DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

**Ecole Nationale Polytechnique**

**Electronic Departement**
**Laboratoire des Dispositifs de Communication et de Conversion Photovoltaique**

*Submitted in partial fulfillment of the requirements for the Master Degree*

# Design of Full-Parallel Non Binary LDPC Decoder

BALI Cherif

Supervised by : Mr. M.TAGHI

presented on : 2/07/2017

**Jury members :**

| | | | |
|---|---|---|---|
| President | Mr. S. AIT CHEIKH | Professor | ENP |
| Examiner | Mr. D. BERKANI | Professor | ENP |
| Supervisor | Mr. M.TAGHI | Assistant Professor | ENP |

**ENP 2017**

*Ecole Nationale Polytechnique 10, Avenue HASSEN Badi, 16200 El Harrach, Alger.*

DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

**Ecole Nationale Polytechnique**

**Electronic Departement**
**Laboratoire des Dispositifs de Communication et de Conversion Photovoltaique**

*Submitted in partial fulfillment of the requirements for the Master Degree*

# Design of Full-Parallel Non Binary LDPC Decoder

BALI Cherif

Supervised by : Mr. M.TAGHI

presented on : 2/07/2017

**Jury members :**

| | | | |
|---|---|---|---|
| President | Mr. S. AIT CHEIKH | Professor | ENP |
| Examiner | Mr. D. BERKANI | Professor | ENP |
| Supervisor | Mr. M.TAGHI | Assistant Professor | ENP |

**ENP 2017**

*Ecole Nationale Polytechnique 10, Avenue HASSEN Badi, 16200 El Harrach, Alger.*

## ملخص

شفرات اختبار التكافؤ منخفضة الكثافة قد تم إدراجها بنجاح في العديد من أنظمة الاتصالات اللاسلكية لأنها تحقق أداء تصحيح الخطأ قريبة جدا من حد شانون .شفرات اختبار التكافؤ منخفضة الكثافة غير الثنائية لديها أداء أفضل من الشفرات الثنائية، في هذه الأطروحة ركزنا على تصميم وإنشاء بنية فعالة للأجزاء الأساسية من مفك شفرات اختبار التكافؤ منخفضة الكثافة غير الثنائية باستخدام خوارزمية الحد الأدنى-الحد الأكبر . ثم توفير مفك شفرات قوي اعتمادا على بنية كلية التوازي.

**الكلمات المفتاحية**: شفرات اختبار التكافؤ منخفضة الكثافة، شفرات اختبار التكافؤ منخفضة الكثافة غير الثنائية ، حد شانون، تصحيح الخطأ، خوارزمية مفكك الشفرات، بنية كلية التوازي، تصميم، إنشاء.

## Résumé

Les codes de contrôle de parité de faible densité (LDPC) ont été inclus avec succès dans de nombreuses standard de communication sans fil, car ils atteignent des performances de correction d'erreur très proches de la limite de Shannon. Les codes LDPC non binaires ont de meilleures performances que les codes LDPC binaires. Dans cette thèse, nous sommes concentrés sur la conception et l'implémentation d'une architecture full-parallel du décodeur NB-LDPC à l'aide de l'algorithme Min-Max. La conception et l'implémentation des composants du décodeur sont détaillées.
**Mots clés :** LDPC, NB-LDPC, Limite de Shannon, Correction d'erreur, Min-Max, Décodeur, Architecture full-parallel, Conception, Implémentation.

## Abstract

Low Density Parity-Check (LDPC) codes have been successfully included in numerous wireless communication standards, since they achieve error correction performance very close to the Shannon limit. Non-Binary LDPC codes has better performance than the binary LDPC codes, In this thesis, we focused on the design and implementation of efficient architecture of the NB-LDPC decoder basic blocks using the Min-Max algorithm. In order to provide flexible decoder. then proposing a full-parallel design for a hight thoughput communications,the design and implementation of the decoder components are detailed.
**Keywords :** LDPC, NB-LDPC, Shannon limit, error correction, Min-Max, Decoder, Full-parallel Architecture, Design, Implementation.

# Dedication

I dedicate this work to my family, my dear parents for all their sacrifices for my education, to all my friends, and all people who taught me along my career.

# Acknowledgments

We would like to express our gratitude towards our supervisor, Mr. M.TAGHI. for his valuable advices, encouragement and guidance throughout the course of this project.

We would also like to thank Mr. S.AIT CHEIKH and Mr. D.BERKANI for their support as members of jury.

# Contents

# List of Figures

# List of Tables

# General Introduction

The reliable transmission of information over noisy channels is one of the basic requirements of wireless communication systems. Since these systems demand for high-speed information exchange between transmitter and receiver nodes, the channel impairments become more harmful, which reduce the reliability of the received information. To overcome this situation and provide more reliable communications, efficient channel coding techniques are required. Due to this requirement, these systems rely heavily on error correction codes to detect and correct transmission errors.

Non-Binary LDPC (NB-LDPC) codes are an extensions of binary LDPC codes. These codes perform better than the binary LDPC codes in case of codes with low and medium codeword length. Despite the error-correcting performance advantages, NB-LDPC codes suffer from high decoding complexity. During the last decade, significant progress has been made in the development of low-complexity NB- LDPC decoding algorithms and the implementation of these algorithms in flexible dedicated very-large-scale integration (VLSI) circuits. The graphical representation of the NB-LDPC codes can be used in the implementation of these algorithms whose effectiveness has been shown on graph models such as the Belief Propagation algorithm generally noted BP. This algorithm guarantee optimal decoding performances but it has not great interest for a hardware implementation. Consequently, other algorithms based on approximations of the BP algorithm have been developed with the aim of ensuring a reasonable performance/complexity compromise, like the Min-Max algorithm which can be implemented by a more efficient architecture then the others with small performance degradation.

The objective of our project is to design a NB-LDPC decoder based on Min-Max algorithm and a full-parallel architecture. In particular, we provide concepts and solutions that enable flexible implementation and a compromise between decoding speed and implementation complexity, which are the basic requirements of modern communication standards.

