Project	IEEE 802.16 Broadband Wireless Access Working Group <**http://ieee802.org/16**>	
Title	LDPC coding for OFDMA PHY	
Date Submitted	2004-08-24	
Source(s)	Brian Classon Yufei Blankenship Motorola	brian.classon@motorola.com yufei.blankenship@motorola.com
Re:	IEEE P802.16-REVe/D4-2004, ballot #14c	
Abstract	This contribution provides the LDPC code used to generate the plots contained in contribution 372. The LDPC code is an updated version of the Motorola code provided in contribution 278r1.	
Purpose	Complete the LDPC specification text.	
Notice	This document has been prepared to assist IEEE 802.16. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.	
Release	The contributor grants a free, irrevocable license to the IEEE to incorporate material contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE's name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE's sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that this contribution may be made public by IEEE 802.16.	
Patent Policy and Procedures	The contributor is familiar with the IEEE 802.16 Patent Policy and Procedures <http://ieee802.org/16/ipr/patents/policy.html>, including the statement "IEEE standards may include the known use of patent(s), including patent applications, provided the IEEE receives assurance from the patent holder or applicant with respect to patents essential for compliance with both mandatory and optional portions of the standard." Early disclosure to the Working Group of patent information that might be relevant to the standard is essential to reduce the possibility for delays in the development process and increase the likelihood that the draft publication will be approved for publication. Please notify the Chair <mailto:chair@wirelessman.org> as early as possible, in written or electronic form, if patented technology (or technology under patent application) might be incorporated into a draft standard being developed within the IEEE 802.16 Working Group. The Chair will disclose this notification via the IEEE 802.16 web site <http://ieee802.org/16/ipr/patents/notices>.	

Overview

An informal LDPC group has been working on the goal of achieving consensus on a proposed LDPC code design as an optional advanced code for the OFDMA PHY. Many excellent code designs have been submitted. The codes have been qualitatively and quantitatively characterized, and it is clear that a LDPC code with excellent flexibility and performance, as well as low encoding and decoding complexity, can be defined for 802.16e.

Eight companies (Intel, LG, Motorola, Nokia, Nortel, Runcom, Samsung, and TI) provided detailed proposals with code descriptions and simulation results to the group on 13 August 2004. Contribution 278r1 provided specification text for three of the proposals that share a large amount of commonality (Intel, Motorola, Samsung). This contribution provides the LDPC code used to generate the plots contained in contribution 372. The LDPC code is an updated version of the Motorola code provided in contribution 278r1.

References

[1] Bo Xia and Eric Jacobsen, “Intel LDPC Proposed for IEEE 802.16e,” Intel submission to informal LDPC group, 13 August 2004.
[2] Min-seok Oh, Kyuhyuk Chung, “Scalable LDPC coding scheme for OFDMA,” LG submission to informal LDPC group, 13 August 2004.
[3] Y. Blankenship, B. Classon, and K. Blankenship, “Motorola Harmonized Structured LDPC Proposal,” Motorola submission to informal LDPC group, 13 August 2004.
[4] V. Stolpman, J. Zhang, N. van Waes, “Irregular structured LPDC codes,” Nokia submission to informal LDPC group, 13 August 2004.
[5] N. Burns, A. Purkovic, S. Sukobok, B. Johnson, “Algebraic low-density parity check codes for OFDMA PHY layer,” Nortel submission to informal LDPC group, 13 August 2004.
[6] E. Shasha, S. Litsyn, and E. Sharon, “Multi-rate LDPC code for OFDMA PHY,” Runcom submission to informal LDPC group, 13 August 2004.
[7] Jerry Kim, Gyubum Kyung, Hongsil Jeong, Sanghyo Kim, Panyuh Joo and DS Park, “Samsung’s Harmonized Structured LDPC Proposal,” Samsung submission to informal LDPC group, r1, 17 August 2004.
[8] D. Hocevar and A. Batra, “LDPC coding scheme for OFDMA,” TI submission to informal LDPC group, 13 August 2004.

Features

The LDPC codes have excellent performance, and contain features that provide flexibility and low encoding/decoding complexity.

- **Structured block LDPC for low complexity decoding**. The entire matrix (i.e., both the sections that correspond to the information and the parity) is composed of the same style of blocks, which reduces decoder implementation complexity and allows structured decoding.
- **Shortening for low-complexity block size flexibility**. The information portion of the matrix is defined with a non-uniform interlacing of column weights such that excellent performance is achieved through shortening.
- **Low-complexity differential-style encoding**. The encoding can be performed in a structured, recursive manner, without hurting performance with multiple weight-1 columns.
- **Designed to match the OFDMA subchannel structure**. No puncturing or rate-matching operations are required to provide exact code rates for many different block sizes.
- **No channel interleaver required**.
- **Compatible with hybrid ARQ** (Chase or Incremental Redundancy).

Simulation Results

Simulation results for rate 1/2, 2/3, and 3/4 code families are shown in Figure 1, Figure 2, Figure 3, respectively. The code sizes considered are $n = 48*z$, where z is the expansion factor. For rate 1/2, 2/3, and 3/4, 19 z values ranging from 12 to 48 are shown, which correspond to 19 code sizes with n ranging from 576 to 2304. The simulation conditions are: AWGN channel, BPSK modulation, maximum of 50 iterations using generic floating-point belief propagation.

