Project	IEEE 802.16 Broadband Wireless Access Working Group <**http://ieee802.org/16**>
Title	LDPC coding for OFDMA PHY
Date Submitted	2004-11-04
Source(s)	Brian Classon brian.classon@motorola.com Yufei Blankenship yufei.blankenship@motorola.com Motorola
Re:	IEEE P802.16-REVe/D5-2004, sponsor ballot
Abstract	This contribution provides an updated version of LDPC code provided by Motorola in contribution 374.
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

This contribution provides an updated version of the Motorola code provided in contribution 374. This updated version of the Motorola code was provided to the informal LDPC group developing an LDPC code design for 802.16e. The code has been qualitatively and quantitatively characterized. It has excellent flexibility and performance, as well as low encoding and decoding complexity.

Features

The LDPC code has excellent performance, and contains 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.
- **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.
- **Compact representation**. The shift values for each block size are derived from the largest block size of that code rate, facilitating the representation and implementation of the encoder/decoder.
- **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. The code families have also been tested on fading channels with the other code proposals in the informal LDPC group, and were shown to have excellent performance.

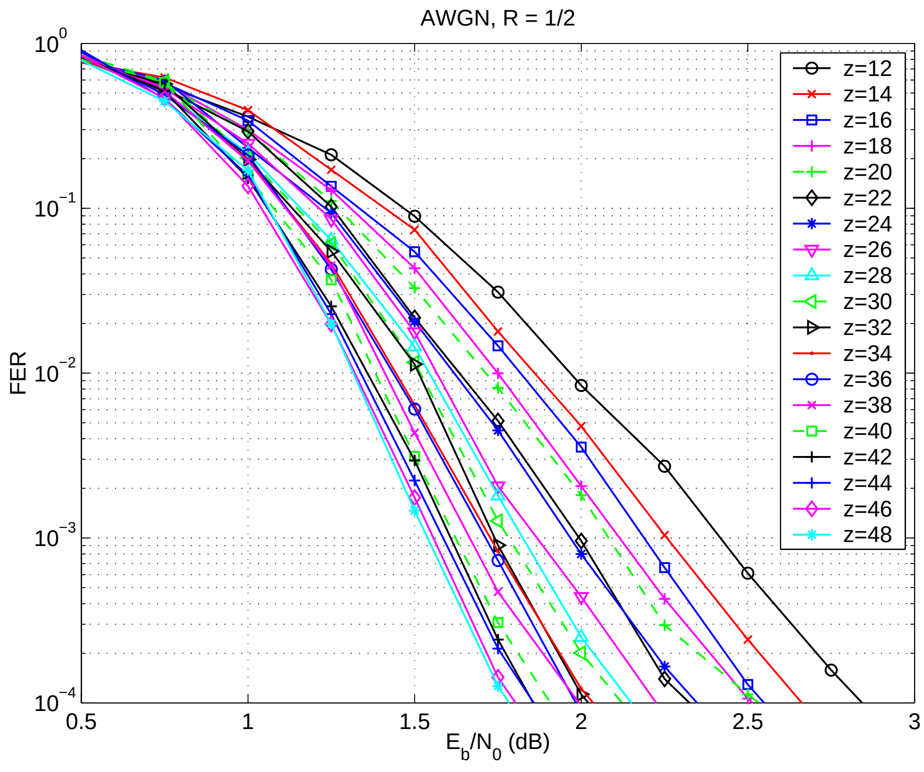

Figure 1. FER performance of 19 R = 1/2 structured codes. Base matrix size: $m_b = 24$, $n_b = 48$. AWGN, BPSK.

