

(51) International Patent Classification:
H03M 13/00 (2006.01)**GEORGI, Marco** [DE/DE]; Caroline-Herschel-Weg 17,
30855 Langenhagen (DE).(21) International Application Number:
PCT/EP2009/063203(74) **Agent: THIES, Stephan**; Deutsche Thomson OHG, Eu-
ropean Patent Operations, Karl-Wiechert-Allee 74, 30625
Hannover (DE).(22) International Filing Date:
9 October 2009 (09.10.2009)(81) **Designated States** (unless otherwise indicated, for every
kind of national protection available): AE, AG, AL, AM,
AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ,
CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO,
DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT,
HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP,
KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD,
ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI,
NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD,
SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT,
TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
08305688.7 16 October 2008 (16.10.2008) EP(71) **Applicant** (for all designated States except US): **THOM-
SON LICENSING** [FR/FR]; 46 Quai Alphonse Le Gal-
lo, F-92100 Boulogne Billancourt (FR).(72) **Inventors; and**(75) **Inventors/Applicants** (for US only): **THEIS, Oliver**
[DE/DE]; Gehaegestr. 20 C, 30655 Hannover (DE).
CHEN, Xiao-Ming [CN/DE]; Vahrenwalderstr. 101,
30165 Hannover (DE). **TIMMERMANN, Friedrich**
[DE/DE]; Moehlenbrinkstr. 6, 30823 Garbsen (DE).(84) **Designated States** (unless otherwise indicated, for every
kind of regional protection available): ARIPO (BW, GH,
GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM,
ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ,
TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE,
ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV,

[Continued on next page]

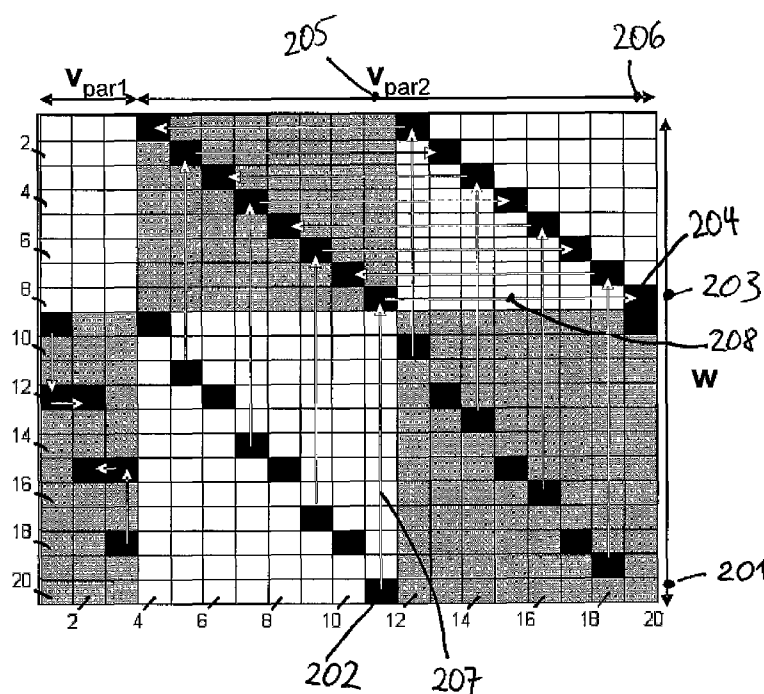
(54) **Title:** METHOD FOR GENERATING A CODE AND METHOD FOR ENCODING

Figure 2

(57) **Abstract:** The invention describes a code
generation method, an encoding method, an
encoding apparatus, and extended versions
thereof. The code generation method compris-
es removing columns and rows from submatri-
ces of a given matrix, and attaching a small
number of additional columns thereto. The en-
coding method derives a codeword v in 3
parts, one of which is the given information
vector u . The remaining 2 parts of the code-
word are derived in a recursive, but paralleliz-
able way. The extended versions use the gen-
erated code in a context of parallel concatena-
ted LDPC coding.



MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, **Published:**

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW,

ML, MR, NE, SN, TD, TG).

— *without international search report and to be republished
upon receipt of that report (Rule 48.2(g))*

Method for generating a code and method for encodingField of the invention

5 The invention relates to the field of error correction codes (ECC) for optical storage systems. It may also be applied to magnetic recording storage devices, Redundant Arrays Of Independent Disks or RAID, and transmission systems.

10 Background of the invention

A class of error correcting codes known as Low Density Parity Check Codes, or LDPC codes, has been adopted for a wide range of applications during past years due to the fact that LDPC Codes show excellent random error correcting behavior when used in
15 conjunction with iterative soft decoding message passing algorithms.

So called quasi-cyclic LDPC codes have proven to be highly suitable for hardware decoder implementation because of their
20 well structured parity check matrix. These codes were originally disclosed 1977 by R.J.G Smith in "Easily Decodable Efficient Selforthogonal block Codes". Smith defined a set of parity check equations on an array of $p \cdot p$ information bits resulting in a code having $k = p^2$ information bits and $(d-1)p$ parity bits and
25 thereby having a codewordlength $k = p^2 + (d-1)p$.

[1] RJG Smith: "Easily decodable efficient self-orthogonal block codes", Electronics Letters, Vol 13, # 7, pp. 173-174; 1977.

[2] L Ping, WK Leung, N Phamdo: "Low density parity check codes with semi-random parity check matrix", Electronics Letters,
30 Vol 35, # 1, pp. 38-39; 1999.

[3] Hongwei Song, BVK Vijaya Kumar: "Low-Density Parity-Check Codes for Partial Response Channels", IEEE Signal Processing Magazine, pp. 56-66; 2004.

Summary of the invention

The invention describes a code generation method, which generates a quasi-cyclic LDPC code that is suitable to be low-complexity encoded without a generator matrix, is column regular, and allows for encoding being parallelized. The invention also describes an encoding method and an encoding apparatus suited to be used with a code as generated by the code generation method. The invention also describes extended versions thereof.

The code generation method according to the invention:

- starts with a Permutation-matrix-based quasi-cyclic LDPC code matrix, which is a block matrix consisting of submatrices that are either identity matrices or circular shift matrices.
- From these submatrices, certain columns and rows are removed.
- A small number of additional columns is attached or inserted to the matrix, e.g. by attaching a so-called z matrix,
- The result is a new parity check matrix H_{mz} , which has certain desirable properties:
 - The matrix continues to be quasi-cyclic, so that it can be used in the known way for low-complexity encoding, a generator matrix is not needed.
 - All column weights are 2, so that performance degradation is avoided.
 - Within each submatrix of the lower half of H , row weights are 0 in regular intervals. This allows for the encoding process to be parallelized, which in turn allows for bigger data throughput.
- The resulting code is also denoted as "zArray code", hereinafter.

More specifically, **a low density parity check LDPC code**

described by its parity check matrix H is generated according to the invention by the following steps:

- A first intermediate matrix H_1 is generated so as to comprise
 5 two rows of square, identically sized, binary sub-matrices, where the first row comprises p identity matrices I of size $p \cdot p$, and the second row comprises p increasing powers S^u of a cyclic shift matrix S of size $p \cdot p$, wherein $u=0, \dots, p-1$. Note that $S^0=I$.
- 10 - From the first intermediate matrix H_1 , a second intermediate matrix H_2 is generated by removing m equidistant columns from each of the sub-matrices of the first intermediate matrix H_1 at column indices $(r+i \cdot (2r+1)+q)$ modulo p , wherein i, m, p, q are integers, $i=0, \dots, m-1$, m, p, q, r are predefined such that
 15 $p=m+2mr$, and column indices within the sub-matrices start with 0.
- From the second intermediate matrix H_2 , a third intermediate matrix H_3 is generated by deleting those matrix rows from the first row of sub-matrices of the second intermediate matrix H_2
 20 which, due to the removing step, contain only zeros, so that the first row of the third intermediate matrix H_3 comprises p small identity matrices I_s of size $(p-m) \cdot (p-m)$.
- From the third intermediate matrix H_3 , the parity check matrix H of the LDPC code is generated by inserting or appending $m-1$
 25 binary column vectors of height $2p-m$ having weight 2, wherein the column vectors have "1" elements in the middle rows of those row ranges of the sub-matrices $[S^0 S^1]$ that have row weight 2.

30

The encoding method according to the invention:

- is based on the known requirement that $v \cdot H^T = 0$, i.e. that the codeword v to be generated, when GF2-multiplied with the parity check matrix H , must result in a zero vector.

- Conceptually, the codeword v is split into 3 parts, one of which is the given information vector u .
- Steps are described, how the remaining 2 parts of the codeword v can be derived in a recursive, but parallelizable way.

