This is the Pre-Published Version.

Generalized LDPC Code With Single-Parity-Check Product Constraints At Super Check Nodes

Y. Min, F. C. M. Lau and C. K. Tse
Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Hong Kong
Email: yue.min@connect.polyu.hk, encmlau@polyu.edu.hk, encktse@polyu.edu.hk

***Abstract*—Due to the more complex constraints, generalized low-density parity-check (GLDPC) codes can achieve better error performance but require much higher decoding complexities compared with the standard LDPC codes. In this paper, single-parity-check product-codes (SPC-PCs) are considered as the constituent codes in the super check nodes of a GLDPC code. Moreover, turbo iterations are used in decoding the SPC-PCs. The error performance and the decoder complexity of the proposed GLDPC code are compared with other LDPC code and GLDPC code.**

***Index Terms*—Generalized low-density parity-check code, single-parity-check product-code, turbo iteration**

I. Introduction

A standard low-density parity-check (LDPC) code can be represented by a bipartite graph consisting of variable nodes, check codes, and edges connecting the variable nodes and check nodes. Each variable node can be regarded as a repetition code while each check node can be viewed as a single-parity-check (SPC) code. In [1], the standard LDPC code has been generalized by replacing the repetition codes and the SPC codes with more complex linear block codes called constituent (or component) codes or subcodes.

Taking advantage of the more powerful constitute codes, generalized LDPC (GLDPC) codes[1] can achieve better bit-error-rate (BER) performance, lower error floor and faster convergence rate compared to its standard LDPC counterparts. At the same time, the computation complexity becomes much higher because more complicated constraints are introduced. Thus, finding proper constituent codes that can achieve a given error performance at a tolerable computation complexity is a key challenge in designing good GLDPC codes.

In [2], the decoding method applied to decode each constitute code in a GLDPC code involves (i) finding all possible sequences that can satisfy the constitute code, (ii) determining the probability of each sequence, and (iii) adding the probabilities together. Besides, the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [3], a trellis-based a posteriori probability (APP) decoding algorithm, is widely considered to decode GLDPC codes [4]. Both algorithms can decode GLDPC codes with all kinds of component codes, such as GLDPC code with Hamming component codes [5] [6]; GLDPC code with BCH component codes [6] [7]; GLDPC code with RS component codes [7]; and GLDPC code with hybrid component codes [8] [9]. However, the complexities of such decoding algorithms are very high.

In [10], a class of GLDPC codes with tailored shortened Hamming codes as constituent codes, named TSHC-GLDPC codes, has been proposed. Such codes can be decoded using the fast-Fourier-transform(FFT)-based APP algorithm. Although the computation complexity is reduced compared with trellis-based APP algorithm, the complexity remains high. In [11], a class of GLDPC codes with Hadamard constraints are proposed. Since Hadamard constituent codes can be fast decoded based on fast Hadamard transform (FHT), the complexity issue can be resolved. Yet, the drawback of such codes is an extremely low code rate ($R < 0.1$), making them only suitable for certain kinds of communication systems.

In this paper, we propose a family of GLDPC codes with single-parity-check product-codes (SPC-PCs) as component codes. In Section II, we review the structures of GLDPC codes and SPC-PCs. We then show our proposed class of GLDPC codes in Section III. We also describe the iterative decoding algorithm in the same section. Finally, the error performance and the decoding complexity of the proposed GLDPC codes are presented and compared with other channel codes in Section V.

II. Review

A. Generalized LDPC Code

A GLDPC code is denoted by (N, K) where N and K represent the number of code bits and the number of information bits, respectively, in each codeword. Moreover, the variable nodes and the check nodes in a GLDPC code are termed as super variable nodes (SVNs) and super check nodes (SCNs), respectively.

Figure 1 illustrates the bipartite graph of a GLDPC code having N_a SVNs and M_a SCNs. The connections among the super nodes correspond to a $M_a \times N_a$ matrix, which is called an *adjacency matrix* and is denoted by $\mathbf{H}_a$. A non-zero (m, n)-th entry in $\mathbf{H}_a$ indicates a connection between the m-th SCN and the n-th SVN. Moreover, the weight of the n-th column in $\mathbf{H}_a$ represents the number of edges that connect the n-th

The work described in this paper was partially supported by a grant from the RGC of the Hong Kong SAR, China (Project No. PolyU 5190/11E).

