Project	**IEEE 802.16 Broadband Wireless Access Working Group <http://ieee802.org/16>**
Title	**High Girth LDPC Coding for OFDMA PHY**
Date Submitted	**2005-1-25**

Source(s)		
	Robert Xu, David Yuan, Li Zeng,Liujun Hu ZTE Inc. 3/F, Bldg.711, Pengji Industrial Park, Liantang, Shenzhen, 518004, P.R.China	Voice: +86 755 26773000 6574 Fax: +86 755 26773000 6616 mailto: xu.jun2@zte.com.cn yuan.liuqing@zte.com.cn zeng.li@zte.com.cn hu.liujun@zte.com.cn

Re:	IEEE P802.16-REVe/D5a
Abstract	High girth LDPC codes design technologies are introduced into the contribution for the first time, to overcome the " error floor " phenomenon, and enhance the BER curve descending speed when SNR is high. Also it has a performance improvement over the LDPC codes proposed in the previous contribution of other companies.
Purpose	Complete the LDPC specification text with our new technologies
Notice	
Release	
Patent Policy and Procedures	

High Girth LDPC Coding for OFDMA PHY

Robert Xu, David Yuan, Li Zeng, Liujun Hu

ZTE, Inc.

Overview

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.

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. We would like to support their work, so we apply our technologies to design new LDPC codes with better performance. In fact, we are one of the first people who suggest that 24 columns base matrices for all code rates should been used.

As we all know, code rate, codeword length and degree distribution decide the performance of LDPC codes. When Message Passing algorithm is used, the short cycles in the bipartite of LDPC codes will obviously degrade the performance of the LDPC codes, especially when SNR is high. Girth was defined as the length of the shortest cycle of the bipartite of LDPC codes, and it has become a criterion on the performance of LDPC codes. During the decoding iteration process, the extrinsic information from one variable node always returns to itself. Some variable nodes are dependent on each other. The higher the girth of one LDPC code is, the more iteration times that the extrinsic information from one variable node return to itself needs, and the less extrinsic information from one variable node returns to itself, so more independent the variable nodes will be. Thus it is very important to construct high girth LDPC codes to satisfy the requirement of Message passing algorithm.

High girth LDPC codes try to overcome the "error floor" phenomenon, and the BER curve of them will descend more steeply. However, normal LDPC codes have the "error floor" phenomenon, BER curve descends more and more slowly. It is always difficult to arrive at the point BER = 10e-6, which efficient data communication needs. So high girth LDPC codes are suitable for the situation where low BER is needed.

By changing base matrix for each specific code rate to design high girth LDPC codes, performance improvement is obtained. Our method not only can be used to design regular LDPC codes, but also can be used to design irregular LDPC codes. In addition, any matrix structure with given code rate, codeword length and degree distribution can be constructed by our method, which largely increases the feasibilities of our method. It has become the solution of systematic shortcomings of normal LDPC codes design.

In fact, if BP decoding algorithm has been used, our codes have the best performance among all provided schemes. For all three rates, the performance advantage not only has appeared in high SNR region, but also in water-fall region. The performance of our codes are better than the performance of contribution #006, the complexity of our codes are similar with the complexity of contribution #006. Actually, hardware can provide enough parallel processing branches(larger than 2*z), then BP will be used, and layered BP cannot be used, thus our codes are the best choice for 16e.

Features

- **Simple encoding and decoding**
- **Less average iteration numbers**
- **Good performance**
- **Eliminate error floor**
- **Uniform scaling method**
- **High girth**

Simulation Results

Simulation results for ZTE high girth codes of the rate 1/2, 2/3, 3/4 code families are shown in Figure 1-3. For these three rates, code sizes considered are all 576-2304. The simulation conditions are: AWGN channel, BPSK modulation, max iterations times 50, using generic floating-point belief propagation. From the simulation results we can find that our codes overcome the “error floor” phenomenon, and the BER curve of them will descend more steeply. When SNR is high, our high girth method obviously obtained an improved performance.

Performance of the ZTE codes design for 802.16e in AWGN channel is shown in Figure 1, 2, and 3 for rate 1/2, 2/3, 3/4. The block sizes *n* range from 576 to 2304 for all three code rates. The expansion factor *z* ranges from 24 to 96, as shown in Figures 1-3. The block size and the expansion factor are related by *n* = 24*z.

