Project	IEEE 802.16 Broadband Wireless Access Working Group <**http://ieee802.org/16**>
Title	LDPC coding for OFDMA PHY
Date Submitted	2004-08-17
Source(s)	Brian Classon brian.classon@motorola.com Yufei Blankenship yufei.blankenship@motorola.com Motorola Jerry Kim kimjy@samsung.com Gyubum Kyung DS Parka Samsung Eric Jacobsen eric.a.jacobsen@intel.com Bo Xia Intel
Re:	IEEE P802.16-REVe/D4-2004, ballot #14c
Abstract	This contribution provides text for an LDPC code with excellent flexibility and performance, as well as low encoding and decoding complexity. Note that the informal LDPC group intends to submit a harmonized reply comment on August 24th.
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

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. This contribution provides specification text for three of the proposals that share a large amount of commonality (Intel, Motorola, Samsung). In addition, sample simulation results are provided from the Motorola proposal. The informal LDPC group intends to submit a harmonized reply comment on 24 August.

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 $\underline{n}$ = 576, 1152, 1728, 2304. The simulation conditions are: AWGN channel, BPSK modulation, maximum of 50 iterations using generic floating-point belief propagation.

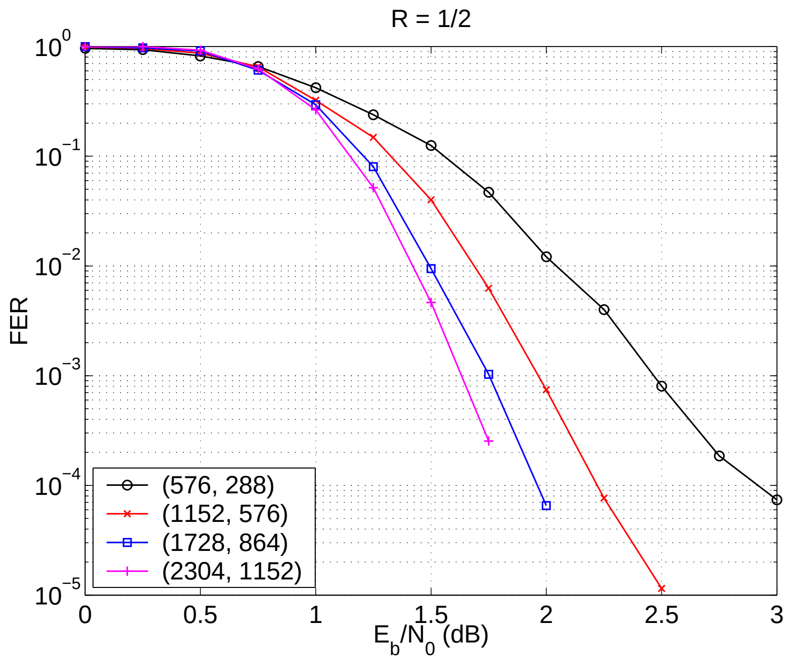


Figure 1. FER performance of four R = 1/2 structured codes. Base matrix size: m_b = 12, n_b = 24. AWGN, BPSK.

