A CLASS OF EFFICIENT-ENCODING GENERALIZED LOW-DENSITY PARITY-CHECK CODES

Tong Zhang and Keshab K. Parhi

Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN 55455, USA
E-mail:{tzhang, parhi}@ece.umn.edu

ABSTRACT

In this paper, we investigate an efficient encoding approach for generalized low-density (GLD) parity check codes, a generalization of Gallager's low-density parity check (LDPC) codes. We propose a systematic approach to construct approximate upper triangular GLD parity check matrix which defines a class of efficient-encoding GLD codes. It's shown that such GLD codes have equally good performance. By effectively exploiting the structure sharing in the encoding process, we also present a hardware/software codesign for the practical encoder implementation of these efficient-encoding GLD codes.

1. INTRODUCTION

Low-Density Parity-Check (LDPC) codes were first introduced by Gallager [1][2] in 1960's. In his original work, two innovative ideas were exploited: iterative decoding and constrained random code ensemble. However, Gallager's work was forgotten by the majority of the scientific community for the next 30 years until the discovery of Turbo codes in which both above ideas are employed. LDPC codes were independently rediscovered by both MacKay and Neal [3] and Wiberg [4]. In past few years, LDPC codes have received a lot of attention and many new developments have been brought in this area.

Recently, a class of pseudo-random error correcting codes, called *Generalized Low-Density (GLD) Parity-Check* codes, were introduced by Lentmaier [5] and Boutros [6], independently. As a generalization of Gallager's LDPC codes, GLD codes are constructed by replacing each parity check equation in LDPC codes with the parity check matrix of a small linear block code called the *constituent code*. GLD codes can be effectively decoded using iterative decoding algorithm based on Soft Input Soft Output (SISO) decoding of the constituent code. It has been shown that GLD codes are asymptotically good in the sense of minimum distance and exhibit an excellent performance on both AWGN and Rayleigh channels. Furthermore, GLD decoder has a regular and parallel structure with high computational localization, which make it very suitable for practical integrated circuit (IC) implementations.

However, like LDPC, the major drawback of GLD codes lies in their high encoding complexity. The straightforward encoding scheme for GLD codes, using the generator matrix, has quadratic complexity in the block length. It's suggested in [7] and [8] that using an approximate upper triangular parity check matrix to construct LDPC code can reduce the encoding complexity significantly without performance degradation. In this paper, inspired by the above idea, we investigate the efficient encoding of GLD codes. We present a systematic approach to construct approximate upper triangular GLD parity check matrix which defines a class of efficient-encoding GLD codes. Such GLD codes can be efficiently encoded by exploiting the sparseness of parity check matrix. It's shown that the performance of such efficient-encoding GLD codes is as good as ordinary GLD codes. Moreover, by exploiting the structure sharing in the encoding process, we propose a hardware/software codesign for the practical encoder implementation of such GLD codes.

This research was supported by the Army Research Office by grant number DA/DAAG55-98-1-0315.

2. GENERALIZED LOW-DENSITY PARITY-CHECK CODES

In this section, according to [5] and [6], we briefly describe the construction of GLD codes and their iterative decoding process.

It's well known that LDPC codes are defined by a sparse parity check matrix, in which each row is a single-error detecting parity check equation. As a generalization of LDPC codes, GLD codes are also specified by a sparse parity check matrix H, constructed by replacing each row in LDPC parity check matrix with $(n-k)$ rows including one copy of the parity check matrix H_0 of constituent code $C_0(n,k)$, a k-dimensional linear code of length n. Such $(n-k)$ rows including one copy of H_0 construct a *constituent submatrix* in H. The structure of a GLD parity check matrix is depicted in Fig. 1, where the construction approach is very similar with that of Gallager's LDPC codes [2]. Matrix H^0 is a *block diagonal* matrix and just produces the direct sum of N/n constituent codes as shown in Fig. 1, where N is the GLD code length. Parity check matrix H of GLD code is divided into J submatrices, $H^1 \cdots H^J$, each containing a single column of constituent parity check matrix H_0 in each column. Each submatrix is constructed as: $H^j = \pi_j(H^0)$ for $j = 1 \cdots J$, where π_j represents a column permutation. For the case of simplicity, we usually let $H^1 = H^0$. Obviously, each H^j contains N/n constituent submatrices. A (N, J, n) GLD code C can be seen as the intersection of J *super-codes* $C^1, \cdots, C^J$, whose parity check matrices are the J submatrices, $H^1, \cdots, H^J$, respectively. In practice, π_i's are chosen at random with the only condition that in parity check matrix H no two constituent submatrices have more than one overlapping nonzero column.