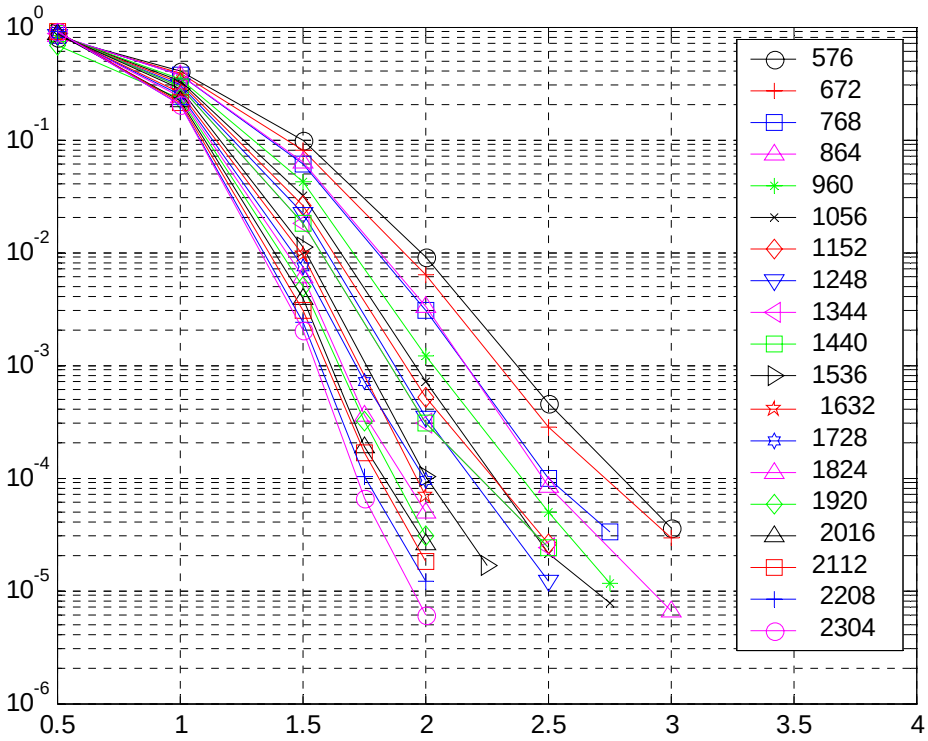

Figure 1. FER vs. E_b/N_0 (dB), BPSK, rate 1/2, AWGN channel.