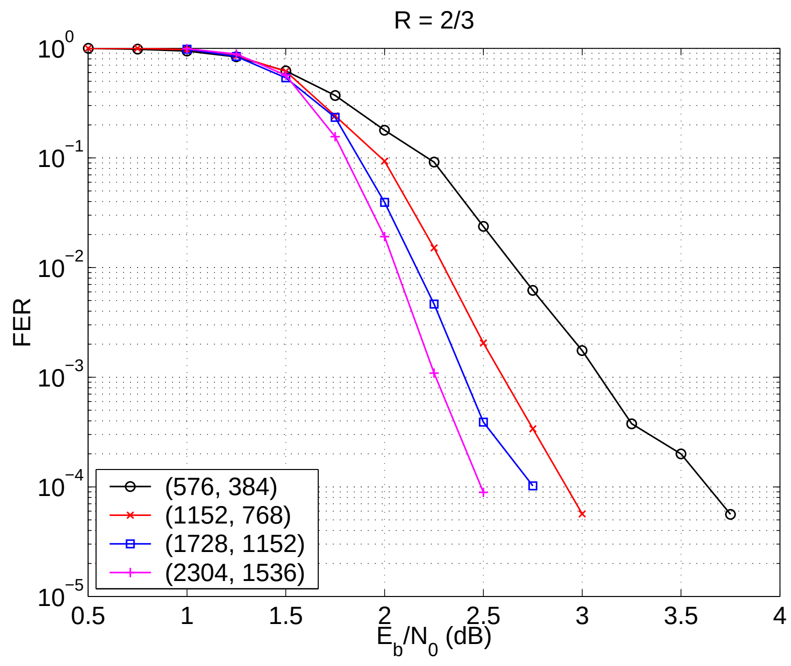


Figure 2. FER performance of four R = 2/3 structured codes. Base matrix size: m_b = 16, n_b = 48. AWGN, BPSK.

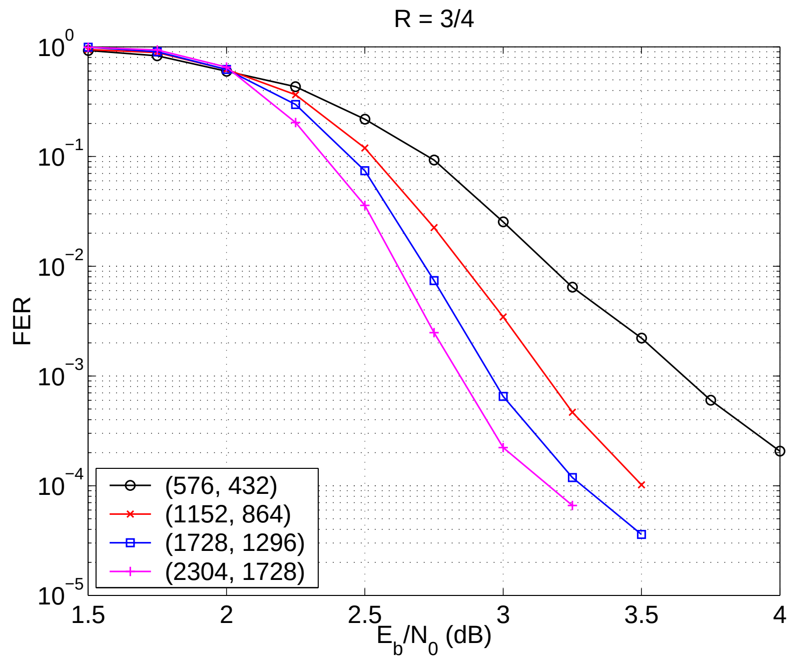


Figure 3. FER performance of four R = 3/4 structured codes. Base matrix size: m_b = 9, n_b = 36. AWGN, BPSK.

Recommended Text Changes:

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

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 8.4.9.2.4.3 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 expanded from a 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.

$\mathbf{H}_b$ 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:

$$\mathbf{H}_{b2} = \left[\mathbf{h}_b \;\vdots\; \mathbf{H}'_{b2} \right]$$
$$= \left[\begin{array}{c|ccccc} h_b(0) & 1 & & & & \\ h_b(1) & 1 & 1 & & \mathbf{0} & \\ . & & 1 & \ddots & & \\ . & & & \ddots & 1 & \\ . & & \mathbf{0} & & 1 & 1 \\ h_b(m-1) & & & & & 1 \end{array} \right].$$

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.

An expanded matrix **H** is created from a base matrix by replacing each 1 in the base matrix with a *z*-by-*z* non-zero submatrix, and each 0 with a *z*-by-*z* zero 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.

For compact representation, each binary entry (i,j) of the base matrix $\mathbf{H}_b$ is replaced to create an m_b-by-n_b 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 $z\times z$ all-zero matrix, and each 1 in $\mathbf{H}_b$ is replaced by a circular shift size $p(i,j)\geq 0$.

Model Matrix Set 1 (Samsung)

Four different model matrices are defined, denoted M1, M2, M3, and M4. The rightmost m_b-by-m_b section of each model matrix is $\mathbf{H}_{bm2}$.

