Project	**IEEE 802.16 Broadband Wireless Access Working Group <http://ieee802.org/16>**
Title	**LDPC Support in IEEE802.16e**
Date Submitted	**2005-01-25**

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 li.zeng@zte.com.cn hu.liujun@zte.com.cn
	Min-seok Oh Kyuhyuk Chung Kihyoung Cho LG Electronics. Inc. 533 LG R&D Complex Hogye-dong Dong-An-gu Anyang-shi Kyunggido Korea	Voice: 82-31-450-2945 Fax: 82-31-450-7912 mailto: minoh@lge.com kyuhyuk@lge.com kihyoung@lge.com

Re:	IEEE P802.16-REVe/D5a, BRC
Abstract	In this contribution, some schemes for LDPC are provided to complete LDPC in IEEE802.16e.
Purpose	Complete the LDPC in IEEE802.16e.
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>.

LDPC Support in IEEE802.16e

Robert Xu, David Yuan, Li Zeng, and Liujun Hu
ZTE Inc.
Min-seok Oh, Kyuhyuk Chung, and Kihyoung Cho
LG Electronics, Inc.

1. Introduction

After downselection confirmation voting in informal ad-hoc group, LGE raised a big technical concern by finding severe error floors in r=2/3 code on current proposal. Those error floors were clearly identified by 5 companies in informal LDPC group. Six companies, Motorola, Intel, Nortel, Runcom, LGE and Samsung recognized as critical problem and agreed to re-design the code for technically fixing current proposal. Accordingly LGE and Samsung participated in re-designing the code for r=2/3. For fair comparison, Intel, Nortel, and Motorola showed the results of their cross-simulations as follows. [See Appendix 2.1]

LGE's performance was comparable or even better without error floor for every code size than Samsung's. In addition, those codes provide the most outstanding feature supporting perfect parallel decoding which made it firstly considered in r=1/2 code of confirmed document. Furthermore, flooring technique in scaling method is used for avoiding additional complexity, since other two rates such as r=1/2 and 3/4 was used already.

On the other hand, Samsung's new design still had error floors, which were identified by cross simulations. In addition, Samsung's re-designed code does not support parallel decoding method. Modulo scaling method is inconsistent with other two code rate so that it makes additional complexity burden in implementation. [See Appendix 2.2 – 2.4]

When we verifies all technical aspects in performance, complexity, and promising feature, it is so natural and fair procedure that LGE's design is accepted as solution of the current proposal.

However, our original concern was ignored by denial of two companies, Nortel and Samsung even in spite their having admitted the critical problem already. The current proposal becomes existed with a big problem.

Consequently, in our view the current proposal must be very wrong materials for standard specification and must not be accepted by standard. On this technical consideration, we propose LDPC codes being perfect for standard.

On the last session, the basic framework of LDPC codes for OFDMA PHY was adopted. So the completion of the LDPC code section is required. We propose low-complexity and high performance LDPC codes supporting all rates and codeword sizes with flexible and efficient rate adjustment. The proposed LDPC codes are based on the contribution IEEE C802.16e-04/373r1.

The merged document has good properties, such as single scaling method, parallel processing design. Specifically, for single scaling method, merged document uses FLOORING for r=1/2, r=2/3. and r=3/4. However, in contribution 006, FLOORING is used for r=1/2 and r=3/4, and MODULO is used for r=2/3. In parallel processing, merged document supports r=1/2 and r=2/3, while the contribution 006 supports parallel processing ONLY for r=1/2.

Description for R=3/4

For our R=3/4 irregular code, our code obviously has better performance than the code r=3/4 in contribution 006 for all code sizes, and for the whole SNR region in r=3/4, about 0.1-0.2dB better than code in contribution 006.

For irregular codes, the maximum column weight of our codes is 6. But, I must emphasize that the complexity of our irregular codes basically is the same as the complexity of semi-regular codes in contribution 006.

Now I'll give the detailed data to illustrate and prove the conclusion.

	Contribution #006	This Proposal
Information portion	Semi-Regular	Irregular
Non-zero elements numbers	85	88
Average Row Weight	14.16667	14.6667
Average Column Weight	3.5417	3.6667

Then we can see that two codes almost have the same Average Row Weight and the Non-zero elements numbers, so we can conclude that the two codes almost have the same complexity during each iteration.

From our simulation results, we can conclude that the two codes have approximately the same average iteration numbers.

As we can see, the complexity of LDPC codes' decoding depends on the decoding complexity of each iteration and the average iteration numbers. So from the above conclusion, we can further conclude that the two codes almost have the same complexity.

From the comparison of performance figure, we can clearly find that the performance curves of our codes are so clear, one by one. There is no cross and no error floor of the performance curves of different code sizes.

However, the performance curves of contribution 006 are in some mess. Performance curves are not distributed equally. There are several crossings above FER=0.0001.

2. Description of proposed codes

Features

- Fast parallel processing implementation feature for r=1/2 and r=2/3.
  - → For r=1/2, fully comply with parallel pipe-line processing. (*see* Appendix).
  - → For r=2/3, fully comply with parallel pipe-line processing. (*see* Appendix).
  - → For all rates, r=1/2, r=2/3, and r=3/4, the same 24 columns are used to help parallel processing implementation
- Good performance over AWGN channel and fading channel environment
  - → No serious performance degradation for all 19 code sizes of each code rate
  - → No error floor at FER=0.0001 in AWGN
- Low complexity encoding and decoding
  - → Low connectivity due to relatively low number of total weight
  - → Easy routing due to column-wise regular design
  - → Very low memory requirement for H matrix description
    - – Use of a single base matrix per each code rate
- Simple code description with minimal number of base matrices
  - → Support all code rates and codeword sizes
    - – Use of a 12-by-24 base matrix M1 for r=1/2
    - – Use of a 8-by-24 base matrix M2 for r=2/3
    - – Use of a 6-by-24 base matrix M3 for r=3/4
  - → Use of a single scaling mechanism by flooring

Code rate	1/2	2/3	3/4
Model matrix size	12 by 24	8 by 24	6 by 24
Number of Non-zero Element	73	81	88
Information portion	Regular		Irregular
Maximum column weight	4		6
Method of modifying the shift sizes	Flooring		