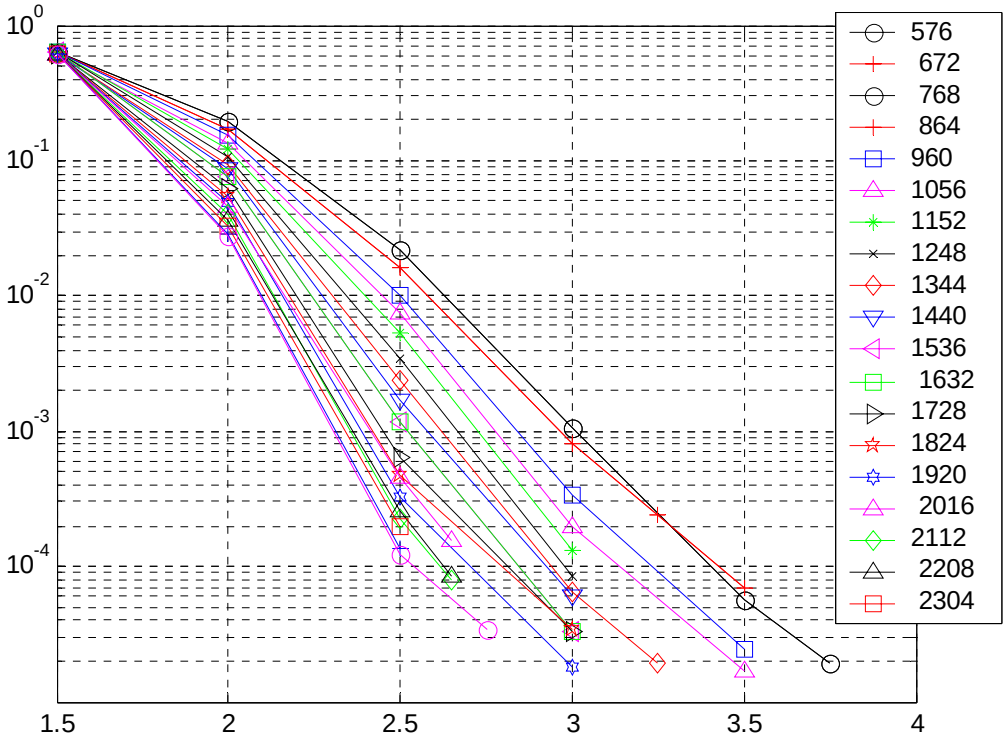

Figure 2. FER vs. E_b/N_0 (dB), BPSK, rate 2/3, AWGN channel.

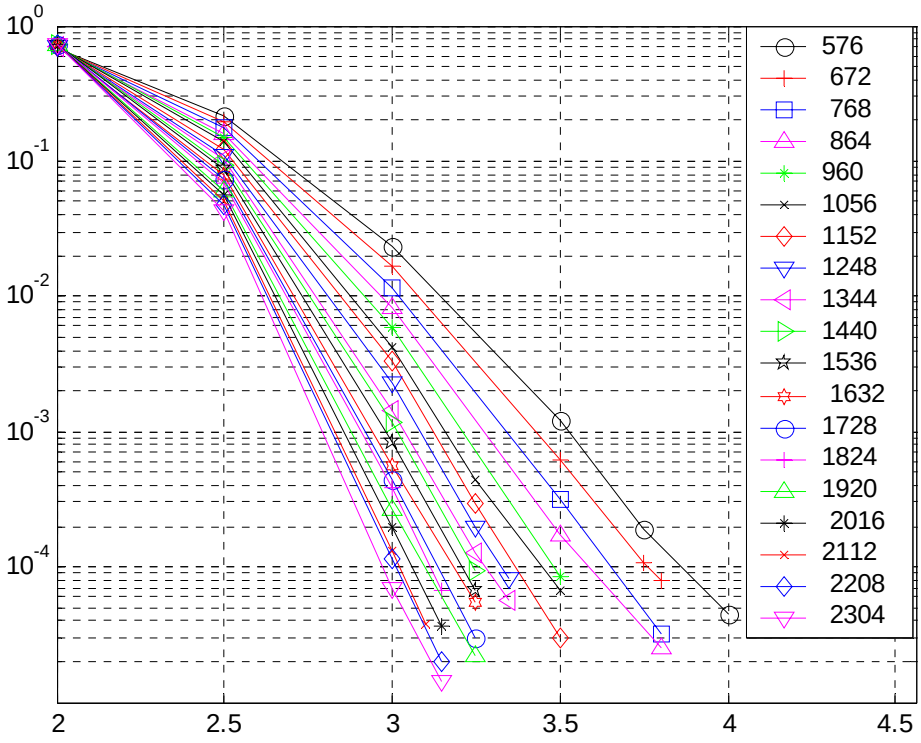

Figure 3. FER vs. E_b/N_0 (dB), BPSK, rate 3/4, AWGN channel.

Simulation showed that normal LDPC codes' BER curves descend more and more slow. It is always difficult to arrive at the point BER = 10e-6, which efficient data communication needs. But high girth LDPC codes can arrive at the point easily. So, high girth LDPC codes are suitable for situations where low BER are needed.

Recommended Text Changes:

Modify the text in 802.16e_D5a as follows, adjusting the numbering as required:

<Delete the text "of scaling and shortening" in section 8.4.9.2.5.1, p. 364, line 22>

<Move the text between the section headings "Direct Encoding (Informative)" and "Method 1" (section 8.4.9.2.5.2 p. 365 line 35 to p366 line 9) to section 8.4.9.2.5.1 Code Description p. 364 line 50, immediately after the sentence "The base matrix $\mathbf{H}_b$ is partitioned into two sections …" Delete "For the two methods, described below" from the moved text.>

