Introduction of Low-density Parity-Check decoding Algorithm design

Master thesis performed in Electronics Systems by Florent Pirou

Reg nr: LiTH-ISY-EX-3529-2004

Linköping, 2004
Introduction of Low-density Parity-Check decoding Algorithm design

Master thesis in Electronics Systems at Linköping Institute of Technology by

Florent Pirou

Reg nr: LiTH-ISY-EX-3529-2004

Supervisor: Pascal Urard - STMicroelectronics
Examiner: Kent Palmvist - LIU-ISY departement

Sammanfattning

Abstract
Recently, low-density parity-check (LDPC) codes have attracted much attention because of their excellent error correcting performance and highly parallelizable decoding scheme. However, the effective VLSI implementation of an LDPC decoder remains a big challenge and is a crucial issue in determining how well we can exploit the benefits of the LDPC codes in the real applications. In this master thesis report, following a coding background, we describe Low-Density Parity-Check codes and their decoding algorithm, and also requirements and architectures of LPDC decoder implementations.

Nyckelord

Keyword
LDPC, Low-density parity-check codes, channel coding, FEC, iterative algorithm, Gallager, message-passing algorithm, belief propagation algorithm,
Abstract

Recently, low-density parity-check (LDPC) codes have attracted much attention because of their excellent error correcting performance and highly parallelizable decoding scheme. However, the effective VLSI implementation of an LDPC decoder remains a big challenge and is a crucial issue in determining how well and quickly we can exploit the benefits of the LDPC codes in the real applications. In this master thesis report, following a short introduction about channel coding background, we describe Low-Density Parity-Check codes and their decoding algorithm, and also requirements and architectures of LPDC decoder implementations.

Keywords: LDPC, Low-density parity-check, channel coding, FEC, iterative algorithm, Gallager, message-passing algorithm, belief propagation algorithm,...
Acknowledgment

I would like to thank M. Pascal URARD, senior engineer in DSP design and team leader in charge of my work placement, for his confidence, his patience and for establishing a generous working environment. I would also like to thank all SHIVA team members for their help and support all along my intership. I enjoyed working in SHIVA team over the last six months.
I would like to thank Kent Palmvist, from ISY departement of Linköping university, for his support along my thesis work.
Notation

The symbol "|" between two matrices denotes concatenation, for example systematic parity-check matrix might be written \([P \mid I_{n-k}]\).

Symbols

- \(d_H\) Hamming distance
- \(K\) Source block length of a code
- \(N\) Transmitted block length of a code
- \(G^T\) Generator matrix
- \(H\) Parity-check matrix
- \(s\) Source vector, length \(K\)
- \(t\) Transmitted vector, length \(N\)
- \(r\) Received vector, length \(N\)
- \(n\) Noise vector
- \(N_0\) the one-sided noise power spectral density
- \(E_b\) energy per bit
- \(P/N\) Signal power[watt] over the total noise power of the channel[watt]
- \(W\) bandwidth of the channel [Hz]
- \(\eta\) bandwidth efficiency [bit/s/Hz]
- \(T\) duration of a symbol
- \(\lambda_i\) LLR of the \(i^{th}\) received digit
- \(R_{j\to i}\) Check-node processor element output register
- \(Q_{i\to j}\) bit-node processor element output register

Operators and functions

\(\mathcal{F}(\beta)\) check-node computation function varies as \(ln\) of \(tanh\)
Abbreviations

LDPC  Low-Density Parity Check.
FEC   Forward Error Control.
LLR   Log-Likelihood Ratio.
SNR   Signal-to-Noise Ratio.
AWGN  Additive white gaussian noise
BER   Bit-error ratio
## Contents

1 Error Control Coding ........................................ 3
   1.1 Modern Telecommunications ................................. 4
      1.1.1 System overview ..................................... 4
      1.1.2 Source and Channel coding ............................ 5
      1.1.3 Channel capacity or Shannon limit ...................... 5
      1.1.4 Error coding performances .............................. 6
   1.2 Forward Error Correction .................................. 8
      1.2.1 Hard and Soft decision decoding ......................... 8
      Log-Likelihood Ratio - LLR ................................ 9
   1.3 Linear block codes ....................................... 9
      1.3.1 Parity-Check codes .................................. 10
      Hamming distance .......................................... 11

2 Low-Density Parity-Check codes ............................... 13
   2.1 History .................................................. 14
      2.1.1 Gallager’s work ........................................ 14
      2.1.2 Bipartite graph or Tanner graph ......................... 17
   2.2 LDPC codes viewed as Concatenated Codes ................... 19
   2.3 Iterative Decoding scheme ................................ 20
      2.3.1 Message-passing algorithm ............................. 22
      2.3.2 Soft decision decoding and Belief propagation algorithm ......................................................... 23
         Initialisation ............................................. 24
         Iterative loop ............................................ 24
         Check-node update ....................................... 25
         Bit-node update ........................................... 25
         Hard decision making .................................... 26

3 Implementation of LDPC decoding algorithm .................... 29
   3.1 Requirements ............................................ 30
      3.1.1 Impact of code construction ............................ 30
      3.1.2 Message-passing requirements .......................... 30
   3.2 LDPC decoder architecture ................................ 31
      3.2.1 Iterative decoding architectures ......................... 32
Introduction

The low-density parity-check (LDPC) codes, originated by Robert Gallager at the Massachusetts Institute of Technology in 1962 [6], is key technology for forward error correction in the next-generation satellite digital video broadcasting standard, DVB-S2. This technology is also very suitable in many applications such as hard disk drive or 4G mobile phones.

Intellectual-property rights covering its theory or concept are not an issue, because LDPC itself was invented so long ago.

Together with the concept of iterative decoding via “message-passing algorithm ”, Gallager’s ideas remained largely neglected until the recent rediscoveries made by Mackay-Neal[5] and Wiberg[11]. It was shown that this class of codes is capable of approaching capacity limit at low decoding complexity.

LDPC codes were further extended in [7] to include irregular LDPC codes yielding bit-error rates that surpass the performance of the best (turbo) codes known so far.

Compared to other codes, decoding LDPC codes requires simpler processing. However, the decoding mechanism involved can be seen as a pseudo-random “internal interleaver”, which leads to implementation challenges.

This report is composed of three main parts:

• A background of Error control coding,

• Description of Low-Density Parity-Check codes and their decoding algorithms,

• A presentation of decoding algorithm implementations.
Chapter 1

Error Control Coding

This chapter gives a background of information theory and a general understanding of forward error coding.

Contents

1.1 Modern Telecommunications .......................... 4
   1.1.1 System overview .................................. 4
   1.1.2 Source and Channel coding ....................... 5
   1.1.3 Channel capacity or Shannon limit ............... 5
   1.1.4 Error coding performances ...................... 6
1.2 Forward Error Correction ............................. 8
   1.2.1 Hard and Soft decision decoding ................. 8
1.3 Linear block codes .................................. 9
   1.3.1 Parity-Check codes ............................... 10
1.1 Modern Telecommunications

In the late 1940’s, Claude Shannon of Bell Laboratories developed a mathematical theory of information that profoundly altered our basic thinking about communication, and stimulated considerable intellectual activity, both practical and theoretical. He started the field of coding theory by demonstrating [9] that it is theoretically possible to achieve error free transmission on noisy communication channel through coding. This theory, among other things, gives us some fundamental boundaries within which communication takes place.

1.1.1 System overview

A general communication system shown schematically in Fig. 1.1 consists of essentially five parts:

1. An information source which produces a message or sequence of messages to be communicated to the receiving terminal.
2. A transmitter which operates on the message in some way to produce a signal suitable for transmission over the channel.
3. The channel is merely the medium used to transmit the signal from transmitter to receiver. It may be a pair of wires, a coaxial cable, a band of radio frequencies, a beam of light, etc.
4. The receiver ordinarily performs the inverse operation of that done by the transmitter, reconstructing the message from the signal.

Figure 1.1. Schematic diagram of a general communication system.
5. The destination is the person (or thing) for whom the message is intended. We wish to consider certain general problems involving communication systems.

