Multiple Parallel Concatenated Gallager Codes:

High Throughput Architecture Design and Implementation

By:

Di Fan

A thesis submitted in partial fulfilment of the requirements for the degree of

Master of Philosophy

The University of Sheffield

Faculty of Engineering

Department of Electrical and Electronics Engineering

September 2016

Abstract

The design of advanced wireless communication systems has been one of the most important research areas in recent years. High performance error correction schemes and high speed data services are at the heart of these systems.

Due to the excellent performance of Low-Density Parity-Check (LDPC) codes, they are good candidates for many new wireless communication standards. However, complexity, latency scalability and flexibility remain a challenge.

This thesis is concerned with investigating a new approach to coding and decoding LDPC codes based on Parallel Concatenated Gallager Code (PCGCs) using multiple constituent codes. These are a class of concatenated codes built from the direct parallel concatenation of LDPC codes without interleavers. They are characterized by a competitive BER performance while still maintaining the low complexity and flexibility attributes. New methods for encoding and decoding are presented together with BER simulation results showing the performance of these codes. Analysis in terms of the number of constituent codes is also carried out.

Complexity analysis is performed and preliminary implementation results are also given based on a proposed high throughput architecture.

Acknowledgement

I would like to express my sincere thanks to my supervisor Dr. Mohammed Benaissa for his kind guidance and loving inspiration at every stages of my researching period at the University of Sheffield. His patience and encouragement also helped sustain a positive atmosphere in which to do science.

I am grateful to Dr. Hatim Behairy for his valuable feedback and suggestion during the progress meetings, and his constructive criticism has contributed extremely to the evolution of my ideas on the project.

Last, but not the least, I would like to thank Ahmed O Aftan, Zia UA Khan for their friendly support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Channel Coding is one of the most important and active fields in digital communication systems. Low-Density Parity-Check (LDPC) codes are related to the class of Linear Block Codes which were introduced by Gallager with an iterative probability based decoding algorithm, they have simple and less complex iterative decoding algorithm, and show good performance that is very close to the Shannon Limit. This feature makes LDPC codes very attractive and widely used in applications which requiring highly efficient information transfer and reliable data transfer over bandwidth or return channel transmission capacity constraints, such as WiGig (802.11ad) and IEEE 802.15.3c for 60GHz wireless LAN and PAN [2]-[4], low-distance high-rate (802.15.3a) and low-rate high-distance (802.15.4a), WiMax (802.16e), Ethernet (802.3a), 10 Gigabit Ethernet (10GBASE-T) and Second generation scheme for satellite communication (DVB-S2) standards and terrestrial digital television (DVB-T2) [5].

Several modifications based on it like Quasi-Cyclic-LDPC, Parallel Concatenated Gallager Codes (PCGC) to meet the less complication and more flexible with less sacrifice of performance and throughput.

1.1 Background and Evolution

To control errors occur in data transmission over unreliable or noisy communication channels, forward error correction (FEC) or channel coding techniques have been used, the transmitter encodes the message using an error-correcting code in a redundant way and decodes them at the receiver. In the 1940s, Richard Hamming pioneered this field and introduced the first error-correcting code called the hamming code in 1950 [6].

Different kinds of channel coding are used for minimizing the effect of the channel after that. There are two main classes channel coding which are Block Codes and Convolutional Codes. Turbo codes is discovered in 1993 for its good performance of capacity limit approaching (in terms of Shannon limit), it can be seen as a hybrid of the Block Codes and Convolutional Codes, the iterated soft-decoding scheme producing block code by two or more simple convolutional codes combination and an interleaver, used for applications such as the satellite communications and deep space network.

However, in 1962, Gallager first introduced Low-Density Parity-check (LDPC) codes with an iterative belief propagation based decoding algorithm in his Ph.D. thesis [7] predates Turbo codes, these codes are a class of linear block error correcting codes and having relatively less complexity of iterative decoding algorithm, and make them very close to the theoretical Shannon Limits. But there is no significant progress was made after that until 1996, because of the computational cost of LDPC codes were beyond the scope of the processors present at that time. Tanner introduced a graphical representation of LDPC codes in 1981 to construct longer codes from smaller ones, it is known as Tanner graph representation [8]. In 1996, Mackay has proposed a practical implementation of LDPC codes in his landmark paper [9]. Mackay’s successful implementation brings interest back to LDPC codes due to the good coding performance, and most of the work has started on it [10]. In the beginning, most works focused on the binary and regular LDPC codes implementation. In 2001, the concept of a new parity check method irregular LDPC codes were proposed by Michael G. Luby and his colleague [11]. The bit-error-rate curves they got from irregular LDPC codes show the relatively better performance than the regular LDPC codes. And based on LDPC, a structured codes consisting of square sub-matrices named Quasi-Cyclic LDPC (QC-LDPC) codes have been proposed [12], [13], the main feature is that their parity check matrix is composed of several cyclic permutation submatrices, which could be either based on the small random matrix [15] or the identity matrix [16], [17]. The main advantage of Quasi-Cyclic structure compare to randomly constructed codes is that they contain several similar blocks to make encoding and decoding procedure easier [14] which saved much more hardware resources. Considering of the hardware implementation, the computational complexity of encoding of LDPC codes is still high, especially for the large size codes, it may cause delay during the real-time applications [18].

Another new coding method named Parallel Concatenated Gallager Code (PCGCs) have been introduced recently, they are a class of concatenated codes built from the direct parallel concatenation of LDPC codes without interleavers, it got a competitive BER performance while still maintaining the low complexity and flexibility attributes. The proposed coding method in this thesis combined PCGCs with Quasi-Cyclic structure for its every single concatenated LDPC codes, it shows a good performance in saving hardware consumption without too much influence on the good bit error rate performance that PCGC has.

1.2 Related Work

Although no significant work has been proposed on FPGA implementation for multiple paralleled concatenated LDPC decoders since it has been developed in [27], much work has been done on highly-parallel with multi-gigabit performance decoder of QC-LDPC. In particular, Swapnil Mhaske proposed a 2.48Gb/s FPGA-based QC-LDPC Decoder implementation operating at 200MHz on the Xilinx Kintex-7 FPGA in [30] in 2015, and improved the parallel decoding structure from single core decoder to multi-cores decoder in [31], they developed several versions for different degree of parallelism in each iteration, which are compiled by LabVIEW FPGA IP compiler with high-level algorithmic description and get the hardware resource utilization and throughput on the specific device, the implementation is very fast for a standard compliant QC-LDPC using an algorithmic compiler at that time. Swapnil Mhaske's decoder based on the layered decoding architecture which is similar to the decoding method in MPCGC decoder [27], what's more, Yeong-Luh Ueng also proposed an excellent design for non-binary QC-LDPC using the permutation network which based on barrel-shifter and minimum value filter [33], they enabled the layered decoders to be realized efficiently.

Many other designs have been developed for the layer LDPC decoder before [40]-[42], the degree of parallelism and memory in hardware of all the above decoders limit the throughput. Even if the clock frequency is increased to get better performance results, the lift range has a limitation due to the single decoder with a layered architecture requiring processing the whole parity check information exchange, and the power would increase dramatically.

1.3 Research Contributions

This thesis is concerned with investigating a new approach to coding and decoding LDPC codes based on Parallel Concatenated Gallager Code (PCGCs) using multiple constituent codes (MPCGC). There are a class of concatenated codes built from direct parallel concatenation of LDPC codes without interleavers. There are characterized by a competitive BER performance while maintaining the low complexity and flexibility attributes.

In this thesis, new methods for encoding and decoding are presented together with BER simulation results showing the performance of these codes. Analysis in terms of the number of constituent codes is also carried out.

The design of MPCGC codes is addressed and a new architecture is proposed for the efficient implementation of this code. To date no hardware architecture has been reported for implementing MPCGC codes.

This proposed architecture extends the popular QC-LDPC coding structure. In this structure, highly-parallelized and pipelined decoder is achieved by dividing the code into three parts for three decoders, the pipeline depth decreases for lower rate codes with large submatrices because less stages of the check nodes' compare-select tree need to be traversed as the number of inputs are less for the finer check nodes. This is only used in large submatrices because the depth of the check nodes is large enough to justify pipelining it.

In addition, the thesis not only covers the theory, simulation models and architecture but also estimates memory costs and evaluates several main blocks in the component decoder, LabVIEW FPGA IP compiler is used to get realistic resource utilization and throughput estimates based on a specific device (Xilinx Kintex-7X160T). According to the clock frequency, we could have a realistic estimate about the throughput this decoder may achieve. The proposed design takes advantage of a layered architecture and fully pipelining to intelligently distribute the hardware resources, and therefore is suitable for multi-Gb/s wireless network.

1.4 Organization

The organization of the rest of the work is as follows, chapter 2 begins by introducing the decoding theory of error correcting code and different classes of LDPC codes as well as the decoding algorithms. Chapter 3 introduce the coding structure of MPCGC including serial and parallel decoder. Chapter 4 discuss the BER performance of MPCGC with different parameters and situations, looking for a better combination and improvement method. Chapter 5 describes the proposed decoder architecture which combine three component decoders, and introduces the hardware detail for several key blocks. Chapter 6 shows the preliminary results from compilation and synthesis in LabVIEW, estimation of hardware memory cost is also presented. Chapter 7 concludes the thesis with a summary.

Chapter 2

Low-Density Parity-Check Codes

Coding for error correction is a common approach to achieving reliable data transmission in communication systems. The LDPC code is a widely used liner error correcting code transmitting a message over a noisy transmission channel, the coding method is based on iterative belief propagation techniques and the constructions it has allowing the noise threshold to be set very close to the theoretical maximum (called Shannon limit which will be discussed later) for a memoryless channel.

2.1 Noisy-channel coding theorem

Claude E. Shannon presents the concept of information theory in his landmark paper [19] in 1948. He determined fundamental limits on the transmit reliability of data over channels such in figure 2.1 with particular bandwidth and noise characteristics, and how they can be calculated, this theory called Shannon capacity or Shannon limit.

