Lehrstuhl für
Eingebettete Syteme
der Informationstechnik
www.esit.rub.de

RUB

Design and Implementation of an LDPC-based FEC encoder/decoder suitable for Storage devices

Schriftliche Prüfungsarbeit für die Bachelorprüfung der Fakultät für Elektrotechnik und Informationstechnik an der Ruhr-Universität Bochum (Bachelor-Prüfungsordnung für den Studiengang „Elektrotechnik und Informationstechnik" an der Ruhr-Universität Bochum vom 12. August 2013)

vorgelegt von:

Henry Bathke

Datum: October 1, 2018

erster Betreuer: Prof. Dr.-Ing Michael Hübner
zweiter Betreuer: M. Sc. Keyvan Shahin

Eidesstattliche Erklärung

Ich erkläre, dass ich keine Arbeit in gleicher oder ähnlicher Fassung bereits für eine andere Prüfung an der Ruhr-Universität Bochum oder einer anderen Hochschule eingereicht habe.

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Die Stellen, die anderen Quellen dem Wortlaut oder dem Sinn nach entnommen sind, habe ich unter Angabe der Quellen kenntlich gemacht. Dies gilt sinngemäß auch für verwendete Zeichnungen, Skizzen, bildliche Darstellungen und dergleichen.

Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Ich erkläre mich damit einverstanden, dass die digitale Version dieser Arbeit zwecks Plagiatsprüfung verwendet wird.

Official Declaration

Hereby I declare, that I have not submitted this thesis in this or similar form to any other examination at the Ruhr-Universität Bochum or any other Institution of University.

I officially ensure, that this paper has been written solely on my own. I herewith officially ensure, that I have not used any other sources but those stated by me. Any and every parts of the text which constitute quotes in original wording or in its essence have been explicitly referred by me by using official marking and proper quotation. This is also valid for used drafts, pictures and similar formats.

I also officially ensure, that the printed version as submitted by me fully confirms with my digital version. I agree that the digital version will be used to subject the paper to plagiarism examination.

Not this English translation, but only the official version in German is legally binding.

Datum / Date

Unterschrift / Signature

1. Abstract

Contents

1. Abstract **3**

2. Motivation **7**

3. Error Correcting Codes **8**

3.1. Low-Density Parity-Check (LDPC) Codes 8

3.1.1. Quasicyclic LDPC (QC-LDPC) codes 9

3.1.2. Encoding 9

3.1.3. Decoding 12

4. Field Programmable Gate Array (FPGA) **13**

5. Approach **14**

6. Implementation **15**

7. Results **16**

A. Appendix **17**

List of Figures

3.1. An example Tanner graph. 8
3.2. Structure of a matrix in approximate lower triangular form. 11

List of Tables

3.1. Calculations for $p_1^T = \boldsymbol{\phi}^{-1}\left(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A} + \boldsymbol{C}\right)s^T$ 11
3.2. Calculations for $p_2^T = -\boldsymbol{T}^{-1}\left(\boldsymbol{A}s^T + \boldsymbol{B}p_1^T\right)$ 12

2. Motivation

3. Error Correcting Codes

For modern communications systems reliable data transmission and storage ist required. To achieve this goal usually error correcting codes are used. There are different possible codes available for error correction, but I will restrain myself to LDPC[2] codes in this thesis. As these codes can archive good performance and can be used at large block lengths[5]. This is especially useful for use with NAND based solid state drives.

When describing a block code there are important parameters as the message length k. The message is what is given into the encoder and the result from the decoder. The block length n, and the rate $R = k/n$.

3.1. Low-Density Parity-Check (LDPC) Codes

The following section will describe LDPC codes invented by Robert Gallager[2]. Starting with a graph representation I will describe the LDPC code and then continue with a matrix representation. LDPC codes can be shown as a bipartite graph also called Tanner graph[6] based on their inventor. figure 3.1 shows an example of one, where the check and parity nodes are connected by edges. This is an effective representation, moreover it will also help understanding the decoding algorithm later.

Instead of using the Tanner graph one can also use a matrix representation. In this matrix the ones represent the edges of the graph. Usually for a LDPC code the matrix is sparse or low density as the name implies. In equation (3.1) a matrix representing the same code as

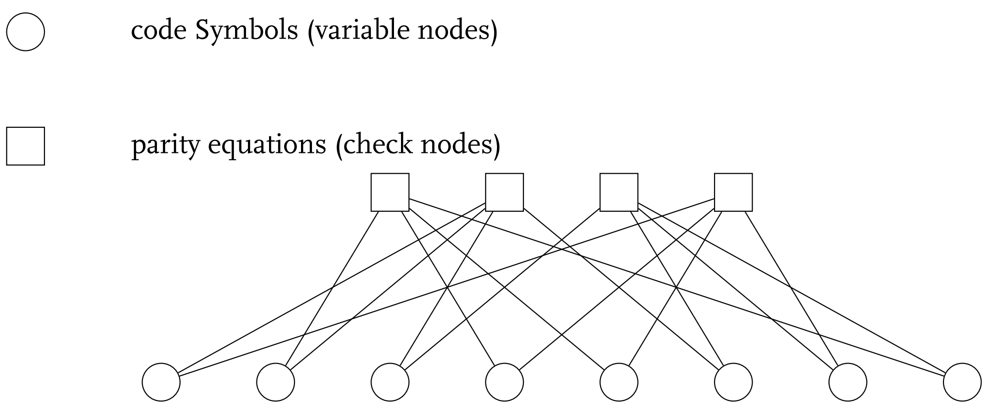

Figure 3.1.: An example Tanner graph.