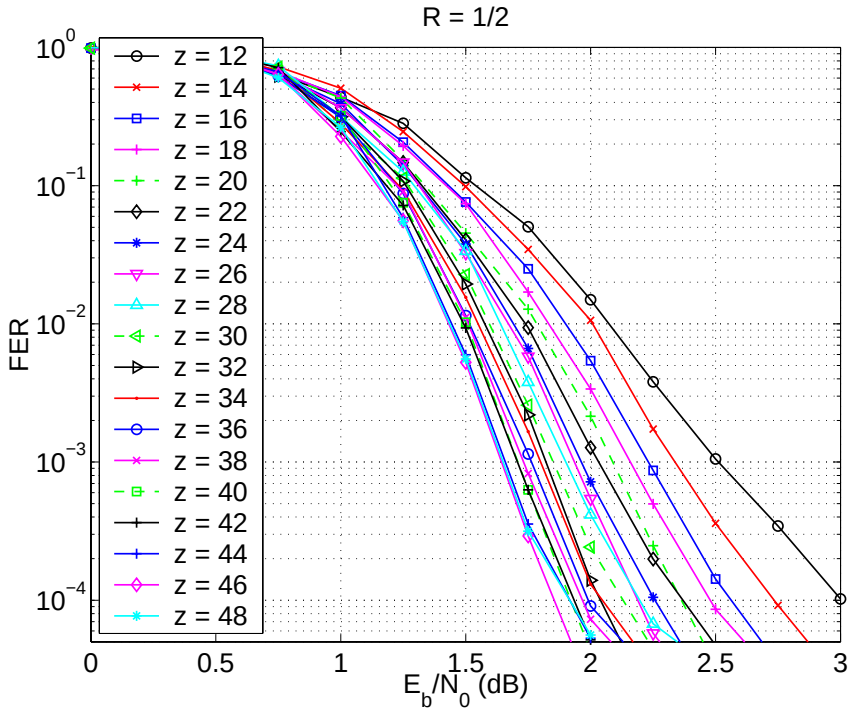


Figure 1. FER performance of 19 R = 1/2 structured codes. Base matrix size: $m_b = 24$, $n_b = 48$. AWGN, BPSK.

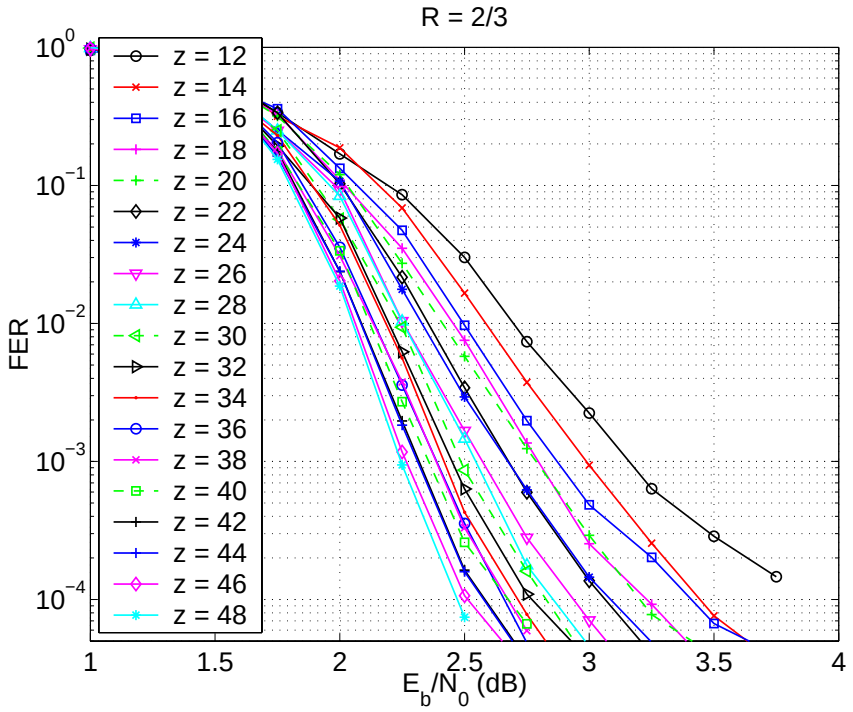


Figure 2. FER performance of 19 R = 2/3 structured codes. Base matrix size: m_b = 16, n_b = 48. AWGN, BPSK.

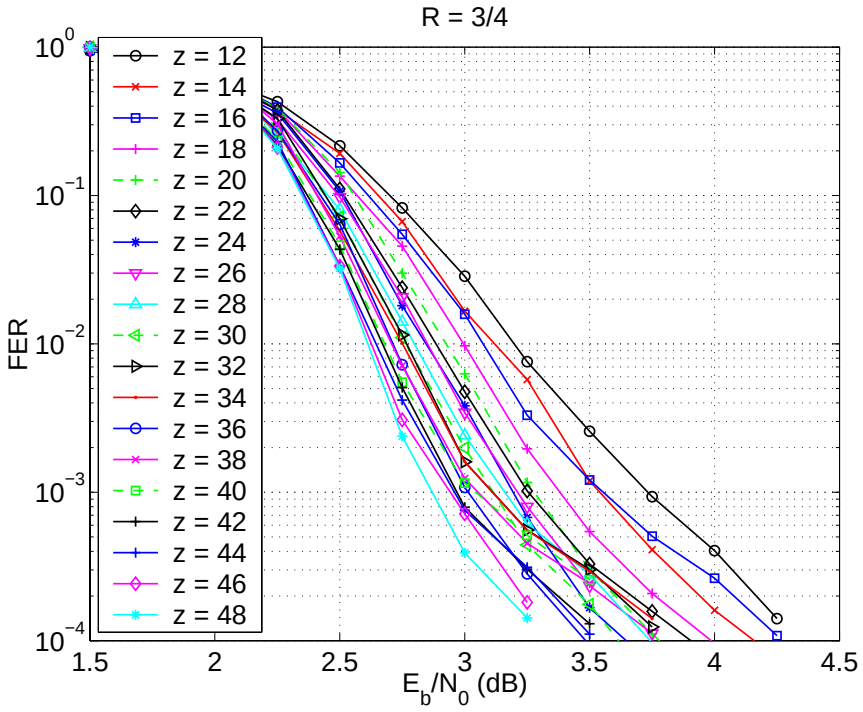


Figure 3. FER performance of 19 R = 3/4 structured codes. Base matrix size: m_b = 12, n_b = 48. AWGN, BPSK.