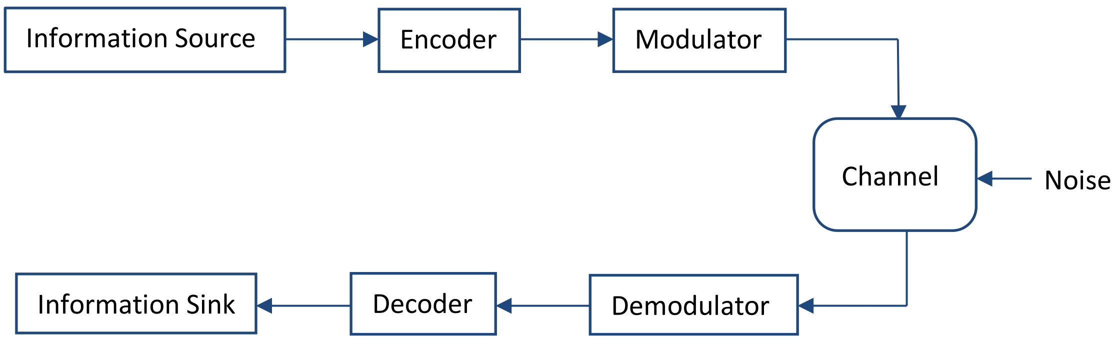


Figure 2.1 Data transmission in noisy-channel

The following capacity formula (2.1) is applying the channel capacity concept to an additive White Gaussian Noise Channel (AWGN).

$$C = W\log_2(1+\frac{S}{N}) \qquad (2.1)$$

Here W is the bandwidth of the channel in Hz, S is the signal power and N is the total noise power of the channel in watt, the S/N is called the signal-to-noise rate, C is measured in bits per second. According to this formula, there is always a theoretical maximum information transfer rate of the communications channel with particular noise level.

The noisy channel coding theorem proves that if properly coded information is transmitted at a rate bellow maximum rate at which data can be sent without error, then the probability of decoding error at the receiver can be made to arbitrarily approach zero exponentially with the code length [20]. This means that, theoretically, we may transmit information nearly without coding error at any rate that below a limit rate. The previous research shows that some good constructed QC-LDPC codes perform about only 1dB to the Shannon limit at the BER of 10^{-6} with sum-product algorithm [21], even some long codes perform very small gap (0.6 dB) between Shannon limit. This makes QC-LDPC a good candidate for the channel coding methods.

2.2 Encoding LDPC codes

The encoding of LDPC codes is rely on the Generator Matrix G which is obtained by taking the transpose of parity check matrix H, it means that generator matrix G and parity check matrix H should be orthogonal to each other, here is mode 2 multiply.

$$G * H^T = 0$$

The encoding process is getting the code vector c by multiplying massage vector m with the generator matrix G.

$$\mathrm{c} = \mathrm{m} * \mathrm{G}$$

The LDPC codes can be represented by C (n, k) which shows the code length n after encoding and the original information bits length k, then the code rate R is defined as R=k/n which give the fraction of information bits in code words.

The generator matrix G is combined with two parts, G=[P^T | I]. The second part is identity matrix to get the original massage vector after encoding. The first part can be obtained either from Mackay's construction theory, or gallager's construction theory which is used for regular LDPC code.

2.3 Classification of LDPC

According to the different parity check matrix H, the LDPC code can be classified into several types. The low density parity check matrix is very sparse which means that there are many '0' elements in the matrix and very less nonzero elements in the matrix, the non-zero elements could be '1', and this kind of codes called binary LDPC codes cause there are only

‘0’ and ‘1’ in the parity check matrix. If the non-zero elements are in the Galois field GF(q), where q>2, it is non-binary LDPC codes or called q-ary LDPC codes. The binary LDPC codes can be classified as regular and irregular codes, the regular LDPC codes means the row weight Wr (number of 1’s in each row of matrix H) should be the same and also the column weight Wc (number of 1’s in each column of matrix H) should be fixed, while the irregular LDPC codes have variable row weight and column weight. And according to how the non-zero elements arranged, it also can be classified into random LDPC codes and structured LDPC codes, which means that the position of non-zero elements in the matrix can be arranged in a specific order.

The early mentioned in Gallager’s paper is a binary, regular, random LDPC codes, and the other kinds of LDPC codes which proposed later are all based on this. Amount of research [22], [23] found that non-binary LDPC codes outperform binary LDPC in the moderate code word length area, and it also can efficiently against mixed types of noise and achieving a good performance in the circumstance of burst errors [24].

2.4 Hard-Decision and Soft-Decision

The advantage of error correcting code is the soft-decision decoding methods, which is a class of algorithm used to decode data that has been encoded with the redundancy information, where the hard-decision takes on a fixed set of possible values. For binary signaling, the received sampled pulses are compared with a single threshold, and they have just two possible results ‘0’ and ‘1’ that decided by the value is greater or less than the threshold, regardless of how close it is to the threshold. The inputs to a soft-decision decoder may take on a whole range of values including of the possibility to be ‘0’ or ‘1’, this extra probability information indicate the reliability of each input data, and estimate the original data more reasonable according to the reliability. Therefore, the soft-decision decoder has better performance in correcting corrupted data than hard-decision.

The soft-in soft-out (SISO) decoder is a type of soft-decision which commonly used in the iterative decoding. The input data contains the possible code bit and also how much likelihood it should be, and also for the output to take on a value indicating the reliability. During the decoding iteration, the soft output is used as the modified soft input to a further iteration until getting the final decision. Or it will input to the outer decoder in a system for concatenated codes.

2.4.1 Tanner Graph

The LDPC codes belong to the class of linear block codes defined by a sparse parity check matrix H, the decoding process relies on the parity check matrix, the structure of parity check matrix have strongly effect on the coding result, so how to build the parity check matrix becomes very important.

Tanner graph is introduced by R. Michael Tanner in 1981 [25], this graph is basically used for the graphical representation of parity check matrix H to make the information calculation in decoding is easy to understand. Tanner graph contains two set of nodes: check nodes and variable nodes(or bits nodes), the every check nodes represent each row of matrix H, and every variable node is each column of matrix H, between the check and variable nodes there are several lines which represent the '1' in matrix H. For example in figure 1, all red lines represent the parity check equation in figure 2.3 which is also marked in red.

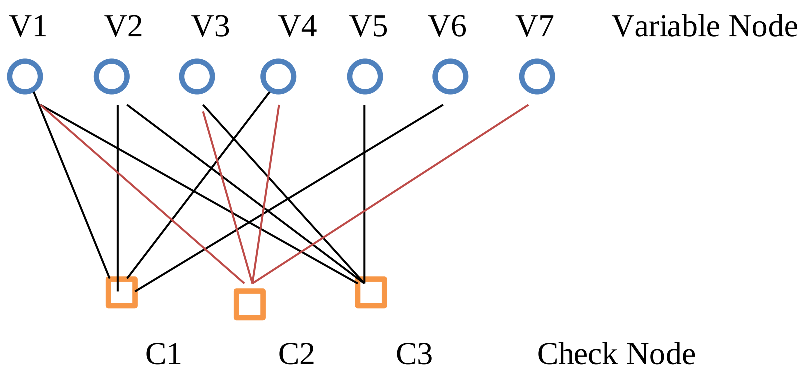


Figure 2.2 Tanner Graph of Parity Check Matrix

$$\underbrace{\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}}_{H} \Rightarrow \begin{matrix} V1+V2+V4+V6=0 \\ V1+V3+V4+V7=0 \\ V1+V2+V3+V5=0 \end{matrix}$$

Figure 2.3 Parity Check of Matrix H

The messages passing between check nodes and variable nodes are the probability that each bit equals '1' or '0', they iterative several times through the lines and calculate by a specific decoding algorithm in every iteration until get the final probability. Here defined the size of parity check matrix H is m*n, m represents the number of parity checks. The code length n is the number of bits in the code which is after the encoding, and should be the same as number of variable nodes, because of k=n-m, then code rate R=(n-m)/n.

2.5 Decoding Algorithms

Compare to the hard decision and one-shot decoders which are used before, LDPC decoder uses soft decision and iterative decoding algorithm called belief propagation. It is highly reduced the Bit Error Rate during the information transmission. LDPC decoder using soft information which has multi-bit resolution, it can represent not only whether the received bits are '1' or '0' (determined by the sign), but also the reliability of this decision (determined by the magnitude), and then the detector decides the received bits is '1' or '0' with the decoding algorithm and output a soft information to next iteration, the soft decoder never making a hard decision until an output is required. The mainly used decoding algorithms will be discussed next.

2.5.1 Sum-Product Algorithm

The Sum-Product Algorithm (SPA) also called Belief Propagation Algorithm (BPA), it is the traditional formulation of the message passing algorithm, using the iteration to calculate the probability information. The complexity of SP algorithm is directly related to the number of '1's in parity check matrix H, more '1's it has then it will be more complexity. Some terms in the algorithm are given below:

q_{ij}: message passing from variable nodes v_i to check node c_j.

r_{ij}: message passing from check node c_j to variable nodes v_i.

$Row_{[j]}$: represents the position of '1' in row number j. $Row_{[j]}=\{i:h_{j,i}=1\}$

$Row_{[j]\backslash\{i\}}$: represents the position of '1' in row number j except column number i.

$Col_{[i]}$: represents the position of '1' in column number i. $Col_{[i]}=\{j:h_{j,i}=1\}$

$Col_{[j]\backslash\{i\}}$: represents the position of '1' in column number i except row number j.

The main idea of SPA is passing the probability information from variable nodes to check nodes, then calculating and passing back the new probability from the check nodes to variable nodes, after several times iterative correcting and getting the final information.

