Project	**IEEE 802.16 Broadband Wireless Access Working Group <http://ieee802.org/16>**
Title	**Low-complexity LDPC coding for OFDMA PHY**
Date Submitted	**2004-05-15**
Source(s)	Brian Classon, Yufei Blankenship Motorola 847-576-5675 Brian.Classon@Motorola.com
Re:	802.16e D2
Abstract	Parity check matrix suggestions for R=1/2, small block sizes, and incremental redundancy
Purpose	Enable IR for optional LDPC mode in 802.16e
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>.

Low-complexity LDPC coding for OFDMA PHY

Brian Classon, Yufei Blankenship
Motorola

Overview

Low-density parity-check (LDPC) codes offer many advantages for mobile subscriber stations, such as simple implementation and lower battery consumption, that the coding modes currently defined in 802.16d cannot provide. A primary concern with LDPC codes is encoding complexity, which can be addressed by using systematic LDPC codes with recursively generated parity check bits (i.e., parity check bits that can be produced using a differential encoder). The Intel LDPC contribution to 802.16d [1] has this desirable feature, but suffers a performance penalty for the R=1/2 code. The low rate code is very important for achieving the good coverage and system throughput. Another contribution [2] makes up the performance difference for the R=1/2 code by using Richardson's encoding method and different code structure for different code rates. The encoding is more expensive than [1], and the different code rates are not-rate compatible.

In this contribution, modifications to the Intel proposal are considered that use a low-complexity differential encoding structure with different **H** matrices in order to improve performance. Performance is improved by not having weight-1 columns in **H**, which can cause additional errors and error floors during decoder implementation. With the modification, the same performance as [2] can be achieved for the R=1/2 code. An incremental redundancy procedure and evidence that LDPC can be used for small block sizes are also provided using the same low-complexity differential encoding structure.

Parity Check Matrix Construction

An LDPC code is specified by a parity-check matrix **H**. A k-bit information block $\mathbf{s}_{1\times k}$ is encoded to become an n-bit codeword $\mathbf{x}_{1\times n}$, and the code rate is $r = k/n$. The parity-check matrix is m-by-n, where $m = n - k$. The codeword **x** satisfies

$$\mathbf{H}\mathbf{x}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}}, \qquad (1)$$

where **0** is a size-m row vector of all zeros.

H can be regular (same number of 1's per row/column) or irregular. Carefully designed irregular LDPC codes have been shown to perform extremely close to the channel capacity, exhibiting a better error-correcting performance than turbo codes and regular LDPC codes. For irregular LDPC codes, row/column weights of the parity-check matrix **H** may not be uniform. During iterative decoding, codeword bits of higher degree (corresponding to columns with higher weight) have lower error probability and are corrected earlier. The iterative decoder then passes more reliable extrinsic information to the codeword bits of lower degree (corresponding to columns with lower weight). In order to keep the density (and complexity) low, the optimal weight distributions reported in the literature generally include a large percentage of weight-2 columns.

To construct an LDPC code that encodes easily and maintains good performance when puncturing, some structure has to be introduced to the **H** matrix. Assuming $\mathbf{x} = [\mathbf{s}\ \mathbf{p}] = [s_0, s_1, \ldots, s_{k-1}, p_0, p_1, \ldots, p_{m-1}]$, the **H** matrix can be divided into two submatrices,

$$\mathbf{H} = [\mathbf{H}_1 \quad \mathbf{H}_2], \qquad (2)$$

where $\mathbf{H}_2$ has a deterministic structure, and $\mathbf{H}_1$ can be any binary matrix of size m-by-k. The deterministic section $\mathbf{H}_2$ is composed of two parts. The first column $\mathbf{h}$ having an odd weight greater than 2, and rest of the columns (denoted by the m-by-(m-1) matrix $\mathbf{H}'_2$) are weight-2 with maximum of 1 overlap between each other. In a formula, $\mathbf{H}'_2$ has matrix elements equal to a 1 for $i=j$ and a 1 for $i=j+1$, and a 0 elsewhere, for row i, column j of $\mathbf{H}'_2$, $0\le i\le m-1$, $0\le j\le m-2$. In a picture,

$$\mathbf{H}_2 = [\mathbf{h} \quad \mathbf{H}'_2]$$
$$= \left[\begin{array}{c:cccccccccc} h_0 & 1 & & & & & & & & & \\ h_1 & 1 & 1 & & & & & & & & \\ . & & 1 & 1 & & \mathbf{0} & & & & & \\ . & & & 1 & 1 & & & & & & \\ . & & & & 1 & 1 & & & & & \\ . & & & & & 1 & 1 & & & & \\ . & & \mathbf{0} & & & & 1 & 1 & & & \\ . & & & & & & & 1 & 1 & & \\ . & & & & & & & & \ddots & \ddots & \\ & & & & & & & & & 1 & 1 \\ . & & & & & & & & & & 1 \quad 1 \\ h_{m-1} & & & & & & & & & & 1 \end{array}\right]. \quad (3)$$