### 1.1.2 Source and Channel coding

Information theory provides profound insights into the situation pictured in Fig.
1.2. The objective is to provide the source information to the receiver with the
greatest fidelity. To that end, Shannon introduced [9] the general idea of coding.

The aim of source coding is to minimize the bit-rate required for representation
of the source at the output of a source coder, subject to a constraint on fidelity. In other words, error correcting codes are used to increase bandwidth and power efficiency of communication systems. Shannon showed [9] that the interface be-
tween the source coder and the channel coder can be a bit stream, regardless of
the nature of the source and channel.

The aim of channel coding is to maximize the information rate that the chan-
nel can convey sufficiently reliably (where reliability is normally measured as a bit
error probability).

Our primary focus in this report is on the channel decoder.

### 1.1.3 Channel capacity or Shannon limit

Shannon [9] defined the channel capacity \( C \) as the maximum rate for which inform-
action can be transmitted over a noisy channel. He stated that if it is possible to
distinguish reliably \( M \) different signal functions of duration \( T \) on a channel, we can
say that the channel can transmit \( \log_2 M \) bits in the time \( T \). The rate of trans-
mition is then \( (\log_2 M)/T \). More precisely, the channel capacity may be defined as,

\[
C = \lim_{T \to \infty} \frac{\log_2 M}{T}
\]

He approached [9] the maximum rate of the transmission of binary digits by,

\[
C = W \log_2 \left( 1 + \frac{P}{N} \right) \quad \text{[bits/second]} \quad (1.1)
\]
where \( W \) is the channel bandwidth starting at zero frequency, and \( P/N \) is the *signal-to-noise ratio (SNR)*. Ordinarily, as we increase \( W \), the noise power \( N \) in the band will increase proportionally; \( N = N_0 W \) where \( N_0 \) is the noise power per cycle. In this case, we have

\[
C = W \log_2 \left( 1 + \frac{P}{N_0 W} \right) \tag{1.2}
\]

If we let \( W_0 = P/N_0 \), i.e., \( W_0 \) is the band for which the noise power is equal to the signal power, this can be written

\[
\frac{C}{W_0} = \frac{W}{W_0} \log_2 \left( 1 + \frac{W_0}{W} \right) \tag{1.3}
\]

In Fig 1.3 [9] the function \( C/W_0 \) is plotted as function of \( C/W \). As we increase the band, the capacity increases rapidly until the total noise power accepted is approximately equal to the signal power. After this, the increase is low, and it approaches an asymptotic value \( \log_2 e \) time the capacity for \( W = 0 \).

![Figure 1.3. Capacity channel as function of the bandwidth](image)

### 1.1.4 Error coding performances

For each channel code we compare the performance of an uncoded system with a coded system. This is done by considering the Signal-to-Noise Ratio (SNR) required at the detector input to achieve a fixed probability of error. The coded system can tolerate a lower SNR than uncoded system. This difference (in dB) is called the *coding gain*, see Fig 1.4. The coding gain can be, alternatively,
viewed as a decrease in the signal power allowable in the coded system for a fixed noise power, or an increase in allowable noise power for a fixed signal power.

![Figure 1.4. Performance of various coding scheme (Convolutional Code, Turbo Code, ...)](image)

For AWGN channel, the SNR can be written as following:

$$S \over N = \eta_{\text{max}} \frac{E_b}{N_0}$$

where $\eta_{\text{max}}$ represents the maximum spectral efficiency and is defined as,

$$\eta_{\text{max}} = \frac{C}{W} \quad \text{[bits/s/Hz]} \quad (1.4)$$
In Substituting Eq. 1.4 from Eq. 1.1 we get,
\[
\eta_{\text{max}} = \log_2 \left( 1 + \eta_{\text{max}} \frac{E_b}{N_0} \right) \Rightarrow \frac{E_b}{N_0} = \frac{2^{\eta_{\text{max}}}-1}{\eta_{\text{max}}} \tag{1.5}
\]

It defines the minimum $E_b/N_0$ required for error-free transmission.

Let $R$ be the rate of the encoder and $M$ the number of states for the given modulation type ($M_{\text{BPSK}} = 2, M_{\text{QPSK}} = 4, M_{\text{8PSK}} = 8,...$). Thus, the spectral efficiency is defined as $\eta_{\text{max}} = R \log_2 M$. In substituting $\eta_{\text{max}}$ in Eq. 1.5, we obtain:
\[
\frac{E_b}{N_0} = \frac{2^{R \log_2 M} - 1}{R \log_2 M}.
\]

### 1.2 Forward Error Correction

As mentioned earlier, A channel coder proceeds the line coder as shown in Fig. 1.2, and is designed specifically for one of the following purposes:

- **Error Detection** can be used to increase reliability. For example, a code can be created that will detect all single bit errors, not just a fraction of them as in line coding. The most straightforward error detection can ensure reliability by causing the retransmission of the blocks of data until they are correctly received. In-service monitoring and error detection are similar, except that with in-service monitoring it is not necessary to detect all or even the majority of errors, since all we need is an indication of the performance of the system rather than the location of each and every error.

- A more ambitious goal is **error correction**. This carries error detection on step further by actually using the redundancy to correct transmission errors.

- Instead of correcting errors after they occur, a still more ambitious goal is **error prevention**. The probability of error is reduced by combining detection and decoding to get a technique known as **Soft decision decoding**.

#### 1.2.1 Hard and Soft decision decoding

Two fundamentally different types of decoding are used, hard and soft. Hard decoding is the easiest to understand, and is illustrated in Fig. 1.5. A slicer makes a **hard decision**, doing its best to detect the transmitted symbols (complete with their redundancy). Redundancy is then removed by inverting the mapping performed in the encoder. Since not all bit patterns are permitted by the code, the encoder can detect or correct bit errors.

A **soft decision decoder**, by contrast, makes direct decision on the information bits without making intermediate decision about transmitted symbols. Soft decoding, starts with the continuous-valued samples of received signal, processing them
Figure 1.5. In hard decision decoding of the channel code, a slicer makes the decision about incoming symbols. The symbols are decoded into bits by a line decoder, and the channel decoder maps these coded bits into uncoded bits.

directly to detect bit sequence, as shown in Fig. 1.6. Instead of correcting errors after they occur, as done by hard decoder, a soft decoder prevents errors by combining slicing and line decoding with channel decoding. We can think of soft decoding as a combination of slicing and removing redundancy.

Figure 1.6. A soft decoder operates directly on samples of the incoming signal

Soft decoding is capable of providing better performance, at expense of implementation complexity, since it makes use of information that slicer would otherwise throw away.

Log-Likelihood Ratio - LLR  By definition, Log-Likelihood Ratio of bit is the log of ratio of probability that the bit is 1 divided by probability that bit is 0.

\[
LLR_d = \ln \left( \frac{P_r[x_d = 0 | y, S]}{P_r[x_d = 1 | y, S]} \right)
\]

1.3 Linear block codes

Both hard and soft decoding are used for block codes. An \( (n, k) \) block coder maps block of \( k \) source bits into blocks of \( n \) coded bits, where \( n > k \). Such a block code is said to have code rate \( k/n \), where the terminology refers to the fraction of the total bit rate devoted to information bits. Thus, there are more coded bits than source bits, and the coded bits have redundant information about the source bits. In other words, the \( n \) coded bits depend only on the \( k \) source bits, so the coder is
1.3.1 Parity-Check codes

A parity check code is fully characterized by a generator matrix $G^T$ which consists only of zeros and ones.

$$G^T = \begin{bmatrix} 1 & 1 & \ldots & 0 & 1 \\ 0 & 1 & \ldots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 1 \end{bmatrix}$$

Codes which have a generator matrix of the form Example 1.6 are called systematic.

$$G^T = [I_k \mid P] = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$ (1.6)

Where $I_k$ is a $k$-dimensional identity matrix and $P$ is binary matrix.

If the input and output are collected into the row vectors $s$ and $t$, then the first $k$ output bits $t_1 \ldots t_k$ are exactly the input bits $s_1 \ldots s_k$. Thus,

$$t = (G^T s) \mod 2$$

where the addition that occurs in the matrix multiplication is modulo-two.