[1]Strictly speaking, GLDPC codes refer to the LDPC codes in which the SPC codes (check nodes) have been replaced with more complex linear block codes (super check nodes). If both the repetition codes (variable node) and the SPC codes in the LDPC codes are replaced with more complex linear block codes, the resultant codes are called doubly-GLDPC codes. However, depending on the context, doubly-GLDPC codes may also referred to as GLDPC codes for simplicity. In this paper, all the GLDPC codes are indeed doubly-GLDPC codes unless otherwise stated.

© 2012 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

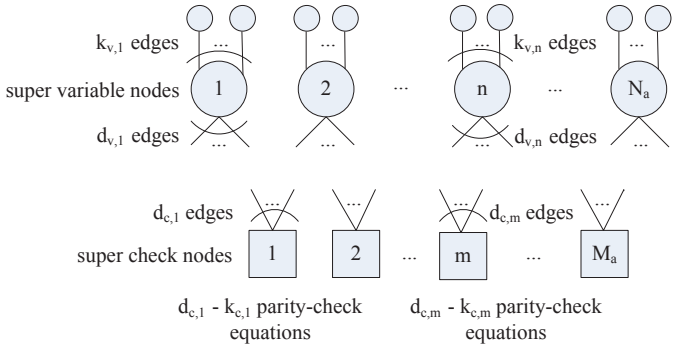

Fig. 1. The bipartite graph of a GLDPC code.

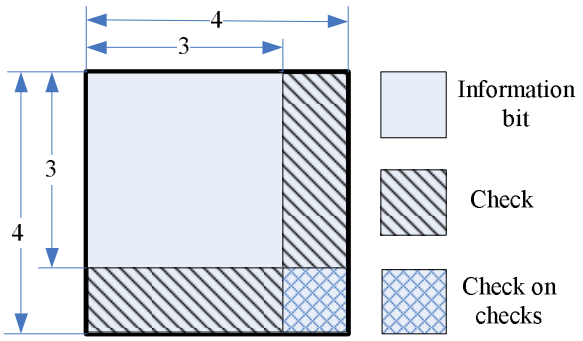

Fig. 2. The structure of the $(4,3)^2$ single-parity-check product-code (SPC-PC).

SVN to the SCNs. Similarly, the weight of the m-th row in $\mathbf{H}_a$ indicates the number of edges that connect the m-th SCN to the SVNs.

We denote the weight of the n-th column and the weight of the m-th row in the adjacency matrix by $d_{v,n}$ and $d_{c,m}$, respectively. We also denote the number of external connections of the n-th SVN by $k_{v,n}$, as shown in Fig. 1. The overall length of the GLDPC code hence equals

$$N = \sum_{n=1}^{N_a} k_{v,n}. \tag{1}$$

Then, for the n-th SVN($1 \le n \le N_a$), we can represent the component code by $(d_{v,n}, k_{v,n})$, meaning that $d_{v,n}$ coded bits are formed for every $k_{v,n}$ information bits. Similarly, for the m-th SCN($1 \le m \le M_a$), we denote the number of check equations by $d_{c,m} - k_{c,m}$. The total number of check bits M in the GLDPC code is hence upper-bounded by

$$M \le \sum_{m=1}^{M_a} (d_{c,m} - k_{c,m}). \tag{2}$$

We can then represent the constituent code for the m-th SCN by $(d_{c,m}, k_{c,m})$, implying that $d_{c,m}$ coded bits are formed for every $k_{c,m}$ information bits. Using (1) and (2), we can compute the overall rate of the GLDPC code as

$$R \ge 1 - \frac{\sum_{m=1}^{M_a} (d_{c,m} - k_{c,m})}{\sum_{n=1}^{N_a} k_{v,n}}. \tag{3}$$