# Chapter 1

# LDPC Code

Low-density parity-check (LDPC) code is a channel code and one of the robust linear block codes. The name comes from the characteristic of their parity-check matrix which contains only a few non-zero elements in comparison to the amount of zeros, They were first proposed in the 1962 PhD thesis of Gallager at MIT. But they remained largely neglected for over 35 years, because of the computational power to exploit iterative decoding schemes was not available until recently.

Today, design techniques for LDPC codes exist which enable the construction of codes which approach the Shannon's capacity limit [1].

This chapter give a brief presentation of LDPC codes, including the Fundamentals and mathematical basics. we will introduce the deferent types of representation for these codes, their proprieties and classes. We will also present the decoding operation with some basic decoding algorithms .

## 1.1 Parity check code

Parity checking is the most basic form of error detection in communications. The simplest coding scheme is the single parity-check code. This code involves the addition of a single extra bit, called a parity-check bit, to the binary message, the value of this bit depends on the bits in the message. In an even-parity code the additional bit added to each message ensures an even number of 1s in every codeword.

Example (1) : Denote a code $\mathbf{C}$ consists of codeword of length $n = 6$, and the vectors $c = [c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ ]$ , where each $c_i$ is either 0 or 1 and every codeword satisfies the constraint:

$$c_1 \oplus c_2 \oplus c_3 \oplus c_4 \oplus c_5 \oplus c_6 = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 0 \tag{1.1}$$

Equation (1.1) is called a parity-check equation. which is an equation linking n binary data to each other by the exclusive or, denoted $\oplus$ operator. It is satisfied if the total number of 1s in the equation is even or null.

While the inversion of a single bit due to channel noise can easily be detected with a single-parity check code, this code is not sufficiently powerful to indicate which bit, or perhaps bits, were inverted. Moreover, since any even number of bit inversions produces a vector satisfying the constraint (1.1), patterns of even numbers of errors go undetected by this simple code.

Detecting more than a single bit error calls for increased redundancy in the form of additional parity bits. These more sophisticated codes contain multiple parity-check equations, every one of which must be satisfied by every codeword in the code.

Example (2): Denote the vector $c = [c_1\ c_2\ c_3\ c_4\ c_5\ c_6]$ that satisfy the three parity-check equations:

$$\begin{aligned} &c_1 \oplus c_2 \oplus c_4 = 0 \\ &c_1 \oplus c_2 \oplus c_3 \oplus c_6 = 0 \\ &c_2 \oplus c_3 \oplus c_5 = 0 \end{aligned} \tag{1.2}$$

Checking the vector $\hat{c} = [1\ 1\ 0\ 0\ 0\ 0]$ we see that :

$$\begin{aligned} &1 \oplus 1 \oplus 0 = 0 \\ &1 \oplus 1 \oplus 0 \oplus 0 = 0 \\ &1 \oplus 0 \oplus 0 = 1\ \times \end{aligned} \tag{1.3}$$

so $\hat{c}$ is not a valid codeword for this code because the parity-check equations did not satisfy . The Codeword constraints are making an equations system in order to simplify the working in these constraints are often written in matrix form, and so the constraints (2.2) become :

$$\underbrace{\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}}_{H} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{1.4}$$

The matrix $H$ is called a parity-check matrix. Each row of H corresponds to a parity-check equation and each column of $H$ corresponds to a bit in the codeword. The $(j, i)th$ entry of $H$ is 1 if the $ith$ codeword bit is included in the $jth$ parity-check equation. Thus for a binary code with $m$ parity-check constraints and length-n codewords the parity-check matrix is an $m \times n$ binary matrix.

In matrix form a vector $\hat{c} = [c_1\ c_2\ c_3\ c_4\ c_5\ c_6]$ is a valid codeword for the code with parity-check matrix $H$ if and only if it satisfies the matrix equation :

$$H\hat{c}^T = 0_m \tag{1.5}$$

More than one parity-check matrix can describe a particular code; two parity check matrices for the same code do not even have to have the same number of rows, but they must satisfy (2.3) for every codeword in the code.

## 1.2 Representation of LDPC Codes

There are two ways to represent the LDPC code. As a linear block code, it can be described via matrices. The second way is via a graphical description.

### 1.2.1 Matrix Representation

**The generator matrix G:** This matrix describes the mapping from source words $x$ to codewords $c$ in the encoding part by the equation $c = G^T x$ .

It is common to consider $G$ in systematic form $G = [I_k|P]$ so that the first $k$ transmitted symbols are the source symbols.The notation $[A|B]$ indicates the concatenation of matrix $A$ with matrix $B$; $I_k$ represents the $k \times k$ identity matrix. The remaining symbols are the parity-checks.

$$\begin{aligned} G = [I_3|P] &= \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \\ H = [P^T|I_3] &= \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{1.6}$$

**Parity Check Matrix :** The LDPC code is also described by a parity check matrix $H$ of dimension $m \times n$ , This matrix can be seen as a linear system of m parity check equations. The words $c$ of the code defined by $H$ simultaneously satisfy the $m$ parity check equations. If the corresponding generator matrix is written in systematic form as above, then $H$ has the form $[-P^T|I_m]$. Note that for codes over finite fields $GF(2^p)$.

### 1.2.2 Graphical Representation

LDPC codes are usually defined in terms of a sparse bipartite graph, the so-called Tanner graph (Tanner, 1981)[2], in which we can represent the parity check matrix, in this graph branches link two different classes of nodes to each other:

- The first class of nodes called variable nodes (bit nodes), correspond to the bits of the codewords $(v_j, j = 1, ..., n)$, and therefore to the columns of $H$.
- The second class of nodes, called parity check nodes, correspond to the parity check equations $(c_i, i = 1, ..., m)$, and therefore to the rows of $H$.

$$\mathbf{H} = \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \left[\begin{array}{ccccccc} 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right] \begin{array}{c} c_1 \\ c_2 \\ c_3 \\ c_4 \end{array} \end{array}$$

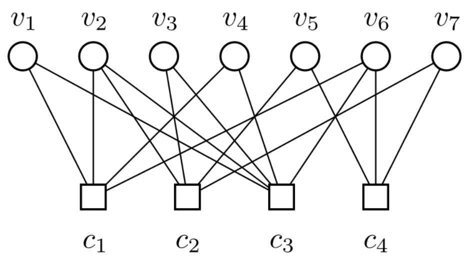

Figure 1.1: An example of H matrix and the corresponding Tanner graph

Thus, to each branch linking a variable node $v_j$ to a parity check node $c_i$ corresponds the 1 that is situated at the intersection of the $j-th$ column and the $i-th$ row of the parity check matrix.