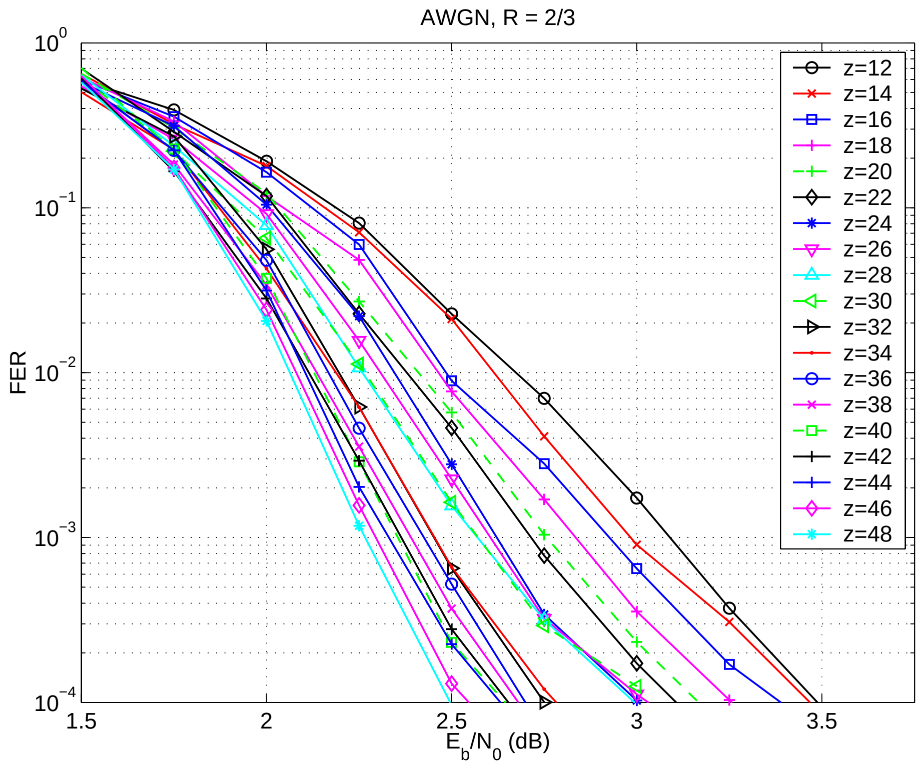

Figure 2. FER performance of 19 R = 2/3 structured codes. Base matrix size: $m_b = 16$, $n_b = 48$. AWGN, BPSK.