More specifically, according to the invention, encoding an information word u into a code word v with a low density parity check LDPC code as generated according to the invention, the code word v consisting of a first parity part v_{par1} , a second parity part v_{par2} , and the information word u , **comprises the steps:**

- a) obtaining a first temporary vector w by GF2-multiplying the information word u with an information matrix H_{inf} ;
 - b) determining the second parity part v_{par2} such that its GF2-product with a second parity matrix H_{par2} equals the first temporary vector w except at the $(p-m + ((i \cdot (2r+1)+q) \bmod p))$ -th vector elements;
 - c) determining a second temporary vector w_2 as the GF2 addition of the GF2-product of the second parity part v_{par2} with the second parity matrix H_{par2} and the first temporary vector w ;
 - d) determining the first parity part v_{par1} such that its GF2-product with a first parity matrix H_{par1} equals the second temporary vector w_2 ;
- wherein the information matrix H_{inf} comprises those $(p-2) \cdot (p-m)$ columns of the parity check matrix H of the LDPC code that are associated with the information word u , the first parity matrix H_{par1} comprises the inserted or appended binary column vectors, and the second parity matrix H_{par2} comprises the remaining columns of the parity check matrix H .

The encoding apparatus according to the invention:

- consists of basic building blocks like permutation, interleaving, matrix multiplication in GF2, mux and demux.
- The calculation of the middle part of the codewords is done in a number of parallel branches of hardware.
- 5 ▪ The number of parallel branches of hardware is choosable by changing some basic code parameters.

Advantages of the invention:

- 10 ▪ By providing column regular codes, the invention avoids performance degrading drawbacks of prior art while maintaining encoding time that grows linearly with codeword length.
- The invention allows parity bit generation to be split into
15 an adjustable number of independent sequential tasks, enabling parallelization of the encoding process.

The general idea of the invention is applicable to transmission or storage systems in contexts where bit error rate requirements
20 are such that random error correction through iterative soft decision decoding must be used.

Extensions:

- 25 ZArray codes, as generated by the code generation method described above, have column weight $j=2$ by definition, allowing efficient encodability. On the other hand, it is known that low column weight LDPC codes are prone to undetected block (codeword) errors and do not show outstanding Bit Error Rate, or
30 BER, performance over binary additive white Gaussian noise, or AWGN, channels.

It may be considered, that BERs of zArray codes are in the range of other Array codes having similar code rate and codeword
35 length. On Partial Response, or PR, channels, especially on

those subjected to Run Length-Limited, or RLL, modulation like storage channels, BER performance of zArray codes can be improved by the use of turbo detection methods using a joint PR-detector and RLL demodulator.

5

It may also be considered, that zArray Codes perform weak on AWGN channels in comparison to other regular high rate codes such as randomly constructed Gallager codes ("Evaluation of Gallager Codes for Short Block Length and High Rate Applications", David J.C. MacKay, Matthew C. Davey, 1999) [4]. MacKay's codes have been chosen for reference since their parity-check matrices are publicly available at "<http://www.inference.phy.cam.ac.uk/mackay/codes/data.html>" which allows easy benchmarking.

15

The performance of zArray codes compared to MacKay's codes mainly has two origins, both due to the way the zArray code's parity check matrix is constructed:

- low column weight 2
- short codeword length

20

Depending on the context, these properties of zArray codes might also be considered as advantages.

In an advantageous **extension**, this invention realizes, that both "shortcomings" of zArray codes mentioned above can be overcome in a way that preserves zArray code advantages, while substantially decreasing BERs.

25

This extension can be seen under 3 different viewpoints of increasing level of abstraction:

30

Viewpoint 1.) The solution to the zArray codes design "shortcomings" in order to decrease BERs is to concatenate two zArray codes in parallel where the second code's information word is interleaved. The interleaver plays the critical role in

35

this scheme since it affects the BER performance significantly. A constraint for the interleaver design will be given.

Viewpoint 2.) Alternatively to interleaving the information word, the columns of the second zArray code parity check matrix corresponding to the information word can be deinterleaved, resulting in a coding scheme of two different parallel "zArray" codes.

10 **Viewpoint 3.)** The two different parity check matrices can be combined into a single parity check matrix resulting in a **new** irregular class of zArray codes, named Parallel Concatenated ZArray, or pczArray, codes in the following. The resulting parity check matrices have column weight 4 on information bits and column weight 2 on parity bits. Advantageously, for a given
15 coderate, pczArray codes have codewordlength substantially larger than zArray codes.

More specifically, according to this invention, an information
20 word u is parallel concatenated LDPC encoded into a codeword vpc by

- applying to the information word u the low density parity check LDPC code as generated by the code generation method described herein, thereby generating first parity bits $vp1$;
- 25 - applying to the information word u an interleaver $P(u)$ and then the low density parity check LDPC code as generated by the code generation method described herein, thereby generating second parity bits $vp2$;
- deriving the codeword vpc by concatenating the information
30 word u , the first parity bits $vp1$, and the second parity bits $vp2$.

Advantages of pczArray codes:

1. Improved BER performance compared to zArray codes
- 35 2. Improved BLER performance compared to zArray codes

3. Reduced probability of undetected block errors compared to zArray codes

4. Efficient zArray code's encoder and decoder can be used for pczArray codes

5

Drawings

10 Exemplary embodiments of the invention are illustrated in the drawings and are explained in more detail in the following description.

In the figures:

15 **Figure 1** shows a Parity Check Matrix $H_{m \times n}$ of a quasi-cyclic LDPC Code with $m=3$ and $r=1$ $p-m=6$. Ones are represented by black squares. White and grey squares represent Zeros.

Figure 2 shows a Parity Check Matrix $[H_{\text{par1}} \ H_{\text{par2}}]$ of a quasi-cyclic LDPC Code with $m=4, r=1$. Ones are represented by black squares. White and grey squares represent Zeros.

20 **Figure 3** shows a block diagram for a $2m$ -parallel implementation of the v_{par2} encoder and for a sequential implementation of the v_{par1} encoder.

Figure 4 shows a Parallel Concatenated zArray LDPC encoder with interleaver and identical encoders.

Figure 5 shows an Informationword block interleaver with depth $p-2-s$.

Figure 6 shows an Informationword block interleaver with depth $p-m$.

30 **Figure 7** shows a Parallel Concatenated zArray LDPC decoder with interleaver and identical decoders.

Figure 8 shows a Parallel Concatenated zArray LDPC encoder with two different encoders.

Figure 9 shows a zArray2 LDPC parity check matrix with parameter $m=3, r=1$ and $s=2$, which substitutes information word

35

block interleaving with depth $p-2-s$.

Figure 10 shows a zArray2 LDPC parity check matrix H_{pc} with parameter $m=3$, $r=1$ and $s=2$, which substitutes information word block interleaving with depth $p-m$.

5 **Figure 11** shows a Parallel Concatenated zArray LDPC decoder with two different decoders.

Figure 12 shows a pczArray LDPC parity check matrix H_{pc} with parameter $m=3$, $r=1$ and $s=2$, which substitutes information word block interleaving with depth $p-m$.

10

Detailed description

For a given minimum distance d and maximum column weight $j=d-1$, Smith's code construction results in a parity check matrix H_{smith} consisting of $p \cdot p$ identity matrices and shift matrices S obtained from the identity matrix through cyclically left-shifting or right-shifting by one position, where p is usually selected to be a prime number.

15

$$S = \begin{bmatrix} & & & 1 \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}_{p \times p} \quad \text{or} \quad S = \begin{bmatrix} & 1 & & \\ & & 1 & \\ & & & 1 \\ 1 & & & \end{bmatrix}_{p \times p}$$

20

Smith did not employ a parity check matrix notation. Instead he gave algebraic equations to define the parity checks, which translate into the use of right-shifts S in the parity check matrix notation. His construction method can be written as

25

$$H_{smith} = \begin{bmatrix} \mathbf{I} & & & \mathbf{I} & \mathbf{I} & \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} \\ & \mathbf{I} & & \mathbf{I} & S & S^2 & S^3 & \dots & S^{p-1} \\ & & \mathbf{I} & \mathbf{I} & S^2 & S^4 & S^6 & \dots & S^{2(p-1)} \\ & & & \ddots & \vdots & \vdots & \vdots & \dots & \vdots \\ & & & \mathbf{I} & \mathbf{I} & S^{j-1} & S^{2(j-1)} & S^{3(j-1)} & \dots & S^{(j-1)(p-1)} \end{bmatrix}_{jp \times p^2 + jp}.$$

Hsmith defines a quasi-cyclic LDPC code from which the identity matrix is usually removed. The parity check matrix H is then given as a $j \times p$ block matrix of $p \times p$ circulant submatrices.

5

$$\mathbf{H} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} \\ \mathbf{I} & S & S^2 & S^3 & \dots & S^{(p-1)} \\ \mathbf{I} & S^2 & S^4 & S^6 & \dots & S^{2(p-1)} \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ \mathbf{I} & S^{(j-1)} & S^{2(j-1)} & S^{3(j-1)} & \dots & S^{(j-1)(p-1)} \end{bmatrix}_{jp \times p^2}$$

H specifies a regular (n, k) block code with codeword length n given by $n=p^2$, information word length $k=n-\text{rank}(H)$ and $n-k$ parity bits corresponding to $n-k$ parity check equations. The code rate is defined to be the ratio of informationwordlength to codewordlength $R=k/n$.