For a particular binary row vector $s$, $t$ is called codeword. A code is the set of all possible codewords. Every codeword is a modulo-two summation of rows of the generator matrix. It has been demonstrated [1] that all parity-check codes are linear, that all linear block codes are parity-check codes with some generator matrix.

The channel adds noise $n$ to the vector $t$ with the resulting received signal $r$ being given by,

$$r = (G^T s + n) \mod 2$$

The decoder task is to deduce $s$ given the received message $r$, and assumed noise properties of the channel. The optimal decoder returns the message $s$ that minimizes the following probability,

$$P(s \mid r, G) = \frac{P(r \mid s, G)P(s)}{P(r \mid G)}$$
1.3 Linear block codes

If the prior probability of $s$ is assumed uniform, and the probability of $n$ is assumed to be independent of $s$, then it is convenient to introduce the $(N-K) \times N$ parity-check matrix, which in systematic form is

$$H = [\hat{P} \mid I_{N-K}]$$

The parity-check has the property

$$HG^T = 0 \mod 2 \Rightarrow H \cdot n = H \cdot r \mod 2$$

Any other $(N-K) \times N$ matrix $A$, whose rows span the same coding-space as $H$, is a valid parity-check matrix.

Decoding problem thus reduces, given the above assumptions, to the task of finding the most probable noise vector such that $H \cdot n \mod 2 = z$ where $z$ is called the syndrome vector.

**Example.** The parity-check matrix for (7,4) code of Example 1.6 is

$$H = [P \mid I_{n-k}] = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

(1.7)

An example of a codeword is $r_1 = [0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1]$, which satisfies $H \cdot r_1 = 0$. By contrast, $r_2 = [0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1]$ is not a codeword, since $H \cdot r_2 = [1 \ 0 \ 0]$.

**Hamming distance**

The Hamming distance $d_H(c_1, c_2)$ between $c_1$ and $c_2$ in $V_n$ is the number of differing bits between $c_1$ and $c_2$. The Hamming weight $w_H(c)$ of $c$ in $V_n$ is the number of ones in $c$.

An $(n,k)$ linear block code $C$ is a $k$-dimensional subspace of $V_n$. Subspace means that $C$ is a vector space, and hence is closed under vector addition, i.e, if $c_1$ and $c_2$ are in $C$, then $c_1 \oplus c_1 \in C$.

Hence,

$$d_{H,\text{min}} = \min_{c \in C} \{w_H(c)\}$$

To find the minimum Hamming distance in a linear code we only need to find the minimum Hamming weight in linear code.
Chapter 2

Low-Density Parity-Check codes

A central challenge in coding theory has always been to design coding schemes that come close to achieving capacity with practical implementation. Nowadays, Low-density parity-check codes are probably the best competitors of this challenge.

Contents

2.1 History .................................................. 14
2.1.1 Gallager’s work ...................................... 14
2.1.2 Bipartite graph or Tanner graph ................. 17
2.2 LDPC codes viewed as Concatenated Codes ...... 19
2.3 Iterative Decoding scheme ............................ 20
2.3.1 Message-passing algorithm ....................... 22
2.3.2 Soft decision decoding and Belief propagation algorithm 23

13
2.1 History

Much of efforts in coding theory over the 50 years has been focused on the construction of highly structured codes with large minimum distance. The structure keeps the decoding complexity manageable, while large minimum distance is supposed to guarantee good performance. This approach, however, is not without drawbacks:

- First, a good code should be randomly chosen codes with high probability. But this differs with the goal of finding highly structured codes that have simple decoding scheme.
- Second, close to channel capacity, minimum distance is a poor parameter for the performance measure of real interest.

Since 1993, the new techniques have enabled to reach performance approaching the Shannon limit of the additive white Gaussian noise channel within a factor of a dB. Moreover, the design complexity has only increased with a small factor compared to standard coding schemes like convolutional codes. These coding techniques, such as turbo codes or LDPC codes, use entirely different approach based on iterative decoding systems.

Recent years have seen significant improvements of our understanding and the ability to design iterative decoding systems. Moreover, it is now apparent that all aspects of the telecommunication chain can be included in the iterative processing framework: source coding, channel coding, modulation, equalization, multiple access, transmission via multiple antennas, and so on ... all of these areas are currently being attacked by iterative signal processing techniques.

2.1.1 Gallager’s work

In 1960 R. Gallager completed his Ph.D. thesis [2], Low-density parity-checking (or LDPC). In this remarkable thesis Gallager introduced at least two lasting concepts: A powerful bounding technique of coding systems, and LPDC codes together with their associated iterative decoding algorithm.

He demonstrated [6] how an arbitrary digit $d$ can be corrected even if its parity-check sets contain more than one transmission error, considering the tree structure Fig. 2.1. Digit $d$ is represented by node at the base of the tree, and each line rising from this node represents one of the parity-check sets containing digit $d$. The other digits in these parity-check sets are represented by the nodes on the first tier of the tree. The lines rising from the first tier to the second tier of the tree represent the other parity-check sets containing the digits in those parity-check sets.

Assuming that both digit $d$ and several digits in the first tier are transmission errors, Gallager [6] showed that the error-free digits in the second tier and their parity-check equations will allow correction of the errors in the first tier. This will allow correction of the digit $d$ on the second decoding attempt.

He extended his demonstration to code words from an $(n, j, k)$ code and using probabilistic methods he proved the theorem 2.1:
Theorem 2.1 Let $P_d$ be the probability that the transmitted digit in position $d$ is a 1 conditional on the received digit in position $d$, and let $P_{il}$ be the same probability for the $i^{th}$ digit in the $l^{th}$ parity-check equation of the first tier of Fig. 2.1. Let the digit be statistically independent of each other, and let $S$ be the event that the transmitted digits satisfy the $j^{th}$ parity-check equations on digit $d$. Then, 

$$
\frac{P_r[x_d = 0 | y, S]}{P_r[x_d = 1 | y, S]} = \frac{1 - P_d}{P_d} \prod_{i=1}^{j} \left[ \frac{1 + \prod_{i=1}^{k-1} (1 - 2P_{il})}{1 - \prod_{i=1}^{k-1} (1 - 2P_{il})} \right]
$$

(2.1)

It appeared to be more convenient to use Eq. 2.1 in terms of log-likelihood ratios. Let

$$
\ln \left( \frac{1 - P_d}{P_d} \right) = \alpha_d \beta_d \\
\ln \left( \frac{1 - P_{il}}{P_{il}} \right) = \alpha_d \beta_{il} \\
\ln \left( P_r[x_d = 0 | y, S] \right) = \alpha_d \beta_d \\
\ln \left( P_r[x_d = 1 | y, S] \right) = \alpha_d \beta_d
$$

Where $\alpha$ is the sign and $\beta$ the magnitude of the log-likelihood ratio. Considering,

$$
\left| \ln \left( \frac{1 - P_{il}}{P_{il}} \right) \right| = \beta_{il} \Rightarrow P_{il} = \frac{1}{1 + e^{\beta_{il}}}
$$

Figure 2.1. Parity check set tree
Then,

\[(1 - 2P_d) = 1 - \frac{2}{1 + e^{\beta_d}} = \frac{e^{\beta_d} - 1}{e^{\beta_d} + 1} = \tanh \left( \frac{\beta_d}{2} \right)\]

Thus,

\[\frac{1 + \prod_{i=1}^{k-1} (1 - 2P_{di})}{1 - \prod_{i=1}^{k-1} (1 - 2P_{di})} = \frac{1 + \prod_{i=1}^{k-1} \tanh \left( \frac{\beta_d}{2} \right)}{1 - \prod_{i=1}^{k-1} \tanh \left( \frac{\beta_d}{2} \right)} \tag{2.2}\]

with,

\[\frac{1 - a}{1 + a} = \tanh \left( \frac{\ln a}{2} \right) \Rightarrow \frac{1 + a}{1 - a} = \frac{1}{\tanh \left( \frac{\ln a}{2} \right)}\]