## 1.3 Classifications of LDPC codes

There are two type of classification for the LDPC codes the first one depends to the regularity of the code which divided into two groups, the regular code and the irregular codes . The second classification based on the representation of data over the Galois Field which we will discuss it in the next chapter, there are also two groups of codes in this classification the Binary codes and the Non-Binary codes .

### 1.3.1 Regular and Irregular LDPC Codes

**Regular Codes :** Regular codes were the first to be introduced when R. Gallager introduced the LDPC codes in 1962 . The regularity of these codes is specified by the constant number of "1" in the rows and columns of the matrix $H$ It's mean that $w_r$ and $w_c$ are constant and connected by the following relation:

$$w_r = w_c.n/m \tag{1.7}$$

The regular LDPC codes are then described by $(n, w_r, w_c)$ which $n$ representing the length of the code word, wr the weight of the lines and wc the weight of the columns. It is clear that $w_r$ (respectively $w_c$) are very small numbers in comparison of $n$ (respectively $m$ ) so that $H$ is sparse ( low density ).

As the ratio $R$ have a relation with $n$ and $m$ we can rewrite it with other way in function of $w_r$ and $w_c$ .

$$R = 1 - w_c/w_r \tag{1.8}$$

**Irregular codes :** Which on case the distribution of the non-zero elements is not uniform. The study shows that the irregular LDPC codes have better performance than the regular codes . In the other hand the irregular codes have more implementation complexity than the regular codes .

### 1.3.2 Binary and Non Binary LDPC Codes

**The binary LDPC code** is described by a binary-valued $m \times n$ parity-check matrix $H$ in $GF(2)$, where the $GF$ is abbreviation of Galois Field. This code may be generalized to finite fields $GF(q)$ , where $q$ is a prime number, The elements of $GF(q)$ will be called symbols and we use the term bits when referring to the binary representation of symbols (when $q = 2$). For a code over $GF(q)$, each received symbol can be any of the $q$ elements in $GF(q)$ .

**The Non binary LDPC codes** over $GF(q)$ can be seen as the generalization of binary LDPC codes over $GF(2)$ vector space projected over a finite field $GF(q)$ , where $q = 2^m$ $(m \in Z^+)$. In this case, each symbol can be represented by a $m - bit$ binary tuple[3].

In the parity check matrix $H$, of a NB-LDPC code over $GF(q)$, the nonzero entries are elements of $GF(q)$. Also each information and codeword symbol is an element of $GF(q)$.

An example of galois field is the GF(8) represented in the Table 2.1. The basic operaions over the Galois Field are the addition and the multiplication, these operations are different from the usual addition and multiplication operaions, and it depends on the chosen GF representation, in what follow we will be using the power representation because of its low complexity hardware implementation.

| power | binary | integer |
|---|---|---|
| 0 | 000 | 0 |
| 1 | 100 | 1 |
| $\alpha$ | 010 | 2 |
| $\alpha^2$ | 001 | 4 |
| $\alpha^3$ | 110 | 3 |
| $\alpha^4$ | 011 | 6 |
| $\alpha^5$ | 111 | 7 |
| $\alpha^6$ | 101 | 5 |

Table 1.1: $GF(8)$ power representation

The addition and the multiplication operations over $GF(2^m)$ are illustrated on the following equations:

$$x + y = x \ XOR \ y \tag{1.9}$$

$$\alpha^i.\alpha^j = \alpha^{(i+j)mod(2^m-1)} \tag{1.10}$$

$$H = \begin{pmatrix} h_{0,0} & h_{0,1} & h_{0,2} & 0 & 0 & 0 \\ 0 & h_{1,1} & 0 & h_{1,3} & h_{1,4} & 0 \\ h_{2,0} & 0 & 0 & h_{2,3} & 0 & h_{2,5} \\ 0 & 0 & h_{3,2} & 0 & h_{3,4} & h_{3,5} \end{pmatrix}$$

$$h_{0,0} \cdot c_0 + h_{0,1} \cdot c_1 + h_{0,2} \cdot c_2 = 0 \text{ ——— } 1$$
$$h_{1,1} \cdot c_1 + h_{1,3} \cdot c_3 + h_{1,4} \cdot c_4 = 0 \text{ ——— } 2$$
$$h_{2,0} \cdot c_0 + h_{2,3} \cdot c_3 + h_{2,5} \cdot c_5 = 0 \text{ ——— } 3$$
$$h_{3,2} \cdot c_2 + h_{3,4} \cdot c_4 + h_{3,5} \cdot c_5 = 0 \text{ ——— } 4$$

Figure 1.2: Non-binary parity-check matrix

The matrix products of the parity equations are performed using the addition and multiplication operations of the Galois Field $GF(2^m)$. It is then preferable to add to the bipartite graph the new family of nodes called the permutation nodes which serve to model the multiplication of the symbols of the code word by the non-zero elements of the parity matrix $h_{ij}$ . Figure 1.3 illustrates the bipartite graph for equation 2 in Figure 1.2 by adding the permutation nodes that correspond to the elements $h_{11}$,$h_{13}$ and $h_{14}$ .

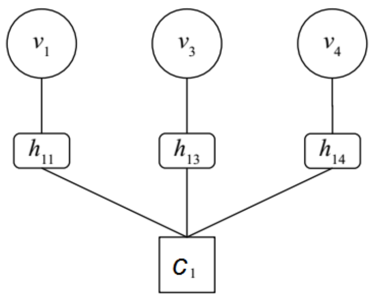

Figure 1.3: Graphical representation of non-binary parity-equation

# Chapter 2

# LDPC Decoding

Actually there is more than one such decoding algorithm. There exists a class of algorithms that are all *iterative* procedures where, at each round of the algorithm, messages are passed from *variable nodes* to *check nodes*, and from check nodes back to variable nodes. Therefore, these algorithms are called *message passing decoding algorithms* [4].