Since the number of ones within H is small compared to the number of Zeros, the matrix is said to have low density. Each codeword v must satisfy all $n-k$ parity checks. This is equivalent to requiring that the vector matrix multiplication of v and H within the Galois field GF2 must yield an all zero vector:

$$\mathbf{vH}^T = \mathbf{0}.$$

Encoding usually requires the existence of a generator matrix G which can be derived from H through Gaussian elimination. GF2-multiplication of the generator matrix with the informationword u requires $O(n^2)$ computations and results in the codeword $v=uG$.

A straightforward implementation of this encoding is of complexity $O(n^2)$. This complexity might be prohibitive for large codewordlength, so that a modification of the above code has been disclosed in **US 2003/0037298**. There it is shown how a generator matrix $\mathbf{H}_U^S$ is derived from $\mathbf{H}$ through shifting and

30

triangularization.

$$\mathbf{H}_U^S = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & S & \dots & S^{(j-2)} & S^{(j-2)} & & S^{(p-2)} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \dots & S^{2(j-3)} & S^{2(j-2)} & & S^{2(p-3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & S^{(j-1)} & \dots & S^{(j-1)(p-1)} \end{bmatrix}$$

5 This form of the parity check matrix enables linear time encoding. Another solution to the encoding problem can be the use of a "zigzag" matrix $\mathbf{H}_{zz}$ instead of triangularization. This method is generic and can be applied to virtually any existing LDPC code $\mathbf{H}_{any}$ to simplify encoding by defining $\mathbf{H}_{PLP} = [\mathbf{H}_{zz} \mathbf{H}_{any}]$
 10 where $\mathbf{H}_{zz}$ is defined as

$$\mathbf{H}_{zz} = \begin{bmatrix} 1 \downarrow \\ 1 \rightarrow 1 \downarrow \\ 1 \rightarrow 1 \downarrow \\ \vdots \rightarrow \vdots \downarrow \\ 1 \rightarrow 1 \end{bmatrix}$$

15 The parity check equation $\mathbf{v}\mathbf{H}^T = \mathbf{0}$ can then be solved iteratively in the way indicated by arrows to generate the parity bit vector $\mathbf{v}_p$. Systematic codewords result from concatenation of parity and information bits.

20
$$\mathbf{v} = [\mathbf{v}_p \mathbf{u}]$$

This method has been disclosed by Ping, Leung and Phamdo in "Low density parity check codes with semirandom parity check matrix".

25 Codes generated using a Zigzag matrix can be seen to have the following Disadvantages:

1. Parity bits have to be generated sequentially. This avoids

parallelization of the encoding algorithm.

2. The code is not column regular any more (the number of ones per column varies) and there is at least one column having weight 1. This usually degrades soft decision decoding performance.

Codes generated according to **US 2003/0037298** can be seen to have the following drawback:

The code is not column regular any more. p columns have row weight one. This usually degrades soft decision decoding performance.

This invention overcomes the disadvantages of quasi-cyclic LDPC code encoding as stated above, while maintaining linear encoding time. For this aim, a new class of LDPC codes is disclosed that allows parity bit generation to be split into an adjustable number of independent sequential tasks enabling parallelization of the encoding process; and is column regular. Additionally, same as the two methods mentioned above, the new class of LDPC codes according to this invention needs no computation of a generator matrix for encoding.

The invention presents a new class of LDPC codes represented by the parity-check matrix H_{mz} . It can be deduced from H in a way that simplifies encoding (linear time & parallelization) while maintaining the structure of the original parity-check matrix H to keep decoder design simple and efficient especially when message passing algorithms are applied.

The method of this invention is restricted to permutation matrix based quasi-cyclic LDPC codes of type $j=2$.

Code Construction

This part describes step by step how a new class of LDPC codes is constructed from the known class of quasi-cyclic LDPC codes as shown above having parameter $j=2$, so that the parity-check matrix is of the form

$$\mathbf{H} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \\ \mathbf{I} & S & S^2 & S^3 & \cdots & S^{p-1} \end{bmatrix}_{2p \times p^2}.$$

Step 1: Define $p = m + 2mr$ with $m, r \geq 1$. This will allow removal of m equidistant rows in the next step.

Step 2: Remove m equidistant columns from each submatrix $\mathbf{I}$ and S^u at column indices (starting at 1) $x = r + i(2r+1) + 1$ with $i = 0, \dots, m-1$. Cyclic shifts of x might also be used for column removal. For the identity matrix $\mathbf{I}$ this results in

$$\mathbf{I}' = \begin{bmatrix} \ddots & & & & \\ & 1 & & & \\ \cdots & 0 & 0 & \cdots & \\ & & 1 & & \\ & & & \ddots & \end{bmatrix}_{p \times (p-1)} \quad \text{for } m=1, r \geq 1$$

$$\text{or } \mathbf{I}' = \begin{bmatrix} \ddots & & & & & & \\ & 1 & & & & & \\ \cdots & 0 & 0 & \cdots & & & \\ & & 1 & & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & & 1 & \\ & & & & \cdots & 0 & 0 & \cdots \\ & & & & & & 1 & \\ & & & & & & & \ddots & \end{bmatrix}_{p \times (p-m)} \quad \text{for } m > 1.$$

For the upper half of the parity check matrix m , zero rows can be removed since they do not define a parity check any more. A second identity matrix

$$\mathbf{I}'' = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}_{(p-m) \times (p-m)} \quad \text{is therefore defined.}$$

Removal of m equidistant columns from the shift matrices S and its permutations S^u defines a new matrix

$$S' = \begin{bmatrix} & 1 \\ 1 & \\ 0 & 0 \end{bmatrix}_{p \times (p-1)} \quad \text{for } m=r=1$$

$$\text{or } S' = \begin{bmatrix} & & 1 \\ 1 & & \\ & 1 & \\ & 0 & \\ & 1 & \end{bmatrix}_{p \times (p-1)} \quad \text{for } m=1 \text{ an } r=2$$

$$\text{or } S' = \begin{bmatrix} & & & 1 \\ 1 & & & \\ 0 & 0 & & \\ & 1 & & \\ & & 1 & \\ & & 0 & \end{bmatrix}_{p \times (p-2)} \quad \text{for } m=2 \text{ an } r=1$$

15

$$\text{or } S' = \begin{bmatrix} & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ \dots & 0 & 0 & \dots & & & & & & & \\ & & 1 & & & & & & & & \\ & & & \ddots & & & & & & & \\ & & & & \ddots & & & & & & \\ & & & & & \ddots & & & & & \\ & & & & & & 1 & & & & \\ & & & & & \dots & 0 & 0 & \dots & & \\ & & & & & & & 1 & & & \\ & & & & & & & & \ddots & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & px(p-m) \end{bmatrix}$$

After removing a total of pm columns and m rows from $\mathbf{H}$, the
5 new parity check matrix reads

$$\mathbf{H}_m = \begin{bmatrix} \mathbf{I}'' & \mathbf{I}'' & \mathbf{I}'' & \mathbf{I}'' & \dots & \mathbf{I}'' \\ \mathbf{I}' & S' & S^{2'} & S^{3'} & \dots & S^{p-1'} \end{bmatrix}_{(2p-m) \times (p-m)p}.$$

Step 3: Expand $\mathbf{H}_m$ by $z=m-1$ columns having weight 2. This yields
10 the parity check matrix $\mathbf{H}_{mz}$:

$$\mathbf{H}_{mz} = \begin{bmatrix} 0 & \mathbf{I}'' & \mathbf{I}'' & \mathbf{I}'' & \mathbf{I}'' & \dots & \mathbf{I}'' \\ \mathbf{Z} & \mathbf{I}' & S' & S^{2'} & S^{3'} & \dots & S^{p-1'} \end{bmatrix}_{(2p-m) \times ((p-m)p+z)}.$$

15 Within $\mathbf{Z}$, ones are set in any combination of two rows having
 neighboring indices $y_z(i)=(2r+1)i$ with $i=0,\dots,m-1$, i.e.
 $y_z(0)\&y_z(1)$, $y_z(1)\&y_z(2)$, ..., $y_z(m-2)\&y_z(m-1)$. If cyclic shifts
 of $\mathbf{x}$ have been used for column removal in step 2, row indices
 y_z will change accordingly.

20

This defines a new LDPC code with codewordlength $n=(p-m)p+z$ and informationwordlength $k=(p-2)(p-m)$. The code is free of 4-

cycles and has only few 6-cycles.

See **Figure 1** for an example of $\mathbf{H}_{mz}$. Figure 1 shows a Parity Check Matrix $\mathbf{H}_{mz}$ of a quasi-cyclic LDPC Code with $m=3$, $r=1$, and $p-m=6$. Ones are represented by black squares. White and grey squares represent Zeros. The first and upper row of submatrices of $\mathbf{H}_{mz}$ contains 9 identical identity matrices 101 ... 109 each of dimension 6×6 . The second and lower row of submatrices of $\mathbf{H}_{mz}$ contains the Z matrix 110, of dimension 2×9 ; and 9 column-removed circular shift matrices 111 ... 119 of dimension 6×9 each. All column weights of $\mathbf{H}_{mz}$ are 2, the row weights in the upper 6 rows are constantly equal to 9, and row weights in the lower 9 rows are 6, 7, or 8.

