Channel Coding for Optical Transmission Systems

Ivan B. Djordjevic

University of Arizona, Department of Electrical and Computer Engineering, Tucson, USA; E-mail: ivan@email.arizona.edu

Abstract— In this invited paper, both binary and nonbinary LDPC codes suitable for optical transmission systems are described. The corresponding FPGA implementation has been discussed as well. The use of adaptive LDPC coding to deal with time-varying optical channel conditions is described as well.

I. INTRODUCTION

The forward error correction (FEC) based on low-density parity-check coding, represents one of the key enabling technologies for the next generation of optical transmission systems [1]-[4]. In this invited paper, we describe both binary and nonbinary LDPC codes suitable for use in optical communications. Several attractive classes of LDPC codes, recently proposed for optical communications including staircase, spatially-coupled-like, and convolutional LDPC codes are related here to the irregular quasi-cyclic (QC) LDPC code design derived from the pairwise block designs (PBDs).

Then we discuss the LDPC decoding algorithms, with spatial attention being devoted to iterative decoding algorithms suitable for FPGA implementation. To deal with time-varying optical channel conditions several strategies for code rate adaptation have been discussed such as (1) partial reconfiguration of the parity-check matrix of a template regular QC-LDPC code, (2) puncturing, (3) shortening, and (4) the re-encoding strategy.

We also describe a rate adaptive LDPC coding scheme we introduced in [2], exhibiting the following unique features: (i) unified hardware architecture has been used for different code rates, (ii) it offers flexibility in rate adjustment, and (iii) it has low implementation complexity.

II. RATE ADAPTIVE CODING ENABLED BY IRREGULAR QC-LDPC CODING DERIVED FROM PBD

In this section, we briefly describe the nonbinary (NB) irregular QC-LDPC code design derived from PBDs, introduced in [5] (see also [1]), defined over $\text{GF}(q)$. The binary LDPC code is just a special case of this code design for $q=2$. We first introduce the concept of PBD. The starting point in PBD definition is the set V of size v , representing a set of elements (points), with any subset being called a block. The pairwise balanced design $\text{PBD}(v, K, \delta)$ is defined as a collection of blocks of different sizes taken from set K on such a way that every pair of points is contained in δ of the blocks. The parity-check matrix of irregular NB QC-LDPC codes based on PBDs is given by Eqn. 1, introduced in [5]. In Eqn. (1) I denotes $B \times B$ identity matrix (with B being a prime number), P is the permutation matrix of the same size

defined as $P=(p_{ij})_{B \times B}$, $p_{i,i+1}=p_{B,1}=1$ (zero otherwise); and r and c represent the number of block-rows and block-columns, respectively. Further, we use $\{b_{ij}\}$ to denote the points of the i -th block in $\text{PBD}(r, K, \delta)$, with the largest size of block k in set of sizes K satisfying the inequality $k \leq r$. α^i are nonzero elements of Galois field of size q , denoted as $\text{GF}(q)$. Finally, the $\mathcal{A}(b_{ij})$ denotes the indicator function, which takes the value 1 for the existing point within the i -th block, and 0 for the non-existing point. To summarize, only those submatrices for which indicator function has value 1 will be preserved from template (mother), regular, QC-LDPC code design. Given the fact that PBDs have regular mathematical structure that can be algebraically described, the irregular NB QC-LDPC codes derived from PBDs have the complexity comparable or lower to that of regular NB QC-LDPC code design.

$$H = \begin{bmatrix} \alpha^0 \mathcal{A}(b_{00})I & \alpha^1 \mathcal{A}(b_{10})I & \cdots & \alpha^{c-1} \mathcal{A}(b_{c-1,0})I \\ \alpha^{c-1} \mathcal{A}(b_{01})I & \alpha^0 \mathcal{A}(b_{11})P^{s[1]} & \cdots & \alpha^{c-2} \mathcal{A}(b_{c-1,1})P^{s[c-1]} \\ \alpha^{c-2} \mathcal{A}(b_{02})I & \alpha^{c-1} \mathcal{A}(b_{12})P^{2s[1]} & \cdots & \alpha^{c-3} \mathcal{A}(b_{c-1,2})P^{2s[c-1]} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{c-(r-1)} \mathcal{A}(b_{0,r-1})I & \alpha^{c-r+2} \mathcal{A}(b_{1,r-1})P^{(r-1)s[1]} & \cdots & \alpha^{c-r} \mathcal{A}(b_{c-1,r-1})P^{(r-1)s[c-1]} \end{bmatrix} \quad (1)$$