GLD codes can be effectively decoded using the following it-

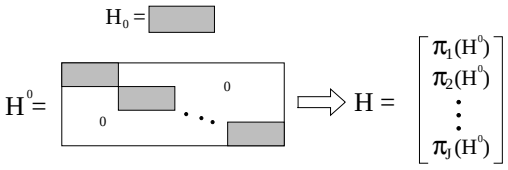

Fig. 1. Structure of GLD parity check matrix H

erative decoding scheme: For each bit, we compute its probability given its received sample considering that it belongs to the super-code C^1. Since each super-code is composed of N/n independent constituent codes, we can use N/n Soft-Input Soft-Output (SISO) decoders working in parallel on each constituent code. This step generates for each coded bit an *a posteriori probability* (APP) and an *extrinsic* probability. The latter one, as an *a priori* information, is fed to the SISO decoders working on the N/n constituent codes of super-code C^2. This process is iterated on each super-code: $C^1 \rightarrow C^2 \rightarrow \cdots \rightarrow C^J \rightarrow C^1 \rightarrow \cdots$.

It has been shown in [5][6] that binary GLD codes with only $J = 2$ levels is asymptotically good. Furthermore, GLD codes with 2 levels have the highest code rate and simple decoder structure, which are desirable in real applications. Thus, in this work, we only consider the efficient encoding of $(N, 2, n)$ GLD codes. Here we note that for $(N, 2, n)$ GLD codes, only if $N/n \geq n$, it is possible to construct a parity check matrix H in which no two constituent submatrices have more than one overlapping nonzero column. Therefore, the $(N, 2, n)$ GLD codes dealt with in the rest of paper always satisfy $N/n \geq n$.

3. EFFICIENT-ENCODING GLD CODES

As we have mentioned, one major drawback of GLD code is its apparent high encoding complexity, which is generally scaled as N^2 if the straightforward encoding approach is used. It has been shown in [7] and [8] that LDPC code can be efficiently encoded based on an approximate upper triangular parity check matrix. Inspired by their work, we investigate a similar efficient encoding scheme for $(N, 2, n)$ GLD codes.

In [8] the greedy algorithms are used to construct approximate upper triangular LDPC parity check matrix. Different with that approach, based on the structure of GLD parity check matrix, we propose the following systematic approach to construct approximate upper triangular $(N, 2, n)$ GLD parity check matrix H under the condition that no two constituent submatrices have more than one overlapping nonzero column. Then we briefly describe how the encoding is efficiently carried out based on such parity check matrix.

3.1. Construction of H

Let the constituent code C_0 be an (n, k) code and its parity check matrix H_0 have systematic form $[I, P]$, where I is an $(n-k)$ by $(n-k)$ identity matrix. We define N/n as s and $s \cdot (n-k)$ as L, respectively. In the following, we present the systematic construction approach of H in two steps:

1. Construct a matrix $\hat{H} = [\hat{H}^{1^T}, \hat{H}^{2^T}]^T$ where both $\hat{H}^1$ and $\hat{H}^2$ are L by N dimensional and contain s constituent submatrices;
2. Obtain H by reordering certain columns of $\hat{H}$.

First, we construct $\hat{H}^1$ as $[I, SP]$, where I is an L by L identity matrix and SP is a block diagonal matrix containing s copies of submatrix P as shown in Fig. 2. We note that $\hat{H}^1$ can be seen as the parity check matrix of a super-code which consists of s constituent codes.

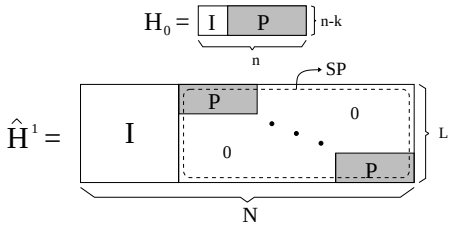

Fig. 2. Structure of matrix $\hat{H}^1$

$\hat{H}^2$ is constructed by permuting columns of matrix Q as shown in Fig. 3. We write matrix Q in block matrix form as $[Q_1, Q_2]$, where Q_1 and Q_2 are L by $(N-L)$ and L by L, respectively. By introducing two column permutations, π_1 and π_2, we construct $\hat{H}^2$ as $[\pi_1(Q_2), \pi_2(Q_1)]$. $\hat{H}^2$ also defines a super-code consisting of s constituent codes. Combining $\hat{H}^1$ and $\hat{H}^2$ together, we get a $(N, 2, n)$ GLD parity check matrix $\hat{H} = [\hat{H}^{1^T}, \hat{H}^{2^T}]^T$. Here π_1 and π_2 are chosen at random with the condition that no two constituent submatrices in $\hat{H}$ have more than one overlapping nonzero column. Based on the prerequisite that $N/n \geq n$ and the structure of $\hat{H}^1$ and Q, we can prove that such two permutations always exist.