Recommended Text Changes:

Add/Modify the following text to 802.16e_D4, adjusting the numbering as required:
3

8.4.9.2 Encoding

<add text to the end of the 'Concatenation' paragraph starting at line 39>
, and for the LDPC encoding scheme (see 8.4.9.2.5) the concatenation rule is defined in 8.4.2.9.5.4.

8.4.9.2.5 Low Density Parity Check Code (optional)

8.4.9.2.5.1 Code Description

The LDPC code is based on a set of one or more fundamental LDPC codes. Each of the fundamental codes is a systematic linear block code. Using the described methods of scaling and shortening in 8.4.9.2.5.3 Code Rate and Block Size Adjustment, the fundamental codes can accommodate various code rates and packet sizes. ~~The code set can be applied to packets from [40] bytes up to ~200 bytes.~~

Each LDPC code in the set of LDPC codes is defined by a matrix **H** of size *m*-by-*n*, where *n* is the length of the code and *m* is the number of parity check bits in the code. The number of systematic bits is *k*=*n*-*m*.

The matrix **H** is defined as

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \cdots & \mathbf{P}_{0,n_b-2} & \mathbf{P}_{0,n_b-1} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \cdots & \mathbf{P}_{1,n_b-2} & \mathbf{P}_{1,n_b-1} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \cdots & \mathbf{P}_{2,n_b-2} & \mathbf{P}_{2,n_b-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ \mathbf{P}_{m_b-1,0} & \mathbf{P}_{m_b-1,1} & \mathbf{P}_{m_b-1,2} & \cdots & \mathbf{P}_{m_b-1,n_b-2} & \mathbf{P}_{m_b-1,n_b-1} \end{bmatrix} = \mathbf{P}^{H_b}$$

where $\mathbf{P}_{i,j}$ is one of a set of *z*-by-*z* permutation matrices or a *z*-by-*z* zero matrix. The matrix **H** is expanded from a binary base matrix $\mathbf{H}_b$ of size m_b-by-n_b, where $n = z \cdot n_b$ and $m = z \cdot m_b$,with *z* an integer ≥ 1. The base matrix is expanded by replacing each 1 in the base matrix with a *z*-by-*z* permutation matrix, and each 0 with a *z*-by-*z* zero matrix. The base matrix n_b is an integer is an integer multiple of 24.

The permutations used are circular right shifts, and the set of permutation matrices contains the *z*×*z* identity matrix and circular right shifted versions of the identity matrix. Because each permutation matrix is specified by a single circular right shift, the binary base matrix information and permutation replacement information can be combined into a single compact model matrix $\mathbf{H}_{bm}$. The model matrix $\mathbf{H}_{bm}$ is the same size as the binary base matrix $\mathbf{H}_b$, with each binary entry (*i*,*j*) of the base matrix $\mathbf{H}_b$ replaced to create the model matrix $\mathbf{H}_{bm}$. Each 0 in $\mathbf{H}_b$ is replaced by a blank or negative value (e.g., by –1) to denote a *z*×*z* all-zero matrix, and each 1 in $\mathbf{H}_b$ is replaced by a circular shift size $p(i,j) \geq 0$. The model matrix $\mathbf{H}_{bm}$ can then be directly expanded to **H**.

$\mathbf{H}_b$ is partitioned into two sections, where $\mathbf{H}_{b1}$ corresponds to the systematic bits and $\mathbf{H}_{b2}$ corresponds to the parity-check bits, such that $\mathbf{H}_b = \left[(\mathbf{H}_{b1})_{m_b \times k_b} \,\vdots\, (\mathbf{H}_{b2})_{m_b \times m_b} \right]$. Section $\mathbf{H}_{b2}$ is further partitioned into two sections, where vector $\mathbf{h}_b$ has odd weight, and $\mathbf{H}'_{b2}$ has a dual-diagonal structure with matrix elements at row *i*, column *j* equal to 1 for *i*=*j*, 1 for *i*=*j*+1, and 0 elsewhere:

$$\mathbf{H}_{b2} = \left[\mathbf{h}_b \,\middle|\, \mathbf{H}'_{b2}\right] = \left[\begin{array}{c|cccccc} h_b(0) & 1 & & & & & \\ h_b(1) & 1 & 1 & & & \mathbf{0} & \\ . & & 1 & \ddots & & & \\ . & & & \ddots & & 1 & \\ . & & \mathbf{0} & & & 1 & 1 \\ h_b(m_b-1) & & & & & & 1 \end{array}\right].$$

The base matrix has $h_b(0)=1$, $h_b(m\text{-}1)=1$, and a third value $h_b(j)$, $0<j<(m_b\text{-}1)$ equal to 1. The base matrix structure avoids having multiple weight-1 columns in the expanded matrix.

In particular, the non-zero submatrices are circularly right shifted by a particular circular shift value. Each 1 in $\mathbf{H}'_{b2}$ is assigned a shift size of 0, and is replaced by a $z\times z$ identity matrix when expanding to **H**. The two 1s located at the top and the bottom of $\mathbf{h}_b$ are assigned equal shift sizes, and the third 1 in the middle of $\mathbf{h}_b$ is given an unpaired shift size.