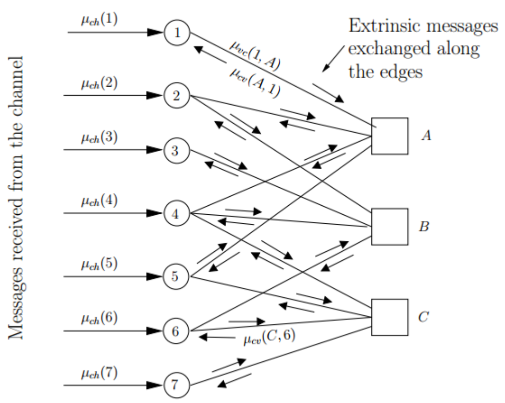

Figure 2.1: One complete iteration message passing example

The basic steps of the message passing algorithm are :

1. **Initialization:**

   The incoming messages received from the channel at the variable nodes are directly passed along the edges to the neighbouring check nodes because there are no incoming messages (extrinsic) from the check nodes in the first iteration.

2. **Updating the check nodes (CNs):**

   The check nodes perform local decoding operations to compute outgoing messages (extrinsic) depending on the incoming messages received from the neighbouring variable nodes. Thereafter, these new outgoing messages are sent back along the edges to the neighbouring variable nodes.

3. **Updating the variable node (VNs):**

The variable nodes will perform the local decoding operations in the same way to compute the outgoing messages from the incoming messages received from both the channel and the neighbouring check nodes.

4. **Tentative Decoding :**

   After a complete iteration ( updating of the CNs and VNs ) the last operation is the calculation of hard decision messages and checking the codeword validity by using the syndrome .

   In this way, the iterations will continue to update the extrinsic messages unless the valid codeword is found or some stopping criterion is fulfilled ( achieving the limit number of iterations )

   One important message-passing algorithm is the belief propagation algorithm which was presented by Robert Gallager in his PhD thesis. This algorithm has developed several times under different names. The most common ones are *sum-product algorithm* (SPA)[5], *min-sum*[6], extended *min-sum* (EMS) algorithms and the *min-max algorithm*[7], these algorithms will be presented in the next chapters .

## 2.1 The Min-Max Algorithm

The logarithmic likelihood ratio (LLR) can take negative values. However, it would be simpler to deal only with positive values. Therefore, there was a proposition to define the LLRs as follows:

$$LLR(\beta) = -\ln \frac{p(c=\beta|r)}{\max\limits_{\theta \in GF(2^m)} \{p(c=\theta|r)\}} \quad \beta \in GF(2^m) \tag{2.1}$$

Where $r = (r_0, r_1, ...r_{M-1})$ is the observation of the channel and $c = (c_0, c_1, ...c_{M-1})$ is the transmitted symbol. In this definition, the normalization is done by the probability of the most reliable symbol. It follows that the LLR of this symbol is always zero and the LLRs of the other symbols are positive.

| GF | 0 | $\alpha^0$ | $\alpha^1$ | $\alpha^2$ | $\alpha^3$ | $\alpha^4$ | $\alpha^5$ | $\alpha^6$ |
|---|---|---|---|---|---|---|---|---|
| $P_s$ | 0.1 | 0.85 | $10^{-3}$ | $10^{-7}$ | $10^{-10}$ | 0.05 | $10^{-10}$ | $10^{-10}$ |
| -ln($P_s$) | 2.3 | 0.2 | 6.9 | 16.1 | 23.0 | 3.0 | 23.0 | 23.0 |
| $LLR_s$ | 2.1 | 0 | 6.7 | 15.9 | 22.8 | 2.8 | 22.8 | 22.8 |

Table 2.1: Exapmle of LLRs values of GF(8)

Min-Max algorithm is an approximation of the MS algorithm, a modifications have been done in which makes it possible to simplify the processing at the CNs by replacing the sum in equation 4.13 by the operator max [8].

We can resume the steps of the Min-Max algorithm by :

- Initialization: $M_{v_j c_i}[\beta] = I_j[\beta]$
  Iterations :

- Check node processing

$$M_{c_i v_j}[\beta] \approx \min_{\sum_{\substack{s\neq j \\ h_{is}\neq 0}} \theta_s=\beta} \{\max_{\substack{s\neq j \\ h_{is}\neq 0}} \tilde{M}_{v_s c_i}[\theta_s]\} \tag{2.2}$$

- Variable node processing

$$M'_{v_j c_i}[\beta] = I_j[\beta] + \sum_{\substack{s\neq j \\ h_{is}\neq 0}} \tilde{M}_{c_s v_j}[\beta] \quad \beta \in GF(q) \tag{2.3}$$

$$M_{v_j c_i}[\beta] = M'_{v_j c_i}[\beta] - \min_{\beta\in GF(q)} (M'_{v_j c_i}[\beta]) \tag{2.4}$$

- A posteriori information computation

$$APP_j[\beta] = I_j[\beta] + \sum_{h_{s,j}\neq 0} \tilde{M}_{c_s v_j}(\beta) \quad \beta \in GF(q) \tag{2.5}$$

After each iteration, hard decision for the ith symbol can be made as:

$$\hat{c}_j = \underset{\beta\in GF(q)}{argmax}\{APP_j[\beta]\} \quad j = 0, 1, ...N-1$$

The iterations can be carried out until $H[\hat{c}_0, \hat{c}_1, \hat{c}_2, ...]^T = 0$ or the maximum iteration number has been reached.

To simplify The CN operations it can be implemented efficiently by forward-backward scheme.This scheme consists in constructing the outgoing messages by a set of elementary operations allowing not to repeat the same calculations and to reduce the latency of processing.

# Chapter 3

# NB-LDPC Decoder Architecture

In this chapter, we will present a design of the full-parallel decoder on hardware based on the Min-max algorithm for NB-LDPC codes. We will start from the decoder system level view of the basic blocks, then we will discuss the layred and non-layred decoder architecture and compare between them.