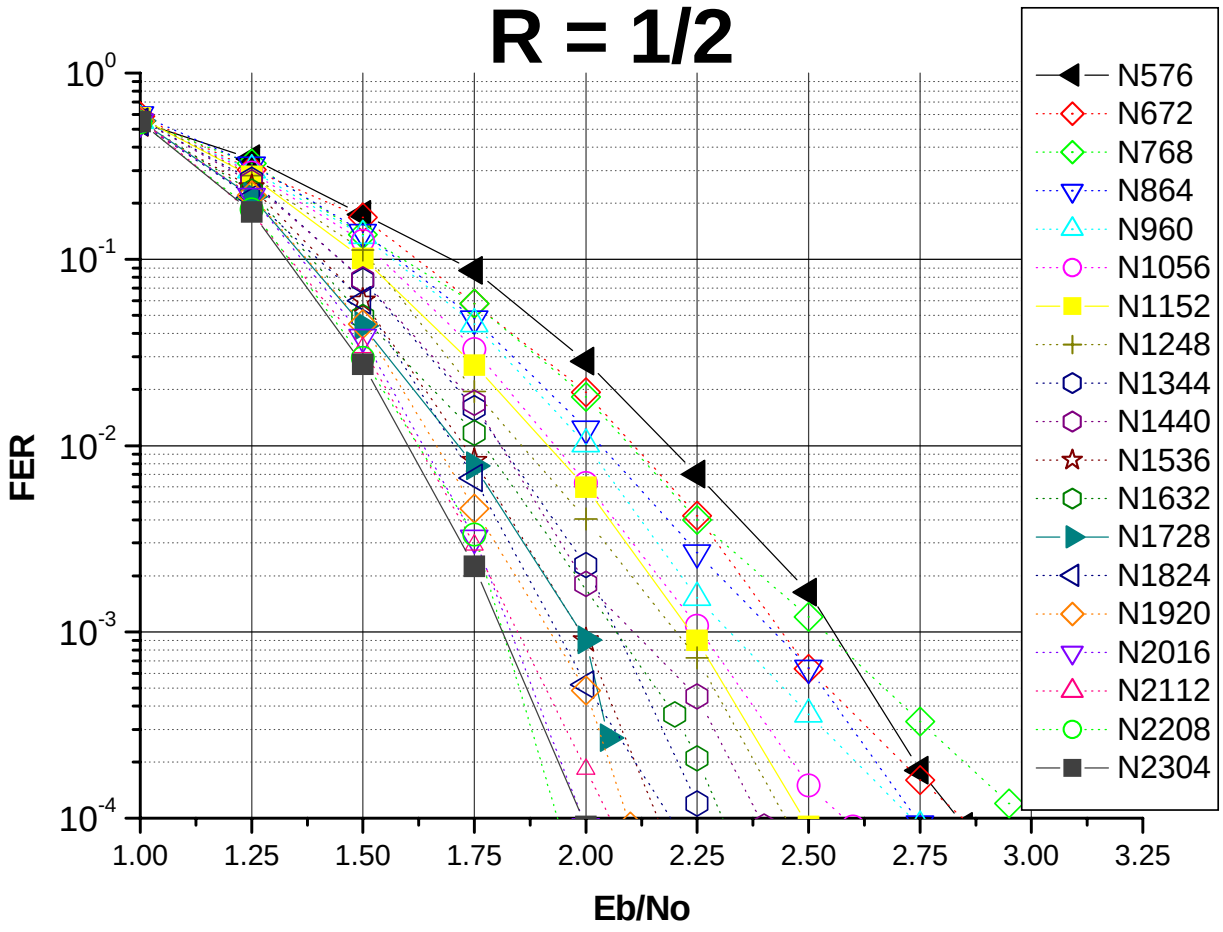
Comparison with current proposal:

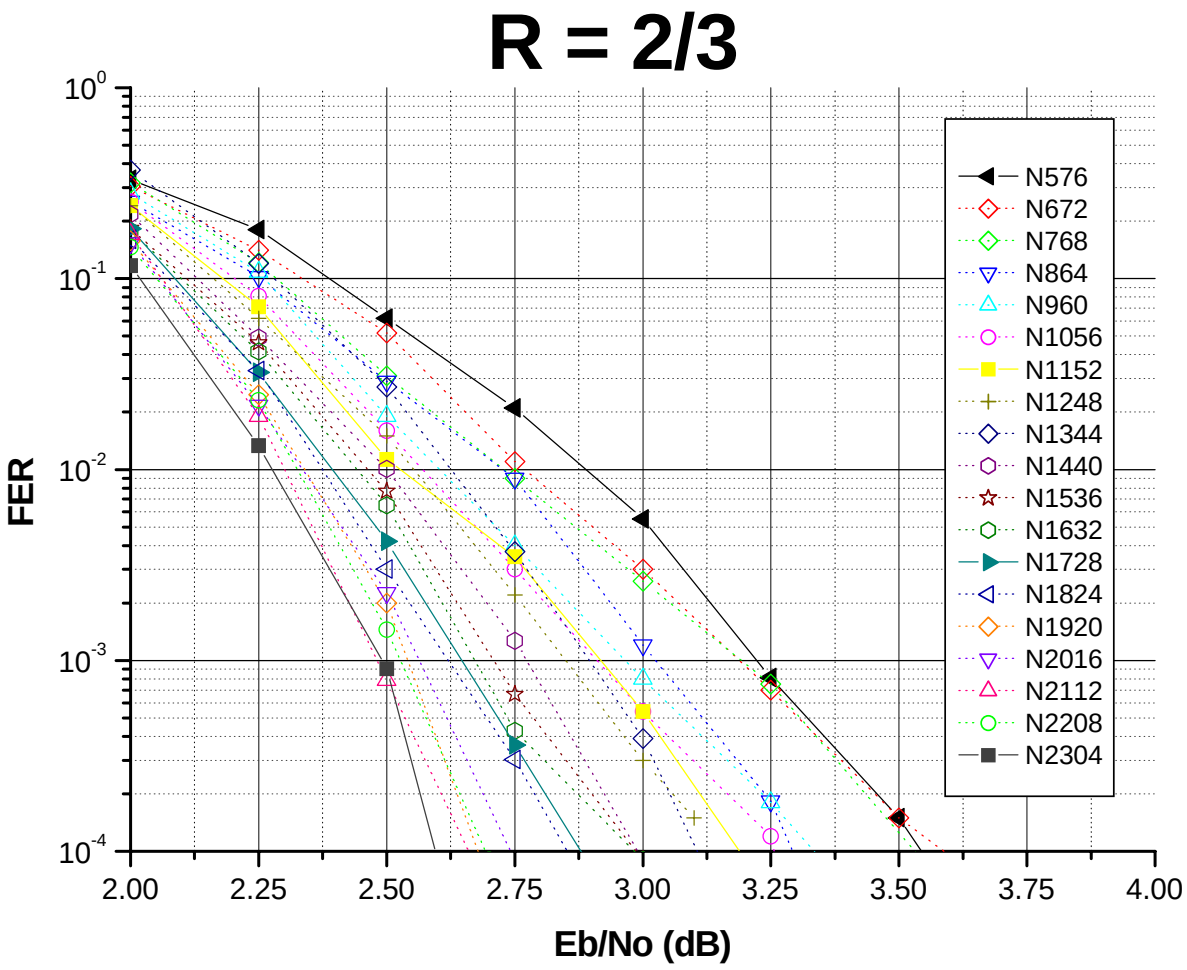
Difference from the proposed document 006

Properties / Proposal	Contribution 066r2	Contribution 006
Fast parallel processing support	Fully support at R=1/2, 2/3	R=1/2 only
Maximum column weight	6	7
Scaling method	Flooring only	Flooring and Modulo
Information portion	Regular at R=1/2, 2/3 Irregular R=3/4	Irregular at R=1/2, 2/3 Regular R=3/4
Error floor	None	Severe error floor for R=2/3 at N=1632, 1536, 768, 672
Performance	R=1/2: Contribution 066r2 is a little better R=2/3: Contribution 066r2 is comparable and better at FER=0.0001 R=3/4: Contribution 066r2 is significantly better	