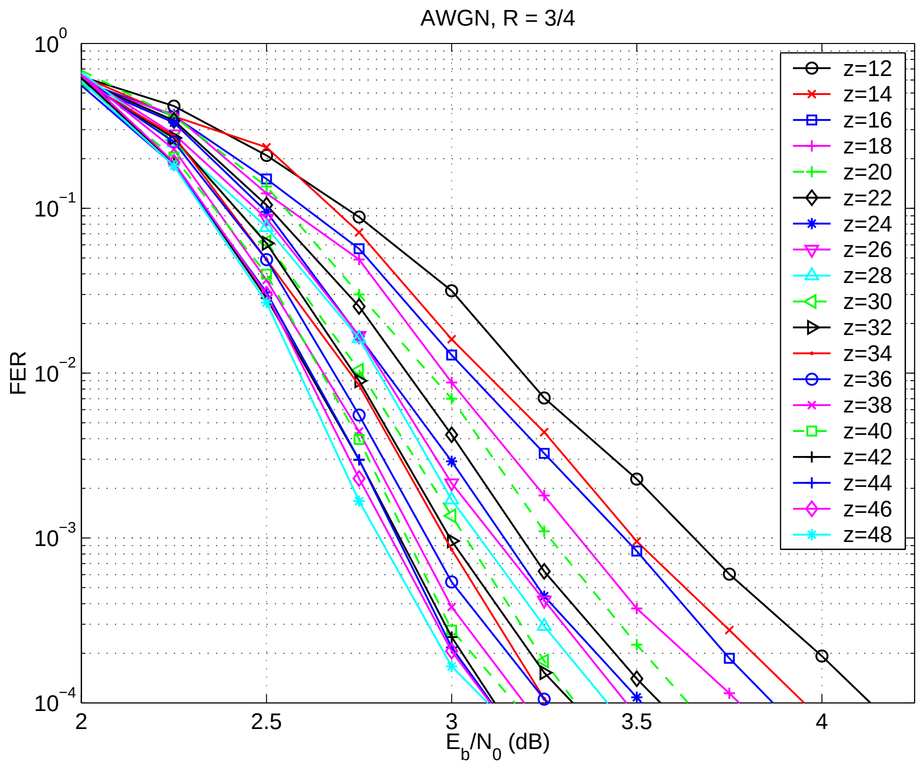

Figure 3. FER performance of 19 R = 3/4 structured codes. Base matrix size: $m_b = 12$, $n_b = 48$. AWGN, BPSK.

Recommended Text Changes:

Add/Modify the following text to 802.16e_D5, adjusting the numbering as required:

<Insert the following text into section 8.4.9.2.5.1 Code Description after the sentence "$\mathbf{H}_b$ 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:

$$\mathbf{H}_{b2} = \left[\mathbf{h}_b \;\vdots\; \mathbf{H}'_{b2} \right]$$

$$= \left[\begin{array}{c|ccccc} h_b(0) & 1 & & & & \\ h_b(1) & 1 & 1 & & \mathbf{0} & \\ . & & 1 & \ddots & & \\ . & & & \ddots & 1 & \\ . & & \mathbf{0} & & 1 & 1 \\ h_b(m_b-1) & & & & & 1 \end{array} \right].$$

The base matrix has $h_b(0)=1$, $h_b(m-1)=1$, and a third value $h_b(j)$, $0<j<(m_b-1)$ 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

The LDPC code design is based on a set of four basic model matrices, one for rate 1/2, one for rate 2/3, two for rate 3/4. The base matrix n_b equals to 48 for all four cases.
Each basic model matrix is designed for a shift size z_0. A set of shift sizes $\{p(i,j)\}$ is defined for the basic model matrix and used for other code sizes of the same rate. For other code sizes, the shift sizes are derived from the basic model matrix as follows [1]. 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,

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

Note that $\alpha_f = z_0/z_f$ and $[x]$ denotes rounding to the integer that differs from x the least. A rate 1/2 example is shown below on how to obtain a model matrix with z_f=36 from the basic model matrix with z_0=48.
The model matrices are tabulated below for each code rate. For brevity, the dual-diagonal portion corresponding to $\mathbf{H}'_{b2}$ is not shown.