The $\mathbf{h}$ used in this note is $\mathbf{h} = [1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ \ldots\ 0]^{\mathrm{T}}$.

Unlike [1], the $\mathbf{H}_2$ matrix is chosen to avoid any weight-1 column, which can hurt performance since a bit corresponds to a weight-1 column does not update the soft information during iterative decoding. With a weight-1 column at higher SNR, almost all frame errors with a small number of bit errors are caused by the weight-1 column.

Encoding Method

The $\mathbf{H}_2$ structure exemplified in (3) makes encoding very simple. Given any block of information bits $\mathbf{s}$, the m parity-check bits can be found by solving the equations defined by

$$\left[(\mathbf{H}_1)_{m\times k} \quad \mathbf{h}_{m\times 1} \quad (\mathbf{H}'_2)_{m\times(m-1)}\right]\begin{bmatrix}\mathbf{s}^{\mathrm{T}} \\ \mathbf{p}^{\mathrm{T}}\end{bmatrix} = 0. \quad (4)$$

Due to the odd column weight of $\boldsymbol{h}$ and the m-1 weight-2 columns of $\mathbf{H}'_2$, the summation of all the equation in (4) yields

$$p_0 = \left(\sum_{\text{row}} \mathbf{H}_1\right)\mathbf{s}^{\mathrm{T}}, \quad (5)$$

where $\sum_{\text{row}} \mathbf{H}_1$ denotes the row vector after summing up all rows of $\mathbf{H}_1$. Note that the summation can equivalently be performed on the intermediate column vector $v = [v_0, v_1, \ldots, v_{m-1}]^{\mathrm{T}} = \mathbf{H}_1\,\mathbf{s}^{\mathrm{T}}$, but performing the summation on the rows of $\mathbf{H}_1$ can be done beforehand and results in fewer operations to compute p_0.
With p_0 determined, the rest of parity check bits, p_1 through p_{m-1} can be found recursively. For example,

$$\begin{aligned}
p_1 &= h_0 p_0 + v_0, \\
p_2 &= h_1 p_0 + p_1 + v_1, \\
p_3 &= h_2 p_0 + p_2 + v_2, \\
&\cdots \\
p_{m-1} &= h_{m-2} p_0 + p_{m-2} + v_{m-2}.,
\end{aligned} \tag{6}$$

where $[h_0, h_1, \ldots, h_{m-2}, h_{m-1}]^T$ is the column **h**, equal to $[1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ \ldots\ 0]^T$ in (3). Since all the variables are binary in (6), the encoding complexity is very low. Further, to simplify implementation, a vector $w = [h_0\ p_0{+}v_0,\ h_1\ p_0{+}v_1,\ \ldots,\ h_{m-2}\ p_0{+}v_{m-2}]$ can be stored so that the parity bits are found by $p_1 = w_1$, $p_i = p_{i-1} + w_{i-1}$, $1 \le i \le m-1$.
The parity check equations can also be recursively solved from p_{m-1} through p_1.

Rate-Compatible Code Modification

The **H** matrix in (3) can be modified to create rate-compatible LDPC codes by concatenating portions similar to $\mathbf{H}_2$. Because $\mathbf{H}_2$ does not have weight-1 columns, the resulting codes do not have multiple weight-1 columns. For example, consider three codes, where the superscript indicates code 1, code 2 and code 3, respectively. The parity bits of code 1 are a subset of the parity bits of code 2, and the parity bits of code 2 are a subset of the parity bits of code 3. Code 1 has a parity-check matrix $\mathbf{H}^{(1)}$ defined by

$$\mathbf{H}^{(1)} = \left\lfloor \mathbf{H}_1^{(1)} \quad \mathbf{H}_2^{(1)} \right\rfloor, \tag{7}$$