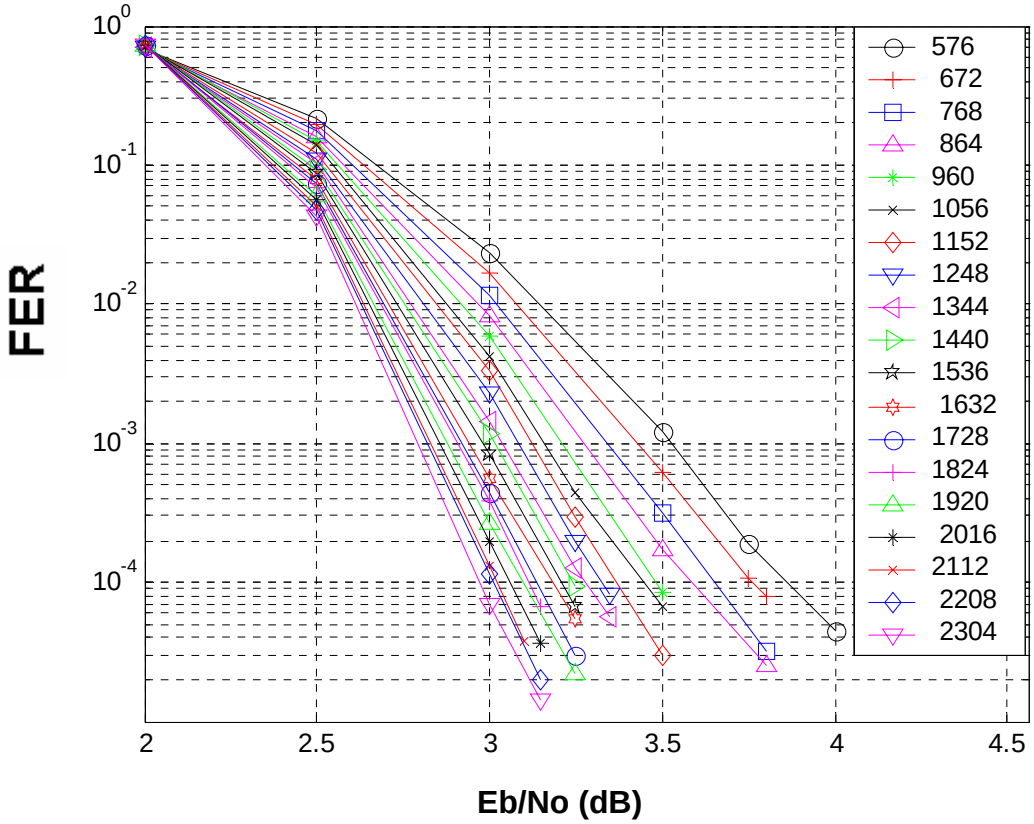
Simulation Results

For all rates, performance curves are provided for all codeword sizes. Three base matrices, M1, M2, and M3 are used for the following simulations. To cover different codeword sizes at a given code rate, expansion is used.




R = 2/3
FER
Eb/No (dB)
10^0
10^-1
10^-2
10^-3
10^-4
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
N576
N672
N768
N864
N960
N1056
N1152
N1248
N1344
N1440
N1536
N1632
N1728
N1824
N1920
N2016
N2112
N2208
N2304

R = 3/4


FER
Eb/No (dB)
10^0
10^-1
10^-2
10^-3
10^-4
2
2.5
3
3.5
4
4.5
576
672
768
864
960
1056
1152
1248
1344
1440
1536
1632
1728
1824
1920
2016
2112
2208
2304

3. Recommended Text Changes:

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

8.4.9.2.5 Low Density Parity Check Code (optional)

[Insert a new paragraph at the end of section 8.4.9.2.5.1 as indicated:]

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.

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 right shifted identity 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 Hbm. The model matrix Hbm is the same size as the binary base matrix Hb, with each binary entry(i,j) of the base matrix Hb replaced to create the model matrix Hbm. Each 0 in Hb is replaced by a blank or negative value (e.g., by .1) to denote a z×z all-zero matrix, and each 1 in Hb is replaced by a circular shift size p(i,j)≥ 0. The model matrix Hbm 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 $H_b = \left[(H_{b1})_{m_b \times k_b} \mid (H_{b2})_{m_b \times m_b} \right]$.

Base matrix

To reduce complexity, the minimal number of base matrices is used. Three base matrices are used to cover all rates and codeword sizes. For r=1/2, the size of the base matrix is 12 by 24, for r=2/3, 8 by 24, and for r=3/4, 6 by 24 respectively. The maximum shift s_m is 96 for all base matrices. A shift s is floored down based on an expansion factor z as follows;

$$s' = \lfloor s \cdot z / s_m \rfloor$$

where $\lfloor \cdot \rfloor$ is the flooring function, i.e, the nearest integer to minus infinity.

Matrix descriptions

For r=1/2;

M1=

−1	−1	−1	20	−1	−1	85	6	−1	−1	77	−1	95	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1
−1	−1	−1	−1	25	50	−1	−1	−1	52	87	−1	−1	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1
−1	−1	−1	49	−1	25	80	−1	−1	−1	−1	78	−1	−1	0	0	−1	−1	−1	−1	−1	−1	−1	−1
−1	−1	−1	61	33	−1	67	−1	−1	−1	−1	19	0	−1	−1	0	0	−1	−1	−1	−1	−1	−1	−1
−1	−1	64	−1	−1	−1	38	73	−1	−1	10	−1	−1	−1	−1	−1	0	0	−1	−1	−1	−1	−1	−1
−1	83	28	−1	−1	86	−1	−1	−1	−1	−1	62	−1	−1	−1	−1	−1	0	0	−1	−1	−1	−1	−1
33	1	−1	−1	−1	−1	−1	−1	89	−1	−1	85	−1	−1	−1	−1	−1	−1	0	0	−1	−1	−1	−1
21	43	47	−1	−1	−1	−1	−1	65	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	−1	−1	−1
−1	−1	37	−1	−1	−1	−1	63	−1	51	13	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	−1	−1
73	62	−1	−1	−1	88	−1	−1	12	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	−1
−1	−1	−1	40	95	−1	−1	−1	42	37	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0
25	−1	−1	−1	61	−1	−1	83	−1	56	−1	−1	95	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0

For r=2/3;

M2=

2	−1	19	−1	47	−1	48	−1	36	−1	82	−1	47	−1	15	−1	95	0	−1	−1	−1	−1	−1	−1
−1	69	−1	88	−1	33	−1	3	−1	16	−1	37	−1	40	−1	48	−1	0	0	−1	−1	−1	−1	−1
10	−1	86	−1	62	−1	28	−1	85	−1	16	−1	34	−1	73	−1	−1	−1	0	0	−1	−1	−1	−1
−1	28	−1	32	−1	81	−1	27	−1	88	−1	5	−1	56	−1	37	−1	−1	−1	0	0	−1	−1	−1
23	−1	29	−1	15	−1	30	−1	66	−1	24	−1	50	−1	62	−1	−1	−1	−1	−1	0	0	−1	−1
−1	30	−1	65	−1	54	−1	14	−1	0	−1	30	−1	74	−1	0	−1	−1	−1	−1	−1	0	0	−1
32	−1	0	−1	15	−1	56	−1	85	−1	5	−1	6	−1	52	−1	0	−1	−1	−1	−1	−1	0	0
−1	0	−1	47	−1	13	−1	61	−1	84	−1	55	−1	78	−1	41	95	−1	−1	−1	−1	−1	−1	0

For r=3/4;

M3=

-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

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.