Model Matrix Set

There are three base matrices, one per code rate of 1/2, 2/3, and 3/4. The three model matrix recommendations are given below for n = 2304. For brevity, the staircase portion $\mathbf{H}'_{bm2}$ is not shown. For other code sizes, the shift sizes are derived from these as follows. One set of shift sizes $\{p(i,j)\}$ is defined for the largest code of a code rate with $z_0 = \max(z_f)$, f=1, 2, 3, …, and used for all the other code sizes of the same rate. For a code size corresponding to expansion factor z_f, its shift sizes $\{p(f, i, j)\}$are derived from $\{p(i,j)\}$ by scaling $p(i,j)$ proportionally,

$$p(f,i,j) = \begin{cases} p(i,j), & p(i,j) \le 0 \\ \left\lfloor \dfrac{p(i,j)z_f}{z_0} \right\rfloor = \left\lfloor \dfrac{p(i,j)}{\alpha_f} \right\rfloor, & p(i,j) > 0 \end{cases}.$$

Note that $\alpha_f = z_0/z_f$ and **⌊***x***⌋** denotes rounding to the integer that differs from x the least.

Rate 1/2:

```
-1 -1  0 -1 -1  0  0 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1  0 -1 -1  0
15 -1 -1 -1 18 -1 30 -1 -1 -1 -1 16 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 45 -1 -1
47 -1 30 -1 -1 -1 -1 -1 17 33 30 -1 23 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1  5 -1 -1 31 -1 -1 -1 11 -1 -1 -1 30 -1 -1 -1 -1 -1 -1 -1 11 16 -1 -1 -1
29 -1 -1 -1 20 -1 36 -1 -1 -1 -1 -1 -1  2 14 -1 -1 -1 43 -1 -1 -1 -1 -1 -1
-1 -1 -1 46 -1 -1 39 -1 29 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 45 -1  0 -1
-1 -1 -1 -1 10 -1 -1 -1 44 -1 29  4 -1 -1 -1 -1 -1 -1 -1 39 -1 -1  8 -1 -1 -1
-1  1 -1 -1 16 -1 -1 32 -1 -1  3 -1 -1 -1 -1  4 -1 -1 23 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 38 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 41 31 28 33 -1 -1 -1 -1 -1 -1
46 -1 -1 -1 -1 41 -1 -1  2 12 -1 -1 14 -1 -1 -1 -1 -1 42 -1 -1 -1 -1 -1 -1
11 -1 24 -1 -1 -1 11 -1 -1 -1 27 -1 -1 -1  1 19 -1 -1 -1 -1 -1 -1 -1 -1 -1
35 -1 -1 -1 -1 -1 -1 -1 47 -1 -1 -1 34 -1 -1 -1 20 -1 -1 -1 -1 35 -1 -1  0
-1 -1 38 -1 -1 -1  2 -1 -1 -1 13 -1 46 -1 -1 -1 -1 -1  6 -1  0 -1 -1 -1 -1
-1 -1  1 -1 -1 -1 21 -1 -1 46 -1 -1 24 -1 -1 -1 -1 -1 47 17 -1 -1 -1 -1 -1
-1 -1 -1 -1 15 -1 -1 21 -1 -1 21 -1 -1 17 -1 -1 -1 -1 -1 -1 -1 41 -1 -1 -1
-1 -1 24 -1 -1 -1 -1 -1  3 -1 -1 -1 26 -1 -1 -1 -1 40 -1 11 -1  2 -1 -1 -1
28  4 -1 -1 42 41 15 -1 -1 -1 -1 -1 -1 -1 -1 -1  3 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1  2 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 21  0 -1 -1 -1 -1  8 -1 20 -1
-1 -1  9 -1 -1 -1 33 -1 -1 -1 -1 -1 23 -1 -1 -1 -1 -1 -1 -1 15 42 10 -1 -1
-1 -1 46 28 -1 -1 -1 -1 45 -1 -1 -1 -1 -1 -1 28 -1  6 25 -1 -1 -1 -1 -1 -1
```

```
32 -1  1 -1 -1 -1 -1 -1 19 -1 29 41 -1 -1 -1 25 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 40 -1 24 -1 -1 -1 -1 -1 46 -1 20 -1 34 -1 -1 -1 34 -1 -1 -1 -1 -1 -1
26 -1 -1 -1 -1 -1 -1 41  9 -1 -1 -1 -1 -1 -1  9 -1 -1 -1 -1 -1 29  5 -1 -1
11 -1 -1 35 -1 -1 -1 -1 -1 -1 16 -1 -1 -1 -1 39 -1 -1 21  9 -1 -1 -1 -1  0
```