By employing different PBDs, different classes of codes can be derived for this template code including staircase, spatially-coupled-like, and convolutional LDPC codes, as discussed in [1],[5].

In the remainder of this section, we discuss few strategies for code rate adaptation. The code rate adaptation can be performed by *partial reconfiguration* of decoder based on Eqn. (1), for a fixed codeword length, by changing the size of the permutation matrix P while keeping the number of block rows constant, and/or by varying the number of employed block-rows while keeping P fixed. It is also possible to perform the puncturing of parity symbols in original code (N, K, d) to obtain a linear block code $(N-p, K, d_p)$, $d_p \leq d$, where p is the number of removed parity symbols. It is also possible to apply shortening, by removing the certain number of block-columns from Eqn. (1). Unfortunately, when either shortening or puncturing is applied to LDPC code design, an early error floor phenomenon can be introduced. This is the reason why the re-encoding approach has been introduced in [1] (and references therein), in which the adaptive LDPC encoder encapsulates the last L symbols of the proceeding codeword and the incoming $K-L$ information symbols into a K -symbol vector. The actual code rate is $R'=(K-L)/(N-L)$, where $0 < L < K$ and can be tuned easily in the range $(0, R]$ ($R=K/N$ is the code rate of template code) by adjusting the re-encoded data size L .

III. DECODING OF LDPC CODES

Given the space limitations, decoding of binary LDPC codes is discussed only here. A binary LDPC code is defined as an (N, K) linear block code (LBC) whose parity-check matrix $\mathbf{H}$ has a low density of ones. In particular, a *regular LDPC code*, suitable for hardware implementations, is an LBC whose $\mathbf{H}$ matrix contains exactly W_c 1's in each column and exactly $W_r = W_c N / (N - K)$ 1's in each row, where $W_c \ll N - K$. The graphical representation of LDPC codes, known as either Tanner or bipartite graph representation, is helpful in efficient description of LDPC decoding algorithms, and it is defined as a graph whose nodes may be separated into two classes (*variable* and *check* nodes), and where *undirected edges* may only connect two nodes not residing in the same class. The Tanner graph of a code is drawn according to the following rule: check (function) node c is connected to variable (bit) node v whenever element h_{cv} in a parity-check matrix $\mathbf{H}$ is a 1. In an $m \times N$ parity-check matrix, there are $m = N - K$ check nodes and N variable nodes.

Now we can summarize the *log-domain sum-product algorithm (SPA)* as follows [1].

- 0) *Initialization*: For $v=0, 1, \dots, N-1$; initialize the messages L_{vc} to be sent from v -node v to c -node c to channel LLRs $L_{ch}(v)$, namely $L_{vc} = L_{ch}(v)$.

- 1) *c-node update rule*: For $c=0, 1, \dots, N-K-1$; compute $L_{cv} = \boxed{+}_{\mathcal{N}(c) \setminus \{v\}} L_{vc}$. The box-plus operator is defined by

$$L_1 \boxed{+} L_2 = \prod_{k=1}^2 \text{sign}(L_k) \cdot \phi \left(\sum_{k=1}^2 \phi(|L_k|) \right),$$

where $\phi(x) = -\log \tanh(x/2)$. The box operator for $|\mathcal{N}(c) \setminus \{v\}|$ components is obtained by recursively applying 2-component version defined above. [$\mathcal{N}(c)$ denotes the neighborhood of check-node c .]