<Add the following text to the end of section 8.4.9.2.5.1 Code Description.>
A base model matrix is defined for the largest code length (n=2304) of each code rate. The set of shifts $\{P_{i,j}\}$ in the base model matrix are used to determine the shift sizes for all other code lengths of the same code rate. Each base model matrix has n_b=24 columns, and the expansion factor z_f is equal to n/24 for code length n. For code length n=2304 the expansion factor is designated z_0=96.
For all code rates, the shift sizes $\{ P_{i,j}(f)\}$ for a code size corresponding to expansion factor z_f are derived from $\{ P_{i,j} \}$ by scaling $P_{i,j}$ proportionally,

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

Note that $\alpha_f = z_0 / z_f$ and $\lfloor x \rfloor$ denotes the flooring function which gives the nearest integer towards -∞.

Rate 1/2:

```
93 -1 -1 -1 77 -1 -1 25 -1 30 56 65  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 87 -1 -1 -1 70  5 -1 -1 88 31 32 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  9 86 10 20 37 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1
45 -1 -1 90 -1  9 -1 -1 -1 92 13  1 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1  2 -1 75 26 95 86 -1 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1
-1 66  5 -1 -1 -1 -1 -1 -1 53 49 25 36 -1 -1 -1 -1  0  0 -1 -1 -1 -1 -1
-1 -1 -1 93 -1 -1 -1 39 -1 25 30 91 -1 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1
-1 -1 -1 -1 82 -1 60 -1 15 52  4 73 -1 -1 -1 -1 -1 -1 -1  0  0 -1 -1 -1
21 -1 22 -1 -1 -1 -1 -1 -1 58 42 38 -1 -1 -1 -1 -1 -1 -1 -1  0  0 -1 -1
-1 23 -1 82 86 -1 -1 -1 -1 19 94 48 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0 -1
-1 -1  0 -1 -1 -1 26 52 -1 75  2 28 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0
-1 -1 -1 -1 -1 67 -1 -1 62 69 39 80  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0
```

Rate 2/3:

```
94 -1 -1  6 -1 69 -1 -1 86 -1 -1 -1  2 81 10 67  0  0 -1 -1 -1 -1 -1 -1
-1  8 -1 -1 53 -1 34 -1 -1  7 67 -1 11  1 23 32 -1  0  0 -1 -1 -1 -1 -1
20 -1 -1 -1 -1 25 -1 79 54 -1 -1 27 19 12 52 74 -1 -1  0  0 -1 -1 -1 -1
-1 -1 39 17 32 -1 -1 -1 -1 -1 19 -1 83 81 73 59  5 -1 -1  0  0 -1 -1 -1
33 24 -1 -1 -1 35 -1 29 17 -1 -1 -1 42 67 82 77 -1 -1 -1 -1  0  0 -1 -1
-1 -1 -1 90 -1 -1 35 -1 -1 51 59 54 91 49 57 89 -1 -1 -1 -1 -1  0  0 -1
44 -1  2 -1 36 -1  5 25 -1 -1 -1 -1 64  7 81 10 -1 -1 -1 -1 -1 -1  0  0
-1 67 -1 -1 -1 88 -1 -1 40  3 -1 36 32 49 54 95  0 -1 -1 -1 -1 -1 -1  0
```

Rate 3/4:

```
-1 81 -1 28 -1 -1 14 25 17 -1 -1 85 29 52 78 95 22 92  0  0 -1 -1 -1 -1
42 -1 14 68 32 -1 -1 -1 -1 70 43 11 36 40 33 57 38 24 -1  0  0 -1 -1 -1
-1 -1 20 -1 -1 63 39 -1 70 67 -1 38  4 72 47 29 60  5 80 -1  0  0 -1 -1
64  2 -1 -1 63 -1 -1  3 51 -1 81 15 94  9 85 36 14 19 -1 -1 -1  0  0 -1
-1 53 60 80 -1 26 75 -1 -1 -1 -1 86 77  1  3 72 60 25 -1 -1 -1 -1  0  0
77 -1 -1 -1 15 28 -1 35 -1 72 30 68 85 84 26 64 11 89  0 -1 -1 -1 -1  0
```

<Replace the contents of section 8.4.9.2.5.3 (p. 369 line 45 to p. 370 line 64) with the following text and table.>

The LDPC code flexibly supports different block sizes for each code rate through the use of an expansion factor. Each base model matrix has n_b=24 columns, and the expansion factor (*z* factor) is equal to *n*/24 for code length *n*. In each case, the number of information bits is equal to the code rate times the coded length *n*.