Rate 1/2:
The basic model matrix has size n_b=48, m_b=24 and an expansion factor z_0=48 (i.e., n=48*48=2304). To achieve other code sizes, the expansion factor z_f can be any even integer between 12 and 48.

```
 3 -1 -1 22 14 -1 -1 26 -1 16 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1  7
-1  1 -1 -1 -1 39 20 -1 30 -1 -1 -1 -1 -1 -1 -1 28 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 28 -1 34  5 -1 -1 -1 -1 -1 -1 -1 -1 30 -1 -1 -1 -1 39 -1 -1 -1 -1 -1
43 -1 -1 40 -1 28 47 -1 -1 -1 -1 -1  5 -1 -1 -1 -1 -1 -1 -1 16 -1 -1 -1 -1
21 45 -1 25 -1 46 -1 -1 47 -1 -1 -1 -1 -1 -1 -1 -1 -1  8 -1 -1 -1 -1 -1 -1
-1  0  0 -1 41 -1 -1 11 -1 -1 38 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 46 -1
-1 -1 28 -1  0 37 -1 -1 -1 -1 -1 -1 -1 -1 -1  3 -1 18 -1 -1 -1 -1 -1 -1 -1
 0 -1 15 -1 -1 11 -1 -1 17 -1 -1 -1 29 -1 -1 -1 -1 -1 -1 -1 -1 -1 47 -1 -1
34  0 -1 -1 25 -1 47 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 -1 -1  5 -1 -1 -1
17 -1  0  4 -1 -1 39 -1 -1 -1 15 -1 -1 -1 -1 -1 -1 -1 -1  2 -1 -1 -1 -1 -1
-1 11 -1 -1 18 22 -1 -1 -1 -1 -1 -1 21 -1 -1 -1 28 -1 -1 -1 -1 -1 -1 -1 -1
-1 38 25 -1 38 14 -1 -1 -1 -1 -1 -1 -1 -1 -1 37 -1 -1 -1 -1 -1 -1 -1 -1  0
-1 -1 47 27 -1 -1 47 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 22 -1 -1 -1 -1
-1 -1 -1 37 38  2 -1 -1 -1 -1 -1 -1 -1 14 -1 -1 -1 20 -1 -1 -1 -1 -1 -1 -1
23 -1  0  0 -1 -1 -1 40 -1 -1 -1 -1 -1 -1 -1 -1  6 -1 -1 -1 -1 36 -1 -1 -1
30 -1 22 -1  0 -1 18 -1 -1 -1 -1 -1 -1 -1 -1 26 -1 -1 43 -1 -1 -1 -1 -1 -1
-1  0 29 40 -1 -1 35 -1 -1 -1 -1 -1 -1 -1  4 -1 -1 -1 -1 -1 28 -1 -1 -1 -1
30 18 -1 -1 -1  0  0 -1 -1 -1 -1 15 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 42 -1
14 -1 35 28 -1 -1  7 -1 -1 10 -1 -1 -1 -1 -1 -1 -1 -1 -1 41 -1 -1 -1 -1 -1
-1 14 -1 -1 34 -1 18 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 15 -1 -1 -1 -1 -1 -1
14 25 -1 31 -1 11 -1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 17 -1 -1
40 43 -1  0 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 40 -1 -1 19 -1 -1 -1 -1 -1 -1 -1
-1 16 -1 30  0 34 -1 -1 -1 33 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1
-1 -1 36 -1 37 -1 46 -1 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 -1 -1 24  7
```

As an example, to achieve z_f =36 (i.e., n=48*36=1728), the shift sizes of the basic model matrix above is scaled and the resulting model matrix is shown below.

```
 2 -1 -1 17 11 -1 -1 20 -1 12 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 24 -1  5
```

```
-1  1 -1 -1 -1 29 15 -1 23 -1 -1 -1 -1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 21 -1 26  4 -1 -1 -1 -1 -1 -1 -1 -1 23 -1 -1 -1 -1 29 -1 -1 -1 -1 -1
32 -1 -1 30 -1 21 35 -1 -1 -1 -1 -1  4 -1 -1 -1 -1 -1 -1 -1 12 -1 -1 -1 -1
16 34 -1 19 -1 35 -1 -1 35 -1 -1 -1 -1 -1 -1 -1 -1 -1  6 -1 -1 -1 -1 -1 -1
-1  0  0 -1 31 -1 -1  8 -1 -1 29 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 35 -1
-1 -1 21 -1  0 28 -1 -1 -1 -1 -1 -1 -1 -1 -1  2 -1 14 -1 -1 -1 -1 -1 -1 -1
 0 -1 11 -1 -1  8 -1 -1 13 -1 -1 -1 22 -1 -1 -1 -1 -1 -1 -1 -1 -1 35 -1 -1
26  0 -1 -1 19 -1 35 -1 -1 -1 -1 26 -1 -1 -1 -1 -1 -1 -1 -1 -1  4 -1 -1 -1
13 -1  0  3 -1 -1 29 -1 -1 -1 11 -1 -1 -1 -1 -1 -1 -1 -1  2 -1 -1 -1 -1 -1
-1  8 -1 -1 14 17 -1 -1 -1 -1 -1 -1 16 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1
-1 29 19 -1 29 11 -1 -1 -1 -1 -1 -1 -1 -1 -1 28 -1 -1 -1 -1 -1 -1 -1 -1  0
-1 -1 35 20 -1 -1 35 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 17 -1 -1 -1 -1
-1 -1 -1 28 29  2 -1 -1 -1 -1 -1 -1 -1 11 -1 -1 -1 15 -1 -1 -1 -1 -1 -1 -1
17 -1  0  0 -1 -1 -1 30 -1 -1 -1 -1 -1 -1 -1 -1  5 -1 -1 -1 -1 27 -1 -1 -1
23 -1 17 -1  0 -1 14 -1 -1 -1 -1 -1 -1 -1 -1 20 -1 -1 32 -1 -1 -1 -1 -1 -1
-1  0 22 30 -1 -1 26 -1 -1 -1 -1 -1 -1 -1  3 -1 -1 -1 -1 -1 21 -1 -1 -1 -1
23 14 -1 -1 -1  0  0 -1 -1 -1 -1 11 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1
11 -1 26 21 -1 -1  5 -1 -1  8 -1 -1 -1 -1 -1 -1 -1 -1 -1 31 -1 -1 -1 -1 -1
-1 11 -1 -1 26 -1 14 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 11 -1 -1 -1 -1 -1 -1
11 19 -1 23 -1  8 -1 -1 -1 -1 16 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 13 -1 -1
30 32 -1  0 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 30 -1 -1 14 -1 -1 -1 -1 -1 -1 -1
-1 12 -1 23  0 26 -1 -1 -1 25 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1
-1 -1 27 -1 28 -1 35 -1 -1 -1 -1 -1 -1 26 -1 -1 -1 -1 -1 -1 -1 -1 -1 18  5
```