In the probability domain, the SP algorithm runs as follow. If $h_{j,i}=1$ then:

(1) Initialization

All variable nodes have their prior values (priori information), they are based on the channel model, these priori information are the probabilities of the corresponding bits equals ‘0’ and ‘1’.

$$q_{ij}(0)=P_i^{pr}(c_i=0|y_i)$$

$$q_{ij}(1)=P_i^{pr}(c_i=1|y_i)$$

Where: $q_{ij}(0)+q_{ij}(1)=1$

Here the $q_{ij}(n)$ (n=0,1) is the information sent from variable node i to check node j, the $q_{ij}(0)$ and $q_{ij}(1)$ are sent from the same node, present the probability of this bit equals zero and one respectively.

(2) Update Check Nodes step

This step will update the information on each check nodes with the received probabilities from variable nodes, because of the information transfer direction in matrix H is horizontal, it also called horizontal step. As it shown on the tanner graph below, depending on the position of ‘1’s in matrix H, the information transmitted through the lines.

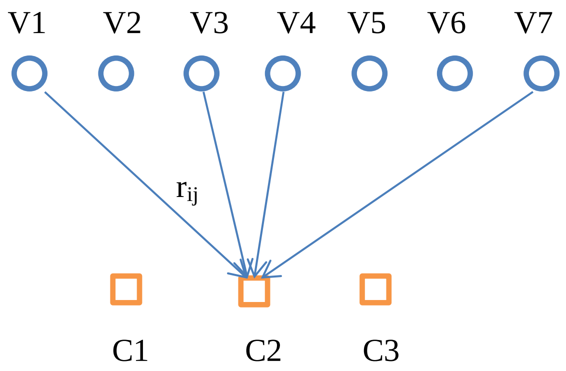


Figure 2.4 Update Check Nodes

All the linked variable nodes passed the information to the check nodes if in the first iteration, if not, every variable nodes passed the information to the check nodes except from c_j which is sent back to it.

$$r_{ij}(0)=\frac{1}{2}\left(1+\prod_{i'\in Row[j]\backslash\{i\}}(q_{i'j}(0)-q_{i'j}(1))\right)$$

$$r_{ij}(1)=\frac{1}{2}\left(1-\prod_{i'\in Row[j]\backslash\{i\}}(q_{i'j}(0)-q_{i'j}(1))\right)$$

For example, if update the value of check node C_j, all the probabilities on variable nodes q_{ij}, i=(0:m), j=(0:n) will be calculated together except the probability r_j which is received from check node C_j itself.

(3) Update Variable Nodes step

This step will update the value on each variable nodes with the received intrinsic probabilities from check nodes last step, because of the information transfer direction in matrix H is vertical, it also called vertical step. It is also shown on the tanner graph below.

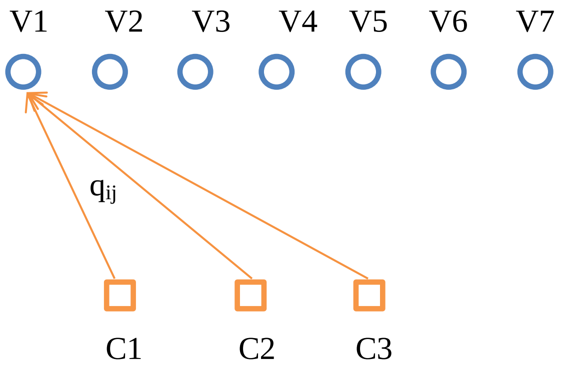


Figure 2.5 Update Variable Nodes

All the check nodes passed the intrinsic probabilities to the variable nodes except from itself. Then calculated as below.

$$q_{ij}(0) = c_{ij} p_i^{pr}(0) \prod_{j' \in Col[i] \setminus \{j\}} r_{ij'}(0)$$

$$q_{ij}(1) = c_{ij} p_i^{pr}(1) \prod_{j' \in Col[i] \setminus \{j\}} r_{ij'}(1)$$

Here c_{ij} is a normalizing factor to make sure $q_{ij}(0)+q_{ij}(1)=1$. Also removed the probability of check node C_j sent to variable node V_i so that only extrinsic information is passed.

(4) Decision

The previous step computes the new information that variable nodes will send to check nodes in the next iteration, which only contains extrinsic information. To obtain the posterior probabilities and decide the final value of each bits,(P_i^{post}), the information from check node j is not excepted and a new normalization constant ci replaced the c_{ij}.

$$p_i^{post}(0) = c_i p_i^{pr}(0) \prod_{j' \in Col[i]} r_{ij'}(0)$$

$$p_i^{post}(1) = c_i p_i^{pr}(1) \prod_{j' \in Col[i]} r_{ij'}(1)$$

Here the p_i^{post} is the soft output of variable node vi, then using the decision rule below to obtain the hard decision output.

$$\hat{c}_i = \begin{cases} 0, & if\ p_i^{post}(0) \geq p_i^{post}(1) \\ 1, & else \end{cases} \tag{2.2}$$

The hard decision can be taken to check if all the parity checks are satisfied (H*bi=0), if it is satisfied or the maximum number of iteration have reached, outputting the final result which is computed in the hard decision.

2.5.2 Logarithm BP algorithm

As the multiplication requires large amounts of hardware consumption and power, to make a mathematical simplification, information can be transmitted with log likelihood ratios (LLR) instead of in probability domain, it is known as Logarithm BP algorithm which changes the multiplications to additions. LLR for the priori information is defined below.

$$L^{pr}(c_i) = \log\frac{p_i^{pr}(c_i = 0|y_i)}{p_i^{pr}(c_i = 1|y_i)} = \frac{2}{\sigma^2}E_c \qquad (2.3)$$

Where σ^2 is the noise variance of the channel and E_c is the energy per transmitted codes. This equation is only for AWGN channel. Similarly the LLR for variable to check nodes, check to variable nodes and posterior information are:

$$L(r_{ij}) = \log\frac{r_{ij}(0)}{r_{ij}(1)}$$

$$L(q_{ij}) = \log\frac{q_{ij}(0)}{q_{ij}(1)}$$

$$L^{post}(c_i) = \log\frac{p_i^{post}(0)}{p_i^{post}(1)}$$

In the LLR domain, the calculation process as follows.

(1) Initialization

Initialize all variable nodes with their corresponding $L^{pr}(c_i)$ which calculated from equation (2.3).

(2) Update Check Nodes step

The equation bellow shows the updating of each check nodes' information from the neighboring variable nodes i, the sign and magnitude here should be processed separately.

$$L(r_{ij}) = -\tanh^{-1}\left(-\sum_{i\prime\in Row[i]\backslash\{j\}} \tanh\frac{|L(q_{i'j})|}{2}\right)\left(\prod_{i\prime\in Row[i]\backslash\{j\}} sign(\,L(q_{i'j}))\right)$$

(3) Update variable nodes step

Variable nodes updating based on the message computed form check nodes and the prior information.

$$L(q_{ij}) = L^{pr}(c_i) + \sum_{j' \in Col[i] \setminus \{j\}} L(r_{ij'}) \tag{2.4}$$

(4) decision

Different from probability domain SPA, the information from check node j should be added to the extrinsic information in equation (2.4) to compute the soft output for decision in LLR domain.

$$L^{post}(c_i) = L^{pr}(c_i) + \sum_{j' \in Col[i]} L(r_{ij'})$$

Then make a hard decision on $L^{post}(c_i)$ for the early termination, typically determined by the sign and decide if bit equals to '1' or '0'.

$$\hat{c}_i = \begin{cases} 0, & if\ L^{post}(c_i) \geq 0 \\ 1, & else \end{cases}$$

Repeating these steps for iteration until the bit code result meet the parity check requirement or the maximum number of iteration is reached.

2.5.3 Min-Sum Algorithm

The SPA and other modifications based on it such as Logarithm Belief Propagation Algorithm shown a good performance on reducing the bit error, but they still have a complex expression for check nodes update which requires high computation, and still not ideal for the hardware implementation cause it needs too much area in hardware. The Min-Sum Algorithm (MSA) simplified the computation only in addition and subtraction and modified the expression of value only require the calculation of sign and minimum value. The optimized for computational cost of MSA will definitely come at the expense of decoding performance, but the performance can be improved by increasing the number of iteration properly, as it just increases the latency of hardware.

(1) Initialize the APP ratio

Here the $L_i^{(0)}$ is the original LLR for c_i which can be calculated by the equation bellow.

$$L_i^{(0)} = \ln\left\{\frac{P(c_i = 0|y_i)}{P(c_i = 1|y_i)}\right\}$$

(2) Update Check Nodes step

The MSA for this step not involves the inverse hyperbolic tangent function which must be implemented with the look up tables (LUTs), it makes the hardware implementation friendly. The k here represents the kth decoding iteration.

$$R_{ij}^{(k)} = \left[\prod_{i' \in Row[j] \backslash \{i\}} sign\left(L_{i'j}^{(k-1)}\right) \right] \cdot \min_{i' \in Row[j] \backslash \{i\}} \left\{ \left| L_{i'j}^{(k-1)} \right| \right\} \tag{2.5}$$

(3) Update Variable Nodes step

The summation step in the variable node adding all neighboring check nodes' information exclusive the information form itself to get posterior LLR.

$$L_{ij}^{(k)} = L_i^{(0)} + \sum_{j' \in Col[i] \backslash \{j\}} R_{ij'}^{(k)} \tag{2.6}$$

(4) Decision

At the end of each iteration, decision is taken as following equations.

$$L_i^{(k)} = L_i^{(0)} + \sum_{j \in Col[i]} R_{ij}^{(k)}$$

$$\hat{c}_i = \begin{cases} 0, & if\ sign(L_i) = 1 \\ 1, & else \end{cases}$$

If $\hat{c}H^T$=0, where $\hat{c} = (\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_n)$, or meet the maximum number of iteration, the code words $\hat{c}$ should be output as the decoding result.

2.5.4 Modified Min-Sum Algorithm

This algorithm simplified the general MSA only on the computation for check nodes which is the most complexity and important part, and the rest steps are as same as the general MSA.