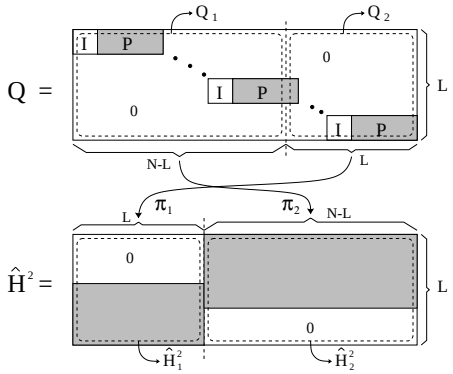

Fig. 3. Structure of matrix H^2

Since $\hat{H}_2^2 = \pi_2(Q_1)$ and Q_1 contains

$$\lfloor \frac{N-L}{n} \rfloor = \lfloor \frac{s \cdot n - s \cdot (n-k)}{n} \rfloor = \lfloor \frac{s \cdot k}{n} \rfloor$$

complete copies of systematic parity check matrix H_0, we can always find a column permutation π_3 which makes $H = \pi_3(\hat{H})$ has the approximate upper triangular form as shown in Fig. 4, in which each P_i, $i = 1 \cdots s$, is obtained by removing some columns from matrix P. As shown in Fig. 4, H can be written as

$$H = \begin{bmatrix} T & B & D \\ A & C & E \end{bmatrix}, \quad (1)$$

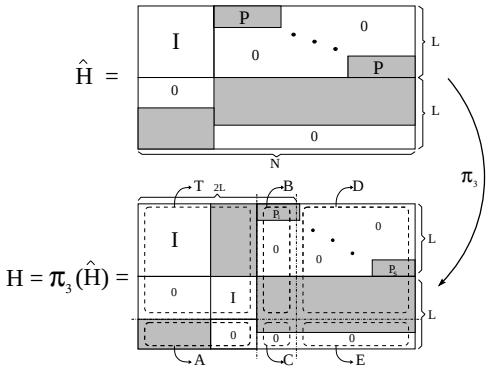

Fig. 4. Structure of matrix H

where the left $2L$ by $2L$ submatrix is

$$\begin{bmatrix} T & B \\ A & C \end{bmatrix}, \tag{2}$$

in which T is K by K upper triangular submatrix, where $K = L + k \cdot \lfloor \frac{s \cdot k}{n} \rfloor$. The GLD codes defined by H, called *efficient-encoding* GLD codes, are a special class of the ordinary GLD codes described in Section 2. Through simulations it's shown that such GLD codes have equally good performance.

Example 3.1 *Let's consider the (N,2,15) GLD codes with Hamming constituent code (15,11). Fig. 5 shows the performance of both ordinary and efficient-encoding GLD codes at two different code length configurations. These GLD codes are modulated by BPSK and transfered over AWGN channel. In both cases, we pick 20 permutation patterns at random and select the one leading to the best results, and in the iterative decoding process each super-code decoding is performed by means of MAP decoder.*

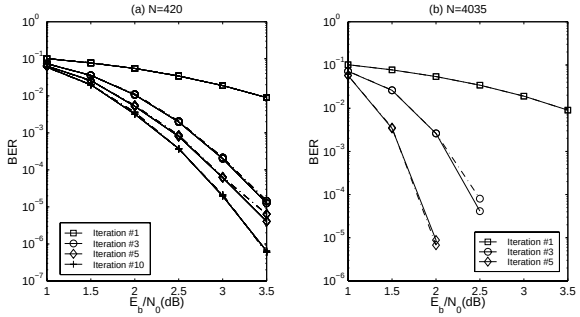

Fig. 5. Simulation results for (a) N=420 and (b) N=4035, where solid lines correspond to ordinary GLD, dash dot lines for efficient-encoding GLD.

3.2. Encoding Process

In the following, we describe how the efficient-encoding is actually carried out based on the approximate upper triangular GLD parity check matrix. Let $x = (x_a, x_b, x_c)$ be a codeword decomposed according to (1), where x_c is the information bits with length of $N - 2L$, redundant bits x_a and x_b have length of $L + k \cdot \lfloor \frac{s \cdot k}{n} \rfloor$ and $L - k \cdot \lfloor \frac{s \cdot k}{n} \rfloor$, respectively.