Rate 2/3:

The basic model matrix has size n_b=48, m_b=16 and an expansion factor z_0=48 (i.e., n=48*48=2304). To achieve other code sizes, the expansion factor z_f can be any even integer between 12 and 48.

```
-1 -1 35 39 30  4 -1 -1 -1 -1 -1 28 25 -1 -1 -1 -1 -1 -1 -1 -1 -1 13 -1 -1 10 -1 -1 -1 -1 14 -1  7
40  1 -1 24  0 -1 -1 44 31 -1 11 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 43 -1 -1 -1 -1  4 -1 -1 -1
28 -1  0  0 15 47 -1  0 -1 -1 -1 -1 -1 -1 13 -1 -1 37 -1 -1 -1 -1 -1 -1 -1 16 -1 -1 18 -1 -1 -1 -1
-1 14 -1 15 47 35 -1 23 -1 -1 -1 -1 -1 -1 -1 13 -1 -1 -1 -1  1 -1 -1 24 -1 -1 -1 -1 -1 -1 -1 17 -1
35 28 38 -1 -1 19 -1 47 47 -1 -1 -1 -1 43 -1 -1 -1 18 -1 -1 -1 29 -1 -1 -1 -1 -1 -1 -1 -1 36 -1 -1
-1  7 -1  0 47 -1 43 -1 -1 13 -1 -1 -1 -1 -1 30 -1 -1 41 -1 -1 -1 -1 -1 -1  9 -1 -1 -1 40 -1 -1 -1
-1 14 30 -1 37 -1 -1 47 -1 -1 39 -1  0 -1 -1 -1 23 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 43 28 -1 -1 -1 -1
-1 -1 40 -1 14 16  0 34 -1 -1 -1 -1 -1 -1 17 11 -1 -1 -1 -1 -1 36 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0
39 15 36 -1 -1  0 21 -1 -1 11 -1 -1 41 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1 47 -1
30  0 -1 20 28 -1  0 -1 -1 -1 -1 32 -1 -1  0 -1 -1 -1 -1 -1 14 -1 -1 -1 -1 -1 39 -1 -1 27 -1 -1 -1
19 -1 35 43 -1 24 30 47 -1 -1 -1 -1 -1 23 -1 -1 -1 -1 -1 -1 -1 -1  5 26 -1 -1 -1  1 -1 -1 -1 -1 -1
22 -1 36 -1 -1 41  9 34 -1 -1 -1 -1 -1 -1 -1 -1 46 -1 -1 -1 33 -1 -1 -1 12 -1 -1 -1 -1 -1 47 -1 -1
 0  0 -1 43 -1  0 47 -1 -1 27 -1 -1 -1 18 -1 -1 -1 -1 -1 27 -1 -1 -1 -1 -1 -1 28 -1  8 -1 -1 -1 -1
10 -1 34 18 22 -1 36 -1 36 -1 -1 -1 -1 -1 -1 -1 14 -1  0 -1 -1 -1 33 -1 -1 -1 -1 -1 -1 -1 -1 32 -1
44 42 35 -1 -1 22 12 19 -1 -1 -1 10 -1 -1 -1 -1 -1 -1 -1 12 -1 11 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1
-1 41 -1 37 34 -1 36  0 -1 -1  7 -1 -1 -1 -1 -1 -1 -1 20 -1 -1 -1 -1 46 -1 -1 23 -1 -1 -1 -1 -1  7
```