,

Table 316b – LDPC Block Sizes and Code Rates

n (bits)	*n* (bytes)	*z* factor	*k* (bytes)			Number of subchannels		
			R=1/2	R=2/3	R=3/4	QPSK	16QAM	64QAM
576	72	24	36	48	54	6	3	2
672	84	28	42	56	63	7		
768	96	32	48	64	72	8	4	
864	108	36	54	72	81	9		3
960	120	40	60	80	90	10	5	
1056	132	44	66	88	99	11		
1152	144	48	72	96	108	12	6	4
1248	156	52	78	104	117	13		
1344	168	56	84	112	126	14	7	
1440	180	60	90	120	135	15		5
1536	192	64	96	128	144	16	8	
1632	204	68	102	136	153	17		
1728	216	72	108	144	162	18	9	6
1824	228	76	114	152	171	19		
1920	240	80	120	160	180	20	10	
2016	252	84	126	168	189	21		7
2112	264	88	132	176	198	22	11	
2208	276	92	138	184	207	23		
2304	288	96	144	192	216	24	12	8

<In 8.4.9.2.5.4 Packet Encoding, p.371 lines 19-23, correct the formatting by inserting delimiters ": " after the variables "j"(line 19), "Nsch" (line 20), "F" (line 21), and "M" (line 22).>

<In 8.4.9.2.5.4 Packet Encoding, p.371 line 19, delete the text "and FEC rate" because the parameter j is independent of the FEC rate for LDPC>

<NOTE to Editor: The remaining items below update the signaling for LDPC>

<In 8.4.4.3 DL Frame Prefix p. 232, line 48, Table 266a, “Coding_Indication” row, insert a line in the “Notes” column “0b100 – LDPC encoding used on DL-MAP”, and modify existing line “0b100 to 0b111 – Reserved” to“0b101 to 0b111 – Reserved”>

<In 11.3.1, p. 390, line 57 add the following text>
11.3.1.1 Uplink burst profile encodings
[Insert the following text in the “Value” column of the first row (“FEC code type and modulation type”) of Table 355 p. 663 of 802.16-REVd/D5, and change “26..255=Reserved” to “35..255=Reserved”]
26=QPSK (LDPC) 1/2
27=QPSK (LDPC) 2/3
28=QPSK (LDPC) 3/4
29=16-QAM (LDPC) 1/2
30=16-QAM (LDPC) 2/3
31=16-QAM (LDPC) 3/4
32=64-QAM (LDPC) 1/2
33=64-QAM (LDPC) 2/3
34=64-QAM (LDPC) 3/4

<In 11.4, p. 394, line 56 add the following text>
11.4.2 Downlink burst profile encodings
[Insert the following text in the “Value” column of the first row (“FEC code type”) of Table 361 p. 668 of 802.16-REVd/D5, and change “26..255=Reserved” to “35..255=Reserved”]
26=QPSK (LDPC) 1/2
27=QPSK (LDPC) 2/3
28=QPSK (LDPC) 3/4
29=16-QAM (LDPC) 1/2
30=16-QAM (LDPC) 2/3
31=16-QAM (LDPC) 3/4
32=64-QAM (LDPC) 1/2
33=64-QAM (LDPC) 2/3
34=64-QAM (LDPC) 3/4

<In 11.8.3.7.2 OFDMA MSS demodulator, p. 408, line 24 change the “Length” column entry of “1” to “2”, and insert the following two entries after line 31 in the “Value” column>
Bit#8: LDPC
Bits#9-15: Reserved; shall be set to zero

<In 11.8.3.7.2 OFDMA MSS demodulator, p. 408, line 32 add the following text>
11.8.3.7.3 OFDMA MSS modulator
[Copy the table from 11.8.3.7.3, and insert the following text in the “Value” column of the first row, and change “Bits#6-7: Reserved; shall be set to zero” to “Bit#7: Reserved; shall be set to zero”]
Bit#6: LDPC