Procedure 3.1

1. *Compute $y_c = D \cdot x_c$ and $z_c = E \cdot x_c$, which is efficient because both D and E are sparse;*
2. *Solve $T \cdot \hat{x}_a = y_c$. Since T has the form as shown in Fig. 4, we can prove that $T^{-1} = T$. Thus we have $\hat{x}_a = T \cdot y_c$, which can be easily computed since T is sparse;*
3. *Evaluate $\hat{s} = A \cdot \hat{x}_a + z_c$, which is also efficient since A is sparse;*
4. *Compute $x_b = \phi \cdot \hat{s}$, where $\phi = (A \cdot T \cdot B + C)^{-1}$. In this step, the complexity is scaled by $(L - k \cdot \lfloor \frac{s \cdot k}{n} \rfloor)^2$.*
5. *Finally we can obtain x_a by solving $T \cdot x_a = B \cdot x_b + y_c$. Since $T^{-1} = T$, $x_a = T \cdot (B \cdot x_b + y_c)$. This is efficient since both T and B are sparse.*

We note that the above encoding process is similar with that in [8] for LDPC codes. However, in [8], $\hat{x}_a$ and x_a have to be solved using the less efficient *back-substitution* method since $T^{-1} \neq T$ in that case.

4. HARDWARE/SOFTWARE CODESIGN OF GLD ENCODER

In a coding system, the decoding delay is much longer than encoding delay by one or two orders of magnitude which is true even for LDPC or GLD codes. Thus encoding process is not speed-critical and software-based encoder is usually employed. In this section, we consider the practical software-based encoder implementation for the above efficient-encoding GLD codes.

As shown in Procedure 3.1, the encoding process mainly consists of several bit level matrix-vector multiplications. However, these bit level computations can't be efficiently performed in general purpose digital signal processors (DSPs). One natural solution is to incorporate a specialized bit manipulation unit (BMU) into the processor's data path to improve its efficiency. Before presenting our proposed architecture design for such BMU, we first introduce the following computation approach for the multiplication of a special matrix and a variable vector.

Recall that H_0, the parity check matrix of constituent code $C_0(n, k)$, has the systematic form $[I, P]$. We define the block diagonal matrix, as shown in Fig. 6, as *(P,t)-block diagonal* matrix, in which each block P_i is obtained by removing l_i columns from P, where $0 \leq l_i \leq k$ for $1 \leq i \leq t$. Thus each P_i can be expressed as $P_i = P \cdot I_i$, where I_i is constructed by removing the corresponding l_i columns from the $(n - k)$ by $(n - k)$ identity matrix.

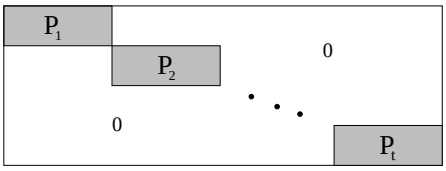

Fig. 6. (P,t)-block diagonal matrix

For the multiplication of a (P, t)-block diagonal matrix U and a vector v, we have

$$U \cdot v = [P_1 \cdot v_1, \cdots, P_t \cdot v_t]^T \tag{3}$$

where $v = [v_1^T, \cdots, v_t^T]^T$. Substituting $P_i = P \cdot I_i$ into (3), we get

$$U \cdot v = [P \cdot \tilde{v}_1, \cdots, P \cdot \tilde{v}_t]^T \quad (4)$$

where $\tilde{v}_i = I_i \cdot v_i$. Since I_i is constructed by removing l_i columns from identity matrix, $\tilde{v}_i$ can be obtained directly by inserting zeroes into v_i at the l_i corresponding positions. Moreover, for a matrix S constructed by column permutation of a (P, t)-block diagonal matrix U, that is $S = \pi(U)$, we have

$$\begin{aligned} S \cdot v &= \pi(U) \cdot v = U \cdot z \\ &= [P \cdot \tilde{z}_1, \cdots, P \cdot \tilde{z}_t] \end{aligned} \quad (5)$$

where $z^T = \pi^{-1}(v^T)$ and $\tilde{z}_i = I_i \cdot z_i$. It's suggested by (4) and (5) that we may reuse one simple dedicated hardware unit, which performs the multiplication of P and a variable vector, by t times to implement the multiplication of a (P, t)-block diagonal matrix U (or a column permutation of U) and a vector v, where v needs to be manipulated (inserting of zeroes and permutation) accordingly.