Rate 3/4:

The base matrix has size n_b=48, m_b=12. The same base matrix is shared by two basic model matrices. The first basic model matrix has z_0=48 (i.e., n=48*48=2304), with expansion factor z_f being any even integer between 24 and 48 to achieve other code sizes. The first basic model matrix is shown below.

```
 5 19 35  3 24 -1  1 21 44 -1 -1 -1 -1 -1 -1 10 -1 -1 -1 40 -1 -1 -1 -1 41 -1 -1 12 -1 18 -1 -1 24 -1 -1 -1  7
23 -1  0 47 47 44 37  0  5 -1 -1 -1 -1 45 -1 -1 -1 44 -1 -1 23 -1 -1 -1 -1 -1 35 -1  1 -1 -1 -1 -1 -1 -1 19 -1
29 47 -1 23 27 36 47 33 -1 -1 -1 44 -1 -1 14 -1 -1 -1 47 -1 43 -1 -1 -1 -1 19 -1 -1 -1  4 -1 -1 -1 46 -1 -1 -1
12 20 30 34 -1 18 38 40 -1 -1 -1 -1 19 -1 -1 36 34 -1 -1 -1 -1  1 -1 -1 -1 -1 47 -1 -1 -1 -1 -1 30 -1  0 -1 -1
-1 34 21 30  0 37 -1 16 -1 -1 -1 13 -1 -1 -1 -1 42 -1 -1  0 -1 -1  1 32 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 47 -1
-1  0 32  3  8 31 -1 47 -1 -1 -1 31 -1 15 -1 -1 -1 47 -1 -1 -1 24 -1 -1 -1 -1 -1 -1 24 -1 16 -1 -1 -1 -1 -1  0
26 17  0 37 -1 27 12 27 -1 -1 24 -1 -1 -1 22 -1 -1 38 -1 -1 -1 -1 -1 35 -1 -1 -1  0 -1 -1 -1 46 24 -1 -1 -1 -1
13 -1  8 -1 22  6 21 47 -1 -1 47 -1 -1  1 -1 -1 -1 -1 47 -1 -1 -1 11 -1 -1 10 -1 -1 -1 -1 43 -1 -1 -1 24 -1 -1
15 25  0 32 18 19 47 -1 -1 -1 37 -1  9 -1 -1 -1 -1 -1 18 -1 -1 31 -1 -1 19 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 33 -1
47 26 38  0 38 -1 41 47 -1  8 -1 -1 17 -1 -1 -1 -1 -1 -1 23  0 -1 -1 -1 -1 -1 -1 25 -1 -1 28 -1 -1  0 -1 -1 -1
40 24 -1 17 19 24 47 -1 46 22 -1 -1 -1 -1 -1 30 -1 -1 -1 -1 -1 -1 -1 38 47  1 -1 -1 24 -1 -1 -1 -1 -1 39 -1 -1
31 34 32 -1  0 40 11 34 -1  0 -1 -1 -1 -1  5 -1 22 -1 -1 -1 -1 -1 14 -1 -1 -1  2 -1 -1 -1 -1 39 -1 24 -1 -1  7
```