The encoding of a packet at the transmitter generates parity-check bits $\mathrm{p} = (p_0, \cdots, p_{m-1})$ based on an information block $\mathrm{s} = (s_0, \cdots, s_{k-1})$, 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 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 of matrix operations conducted over GF(2).

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

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 Adjustment of the one or more H matrices of the fundamental code set. For each supported rate and block size. Some combinations 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. 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 H_1 and the rest of 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, $s' = (s_0, \cdots, s_{k-L-1})$. When encoding, the encoder first prepends *L* zeros to s' of length *(k-L)*. Then the zero-padded information vector $s = [0_L\ s']$ is encoded using H as if un shortened 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 using the last *(n-L)* columns of the 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 aaa Code rate and block size adjustment with variable expansion

code rate	Code word Size	576	672	768	864	960	1056	1152	1248	1344	1440	1536	1632	1728	1824	1920	2016	2112	2208	2304
	base matrix	Expansion Factor																		
r=1/2	M1	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80	84	88	92	96
r=2/3	M2	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80	84	88	92	96
r=3/4	M3	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80	84	88	92	96

8.4.9.2.5.4 Packet Encoding

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 dose 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 isequal 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

Control information and packets that result in a codeword size n of less than 576 bits are encoded using convolutional coding (CC) with appropriate code rates and modulation orders, as described in section 8.4.9.2.1.

4. Appendix

1. parallel processing

■ Original matrix (r=1/2)

0		−1	−1	−1	20	−1	−1	85	6	−1	−1	77	−1	95	0	−1	−1	−1	−1	−1	−1	−1	−1	−1
1		−1	−1	−1	−1	25	50	−1	−1	−1	52	87	−1	−1	0	0	−1	−1	−1	−1	−1	−1	−1	−1
2		−1	−1	−1	49	−1	25	80	−1	−1	−1	−1	78	−1	−1	0	0	−1	−1	−1	−1	−1	−1	−1
3		−1	−1	−1	61	33	−1	67	−1	−1	−1	−1	19	0	−1	−1	0	0	−1	−1	−1	−1	−1	−1
4		−1	−1	64	−1	−1	−1	38	73	−1	−1	10	−1	−1	−1	−1	−1	0	0	−1	−1	−1	−1	−1
5		−1	83	28	−1	−1	86	−1	−1	−1	−1	−1	62	−1	−1	−1	−1	−1	0	0	−1	−1	−1	−1
6		33	1	−1	−1	−1	−1	−1	−1	89	−1	−1	85	−1	−1	−1	−1	−1	−1	0	0	−1	−1	−1
7		21	43	47	−1	−1	−1	−1	−1	65	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	−1	−1
8		−1	−1	37	−1	−1	−1	−1	63	−1	51	13	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	−1	−1
9		73	62	−1	−1	−1	88	−1	−1	12	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	−1
10		−1	−1	−1	40	95	−1	−1	−1	42	37	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0
11		25	−1	−1	−1	61	−1	−1	83	−1	56	−1	−1	95	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0

■ Permuted matrix (No two adjacent rows overlap in permuted matrix, including first and last rows)

11		25	−1	−1	−1	61	−1	−1	83	−1	56	−1	−1	95	−1	−1	−1	−1	−1	−1	−1	−1	−1	0
2		−1	−1	−1	49	−1	25	80	−1	−1	−1	−1	78	−1	−1	0	0	−1	−1	−1	−1	−1	−1	−1
8		−1	−1	37	−1	−1	−1	−1	63	−1	51	13	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	−1	−1
3		−1	−1	−1	61	33	−1	67	−1	−1	−1	−1	19	0	−1	−1	0	0	−1	−1	−1	−1	−1	−1
7		21	43	47	−1	−1	−1	−1	−1	65	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	−1	−1
1		−1	−1	−1	−1	25	50	−1	−1	−1	52	87	−1	−1	0	0	−1	−1	−1	−1	−1	−1	−1	−1
6		33	1	−1	−1	−1	−1	−1	−1	89	−1	−1	85	−1	−1	−1	−1	−1	−1	0	0	−1	−1	−1
0		−1	−1	−1	20	−1	−1	85	6	−1	−1	77	−1	95	0	−1	−1	−1	−1	−1	−1	−1	−1	−1
9		73	62	−1	−1	−1	88	−1	−1	12	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	−1
4		−1	−1	64	−1	−1	−1	38	73	−1	−1	10	−1	−1	−1	−1	−1	0	0	−1	−1	−1	−1	−1
10		−1	−1	−1	40	95	−1	−1	−1	42	37	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0
5		−1	83	28	−1	−1	86	−1	−1	−1	−1	−1	62	−1	−1	−1	−1	−1	0	0	−1	−1	−1	−1	−1
11		25	−1	−1	−1	61	−1	−1	83	−1	56	−1	−1	95	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0

■ Original matrix (r=2/3)

[0]		2	−1	19	−1	47	−1	48	−1	36	−1	82	−1	47	−1	15	−1	95	0	−1	−1	−1	−1	−1	−1
[1]		−1	69	−1	88	−1	33	−1	3	−1	16	−1	37	−1	40	−1	48	−1	0	0	−1	−1	−1	−1	−1
[2]		10	−1	86	−1	62	−1	28	−1	85	−1	16	−1	34	−1	73	−1	−1	−1	0	0	−1	−1	−1	−1
[3]		−1	28	−1	32	−1	81	−1	27	−1	88	−1	5	−1	56	−1	37	−1	−1	−1	0	0	−1	−1	−1
[4]		23	−1	29	−1	15	−1	30	−1	66	−1	24	−1	50	−1	62	−1	−1	−1	−1	−1	0	0	−1	−1
[5]		−1	30	−1	65	−1	54	−1	14	−1	0	−1	30	−1	74	−1	0	−1	−1	−1	−1	−1	0	0	−1
[6]		32	−1	0	−1	15	−1	56	−1	85	−1	5	−1	6	−1	52	−1	0	−1	−1	−1	−1	−1	0	0
[7]		−1	0	−1	47	−1	13	−1	61	−1	84	−1	55	−1	78	−1	41	95	−1	−1	−1	−1	−1	−1	0