Rate 2/3:

```
 0 -1 -1  0  0  0 -1  0 -1  0 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1  0
 6 -1 -1 10 -1 -1 -1 -1 26 -1 -1 34 16 -1 -1 -1 -1 -1 -1 -1 39 13 21 -1 -1 -1 -1 -1 17 24 -1 -1 -1
-1 23 -1 29 -1 -1 -1 32 -1 -1 -1 -1 41 -1 -1 -1 -1 -1 12 -1 -1 32 -1 -1 -1  4 -1 -1 28 14 -1 21 -1 -1
-1 -1  6 21 29 -1 31 -1 -1 12 -1 -1 47 -1 -1 -1 -1 -1 -1 27  7 -1 47 -1 41 -1 -1 -1 -1 -1 -1 -1 -1
 0 -1 -1 -1 -1 -1  7 -1 -1 18 -1 -1 -1 -1 22  8 38 -1 -1 -1 -1 -1 -1 -1  5 18 -1 -1 30 -1 -1 14 -1
11 15 -1 -1 -1 -1 13 -1 -1 34 33 -1 12 -1 -1 -1 -1 -1 -1 -1  6 -1  6 31 29 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1  1 23 -1 -1  7 -1 -1 -1 -1 -1 -1 16 -1 -1 19 -1 -1 -1 40 -1 -1 -1 -1  3 26 32 36 -1 -1 -1 -1
25 -1 -1 14 -1 -1 -1 -1 -1 19 13 -1  2 -1 -1 -1  6 -1 -1 -1 -1 -1 -1 -1  3 -1 -1 -1 -1  1  2 -1  0
-1 38 -1 -1 -1  9 44 -1 -1 -1 -1 26 27  9 -1 -1  2 -1 37 -1 -1 -1 -1 -1 34 -1 -1 -1 16 -1 -1 -1 -1
44 -1 -1 -1 -1 -1 18 -1 -1  3 -1 -1 24 -1 41 -1 -1 -1 -1 -1 26 -1 -1 28 -1 -1 30 -1 33  8 -1 -1 -1
 8 -1 -1 33 -1 -1 -1 43 39 47 -1 -1 27 -1 -1 -1 -1 -1 -1 -1  1 33 -1 -1 -1 -1 20 -1 -1 22 -1 -1 -1 -1
45 -1 -1 40 -1 18 33 -1 -1 -1 -1 -1 10 -1 14 -1  2 -1 -1 -1 -1 10 -1 -1 37 -1 -1  6 -1 -1 -1 -1 -1
 8 -1 -1 -1 -1 -1 -1 -1 17 12 16 -1 -1  9 -1 32 43 42 -1 -1 20 -1 -1 -1 16 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 44 -1 27 -1 -1 17 -1 -1 29 -1 -1 -1  7 -1 32 -1 -1 -1 -1 10 34 -1 -1 -1 36 -1 24 -1 -1
24 -1 43 34 -1 -1 35 -1 -1 -1 -1 42 -1 -1 -1 -1  1 44 -1 -1 31 -1 -1 -1 -1 -1 -1  0 -1 -1 34 -1
-1 -1 -1  3 -1 -1 26 -1 -1 33 -1 -1 -1 -1 -1 -1 28 -1 25 40 14 27 -1 -1 -1 -1  9 -1 24 -1 -1 -1  0
```

Rate 3/4:

```
 0  0 -1  0 -1 -1  0  0 -1 -1 -1  0  0  0 -1 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1  0  0  0  0 -1 -1 -1 -1 -1 -1  0
-1 31 -1  5 45 35 12 -1 -1 -1 -1 -1 39 -1 -1 -1 17 46 23 -1 -1 25 -1 20 -1 12 -1 -1 15 -1 -1 -1 44 -1 -1 -1 -1
32 -1 -1 41 11 -1 38 -1 -1 -1 -1 -1 44 -1 -1 -1 -1 18 -1 -1 11 16 -1 -1 29 43 26 -1 -1 -1 15 -1 -1 10 -1 35 -1
47 22 -1 -1 -1 -1 41 -1 -1 40 -1 18 35 25  0 -1 17 -1 -1 -1 -1 -1 14 -1 25 15 -1 -1 -1 -1 -1 47 -1 39 -1 -1 -1
16 12 16 39 -1 -1 -1 -1  8  6 23 -1  8 -1 -1 -1 -1 21 -1 -1 20  3 -1 17 -1 -1 -1 -1 25 36 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 43 -1 -1 -1 -1 -1  0 -1 -1 -1  5 43 -1 28 -1  7 -1 -1 23 -1 -1 27 -1 -1 -1 32 11 -1 -1  3 26 27 -1  0
40 -1 -1 -1 -1 38 -1 -1 -1 44  7 -1 -1 13 -1 -1 47 42 -1 25 16 -1 -1 -1 41 -1 26 -1 -1 46 -1 -1  8  1 -1 -1 -1
-1 40 -1 -1 -1 -1  1 43 -1 39 -1 -1 -1 -1 -1  1 22 32 -1 -1 36 -1 -1 -1 -1 26 -1 -1 16 -1 -1 28 15 29 42 -1 -1
-1 15 14 -1 -1 -1 39 -1 -1 -1 -1 -1  3 26 19 -1 -1 47 -1 -1  8 -1 -1 -1 -1 21 -1 11  0 20  4 -1 18 -1 -1 -1 -1
-1 -1 -1 16  2 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 19 -1 -1 -1 -1 24 -1 42  5 -1 -1 14  9 -1 -1 30 36 45 42 -1
44 -1 -1 -1 -1 -1 31 38 44 -1 -1 -1 -1 41 -1 -1 30 -1 16 22 25 35 -1 -1 10 40 -1 -1 -1 -1 -1 -1  1  0 -1 -1 -1
27 12 -1 16 -1 -1 -1 -1 -1 24 38 -1 41 34 -1 12 16 -1 -1 -1 -1 31 30 -1 18 -1 -1 -1 -1 44 19 -1 -1 -1 -1 -1  0
```

8.4.9.2.5.2 LDPC encoding

The code is flexible in that it can accommodate various code rates as well as packet sizes. ~~Since LDPCs are block-oriented codes, some restrictions are necessary on the combinations of available code rates and codeword sizes in order to control complexity.~~

The encoding of a packet at the transmitter generates parity-check bits $\mathbf{p}=(p_0, \ldots, p_{m-1})$ based on an information block $\mathbf{s}=(s_0, \ldots, s_{k-1})$, and transmits the parity-check bits along with the information block. Because the current symbol set to be encoded and transmitted is contained in the transmitted codeword, the information block is also known as systematic bits. The encoder receives the information block $\mathbf{s}=(s_0, \ldots, s_{k-1})$ and uses the matrix $\mathbf{H}_{bm}$ to determine the parity-check bits. The expanded matrix **H** is determined from the model matrix $\mathbf{H}_{bm}$. Since the expanded matrix **H** is a binary matrix, encoding of a packet can be performed with vector or matrix operations conducted over GF(2).