## 3.1 System level architecture

### 3.1.1 Check Node architecture

Computing the *c-to-v* messages in a straight-forward manner requires a complexity of $O(q^{w_r-1})$, where $w_r$ is the number of variable nodes connected to a check node. This manner requires complicated computations on $GF(q)$ elements, and it is not suitable for efficient hardware implementation.

Alternatively, the forward-backward scheme can be applied to the check node processing to avoid computing output message directly. This scheme consists in constructing the outgoing messages by a set of elementary operations, making it possible not to repeat the same computations and to reduce the processing latency. These elementary operations are performed by Elementary Check Node (ECN). Each ECN receives two sorted messages $M_1$ and $M_2$ and generates a sorted message $M_o$. The message $M_o$ is constructed by selecting the $q$ most reliable symbols from all the possible combinations ( as explained in min-max CN updating operation ).

Figure 3.1 illustrates the Forward-Backward architecture of CN with degree $w_r = 4$. For clarity, the incoming CN messages are denoted by $M_{v_jc}$ and the outgoing messages are denoted by $M_{cv_j}$, $j = 1, ..., 4$.

#### Elementary Check Node architecture

The EVN compute an outgoing message vector from two incoming message vectors. The message vector consists of two parts: LLRs and corresponding $GF(q)$ elements. Denote the LLR vectors by $L_A = [L_A(0), L_A(1), ..., L_A(q-1)]$ and $L_B = [L_B(0), L_B(1), ..., L_B(q-1)]$, and the corresponding finite field element vectors by $GF_A = [GF_A(0), GF_A(1), ..., GF_A(q-1)]$ and $GF_B = [GF_B(0), GF_B(1), ..., GF_B(q-1)]$, also denote the output LLR and corresponding finite field element vectors by $L_O$ and $GF_O$. The entries in the output LLR vector for the Min-max decoding are the $q$ minimum values of $\max(L_A(i), L_B(j))$ with different $GF_A(i)+GF_B(j)$

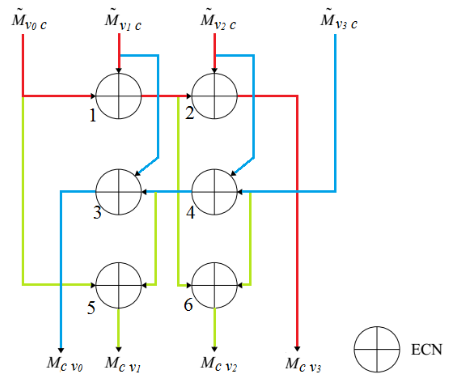

Figure 3.1: Forward-Backward Check-node architecture

for any combination of $i$ and $j$ less than $q$.

The traditional solution of computing the outgoing message consists on comparing $q^2$ pairs of messages from the two input vectors to find the ones with larger LLRs, then find the $q$ minimums among them. Taking care of all these operations in parallel is hardware-demanding, also performing them serially make a latency problem. But the ECN developed in our project uses two incoming messages stored in the order of increasing LLR and generate sorted outgoing message of the most reliable messages, by using an efficient algorithm to find the $q$ most reliable LLRs in minimum clock cycles using serial computation to reduce the hardware area.

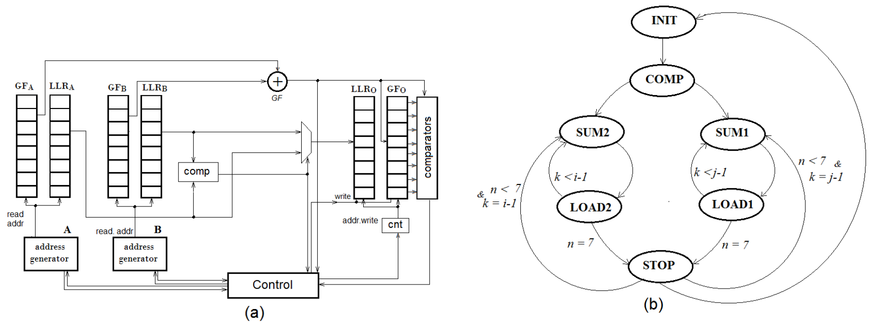

Figure 3.2: ECN architecture and its control FSM graph

### 3.1.2 Variable Node architecture

The VNs are divided into two categories according to whether all $q$ messages are kept for each vector or not. In our project we will design a VN with degree $w_c = 2$, that means two extrinsic messages and one intrinsic message are the incoming messages of the VN. In our architecture all $q$ messages are kept in each vector.

The architecture of a VN is given in Figure 3.3. It contains several Elementary Blocks that can make it capable to operates on three basic functions :

- Updating the VN: This operation is done by the EVN block, The EVN receives two incoming message from the CNs and generate an new outgoing message to the normalization block .

- Normalization : This operation is done by the Norm block, It subtracts the smallest LLR in the input vector from each LLR in the vector so that the smallest LLR in each vector is brought back to zero.
- Decision: This operation is done by the Decision block. Firstly it calculates the APP by the sum of all VN input message vectors, then determine the estimated symbol $\hat{c}$ by taking the GF element corresponding to the minimum LLR value.

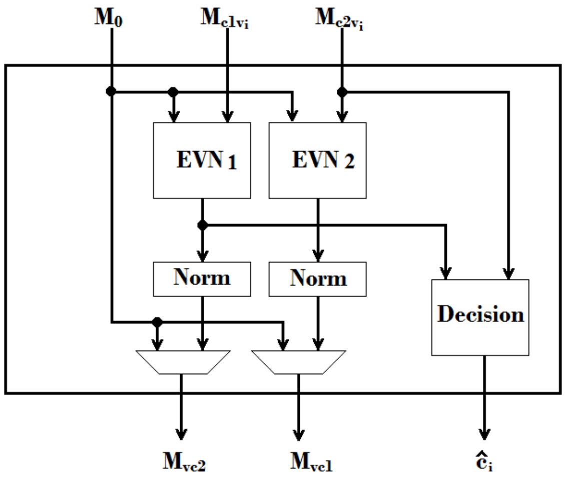

Figure 3.3: Variable Node Architecture

## Elementary Variable Node architecture

The goal of an EVN is to compute the outgoing message vector stored in the order of increasing LLR by adding the LLRs of the two incoming message vectors corresponding to the same $GF(q)$ element. Figure 3.4 illustrate the EVN architecture, two RAMs are used to storage the incoming message vectors, adder to add the LLR values, parallel GF elements comparators to help in finding the GF element address, sorter for the output vector and control unit to generate the control signals.