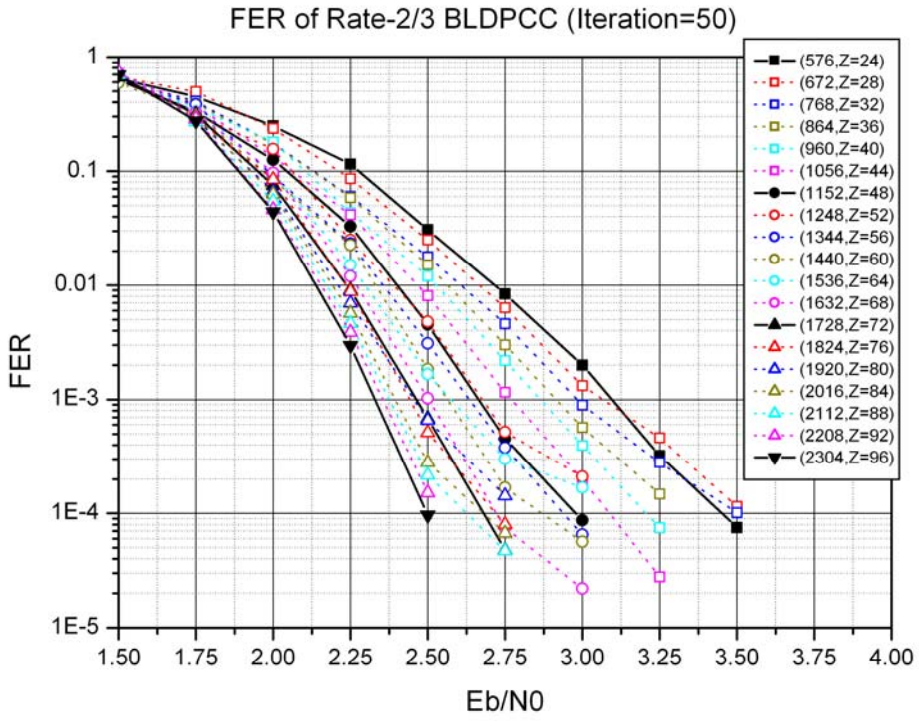
■ Permuted matrix (No two adjacent rows overlap in permuted matrix, including first and last rows)

[0]	2	−1	19	−1	47	−1	48	−1	36	−1	82	−1	47	−1	15	−1	95	0	−1	−1	−1	−1	−1	−1
[3]	−1	28	−1	32	−1	81	−1	27	−1	88	−1	5	−1	56	−1	37	−1	−1	−1	0	0	−1	−1	−1
[6]	32	−1	0	−1	15	−1	56	−1	85	−1	5	−1	6	−1	52	−1	0	−1	−1	−1	−1	−1	0	0
[1]	−1	69	−1	88	−1	33	−1	3	−1	16	−1	37	−1	40	−1	48	−1	0	0	−1	−1	−1	−1	−1
[4]	23	−1	29	−1	15	−1	30	−1	66	−1	24	−1	50	−1	62	−1	−1	−1	−1	−1	0	0	−1	−1
[7]	−1	0	−1	47	−1	13	−1	61	−1	84	−1	55	−1	78	−1	41	95	−1	−1	−1	−1	−1	−1	0
[2]	10	−1	86	−1	62	−1	28	−1	85	−1	16	−1	34	−1	73	−1	−1	−1	0	0	−1	−1	−1	−1
[5]	−1	30	−1	65	−1	54	−1	14	−1	0	−1	30	−1	74	−1	0	−1	−1	−1	−1	−1	0	0	−1

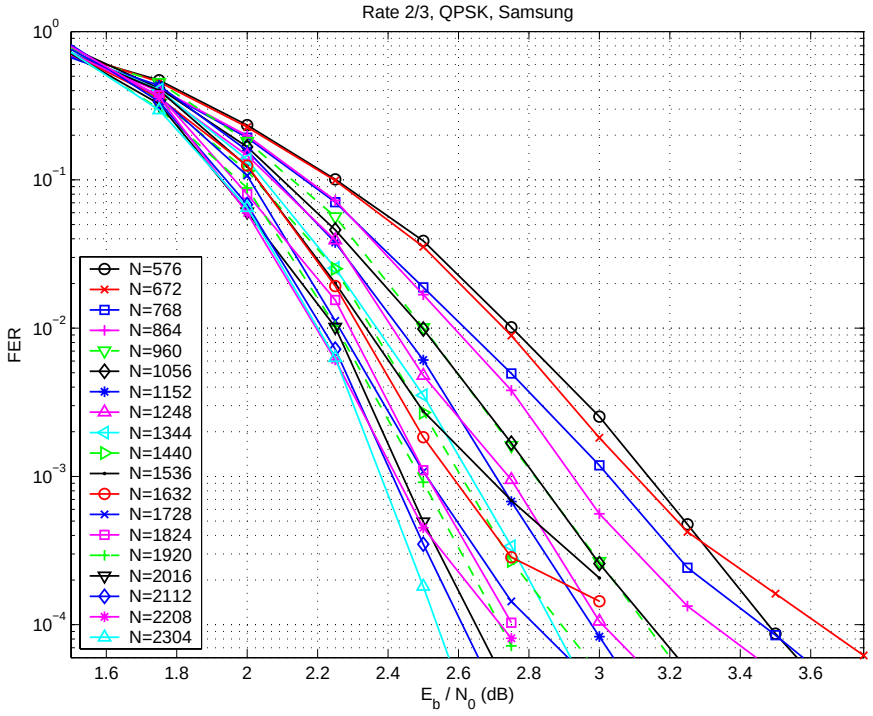
[0]	2	−1	19	−1	47	−1	48	−1	36	−1	82	−1	47	−1	15	−1	X	0	−1	−1	−1	−1	−1	−1

2. cross simulations for R=2/3

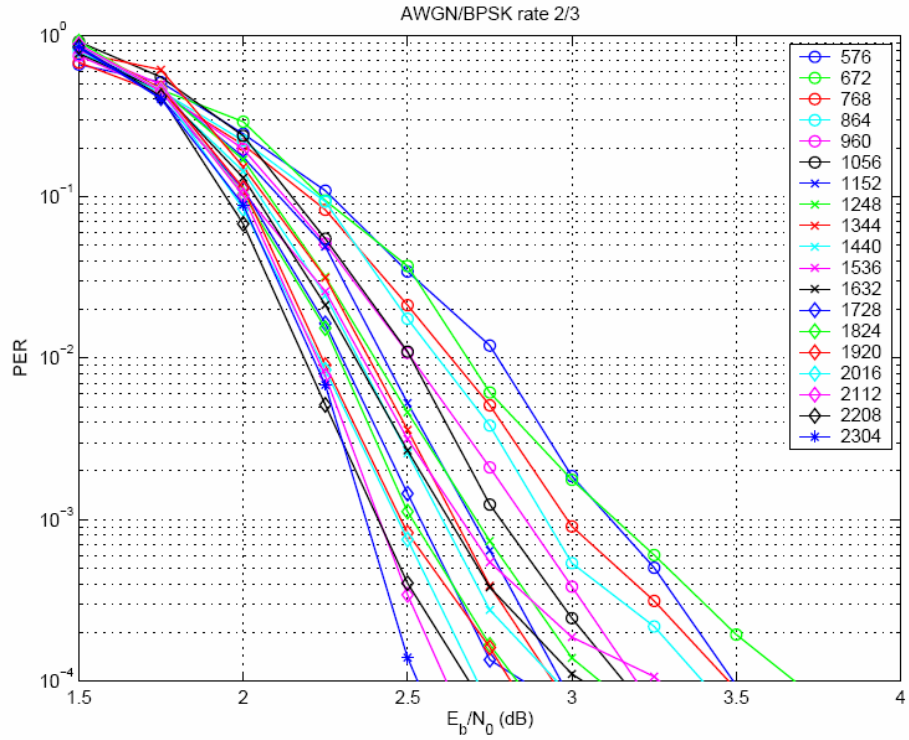
2.1 Samsung simulation for r=2/3 in contribution 006 (severe error floors at N=1632,1536,768,672)