In check nodes updating step here, the minimum value will be selected from all the input values instead of excluding the input which will be passing back, the equation modified as (2.7). The performance results showing a slight difference between modified MSA and general MSA, compare to the cost of complexity, the modified MSA is more implementation friendly.

$$R_{ij}^{(k)} = \left[\prod_{i' \in Row[j] \backslash \{i\}} sign\left(L_{i'j}^{(k-1)}\right) \right] \cdot \min_{i' \in Row[j]} \left\{ \left| L_{i'j}^{(k-1)} \right| \right\} \tag{2.7}$$

Selecting the smallest magnitude in the equation above may results in an overestimation of the check node to variable node information because of the large summation, there are another two options to correct this, one is optimizing the magnitude by a constant α which is greater than one, the equation (2.8) shows how this constant works and the other one subtracts an offset γ which is shown in equation (2.9), meanwhile, keep the magnitude value always large than zero.

$$R_{ij}^{(k)} = \left[\prod_{i' \in Row[j] \setminus \{i\}} sign\left(L_{i'j}^{(k-1)}\right) \right] \cdot \frac{\min_{i' \in Row[j]} \left\{ \left| L_{i'j}^{(k-1)} \right| \right\}}{\alpha} \tag{2.8}$$

$$R_{ij}^{(k)} = \left[\prod_{i' \in Row[j] \setminus \{i\}} sign\left(L_{i'j}^{(k-1)}\right) \right] \cdot \max\left\{ \min_{i' \in Row[j]} \left\{ \left| L_{i'j}^{(k-1)} \right| - \gamma \right\}, 0 \right\} \tag{2.9}$$

The correction parameters α and γ can be designed to have a different value according to the specific decoder, and also have different times during the iteration. It is obviously that the method of γ correction has a better implementation since it only needs a subtractor instead of a divider in α correction, and also has a finer tuning range than α correction [26].

Chapter 3

Multiple Parallel Concatenated Gallager Codes

As LDPC codes shows such a good performance (in terms of error probability) especially for the binary symmetric channels, a new class of concatenated codes named Multiple Parallel Concatenated Gallager Codes (MPCGCs) designed from the parallel concatenation of LDPC codes has been proposed [27], the motivation is using the good LDPC codes in the turbo structure. They are considered as a one of the best error correcting procedure based on linear block code.

3.1 MPCGCs Encoding

MPCGCs breaking a long code into multiple small LDPC codes to offer scalability and scope for improving performance in practical implementation, especially for the resource constrained and delay sensitive applications.

There are m component LDPC encoders in the encoding part of MPCGCs, each encoder has its parity check matrix and generator matrix, the original code words $\underline{c}$ will go through to every component encoder before go into the channel. As shown in Figure 3.1, the $\underline{c}$ denotes the systematic information bits, and $\underline{e}^m$ is the parity information bits which is generated based on systematic information by the m^{th} encoder. For example, if the code rate of each LDPC encoder is half, then the total code rate of MPCGCs is R=1/(m+1).

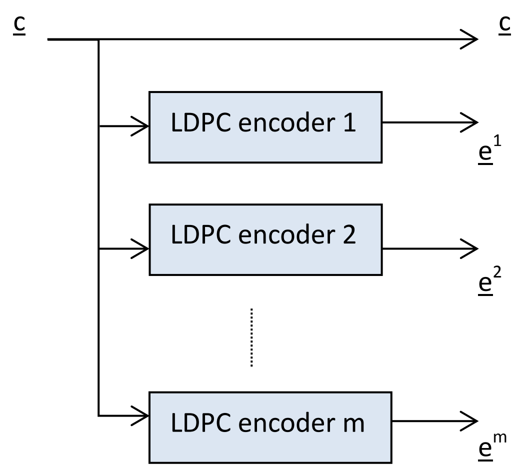


Figure 3.1 MPCGCs Encoder

The final encoded code words should contain the original systematic information $\underline{c}$ and several parts of parity information $[\underline{e}^1\ \underline{e}^2 \ldots \underline{e}^m]$ which will be send to demultiplexer to prepare for decoding.

3.2 MPCGCs Decoding

The decoder should have matching component decoders which seem like the single LDPC decoders, using soft probability information and iterative decoding algorithm for calculating. There are two techniques for decoding, serial decoding and parallel decoding structure, they all contain several component LDPC decoders to compute the a posteriori probability with a specified decoding algorithm from received a priori information during each super iteration.

3.2.1 Serial Decoder

The MPCGCs serial decoder illustrated in Figure 3.1, each one circle here is one super iteration which using m component decoders. Firstly input the initial probability $\underline{P}^0(\underline{c})$ and encoded words $\underline{d}^m$ after adding noise ($\underline{d}^m=[\underline{e}^0\ \underline{e}^m]$) to the matching component decoder respectively and prepare for the iterative calculation. During first super iteration, the first component decoder received $\underline{d}^1$(sequence $\underline{e}^0$ and $\underline{e}^1$ which is systematic information and parity information respectively) and computes the a posteriori probability $\underline{P}^1(\underline{c})$ which will be the input of the second component decoder as a priori information, and there is no a priori information for the first component decoder cause it is the first iteration and the information bits are -1 or +1for binary phase shift keying (BPSK) modulation. And then, the second component decoder using the received $\underline{d}^2$ ($\underline{d}^2=[\underline{e}^0\ \underline{e}^2]$) with an extrinsic information $\underline{P}^1(\underline{c})$ computes the a posteriori probability $\underline{P}^2(\underline{c})$. The decoding operation is similar for the next several decoders, they all calculates the a posteriori probability $\underline{P}^m(\underline{c})$ according to the received $\underline{d}^m$(sequences $\underline{e}^0$ and $\underline{e}^m$) and an extrinsic information $\underline{P}^{m-1}(\underline{c})$.

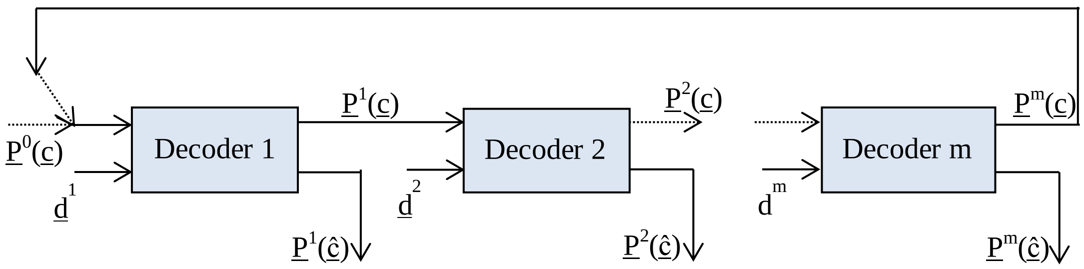


Figure 3.2 MPCGCs Serial Decoder

What’s more, each component decoder has an output $\underline{P}^m(\underline{\hat{c}})$ for the decision of early convergence, the iteration will stop early when meet the condition.

3.2.2 Parallel Decoder

As the performance of serial decoding can't satisfy the basic requirement for bit error rate, even it is worse than a simple LDPC decoder, our work is focused on another decoding technique which is parallel decoding. The structure is illustrated in Figure 3.3, the all component decoders in each column is in a same super iteration, each one using its own parity check matrix to decodes its own code words, and computes the corresponding a posteriori probabilities $\underline{P}^{m}(\underline{c})$, every component decoder sends their a posteriori probabilities to the next super iteration as the a priori information, and during next super iteration, each component decoder computes the a posteriori probability using the received a priori information from previous super iteration except its own, that is to say, the a priori information from all other m-1 component decoders are received. The super iteration process will continue until all component decoders converge to valid code words or reach the maximum number of super iterations.

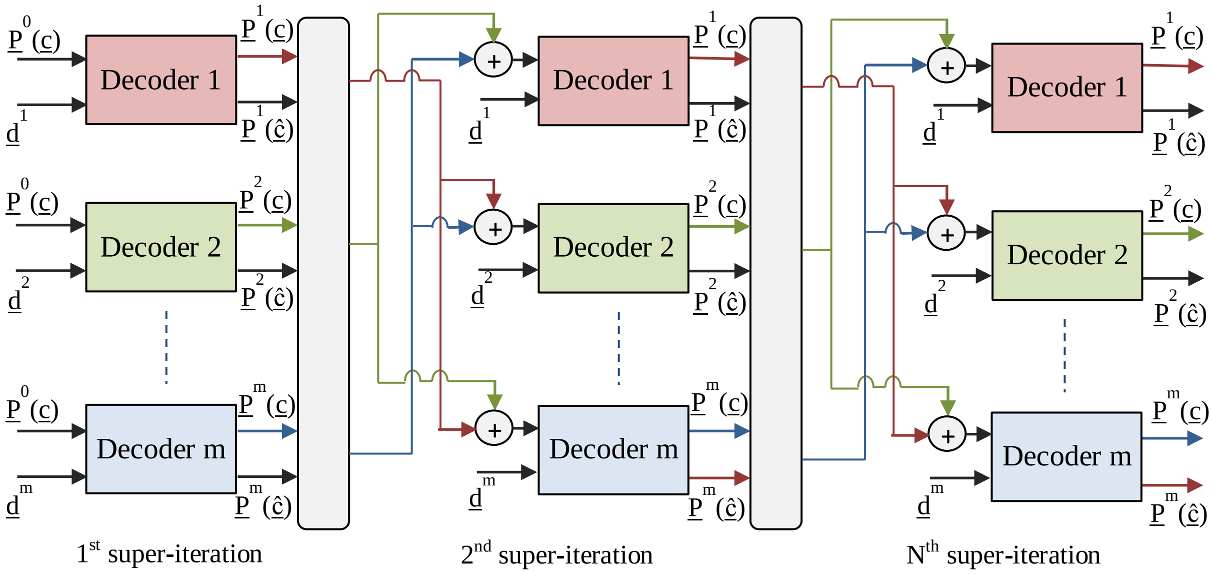


Figure 3.3 MPCGCs Parallel Decoder

(1) MPCGCs Parallel Decoder with two component decoders

To understand how does the extrinsic (a priori) information calculates between component decoders exactly, let's start from the simplest MPCGCs which only have two component decoders. The decoding process in Figure 3.4 bellow shows how do the two decoders work together. Firstly, the MPCGCs decoder de-multiplex the code words which is received from the noise channel and divided them into two vectors $\underline{d}^{1}$ and $\underline{d}^{2}$, and send them to the first and second component decoders respectively. After decoding through each single LDPC decoder, the extrinsic information we got will be exchanged and send to the other component decoder as a priori information for next super iteration.