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

Mat 1. M1 mother code with the original code rate 1/2

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

Mat 2. M2 mother code with the original code rate 2/3

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

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

Mat 3. M3 mother code with the original code rate 3/4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	0									0	0	0	0	0							0	0	
1			0	0							21					0	0					2	1	0
2	36				0	0					34							0	0	0		25	9	3
3			2				0	0			10	38	21								0		14	39
4		30							0	0		40			39	1			19			22		2
5			24				12				7	24		9						27			31	15
6		14				18					34	39					20	40				3	8	36
7				35	39					7	24	38	22			10						27		1
8	24							18			37	11			9						35	7	30	21
9						17			17		15	21		30				24				19	13	16
10				4				34		30	24	39							2	27		40	23	
11					9		18		0			13					27				15	20	12	33

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

Mat 4. M4 mother code with the original code rate 3/4

Model Matrix Set 2 (Intel)

One model matrix is defined for R=3/4, with the model matrices for R=2/3 and R=1/2 defined by shortening. The rightmost m_b-by-m_b section of the model matrix H_p is $\mathbf{H}_{\text{bm2}}$.

The rate-¾ regular mother code is described as

$$H_{3/4} = \lfloor H_3 \mid H_2 \mid H_1 \mid H_p \rfloor$$

where H_1, H_2 and H_3 are each a 12×12 square matrices. H_2 and H_3 are regular with row/column weights of 4, and H_1 is irregular with regular row weight of 4 and mixed column weights of 3 and 6. The rate 1/2 and 2/3 code matrix definitions are

$$H_{1/2} = \lfloor H_1 \mid H_p \rfloor$$

and

$$H_{2/3} = \lfloor H_2 \mid H_1 \mid H_p \rfloor,$$

respectively.

```
H3 = [
      -2    -2    -2     7    -2    14    -2    -2    -2     0    35    -2
       4    -2    -2    -2    33    -2    -2    -2     6    -2    -2    29
      18    10    -2    -2    -2    -2    -2    -2    34    38    -2    -2
      -2    -2    -2    -2    39    -2    -2    37    25    -2    17    -2
      -2    -2    32     1    -2    -2    -2    -2    -2    -2     5    27
      -2    34     9    -2    -2    -2     4    -2    -2    34    -2    -2
      -2    10    -2    25    -2    -2    23    -2    -2    -2    -2     6
      35    -2    -2    -2     6    -2    -2    34     9    -2    -2    -2
      -2    -2    -2     3    -2    -2    15    18    -2    26    -2    -2
      28    -2     9    -2    -2    13    -2    -2    -2    -2    32    -2
      -2    -2    36    -2     6    17    -2    -2    -2    -2    -2    13
      -2     9    -2    -2    -2    12     1    10    -2    -2    -2    -2
  ]

H2 = [
      -2    -2    17    12    -2    -2    -2    -2    -2    -2    24     2
       0    -2    -2    -2    17    -2    -2    23     0    -2    -2    -2
      34    -2     8    -2    -2     8    -2    -2    -2    -2     0    -2
      -2    -2    -2    38    -2    -2     5    14    -2     3    -2    -2
      -2    -2    19    -2    10    27    -2    -2    -2    -2    -2    14
      -2    31    -2    -2    -2    33    25     1    -2    -2    -2    -2
      -2    -2    -2    16    -2    25    -2    -2    -2    24     7    -2
      -2    38    -2    29    -2    -2     8    -2    -2    -2    -2    25
      12    30    -2    -2    -2    -2    -2    -2    12    27    -2    -2
      -2    39    25    -2    -2    -2    24    -2    -2     1    -2    -2
      16    -2    -2    -2    29    -2    -2    -2     1    -2    -2    23
      -2    -2    -2    -2    37    -2    -2    18    15    -2     0    -2
]

H1 = [
     0     0    -2    -2    -2    -2    -2    -2     0     0    -2    -2
    -2    -2     0     0    -2    -2    -2    -2    -2    -2     0     0
    -2    -2    -2    -2     0     0    -2    -2    -2    26    -2     9
    -2    -2    -2    -2    -2    -2     0     0    -2    13    39    -2
    -2    -2     5    -2    -2    18    -2    -2    35    -2    16    -2
    30    -2    -2    -2    -2    -2    -2    29    28    17    -2    -2
    -2    -2    -2    33    -2    -2    19    -2    34    -2    -2    -2
    -2    25    26    -2    -2    -2    -2    -2    -2    15    -2     4
    -2    -2    -2    -2    36    -2    -2     1    38    -2     9     7
    -2    23    -2    -2    -2    -2     2    -2    -2    -2    30    21
    -2    -2    -2    29    -2    27    -2    -2    -2    12    20    -2
    39    -2    -2    -2    18    -2    -2    -2    22    -2    -2    16
]
Hp = [
     0     0    -2    -2    -2    -2    -2    -2    -2    -2    -2    -2
    -2     0     0    -2    -2    -2    -2    -2    -2    -2    -2    -2
    -2    -2     0     0    -2    -2    -2    -2    -2    -2    -2    -2
    -2    -2    -2     0     0    -2    -2    -2    -2    -2    -2    -2
    -2    -2    -2    -2     0     0    -2    -2    -2    -2    -2    -2
    -2    -2    -2    -2    -2     0     0    -2    -2    -2    -2    -2
     0    -2    -2    -2    -2    -2     0     0    -2    -2    -2    -2
    -2    -2    -2    -2    -2    -2    -2     0     0    -2    -2    -2
    -2    -2    -2    -2    -2    -2    -2    -2     0     0    -2    -2
    -2    -2    -2    -2    -2    -2    -2    -2    -2     0     0    -2
    -2    -2    -2    -2    -2    -2    -2    -2    -2    -2     0     0
     0    -2    -2    -2    -2    -2    -2    -2    -2    -2    -2     0
]
```