As shown in Fig. 4, we can write several block submatrices of H as: $A = [A_1, 0]$, $B = [B_1^T, 0, B_2^T]^T$, $D = [D_1^T, D_2^T]^T$, where D_1^T consists of upper L rows of D, and

$$T = \begin{bmatrix} I & F \\ 0 & I \end{bmatrix}. \quad (6)$$

Let's introduce a set $\mathcal{G} = \{A_1, B_1, B_2, D_1, D_2, F\}$. According to the construction approach of H presented in Section 3.1, each matrix in set $\mathcal{G}$ is either (P, t)-block diagonal matrix or column permutation of (P, t)-block diagonal matrix. Therefore, except the multiplication by ϕ, all other matrix-vector multiplications in the encoding process can be performed using the above computation scheme in which a dedicated hardware unit is introduced to carry out the multiplication of matrix P and a variable vector.

Based on the above discussion, we propose a hardware/software codesign for the encoder of efficient-encoding GLD codes, where the DSPs data path architecture is shown in Fig. 7. The incorporated BMU includes two hardware units: Direct multiplication unit (DMU) is a dedicated hardware unit to perform the multiplication of matrix P and a variable vector; Barrel shifter efficiently performs the bit manipulation operations which will be extensively used when inserting zeroes into vectors and permuting vectors. Here it should be pointed out that many modern processors provide a barrel shifter in the main data path, in these cases we only need to incorporate the DMU into the data path. Using the above approach, most of the matrix-vector multiplications in encoding process can be efficiently performed. Since the encoding process is matrix-vector multiplication intensive, such hardware/software codesign approach may provide a much more efficient solution compared with a solely general processor based encoder.

5. CONCLUSION

In this paper, we have presented an efficient encoding scheme for the 2-level GLD codes. We proposed a systematic approach to construct approximate upper triangular GLD parity check matrix under the condition that no two constituent submatrices have more than one overlapping nonzero column. We showed that GLD codes defined by such parity check matrix can be efficiently encoded, and their performance is as good as the ordinary GLD codes. Such GLD codes may have practical importance for the applications with limited computation power. Moreover, by incorporating special function units into the general purpose DSPs data path, we presented an efficient hardware/software codesign for encoder of these GLD codes.

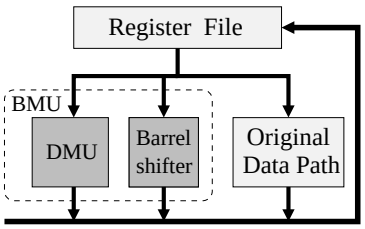

Fig. 7. Proposed Data Path architecture in a DSP processor

6. REFERENCES

[1] R. G. Gallager, "Low-density parity-check codes", *IRE Trans. Inform. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[2] R. G. Gallager, *Low-Density Parity-Check Codes*, M.I.T Press, 1963.

[3] D. J. C. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes", *Electronics Letters*, vol. 32, pp. 1645–1646, Aug. 1996.

[4] N. Wiberg, "Codes and decoding on general graphs", Ph.D. Dissertation, Linkoping University, Sweden, 1996.

[5] M. Lentmaier and K. S. Ziganfirov, "Iterative decoding of generalized low-density parity-check codes", *in Proceedings of IEEE International Symposium on Information Theory*, p. 149, 1998.

[6] J. Boutros, O. Pothier, and G. Zemor, "Generalized low density (Tanner) codes", *in Proceedings of ICC'99*, pp. 441 – 445, Vancouver, June 1999.

[7] David J. C. MacKay, Simon T. Wilson, and Matthew C. Davey, "Comparison of constructions of irregular gallager codes", *IEEE Transactions on Communications*, vol. 47, pp. 1449–1454, Oct. 1999.

[8] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes", *submitted IEEE Transactions on Information Theory*.

[9] O. Pothier, "Compound codes based on graphs and their iterative decoding", Ph.D. thesis, Ecole Nationale Suprieure des Tlcommunications, Jan. 2000.

[10] L. Song, K. K. Parhi, I. Kuroda, and T. Nishitani, "Hardware/software codesign of finite field datapath for low-energy Reed-Solomon codecs", *IEEE Trans. on VLSI Systems*, vol. 8, pp. 160–172, April 2000.

[11] P. Lapsley, J. Bier, A. Shoham, and E. A. Lee, *DSP Processor Fundamentals: architectures and features*, IEEE Press, 1997.