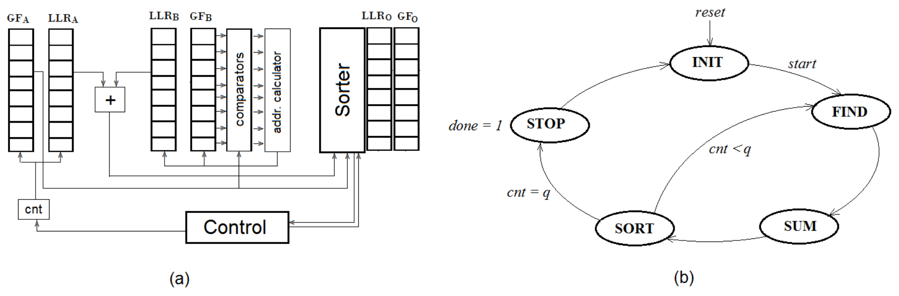

Figure 3.4: Elementary Variable Node Architecture

At first, one entry of the A vector is read out. Since the vectors are not sorted by $GF$ element, the $GF$ element of the $GF_A$ vector entry is compared with all those in the $GF_B$ vector. If there is a match, the addr calculator can give the corresponding $GF$ element address, then using the addresses of A and B vectors to read theirs corresponding LLRs and perform the addition, the LLR addition result and its corresponding $GF$ element are the entries of the output vector. The adder's output is connected to a sorter in order to sort and store the output elements. Using a counter to read out the next A vector entry and repeat these steps for all the $q$ entries. All previous operations are controlled by the control unit.

## 3.2 High level architecture

### 3.2.1 Global Architecture Description

From a high-level perspective, virtually all implementations of message passing LDPC decoders found in the open literature are derived from an isomorphic architecture [9] which is a direct mapping of the tanner graph.

The global architecture of decoder as illustrated in Figure 3.5 consists of different types of hardware components:

- VN unit (VNU) block and CN unit (CNU) block to compute the update equations.
- Interconnect network representing the edges of the graph and the $h_{ij}/h_{ij}^{-1}$ multiplication block.
- Storage devices in order to save the extrinsic and the intrinsic messages.
- LLRs calculator block to calculate the $GF(q)$'s LLR values and the Syndrome block to check the codeword validity.
- Control unit which generate control signals in order to synchronize and control the data flow between the blocks.

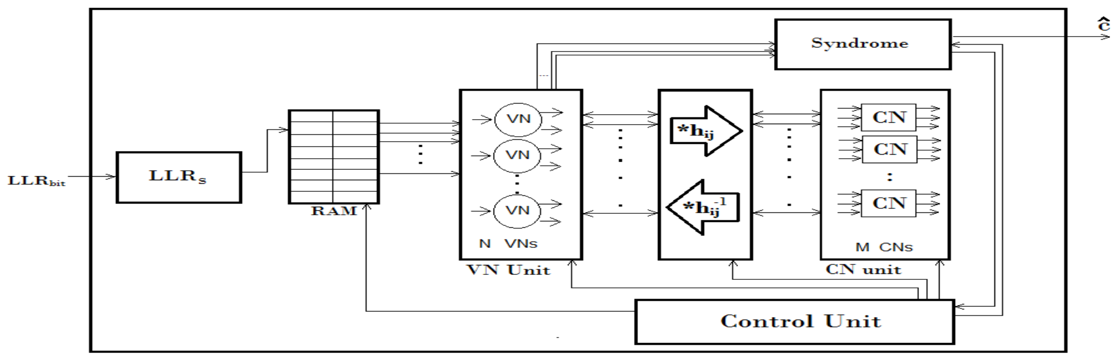

Figure 3.5: Top level NB-LDPC Decoder architecture

Based on this prototype architecture, different implementation trade-offs are obtained through architectural transformations such as resource sharing across VNUs and CNUs and iterative decomposition of the update equations.

### 3.2.2 Layered and Non-Layered architecture

The design can be partitioned into two basic architecture classes : the non-layered (*Full-parallel*) and the layered architecture [10]. The last one also can be divided into two strategies *row parallel*, and *block-parallel*. These architecture classes are depicted in Figure 3.6.

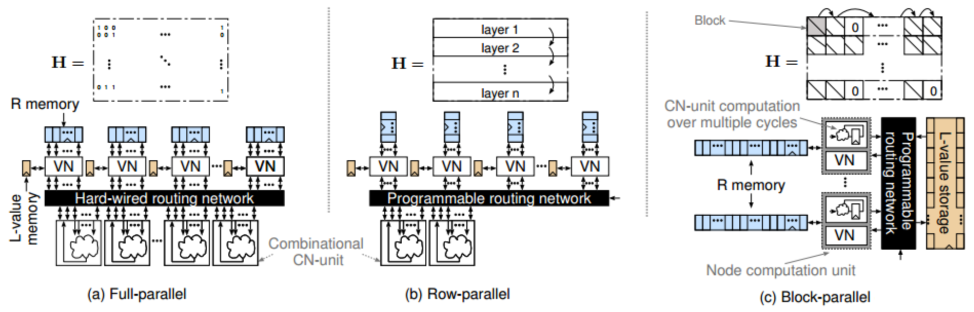

Figure 3.6: Layred and Non-layred Decoder architecture

Layered decoding has been widely adopted to reduce the memory requirement and increase the convergence speed of LDPC decoding.

**The row-parallel** design is a step towards less parallelism. The main objective is to reduce the area while maintaining very high throughput. The principle underlying row-parallel architectures is illustrated in Figure 3.6(b). Essentially, the parity-check matrix is partitioned vertically into layers. An iteration now consists of multiple cycles in which the VNUs access the messages corresponding to the current layer sequentially from a small storage array to compute the output messages and send them to the CNUs through a programmable routing network. The complexity and the amount of bits required to control this programmable routing network heavily depend on the structure of the code and on its partitioning into layers [10].