Encoding

Given $\mathbf{H}_{mz}$, the parity check equation can be expressed as $\mathbf{v}\mathbf{H}_{mz}^T = \mathbf{0}$.

If systematic encoding of the codeword $\mathbf{v} = [\mathbf{v}_{par1} \mathbf{v}_{par2} \mathbf{u}]$ is assumed, $\mathbf{H}_{mz}$ splits into two parity parts and one information bit part.

$$\mathbf{H}_{mz} = \begin{bmatrix} \mathbf{H}_z^{par1} & \mathbf{H}_{2(p-m)}^{par2} & \mathbf{H}_{(p-2)(p-m)}^{inf} \end{bmatrix}$$

In the example Parity Check Matrix $\mathbf{H}_{mz}$ of Figure 1, the leftmost $z=2$ columns constitute the matrix part $\mathbf{H}^{par1}$. Contiguous to it, the next $2(p-m)=12$ columns, with other words the column range from and including column 3 to and including column 14, constitute the matrix part $\mathbf{H}^{par2}$. Contiguous to it, the next $(p-2)(p-m)=42$ columns, with other words the column range from and including column 15 to and including column 56, constitute the matrix part $\mathbf{H}^{inf}$.

Encoding Step 1: Multiply (in GF2) $\mathbf{u}$ and $\mathbf{H}_{inf}$ to obtain a

temporary vector $\mathbf{w} = \mathbf{u}\mathbf{H}_{\text{inf}}^T$.

Encoding Step 2: Find $\mathbf{v}_{\text{par2}}$, also denoted as $\mathbf{v}_{\text{par2}}$, recursively by solving

$$\mathbf{v}_{\text{par2}}\mathbf{H}_{\text{par2}}^T = \mathbf{w} \quad (1)$$

in $2m$ independent parallel tasks. Each row of $\mathbf{H}_{\text{par2}}$ having weight 1 serves as a starting point for one of the different recursions, resulting in $2r$ parity bits each. Alternatively m parallel tasks might be applied each resulting in $4r$ parity bits. Notice that rows having index $y_{mz}(i) = y_z(i) + (p-m) + 1$ are not taken into account within the recursions. Equation (1) will therefore likely be violated by these rows since

$$\text{rank}(\mathbf{H}_{\text{par2}}) = 4r - m.$$

Encoding Step 3: Compute a vector $\mathbf{w}' = \mathbf{v}_{\text{par2}}\mathbf{H}_{\text{par2}}^T \oplus \mathbf{w}$. Actually, only $\mathbf{w}'(y_{mz})$ is needed for the next step.

Encoding Step 4: Find $\mathbf{v}_{\text{par1}}$ recursively by solving $\mathbf{v}_{\text{par1}}\mathbf{H}_{\text{par1}}^T = \mathbf{w}'$. This can be done within 2 independent recursive tasks since $\text{rank}(\mathbf{H}_{\text{par1}}) = z$. One task starts with $\mathbf{v}_{\text{par1}}(1) = \mathbf{w}'(y_{mz}(0))$, and the other task starts with $\mathbf{v}_{\text{par1}}(m-1) = \mathbf{w}'(y_{mz}(m-1))$.

Encoding Step 5: Assemble $\mathbf{v} = [\mathbf{v}_{\text{par1}} \mathbf{v}_{\text{par2}} \mathbf{u}]$

Figure 2 gives an example of how the encoding process is split up into parallel tasks to compute $\mathbf{v}_{\text{par2}}$ first and afterwards $\mathbf{v}_{\text{par1}}$. Figure 2 shows a Parity Check Matrix part $[\mathbf{H}_{\text{par1}} \mathbf{H}_{\text{par2}}]$ of a quasi-cyclic LDPC Code with $m=4$, $r=1$. Ones are represented by black squares. White and grey squares represent Zeros. The Figure illustrates a situation where encoding step 1 has already been performed, so that the temporary vector $\mathbf{w}$ is known. The Figure can be used to illustrate that

- the GF2 scalar product of the second parity part vector

vpar2 and the row 1 vector of $\mathbf{H}^{\text{par2}}$ must be equal to the first element of the temporary vector w,

- the GF2 scalar product of the second parity part vector vpar2 and the row 2 vector of $\mathbf{H}^{\text{par2}}$ must be equal to the

5 second element of the temporary vector w,
and so on until

- the GF2 scalar product of the second parity part vector vpar2 and the row 20 vector of $\mathbf{H}^{\text{par2}}$ must be equal to the twentieth element of the temporary vector w.

10

Among the rows of $\mathbf{H}^{\text{par2}}$, rows 10, 11, 13, 14, 16, 17, 19, and 20 have row weight 1, so that the "1" elements in these rows constitute the starting points for the 2m=8 recursions. The recursion starting in line 20 comprises the following steps:

- 15 • From the 20-th element 201 of the temporary vector w, the 8-th element 205 of the second parity part vector vpar2 can be determined. This is based on the fact that the 8-th element 202 of the row 20 vector of $\mathbf{H}^{\text{par2}}$ is set, as illustrated by the black square in column 11 of the overall
- 20 matrix.
- The knowledge about the 8-th element 205 of the second parity part vector vpar2 is then exploited in the 8-th row of $\mathbf{H}^{\text{par2}}$ which is the second row that has a set element in column 11. This transfer is symbolized by the arrow 207.
- 25 • In the 8-th row, given the 8-th element of the temporary vector w, and given the 8-th element of the second parity part vector vpar2, the 16-th element 206 of the second parity part vector vpar2 can be determined. This is based on the fact that the 16-th element 204 of the row 8 vector
- 30 of $\mathbf{H}^{\text{par2}}$ is also set. This transfer is symbolized by the arrow 208.

Encoding apparatus

Figure 3 shows a block diagram for a $2m$ -parallel implementation of the vpar2 encoder and for a sequential implementation of the vpar1 encoder.

5

A hardware realization for the algorithm described above can be directly deduced from the block diagram shown in Figure 3. The vector $\mathbf{w}$ is generated through the block Hinf 301, which performs matrix vector multiplication with $\mathbf{u}$ in GF2. After interleaving $\mathbf{w}$ through block Pw 302 also denoted as Π_w , a demultiplexer 303 feeds the $2m$ (or m) XORs 304, 305. Again, vpar2 results from interleaving in blocks Π_1 306 through Π_{2m} 307, and subsequent concatenation by the multiplexer 308. vpar1(i) is then recursively computed by selecting, in a selector SEL_y 309 the element $w(y_{mz})$ of the vector w , and by XORing, in XOR 310, the element $\mathbf{w}(y_{mz})$ with vpar2($2ri+1$) which is the element of vpar2 at the bit position selected by selector SEL₁ 311, then XORing the result, in XOR 312, with vpar2(mod($2ri-1, p-m$)+($p-m$)+1) which is the element of vpar2 at the bit position selected by selector SEL₂ 313, and finally by XORing the result with vpar1($i-1$). Finally $\mathbf{v}$ is found through concatenation, in MUX 314, of [vpar2 u] and vpar1.

25

For the case of $m=1$ ($\mathbf{H}_m = \mathbf{H}_{mz}$), there exists a very similar class of array codes invented by Mario Blaum (IBM) and published in "A CODING TECHNIQUE FOR RECOVERY AGAINST DOUBLE DISK FAILURE IN DISK ARRAYS" in 1992. These codes can be derived from the code according to this invention by removing column $x=p$ instead of removing column $x=r+1$ from any submatrix and by setting p to a prime number.

30

Furthermore, row $2p-1$ has to be removed from $\mathbf{H}_m$ to obtain the

parity check matrix

$$\mathbf{H}_{\text{array}} = \begin{bmatrix} \mathbf{I}^* & \mathbf{I}^* & \mathbf{I}^* & \mathbf{I}^* & \cdots & \mathbf{I}^* \\ \mathbf{I}^* & \mathbf{S}^* & \mathbf{S}^{*2} & \mathbf{S}^{*3} & \cdots & \mathbf{S}^{*p-1} \end{bmatrix}_{2(p-1) \times (p-1)p}$$

5 with

$$\mathbf{I}^* = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}_{(p-1) \times (p-1)} \quad \text{and} \quad \mathbf{S}^* = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix}_{(p-1) \times (p-1)}.$$

Note that $\mathbf{S}^{*x}$ results from $\mathbf{S}^x$ by deletion of column p and row p .

10

Both codes then result in the same codeword $\mathbf{v}$ for a given information vector $\mathbf{u}$ and can therefore not be distinguished.