B. Single-Parity-Check Product-Code (SPC-PC)

In the $(N_{spc}, N_{spc}-1)^D$ SPC-PC, the information bits are first arranged in a D-dimensional hypercube and then encoded by a $(N_{spc}, N_{spc}-1)$ SPC in each dimension [12]. Figure 2 shows the $(4,3)^2$ SPC-PC. The construction is elaborated as follows. First, 9 information bits are arranged in a square (of 2-dimension) of size 3×3. The bits are then encoded by the $(4,3)$ SPC code row-by-row and column-by-column. Afterward, a check on checks satisfying the $(4,3)$ SPC constraint is added. In this example, the number of information bits is 9; the number of check bits is 7; the code length is 16 and hence the code rate equals $9/16$. In general, the number of information bits equals $K_{spc-pc} = (N_{spc}-1)^D$; the number of check bits is $M_{spc-pc} = N_{spc}^D - (N_{spc}-1)^D$; the code length is $N_{spc-pc} = N_{spc}^D$ and hence the code rate equals $R_{spc-pc} = (N_{spc}-1)^D / N_{spc}^D$. Furthermore, the parity-check matrix of the $(N_{spc}, N_{spc}-1)^D$ SPC-PC, denoted by $\mathbf{H}_{spc-pc}$ is of size $M_{spc-pc} \times N_{spc}^D$.

In [13], it has been proved that GLDPC codes with constituent codes having a minimum distance $d_{min} \ge 3$ performs asymptotically good. The minimum distance of the $(N_{spc}, N_{spc}-1)^D$ SPC-PC, moreover, has been shown equal to $d_{min} = 2^D$ [12]. Such an outstanding distance property therefore makes SPC-PC an attractive candidate to be constituent codes of a GLDPC code.

III. Proposed GLDPC Code

A. Code Construction

For simplicity, we only consider regular GLDPC codes. We use the $(N_{spc}, N_{spc}-1)^D$ SPC-PC as the component code in each of the SCNs. The corresponding parity-check matrix $\mathbf{H}_{spc-pc}$ is therefore of size $M_{spc-pc} \times N_{spc}^D$. It can also be readily shown that each SCN has a degree of N_{spc}^D. Moreover, to allow a high overall code rate of the GLDPC code without increasing the decoding complexity much, SPC codes are employed as component codes in all SVNs. Denoting the number of information bits entering each SVN by $d_v - 1$, the generator matrix $\mathbf{G}_{svn}$ corresponding to the SPC code is of size $(d_v - 1) \times d_v$.

We consider an adjacency matrix $\mathbf{H}_a$ of size $M_a \times N_a$. (Note that the row weight of $\mathbf{H}_a$ must equal N_{spc}^D while the column weight equals d_v.) Based on the matrices $\mathbf{H}_{spc-pc}$ and $\mathbf{G}_{svn}$, the parity-check matrix of the GLDPC code is constructed as follows.

- Step 1 (Row Expansion): For each row in $\mathbf{H}_a$, each "1" is substituted by one column vector in $\mathbf{H}_{spc-pc}$ while each "0" is substituted by an all-zero column vector. However, each column vector in $\mathbf{H}_{spc-pc}$ can only be used once for each row in $\mathbf{H}_a$. After the substitutions are made, the resultant matrix is of size $M_a M_{spc-pc} \times N_a$. We denote this matrix by $\mathbf{H}'_a$.
- Step 2 (Column Expansion): Consider a column in $\mathbf{H}'_a$. For every M_{spc-pc} entries (called a row block), all "1"s within this block are substituted by the same row

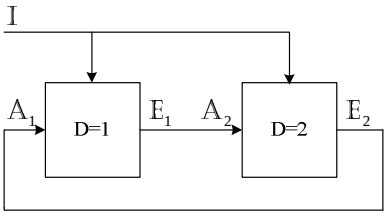

Fig. 3. Passing of extrinsic LLR messages in a 2-dimensional SPC-PC decoder. $\mathbb{I}$ denotes the "channel messages"; $\mathbb{A}$ denotes the a priori messages at the input of a SISO decoder and $\mathbb{E}$ denotes the extrinsic LLR messages at the output of a SISO decoder. The subscript corresponds to the identity of the SISO decoder.

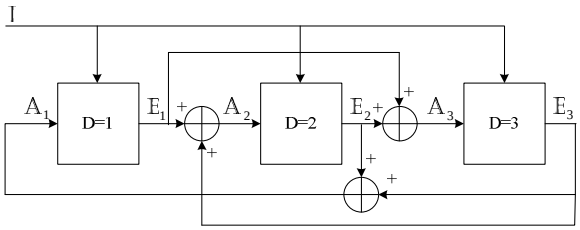

Fig. 4. Passing of extrinsic LLR messages in a 3-dimensional SPC-PC decoder. $\mathbb{I}$ denotes the "channel messages"; $\mathbb{A}$ denotes the a priori messages at the input of a SISO decoder and $\mathbb{E}$ denotes the extrinsic LLR messages at the output of a SISO decoder. The subscript corresponds to the identity of the SISO decoder.

