Project	**IEEE 802.16 Broadband Wireless Access Working Group <http://ieee802.org/16>**
Title	**Algebraic Low-Density Parity-Check Codes for OFDMA PHY Layer**
Date Submitted	**2004-08-17**
Source(s):	Nina Burns, Aleksandar Purkovic, Sergey Sukobok, Brian Johnson Nortel Networks 20410 Century Blvd Gaithersburg, MD 20874 USA Voice: 1-301-528-3255 Fax: 1-301-528-3298 nburns@nortelnetworks.com
Re:	Contribution in conjunction with comments to IEEE 802.16e
Abstract	This contribution describes construction and encoding of optional LDPC codes for 802.16e OFDMA
Purpose	Support of the comments
Notice	
Release	
Patent Policy and Procedures	

Note to editor: add the following sections and text adjusting the numbering as required. The appendix of this document is not part of the text proposal.

8.4.9.2.4 Algebraic Low-Density Parity-Check Code (optional)

This document describes Nortel Networks' contribution towards inclusion of Low-Density Parity-Check (LDPC) codes in the 802.11e standard.

We describe a particular algebraic construction of a parity check matrix which specifies the LDPC code and outline the encoding procedure for such codes. Frame Error Rate (FER) results are also included for comparison purposes. In addition, we discuss the potential benefits that a structured code may have in hardware implementation.

8.4.9.2.4.1 LDPC Parity Check Matrix Description

8.4.9.2.4.1.1 Base Parity Check Matrix Specification

The LDPC matrix is constructed in two steps, based on the information block size (K) and desired code rate, R=K/N, where N equals the codeword length. In the first step the base matrix is generated whereas in the second step the base matrix is expanded, if necessary.

The base LDPC matrix has the size MxN, where M is the number of check equations, and can be represented in the form:

$$H = [H^p \mid H_1^d \mid H_2^d \mid H_3^d]$$

All of the component sub-matrices, H^p, H_1^d, H_2^d, H_3^d are square matrices of the size MxM. The modular structure of this matrix is used to generate codes with a series of code rates: 1/2, 2/3 and 3/4. This is illustrated below, where only a portion of the whole matrix is used (outlined in bold letters) for rates less than 3/4:

R = 1/2: $\boldsymbol{H} = [\boldsymbol{H}^{\boldsymbol{p}} | \boldsymbol{H}_{\boldsymbol{1}}^{\boldsymbol{d}} \mid H_2^d \mid H_3^d]$

R = 2/3: $\boldsymbol{H} = [\boldsymbol{H}^{\boldsymbol{p}} | \boldsymbol{H}_{\boldsymbol{1}}^{\boldsymbol{d}} \mid \boldsymbol{H}_{\boldsymbol{2}}^{\boldsymbol{d}} \mid H_3^d]$

R = 3/4: $\boldsymbol{H} = [\boldsymbol{H}^{\boldsymbol{p}} | \boldsymbol{H}_{\boldsymbol{1}}^{\boldsymbol{d}} \mid \boldsymbol{H}_{\boldsymbol{2}}^{\boldsymbol{d}} \mid \boldsymbol{H}_{\boldsymbol{3}}^{\boldsymbol{d}}]$

Matrix H^p is in a "dual diagonal" form. There is always only one sub-matrix H^p in the parity check matrix H, corresponding to the parity check bits of the codeword.

Matrices H_i^d, $i = 1,2,3$ are constructed by using a modification of the π-rotation technique, originally described in [1]. Each of the matrices H_i^d can be represented in the following form:

$$\boldsymbol{H}_{\boldsymbol{i}}^{\boldsymbol{d}} = \begin{bmatrix} \pi_{A,i}\pi_{B,i}\pi_{C,i}\pi_{D,i} \\ \pi_{B,i}\pi_{C,i}\pi_{D,i}\pi_{A,i} \\ \pi_{C,i}\pi_{D,i}\pi_{A,i}\pi_{B,i} \\ \pi_{D,i}\pi_{A,i}\pi_{B,i}\pi_{C,i} \end{bmatrix} \boldsymbol{i} = 1,2,3$$

Figure 1. Structure of the matrix H_i^d

$\pi_{A,i}$ is a 24x24 permutation matrix. $\pi_{B,i}, \pi_{C,i}$, and $\pi_{D,i}$ are obtained by 90°, 180°, and 270° counterclockwise rotations of the permutation matrix $\pi_{A,i}$, respectively. The variable m=24 is used to describe the size of the permutation matrices. It follows that 4m=M.

As one can see, each H_i^d is fully specified by $\pi_{A,i}$ which in turn is described by a length-24 vector. Therefore the entire LDPC matrix is fully specified by three length-24 vectors from which the component sub-matrices are generated:

A,1 = *[6 5 10 24 8 7 17 23 11 12 22 2 14 18 15 4 19 21 20 1 9 16 13 3]*

A,2 = *[20 19 24 5 9 7 17 12 4 13 21 11 15 23 16 8 6 14 2 10 22 3 18 1]*

A,3 = *[21 19 7 3 11 14 20 4 18 13 5 23 16 12 22 10 9 6 8 17 1 24 2 15]*

Mapping between the permutation vector ***A,i*** and the permutation matrix $\pi_{A,i}$ is as follows: the first element of the row vector ***A,i*** specifies the position of a "1" in the first column of $\pi_{A,i}$, (counting from the bottom), the second element of the row vector ***A,i*** specifies the position of a "1" in the second column of $\pi_{A,i}$, and so on. An example of deriving the permutation matrix $\pi_{A,1}$ from the vector ***A,1*** is shown in Figure 2. The matrices $\pi_{A,2}$ and $\pi_{A,3}$ are derived in a similar manner and rotated to form the corresponding $\pi_{B,i}$, $\pi_{C,i}$ and $\pi_{D,i}$ matrices (i=2,3). These matrices are concatenated as explained previously to form the base matrix.

$\pi_{A,1} =$

0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 2. Structure of the matrix $\pi_{A,1}$

8.4.9.2.4.1.2 Expansion of the Base Parity Check Matrix

If there is a need to accommodate a larger block size, the base parity check matrix may be expanded by an integer factor *L*, where *L* is an integer between 1 and 12.

Expansion of H^p is done by replacing each "0" element by an *L*x*L* zero matrix, $\mathbf{0}_{LxL}$, and each "1" element by an *L*x*L* identity matrix, $\mathbf{I}_{LxL}$.

Expansion of $\boldsymbol{H}^{\boldsymbol{d}}$ is done by replacing each "0" element by an *L*x*L* zero matrix, $\mathbf{0}_{LxL}$, and each "1" element by a rotated version of an *L*x*L* identity matrix, $\mathbf{I}_{LxL}$. The rotation order, *s* (number of circular shifts to the right, for example) can be determined by using a simple rule: $s = (r * c)_{\text{mod } L}$, where *r* and *c* correspond to the row and column index of the particular "1" element, respectively. Row indices, r, are in the range [1,M] and column indices, c, are in the range [M+1,N].