15

Blaum's array codes were originally designed for double symbol erasure or single symbol error correction of blocks having length $p-1$. But they can also be considered as LDPC codes to which message passing algorithms can be applied for random error correction. This was discovered by John Fan in "ARRAY CODES AS LOW-DENSITY PARITY-CHECK CODES".

20

Note that Blaum's array codes can not take advantage of 2-way parallel encoding due to the missing last parity row.

25

Permutation-matrix-based quasi-cyclic LDPC codes can be derived from a class of codes introduced in a 1977 paper by RJG Smith [1] through shortening of the parity bits and sacrificing parts of the original information for redundancy instead, see [3].

Because of their regularly structured parity check matrix, they are very efficiently decodable.

Ping, Leung and Phamdo, in their 1999 paper [2], have described a way to modify any given parity check matrix into a form that allows to use it in a low-complexity, iterative way yielding the parity bits to a given information word u . This enables for efficient encoding and circumvents the necessity to calculate a generator matrix.

The method of [2] can be seen to have the following drawbacks:

- Parity bits have to be generated sequentially, which disallows parallelization.
- The resulting code is not column regular, i.e. the column weight of the parity check matrix varies. At least one column has weight 1, i.e. one information digit is contained in only one of the parity equations. All this typically degrades soft-decision decoding performance.

US 2003/0037298 has modified Permutation-matrix-based quasi-cyclic LDPC codes in such a way that they are more easily encodable in addition. This method, too, may be seen to have the drawback that the proposed code is not column regular. A number "p" of columns have weight 1, which typically degrades soft-decision decoding performance.

The invention builds on permutation-matrix-based quasi-cyclic LDPC codes having column weight 2.

In the following, it will be described how the zArray codes can be extended by using them in a parallel concatenated coding scheme. This extension can be seen under 3 different viewpoints of increasing level of abstraction:

Viewpoint 1.) The solution to the zArray code's design

"shortcomings" in order to decrease BERs is to concatenate two zArray codes in parallel where the second code's information word is interleaved in a way defined by the invention. The interleaver P also denoted as Π plays a critical role in this scheme since it affects the BER performance significantly.

Figure 4 shows a Parallel Concatenated zArray LDPC encoder with interleaver 401 and identical encoders 402, 403. The codeword $\mathbf{v}_{pc} = [\mathbf{v}_{p2} \mathbf{v}_{p1} \mathbf{u}]$ 407 of a pczArray code is derived from the information word $\mathbf{u}$ 404 and its concatenation with parity bits $\mathbf{v}_{p1}$ 405 from a first zArray encoder 402 working on $\mathbf{u}$ 404 and with parity bits $\mathbf{v}_{p2}$ 406 from a second identical zArray encoder 403 working on $\Pi(\mathbf{u})$.

With the zArray code variable $p = m + 2mr$ depending on the two code parameters m and r as defined above, the number of symbols per information word $\mathbf{u}$ results as $p-2$, where the symbol length is given by $p-m$.

In order to obtain low block error rates or BLERs and low bit error rates or BERs, it is important that the number of symbols within $\mathbf{u}$ and the symbol length must be relatively prime to each other, i.e. their greatest common divisor (gcd) must equal 1.

$$\gcd(p-2, p-m) \equiv 1 \quad (\text{I})$$

It is often necessary to adjust code rate and block length to application needs by information word shortening, i.e. replacing a certain amount of information bits of $\mathbf{u}$ by zeros which will not be transmitted. Without loss of generality it will be assumed hereinafter, that the number of zeros is a multiple of the symbol length $p-m$ and that zeros will be inserted at the end of $\mathbf{u}$. For this purpose, a third zArray code parameter s is introduced, denoting the number of zero filled symbols. The

information wordlength k then results as $k=(p-2-s)(p-m)$. With $s>0$, the above condition (I) then becomes:

$$\gcd(p-2-s, p-m) \equiv 1 \quad (\text{II})$$

5 It is the intention not to define the interleaver Π more specifically for keeping as much design freedom as possible. As one example, a block interleaving scheme having depth $p-2-s$, as shown in **Figure 5**, where bits of $\mathbf{u}$ are written row-wise top down into a mother matrix of dimension $[p-2-s, p-m]$ and are read out
10 column-wise from left to right can be used. Figure 5 shows an Informationword block interleaver with depth $p-2-s$.

Another example is an interleaver having depth $p-m$, as shown in **Figure 6**. Figure 6 shows an Informationword block interleaver
15 with depth $p-m$.

Decoding can be done in a turbo decoding scheme, where extrinsic information from one decoder serves as apriori knowledge for the other decoder within several iterations, as shown in **Figure 7**.
20 Figure 7 shows a Parallel Concatenated zArray LDPC decoder with interleaver and identical decoders.

The idea of parallel concatenated codes and turbo decoding was successfully applied to convolutional codes by C. Berrou,
25 A.Glavieux and Punya Thitimayshima who disclosed their landmark paper "Near Shannon limit error-correction coding and decoding: Turbo-codes" in 1993. The corresponding patent **US 5,446,747** discloses "systematic convolutional coding" as the constituent coders, although it is stated that other codes might be used as
30 well.

Viewpoint 2.) Instead of interleaving the information word with an interleaver P or Π , the columns of the second zArray code
35 parity check matrix corresponding to the information word can be

deinterleaved with the inverse interleaver P^{-1} or Π^{-1} , resulting in a coding scheme as shown in **Figure 8**, comprising a conventional zArray encoder zArray1 and a second encoder zArray2. Figure 8 shows a Parallel concatenated zArray LDPC encoder with two different encoders.

Notice that parity check matrix column deinterleaving for the second code basically destroys the quasicyclic block structure of the zArray code. Therefore zArray2 is not a zArray code in the original sense anymore!

An example for a parity check matrix of zArray2 substituting an information word block interleaver Π having depth $p-2-s$ (as given by **Figure 5**) is given in **Figure 9**.

Another example for a parity check matrix of zArray2 substituting a information word block interleaver Π having depth $p-m$ (as given by **Figure 6**) is given in **Figure 10**.

Since parity check bit order has not been changed, the encoding schedule as described above basically stays the same, despite the fact that the non quasi-cyclic code structure makes computation of the temporary vector **w** more difficult.

The scheme of two different parallel concatenated LDPC codes is already known as PCGC (parallel concatenated Gallager Codes) and has been published by H. Behairy and S.-C. Chang in November of 2000 in "Parallel concatenated Gallager Codes". They confirmed and complemented their results in 2002 in "Analysis and design of parallel concatenated Gallager codes". Behairy and Chang propose to use LDPC codes with different mean column weight (MCW).

In contrast, in this invention the second zArray code results from the first zArray code through column (de)interleaving of

information word columns, so that both codes have regular column weight 2.

Behairy/Chang also propose to use a LDPC decoder for each code in a turbo decoding way (as it is shown in **Figure 11** for pczArray codes), where extrinsic information from one decoder serves as apriori knowledge for the other. This decoding scheme has the advantage that no interleaver delay occurs during decoding in contrast to the one given in **Figure 7**. Nevertheless high throughput decoding might be difficult due to the non quasi-cyclic code structure of zArray2.

Figure 9 shows a zArray2 LDPC parity check matrix with parameter $m=3$, $r=1$ and $s=2$, substituting information word block interleaving with depth $p-2-s$. **Figure 10** shows a zArray2 LDPC parity check matrix H_{pc} with parameter $m=3$, $r=1$ and $s=2$, substituting information word block interleaving with depth $p-m$. **Figure 11** shows a Parallel concatenated zArray LDPC decoder with two different decoders.

Viewpoint 3.) The two different parity check matrices of zArray1 and zArray2 can be combined into a single parity check matrix H_{pc} , also denoted as H_{pc} , resulting in a **new** irregular class of zArray codes, named Parallel Concatenated ZArray codes or pczArray codes in the following. These codes have column weight 4 on information bits and column weight 2 on parity bits. For a given coderate, pczArray codes have codewordlength substantially larger than zArray codes. **Figure 12** shows the pczArray code parity check matrix corresponding to **Figure 10**.

Since LDPC codes are defined by their parity check matrix alone, the method of parallel code concatenation is hidden in the approach according to the viewpoint 3. The invention can therefore also be seen as a method for LDPC code construction especially for extending the zArray codes construction design space.

Figure 12 shows a pczArray LDPC parity check matrix H_{pc} with parameter $m=3$, $r=1$ and $s=2$, substituting information word block interleaving with depth $p-m$.

5

Encoding can be done by an implementation given by **Figure 4** or **Figure 8**.

10 Instead of performing turbo decoding on $zArray1$ and $zArray2$, a single pczArray LDPC decoder is used where for example a message passing schedule can be carried out over H_{pc} .

15 With other words, the invention describes a code generation method, an encoding method, an encoding apparatus, and extended versions thereof. The code generation method comprises removing columns and rows from submatrices of a given matrix, and attaching a small number of additional columns thereto. The encoding method derives a codeword v in 3 parts, one of which is
20 the given information vector u . The remaining 2 parts of the codeword are derived in a recursive, but parallelizable way. The extended versions use the generated code in a context of parallel concatenated LDPC coding.