One method of encoding is to determine a generator matrix **G** from **H** such that $\mathbf{G}\,\mathbf{H}^T = \mathbf{0}$. A k-bit information block $\mathbf{s}_{1\times k}$ can be encoded by the code generator matrix $\mathbf{G}_{k\times n}$ via the operation $\mathbf{x} = \mathbf{s}\,\mathbf{G}$ to become an n-bit codeword $\mathbf{x}_{1\times n}$, with codeword $\mathbf{x}=[\mathbf{s}\ \mathbf{p}]=[s_0, s_1, \ldots, s_{k-1}, p_0, p_1, \ldots, p_{m-1}]$, where $p_0, \ldots p_{m-1}$ are the parity-check bits; and $s_0, \ldots s_{k-1}$ are the systematic bits.

Encoding an LDPC code from **G** can be quite complex. The LDPC codes are defined such that very low complexity encoding directly from **H** is possible.

Direct Encoding (Method 1)

Encoding is the process of determining the parity sequence **p** given an information sequence **s**. To encode, the information block **s** is divided into $k_b = n_b - m_b$ groups of z bits. Let this grouped **s** be denoted **u**,

$$\mathbf{u} = [\mathbf{u}(0) \quad \mathbf{u}(1) \quad \cdots \quad \mathbf{u}(k_b - 1)],$$

where each element of **u** is a column vector as follows

$$\mathbf{u}(i) = [s_{iz} \quad s_{iz+1} \quad \cdots \quad s_{(i+1)z-1}]^T$$

Using the model matrix $\mathbf{H}_{\text{bm}}$, the parity sequence **p** is determined in groups of z. Let the grouped parity sequence **p** by denoted **v**,

$$\mathbf{v} = [\mathbf{v}(0) \quad \mathbf{v}(1) \quad \cdots \quad \mathbf{v}(m_b - 1)],$$

where each element of **v** is a column vector as follows

$$\mathbf{v}(i) = [p_{iz} \quad p_{iz+1} \quad \cdots \quad p_{(i+1)z-1}]^T$$

Encoding proceeds in two steps, (a) initialization, which determines **v**(0), and (b) recursion, which determines **v**(i+1) from **v**(i), $0 \le i \le m_b - 2$.

An expression for **v**(0) can be derived by summing over the rows of $\mathbf{H}_{\text{bm}}$ to obtain

$$\mathbf{P}_{p(x,k_b)}\mathbf{v}(0) = \sum_{j=0}^{k_b-1}\sum_{i=0}^{m_b-1}\mathbf{P}_{p(i,j)}\mathbf{u}(j) \qquad (1)$$

where x , $1 \le x \le m_b - 2$, is the row index of $\mathbf{h}_{\text{bm}}$ where the entry is nonnegative and unpaired, and $\mathbf{P}_i$ represents the $z \times z$ identity matrix circularly right shifted by size i. Equation (1) is solved for **v**(0) by multiplying by $\mathbf{P}^{-1}_{p(x,k_b)}$, and $\mathbf{P}^{-1}_{p(x,k_b)} = \mathbf{P}_{z-p(x,k_b)}$ since $p(x,k_b)$ represents a circular shift.

The recursion expressed in Equation (2) can be derived by considering the structure of $\mathbf{H}'_{\text{b2}}$,

$$\mathbf{v}(i+1) = \mathbf{v}(i) + \sum_{j=0}^{k_b-1}\mathbf{P}_{p(i,j)}\mathbf{u}(j) + \mathbf{P}_{p(i,k_b)}\mathbf{v}(0), \quad i = 1,\ldots,m_b - 1 \qquad (2)$$

where

$$\mathbf{P}_{-1} \equiv \mathbf{0}_{z\times z}.$$

Thus all parity bits not in **v**(0) are determined by evaluating Equation (2) for $0 \le i \le m_b - 2$.

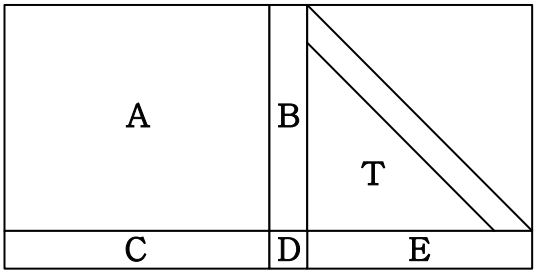
Equations (1) and (2) completely describe the encoding algorithm. These equations also have a straightforward interpretation in terms of standard digital logic architectures. Since the non-zero elements $p(i,j)$ of $\mathbf{H}_{\text{bm}}$ represent circular shift sizes of a vector, all products of the form $\mathbf{P}_{p(i,j)}\mathbf{u}(j)$ can be implemented by a size-z barrel shifter.

Direct Encoding (Method 2)

For efficient encoding of LDPC, **H** are divided into the form

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{T} \\ \boldsymbol{C} & \boldsymbol{D} & \boldsymbol{E} \end{pmatrix} \qquad (1)$$

where $\boldsymbol{A}$ is $(N_p - g)\times N_k$, $\boldsymbol{B}$ is $(N_p - g)\times g$, $\boldsymbol{T}$ is $(N_p - g)\times(N_p - g)$, $\boldsymbol{C}$ is $g \times N_k$, $\boldsymbol{D}$ is $g \times g$, and finally, $\boldsymbol{E}$ is $g\times(N_p - g)$. The basic structure of the **H** matrix is



Further, all these matrices are sparse and $\boldsymbol{T}$ is lower triangular with ones along the diagonal. ***B*** and ***D*** part have the column degree 3 and ***D*** has shift value of 1. ***B*** is with the first entry of 1 and shift value 0 in the middle of the column. This other entry is non-zero.