8.4.9.2.4.2 Encoding

The described structure of the LDPC parity check matrix enables a simple encoding algorithm. In general, by the definition of the parity check matrix, the following holds:

$\mathbf{H}_{MxN}\mathbf{c}_{Nx1}=\mathbf{0}_{Mx1}$,

where **H** is the parity check matrix, $\mathbf{0}_{Mx1}$, an all-zero Mx1 vector, and **c** is a codeword vector given by:

$\mathbf{c} = [p_1,p_2,\ldots,p_M,d_1,d_2,\ldots,d_K]^T$

In the expression above, $p_1,p_2,\ldots,p_M$ represent parity bits (redundancy part) whereas $d_1,d_2,\ldots,d_K$ represent information bits (systematic part). Due to the dual diagonal structure of the sub-matrix H^p, parity bits can be computed recursively as:

$p_M = \sum_{n=1}^{4q} h_{M,M+k_n^M} d_{k_n^M}$, where $\mathrm{M}+k_n^M$ is the index of the column in which row M contains a "1"

$p_{M-1} = p_M + \sum_{n=1}^{4q} h_{M-1,M+k_n^{M-1}} d_{k_n^{M-1}}$, where $\mathrm{M}+k_n^{M-1}$ is the index of the column in which row M-1 contains a "1"

…

$p_1 = p_2 + \sum_{n=1}^{4q} h_{1,M+k_n^1} d_{k_n^1}$, where $\mathrm{M}+k_n^1$ is the index of the column in which row 1 contains a "1"

where $\boldsymbol{k}_{\boldsymbol{n}}^1, \boldsymbol{k}_{\boldsymbol{n}}^2, \ldots, \boldsymbol{k}_{\boldsymbol{n}}^{\boldsymbol{M-1}}, \boldsymbol{k}_{\boldsymbol{n}}^{\boldsymbol{M}}$ and $h_{r,c}$ are non-zero elements (=1) of the parity check matrix in the portion corresponding to H_i^d's. Therefore, the encoding operation can be performed by using only simple AND and XOR circuits.

For the cases when the information block size is greater than 144, 192 and 216 bytes for R=1/2, 2/3 and 3/4, respectively, concatenation of the code words is employed. Based on the rate and number of information bytes to be sent, the required number of code words is calculated. The amount of zero padding (or shortening) is divided as equally as possible among the total number of code word blocks. This way, each code word block has the same effective rate that approaches the desired rate.

Appendix A

A.1 Performance

In this section we present our simulation results obtained using the consensus criteria: AWGN channel, BPSK modulation, using a generic floating point sum-product decoding with a maximum of 50 iterations. Results are presented for rates 1/2, 2/3, and 3/4. For each rate, results for all supported block lengths are presented. Table 1 summarizes the supported rates and block lengths.

LDPC code block size support

N: encoder output
Nsch: number of subchannels to be mapped
K: encoder input (bytes)

QPSK (N in bits is a multiple of 48x2)

N (bytes)	N (bits)	Nsch	K (R=1/2)	K (R=2/3)	K (R=3/4)	Expansion factor L		
24	192	2	12	16	18	1		
36	288	3	18	24	27		1	
48	384	4	24	32	36	2		1
60	480	5	30	40	45			
72	576	6	36	48	54	3	2	
84	672	7	42	56	63			
96	768	8	48	64	72	4		2
108	864	9	54	72	81		3	
120	960	10	60	80	90	5		
132	1056	11	66	88	99			
144	1152	12	72	96	108	6	4	3
156	1248	13	78	104	117			
168	1344	14	84	112	126	7		
180	1440	15	90	120	135		5	
192	1536	16	96	128	144	8		4
204	1632	17	102	136	153			
216	1728	18	108	144	162	9	6	
228	1824	19	114	152	171			
240	1920	20	120	160	180	10		5
252	2016	21	126	168	189		7	
264	2112	22	132	176	198	11		
276	2208	23	138	184	207			
288	2304	24	144	192	216	12	8	6

16-QAM (N in bits is a multiple of 48x4)

N (bytes)	N (bits)	Nsch	K (R=1/2)	K (R=2/3)	K (R=3/4)			
24	192	1	12	16	18	1		
48	384	2	24	32	36	2		1
72	576	3	36	48	54	3	2	
96	768	4	48	64	72	4		2
120	960	5	60	80	90	5		
144	1152	6	72	96	108	6	4	3
168	1344	7	84	112	126	7		
192	1536	8	96	128	144	8		4
216	1728	9	108	144	162	9	6	
240	1920	10	120	160	180	10		5
264	2112	11	132	176	198	11		
288	2304	12	144	192	216	12	8	6

64-QAM (N in bits is a multiple of 48x6)

N (bytes)	N (bits)	Nsch	K (R=1/2)	K (R=2/3)	K (R=3/4)			
36	288	1	18	24	27		1	
72	576	2	36	48	54	3	2	
108	864	3	54	72	81		3	
144	1152	4	72	96	108	6	4	3
180	1440	5	90	120	135		5	
216	1728	6	108	144	162	9	6	
252	2016	7	126	168	189		7	
288	2304	8	144	192	216	12	8	6

Table 1. Supported information lengths and rates. The shaded portions of the leftmost table correspond to the lengths and rates that are required for comparison. The lighter shaded portions of the center table correspond to the combinations supported by the proposed code. The darker shaded portions of the center table indicate the overlap between the left and right tables. The rightmost table shows which expansion factor was applied.