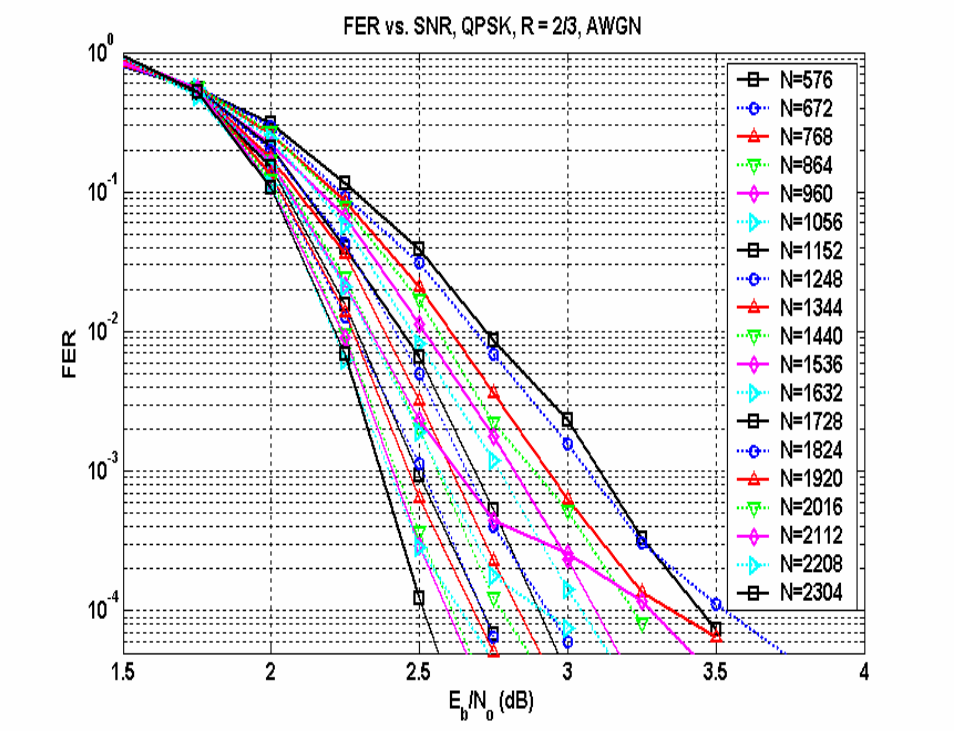
2.2 Motorola cross simulation for new r=2/3 (still severe error floors at N=1632, 1536, 672)