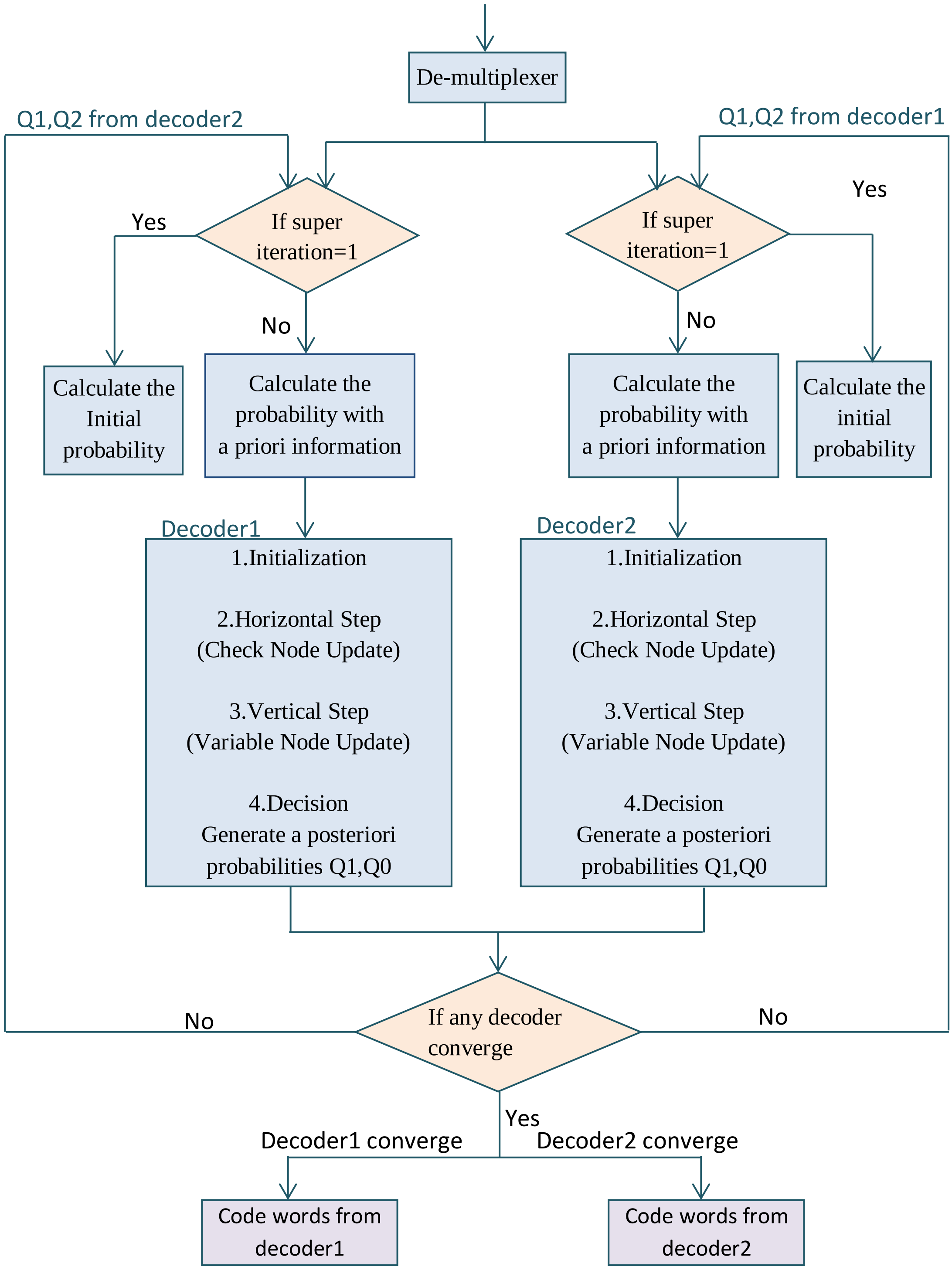


Figure 3.4 Decoding process for MPCGC with two component decoders

According to the Gaussian probability density function, the probability for code word equals to +1 at site l should be calculated as equation (3.1)

$$f_l(1) = p(c_l = +1|d_l) = \frac{1}{1 + \exp\left(\frac{-2d_l}{\sigma^2}\right)} \tag{3.1}$$

Here d_l is code word after BPSK modulation in AWGN and σ denote the channel model. When the probability of source information bits are $p(u_l=-1)=p(u_l=+1)=1/2$, then the probability that the code word is -1 at site l is

$$f_l(0) = p(c_l = -1|d_l) = 1 - f_l^1 = \frac{1}{1 + \exp\left(\frac{2d_l}{\sigma^2}\right)} \tag{3.2}$$

Then the new extrinsic information for next super iteration should be generated from the outputs of the other two decoders as the a priori information which is shown bellow.

$$f_l(1) = \frac{1}{1 + \left[\left(\frac{p(c_l = -1|d_l)}{p(c_l = +1|d_l)}\right)\exp\left(\frac{-2d_l}{\sigma^2}\right)\right]} \tag{3.3}$$

$$f_l(0) = \frac{1}{1 + \left[\left(\frac{p(c_l = -1|d_l)}{p(c_l = +1|d_l)}\right)\exp\left(\frac{2d_l}{\sigma^2}\right)\right]} \tag{3.4}$$

(2) MPCGCs Parallel Decoder with three component decoders

The three component decoders show better performance than two in practical application, but more complex in exchanging the extrinsic information. Firstly, the de-multiplexer dividing the received vectors into three part of code words $\underline{d}^1$ $\underline{d}^2$ and $\underline{d}^3$, and sending them to the first, second and third component decoders respectively. After the normal LDPC decoding during first super iteration, the exchanging calculation of extrinsic information which from the other two component decoders is needed during the all remaining super iterations, update these two equations (3.1) and (3.2) above to (3.5) and (3.6), adding the extrinsic information and their modulus k1 and k2 as a priori information, the extrinsic information is used in every component decoders are come from another two component decoders except itself.

$$f_l(1) = \frac{1}{1 + \left[k_1\left(\frac{p(c_l = -1|d_l)}{p(c_l = +1|d_l)}\right)k_2\left(\frac{p(c_l = -1|d_l)}{p(c_l = +1|d_l)}\right)\exp\left(\frac{-2d_l}{\sigma^2}\right)\right]} \tag{3.5}$$

$$f_l(0) = \frac{1}{1 + \left[k_1\left(\frac{p(c_l = -1|d_l)}{p(c_l = +1|d_l)}\right)k_2\left(\frac{p(c_l = -1|d_l)}{p(c_l = +1|d_l)}\right)\exp\left(\frac{2d_l}{\sigma^2}\right)\right]} \tag{3.6}$$

The equations (3.5) and (3.6) here are how the extrinsic information be calculated among three decoders, and they are specific for BP algorithm. For Min-Sum algorithm or modified Min-Sum algorithm, the calculations are shown below:

According to the calculating method of the probability of a priori information P_l^{pr}.

$$P_l^{pr}(c_l = +1|d_l) = \frac{1}{1 + \exp(\frac{-2d_l}{\sigma^2})}$$

$$P_l^{pr}(c_l = -1|d_l) = 1 - P_l^{pr}(c_l = +1|d_l)$$

And the a priori information λ_l^m (m is the number of component decoders, here m=[1,2,3]) coming from the other component decoders is calculated as:

$$\lambda_l^m = \frac{p_l^m(c_l = +1|d_l)}{p_l^m(c_l = -1|d_l)}$$

Then we can get the modified equation of (3.5).

$$f_l(1) = \frac{1}{1 + \left[k_1\lambda_l^m \cdot k_2\lambda_l^m \cdot \exp\left(\frac{-2d_l}{\sigma^2}\right)\right]}$$

For Min-Sum algorithm, the new a priori information is L_l which computed bellow, F_l^m represent the new information after extrinsic exchanging.

$$L_l = \ln\frac{p_l^m(0)}{p_l^m(1)} = \frac{-2d_l}{\sigma^2}$$

$$F_l^m = \ln\left(k_1\lambda_l^m \cdot k_2\lambda_l^m \cdot \exp\left(\frac{-2d_l}{\sigma^2}\right)\right)$$

In this case, the modulus k_1 and k_2 are fixed and equal to 1, the information exchange for each component decoders are:

$$\begin{cases} F_l^1 = L_l^2 + L_l^3 - \dfrac{2d_l}{\sigma^2} \\ F_l^2 = L_l^1 + L_l^3 - \dfrac{2d_l}{\sigma^2} \\ F_l^3 = L_l^1 + L_l^2 - \dfrac{2d_l}{\sigma^2} \end{cases}$$

Chapter 4

Simulation Results

4.1 Different Number of Component Decoders

The number selecting of MPCGCs component decoders have an effect on the whole decoder's performance, it may get a better result with more component decoders theoretically cause there much more parity check matrix be taken to correct the code words. But as long as the number increased, more utilization of hardware source and decoding time will be taken, the appropriate number of decoders becomes vital in this design.

For the code (192,768) with code rate 1/4 here, two, three and four component decoders are simulated.

(1) Case for 2 component decoders:

As shown in Figure 4.1, separating the original systematic information into 2 parts equally for 2 component decoders respectively, the new partial systematic information linked with parity information to get new partial code words with code rate 2/5, each component decoder process this partial code words respectively. The size of parity matrix for partial code words will be (288,480).