Claims

1. A **method for generating a low density parity check LDPC code** that is suitable to be low-complexity encoded without a generator matrix, is column regular, and allows for encoding being parallelized, the code described by its parity check matrix (H), the method comprising the steps of:

- generating a first intermediate matrix (H1) comprising two rows of square, identically sized, binary sub-matrices, where the first row comprises p identity matrices (I) of size $p \cdot p$, and the second row comprises p increasing powers (S^u) of a cyclic shift matrix (S) of size $p \cdot p$, wherein $u=0, \dots, p-1$;
 - generating from the first intermediate matrix (H1) a second intermediate matrix (H2) by removing m equidistant columns from each of the sub-matrices of the first intermediate matrix (H1) at column indices $(r+i \cdot (2r+1)+q)$ modulo p , wherein i, m, p, q are integers, $i=0, \dots, m-1$, m, p, q, r are predefined such that $p=m+2mr$, and column indices within the sub-matrices start with 0;
 - generating from the second intermediate matrix (H2) a third intermediate matrix (H3) by deleting those matrix rows from the first row of sub-matrices of the second intermediate matrix (H2) which, due to the removing step, contain only zeros, so that the first row of the third intermediate matrix (H3) comprises p small identity matrices (Is) of size $(p-m) \cdot (p-m)$;
 - generating from the third intermediate matrix (H3) the parity check matrix (H) of the LDPC code by inserting or appending $m-1$ binary column vectors of height $2p-m$ having weight 2, wherein the column vectors have "1" elements in the middle rows of the row ranges of the sub-matrices $[S^0 S^1]$ that have row weight 2.
2. A method according to claim 1, wherein in an i -th one of the column vectors elements are set to "1" at row

$p-m + ((i \cdot (2r+1) + q) \bmod p)$ and at row
 $p-m + (((i+1) \cdot (2r+1) + q) \bmod p)$, wherein $i=0, \dots, m-2$ and row
 indices start with 0, and wherein the elements are set to "0"
 at all other rows.

5

3. A **method for encoding** an information word (u) into a code
 word (v) with a low density parity check LDPC code as
 generated by the method of claim 1, wherein the code word (v)
 consists of a first parity part (vpar1), a second parity part
 10 (vpar2), and the information word (u), the method **comprising**
the steps:

a) obtaining a first temporary vector (w) by GF2-multiplying
 the information word (u) with an information matrix (Hinf);

b) determining the second parity part (vpar2) such that its
 15 GF2-product with a second parity matrix (Hpar2) equals the
 first temporary vector (w) except at the
 $(p-m + ((i \cdot (2r+1) + q) \bmod p))$ -th vector elements;

c) determining a second temporary vector (w2) as the GF2
 addition of the GF2-product of the second parity part (vpar2)
 20 with the second parity matrix (Hpar2) and the first temporary
 vector (w);

d) determining the first parity part (vpar1) such that its
 GF2-product with a first parity matrix (Hpar1) equals the
 second temporary vector (w2);

25 wherein the information matrix (Hinf) comprises those
 $(p-2) \cdot (p-m)$ columns of the parity check matrix (H) of the LDPC
 code that are associated with the information word (u), the
 first parity matrix (Hpar1) comprises the inserted or appended
 binary column vectors, and the second parity matrix (Hpar2)
 30 comprises the remaining columns of the parity check matrix
 (H).

4. A method according to claim 3, wherein the determining of
 method step b) is performed in $2 \cdot m$ sequences of calculation
 35 starting at those $2 \cdot m$ rows of the second parity matrix (Hpar2)

that have row weight 1, with each sequence of calculation producing $2 \cdot r$ parity bits.

5. A method according to claim 3, wherein the determining of method step b) is performed in m sequences of calculation starting at m of the $2 \cdot m$ rows of the second parity matrix (H_{par2}) that have row weight 1, with each sequence of calculation producing $4 \cdot r$ parity bits.

10 6. A method according to claim 4 or 5, wherein the sequences of calculation are performed in parallel.

7. A method according to claim 3, wherein the determining of method step d) is performed in 2 sequences of calculation starting at those 2 rows of the first parity matrix (H_{par1}) that have row weight 1, with each sequence of calculation producing $m-1$ parity bits.

8. **Method for parallel concatenated LDPC encoding** an information word (u) into a codeword (vpc), the method comprising the steps of:

- applying to the information word (u) the low density parity check LDPC code as generated by the method of claim 1, thereby generating first parity bits ($vp1$);
- 25 - applying to the information word (u) an interleaver ($P(u)$) and then the low density parity check LDPC code as generated by the method of claim 1, thereby generating second parity bits ($vp2$);
- deriving the codeword (vpc) by concatenating the informationword (u), the first parity bits ($vp1$), and the second parity bits ($vp2$).

9. Method for parallel concatenated LDPC encoding an information word (u) into a codeword (vpc), the method comprising the steps of:

- applying to the information word (u) the low density parity check LDPC code as generated by the method of claim 1, thereby generating first parity bits (vp1);
- applying to the information word (u) a low density parity
5 check LDPC code that is equivalent to the code as generated by the method of claim 1 after interleaving the columns of the code's parity check matrix, thereby generating second parity bits (vp2);
- deriving the codeword (vpc) by concatenating the
10 informationword (u), the first parity bits (vp1), and the second parity bits (vp2).

10. Method for LDPC encoding information words (u),
characterized in that the parity check matrix (H) of the LDPC
15 code is a parallel concatenated parity check matrix (Hpc)
comprising a parity check matrix of the LDPC code as generated
by the method of claim 1, juxtaposed to a column interleaved
parity check matrix of the LDPC code as generated by the
method of claim 1.

