes,” in *2015 IEEE International Symposium on Information Theory (ISIT)*, pp. 889–893, 2015.

[23] M. Stinner and P. M. Olmos, “On the Waterfall Performance of Finite-Length SC-LDPC Codes Constructed From Protographs,” *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 345–361, 2016.

[24] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, “Spatially coupled LDPC codes constructed from protographs,” *IEEE Transactions on Information Theory*, vol. 61, no. 9, pp. 4866–4889, 2015.

[25] Y. Fang, P. Chen, G. Cai, F. C. M. Lau, S. C. Liew, and G. Han, “Outage-Limit-Approaching Channel Coding for Future Wireless Communications: Root-Protograph Low-Density Parity-Check Codes,” *IEEE Vehicular Technology Magazine*, vol. 14, no. 2, pp. 85–93, 2019.

[26] L. Lentmaier and K. S. Zigangirov, “On Generalized Low-Density Parity-Check Codes Based on Hamming Component Codes,” *IEEE Commun. Lett.*, vol. 3, no. 8, pp. 248–250, Aug. 1999.

[27] J. Boutros, O. Pothier, and G. Zemor, “Generalized Low-Density (Tanner) Codes,” in *Proc. IEEE Int. Conf. Commun.*, vol. 1, pp. 441–445, Jun. 1999.

[28] Y. Min, F. C. M. Lau, and C. K. Tse, “Generalized LDPC code with Single-Parity-Check Product Constraints at Super Check Nodes,” in *2012 ISTC*, pp. 165–169, 2012.

[29] S. Abu-Surra, G. Liva, and W. E. Ryan, “Low-Floor Tanner Codes via Hamming-Node or RSCC-Node Doping,” in *Proc. Int. Symp. Appl. Algebra, Algebraic Algorithms Error-Correcting Codes*, pp. 245–254, Feb. 2006.

[30] G. Liva, W. E. Ryan, and M. Chiani, “Quasi-Cyclic Generalized LDPC Codes with Low Error Floors,” *IEEE Trans. Commun.*, vol. 56, no. 1, pp. 49–57, Jan. 2008.

[31] S. Abu-Surra, D. Divsalar, and W. Ryan, “Enumerators for Protograph-Based Ensembles of LDPC and Generalized LDPC Codes,” *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 856–886, Feb. 2011.

[32] E. Sharon, A. Ashikhmin, and S. Litsyn, "EXIT functions for binary input memoryless symmetric channels," *IEEE Transactions on Communications*, vol. 54, no. 7, pp. 1207–1214, 2006.

[33] E. Sharon, A. Ashikhmin, and S. Litsyn, "Analysis of low-density parity-check codes based on EXIT functions," *IEEE Transactions on Communications*, vol. 54, no. 8, pp. 1407–1414, 2006.

[34] E. Sharon, R. Zamir, and D. Avraham, "Generalized Low Density Parity Check Codes," in *10th Annual Non-Volatile Memories Workshop*, Mar. 2019.

[35] G. Yue, L. Ping, and X. Wang, "Low-rate generalized low-density parity-check codes with Hadamard constraints," in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, pp. 1377–1381, 2005.

[36] G. Yue, L. Ping, and X. Wang, "Generalized Low-Density Parity-Check Codes Based on Hadamard Constraints," *IEEE Transactions on Information Theory*, vol. 53, no. 3, pp. 1058–1079, 2007.

[37] R. M. Buehrer, *Code Division Multiple Access (CDMA)*. 2006.

[38] L. Ping, L. Liu, Keying Wu, and W. K. Leung, "Interleave division multiple-access," *IEEE Transactions on Wireless Communications*, vol. 5, no. 4, pp. 938–947, 2006.

[39] 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.

[40] H. H. Xu and Z. S. Bie, "CRC-aided Iterative Decoding of Turbo-Hadamard Codes," in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, pp. 1–5, 2018.

[41] S. Jiang, P. W. Zhang, F. C. M. Lau, C.-W. Sham, and K. Huang, "A Turbo-Hadamard Encoder/Decoder System with Hundreds of Mbps Throughput," in *2018 ISTC*, pp. 1–5, 2018.

[42] 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. on Circuits and Systems II: Express Briefs*, vol. 67, no. 10, pp. 2169–2173, 2020.

[43] W. K. R. Leung, G. Yue, L. Ping, and X. Wang, "Concatenated zigzag Hadamard codes," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1711–1723, 2006.

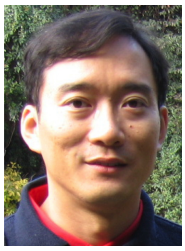
[44] S. Jiang, F. C. M. Lau, and C.-W. Sham, "Hardware Design of Concatenated Zigzag Hadamard Encoder/Decoder System With High Throughput," *IEEE Access*, vol. 8, pp. 165298–165306, 2020.

[45] Y. Liu, P. M. Olmos, and D. G. M. Mitchell, "On Generalized LDPC Codes for 5G Ultra Reliable Communication," in *2018 IEEE Information Theory Workshop (ITW)*, pp. 1–5, 2018.

[46] K. Li, X. Wang, and A. Ashikhmin, "EXIT Functions of Hadamard Components in Repeat-Zigzag-Hadamard (RZH) Codes with Parallel Decoding," *IEEE Trans. Inf. Theory*, vol. 54, no. 4, pp. 1773–1785, Apr. 2008.

[47] P. W. Zhang, F. C. M. Lau, and C.-W. Sham, "Protograph-Based Low-Density Parity-Check Hadamard Codes," *arXiv:2010.08285*, 2021.

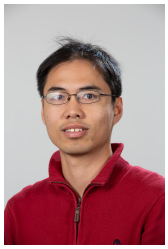
Peng-Wei Zhang received the Bachelor of Engineering degree in Electronics and Information Engineering and the Master of Engineering degree in Electronics and Communication Engineering from Chongqing University of Posts and Telecommunications, China, in 2013 and 2016, respectively. He is currently pursuing his PhD degree at the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong.



Francis C. M. Lau received the BEng(Hons) degree in electrical and electronic engineering and the PhD degree from King's College London, University of London, UK. He is a Professor at the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong. He is also a Fellow of IEEE and a Fellow of IET.

He is a co-author of *Chaos-Based Digital Communication Systems* and *Digital Communications with Chaos: Multiple Access Techniques and Performance Evaluation*. He is also a co-holder of five US patents and one pending US patent. He has published more than 320 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. He is a co-recipient of one Natural Science Award from the Guangdong Provincial Government, China; eight best/outstanding conference paper awards; one technology transfer award; two young scientist awards from International Union of Radio Science; and one FPGA design competition award.

He was the General Co-chair of International Symposium on Turbo Codes & Iterative Information Processing (2018) and the Chair of Technical Committee on Nonlinear Circuits and Systems, IEEE Circuits and Systems Society (2012-13). He served as an associate editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II (2004-2005 and 2015-2019), IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I (2006-2007), and IEEE CIRCUITS AND SYSTEMS MAGAZINE (2012-2015). He has been a guest associate editor of INTERNATIONAL JOURNAL AND BIFURCATION AND CHAOS since 2010.



Chiu-Wing Sham received his Bachelor degree (Computer Engineering) and MPhil. degree from The Chinese University of Hong Kong in 2000 and 2002 respectively, and received his Ph.D. degree from the same university in 2006. He has worked as an Electronic Engineer on the FPGA applications of the motion-control system and system security with cryptography in ASM Pacific Technology Ltd (HK). 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 now working at The University of Auckland as a Senior Lecturer. He is also an IEEE Senior Member and an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II (2017-present).