The second basic model matrix has z_0=22 (i.e., n=48*22=1056), with the expansion factor z_f being any even integer between 12 and 22 to achieve other code sizes. The second basic model matrix is shown below.

```
 9  3  6 14 11 -1 17  7 18 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 21 -1 -1 -1 -1  0 -1 -1 15 -1  8 -1 -1  2 -1 -1 -1  7
10 -1 11 11  0 10 21  0  4 -1 -1 -1 -1 21 -1 -1 -1  7 -1 -1  9 -1 -1 -1 -1 -1 15 -1 13 -1 -1 -1 -1 -1 -1 21 -1
17  4 -1 10  8 14 12 17 -1 -1 -1 14 -1 -1 11 -1 -1 -1 18 -1 18 -1 -1 -1 -1 21 -1 -1 -1 13 -1 -1 -1 19 -1 -1 -1
 3 17 10 14 -1  0  5  0 -1 -1 -1 -1 14 -1 -1  0 10 -1 -1 -1 -1  0 -1 -1 -1 -1  8 -1 -1 -1 -1 -1 21 -1 16 -1 -1
```

```
-1 10 19  3  7  1 -1  4 -1 -1 -1  7 -1 -1 -1 -1 19 -1 -1 11 -1 -1  3 15 -1 -1 -1 -1 -1  9 -1 -1 -1 -1 -1  7 -1
-1  9 10 11  1  5 -1  8 -1 -1 -1 14 -1 11 -1 -1 -1 12 -1 -1 -1  3 -1 -1 -1 -1 -1 -1 15 -1 11 -1 -1 -1 -1 -1  0
18 11 17 10 -1  0  7  4 -1 -1  6 -1 -1 -1  7 -1 -1  5 -1 -1 -1 -1 -1  1 -1 -1 -1 20 -1 -1 -1  9 12 -1 -1 -1 -1
16 -1  3 -1 21  4  3  5 -1 -1 18 -1 -1 17 -1 -1 -1 -1  8 -1 -1 -1 14 -1 -1 14 -1 -1 -1 -1  3 -1 -1 -1  0 -1 -1
18 14 19  1  8 11  6 -1 -1 -1  0 -1 19 -1 -1 -1 -1 -1 20 -1 -1 14 -1 -1  0 -1 -1 -1 -1 -1 -1 17 -1 -1 -1 11 -1
19  5 10  0  7 -1  7 17 -1  0 -1 -1  3 -1 -1 -1 -1 -1 -1 19  1 -1 -1 -1 -1 -1 -1  0 -1 -1 10 -1 -1 13 -1 -1 -1
21  9 -1  3 15 11  4 -1 11 10 -1 -1 -1 -1 -1 10 -1 -1 -1 -1 -1 -1 -1  0  2 17 -1 -1  8 -1 -1 -1 -1 -1 13 -1 -1
13 19  8 -1  1  9  3 18 -1  7 -1 -1 -1 -1 13 -1  2 -1 -1 -1 -1 -1  3 -1 -1 -1  9 -1 -1 -1 -1  9 -1  8 -1 -1  7
```

<Insert the following text at the end of section 8.4.9.2.5.2 LDPC encoding>

Direct Encoding

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 $\mathbf{P}_{p(x,k_b)}^{-1}$, and $\mathbf{P}_{p(x,k_b)}^{-1} = \mathbf{P}_{z-p(x,k_b)}$ since $p(x,k_b)$ represents a circular shift.
Considering the structure of $\mathbf{H}'_{b2}$, the recursion can be derived as follows,

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

$$\mathbf{v}(i+1) = \mathbf{v}(i) + \sum_{j=0}^{k_b-1}\mathbf{P}_{p(i,j)}\mathbf{u}(j) + \mathbf{P}_{p(i,k_b)}\mathbf{v}(0), \quad i = 1,\ldots,m_b - 2\,, \qquad (3)$$

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.