vector randomly selected from $\mathbf{G}_{svn}^{\mathrm{T}}$ where $^{\mathrm{T}}$ denotes the transpose operator; and all "0"s within this block are substituted by the all-zero vector of size $1 \times (d_v - 1)$. However, each row vector in $\mathbf{G}_{svn}^{\mathrm{T}}$ can only be used once for each column in $\mathbf{H}'_a$. The substitution process applies to each column in $\mathbf{H}'_a$. After all substitutions are made, the parity-check matrix $\mathbf{H}$ of the GLDPC code is obtained and is of size $M_a M_{spc-pc} \times N_a(d_v - 1)$.

Note that the encoding complexity of GLDPC codes can be reduced by using quasi-cyclic adjacency matrix, as illustrated in [14] [15].

IV. Decoding Algorithm

Similarly to the standard LDPC code, the message-passing algorithm is used to allow the exchange of extrinsic information between the SVNs and the SCNs iteratively (based on the bipartite graph of the adjacency matrix). Moreover, each super node is regarded as a local decoder, in which a soft-input soft-output (SISO) decoding algorithm is used to decode the constituent code. Since a SPC code is adopted in each of the SVNs, the corresponding SISO decoder can be readily implemented with a relatively low complexity [16], [17]. The complexity of the GLDPC decoder is therefore dominated by the complexity of the local decoders at the SCNs.

Suppose the $(4, 3)^2$ SPC-PC shown in Fig. 2 is used as the constituent code in the SCNs. We propose using local turbo iterations in the SISO decoder for this code. When messages are passed from the SVNs to the SCNs, each SCN treats the incoming messages as "channel messages". To begin the decoding process, we set all the a priori log-likelihood-ratio (LLR) values corresponding to the incoming bit sequence to zero. Then, for the horizontal dimension (first dimension), the extrinsic LLR values are evaluated for all bits in each $(4, 3)$ SPC code [16], [17]. Such extrinsic LLR values are then passed to the vertical dimension (second dimension) which uses them as the a priori LLR values. Subsequently, the extrinsic LLR values are evaluated for all bits in each $(4, 3)$ SPC code (in the vertical dimension). Such extrinsic LLR values are then returned to the horizontal dimension which uses them as the a priori LLR values. One *local* turbo iteration is completed. In the second iteration, the extrinsic LLR values are evaluated for all bits in each $(4, 3)$ SPC code for the horizontal dimension, and so on. After a fixed number of turbo iterations have been carried out, the overall extrinsic LLR value for a particular bit is obtained by summing the corresponding extrinsic LLR values in all dimensions (two dimensions in this case). Finally, these overall extrinsic LLR values are passed to the connected SVNs. The decoder in each SVN, based on all incoming messages, evaluates the extrinsic LLRs and return them to the connected SCNs. One *global* iteration is completed. When a sufficient number of global iterations are performed, the SVNs decode the codeword.

Figures 3 and 4, respectively, illustrate how the extrinsic LLR messages are passed among the SISO decoders for the 2-dimensional and 3-dimensional SPC-PC decoders.

V. Results

A. Error Performance

1) Different decoding algorithms at SCNs: We adopt $(4, 3)$ SPC codes as constituent codes in all SVNs and $(4, 3)^2$ SPC-PCs as constituent codes in all SCNs. First, we use a $M_{a,1} \times N_{a,1} = 250 \times 1000$ adjacency matrix to construct a GLDPC code of rate 0.417 and length 3000. We denote this code as GLDPC-1. We send the all-zero codewords and we assume an additive white Gaussian noise (AWGN) channel. In our simulations, we set the maximum number of global iterations to 50. (Unless otherwise stated, a maximum of 50 global iterations are used throughout this paper.)

Figure 5 depicts the bit-error-rate (BER) performance when GLDPC-1 is decoded with the algorithm described in Section IV. The number of local turbo iterations used at the SCN is set to $1, 3$ and 5. In the same figure, we also plot the BER curve when GLDPC-1 is decoded by using the algorithm in [2] to decode the SCNs. The simulation results show that the BER performance of GLDPC-1 improves when the number of local turbo iterations used to decode the SCNs increases from 1 to 5. We also observe that when 5 local iterations are used, the proposed decoding algorithm outperforms that in [2] by about 0.3 dB at a BER of 2×10^{-7}.