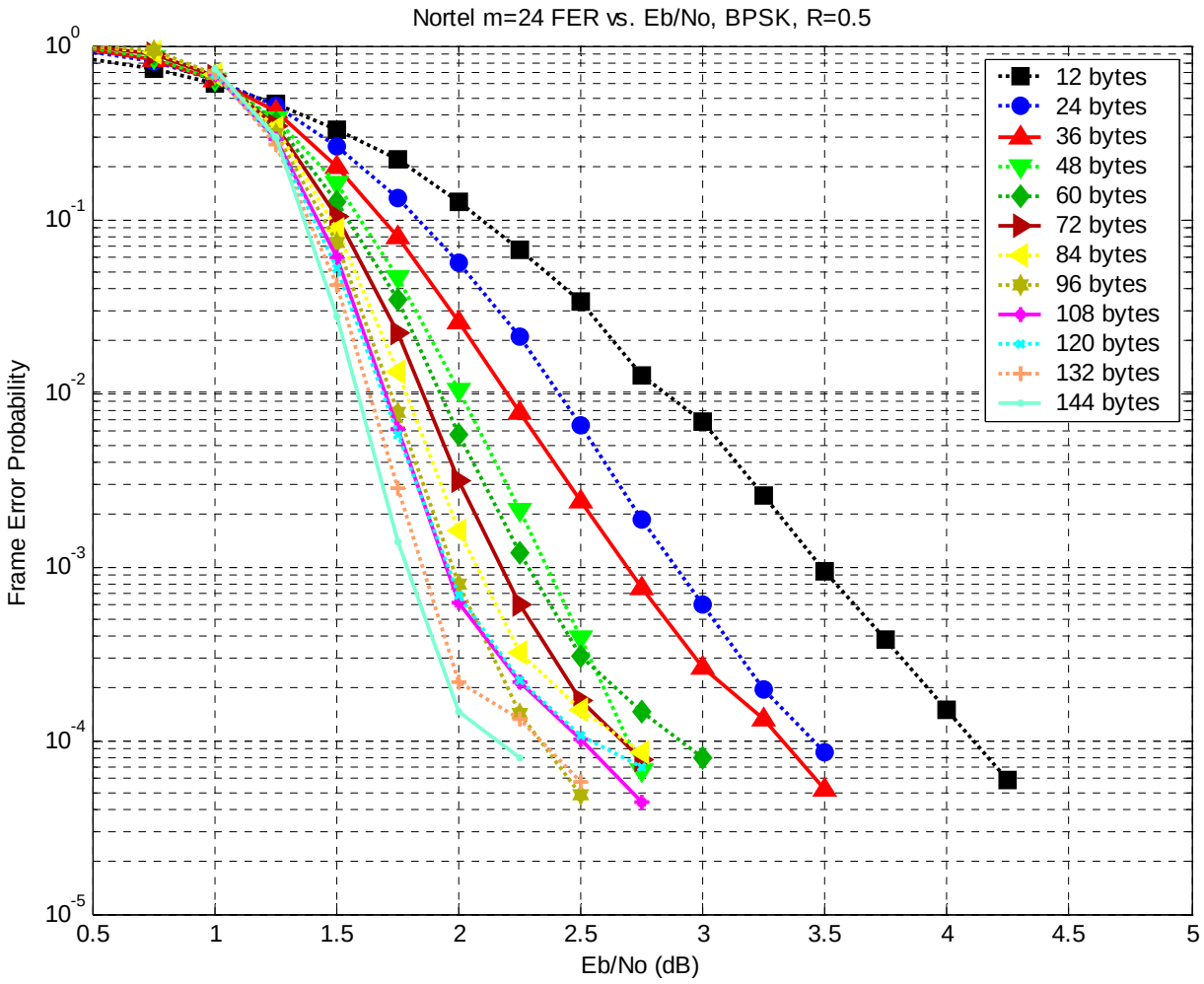


Figure 3. FER vs. Eb/No, AWGN R=1/2. Information lengths are in bytes.

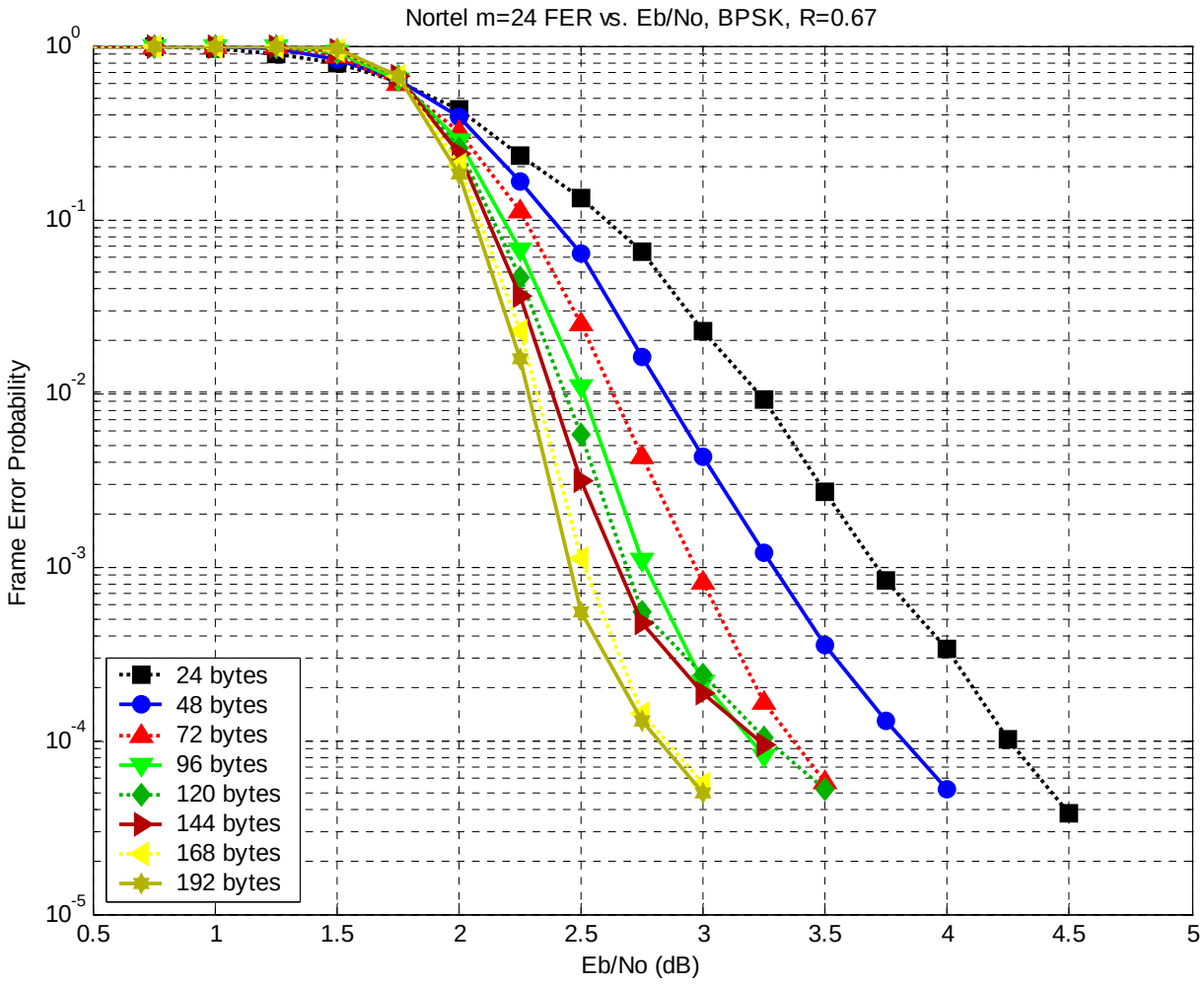


Figure 4. FER vs. Eb/No, AWGN R=2/3. Information lengths are in bytes.