Model Matrix Set 3 (Motorola)

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 $\mathbf{H}'_{\mathrm{bm2}}$ is not shown for rate 2/3 and rate 3/4. 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.

<u>Rate 1/2:</u>

```
  0 -1  0  0  0 -1  0 -1 -1  0 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 66 -1 29 -1 24 48 -1 -1 -1 38 56 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1
 59 67 19 -1 -1 -1 89 -1  8 65 -1 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 78 -1  5 -1 29 -1 -1 47 53 70 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1
 -1 58  4 -1 53 62 66 -1 -1 28 -1 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1
 48 -1 54 -1 23 -1 60 63 -1 -1 -1 -1  3 -1 -1 -1 -1  0  0 -1 -1 -1 -1 -1
 76 -1 44 26 64 -1 30 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1
 33 -1 -1 -1 94 -1 68 -1 26 77 66 -1 -1 -1 -1 -1 -1 -1 -1  0  0 -1 -1 -1
 86 -1 62 -1 -1 15  8 56 -1 50 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0 -1 -1
  4 90 -1 66 12 -1 43 -1 -1 40 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0 -1
 79 -1 18 -1 47 -1  7 80 -1 -1 -1 53 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0
 20 -1 38 -1 77 -1 -1 -1 38 12 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0
```

<u>Rate 2/3:</u>

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

<u>Rate 3/4:</u>

```
  0  0 -1  0  0 -1  0 -1  0 -1  0  0  0 -1 -1  0  0 -1 -1 -1  0  0 -1  0 -1 -1 -1  0
 32 -1 -1 -1 45 48 -1 25 -1 -1 -1 23 52 -1 63  9 -1  2 -1  1 57 -1 56 48 45 -1 -1 -1
 -1 -1 -1 46 10 33 -1 45 13 -1 -1 16 56 -1 -1 -1 19 -1 17 13 -1 28 -1 25 60 19 -1 -1
 60 -1 53 -1  6 -1 -1 14 16 56 -1 55 -1 23 -1 24 35 -1 -1 -1 42 -1 -1 51 40 -1 -1  3
 14 43 -1 48 60 -1 36 -1 40 -1 -1 21  0 -1 29  3 18 -1 -1 28 28 -1 -1 -1 -1 40 -1 -1
 21 -1 -1  5 54 -1 -1 21 13 49 -1  2 28 -1 -1 58 -1  4 -1  6 16 39 -1 -1 27 -1 39 -1
 -1 48 36 18 60 -1 -1  0 30 -1 -1 -1 21 55 -1 27 11 -1 -1 57 -1 -1 14 57 49 -1 -1 -1
 12 -1 -1 53 -1  9 -1 42 -1 30  6 -1 24 -1 -1 57 21 -1  5  7 55 -1 -1 51 32 -1 -1 -1
 19 -1 -1 33 -1 -1 -1  0  7 -1 -1 56 -1  0 -1 -1 16 16 -1 39  4 -1 -1 54  9 27 11  0
```