in the graph in figure 3.1 is shown. The $\boldsymbol{H}$ matrix is of size $(n-k) \times n$. And the possible code words are given by the null space of $\boldsymbol{H}$, so in other words c is a code word if and only if $c\boldsymbol{H}^T = \mathbf{0}$[4].

$$\boldsymbol{H} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{3.1}$$

3.1.1. Quasicyclic LDPC (QC-LDPC) codes

LDPC codes can be difficult to implement, especially randomly generated ones. On the other hand structured codes can be devised to be more easily implemented. One class of these structured codes are quasicyclic LDPC codes. In these codes the parity check matrix is only built from circulant matrices and zero matrices[1]. The circulant matrices are defined by their first row as the following rows are the first one shifted. The base matrix specifies where zero matrices and where rotated circulant matrices are placed. Each circulant matrix has size $p\times$. The parity check matrix has size $(n-k) \times n = p(N-K) \times pN$. Thus the base matrix $\boldsymbol{B}$ has $(N-K) \times N$ entries. Often times the circulant matrix is the identity matrix and is rotated by the corresponding amount in the base matrix.

$$\boldsymbol{B} = \begin{bmatrix} \\ \\ \\ \end{bmatrix} \tag{3.2}$$

3.1.2. Encoding

Genrator Matrix

For encoding the probably simplest algorithm is transforming the parity check matrix into systematic form $\boldsymbol{H} = \begin{bmatrix} -\boldsymbol{A^T} & \boldsymbol{I_{n-k}} \end{bmatrix}$. Where $\boldsymbol{I_{n-k}}$ is a $n-k \times n-k$ identity matrix and $\boldsymbol{A}$ has $k \times n-k$ elements. To archive this form one could for example use gaussian elimination. With $\boldsymbol{A}$ known we can construct the generator matrix $\boldsymbol{G} = \begin{bmatrix} \boldsymbol{I_k} & \boldsymbol{A} \end{bmatrix}$. Now encoding can be done with a simple matrix multiplication. With u the information word and v the code word is given by $v = u\boldsymbol{G}$.

Take for example the matrix from equation (3.1). If we use gaussian elimination to bring the right side to identity we are left with equation (3.3). Now we take the left part of the matrix and transpose it to get $\boldsymbol{A}$. With we build $\boldsymbol{G}$ in equation (3.4).

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$G = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

The main disadvantage of this strategy is the high computational complexity. When transforming the parity check matrix into systematic form we have a complexity of $\mathcal{O}(n^3)$. This is not to bad as it will mostly be done offline and only the $\boldsymbol{G}$ matrix stored in the encoder, but the bigger problem is that due to the gaussian elimination the matrix is no longer sparse. Thus the matrix multiplication will result in a complexity of $\mathcal{O}(n^2)$[3].

Approximate Lower Triangular Form

Richardson and Urbanke[4] describe a way to reorder the parity check matrix to reduce the encoding complexity. They bring the matrix into a so called approximate lower triangular form. This is done by only doing row and column permutation, so the low density of the matrix is kept. The resulting structure of the matrix is shown in figure 3.2. Especially advantageous is the reduced complexity for encoding, here the encoding complexity is reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(n+g^2)$, where g is the gap. This gap is the number of rows that cannot be brought into triangular form, as seen in figure 3.2. We can also write

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{T} \\ \boldsymbol{C} & \boldsymbol{D} & \boldsymbol{E} \end{bmatrix} \tag{3.5}$$

the submatrices all have the dimensions given in figure 3.2. By multiplying equation (3.5) with

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{E}\boldsymbol{T}^{-1} & \boldsymbol{I} \end{bmatrix} \tag{3.6}$$

the resulting matrix is

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{T} \\ -\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}+\boldsymbol{C} & -\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{B}+\boldsymbol{D} & \boldsymbol{0} \end{bmatrix} \tag{3.7}$$

. By splitting the codeword into three parts $c = \begin{bmatrix} s & p_1 & p_2 \end{bmatrix}$ and applying the definition for valid code words $\boldsymbol{H}^T = \boldsymbol{0}$. It splits into

$$\boldsymbol{A}s^T + \boldsymbol{B}p_1^T + \boldsymbol{T}p_2^T = 0 \tag{3.8}$$

$$\left(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}+\boldsymbol{C}\right)s^T + \left(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{B}+\boldsymbol{D}\right)p^T = 0 \tag{3.9}$$