Figure 6 presents the cumulative distribution function (CDF) of LLRs at the SCNs for GLDPC-1 during the first global iteration. Large LLR values imply the code bits are decoded correctly with higher reliabilities. The results draw the same conclusion as in the above, i.e., the proposed decoding algorithm with 5 local iterations outperforms that in [2].

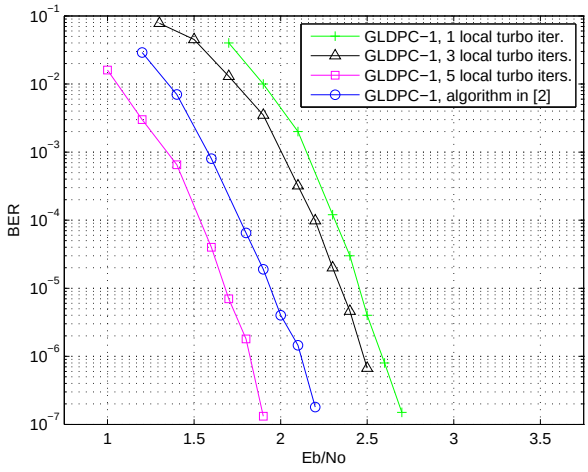

Fig. 5. Bit-error-rate (BER) performance of GLDPC-1 under different decoding algorithms at super check nodes (SCNs).

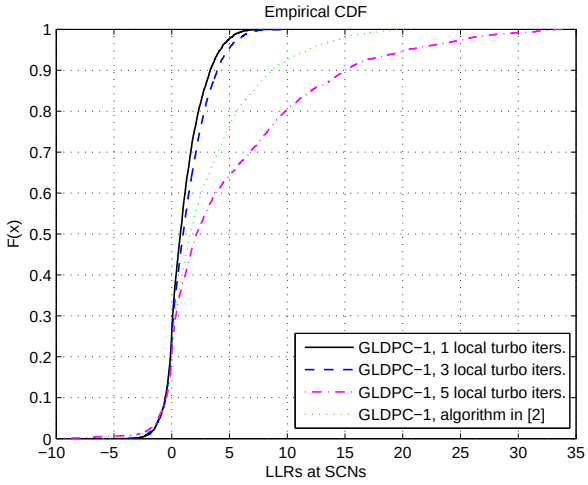

Fig. 6. The CDF of LLRs at SCNs for GLDPC-1 under different decoding algorithms during the first global iteration. $E_b/N_0 = 1.8$ dB.

2) Comparison with other channel codes: We construct another GLDPC code using $(8, 7)$ SPC codes as constituent codes in all SVNs and $(4, 3)^2$ SPC-PCs as constituent codes in all SCNs. The size of the *adjacency matrix* is set to be $M_{a,2} \times N_{a,2} = 146 \times 292$. We then obtain a GLDPC code with rate 0.5 and length 2044, and we denote it as GLDPC-2. Using our proposed decoding algorithm with 3 local iterations at the SCNs, the BER and frame error rate (FER) of GLDPC-2 is shown in Fig. 7.

We also simulate the error performance of (i) the accumulate-repeat-by-4-accumulate (AR4A) code with rate-0.5 and length-2048 proposed in [18], and (ii) the optimized binary LDPC code provided in [19]. The rate of the binary LDPC code is 0.4971 and the parity-check matrix is of size 1030×2048. Moreover, the variable-node and check-node degree distributions of the binary LDPC code are given,

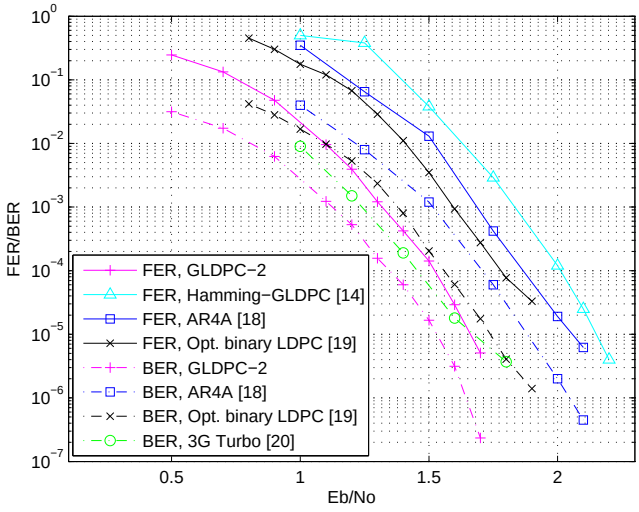

Fig. 7. BER and FER performance of different codes.