8.4.9.2.4.2 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 **p**=(p_0, …, p_{m-1}) based on an information block **s**=(s_0, …, 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 **s**=(s_0, …, s_{k-1}) and uses the matrix $\mathbf{H}_{\mathrm{bm}}$ to determine the parity-check bits. The expanded matrix **H** is determined from the model matrix $\mathbf{H}_{\mathrm{bm}}$. 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 **x**=[**s p**]=[s_0, s_1, …,$s_{k-1,}$ p_0, p_1, …,p_{m-1}], where p_0, . . . p_{m-1} are the parity-check bits; and s_0, . . . 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 $\mathbf{H}_{\mathrm{bm}}$, 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 $\mathbf{H}_{\mathrm{bm}}$ 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 $\mathbf{h}_{\mathrm{bm}}$ 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 $\mathbf{H}'_{b2}$,

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

where

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

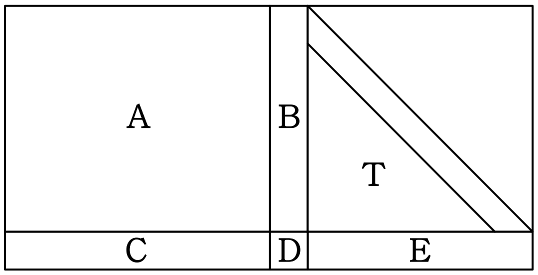
Thus all parity bits not in **v**(0) are determined by evaluating Equation (2) for $0 \le i \le m_b-2$.
Equations (1) and (2) completely describe the encoding algorithm. These equations also have a straightforward interpretation in terms of standard digital logic architectures. Since the non-zero elements $p(i,j)$ of $\mathbf{H}_{\mathrm{bm}}$ 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. $\boldsymbol{B}$ and $\boldsymbol{D}$ part have the column degree 3 and $\boldsymbol{D}$ has shift value of 1. $\boldsymbol{B}$ 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-g). The definition equation $\boldsymbol{H} \cdot \boldsymbol{v}^t = 0$ splits into two equations, as in equation 3 and 4 namely

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

and

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

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

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

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 $\boldsymbol{Au}^T$ and $\boldsymbol{Cu}^T$.

Step 2) Compute $\boldsymbol{ET}^{-1}\left(\boldsymbol{Au}^T\right)$.

Step 3) Compute $\boldsymbol{p}_1^T$ by $\boldsymbol{p}_1^T = \boldsymbol{ET}^{-1}\left(\boldsymbol{Au}^T\right)+\boldsymbol{Cu}^T$.

Step 4) Compute $\boldsymbol{p}_2^T$ by $\boldsymbol{Tp}_2^T = \boldsymbol{Au}^T + \boldsymbol{Bp}_1^T$.

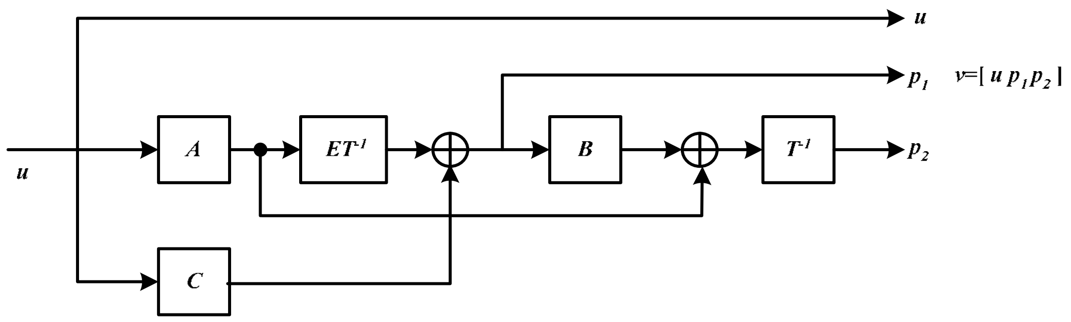


Fig. 2 Block diagram of the encoder architecture for the block LDPC code.

~~TBD description of packet encoding.~~

8.4.9.2.4.3 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.

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*.