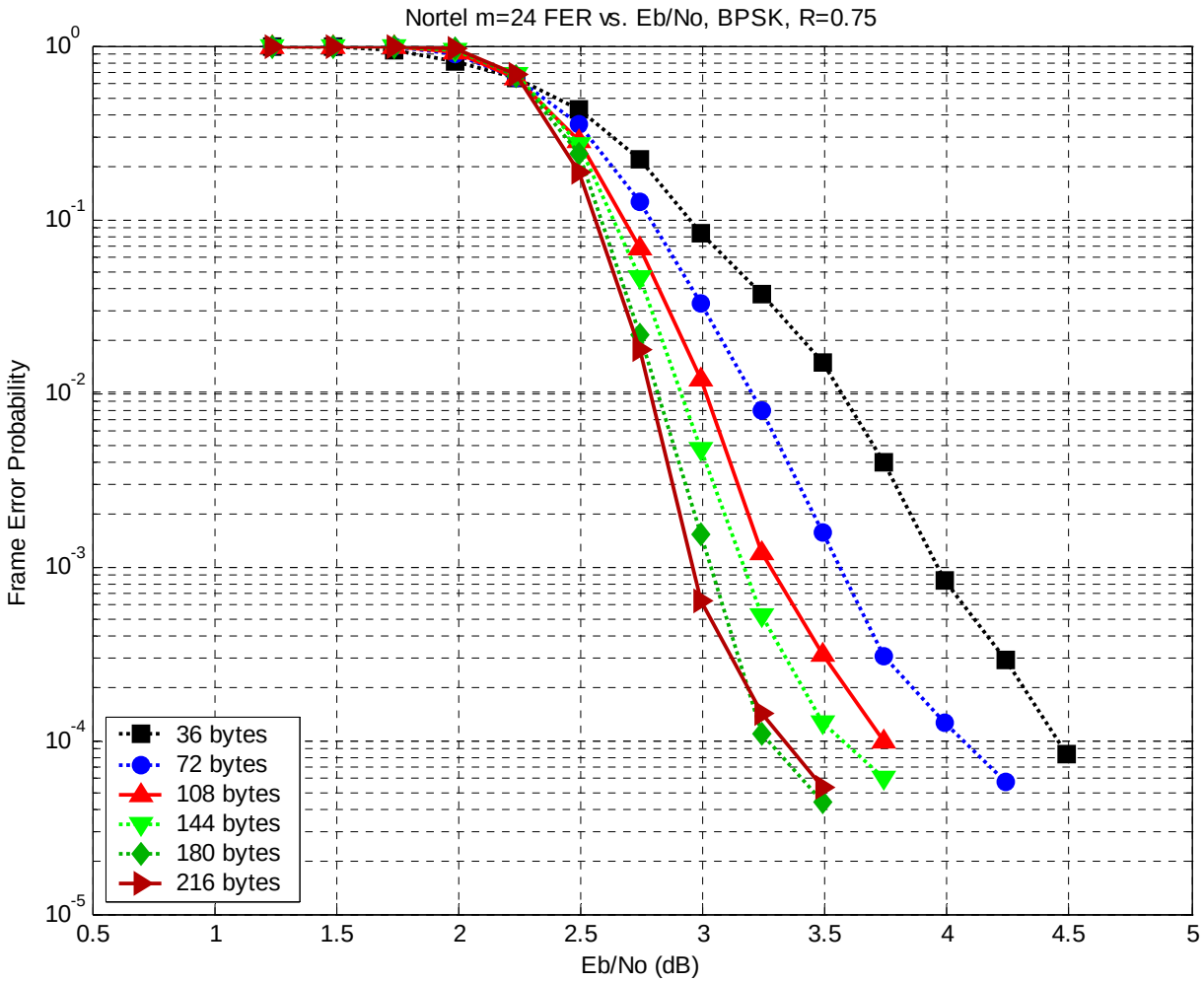


Figure 5. FER vs. Eb/No, AWGN R=3/4. Information lengths are in bytes.