Rewriting Eq. 2.2,

\[\frac{1 + \prod_{i=1}^{k-1} (1 - 2P_{di})}{1 - \prod_{i=1}^{k-1} (1 - 2P_{di})} = \frac{1 + \prod_{i=1}^{k-1} \tanh \left( \frac{\beta_d}{2} \right)}{1 - \prod_{i=1}^{k-1} \tanh \left( \frac{\beta_d}{2} \right)} = \frac{1}{\tanh \left( \sum_{i=1}^{k-1} \ln \left[ \tanh \left( \frac{\beta_d}{2} \right) \right] / 2 \right)}\]

Finally,

\[\alpha_d \beta_d = \ln \left( \frac{P_r[x_d = 0 | y, S]}{P_r[x_d = 1 | y, S]} \right)\]

\[= \ln \left( \frac{1 - P_d}{P_d} \prod_{i=1}^{j} \left[ \frac{1 + \prod_{l=1}^{k-1} (1 - 2P_{di})}{1 - \prod_{l=1}^{k-1} (1 - 2P_{di})} \right] \right)\]

\[= \alpha_d \beta_d + \sum_{i=1}^{j} \left\{ \left( \prod_{l=1}^{k-1} \alpha_d \right) \mathcal{F} \left[ \sum_{i=1}^{k-1} \mathcal{F} (\beta_d) \right] \right\} \tag{2.3}\]

where,

\[\mathcal{F} (\beta) = \ln \left( \frac{e^{\beta} + 1}{e^{\beta} - 1} \right) = -\ln \left[ \tanh \left( \frac{\beta}{2} \right) \right] \tag{2.4}\]
2.1 History

The most significant feature of this decoding scheme is that the commutation per digit per iteration is independent on the block length \( n \). Furthermore, Gallager showed [6] that the average number of iterations required is bounded by a quantity proportional the log of the log of the block length.

2.1.2 Bipartite graph or Tanner graph

Gallager’s LDPC codes were not originally described in the language of graphical theory. R.M. Tanner, in 1981, [10] introduced a recursive approach to the construction of codes which generalized the LDPC code construction and suggested that the design of algorithms for encoding and decoding is amenable to the basic techniques of graph theory.

Indeed, graph provides a mean of visualization the constraints that define the code. More importantly, the graphs directly specify iterative decoding algorithms. Certain graph properties determine the decoding complexity, while other properties are connected to the performance of the iterative algorithms.

Tanner defined a new code formed from a graphical representation of the parity-check matrix, so called bipartite graph or Tanner graph, and one or more codes of shorter length, which he referred to as subcodes, as shown in Fig 2.2.

The central idea was to use the graph to structure the parity-check equations defining the code in a way that facilitates encoding and decoding.

![Figure 2.2. Code definition](image)

Figure 2.2. Code definition
A bipartite graph representing a code can be obtained from any system of parity check equations that define the code. Given the parity-check matrix of an arbitrary linear code, bipartite graph is constructed by identifying each row with a subcode node, or check node, each column with a digit node, or bit node, and creating an edge in the graph for every non zero matrix entry, as shown in Fig. 2.3 for the following H matrix,

$$H = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1
\end{bmatrix} \bigg\{ C_N \bigg\}$$

\[ V_{N-K} \]

Figure 2.3. Bipartite graph for matrix

Let the degree of every bit node be a constant $d_v$ and the degree of every check node be a constant $d_c$. By letting each of the check nodes represent a simple parity check equation, the graph defines a low-density parity-check code. Note that if as
2.2 LDPC codes viewed as Concatenated Codes

the name implies $d_c$ and $d_v$ are very small compared to the number of bit nodes (the length of the code), the graph corresponds to a sparse parity-check. When all of the check nodes are simple binary parity checks, there is obviously no need to assign position number to each of the bit nodes in the parity-check equation, as each of the check nodes is invariant under arbitrary permutation of its bits.

2.2 LDPC codes viewed as Concatenated Codes

Low-density parity-check codes are codes specified by matrix containing mostly 0’s and only a small number of 1’s. In particular, an $(n, j, k)$ low-density code is a code of block length $n$ with a matrix like that Fig. 2.4 where each column contains a small fixed number, $j$, of 1’s and each row has a small fixed number, $k$, of 1’s.

The analysis of a low-density code of long block length is difficult because of the large number of code words involved. It is simpler to statistically analyse a whole ensemble of such codes. From the ensemble behavior, one can make statistical statement about the properties of the member codes. By random selection from the ensemble, one can find a code with these properties with high probability of decoding error.

Gallager [6] constructed ensembles of regular LDPC codes. Regular LDPC codes are those for which all nodes of the same type have the same degree. For example, a $(20,3,4)$ has graphical bipartite representation in which all bit nodes have a degree 3 and all check nodes have a degree 4, has shown in the Fig. 2.4

Gallager described the parity-check matrix $H_{m \times n}$ of a code $C$ as concatenation of $c$ submatrices, each containing a single 1 in each column, as shown in Fig. 2.5. The first of these submatrix $H_1$ having size of $\frac{n}{k} \times n$ defines a super-code $C^1$. Note that $C^1$ satisfies a subset of the parity-check equation $C$, and hence $C$ is a subspace of $C^1$.

The other submatrices $H_2, \ldots, H_{d_c}$ are pseudo-random permutations of the columns of $H_1$, with equal probability assigned to each permutation. Thus, each $H_i$ defines a super-code $C^i$ on the corresponding subset of parity-check equations. Hence, $C$ is the intersection of the super-codes $C^1, \ldots, C^{d_c}$ as shown in Fig. 2.6.
20 Low-Density Parity-Check codes

| 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 |
| 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 |
| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 |

Figure 2.4. An example of Gallager’s \((n, j, k)\)regular parity-check matrix \(H\); \(n = 20\), \(j = 3\) and \(k = 4\)

Irregular LDPC code can similarly be defined by puncturing the super-codes. For such an irregular LDPC code, the degrees of each set of nodes are chosen according to some distribution. For example, an irregular LDPC code might have a graphical representation in half the bit nodes have degree 3 and half have a degree 5, while half the check nodes have a degree 4 and half have the degree 6.

Recent advances in the performance of Gallager codes [5] [4] are summarized in Fig 2.7.

2.3 Iterative Decoding scheme

The principal objective of defining the code in terms of explicit subcodes, such as LDPC codes, is to reduce the complexity of the decoding process to provide high quality codes that can be decoded effectively by computational process whose complexity grows only very slowly with increasing code length at fixed code rate.

For many parity-check codes the number of 0’s and 1’s in the parity-check matrix \(H\) are approximately the same. As mentioned previously, for LDPC codes the number of 1’s is very small compared to the number of 0’s: The parity-check matrix \(H\) has a low density of 1’s. Equivalently, the Tanner graph of the code has a low-density of edges. The complexity of the decoding algorithm of LDPC codes depends directly on this density so that a concerned designer of LDPC codes will try to keep this density as low as possible.
2.3 Iterative Decoding scheme

Figure 2.5. A parity check-matrix $H_{12 \times 16}$ defined as a concatenation of 3 submatrices of size $4 \times 16$. Puncturing $C^2$ and $C^3$ at the encircle 1’s results in an irregular code.

Figure 2.6. A code viewed as $C^1 \cap C^2 \cap C^3$

Assume we would like to transmit at a fraction $1 - \delta$ of capacity, where $\delta$ is some small positive constant. It is known [6] that the number of ones in the parity-check matrix $H$ has to scale at least like $n \ln(1/\delta)$ as a function of $\delta$, as $\delta$ approaches zero, where $n$ is the block length of the code. As we discuss in some more details in the next section, message-passing decoding algorithm works by performing several (simple) decoding rounds, and the required number of such rounds is guessed to grow like $1/\delta \ln(1/\delta)$.

Classical coding systems suffer from an exponential increase in complexity as a
function of $1/\delta$ as we are approaching channel capacity. Thus, iterative coding systems are such a promising choice for reliable transmission close to Shannon capacity.

The iterative decoding algorithm we will discuss directly on bipartite graph.