where $\mathbf{H}_1^{(1)}$ is an m_1-by-k matrix and $\mathbf{H}_2^{(1)}$ is an m_1-by-m_1 matrix whose structure can follow (3), and produces parity bits $p_0, p_1, \ldots, p_{m_1-1}$.
Code 2 has a parity-check matrix $\mathbf{H}^{(2)}$ defined by

$$\mathbf{H}^{(2)} = \left[\begin{array}{cc:c} \mathbf{H}_1^{(1)} & \mathbf{H}_2^{(1)} & \mathbf{0} \\ \hdashline \multicolumn{2}{c:}{\mathbf{H}_1^{(2)}} & \mathbf{H}_2^{(2)} \end{array}\right], \tag{8}$$

where $\mathbf{H}_1^{(2)}$ is an $(m_2{-}m_1)$-by-$(k{+}m_1)$ matrix and $\mathbf{H}_2^{(2)}$ is an $(m_2{-}m_1)$-by-$(m_2{-}m_1)$ matrix whose structure can follow (3), and produces parity bits $p_0, p_1, \ldots, p_{m_2-1}$. Due to the nesting structure, parity bits $p_{m_1}, \ldots, p_{m_2-1}$ can be obtained from $\mathbf{H}_1^{(2)}$, $\mathbf{H}_2^{(2)}$, the systematic bits **s**, and the previously computed parity bits $p_0, p_1, \ldots, p_{m_1-1}$ using the encoding techniques of equations (5) and (6).
Code 3 has a parity-check matrix $\mathbf{H}^{(3)}$ defined by

$$\mathbf{H}^{(3)} = \left[\begin{array}{ccc:c} \mathbf{H}_1^{(1)} & \mathbf{H}_2^{(1)} & \mathbf{0} & \\ \multicolumn{2}{c}{\mathbf{H}_1^{(2)}} & \mathbf{H}_2^{(2)} & \mathbf{0} \\ \hdashline \multicolumn{3}{c:}{\mathbf{H}_1^{(3)}} & \mathbf{H}_2^{(3)} \end{array}\right], \tag{9}$$

where $\mathbf{H}_1^{(3)}$ is an $(m_3{-}m_2)$-by-$(k\ {+}m_2)$ matrix and $\mathbf{H}_2^{(3)}$ is an $(m_3{-}m_2)$-by-$(m_3{-}m_2)$ matrix whose structure can follow (3), and produces parity bits $p_0, p_1, \ldots, p_{m_3-1}$. Due to the nesting structure, parity bits $p_{m_2}, \ldots, p_{m_3-1}$ can be obtained from $\mathbf{H}_1^{(3)}$, $\mathbf{H}_2^{(3)}$, the systematic bits **s**, and the previously computed parity bits $p_0, p_1, \ldots, p_{m_2-1}$ using the encoding techniques of equations (5) and (6).
The "mother code" defined in (9) effectively defines codes of three different rates: $k/(k{+}m_1)$, $k/(k{+}m_2)$, and $k/(k{+}m_3)$. The three pieces ([**s**, $p_0, p_1, \ldots p_{m_1-1}$], [**s**, $p_0, p_1, \ldots p_{m_2-1}$], and [**s**, $p_0, p_1, \ldots p_{m_3-1}$]) could be used for

different code rates in a system, and because they are rate-compatible, can be used for IR. In IR, a first transmission may send [$\mathbf{s}$, p_0, p_1, … p_{m_1-1}], a second transmission may send [p_{m_1}, …, p_{m_2-1}] (if the first transmission was not received correctly), and a third transmission (if the second transmission was not received correctly) may send [p_{m_2}, …, p_{m_3-1}].

Note that the IR procedure described above in equations (7)(8)(9) could be applied to other H matrix constructions, but at a performance penalty (multiple wt 1 columns in [1]) or much more complex encoding using a layered Richardson approach.

Simulation results

This section presents several example LDPC codes built based on this contribution in comparison to other code types. The simulation is run with floating-point, AWGN channel, and BPSK modulation. The stopping rule is used for LDPC codes such that the iteration is stopped when $\mathbf{Hx}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}}$, or a maximum of 50 iterations is reached.

R=1/2 code

A (800, 400) LDPC code with the proposed structure is compared with another (800, 400) LDPC code which is obtained by shortening the (2000, 1600) code of [1] by 1200 bits. The simulation curves in Figure 1 shows that 0.5 dB performance gain achieved at FER = 10^{-2}, which is approximately the same as achieved by [2].