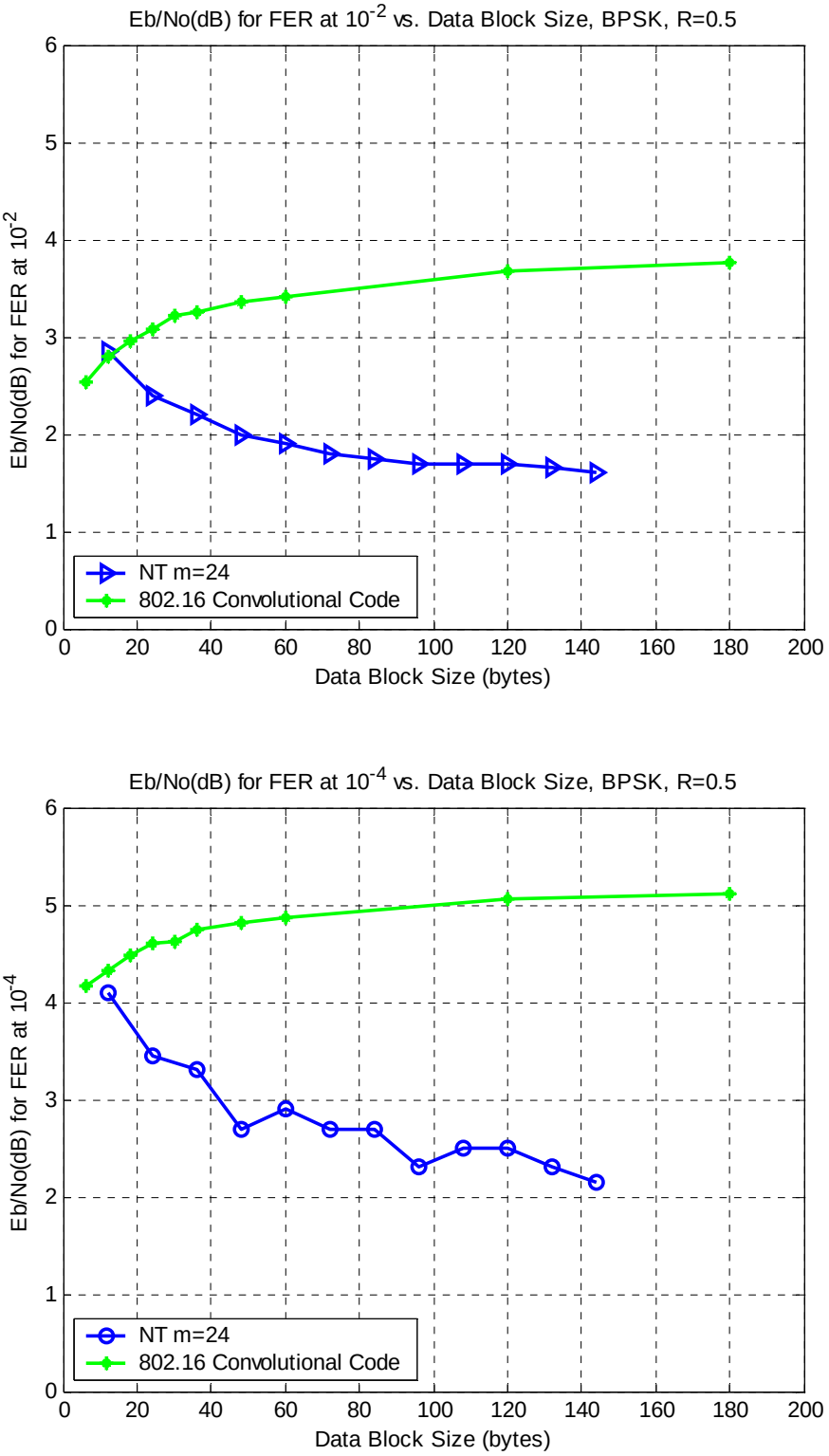


Figure 6. SNRs for FER=10^{-2} and 10^{-4} vs. information length, R=0.5

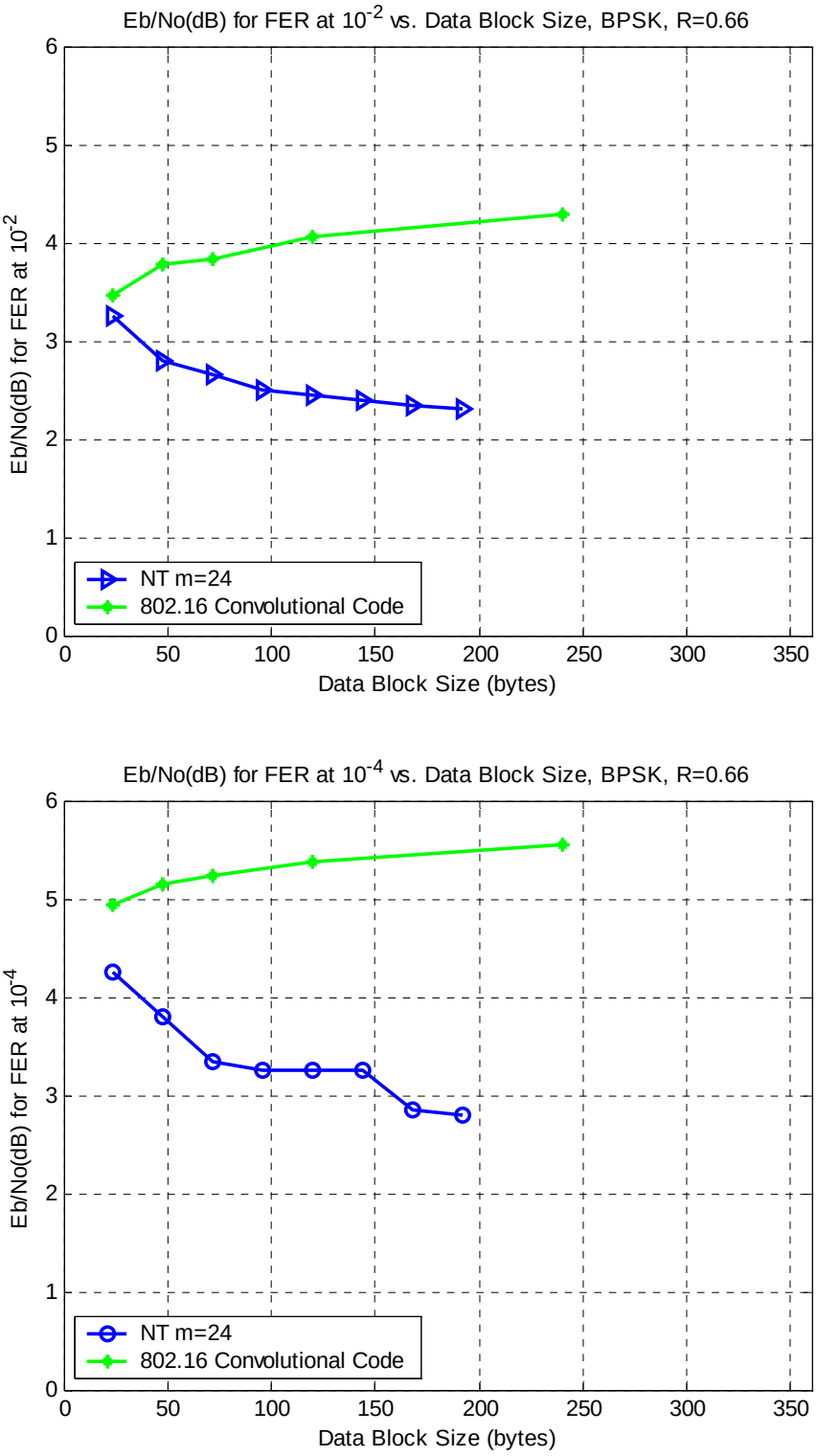
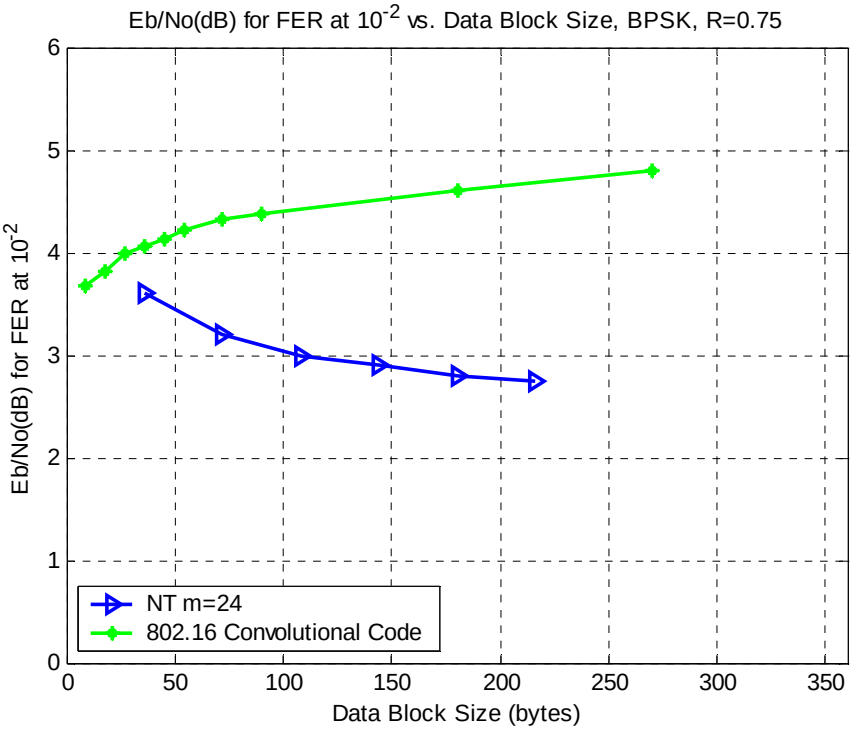


Figure 7. SNRs for FER=10^{-2} and 10^{-4} vs. information length, R=0.66



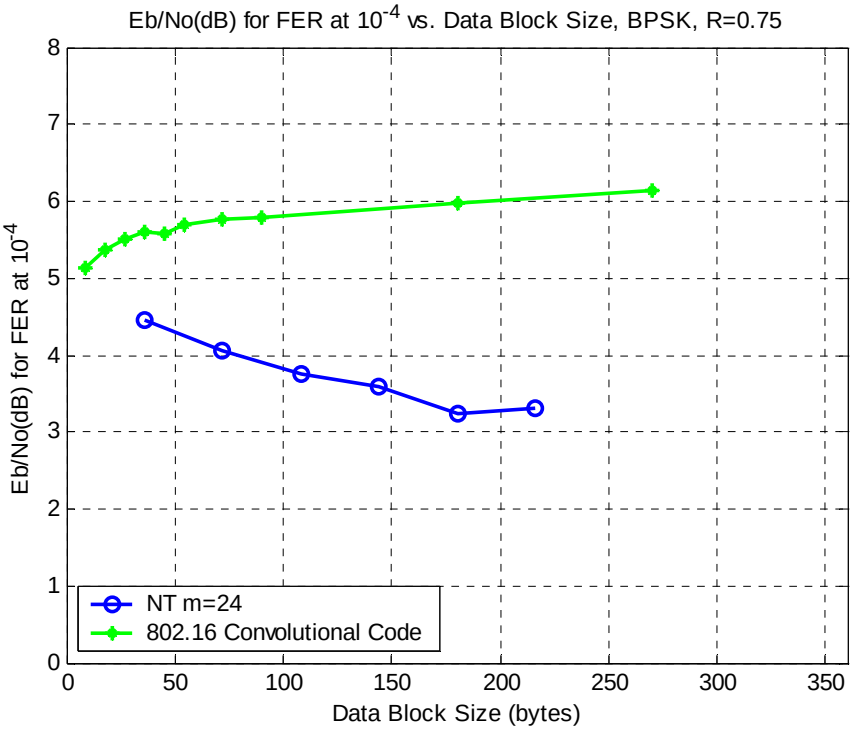


Figure 8. SNRs for FER=10^{-2} and 10^{-4} vs. information length, R=0.75

A.2 Hardware Implementation Considerations

The main features of the proposed LDPC code are its performance *and* its deterministic structure. In general randomized, irregular matrices outperform regular, structured matrices [2]. But with careful design and searching, we have found codes with competitive performance that also lend themselves to simpler hardware implementations. Specifically, the construction based on permutation matrices reduces the wiring and routing requirements. In addition, the pi-rotation structure offers the following:

- further simplified encoding circuits
- more straightforward fixed-point implementation
- possibility of hardware re-use among portions of the matrix

The following is a description of the encoder and decoder hardware simplifications.