respectively, by

$$\lambda(x) = 0.0002x^0 + 0.2648x + 0.2400x^2 + 0.1587x^5 + 0.0871x^6 + 0.0312x^{13} + 0.2180x^{14} \quad (4)$$

$$\text{and} \quad \rho(x) = 0.5546x^6 + 0.4442x^7 + 0.0012x^8. \quad (5)$$

(The average check-node degree of the LDPC code, denoted by $\bar{d}_{LDPC}$, equals 7.41; and the number of check nodes, denoted by M_{LDPC}, equals 1030.) The BER and FER of the AR4A and the optimized binary LDPC code are shown in Fig. 7. We can observe that GLDPC-2 outperforms (i) the AR4A code by 0.4 dB at a BER of 10^{-6} and (ii) the optimized binary LDPC code by 0.25 dB at a BER of 2×10^{-6}.

In Fig. 7, we further re-plot (i) the FER performance of the $(2044, 1022)$ quasi-cyclic GLDPC code (a pure GLDPC code with Hamming constraints at SCNs and repetition codes at SVNs, named as Hamming-GLDPC code) proposed in [14] and (ii) the BER performance of the $(2048, 1024, 0.5)$ 3G Turbo code shown in [20]. Note that the Hamming-GLDPC code possesses the same code length and code rate as GLDPC-2, and it is decoded by the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm at the SCNs. Moreover, a maximum of 50 (global) iterations are used when generating the curves in [14] and [20]. The results depicted in Fig. 7 indicate that GLDPC-2 outperforms (i) the Hamming-GLDPC code by 0.5 dB at a FER of 10^{-5} and (ii) the 3G Turbo code by 0.17 dB at a BER of 10^{-5}.

B. Computation Complexity

In Table I, we show the complexity of the decoder for different codes and different decoding algorithms. We observe that for the GLDPC-1 code, the complexity of our proposed decoder with $\omega_{max} = 5$ local turbo iterations is about 64 times lower than that of the decoding algorithm described in [2]. In addition, our proposed decoder using 3 local turbo

TABLE I
COMPLEXITY OF THE DECODER FOR DIFFERENT CODES AND DIFFERENT DECODING ALGORITHMS.

Code	No. of Multiplications
GLDPC-1 $\omega_{max} = 5$ local turbo iterations	$M_{a,1}\ \omega_{max}\ DN_{spc-pc}\ (N_{spc}-1)$ $= 250 \times 5 \times 2 \times 16 \times (4-1) = 120,000$
GLDPC-1 Algorithm in [2]	$M_{a,1} 2^{K_{spc-pc}} N_{spc-pc}(N_{spc-pc}-1)$ $= 250 \times 2^{(16-9)} \times 16 \times (16-1) = 7,680,000$
GLDPC-2 $\omega_{max} = 3$ local turbo iterations	$M_{a,2}\omega_{max} DN_{spc-pc}\ (N_{spc}-1)$ $= 146 \times 3 \times 2 \times 16 \times (4-1) = 42,048$
Optimized binary LDPC in [19]	$M_{LDPC}\ \bar{d}_{LDPC}\ (\bar{d}_{LDPC}-1)$ $= 1030 \times 7.41 \times (7.41-1) \approx 48,974$

iterations for decoding GLDPC-2 has a similar complexity as the decoder that decodes the binary LDPC code in [19].

What is more, the computation of each SPC component code in each dimension of each SCN is the same. It implies that such computations can possibly be implemented in a highly parallel manner.

VI. CONCLUSION

In this paper, we introduce a class of GLDPC code with single-parity-check product-codes (SPC-PCs) as component codes in the super check nodes (SCNs). Simulation results show that the proposed GLDPC codes can be decoded with low-complexity decoder and have remarkable error performance.

There are two advantages of using PCs as component codes in the SCNs. Firstly, the minimum distance of a PC increases exponentially with the number of dimensions, making the GLDPC ensemble more powerful in terms of error-correction capability. Secondly, due to its special structure, a PC can be split into shorter component codes. Since the shorter codes can be decoded with reduced complexity, the overall complexity of the turbo decoder at the SCN is also reduced.