The row-parallel architectures provide an area advantage over full-parallel designs. Note that additional storage to hold the LLR values computed in the previous iteration, while processing layers of the current iteration, can be avoided for a layered schedule with proper layer selection. In general, the programmable routing network illustrated in Figure 3.6(b) required by the row-parallel architecture provides the flexibility to support multiple parity-check matrices with a single decoder. For this reason, this architecture class has been recently considered in several flexible QC-LDPC decoders tailored to the emerging high-throughput wireless standards IEEE 802.11ad and IEEE 802.15.3c [11].

**Block-parallel** designs rely on further resource sharing and further iterative decomposition. Figure 3.6(c) outlines the architectural principle, which is usually used in combination with the layered message-passing schedule. In essence, this architecture class is obtained by starting from the row-parallel approach and by partitioning the computation of a layer further into multiple cycles, corresponding to multiple blocks in the parity-check matrix. This iterative decomposition simplifies the CN processing and allows for resource sharing also across the VNUs. The block-based processing in conjunction with structured codes significantly facilitates reconfigurability. Due to this reason, this architecture class has been widely employed for flexible decoder implementations.

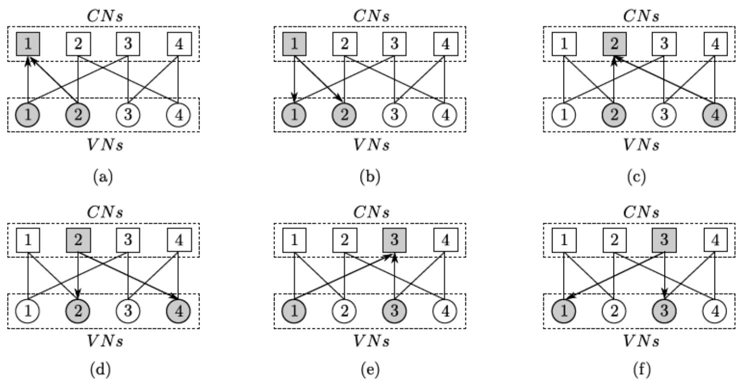

Figure 3.7: Explaining the layred process using Taner graph example

## 3.3 Full-parallel Decoder Architecture

A high-level block-diagram of a **non-layered** (full-parallel) design is shown in Figure 3.6(a). The update equations are mapped into individual VNUs and CNUs that exchange messages through a *hard-wired routing network* [11]. The parallel processing allows each iteration to be performed in a fewer number of clock cycles, by updating all VN units then all CN units. This architecture enables very high throughput since one iteration is performed per fewer number of clock cycles with simple computations that allow for high operating clock frequencies. Unfortunately, the complex routing network that connects CNUs and VNUs turns out to be a major implementation bottleneck for these designs.

Full-parallel architectures have been considered mainly for wire-line communication standards (e.g. 10GBASE-T). Since the full-parallel architecture represents the direct hardware mapping of a specific parity-check matrix, this class cannot provide any flexibility. Therefore, it is generally ill-suited for wireless communication standards that require the support for different parity-check matrices in order to tune code rate and block length.

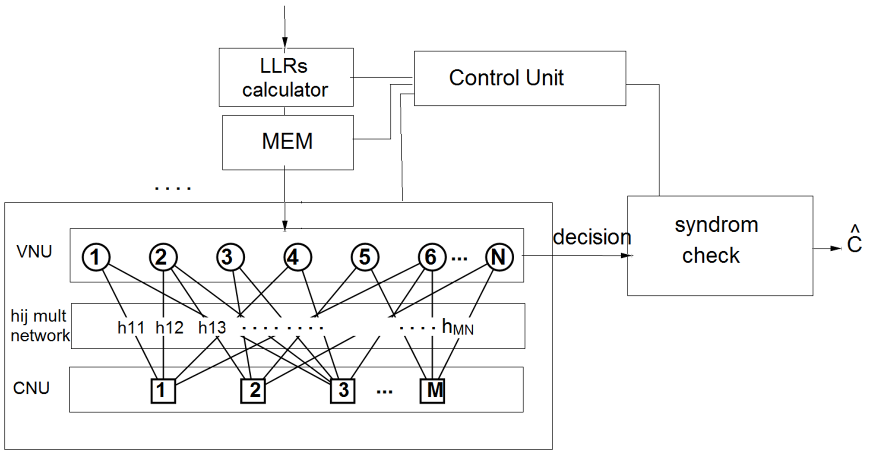

Figure 3.8: full-parallel NB-LDPC Decoder architecture

In our proposed architecture either the VNU or the CNU is working, we can propose another modified architecture which instead of decoding only one codeword ,it estimate two codewords in parallel using a pipeline. the VNU will take two messages ,first for the

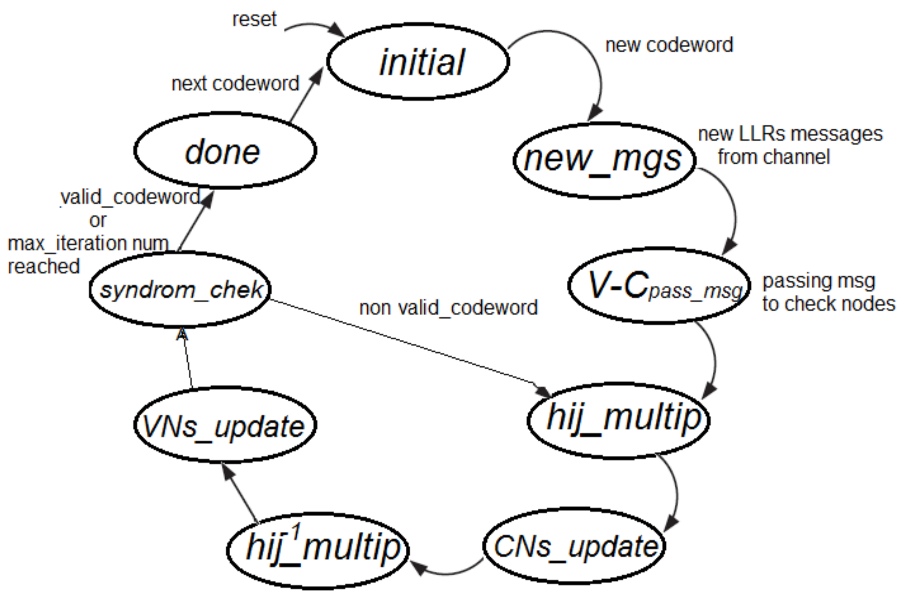

Figure 3.9: Global Control FSM for full-parallel architecture

first codeword, pass it to the CNU which by its role process it and then store the output messages in the memory, at the same time (in parallel) the VNU gets the second codeword correspending messages and pass them to the CNU, At this point ,the VNU and CNU start exchanging their messages with the memory, this architecture has a double throughput as advantage but it suffers from the complexity and the hight hardware memory consuming.