A.2.1 Encoder Complexity Reduction

This technique was first described in [1] and we describe it here for ½-rate matrices for simplicity. The codeword $\boldsymbol{c}$ can be represented in systematic form corresponding to the parity check matrices as $\boldsymbol{c}=[\boldsymbol{p} \mid \boldsymbol{d}]$ and consists of the parity vector $\boldsymbol{p}$ and the source vector $\boldsymbol{d}$. Assuming a systematic code we define projection vector $\boldsymbol{v}$ from the equations below.

$$\boldsymbol{H}^{p} \cdot \boldsymbol{p} = \boldsymbol{H}^{d} \cdot \boldsymbol{d} = \boldsymbol{v} \qquad \text{(A.2.1)}$$

Now the encoding process can be split into two steps:

- Calculation of the projection vector
- Calculation of the parity bits

Note that once the projection vector is calculated, the parity bits are computed using the back-substitution method. Representing the information vector $\boldsymbol{d}$ as four sub-vectors (or segments of length $m=24$) $\boldsymbol{[d_1\ d_2\ d_3\ d_4]}$, we can write the following equation:

$$\begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \boldsymbol{v}_3 \\ \boldsymbol{v}_4 \end{bmatrix} = \begin{bmatrix} \pi_{\mathrm{A}} & \pi_{\mathrm{B}} & \pi_{\mathrm{C}} & \pi_{\mathrm{D}} \\ \pi_{\mathrm{B}} & \pi_{\mathrm{C}} & \pi_{\boldsymbol{D}} & \pi_{\mathrm{A}} \\ \pi_{\mathrm{C}} & \pi_{\mathrm{D}} & \pi_{\mathrm{A}} & \pi_{\mathrm{B}} \\ \pi_{\mathrm{D}} & \pi_{\mathrm{A}} & \pi_{\mathrm{B}} & \pi_{\mathrm{C}} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \\ \boldsymbol{d}_3 \\ \boldsymbol{d}_4 \end{bmatrix} \qquad \text{(A.2.2)}$$

Now we can derive an equation for each segment of $\boldsymbol{v}$ as follows:

$$\begin{aligned} \boldsymbol{v}_1 &= \pi_{\boldsymbol{A}} * \boldsymbol{d}_1 + \pi_{\boldsymbol{B}} * \boldsymbol{d}_2 + \pi_{\boldsymbol{D}} * \boldsymbol{d}_3 + \pi_{\boldsymbol{C}} * \boldsymbol{d}_4 \\ \boldsymbol{v}_2 &= \pi_{\boldsymbol{B}} * \boldsymbol{d}_1 + \pi_{\boldsymbol{C}} * \boldsymbol{d}_2 + \pi_{\boldsymbol{D}} * \boldsymbol{d}_3 + \pi_{\boldsymbol{A}} * \boldsymbol{d}_4 \\ \boldsymbol{v}_3 &= \pi_{\boldsymbol{C}} * \boldsymbol{d}_1 + \pi_{\boldsymbol{D}} * \boldsymbol{d}_2 + \pi_{\boldsymbol{A}} * \boldsymbol{d}_3 + \pi_{\boldsymbol{B}} * \boldsymbol{d}_4 \\ \boldsymbol{v}_4 &= \pi_{\boldsymbol{D}} * \boldsymbol{d}_1 + \pi_{\boldsymbol{A}} * \boldsymbol{d}_2 + \pi_{\boldsymbol{B}} * \boldsymbol{d}_3 + \pi_{\boldsymbol{C}} * \boldsymbol{d}_4 \end{aligned} \qquad \text{(A.2.3)}$$

The projection vector can then be calculated by sequentially shifting the info sub-segments d_i clockwise around a single permutation matrix. Figure 9 depicts this in a diagram.

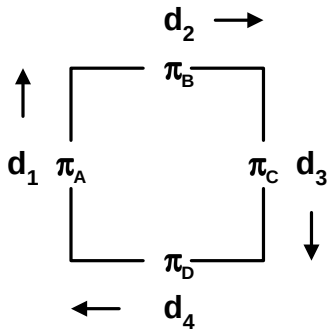


Figure 9. Description of π-rotation encoding.

The circuit complexity is reduced by using the symmetries of the π–rotation structure. The efficiency of the encoding process can be increased using a pipelined encoder architecture.

A.2.2 Decoder Complexity Reduction

The challenges in implementation of a sum-product decoder lie in the circuit representation of the parity check matrix. A fully randomized parity check matrix must incorporate a wiring network that can connect any bit node to any check node and vice-versa. However, a parity matrix built from permutation matrices such as the one we propose can reduce the complexity of the interconnect by *O(n)* [3]. Let m represent the square size of permutation matrices, as in our proposed code. Since each row of the permutation matrices $H_{d,i}$ has a single nonzero element, only m:1 instead of N:1 multiplexers are needed to implement connectivity. This is also true for the de-multiplexers in the reverse direction.

The pi-rotation structure of our matrix consists of repeated permutation matrices within individual $H_{d,i}$'s (Figure 10). We suggest that this repetition may be an advantage when designing circuitry for implementation.

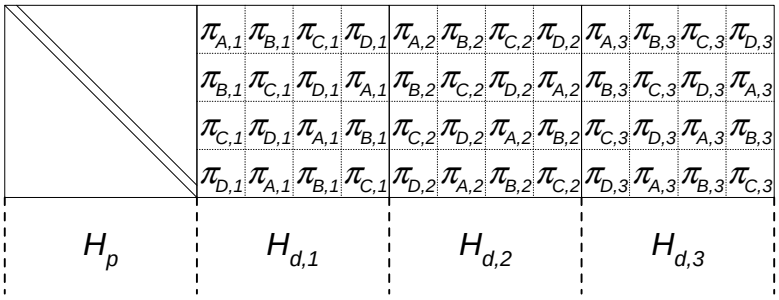


Figure 10. Diagram of parity check matrix showing repeated permutation matrices in its structure.

Also noteworthy is the regular row and column weights of our proposed matrix. With the exception of 1 row and column in our base matrix, all column weights are 2 or 4 and all row weights are 6 (for the ½ rate code). The sum-product algorithm requires summations over the number of column and row weights. If the number of addends vary widely, the circuitry must use wider accumulators to prevent overflow for those cases. Having a limited number row and column weights allows for a more efficient and straightforward implementation.