### 2.3.1 Message-passing algorithm

In message-passing algorithm, messages are exchanged along the edges of the graph, and computations are performed at the nodes, as shown in Fig. 2.8.

Each message represents an estimate of the bit associated with the edge carrying the message.

These decoders can be understood by focusing on one bit as follows:

Suppose the bits of an LDPC codeword are transmitted over a communications channel and, during transmission, some of them are corrupted so that a 1 becomes a 0 or vice versa. Each bit node in the decoder gets to see the bit that arrived at the receiver corresponding to the one that was transmitted from the equivalent node at the transmitter. Imagine that the node would like to know if that bit is in error or not and therefore asks all its neighboring check nodes what the bit’s value should be. Each neighboring check node then asks its other neighbors what their values are and sends back to the original bit node the modulo 2 sum of those values.
2.3 Iterative Decoding scheme

Figure 2.8. Illustration of message-passing algorithm on a bipartite graph

The bit node now has several opinions as to the bit’s correct value must somehow reconcile these opinions; it could, for example, take a majority vote. In order to improve performance, a further such iteration can be performed.

In actual decoding all nodes decode concurrently. Each node gather opinions from all its neighbors and forwards to each neighbor an opinion formed by combining the opinions of the other neighbors. This is the source of the term message-passing. The process continues until either a set of bits is found that satisfies all parity-check equations or time runs out. Note that with LDPC code, convergence to a codeword is easy to detect since one need only to verify that parity check equations are satisfied.

2.3.2 Soft decision decoding and Belief propagation algorithm

In hard decision decoding, opinions about a bit are expressed as binary value for that bit. Much better decoding is possible if opinions are expressed as probability, or LLR. If transmitted bits are received in error with the probability \( p \), the probability that observed bit is correct is \( 1 - p \). If a check node forms a modulo 2 sum of \( k \) of these bits, the probability that the sum is correct is \( (1 + (1 - 2p)^k)/2 \). Thus, the opinions returned from the check node have a different probability of being correct that those coming from the channel. If bit nodes properly take these probabilities into account when combining the opinions, better performance results. In the belief propagation algorithm the nodes assume that all incoming probability are independent and then combine them by applying the rules of probability.
Description of the message-passing algorithm will be facilitated by establishing a formal indexing for the registers required.

Let $Q_{i\rightarrow j}$ be the register associated with bit node $i$, with $i = 1, 2, \ldots, N$, that is accessed by check node processor $j$, with $j = 1, 2, \ldots, N$. $Q_{i\rightarrow j}(t)$ the value stored by the register after the $t^{th}$ iteration, and $R_{j\rightarrow i}$ a corresponding temporary storage register. Similarly let $\lambda_i$, with $i = 1, 2, \ldots, N$, be a register storing the value $\lambda_i(0)$ the LLR of the received digits. Finally let $J_i$ be the index set of the check node processors accessing bit node $i$, and let $I_i$ be the index set of bit nodes accessed by check node processor $j$.

**Figure 2.9. Initialize Bit-to-Check messages**

**Initialisation** Initialize all bit nodes and their outgoing edges to the value of the corresponding received bit $\lambda_i$, $i = 1, 2, \ldots, N$, represented as a log-likelihood ratio (LLR). Then, for $i = 1, 2, \ldots, N$ each register $Q_{i\rightarrow j}$, $j \in J_i$ is assigned with the value $R_{i\rightarrow j}(0) = \lambda_i$, see Fig 2.9.

**Iterative loop** For $t = 1, 2, \ldots, T$ the following two phases are performed:

**Figure 2.10. Check-to-Bit message-passing**
2.3 Iterative Decoding scheme

**Check-node update** For \( j = 1, 2, \ldots, N \) the \( j^{th} \) check node processor computes a parity-check (XOR) on the sign bits of the incoming edges, to form the parity-check equation result, see Fig 2.10. In addition, compute an intermediate check-node parity reliability function defined, in Eq. 2.4, as

\[
\hat{R}_{j \rightarrow i} = \left( \prod_{k \in J_i, k \neq i} \text{sgn}(Q_{k \rightarrow j}) \right) F \left| \sum_{k \in J_i, k \neq i} F | Q_{k \rightarrow j} | \right| \tag{2.5}
\]

For each \( i \in I_j \), and where \( F \) is defined as followed,

\[
F(\beta) = -\ln \left( \tanh \left( \frac{\beta}{2} \right) \right)
\]

Where \( \hat{R}_{j \rightarrow i} \) is the computed reliability of the message sent to the \( j^{th} \) check node processor from the \( i^{th} \) bit node.

![Figure 2.11. Bit-to-Check message passing](image)

**Bit-node update** The bit nodes update phase estimates the decoded bit using a summation of the log-likelihood of the received bit, \( \lambda_i \), and all of the check node message log-likelihoods, \( \hat{R}_{j \rightarrow i} \), see Fig 2.11. For each \( j = 1, 2, \ldots, N \), the register for the \( i^{th} \) bit node is updated according to

\[
Q_{i \rightarrow j} = \lambda_i + \left( \sum_{k \in I_j, k \neq j} \hat{R}_{k \rightarrow i} \right) \tag{2.6}
\]
The Hard decision can be considered as the check messages voting for the decoded bits value where the votes are weighted by their associated reliability, see Fig 2.12. Then the output value of the \(i^{th}\) bit is one if,

\[
d_{i^{th \; \text{decoded}}} = \begin{cases} 
1 & , \text{ if } \lambda_i + \sum_{k \in I_j} \hat{r}_{k \rightarrow i} > 0 \\
0 & , \text{ otherwise}
\end{cases}
\] (2.7)

The iteration loop is repeated until a termination condition is met. Possible iteration termination conditions include the following:

1. The estimated decode block \(\hat{s}\) satisfies \(\hat{s}H = 0\), where \(H\) is the parity-check matrix.

2. The current messages passed to the parity check nodes satisfy all of the parity-check equations. This does not guarantee that \(\hat{s}H = 0\) is satisfied, but is almost sufficient and simple to test.

3. Stop the decoding after a fixed number of iterations.

The message passing algorithm is optimal as long as the algorithm is propagating decision from uncorrelated sources [6].

Fig. 2.13 and Fig. 2.14 illustrate the full LDPC coding and decoding process.
Figure 2.13. Demonstration of encoding with a rate 1/2 Gallager code. The encoder is derived from very sparse $10000 \times 20000$ parity-check matrix $H$, with a bit-node degree of 3.(a) The code creates transmitted vectors consisting of $10000$ source bits and $10000$ parity-check bits.(b) Here, the source sequence has been altered by changing the $1^{st}$ bit. Notice that many of the parity-check bits are changed. Each parity bit depends on about half of the source bits.(c) The transmission for the case $s = (1, 0, 0, \ldots, 0)$. This vector is the difference (modulo 2) between (a) and (b). (reproduced from [5])
Figure 2.14. Iterative probabilistic decoding of a Gallager code. The sequence of figures shows the best guess, bit by bit, given by iteration decoder, after 0, 1, 2, 3, 10, 11, 12 and 13 iterations loop. The decoder halts after the 13th iteration when the best guess violated no parity check set. This final decoding is error free. (reproduced from [5])
Chapter 3

Implementation of LDPC decoding algorithm

In this chapter, design challenges for low-density parity-check (LDPC) decoders are discussed. We emphasize on the message-passing algorithm in terms of implementation complexity.

Contents

3.1 Requirements ........................................... 30
   3.1.1 Impact of code construction .................. 30
   3.1.2 Message-passing requirements ................. 30
3.2 LDPC decoder architecture ............................. 31
   3.2.1 Iterative decoding architectures ............... 32
3.3 Physical implementation .............................. 37
   3.3.1 Design platform ................................. 37
3.1 Requirements

3.1.1 Impact of code construction

The desire for large coding gains frequently conflicts with the requirements for low complexity and high flexibility of the decoder. In most classes of iterative decoders, the properties that dominate the architectural consideration are the size of code (or the block length) and the number of iterations. It is known that the BER performance of a code improves as the value of these numbers increases. However, considering that decoding scheme starts only after the final symbol in the block is received, block code with a large block length imposes heavy computational and memory requirements on the decoder. This also leads to extend the latencies, thus decreases the throughput. Likewise, a large number of iterations increases decoder latency and power while lower effective throughput.