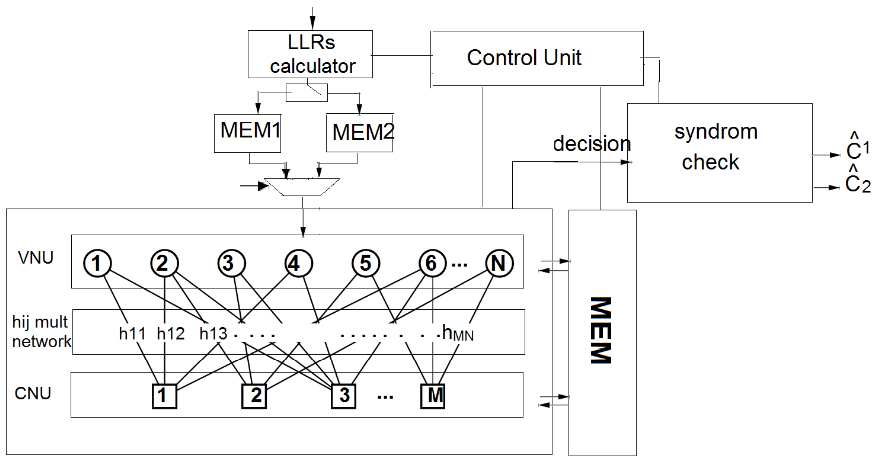

Figure 3.10: Full-Parallel two codewords Decoder

# Conclusion

The project started with an understanding of the NB-LDPC codes concepts, and various variants of the error correcting decoding algorithms were investigated with an eye towards performance and feasible hardware design. Then we have chosen the Min-Max algorithm because of its low complexity in comparison with others algorithms. The complexity of the check node processing is further reduced in the Min-max algorithm with slightly lower coding gain.

It has been shown that the design of NB-LDPC decoders can be partitioned into two main architecture. The non-layered architecture achieve very high throughput in excess of 10 Gbps but cannot provide any flexibility in terms of code rate or block length. The layered architecture which consist of row-parallel class and block-parallel class designs reduce the area compared to the non-layered architecture with different degrees of resource sharing and iterative decomposition at the expense of a degradation in throughput. While both architecture classes offer flexibility, the block-parallel class is especially suitable for QC-LDPC codes as it naturally fits together with the block structure of these codes.

In our project we have focused on the design and implementation of an efficient architecture for the NB-LDPC decoder basic blocks, since wireless communication systems decoders must support a wide range of different parity-check matrices,and because of that we provide flexible decoder which can works with different block lengths and code rates.

We have designed the check node block using the forward backward technique to reduce the implementation complexity. In order to optimize the latency, we have used a practical algorithm to design the elementary check node block, that can work for both cases : all $q$ messages are kept or only the $n_m << q$ most reliable elements. We have also designed the variable node block using elementary blocks for the same reason.
The decoder components were implemented on a Xilinx ISE platform using VHDL language and tested using test benches.

It has been shown that the row-parallel architecture class is highly scalable and supports the full range of throughput requirements found in modern wireless standards. Due to its favorable properties and spatially its flexibility i.e. be able to implement any given code, without changing the design of the decoder. Moreover, it can be adopted to reduce the memory resources.

Finally we proposed a full-parallel architecture for a very hight throughput communication, which is generally used for wire-line communication since it presents hight hardware costs.

# Bibliography

[1] Peter Elias. *Error-free coding.* Technical report (Massachusetts Institute of Technology. Research Laboratory of Electronics) ; 285. Massachusetts Institute of Technology. Research Laboratory of Electronics, 1954.

[2] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, Sep 1981.

[3] Daniel J. Costello Shu Lin. *Error Control Coding: Fundamentals and Applications.* Prentice-Hall Computer Applications in Electrical Engineering Series. Prentice Hall, first edition edition, 1983.

[4] Sarah J Johnson. *Iterative error correction: Turbo, low-density parity-check and repeat-accumulate codes.* Cambridge University Press, 2009.

[5] C. Qian, W. Lei, and Z. Wang. Low complexity ldpc decoder with modified sum-product algorithm. *Tsinghua Science and Technology*, 18(1):57–61, Feb 2013.

[6] Li Zhang, Qin Huang, Shu Lin, Khaled Abdel-Ghaffar, and Ian F Blake. Quasi-cyclic ldpc codes: an algebraic construction, rank analysis, and codes on latin squares. *IEEE transactions on communications*, 58(11):3126–3139, 2010.

[7] Shu Lin, Shumei Song, Bo Zhou, Jingyu Kang, Ying Y Tai, and Qin Huang. Algebraic constructions of nonbinary quasi-cyclic ldpc codes: Array masking and dispersion. In *9th International Symposium on Communication Theory and Applications (ISCTA)*, 2007.

[8] Fang Cai. *Low-complexity Decoding Algorithms and Architectures for Non-binary LDPC Codes.* PhD thesis, Case Western Reserve University, 2013.

[9] Hubert Kaeslin. *Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrication.* Cambridge University Press, 1 edition, 2008.

[10] M. M. Mansour and N. R. Shanbhag. A 640-mb/s 2048-bit programmable ldpc decoder chip. *IEEE Journal of Solid-State Circuits*, 41(3):684–698, March 2006.

[11] Christoph Roth. Vlsi design, optimization, and implementation of channel decoding in wireless systems. 2015.