A.3 References

[1] R. Echard et al, “The π-rotation low-density parity check codes,” In Proc. GLOBECOM 2001, pp. 980-984, Nov. 2001.
[2] T. J. Richardson, M. A. Shokrollahi and R. L. Urbanke, “Design of Capacity-Approaching Irregular Low-Density Parity Check Codes”,IEEE Trans. Info. Theory, Vol. 47, No. 2, February 2001.
[3] M.M.Mansour and N.R.Shanbhag, “High-Throughput LDPC Decoders,” IEEE Trans. On VLSI Systems, vol. 11, No 6, pp. 976-996, December 2003.

A.4 Informative Example

The following matrix description is provided as a construction reference. The matrix is in the standard distribution format and represents the proposed matrix (R=3/4) with expansion factor L=2.

```
768 192
1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4
1
2
1 3
2 4
3 5
4 6
5 7
6 8
7 9
8 10
9 11
10 12
11 13
12 14
13 15
14 16
15 17
16 18
17 19
18 20
19 21
20 22
21 23
22 24
```

23 25
24 26
25 27
26 28
27 29
28 30
29 31
30 32
31 33
32 34
33 35
34 36
35 37
36 38
37 39
38 40
39 41
40 42
41 43
42 44
43 45
44 46
45 47
46 48
47 49
48 50
49 51
50 52
51 53
52 54
53 55
54 56
55 57
56 58
57 59
58 60
59 61
60 62
61 63
62 64
63 65
64 66
65 67
66 68
67 69
68 70
69 71
70 72
71 73
72 74
73 75
74 76
75 77
76 78
77 79
78 80
79 81
80 82
81 83
82 84
83 85
84 86
85 87
86 88