Let $\boldsymbol{v}=(\boldsymbol{u}, \boldsymbol{p}_1, \boldsymbol{p}_2)$ that $\boldsymbol{u}$ denotes the systematic part, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ combined denote the parity part, $\boldsymbol{p}_1$ has length g, and $\boldsymbol{p}_2$ has length $(N_p\text{-}g)$. The definition equation $\boldsymbol{H} \cdot \boldsymbol{v}^t = 0$ splits into two equations, as in equation 3 and 4 namely

$$\boldsymbol{A}\boldsymbol{u}^T + \boldsymbol{B}\boldsymbol{p}_1^T + \boldsymbol{T}\boldsymbol{p}_2^T = \boldsymbol{0} \qquad (2)$$

and

$$\left(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A} + \boldsymbol{C}\right)\boldsymbol{u}^T + \left(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{B} + \boldsymbol{D}\right)\boldsymbol{p}_1^T = \boldsymbol{0} \qquad (3)$$

Define $\phi := -\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{B} + \boldsymbol{D}$ and when we use the parity check matrix as indicated appendix we can get $\boldsymbol{\phi} = \boldsymbol{I}$. Then from (4) we conclude that

$$\boldsymbol{p}_1^T = \left(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A} + \boldsymbol{C}\right)\boldsymbol{u}^T \qquad (5)$$

and

$$\boldsymbol{p}_2^T = \boldsymbol{T}^{-1}\left(\boldsymbol{A}\boldsymbol{u}^T + \boldsymbol{B}\boldsymbol{p}_1^T\right). \qquad (6)$$

As a result, the encoding procedures and the corresponding operations can be summarized below and illustrated in Fig. 1.

Encoding procedure

Step 1) Compute $\boldsymbol{A}\boldsymbol{u}^T$ and $\boldsymbol{C}\boldsymbol{u}^T$.

Step 2) Compute $\boldsymbol{E}\boldsymbol{T}^{-1}\left(\boldsymbol{A}\boldsymbol{u}^T\right)$.

Step 3) Compute $\boldsymbol{p}_1^T$ by $\boldsymbol{p}_1^T = \boldsymbol{E}\boldsymbol{T}^{-1}\left(\boldsymbol{A}\boldsymbol{u}^T\right) + \boldsymbol{C}\boldsymbol{u}^T$.

Step 4) Compute $\boldsymbol{p}_2^T$ by $\boldsymbol{T}\boldsymbol{p}_2^T = \boldsymbol{A}\boldsymbol{u}^T + \boldsymbol{B}\boldsymbol{p}_1^T$.

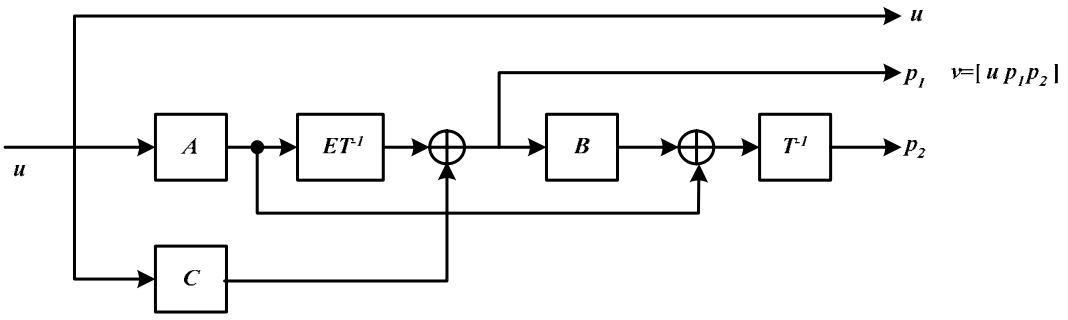


Fig. 2 Block diagram of the encoder architecture for the block LDPC code.

~~TBD description of packet encoding.~~

8.4.9.2.5.3 Code Rate and Block Size Adjustment

The code design will be flexible to support a range of code rates and block sizes through code rate and block size adjustment of the one or more H matrices of the fundamental code set. ~~The exact methods for supporting code rate and block size adjustment will depend on the final design.~~ For each supported rate and block size, ~~there will be~~ some combination of matrix selection, shortening, repetition, matrix expansion, and/or concatenation will be used.

Different block sizes and code rates are supported through using a variable *z* expansion factor. The *z* expansion factors for coded block sizes *n* corresponding to integer numbers of subchannels are provided below. In each case, the number of information bits is equal to the code rate times the coded block size *n*. In addition to matrix expansion, shortening is used and puncturing may be used to support some coded block sizes and code rates.

n (bits)	*n* (bytes)	*k* (bytes)			Number of subchannels		
		R=1/2	R=2/3	R=3/4	QPSK	16QAM	64QAM
576	72	36	48	54	6	3	2
672	84	42	56	63	7		
768	96	48	64	72	8	4	
864	108	54	72	81	9		3
960	120	60	80	90	10	5	
1056	132	66	88	99	11		
1152	144	72	96	108	12	6	4
1248	156	78	104	117	13		
1344	168	84	112	126	14	7	
1440	180	90	120	135	15		5
1536	192	96	128	144	16	8	
1632	204	102	136	153	17		
1728	216	108	144	162	18	9	6
1824	228	114	152	171	19		
1920	240	120	160	180	20	10	
2016	252	126	168	189	21		7
2112	264	132	176	198	22	11	
2208	276	138	184	207	23		
2304	288	144	192	216	24	12	8