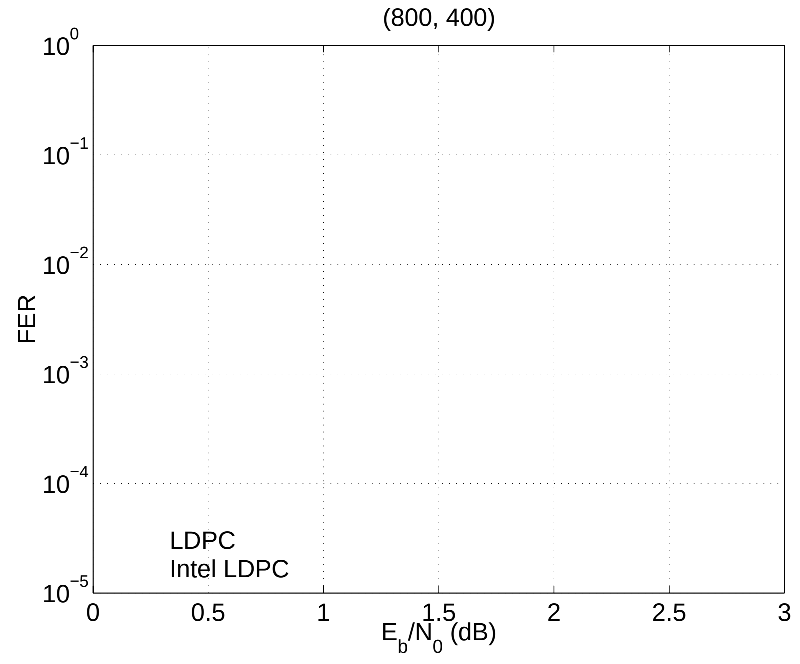

Figure 1. FER performance comparison of two LDPC codes, each having size (800, 400). The circled line is built based on this contribution with $\mathbf{h}$=[1 0 0 0 1 0 0 0 1 0 … 0]$^{\mathrm{T}}$; the crossed line is shortened from the Intel (2000, 1600) LDPC code by 1200 bits.

IR Example

To illustrate the performance of the IR procedure proposed in this contribution, an example is simulated where the first transmission uses a (384, 288) LDPC code. The second transmission either transmits 192 extra parity bits with the IR structure in (8), or repeat 192 bits of the original (384, 288) code (i.e., partial Chase combining). The FER performance comparison is shown in Figure 2. For this example, the IR procedure proposed in this contribution brings about 1.4 dB over partial Chase combining at FER = 10^{-2}.

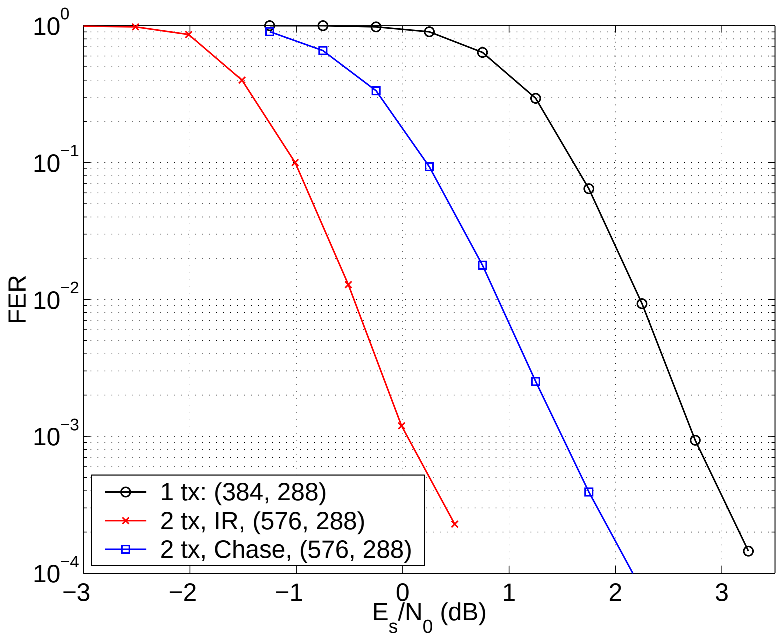

Figure 2. FER performance comparison of (a) a first transmission of (384, 288) LDPC code; (b) after a second transmission of 192 extra parity bits with the IR procedure proposed in this contribution; (c) after a second transmission of repeating 192 code bits (i.e., a partial Chase combining).