87 89
88 90
89 91
90 92
91 93
92 94
93 95
94 96
95 97
96 98
97 99
98 100
99 101
100 102
101 103
102 104
103 105
104 106
105 107
106 108
107 109
108 110
109 111
110 112
111 113
112 114
113 115
114 116
115 117
116 118
117 119
118 120
119 121
120 122
121 123
122 124
123 125
124 126
125 127
126 128
127 129
128 130
129 131
130 132
131 133
132 134
133 135
134 136
135 137
136 138
137 139
138 140
139 141
140 142
141 143
142 144
143 145
144 146
145 147
146 148
147 149
148 150
149 151
150 152

```
151 153
152 154
153 155
154 156
155 157
156 158
157 159
158 160
159 161
160 162
161 163
162 164
163 165
164 166
165 167
166 168
167 169
168 170
169 171
170 172
171 173
172 174
173 175
174 176
175 177
176 178
177 179
178 180
179 181
180 182
181 183
182 184
183 185
184 186
185 187
186 188
187 189
188 190
189 191
190 192
38 90 102 183
37 89 101 184
39 81 121 167
40 82 122 168
30 75 127 191
29 76 128 192
1 61 113 175
2 62 114 176
34 59 98 147
33 60 97 148
35 63 135 145
36 64 136 146
15 70 138 155
16 69 137 156
3 83 133 153
4 84 134 154
27 54 103 186
28 53 104 185
25 67 125 149
26 68 126 150
6 71 131 162
5 72 132 161
45 51 123 163
46 52 124 164
```

```
22 78 99 190
21 77 100 189
13 79 139 169
14 80 140 170
19 91 119 174
20 92 120 173
41 55 117 187
42 56 118 188
11 87 142 158
12 88 141 157
7 85 129 171
8 86 130 172
10 95 110 178
9 96 109 177
47 93 111 181
48 94 112 182
31 66 143 179
32 65 144 180
17 49 115 165
18 50 116 166
23 74 106 159
24 73 105 160
43 57 107 151
44 58 108 152
42 54 135 182
41 53 136 181
33 73 119 183
34 74 120 184
27 79 143 174
28 80 144 173
13 65 127 145
14 66 128 146
11 50 99 178
12 49 100 177
15 87 97 179
16 88 98 180
22 90 107 159
21 89 108 160
35 85 105 147
36 86 106 148
6 55 138 171
5 56 137 172
19 77 101 169
20 78 102 170
23 83 114 150
24 84 113 149
3 75 115 189
4 76 116 190
30 51 142 166
29 52 141 165
31 91 121 157
32 92 122 158
43 71 126 163
44 72 125 164
7 69 139 185
8 70 140 186
39 94 110 155
40 93 109 156
37 81 123 151
38 82 124 152
47 62 130 154
48 61 129 153
45 63 133 191
46 64 134 192
```

```
18 95 131 175
17 96 132 176
1 67 117 161
2 68 118 162
26 58 111 167
25 57 112 168
9 59 103 187
10 60 104 188
6 87 134 186
5 88 133 185
25 71 135 177
26 72 136 178
31 95 126 171
32 96 125 172
17 79 97 157
18 80 98 158
2 51 130 155
1 52 129 156
39 49 131 159
40 50 132 160
42 59 111 166
41 60 112 165
37 57 99 179
38 58 100 180
7 90 123 150
8 89 124 149
29 53 121 163
30 54 122 164
35 66 102 167
36 65 101 168
27 67 141 147
28 68 142 148
3 94 118 174
4 93 117 173
43 73 109 175
44 74 110 176
23 78 115 187
24 77 116 188
21 91 137 151
22 92 138 152
46 62 107 183
45 61 108 184
33 75 103 181
34 76 104 182
14 82 106 191
13 81 105 192
15 85 143 189
16 86 144 190
47 83 127 162
48 84 128 161
19 69 113 145
20 70 114 146
10 63 119 170
9 64 120 169
11 55 139 153
12 56 140 154
39 86 138 150
40 85 137 149
23 87 129 169
24 88 130 170
47 78 123 175
48 77 124 176
31 49 109 161
32 50 110 162
```

```
3 82 107 146
4 81 108 145
1 83 111 183
2 84 112 184
11 63 118 186
12 64 117 185
9 51 131 181
10 52 132 182
42 75 102 151
41 76 101 152
5 73 115 173
6 74 116 174
18 54 119 179
17 53 120 180
19 93 99 171
20 94 100 172
46 70 126 147
45 69 125 148
25 61 127 187
26 62 128 188
30 67 139 167
29 68 140 168
43 89 103 165
44 90 104 166
14 59 135 190
13 60 136 189
27 55 133 177
28 56 134 178
34 58 143 158
33 57 144 157
37 95 141 159
38 96 142 160
35 79 114 191
36 80 113 192
21 65 97 163
22 66 98 164
15 71 122 154
16 72 121 153
7 91 105 155
8 92 106 156
10 91 98 191
9 92 97 192
11 69 131 181
12 70 132 182
2 55 102 187
1 56 101 188
39 75 139 161
40 76 140 162
31 95 115 151
32 96 116 152
35 93 99 177
36 94 100 178
15 51 123 155
16 52 124 156
25 83 107 175
26 84 108 176
42 67 111 154
41 68 112 153
23 71 127 183
24 72 128 184
7 62 142 167
8 61 141 168
27 77 125 159
28 78 126 160
```

```
19 82 118 163
20 81 117 164
3 73 137 179
4 74 138 180
18 58 122 170
17 57 121 169
33 87 103 173
34 88 104 174
38 66 119 158
37 65 120 157
21 85 129 189
22 86 130 190
46 63 110 147
45 64 109 148
29 89 113 145
30 90 114 146
6 79 106 166
5 80 105 165
43 53 143 185
44 54 144 186
14 59 134 171
13 60 133 172
47 49 135 149
48 50 136 150
43 50 143 154
44 49 144 153
21 83 133 155
22 84 134 156
7 54 139 146
8 53 140 145
27 91 113 183
28 92 114 184
47 67 103 175
48 68 104 176
45 51 129 179
46 52 130 180
3 75 107 159
4 76 108 160
35 59 127 169
36 60 128 170
19 63 106 186
20 64 105 185
23 79 135 167
24 80 136 168
14 94 119 151
13 93 120 152
29 77 111 171
30 78 112 172
34 70 115 163
33 69 116 164
25 89 131 147
26 90 132 148
10 74 122 162
9 73 121 161
39 55 125 177
40 56 126 178
18 71 110 182
17 72 109 181
37 81 141 165
38 82 142 166
15 62 99 190
16 61 100 189
41 65 97 173
42 66 98 174
```

```
31 58 118 150
32 57 117 149
5 95 137 187
6 96 138 188
11 86 123 158
12 85 124 157
1 87 101 191
2 88 102 192
2 95 106 187
1 96 105 188
35 85 107 165
36 86 108 166
6 91 98 151
5 92 97 152
43 65 135 171
44 66 136 172
19 55 127 191
20 56 128 192
3 81 131 189
4 82 132 190
27 59 111 147
28 60 112 148
11 79 121 179
12 80 122 180
15 58 138 163
16 57 137 164
31 87 119 167
32 88 120 168
46 71 103 158
45 72 104 157
29 63 123 173
30 64 124 174
22 67 115 178
21 68 116 177
41 83 99 169
42 84 100 170
26 74 114 154
25 73 113 153
7 77 129 183
8 78 130 184
23 62 134 162
24 61 133 161
33 93 117 181
34 94 118 182
14 51 142 159
13 52 141 160
17 49 125 185
18 50 126 186
10 70 102 175
9 69 101 176
47 89 139 149
48 90 140 150
38 75 110 155
37 76 109 156
39 53 143 145
40 54 144 146
47 58 139 146
48 57 140 145
37 59 117 179
38 60 118 180
43 50 103 150
44 49 104 149
17 87 123 187
18 88 124 188
```

```
7 79 143 163
8 80 144 164
33 83 141 147
34 84 142 148
11 63 99 171
12 64 100 172
31 73 131 155
32 74 132 156
10 90 115 159
9 89 116 160
39 71 119 175
40 72 120 176
23 55 110 190
24 56 109 189
15 75 125 173
16 76 126 174
19 67 130 166
20 68 129 165
35 51 121 185
36 52 122 186
26 66 106 170
25 65 105 169
29 81 135 151
30 82 136 152
14 86 114 167
13 85 113 168
45 69 133 177
46 70 134 178
3 94 111 158
4 93 112 157
1 77 137 161
2 78 138 162
22 54 127 154
21 53 128 153
41 91 101 191
42 92 102 192
27 62 107 182
28 61 108 181
5 95 97 183
6 96 98 184
7 54 126 186
8 53 125 185
11 73 99 189
12 74 100 190
35 67 143 151
36 68 144 152
43 95 97 159
44 96 98 160
27 83 130 166
28 84 129 165
21 93 111 179
22 94 112 180
10 79 107 150
9 80 108 149
41 57 113 181
42 58 114 182
14 71 115 178
13 72 116 177
23 49 139 175
24 50 140 176
39 86 119 154
40 85 120 153
3 77 127 171
4 78 128 172
```

```
18 70 142 163
17 69 141 164
25 87 105 155
26 88 106 156
6 66 122 191
5 65 121 192
29 63 131 169
30 64 132 170
31 59 103 183
32 60 104 184
37 91 135 161
38 92 136 162
34 62 123 147
33 61 124 148
15 75 117 157
16 76 118 158
47 82 102 146
48 81 101 145
1 89 109 173
2 90 110 174
46 51 134 167
45 52 133 168
19 55 137 187
20 56 138 188
6 78 138 151
5 77 137 152
25 51 141 155
26 52 142 156
19 95 103 179
20 96 104 180
47 49 111 187
48 50 112 188
35 82 118 171
36 81 117 172
45 63 131 165
46 64 132 166
31 59 102 154
32 60 101 153
9 65 133 185
10 66 134 186
23 67 130 158
24 68 129 157
1 91 127 167
2 92 128 168
38 71 106 183
37 72 105 184
29 79 123 147
30 80 124 148
22 94 115 162
21 93 116 161
39 57 107 169
40 58 108 170
18 74 143 150
17 73 144 149
15 83 121 173
16 84 122 174
11 55 135 175
12 56 136 176
43 87 113 181
44 88 114 182
14 75 99 178
13 76 100 177
27 69 109 159
28 70 110 160
```

```
34 54 98 191
33 53 97 192
41 61 125 145
42 62 126 146
3 86 119 190
4 85 120 189
7 89 139 163
8 90 140 164
30 90 103 150
29 89 104 149
3 93 107 169
4 94 108 170
47 55 131 163
48 56 132 164
1 63 139 191
2 64 140 192
34 70 123 179
33 69 124 180
15 83 117 189
16 84 118 190
11 54 106 175
12 53 105 176
17 85 137 153
18 86 138 154
19 82 110 167
20 81 109 168
43 79 119 145
44 80 120 146
23 58 135 182
24 57 136 181
31 75 99 173
32 76 100 174
46 67 114 166
45 68 113 165
9 59 121 183
10 60 122 184
26 95 102 162
25 96 101 161
35 73 125 159
36 74 126 160
7 87 127 155
8 88 128 156
39 65 133 187
40 66 134 188
27 51 130 158
28 52 129 157
21 61 111 171
22 62 112 172
6 50 143 178
5 49 144 177
13 77 97 185
14 78 98 186
38 71 142 147
37 72 141 148
41 91 115 151
42 92 116 152
42 55 102 174
41 56 101 173
45 59 121 147
46 60 122 148
7 83 115 191
8 84 116 192
15 91 143 145
16 92 144 146
```

```
22 75 131 178
21 76 132 177
35 69 141 159
36 70 142 160
6 58 127 155
5 57 128 156
37 89 105 161
38 90 106 162
34 62 119 163
33 61 120 164
31 71 97 187
32 72 98 188
10 87 134 167
9 88 133 168
27 51 125 175
28 52 126 176
19 66 118 190
20 65 117 189
11 73 135 153
12 74 136 154
47 54 114 170
48 53 113 169
25 77 111 179
26 78 112 180
39 79 107 151
40 80 108 152
17 85 139 183
18 86 140 184
3 82 110 171
4 81 109 172
13 63 123 165
14 64 124 166
2 95 130 150
1 96 129 149
29 49 137 157
30 50 138 158
23 94 99 182
24 93 100 181
43 67 103 185
44 68 104 186
```