Shortening may be applied to any expanded H matrix by reducing the number of subchannels available for the codeword. The number of bit corresponding to the reduced number of subchannels is equal to the number of shortened bits L. The matrix **H** is designed such that excellent performance is achieved under shortening, with different column weights interlaced between the first L columns of $\mathbf{H}_1$ and the rest of $\mathbf{H}_1$. Encoding with shortening is similar to encoding without shortening, except that the current symbol set has only k-L systematic bits in the information block, **s'**=(s_0, …, $s_{k\text{-}L\text{-}1}$). When encoding, the encoder first prepends L zeros to **s'** of length (k-L). Then the zero-padded information vector **s**=[$\mathbf{0}_L$ **s'**] is encoded using **H** as if unshortened to generate parity bit vector **p** (length m). After removing the prepended zeros, the code bit vector **x**=[**s' p**] is transmitted over the channel. This encoding procedure is equivalent to encoding **s'** using the last (n-L) columns of matrix **H** to determine the parity-check vector **p**.

The z expansion factors are determined by the target block size n and the base matrix size n_b. Examples of the z expansion factors are given in the tables below. The base matrix n_b is an integer is an integer multiple of 24.

Table

***n* (bits)**	***n* (bytes)**	**z expansion factor**			**Number of subchannels**		
		R=1/2	**R=2/3**	**R=3/4**	**QPSK**	**16QAM**	**64QAM**
96	12	2	2	2	1		
192	24	4	4	4	2	1	
288	36	6	6	6	3		1
384	48	8	8	8	4	2	
480	60	10	10	10	5		
576	72	12	12	12	6	3	2
672	84	14	14	14	7		
768	96	16	16	16	8	4	
864	108	18	18	18	9		3
960	120	20	20	20	10	5	
1056	132	22	22	22	11		
1152	144	24	24	24	12	6	4
1248	156	26	26	26	13		
1344	168	28	28	28	14	7	
1440	180	30	30	30	15		5
1536	192	32	32	32	16	8	
1632	204	34	34	34	17		
1728	216	36	36	36	18	9	6
1824	228	38	38	38	19		
1920	240	40	40	40	20	10	
2016	252	42	42	42	21		7
2112	264	44	44	44	22	11	

2208	276	46	46	46	23		
2304	288	48	48	48	24	12	8
base N-K		**24**	**16**	**12**			
base N		**48**	**48**	**48**			

~~TBD description of code adjustment.~~

8.4.9.2.5.4 Packet Encoding

~~After harmonization, this section will be replaced with something equivalent to what is in section 8.2.1.2.4.1 which is the concatenation/packet encoding scheme for the CTC.~~

~~Since transported data packets can be any size from typically about 40 bytes up to 12000 bits and larger, the system must be able to encode variable length packets in a consistent manner. This consistency is required to ensure that the receiver always knows how to reconstruct the information field from the encoded transmitted data.~~

The encoding block size k shall depend on the number of subchannels allocated and the modulation specified for the current transmission. Concatenation of a number of subchannels shall be performed in order to make larger blocks of coding where it is possible, with the limitation of not passing the largest block under the same coding rate (the block defined by the 64-QAM modulation). The table below specifies the concatenation of subchannels for different allocations and modulations. The concatenation rule follows the subchannel concatenation rule for CC (Table 315) except that for LDPC the concatenation does not depend on the code rate.

For any modulation and FEC rate, given an allocation of N_{sch} subchannels, we define the following parameters:

j parameter dependent on the modulation and FEC rate
N_{sch} number of allocated subchannels
F floor(N_{sch}/j)
M N_{sch} mod j

The subchannel concatenation rule for CC in Table 315 is applied, noting that in Table 315 the parameter n is equal to N_{sch}, the parameter k is equal to F, and the parameter m is equal to M. The parameter j for LDPC is determined as shown in the table below.

Modulation	*j*
QPSK	*j*=24
16-QAM	*j*=12
64-QAM	*j*=8

~~Each packet is encoded as an entity. In other words, the data boundary of a packet is respected by the encoder.~~ Control information and packets <u>that result in a codeword size *n* of less than 576 bits</u> ~~smaller than 40 bytes~~ are encoded using convolutional coding (CC) with appropriate code rates and modulation orders, as described in section 8.4.9.2.1.

~~The length and required rate of the packet that is to be encoded is all that is needed to encode or decode the packet using the following rules:~~

~~If Length <= N_i bits, then TBD.~~
~~If Length > N_i bits and <= 2 N_i bits, then TBD~~
~~If Length > 2 N_i bits, then compute N_r = modulo(Length, N_i) (in bits), then TBD.~~
~~Concatenation when TBD~~
~~Combination of shortening and concatenation when TBD~~

~~The intent of the above rule set is to provide a means for data transmission without the need for additional information beyond the packet field length. This scheme does so with a simple rule set that reduces the rate of the last codewords in order to reduce the number of iterations (and therefore the latency) that must be performed on the last portion of the data. The length and position of the shortened codewords and erased bits are deterministic when the above rules are followed.~~

~~For all packets the codeword bits can be indexed using the corresponding column indices of the H matrix. Using this convention the systematic codeword bits comprise the leftmost bits starting at bit location zero, and fill the codeword to bit k-1. The remaining N-k bits of the codeword, from indices k to N-1 are the parity bits. The codeword systematic bits are filled in an order consistent with the indices, so that the first bits of the packet fill the codeword from the lowest indices linearly to the highest indices. The codeword is then transmitted in a linear fashion starting from the lowest indices so that the systematic bits are transmitted first, followed by the parity bits. For shortened codewords the zeros are padded in the low order bits, so that the final codeword starting at the lowest indices contains first zero-padded bits and then the systematic data bits followed by the parity bits. The zero-padded bits are not typically sent over the channel.~~