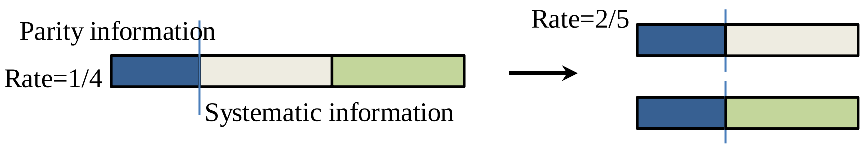


Figure 4.1 Case for 2 component decoders

(2) Case for 3 component decoders:

As shown in Figure 4.2, separating the original systematic information into 3 parts equally for 3 component decoders respectively, the new partial systematic information linked with parity information to get new partial code words with code rate 1/2, each component decoder process this partial code words respectively. The size of parity matrix for partial code words will be (192,384).

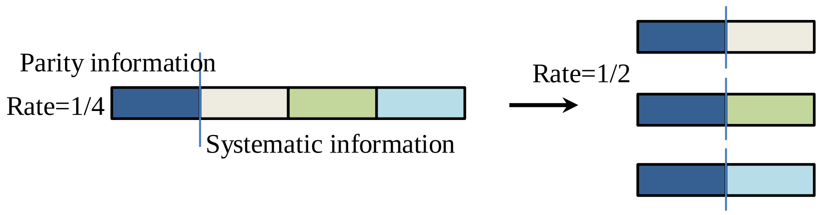


Figure 4.2 Case for 3 component decoders

(3) Case for 4 component decoders:

As shown in Figure 4.3, separating the original systematic information into 4 parts equally for 4 component decoders respectively, the new partial systematic information linked with parity information to get new partial code words with code rate 4/7, each component decoder process this partial code words respectively. The size of parity matrix for partial code words will be (144,336).

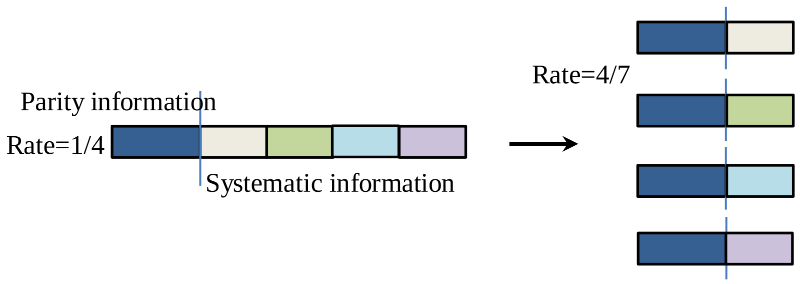


Figure 4.3 Case for 4 component decoders

As shown in Figure 4.4, The BER performance may increase while more component decoders are adapt, but the increasing of decoders lead to a huge utilization of hardware source, as there are no more obviously improve performance while decoder more than 3, then 3 parallel component decoders adapted in this proposed design. Also take into consideration that the density of parity check matrix in each component decoder might affect the performance, we will discuss it in next paragraph, then making an appropriate combination of matrices in 3 decoders may getting good results with relatively low area consumption.

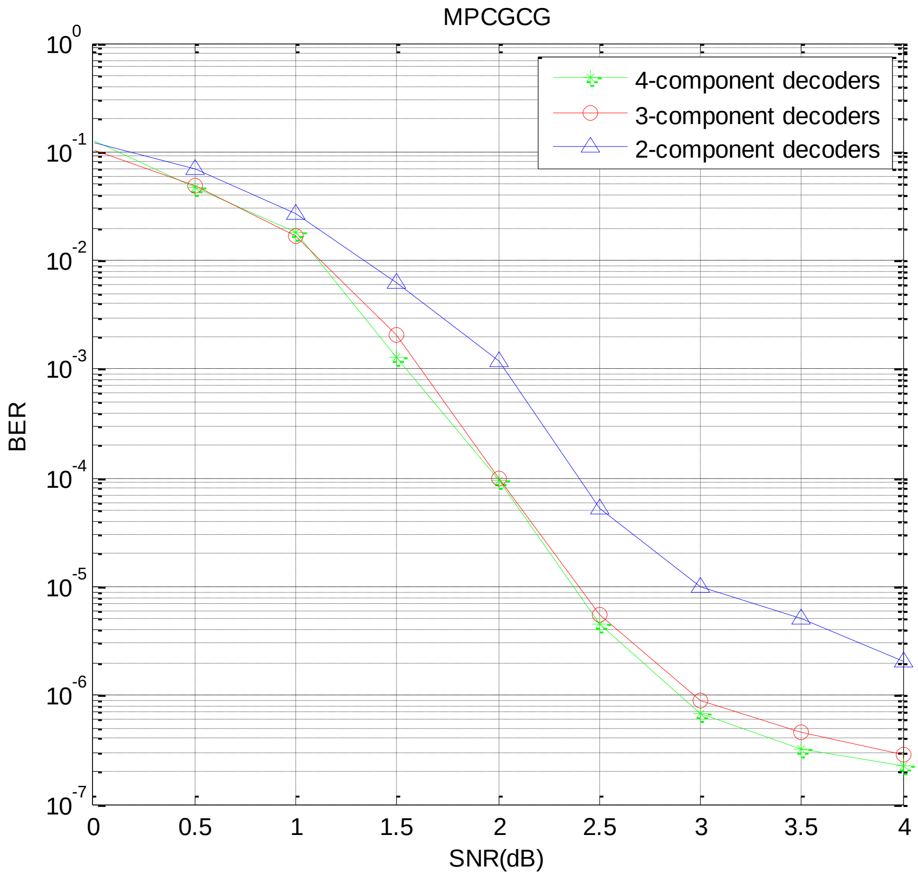


Figure 4.4 Comparison of different number of component decoders

4.2 MCW Combination

As there are several component decoders in MPCGC, and the parity check matrices should be different, choosing an appropriate combination of matrices is vital because it affects the BER performance directly. Firstly, let's discuss the concept of Mean Column Weight (MCW), it is the average number of column weight (the number of '1's) in whole matrix, as the column weight in each column is different, define the maximum value is m and the minimum value should be 1. λ_i is a fraction represents the proportion of columns which weight equal to i in the parity matrix. Then the MCW can be calculated bellow:

$$MCW \triangleq \sum_{i=1}^{m} i\lambda_i \tag{4.1}$$

The research was done before found that the BER performance of LDPC codes with low MCW is better than the relatively higher MCWs at low to moderate signal noise rate (E_b/N_0) region, while it is worse at high signal noise rate [28]. The aim of this design is to combine the strength of LDPC codes with low MCW during the low to moderate E_b/N_0, and high MCW in the high E_b/N_0 region.

The main idea of super-iterative decoding in MPCGCs is sharing the extrinsic information with each component decoders on the systematic information. To measure the quality of extrinsic information (a priori messages), a Gaussian approximation to the probability density function for the extrinsic information can be used [29], because of the density value approach Gaussian distribution for SP algorithm with the increasing iterative number. The calculation of signal-to-noise rate (SNR) is based on the variance value σ^2 and the mean value μ which are shown bellow.

$$SNR = \frac{\mu^2}{\sigma^2}$$

Here for log-likelihood based on Gaussian distribution, mean value μ can be approximated by equation bellow.

$$\mu = \frac{\sigma^2}{2}$$

The Gaussian approximation simplified the analysis of probability for each individual LDPC codes with different MCW by estimating instead of simulating the actual probability density of extrinsic information, here the higher SNR value represents the better quality of extrinsic information.

Figure 4.5 illustrates the quality of extrinsic information with three different MCWs (MCW1=1.94, MCW2=2.81, MCW3=1.81) in each MPCGCs component decoder and MCW combination of these three decoders at E_b/N_0=0.5dB, the SNR_i denotes the SNR for the a priori information which is the input bits for each single LDPC decoder, and SNR_o is SNR for the extrinsic information which is the output bits. It is clearly that code with low MCW shows better performance in low SNR_i and as SNR_i increases in moderate area, the SNR_o starts increasing fast with relatively high MCW. The combination of three serial decoders scheme increase the decoding performance by maintaining the good asymptotic performance, and the value of three MCWs is crucial for reaching a good performance. We discuss the first graph of Figure 4.5 bellow to show how extrinsic information transmitted among three single decoders.