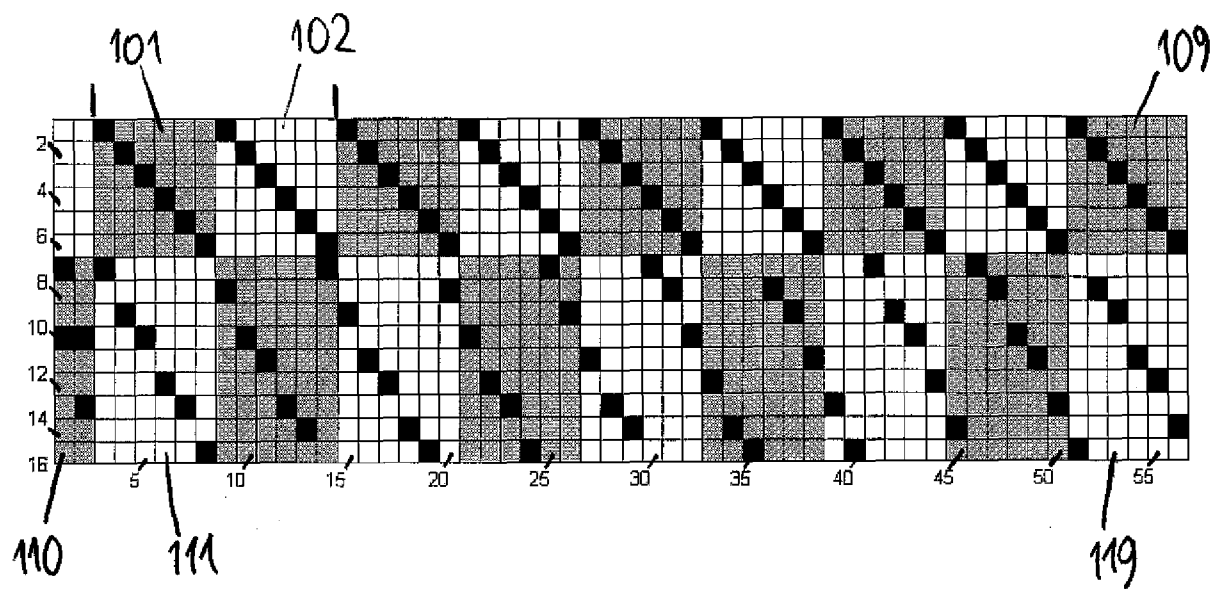


Figure 1

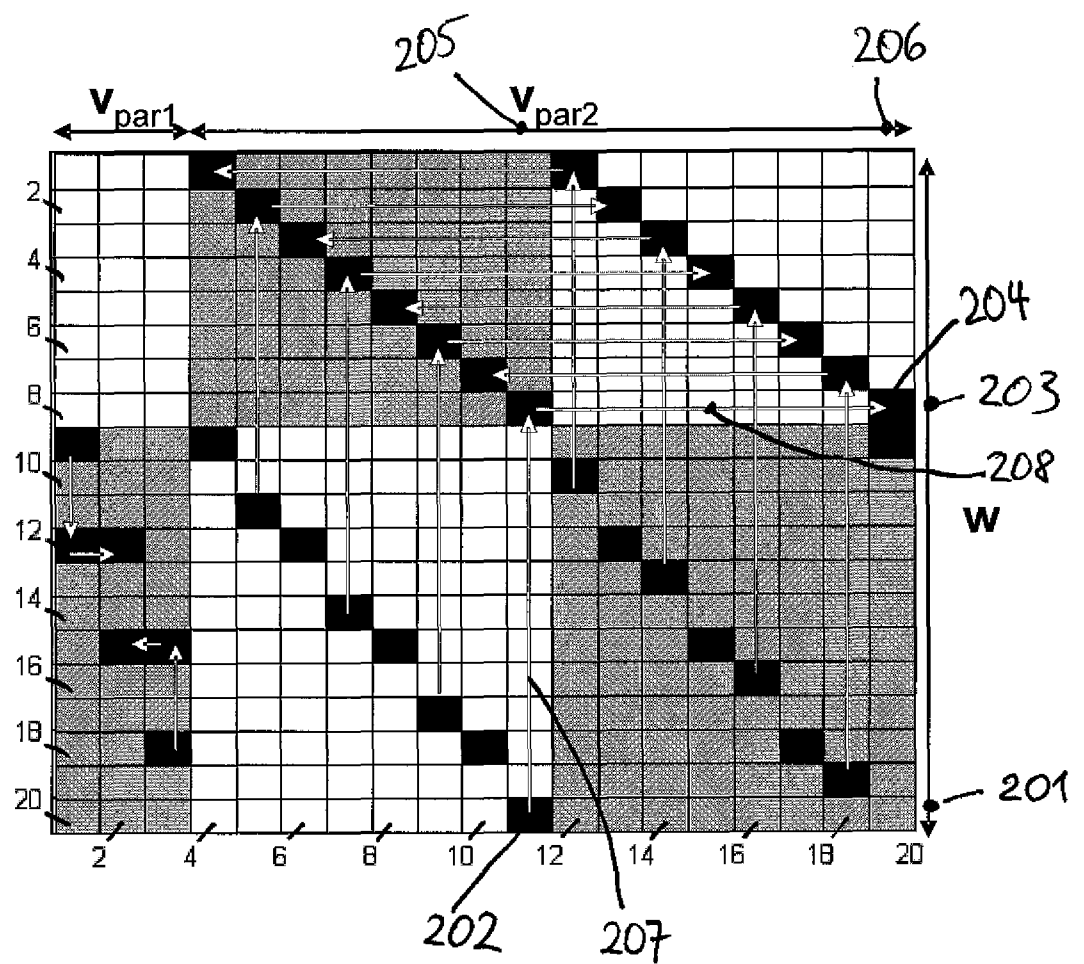


Figure 2

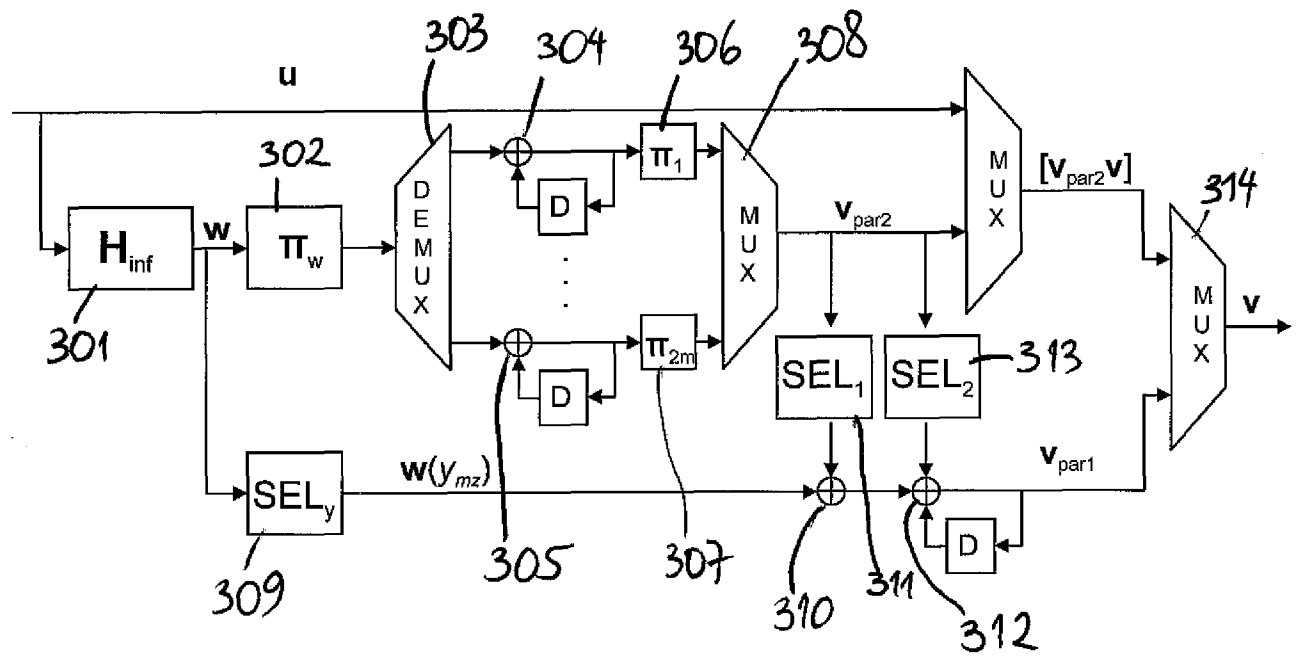


Figure 3

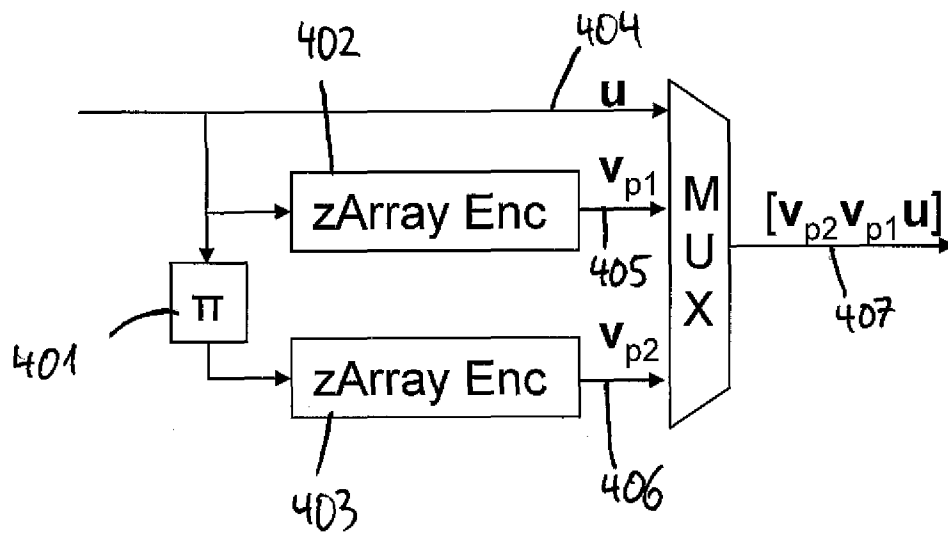


Figure 4

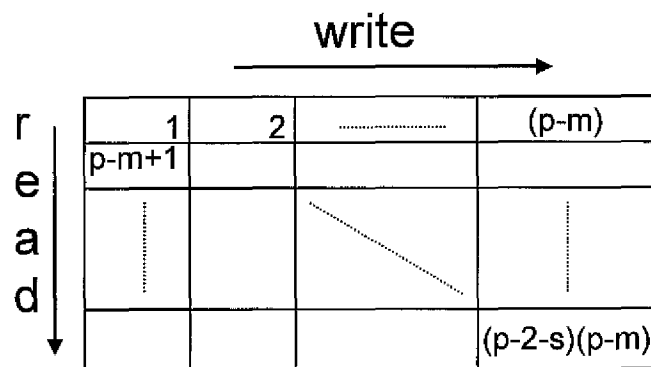


Figure 5

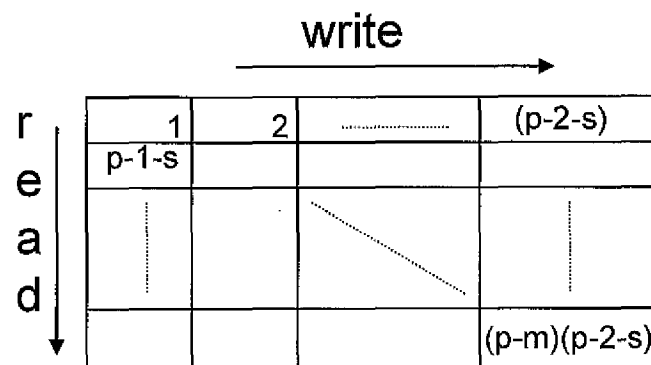


Figure 6