- 2) *v-node update rule*: For $v=0, 1, \dots, N-1$; set $L_{vc} = L_{ch}(v) + \sum_{\mathcal{N}(v) \setminus \{c\}} L_{cv}$ for all c -nodes for which $h_{cv}=1$.
- 3) *Bit decisions*: Update $L(v)$ ($v=0, \dots, N-1$) by $L(v) = L_{ch}(v) + \sum_{\mathcal{N}(v)} L_{cv}$ and set $\hat{v}=1$ when $L(v) < 0$

(otherwise, $\hat{v}=0$). If $\hat{\mathbf{v}}\mathbf{H}^T = \mathbf{0}$ or pre-determined number of iterations has been reached then stop, otherwise go to step 1).

Because the c -node update rule involves log and tanh functions, it is computationally intensive, and there exist many approximations. The very popular is the min-sum-plus-correction-term approximation, in which the “box-plus” operator $\boxed{+}$ has been expressed as

$$L_1 \boxed{+} L_2 = \prod_{k=1}^2 \text{sign}(L_k) \cdot \min(|L_1|, |L_2|) + c(x, y), \quad (2)$$

where $c(x, y)$ denotes the correction factor defined by

$$c(x, y) = \log(1 + e^{-|x+y|}) - \log(1 + e^{-|x-y|}), \quad (3)$$

commonly implemented as a look-up table (LUT). Given the fact that $|c(x, y)| < 0.693$, very often this term can be ignored. Another popular decoding algorithm is min-sum

algorithm in which we simply ignore the correction term in (2).

IV. RATE ADAPTIVE FPGA DECODER ARCHITECTURE

In our recent paper [2], we proposed a rate adaptive LDPC code design, in which code design consisted of two steps: (1) a large-girth QC-LDPC code is designed based on Eqn. (1); (2) the parity-check matrix is modified either by puncturing several rightmost columns or doping selected single-parity check (SPC) codes with simple linear block codes. For coarse-tuning we shorten the code by removing several block-columns from the parity-check matrix, while for fine tuning we replace selected number SPCs with low-complexity LBCs.

For SPC nodes, the check-node update rule based on Eqn. (2) is used. For generalized parity-checks, derived from LBCs, the maximum *a posteriori* probability (MAP) decoding based on BCJR algorithm is used. The corresponding FPGA decoding architecture is summarized in Fig. 1, where Fig. 1(left) represents an overview of the reconfigurable binary LDPC decoder, which is composed memories and processors. The processors can be classified into following categories: (i) variable-node processor (VNP) corresponding to step 2); (ii) scaled min-sum check-node processor (SMS-CNP) corresponding to scaled version of Eqn. (2), (iii) MAP based CNP, referred here as generalized-CNP, employing BCJR algorithm on trellis of corresponding LBC, and (iv) early termination unit (ETU). Four sets of block RAMs are used in the implementation of rate adaptive LDPC decoder: (i) block RAM that stores the channel LLRs, (ii) block RAM that stores the messages from check-nodes to variable-nodes, (iii) block RAM storing the decoded bits, and (iv) additional block RAM inside each CNP storing the intermediate values.

Fig. 1 Rate-adaptive LDPC decoder architecture proposed in [2]: (left) overall architecture, (middle) architecture of regular CNP, and (right) architecture of BCJR-based generalized CNP.

Additional details of this rate adaptive FPGA decoding architecture as well as the emulation results will be presented at the conference.

REFERENCES

- [1] I. B. Djordjevic, *IEEE Communications Surveys and Tutorials*, vol. 18, no. 3, pp. 1920-1951, third quarter 2016, August 19, 2016
- [2] D. Zou, I. B. Djordjevic, *IEEE Photonics Journal*, vol. 8, no. 5, pp. 7906008-1–7906008-8, Oct. 2016.
- [3] D. Zou, I. B. Djordjevic, *Optics Express*, vol. 24, no. 18, pp. 21159-21166, 2016.
- [4] D. Zou, I. B. Djordjevic, *Optics Express*, vol. 23, no. 10, pp. 14501-14509, 18 May 2015.
- [5] I. B. Djordjevic, *J. Lightw. Technol.*, vol. 31, no. 16, pp. 2969-2975, Aug. 15, 2013.