A LDPC code design consists, in general, of two components;

- The choice of structure.
- The placement of the edges.

Structure means the broad description of the Tanner graph: The degree of the various nodes, restrictions on their interconnections, whether bit nodes are punctured, and so on. Gallager considered only regular graphs where all bit and check nodes have the same degree. Better performance can be realized by allowing irregular codes, containing nodes of various degree. A further improvement can be achieved by introducing more elaborate structures [8].

There is a typical trade-off that one encounters when designing LDPC codes. For a given structure, complexity and block length, one can either push the "waterfall portion" (see 2.7) of the error as far as possible toward the Shannon limit, or back off slightly and aim for very low error floors.

A second part of the design process, the placement of the edges requires some care to get the best possible performance. Moreover, hardware complexity can depend strongly on how this is done. In general, it is possible to produce a very-low-complexity LDPC decoder implementation that supports the best performing designs. But, producing those designs requires some expertise.

3.1.2 Message-passing requirements

While each node in the graph is associated with a certain arithmetic computation, each edge in the graph defines the origin of the destination of a particular message. An LDPC decoder is required to provide a network for messages to be passed between a large number of nodes. Direct wiring of network leads to congestion in the interconnect network due to the disorganized nature of the defining graph. Moreover, message updating do not need to be synchronized. Consequently, there is
3.2 LDPC decoder architecture

a great freedom to distribute in time and space the computation required to decode LDPC code.

The issue requiring most consideration is the implementation of the bandwidth required for message-passing between bit-nodes and check-nodes. The message bandwidth $W_{BANDWIDTH}$ measured in [bit/s] of an LDPC code with average column weight (or bit-node degree) $\Lambda_{avg}$ can be computed according to

$$W_{BANDWIDTH} = 2\Lambda_{avg} \cdot NbBitMESSAGE \cdot N_{iter} \cdot T$$

Where $NbBitMESSAGE$ is the number of bits used to represent each message, $N_{iter}$ is the number of decoder iterations, $T$ is the target coded throughput in [bit/s], and the factor of 2 is to count both bit and check messages. The realization of the message passing bandwidth results in very different and difficult challenges depending on whether a serial or parallel architecture is pursued.

3.2 LDPC decoder architecture

In his paper [6], Gallager sketched a simplified block diagram to show how message-passing algorithm can be done.

![Decoding Implementation](reproduced from [6])

He guessed from the Fig. 3.1 that a parallel computer can be simply instrumented requiring principally a number of proportional to $n$ analog adders, modulo 2 adders, amplifiers, and non-linear circuits to approximate the function $F(\beta)$. 
However, evaluating the sum in the log-probability domain requires a combination of exponential and logarithmic functions. In order to simplify the implementation, the computation can be approximated with the maximum value of the input operands, followed by an additive correction factor determined by a table lookup, as illustrated in the Fig. 3.2

![Check-node processor element (reproduced from [12])](image)

3.2.1 Iterative decoding architectures

In practice, the implementation of message-passing algorithm is constrained by the formats and throughput/latency requirements of specific communications standards. A practical implementation of a given algorithm in hardware is evaluated by its cost (silicon area), power, speed, latency, flexibility, and scalability.
3.2 LDPC decoder architecture

Parallel architecture

The message passing algorithm described in Section 2.3.1 is inherently parallel because there is no dependency between computation of either $Q_{i\rightarrow j}$ for $i = 1, 2, \ldots, N$ or $R_{j\rightarrow i}$ for $j = 1, 2, \ldots, N$.

Parallel decoder architectures directly map the nodes of a bipartite graph onto message computation units known as processing elements, and the edges of the graph onto a network of interconnect. Thus, such decoders benefit from a small switching activity, resulting in low power dissipation. Very little control logic is needed for the parallel architecture, because the LDPC code graph is directly instantiated by the interconnection of processing elements. Higher throughput with parallel decoder can be achieved by implementing a code with a large block size and maintaining the same clock frequency.

The main challenge when implementing a parallel decoder architecture for LDPC codes is the interconnection of processor elements at the top level. For an LDPC code to provide strong coding performance, the check node must necessarily connect to bit node distributed across a large fraction of the data block length. This results in a large number of long routes at the top level. In order to ease the difficulty in routing, a common "divide and conquer" approach is used to partition a design into smaller subsets with minimum overlap. However, in the case of irregular LDPC codes, due to irregularity in the parity check matrix, design partitioning...
is difficult and yields little advantage.

The major drawbacks with parallel decoder architecture are the **relatively large area** and the **inability to support multiple block size and code rates** on the same core. However, for application that requires high throughput and low power dissipation and can tolerate a fixed code format and large area, the parallel architecture is very suitable.

An example [3] of a parallel LDPC decoder for 1024-bit rate-1/2 code requires 1536 processing elements with an excess of 26000 interconnection wires to carry the messages between the processing elements.

### Serial based decoder architecture

An alternative approach is to serialize and distribute an inherently parallel algorithm among a small number of processing elements as shown in Fig. 3.4.

![Serial LDPC decoder architecture](image)

**Figure 3.4.** Serial LDPC decoder architecture

In order to capitalize on all hardware resources, it is more efficient to schedule all available processing elements to compute the messages in each round of decoding. Thus, the messages $Q_{i\rightarrow j}$ and $R_{j\rightarrow i}$ are stored temporarily in memory between their generation and consumption. By traversing multiple steps through the bipartite graph, it can be shown that the computation of messages in the decoding algorithm has data dependencies on messages corresponding to a large number of edges. This implies that the decoder will be required to have written most of the $R_{j\rightarrow i}$ computed messages into memory, before the computation of $Q_{i\rightarrow j}$ messages can proceed (and vice-versa). The size of the memory required is dependent on the total number of edges in the particular code design, which is the product of the average edge degree per bit node and number of bits in each block of LDPC code.

The advantages of serial based architecture are that they:

- minimize the area of the decoder.
- support multiple block size.
3.2 LDPC decoder architecture

- support multiple coderate.

However, the throughput is limited by the need for the functional units to be used and the edge memory accessed multiple times to perform each decoder iteration. Using multiple memories to achieve the requirement memory bandwidth is difficult because the essentially random or unstructured nature of the LDPC graph resists a memory architecture that allows both bit node and check node message to be addressed efficiently. Enforcing structure in the code graph to simplify the memory architecture typically introduces short cycles in the graph and reduces coding gain. **High memory bandwidth** requirements are also likely to translate into significant power dissipation. Another major issue with the serial based decoder architecture is the complexity of the control logic required for representation of the graph connectivity and corresponding address generation needed for fetching and storing the messages.

An example [12] of a serial LDPC decoder for 4608-bit rate-8/9 code with bit node average degree of 4, will have more the 18000 edges in the underlying graph. It would have to perform 37000 memory read or write operations for each iteration of decoding, which limits the total throughput.

**Partially-parallel decoder architecture**

Another approach [13], shown in Fig. 3.5, consists of an array of node computation units to perform all the node computation in time-division multiplexing mode and an array of memory blocks to store all the decoding messages.

The message-passing that reflects the bipartite graph connectivity is jointly realized by the memory address generation and the interconnection among memory blocks and node computation units. Suppose the base matrix is $H_{M \times N}$ and contains $L$ 1’s, and the expansion factor is $p$. The expanded matrix contains $L$ permuted identity matrices, each one denoted as $T_{U,V}$ as illustrated in Fig.3.6.

The LDPC code defined by such an expanded matrix exactly fits to the partially parallel decoder as shown in Fig. 3.5. This partially parallel decoder contains $M$ check node processor elements (CNUs), $N$ bit node processor elements (VNU), and $L+N$ memory blocks among which $L$ blocks store the iterative decoding messages, each one denoted as $DMEM_{U,V}$, and $N$ blocks store the channel messages, each one denoted as $CMEM_{V}$.