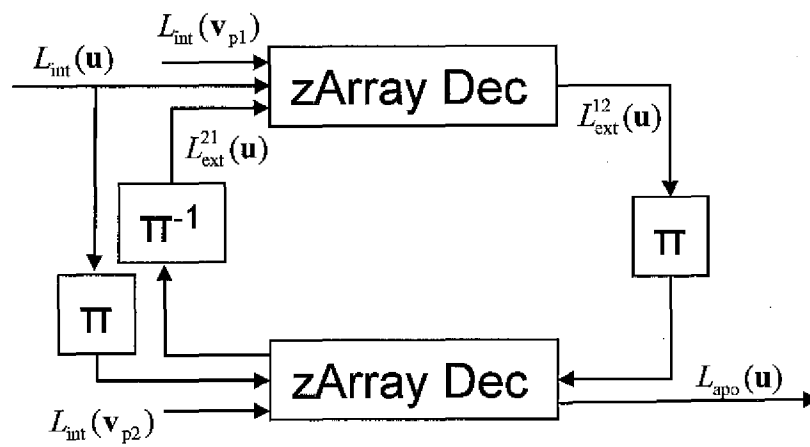


Figure 7

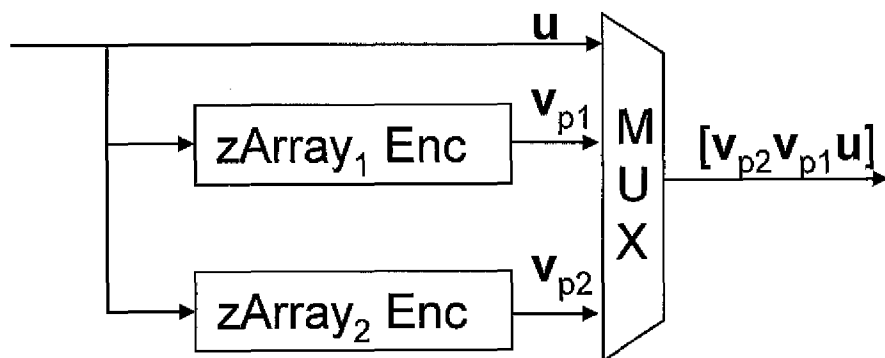


Figure 8

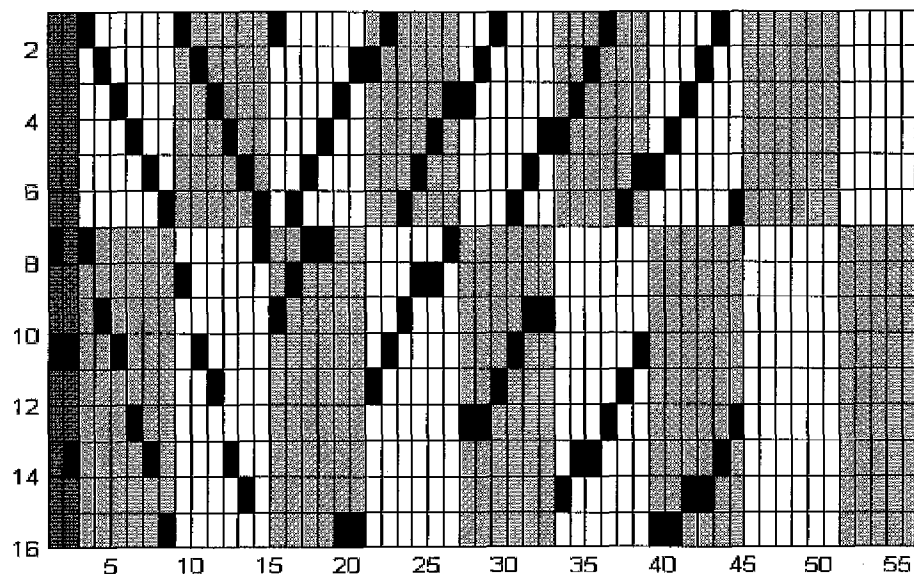


Figure 9

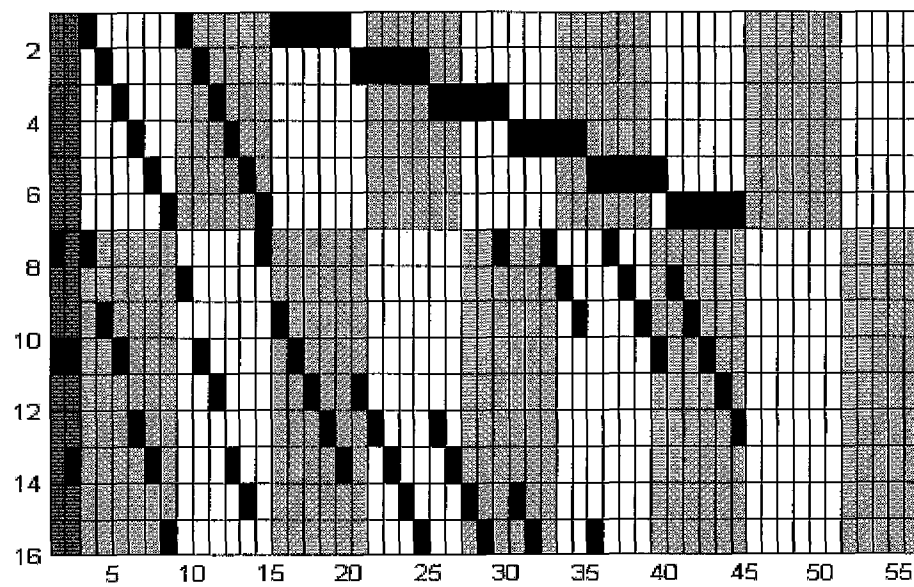


Figure 10

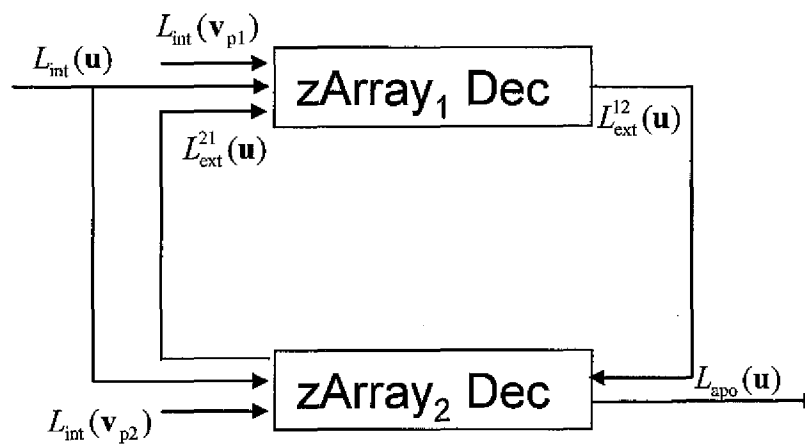


Figure 11

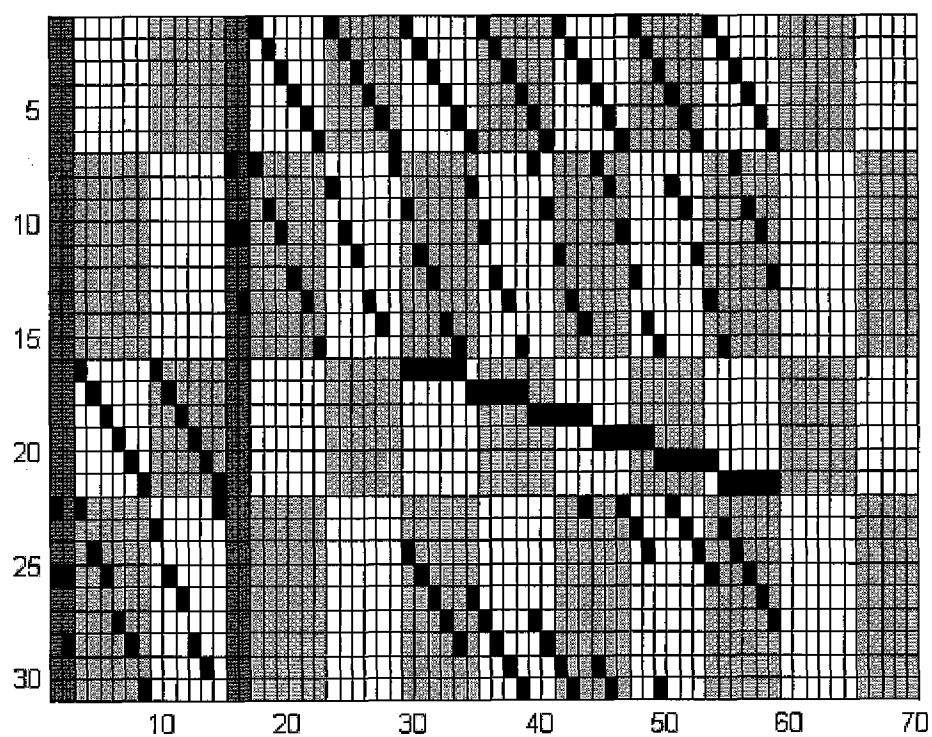


Figure 12