2.3 Intel cross simulation for new r=2/3 (still severe error floors at N=1632, 1536, 672)

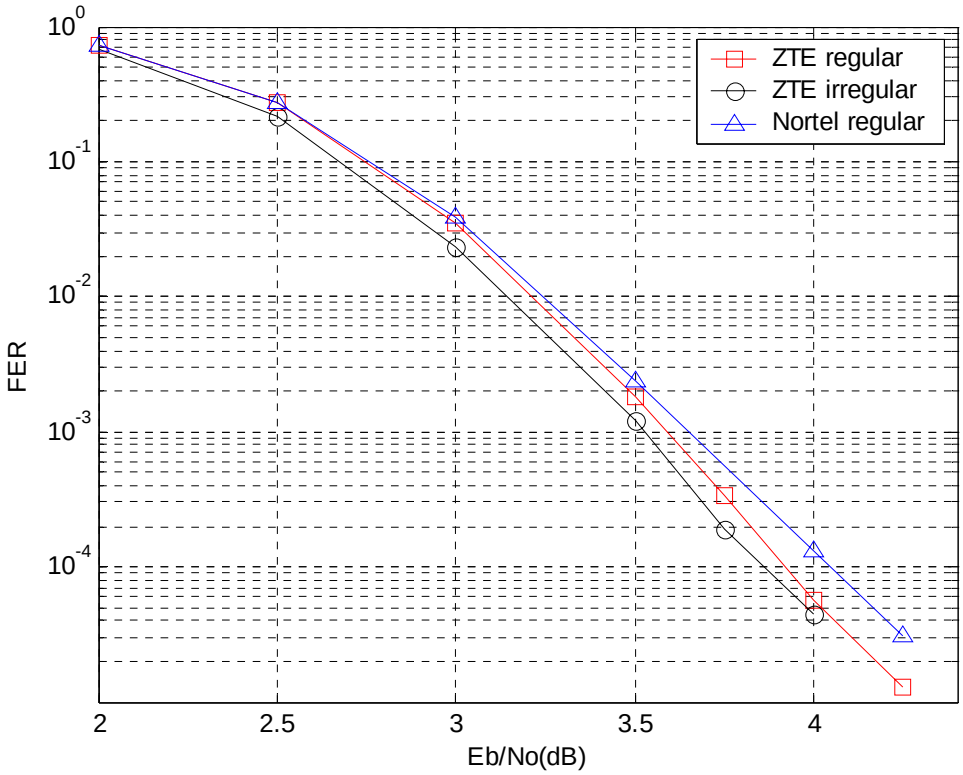


2.4 Nortel cross simulation for new r=2/3 (still severe error floors at N=1632, 1536, 672)

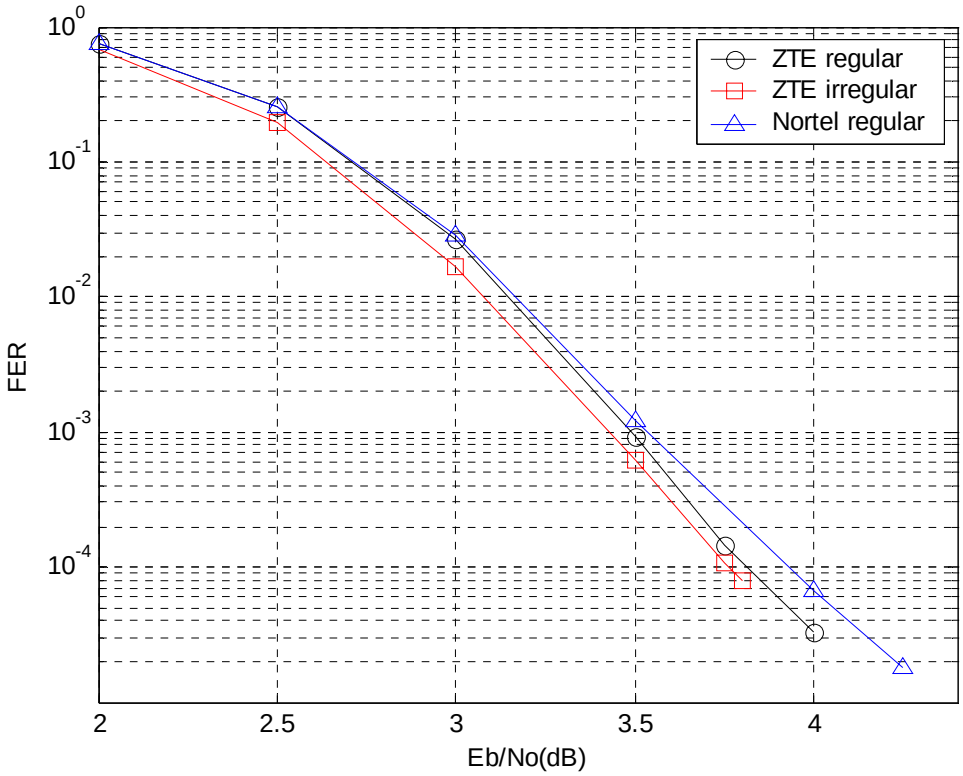


3. Simulation Comparison at R=3/4

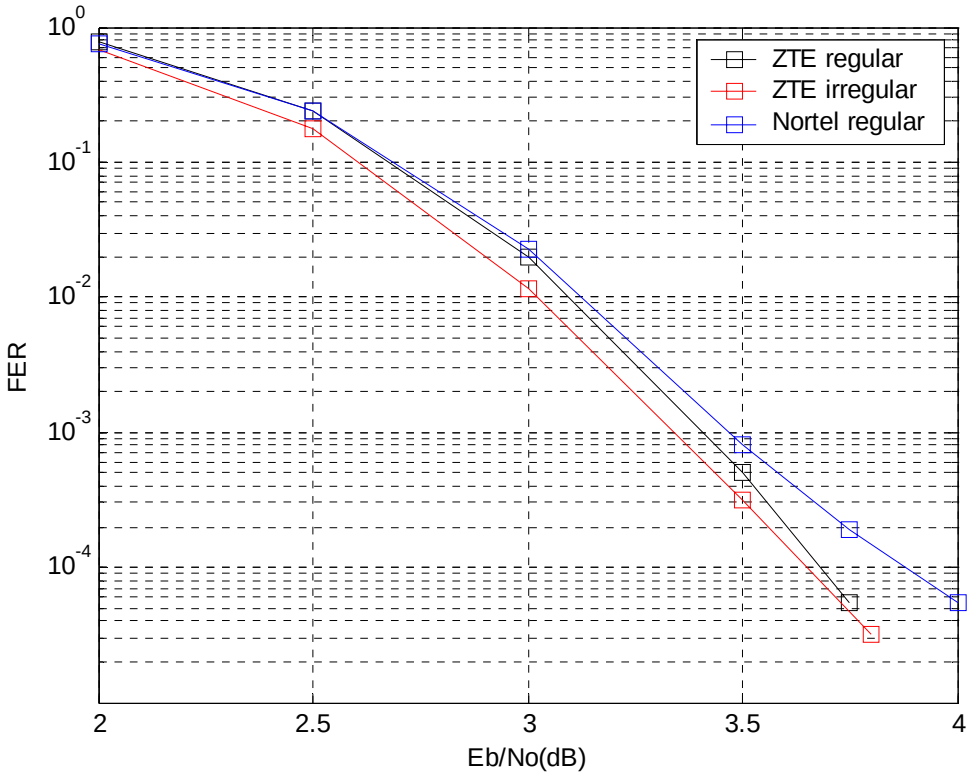
Small Code Size Performance Comparison



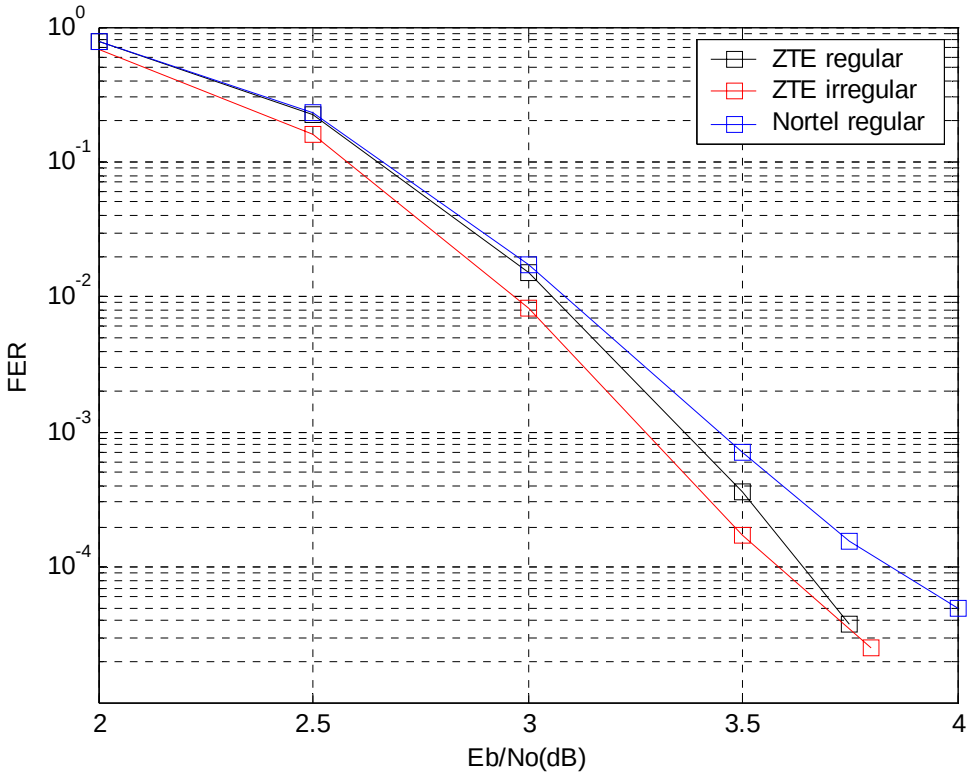
Comparing ZTE codes with Nortel codes when N = 576