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, $\mathbf{s'}=(s_0, \ldots, s_{k\text{-L-1}})$. When encoding, the encoder first prepends *L* zeros to **s'** of length (*k*-*L*). Then the zero-padded information vector $\mathbf{s}=[\mathbf{0}_L\ \mathbf{s'}]$ is encoded using **H** as if unshortened to generate parity bit vector **p** (length *m*). After removing the prepended zeros, the code bit vector $\mathbf{x}=[\mathbf{s'}\ \mathbf{p}]$ 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**.

Table (Samsung)

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

N	**Rate 1/2**	**Rate 2/3**	**Rate 3/4**	***L* (*Ns*)**
576	**M1(12)**	**M2(12)**	**M3(12)**	**12**
672	M1(14)	M2(14)	M4(14)	14
768	M2(24)	M2(16)	M3(16)	16
864	M1(18)	M3(24)	M4(18)	18
960	M3(40)	M2(20)	M4(20)	20

1056	M3(44)	M2(22)	M4(22)	22
1152	**M1(24)**	**M2(24)**	**M3(24)**	**24**
1248	M1(26)	M2(26)	M3(26)	26
1344	M2(42)	M2(28)	M4(28)	28
1440	M1(30)	M3(40)	M4(30)	30
1536	M2(48)	M2(32)	M3(32)	32
1632	M1(34)	M2(34)	M4(34)	34
1728	**M1(36)**	**M3(48)**	**M3(36)**	**36**
1824	M1(38)	M2(38)	M3(38)	38
1920	M1(40)	M2(40)	M4(40)	40
2016	M1(42)	M2(42)	M4(42)	42
2112	M1(44)	M2(44)	M4(44)	44
2208	M1(46)	M2(46)	M3(46)	46
2304	**M1(48)**	**M2(48)**	**M3(48)**	**48**

Table (Motorola)

[*Simulation results for the bold entries have been provided; all non-blank z factors are achievable with z expansion.*]

N(bits)	N(bytes)	z expansion factor			Number of subchannels		
		R=1/2	R=2/3	R=3/4	QPSK	16QAM	64QAM
96	12	4	2		1		
192	24	8	4		2	1	
288	36	12	6	8	3		1
384	48	16	8		4	2	
480	60	20	10		5		
576	**72**	24	12	16	6	3	2
672	84	28	14		7		
768	96	32	16		8	4	
864	108	36	18	24	9		3
960	120	40	20		10	5	
1056	132	44	22		11		
1152	**144**	48	24	32	12	6	4

1248	156	52	26		13		
1344	168	56	28		14	7	
1440	180	60	30	40	15		5
1536	192	64	32		16	8	
1632	204	68	34		17		
1728	**216**	72	36	48	18	9	6
1824	228	76	38		19		
1920	240	80	40		20	10	
2016	252	84	42	56	21		7
2112	264	88	44		22	11	
2208	276	92	46		23		
2304	**288**	96	48	64	24	12	8
	base N-K	12	16	9			
	base N	24	48	36			

Table (Intel)

[*Simulation results for the bold entries have been provided; all non-blank z factors are achievable with z expansion.*]

N(bits)	N(bytes)	z expansion factor			Number of subchannels		
		R=1/2	R=2/3	R=3/4	QPSK	16QAM	64QAM
96	12	4		2	1		
192	24	8		4	2	1	
288	36	12	8	6	3		1
384	48	16		8	4	2	
480	60	20		10	5		
576	**72**	24	16	12	6	3	2
672	84	28		14	7		
768	96	32		16	8	4	
864	108	36	24	18	9		3
960	120	40		20	10	5	

1056	132	44		22	11		
1152	**144**	48	32	24	12	6	4
1248	156	52		26	13		
1344	168	56		28	14	7	
1440	180	60	40	30	15		5
1536	192	64		32	16	8	
1632	204	68		34	17		
1728	**216**	72	48	36	18	9	6
1824	228	76		38	19		
1920	240	80		40	20	10	
2016	252	84	56	42	21		7
2112	264	88		44	22	11	
2208	276	92		46	23		
2304	**288**	96	64	48	24	12	8
base N-K		12	12	12			
base N		24	36	48			

~~TBD description of code adjustment.~~

8.4.9.2.4.4 Packet Encoding

[*unchanged*]
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.

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 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.