Each $DMEM_{U,V}$ connects with CNUu and VNUv, and stores $p$ decoding messages associated with the $p$ 1’s in the permuted matrix $T_{U,V}$. This decoder completes each decoding iteration in $2 \cdot p$ clock cycles. It works in check node processing mode during the 1st $p$ clock cycles, and bit node processing mode during the 2nd $p$ clock cycles. The operations in the two modes are as following:

**Check Node Processing**  
CNUs compute check-to-bit messages for all the check nodes in a time division multiplexing fashion. All the DMEs store the bit-to-check messages at the beginning. In each clock cycle, one bit-to-check message
in each DMEM is converted to the corresponding check-to-bit message by a read-computation-write process. The memory access address of each $DMEM_{U,V}$ is generated by a counter that starts from the block permutation value $k_{U,V}$.

**Bit Node Processing** VNU$s$ calculate extrinsic bit-to-check messages and update the decoding decision of all the bit nodes in a time division multiplexing
fashion. All the DMEMs store the check-to-bit messages at the beginning. Similarly, in each clock cycle, one check-to-bit message in each DMEM is converted to a bit-to-check message and the decoding decision of the corresponding is updated. The memory access addresses of all the DMEMs and CMEMs are generated by a counter that starts from 0.

Clearly, the number of node decoding units in this partially parallel decoder is reduced by the expansion factor $p$ compared with its fully parallel counterpart. This partially parallel decoder is well suited for efficient high speed hardware implementation because of the regular structure and simple control logic. Compared with the previous architectures, this design scheme supports much more flexible code rate configurations and degree distributions, hence has great potential on achieving very good error-correcting performance. However, the fully randomly constructed codes have little chance of fitting to efficient partially parallel decoder implementations.

### 3.3 Physical implementation

LDPC codes are applicable to wireless, wired and optical communications. The type of application dictates the particular class of platforms suitable for implementation of an LDPC decoder.

- Wireless applications are focused on low power implementation with rates at a few hundreds of kbps to several Mbps.
- Wireline access technologies such as VDSL have envisaged data rates up to 52 Mb/s downstream.
- Wireless LANs require data rates of the order of 100Mb/s.
- Storage applications require about 1Gbps.
- Optical communication throughputs can be above 10Gbps.

#### 3.3.1 Design platform

The choice of platform is dictated primarily by the performance constraints such as throughput, power, area, and latency, as well as flexibility and scalability.

- *Flexibility* of the platform represents the ease with which an implementation can be updated for changes in target specification.
- *Scalability* captures the ease of using the same platform for extensions of the application that may require higher throughputs, increased code block sizes, higher edge degrees for low-density parity-check codes.
Microprocessors and digital signal processors (DSPs) have a limited number of execution units but provide the most flexibility. These platforms naturally implement the serial architecture for LDPC decoding. Although an optimized program may decode at throughput rates of a few hundreds of kbps, practical use of microprocessors have to address operating system overhead. As a result, sustained decoding throughputs up to 100kbps are more realistic. Microprocessors and DSPs are used as tools for the majority of researchers in this field to design, simulate, and perform comparative analysis of LDPC codes. Performing simulations with bit error rates below $10^{-6}$, however, is a lengthy process.

Field programmable gate arrays (FPGAs) and custom ASICs are suitable for direct mapping of the message-passing algorithm, and offer more parallelism with reduced flexibility. Each computational logic block (CLB) in an example [12] Xilinx® Virtex-E FPGA can implement a 4-bit adder, or two 5-input XORs, or four 4-bit table lookups. The array of 104 × 156 CLBs in a XCV3200E is sufficient to execute the decoding logic of a fully parallel decoder for a 1024-bit, rate $1/2$, (3,6) regular LDPC code. The implementation of each bit-to-check (5 adders) and check-to-bit (eleven adders, six 5-input XORs, and twelve table lookups) processing element requires 5 CLBs and 17 CLBs respectively.

However, fully parallel LDPC decoding architectures will face mismatch between the routing requirements of the programmable interconnect fabric and bipartite graph. FPGAs are intended for datapath intensive designs, and thus have an interconnect grid optimized for local routing. The sparse nature of the LDPC graph, however, requires global and significantly longer routing. Existing implementations solve this problem by using time-shared hardware and memories in place of interconnect. This serial method limits the internal throughput to 56Mbps.

The decoding throughputs of several platforms implementing rate $1/2$ codes are compared in Fig. 3.7.

A direct-mapped custom ASIC implementation has been demonstrated on a rate $1/2$, 1024-bit parallel LDPC decoder [3] in 0.16μm technology. It dissipates 690mW at 1Gbps decoding throughput, and has an area of 7mm × 7mm.

An approach to avoid the routing congestion is through timesharing of hardware units; with hardware pipelining (through segmenting the check-to-bit and bit-to-check stages) to sustain the high throughput rates. Full utilization of all processing elements in the pipeline is only achievable if the Throughput (bps) computation of each class of messages is operating on an independent block of data. An LDPC decoder core that exemplifies this approach has become available as a commercial IP. It supports a maximum parallelism factor of 128, though details of the particular LDPC code have not been published.

Additional reduction of the memory requirement has been proposed through a staggered decoding schedule. This approach does not perform marginalization of the bit-to-check messages. By not computing the last term in, it has a memory requirement that is dependent only on the total number of bit nodes in the block.
Decoders with area or power constraints that limit the number of iterations to five or less will benefit from more than 75% reduction in memory requirement, while yielding to less than 0.5dB loss in BER performance. It is noted that the staggered decoding will not achieve the same asymptotic results as LDPC decoding under belief propagation.

**Figure 3.7.** Various platform vs. Realistic throughput of rates 1/2 decoders (Reproduced from [12])
Implementation of LDPC decoding algorithm
Chapter 4

Conclusions

4.1 Experimental details

As trainee engineer, I worked on a LDPC decoder testchip design project, as a part of a Central R&D team, at STMicroelectronics France. My primary focus on LDPC decoder design has been on the control-path. Control-path design is mainly constrained by the lack of any structural regularity of the DVB-S2 specified set of large sparse parity-check matrices. The need multiple code rate support and different block code length leads us to design a strongly iterative implementation of the message-passing algorithm. This tends to limit LDPC decoder implementation throughput. It also increases the number of control instruction patterns and their complexity.

We have used direct-mapping technique to map the message-passing algorithm on an ASIC. Several design iterations involving bottom-up evaluation have been carried out in order to reach an acceptable solution. Moreover, DVB-S2 specifications were not stable during the time of the design.

My thesis work has been partitioned on four phases:

1. Low-density Parity-check codes and message-passing algorithm understanding.
2. Scheduling of operations, considering the execution time that are assigned to the arithmetic and logic operation in the algorithm. Control instruction signals are derived from the schedule.
3. Design and Validation of the fully generic control-path.
4. ASIC oriented synthesis and gate-simulations validations.

The corresponding project planning GANTT chart is included to the appendix.
4.2 Results

Performances are very promising, however the project results must not be broadcasted outside STMicroelectronics.
Personally, I have been very pleased to participate in such innovative design. I definitely gained experience in design methodology for digital signal processing ASIC circuit. I hope that my study of Low-Density Parity-Check codes will allow me to be in phase with future High-Tech industries needs.