Comparing ZTE codes with Nortel codes when N = 672

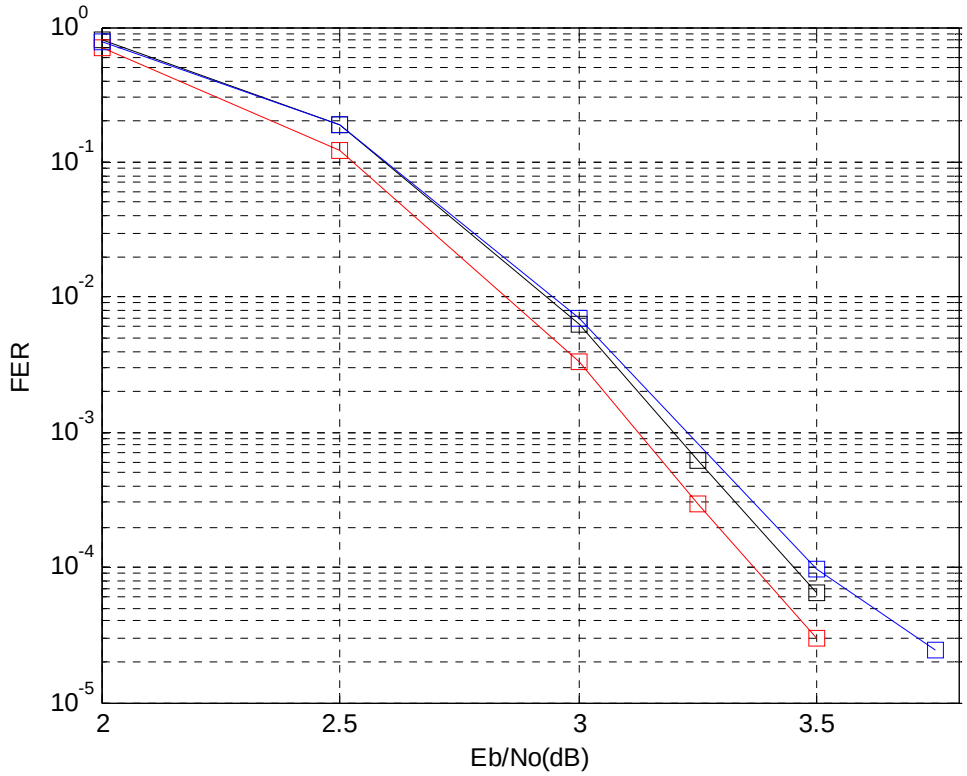


Comparing ZTE codes with Nortel codes when N = 768

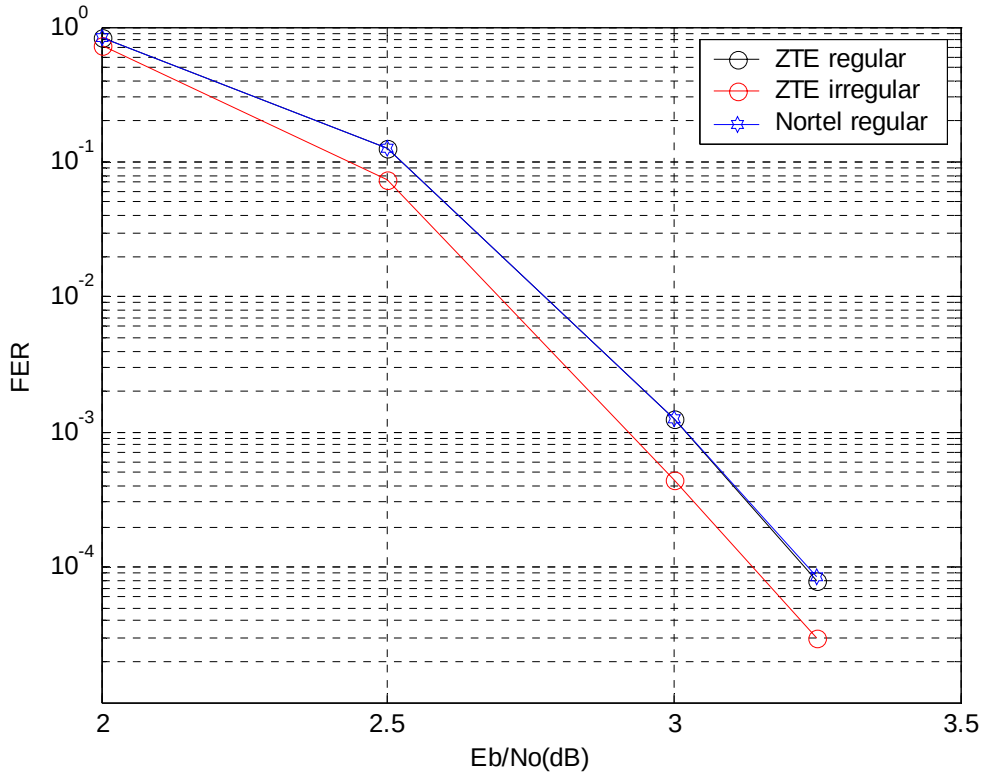


Comparing ZTE codes with Nortel codes when N = 864

Middle Code Size Performance Comparison

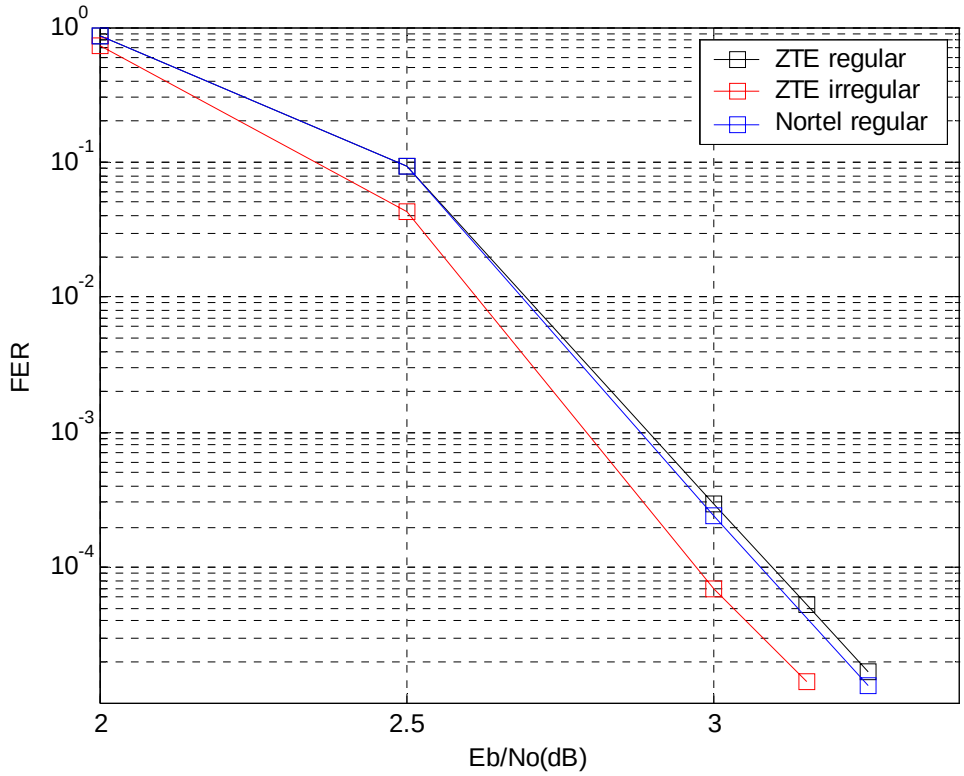


Comparing ZTE codes with Nortel codes when N = 1152



Comparing ZTE codes with Nortel codes when N = 1728

Large Code Size Performance Comparison



Comparing ZTE codes with Nortel codes when N = 2304