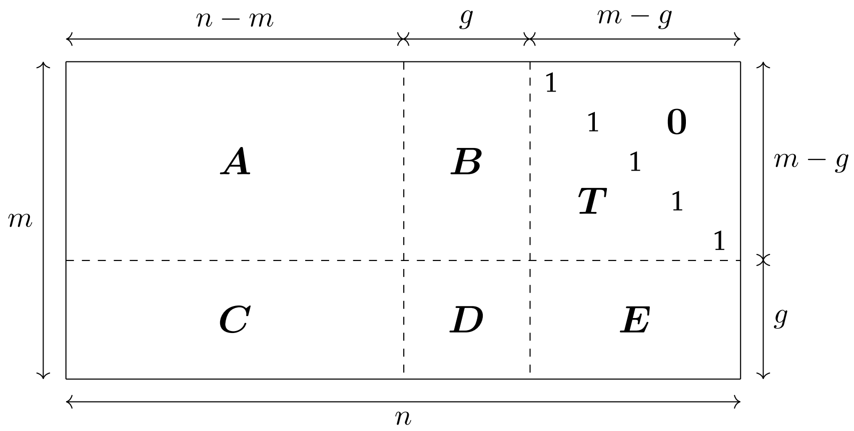

Figure 3.2.: Structure of a matrix in approximate lower triangular form.

Operation	Type
$\boldsymbol{A}s^T$	sparse multiplication
$\boldsymbol{T}^{-1}\boldsymbol{A}s^T$	sparse back substitution
$-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}s^T$	sparse multiplication
$\boldsymbol{C}s^T$	sparse multiplication
$\left(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}s^T\right)+\left(\boldsymbol{C}s^T\right)$	vector addition
$\boldsymbol{\phi}^{-1}\left(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}s^T+\boldsymbol{C}s^T\right)$	dense $g \times g$ multiplication

Table 3.1.: Calculations for $p_1^T = \boldsymbol{\phi}^{-1}\left(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}+\boldsymbol{C}\right)s^T$

. The resulting equations for p_1 and p_2 are

$$p_1^T = -\boldsymbol{\phi}^{-1}\left(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}+\boldsymbol{C}\right)s^T \tag{3.10}$$

$$p_2^T = -\boldsymbol{T}^{-1}\left(\boldsymbol{A}s^T+\boldsymbol{B}p_1^T\right) \tag{3.11}$$

. The complexity of the computations can be reduced by computing $\boldsymbol{\phi}^{-1}$ offline. Offline meaning that it is precomputed and when encoding multiplying by the matrix. All the other matrix multiplications from equations (3.10) and (3.11) are done separately. Multiplications by $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{E}$ are sparse and the resulting complexity for these is $\mathcal{O}(n)$. The multiplication with $\boldsymbol{T}^{-1}$ is replaced by the system $x^T = \boldsymbol{T}y^T$. As $\boldsymbol{T}$ is a sparse lower triangular matrix the system can be solved by back substitution in $\mathcal{O}(n)$. The only part with higher complexity is the multiplication with the dense $g \times g$ matrix $\boldsymbol{\phi}$ where the complexity is $\mathcal{O}(g^2)$.

do i want do describe the algorithm to get into alt form? yes!

Operation	Type
$\boldsymbol{A}s^T$	sparse multiplication
$\boldsymbol{B}p_1^T$	sparse multiplication
$\left(\boldsymbol{A}s^T\right) + \left(\boldsymbol{B}p_1^T\right)$	vector addition
$-\boldsymbol{T}^{-1}\left(\boldsymbol{A}s^T + \boldsymbol{B}p_1^T\right)$	sparse back substitution

Table 3.2.: Calculations for $p_2^T = -\boldsymbol{T}^{-1}\left(\boldsymbol{A}s^T + \boldsymbol{B}p_1^T\right)$

3.1.3. Decoding

Decoding LDPC codes is a nontrivial problem, in fact maximum likelihood decoding is computationally infeasible. Therefore other decoding methods were developed.

decode dat

4. Field Programmable Gate Array (FPGA)

In modern hardware design it is advantageous to have reprogrammable hardware elements. To create reconfigurable logic circuits basic elements consisting of lookup tables (LUT) and registers is built into an array and connected by programmable interconnects.

write some basics about FPGA

5. Approach

6. Implementation

7. Results

A. Appendix

Bibliography

[1] M. P. C. Fossorier. “Quasicyclic low-density parity-check codes from circulant permutation matrices”. In: 50.8 (2004), pp. 1788–1793. DOI: `10.1109/TIT.2004.831841`.

[2] Robert R. Gallager. “Low-Density Parity-Check Codes”. In: (1963).

[3] Hanghang Qi and Norbert Goertz. “Low-Complexity Encoding of LDPC Codes: A New Algorithm and its Performance”. In: ().

[4] Thomas J. Richardson and Rüdiger L. Urbanke. *Efficient Encoding of Low-Density Parity-Check Codes*. 2001. DOI: `10.1109/18.910579`.

[5] Bashar Tahir, Stefan Schwarz, and Markus Rupp. “BER comparison between Convolutional, Turbo, LDPC, and Polar codes”. In: (2017). DOI: `10.1109/ICT.2017.7998249`.

[6] M. Tanner. “A recursive approach to low complexity codes”. In: (1981). DOI: `10.1109/TIT.1981.1056404`.