Smaller block sizes

To show that LDPC codes can achieve performance comparable to tail-biting convolutional codes even when the code size is small, a (192, 96) LDPC code is compared to a 64-state tail-biting convolutional code of the same code size. The simulation curves in Figure 3 shows that the LDPC code is slightly worse at low SNR region, but has a steeper slope and performs better at high SNR region.

When the code size is further reduced, the performance of LDPC code may be worse than the tailbiting convolutional code. As illustrated in Figure 4, when the code size (96, 48), the LDPC code is about 0.8 dB worse than the 64-state tail-biting convolutional code at FER = 10^{-2}.

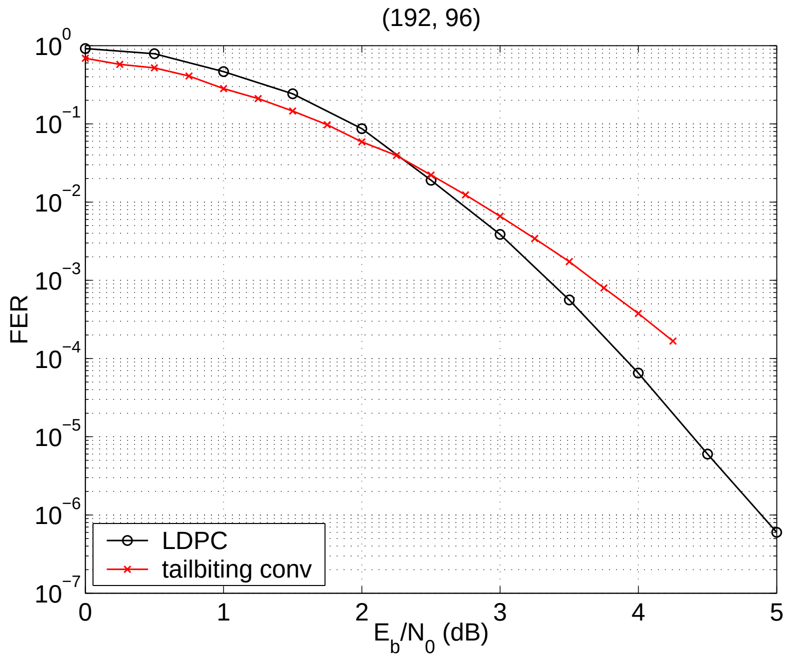

Figure 3. FER performance comparison of a LDPC code and a tail-biting convolutional code, each having size (192, 96).

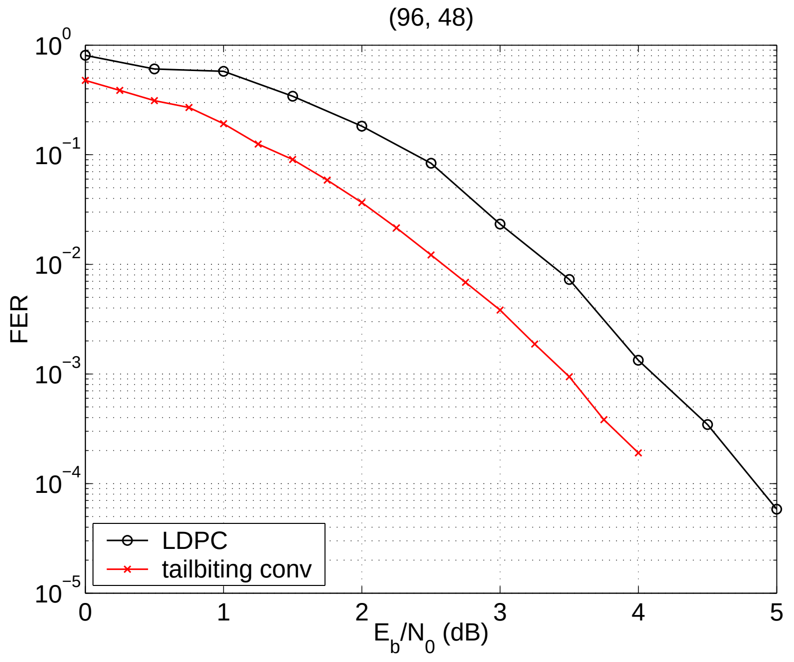

Figure 4. FER performance comparison of a LDPC code and a tail-biting convolutional code, each having size (96, 48).

References

[1] IEEE C802.16d-04/82, E. Jacobsen, "Draft text for LDPC coding scheme for OFDMA," April 26, 2004.
[2] IEEE C802.16d-04/86r1, P. Joo, "BLDPC coding for OFDMA PHY," April 30, 2004.