In general, PCs consisting of any type of component codes can be used in the SCNs of a GLDPC code. At each SCN, each of the component codes of the PC can be decoded with a SISO decoder using the BCJR algorithm. Then the extrinsic information of each SISO decoder output can be exchanged among the different dimensions of the PC by using a small number of local turbo iterations. By adjusting the number of local turbo iterations, a tradeoff between the error performance and the computation complexity can be obtained. This is one of the future works that are being pursued. We are also investigating the performance of the GLDPC code proposed in this paper when the dimension of the SPC-PCs increases. Finally, we have considered only regular GLDPC codes in this paper. Our another goal is to optimize the GLDPC code structure based on the extrinsic-information-transfer (EXIT) chart. We aim at finding irregular GLDPC code structures that possess excellent error performance.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their constructive comments that have helped improving the overall quality of the paper.

REFERENCES

[1] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, pp. 533–547, Sep. 1981.
[2] Y. Wang and M. Fossorier, "Doubly generalized LDPC codes over the AWGN channel," *IEEE Trans. Commun.*, vol. 57, pp. 1312–1319, May 2009.
[3] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, Mar. 1974.
[4] E. Paolini, M. Fossorier, and M. Chiani, "Generalized stability condition for generalized and doubly-generalized LDPC codes," in *Proc. IEEE ISIT*, pp. 1536–1540, Jun. 2007.
[5] M. Lentmaier and K. Zigangirov, "On generalized low-density parity-check codes based on Hamming component codes," *IEEE Commun. Letters*, vol. 3, pp. 248–250, Aug. 1999.
[6] 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.
[7] N. Miladinovic and M. Fossorier, "Generalized LDPC codes with Reed-Solomon and BCH codes as component codes for binary channels," in *Proc. IEEE GLOBECOM*, vol. 3, p. 6, Nov. 2005.
[8] J. Chen and R. M. Tanner, "A hybrid coding scheme for the Gilbert-Elliott channel," *IEEE Trans. Commun.*, vol. 54, pp. 1787–1796, 2006.
[9] S. Abu-Surra, G. Liva, and W. E. Ryan, "Low-floor Tanner codes via Hamming-node or RSCC-Node doping," in *Proc. the 16th international conference on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pp. 245–254, 2006.
[10] X. Wang and X. Ma, "A class of generalized LDPC codes with fast parallel decoding algorithms," *IEEE Commun. Letters*, vol. 13, pp. 531–533, Jul. 2009.
[11] G. Yue, L. Ping, and X. Wang, "Generalized low-density parity-check codes based on Hadamard constraints," *IEEE Trans. Inform. Theory*, vol. 53, pp. 1058–1079, Mar. 2007.
[12] D. Rankin and T. Gulliver, "Single parity check product codes," *IEEE Trans. Commun.*, vol. 49, pp. 1354–1362, Aug. 2001.
[13] A. Barg and G. Zemor, "Distance properties of expander codes," *IEEE Trans. Inform. Theory*, vol. 52, pp. 78–90, Jan. 2006.
[14] G. Liva, W. Ryan, and M. Chiani, "Quasi-cyclic generalized LDPC codes with low error floors," *IEEE Trans. Inform. Theory*, vol. 56, pp. 49–57, Jan. 2008.
[15] Z. Li, L. Chen, L. Zeng, S. Lin, and W. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 54, pp. 71–81, Jan. 2006.
[16] B. Sklar, *Digital Communications: Fundamentals and Applications*. second edition, Prentice Hall, 2004.
[17] A. Neubaue, J. Freudenberger, and V. Kuhn, *Coding Theory: Algorithms, Architectures and Application*. John Wiley and Sons, 2007.
[18] G. Liva, S. Song, L. Lan, Y. Zhang, S. Lin, and W. Ryan, "Design of LDPC codes: a survey and new results," *IEEE Journal of Commun. Software and Systems*, vol. 2, pp. 191–211, Sept. 2006.
[19] D. J. Mackay, *Encyclopedia of Sparse Graph Codes*. http://www.inference.phy.cam.ac.uk/mackay/codes/EN/C/PEGirUppTriang1030x2048.gz, 2008.
[20] M. Mansour, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," *IEEE Journal of Solid-State Circuits*, vol. 41, pp. 684–698, Mar. 2006.