4.3 Discussion: Problem analysis

Low-Density Parity-Check (LDPC) codes have become a topic of great current interest because of their excellent error correcting performance and highly parallelizable decoding scheme. The past few years have experienced significant improvement on the design and analysis of the LDPC codes with near Shannon-limit performance. However, the realization of an LDPC decoder VLSI implementation still remains a big challenge and is a crucial issue determining how well we can exploit the unmatched merits of the LDPC codes in the real application.
Bibliography


Appendix A

GANTT chart
<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DVB-S2 specified LPDC decoder understanding</td>
<td>14 days</td>
<td>Mon 8/18/03</td>
<td>Thu 9/4/03</td>
</tr>
<tr>
<td>2</td>
<td>Data-path architecture understanding</td>
<td>9 days</td>
<td>Fri 9/5/03</td>
<td>Wed 9/17/03</td>
</tr>
<tr>
<td>3</td>
<td>Control-path architecture 1st release</td>
<td>38 days</td>
<td>Thu 9/18/03</td>
<td>Mon 11/10/03</td>
</tr>
<tr>
<td>4</td>
<td>Specifications_r1</td>
<td>18 days</td>
<td>Thu 9/18/03</td>
<td>Mon 10/13/03</td>
</tr>
<tr>
<td>5</td>
<td>RTL design_r1</td>
<td>9 days</td>
<td>Tue 10/14/03</td>
<td>Fri 10/24/03</td>
</tr>
<tr>
<td>6</td>
<td>Validation_r1</td>
<td>6 days</td>
<td>Mon 10/27/03</td>
<td>Mon 11/3/03</td>
</tr>
<tr>
<td>7</td>
<td>Documentation_r1</td>
<td>5 days</td>
<td>Tue 11/4/03</td>
<td>Mon 11/10/03</td>
</tr>
<tr>
<td>8</td>
<td>1st Milestone_functional RTL</td>
<td>1 day</td>
<td>Wed 11/12/03</td>
<td>Wed 11/12/03</td>
</tr>
</tbody>
</table>

**Project: MSc_thesis**

**Date: Thu 1/29/04**
<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Control-path architecture 2nd release</td>
<td>29 days</td>
<td>Wed 11/12/03</td>
<td>Mon 12/22/03</td>
</tr>
<tr>
<td>12</td>
<td>Specifications_r2</td>
<td>8 days</td>
<td>Wed 11/12/03</td>
<td>Fri 11/21/03</td>
</tr>
<tr>
<td>13</td>
<td>RTL design_r2</td>
<td>7 days</td>
<td>Mon 11/24/03</td>
<td>Tue 12/2/03</td>
</tr>
<tr>
<td>14</td>
<td>Validation_r2</td>
<td>11 days</td>
<td>Wed 12/3/03</td>
<td>Wed 12/17/03</td>
</tr>
<tr>
<td>15</td>
<td>Synthesis_r2</td>
<td>3 days</td>
<td>Thu 12/18/03</td>
<td>Mon 12/22/03</td>
</tr>
<tr>
<td>16</td>
<td>Documentation_r2</td>
<td>6 days</td>
<td>Mon 12/1/03</td>
<td>Wed 12/17/03</td>
</tr>
<tr>
<td>17</td>
<td>2st MilestoneRTL control-path frozen</td>
<td>1 day</td>
<td>Tue 12/23/03</td>
<td>Tue 12/23/03</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>LDPC decoder validation</td>
<td>20 days</td>
<td>Tue 12/23/03</td>
<td>Wed 1/28/04</td>
</tr>
<tr>
<td>20</td>
<td>LDPC decoder synthesis</td>
<td>7 days</td>
<td>Thu 1/29/04</td>
<td>Fri 2/6/04</td>
</tr>
<tr>
<td>21</td>
<td>3rd MilestoneRTL frozen</td>
<td>1 day</td>
<td>Mon 2/9/04</td>
<td>Mon 2/9/04</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>MSc Thesis introductory paper</td>
<td>5 days</td>
<td>Mon 11/24/03</td>
<td>Thu 12/4/03</td>
</tr>
<tr>
<td>24</td>
<td>MSc Thesis report</td>
<td>30 days</td>
<td>Wed 12/10/03</td>
<td>Thu 1/29/04</td>
</tr>
<tr>
<td>25</td>
<td>4th Milestone_report feed back</td>
<td>1 day</td>
<td>Fri 1/30/04</td>
<td>Fri 1/30/04</td>
</tr>
<tr>
<td>26</td>
<td>MSc slide show</td>
<td>14 days</td>
<td>Tue 2/3/04</td>
<td>Fri 2/20/04</td>
</tr>
<tr>
<td>27</td>
<td>5th Milestone_MSc Presentation</td>
<td>1 day</td>
<td>Mon 2/23/04</td>
<td>Mon 2/23/04</td>
</tr>
</tbody>
</table>

Project: MSc_thesis
Date: Thu 1/29/04
List of Figures

1.1 Schematic diagram of a general communication system. ............ 4
1.2 A channel translates source bits into coded bits to achieve error
detection, correction, or prevention. ................................. 5
1.3 Capacity channel as function of the bandwidth ...................... 6
1.4 Performance of various coding scheme (Convolutional Code, Turbo
Code, ...) ........................................................................ 7
1.5 In hard decision decoding of the channel code, a slicer makes the
decision about incoming symbols. The symbols are decoded into
bits by a line decoder, and the channel decoder maps these coded
bits into uncoded bits. ....................................................... 9
1.6 A soft decoder operates directly on samples of the incoming signal . 9

2.1 Parity check set tree ..................................................... 15
2.2 Code definition ............................................................ 17
2.3 Bipartite graph for matrix ............................................... 18
2.4 An example of Gallager’s \((n, j, k)\)regular parity-check matrix \(H\); \(n =
20, j = 3\) and \(k = 4\). ...................................................... 20
2.5 A parity check-matrix \(H_{12 \times 16}\) defined as a concatenation of 3 sub-
matrices of size \(4 \times 16\). Puncturing \(C^2\) and \(C^3\) at the encircle 1’s
results in an irregular code. ................................................. 21
2.6 A code viewed as \(C^1 \cap C^2 \cap C^3\). ......................................... 21
2.7 Empirical results for Gaussian channel rate 1/4 left-right: Irregular
LDPC block length 24000 bits; JPL turbo of block length 65535 bits;
regular LDPC block length 24448 bits; regular LDPC block length
40000 bits (Reproduced from [4]) .......................................... 22
2.8 Illustration of message-passing algorithm on a bipartite graph . . . . 23
2.9 Initialize Bit-to-Check messages ........................................ 24
2.10 Check-to-Bit message passing .......................................... 24
2.11 Bit-to-Check message passing ......................................... 25
2.12 Hard decision making step .............................................. 26
2.13 Demonstration of encoding with a rate 1/2 Gallager code. The en-
coder is derived from very sparse \(10000 \times 20000\) parity-c hack matrix
\(H\), with a bit-node degree of 3(a) The code creates transmitted vec-
tors consisting of 10000 source bits and 10000 parity-check bits.(b)
Here, the source sequence has been altered by changing the 1st bit.
Notice that many of the parity-check bits are changed. Each parity
bit depends on about half of the source bits.(c) The transmission for
the case \(s = (1, 0, 0, \ldots, 0)\). This vector is the difference (modulo 2)
between (a) and (b). (reproduced from [5]) .............................. 27
2.14 Iterative probabilistic decoding of a Gallager code. The sequence of figures shows the best guess, bit by bit, given by iteration decoder, after 0, 1, 2, 3, 10, 11, 12 and 13 iterations loop. The decoder halts after the 13th iteration when the best guess violated no parity check set. This final decoding is error free. (reproduced from [5])

3.1 Decoding implementation (reproduced from [6])
3.2 Check-node processor element (reproduced from [12])
3.3 Parallel LDPC decoder architecture
3.4 Serial LDPC decoder architecture
3.5 Partially parallel decoder structure
3.6 Matrix expansion
3.7 Various platform vs. Realistic throughput of rates 1/2 decoders (Reproduced from [12])
Index

belief propagation, 23
bipartite graph, 17
bit node, 17, 22
block codes, 9, 11

code rate, 5, 9
codeword, 10
coding, 5
coding gain, 6
degree, 17
decoding, 8
decomposition, 37
dependence, 8

effective bandwidth, 7
edge, 17
forward error correction, 8

forwarding, 37
Gallographic, 8

generators, 10
generator matrix, 10
generic, 37

graph, 17
Hamming distance, 11
Hamming weight, 11
hard decision, 8

irregular codes, 20
iterative decoding, 14

Low-density parity-checking, 14

optimal decoder, 10
parity-check matrix, 11

regular codes, 18
Scalability, 37

signal-tonoise ratio, 6
soft decision decoding, 8

Source coding, 5

spectral efficiency, 7

subcodes, 17
syndrome vector, 11
systematic code, 10
På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Uphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Florent PIROU