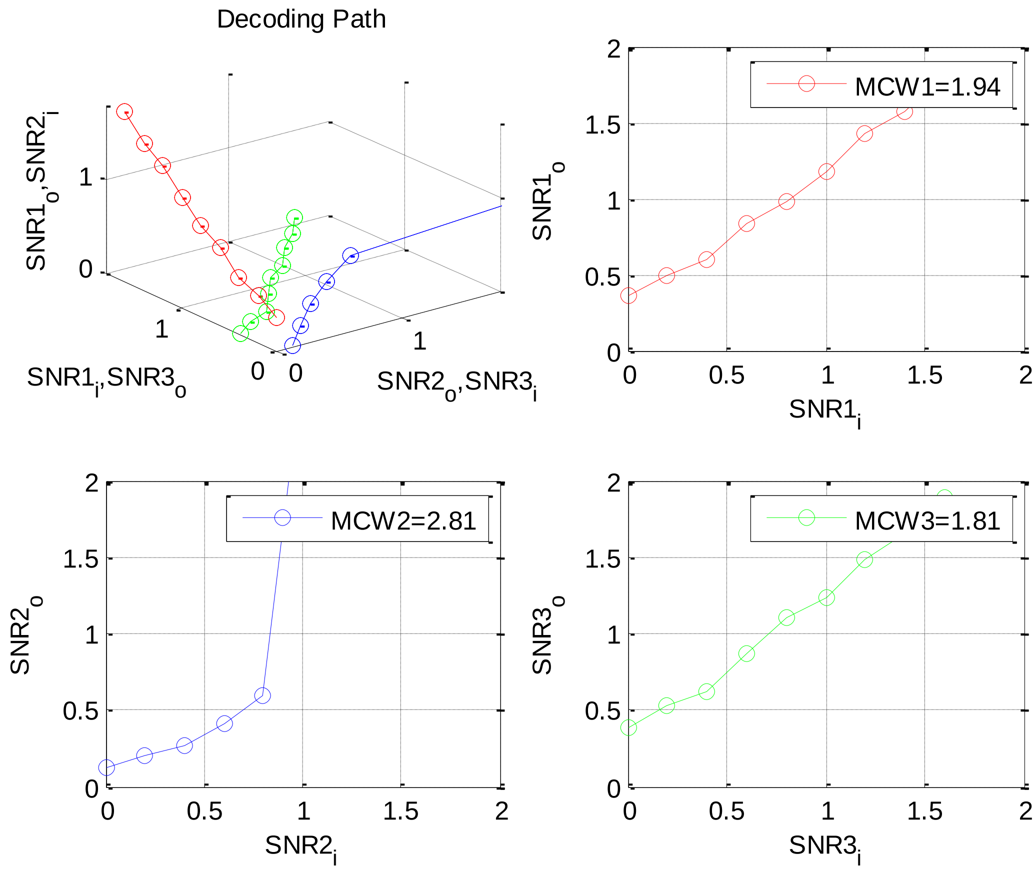


Figure 4.5 Effect of different MCWs on extrinsic information

Firstly, the a priori information input into the first decoder and output as extrinsic information which will be passed to the second decoder as its a priori information, this process is shown in Figure 4.6 where $SNR1_o$ becomes $SNR2_i$ on the surface of red curve (MCW1=1.94), and then $SNR2_o$ becomes $SNR3_i$ and $SNR3_o$ becomes $SNR1_i$ after passing through second and third decoders respectively in each super iteration, the green arrows represent the decoding path during first super iteration. Similarly, the remaining iterative decoding repeats this process until converging. To decrease the BER and get faster convergence, the total decoding path should be reduced as much as possible. Here setting a relatively lower MCW for first decoder that can provide higher $SNR1_o$ during first few iterations while $SNR1_i$ is low, and it leads a high $SNR2_i$ for next decoder, then setting a higher MCW for the second decoder to have a balance that making the $SNR2_o$ increase slightly, therefore, a lower MCW can be used in the third decoder.

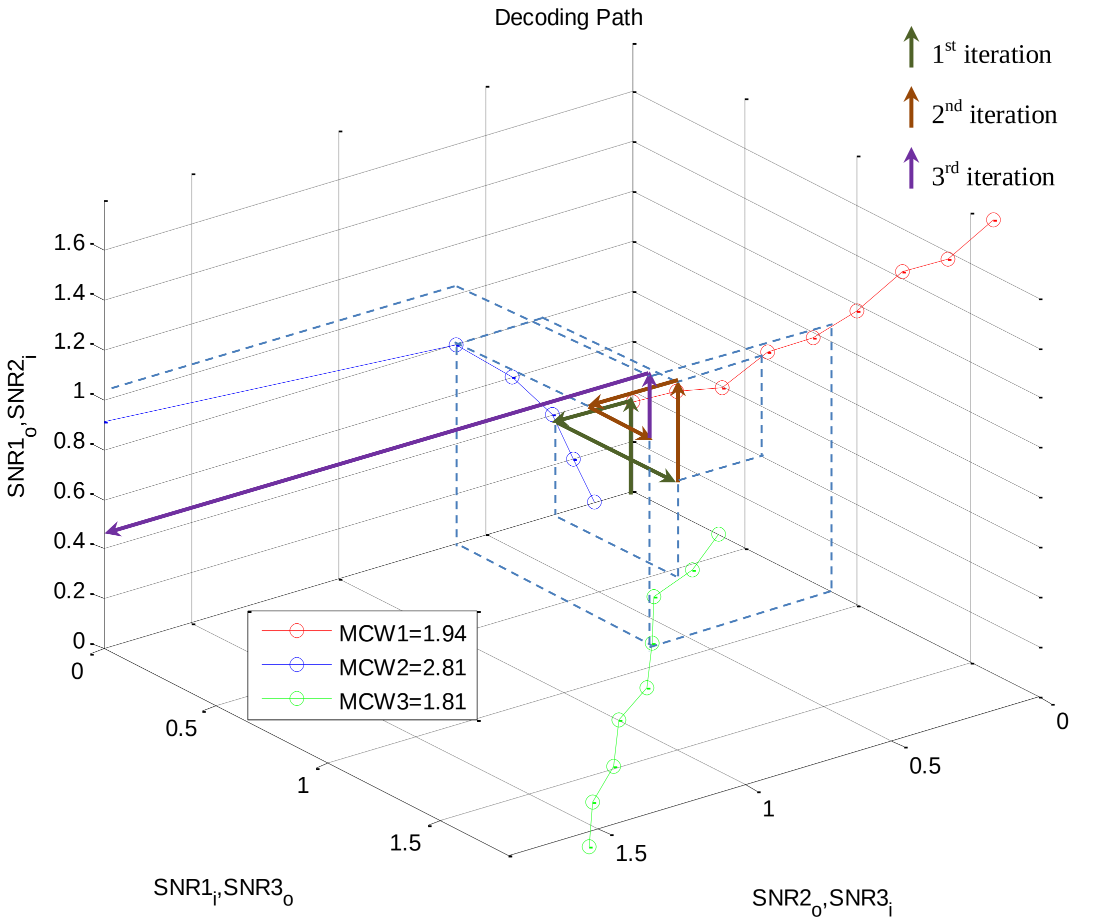


Figure 4.6 Decoding path between three different MCWs in MPCGCs at E_b/N_0=0.5dB

4.3 Complexity Comparison

As the probability information go back and forth along the edges between variable nodes and check nodes in tanner graph to iterative updating, the complexity of iterative decoding can be measured by the density of parity-check matrix (in terms of edges). To compare the complexity of MPCGC with LDPC, first computing the average number of iterations needed at different SNR levels. For MPCGC, each component decoder performs 38 local iterations before passing the extrinsic output to the other component decoders, and there are 30 super iterations. During each super iteration, the 38 local iterations are done by three component decoders, to make a fair comparison, the number of iteration of LDPC can be calculated by:

$$I = I_s \times (N_d \times I_l)$$

Here the local iteration I_l and super iteration I_s equals 38 and 30 respectively, and the component decoder N_d is 3, then the iterative number I for LDPC should be 3420.

To represent the decoding complexity per iteration, counting the edges based on the code length and the density of parity-check matrix (in terms of MCW). The complexity of LDPC here is 768*2.82=2166 edges, and for some code length and code rate MPCGC with different MCWs, there are 384*(1.9+2.82+1.79)*38=2500 edges in each super iteration. Table 4.6 illustrates the comparison of complexity in different condition of SNRs. The edges there is the total complexity for the whole super iteration in MPCGC and whole LDPC iteration.

	Iteration		**Total Edges**	
SNR	LDPC	MPCGC	LDPC	MPCGC
0	3.42e+3	3.42e+3	7.41e+6	2.85e+6
0.5	3.39e+3	3.08e+3	7.34e+6	2.56e+6
1	9.31e+2	9.18e+2	2.02e+6	7.65e+5
1.5	4.25e+2	5.40e+2	9.20e+5	4.50e+5
2	1.83e+2	3.60e+2	3.96e+5	3.00e+5

Table 4.1 Complexity comparison between LDPC and MPCGC

The result shows that in a low to moderate SNR area, the coding iteration and message passing edges of LDPC are more complex than MPCGC, while with the increasing of SNR, complexity for both of them are decrease and the LDPC decreasing faster than MPCGC. The decreasing tendency of complexity is illustrated in Figure 4.7. Predicting with this tendency, the number of MPCGCs edges will go beyond LDPC's in relatively high SNR area.

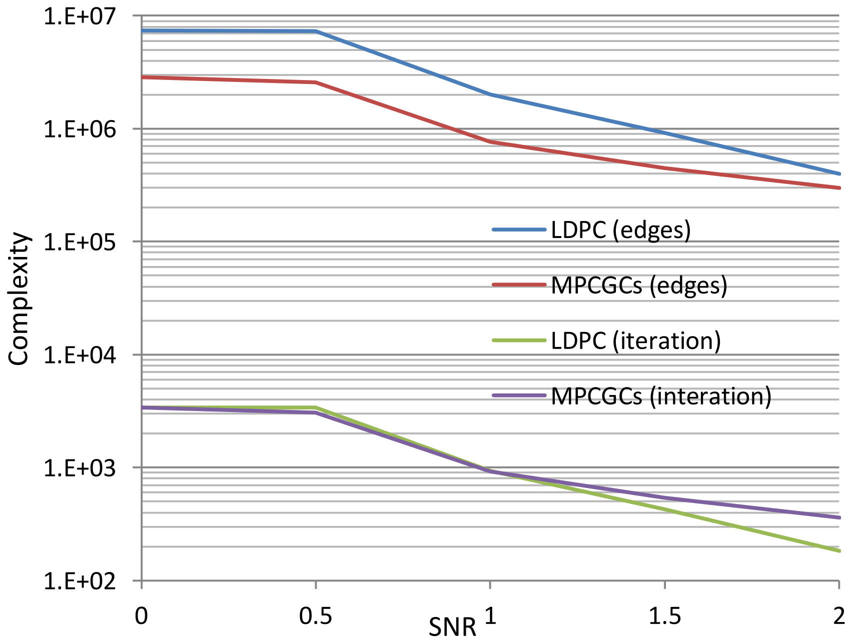


Figure 4.7 Complexity comparison between LDPC and MPCGC

Chapter 5

Proposed Decoder Architecture

To meet the needs of multi-Gb/s wireless application, there already lots of architectures which based on the fully parallel and fixed decoder are developed, due to the fixed decoder is not available for different coding parameters, another scalable and reconfigurable decoder architecture is proposed in [30], [31], it is more flexibility and still got a good performance in high-throughput. The decoder architecture, in this case, is based on the idea of Quasi-Cyclic LDPC decoder, this chapter discusses the strategies for optimizing the parallel decoder architecture, the modified features and other decoder parameters.

5.1 Techniques for High Throughput.

For a better understanding of the high-throughput requirements for MPCGCs decoder, we can define the iterative MPCGCs decoding throughput T as follow:

$$T = \frac{n * r * F_c}{N_c[N_{si}(N_i + 1) - 1]} \quad (b/s)$$

Here, n is the code length and r is the code rate, F_c is the Top-level clock frequency, N_i is the number of local iterations and N_{si} is the number of super iterations, N_c is the number of clock cycles for each local iteration. The n, r, N_i and N_{si} are all depends on the parameter of transmit code and the decoding algorithm methods, the Fc and N_c are determined by the hardware architecture.

The optimization of architecture is looking for a higher decoder throughput with a relatively lower resource utilization, which means that the optimized architecture could operate at higher clock frequency and have minimal latency between each iteration. Although the fully parallel architecture could achieving the highest throughput, the maximum throughput cannot be always accepted by application, compare to the fully parallel, layered and serial-parallel architecture, they show the different advantages in different aspects.

5.2 Parity Check Matrix Structure

5.2.1 Quasi-Cyclic Matrix

Although random parity check matrix with different column weight can achieve a good decoding performance, it needs amount of hardware consumption doing different number of check nodes and variable nodes in each column and row respectively, due to the irregular position of '1's in matrix. Instead of random matrix, the matrix in constant column weight can simplify the hardware structure but in sacrifice the bit error rate performance, to have a different column weight without increase too much hardware consumption, the quasi-cyclic matrix gives flexibility of matrix have different column weight in each column and more easily to manage the MCW in total.

The size of parity check matrix in every component decoder is (192,384) in this case study. Separating the whole matrix into several parts, we call them sub-matrix, the size of them are the same z*z, here z=24. They are either cyclic permuted from the identity matrix with different shift amount or all-zero sub-matrices. Figure 5.1 shows the example of shift amount 0,1 and 22 (the superscript gives shift amount), the identity matrix shift to right cyclically according to the shift amount gives.

$$\beta^0 = \begin{bmatrix} 1 & 0 & 0 & & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & & 0 & 0 & 1 \end{bmatrix}$$

$$\beta^1 = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ & \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 \\ 1 & 0 & 0 & & 0 & 0 & 0 \end{bmatrix} \qquad \beta^{22} = \begin{bmatrix} 0 & 0 & 0 & & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 \\ 1 & 0 & 0 & & 0 & 0 & 0 \\ & \ddots & & & & \vdots & \\ 0 & 0 & 1 & & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & & 1 & 0 & 0 \end{bmatrix}$$

Figure 5.1 Example of cyclically right shifted identity matrices

5.2.2 Non-Overlapping Layers

Although the fully parallel implementation may seem as an attractive option for achieving high-throughput performance, it has its own drawbacks, firstly, it becomes quickly intractable in hardware due to the complex interconnect pattern between CN units and VN units, secondly, such an implementation usually restricts itself to a specific code structure. In spite of the serial nature of the algorithm which mentioned before (scaled min-sum), one can process multiple nodes at the same time if the following condition is satisfied.

From the perspective of CN processing, two or more CNs can be processed at the same time (i.e. they are independent of each other) if they do not have one or more VNs (code bits) in common. The row-layering technique used in this work essentially relies on the above condition being satisfied. In terms of H, an arbitrary subset of rows can be processed at the same time provided that, no two or more rows have a 1 in the same column of H. This subset of rows is termed as a row-layer.

Figure 5.2 illustrate the parity check matrix that creates with several sub-matrices, these sub-matrices are either quasi-cyclic matrix or null matrix (zero matrix), we can control the MCW flexibly with this layered structure. The number in boxes are the shift amount and the blank boxes are zero matrces, no matter how much the quasi-cyclic matrix shift, the column weight always be 1, calculating by the equation (4.1), the MCW here is 3.25.

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16
L1	21		16		12		23		1							
L2	5		7		18			14	9	20						
L3		4		3		16		6		4	13					
L4		12		19		10	5				14	6				
L5	19		17		14		13		22			15	18			
L6	8		5			20		2		18		3		4		
L7		22		11		6		19			20			23	7	
L8		7		20	15		21		2				12		19	11

Figure 5.2 sub-matrices distribution in parity check matrix

Obviously, the layer 1 and layer 3 here are independent of each other, they do not have any VN units in common, it is the same to L6 with L8, L2 with L4 and L5 with L7, In this case, the matrix can overlapping as the figure 5.3 shown below, it reduce the layers from 8 to 4 super-layers.

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16
L'1	21	4	16	3	12	16	23	6	1	4	13					
L'2	5	12	7	19	18	10	5	14	9	20	14	6				
L'3	19	22	17	11	14	6	13	19	22		20	15	18	23	7	
L'4	8	7	5	20	15	20	21	2	2	18		3	12	4	19	11

Figure 5.3 Structure of matrix with non-overlapping layers

The component decoder processing 16 sub-matrix blocks for each layers and processing 4 blocks for each columns with this non-overlapping structure.

5.3 Component Decoder Design

The MPCGC decoder in this scenario has three component decoders, each one has its own parity check matrix in the same dimension with each other. Take one decoder for example, as mentioned before, the parity check matrix with overlapping layers can modify the check node architecture, the component decoder here has z Check Node Units (CNs), z is the dimension of sub-matrix Hs, each CN has N inputs, which N is the number of columns in Hs, and there are N*z Variable Node Units (VNs) that are combined into N groups of z VNs in each group, and each VN in this group connects to one port of a z-input barrel shifter to complete the cyclically right shift in a specific number, and all the outputs are further routed in pre-routers and then connect to the CNs. During the super iteration, all of the VNs in each component decoder will exchange the extrinsic information with the others in Extrinsic VtoC Updating Unit, and get back the updated a priori information which going to the CNs. Figure 5.4 shows the block connection of one component decoder.

The decoder architecture in this scenario is based on the layered property of the parity check matrix and implements with the Modified Min-Sum algorithm.

z: submatrix size

N: numbers of the columns in sub-matrix B

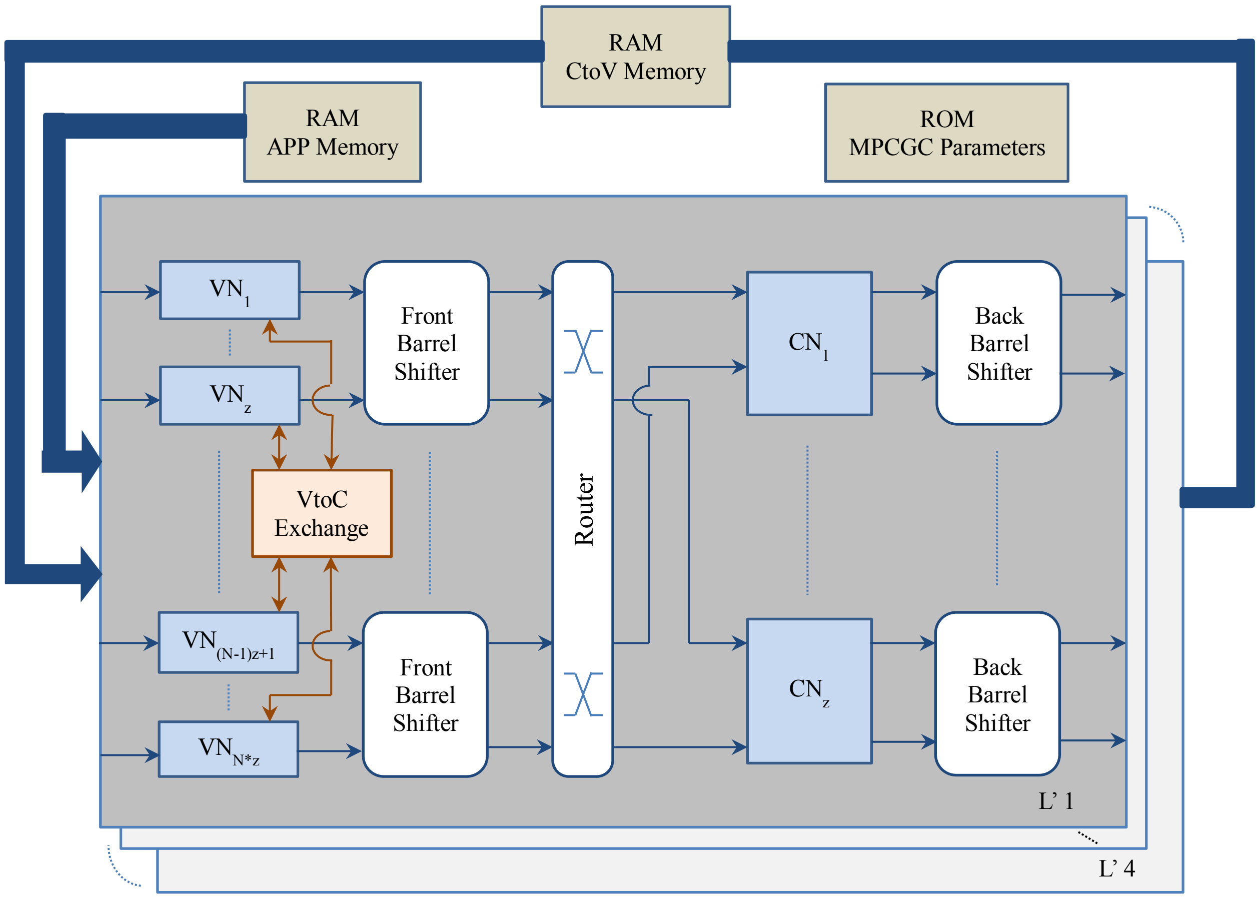


Figure 5.4 Component decoder architecture for MPCGC

5.3.1 Check Nodes Unit

To reduce the complexity of processing the minimum value in equation (2.5) of Check Node (CN) Unit, here divide the process into two phases, one is in the check node step that selects two smallest values (first minimum and second minimum value) in the set instead of just output one minimum value. Another one is during the variable node step which will discuss later. This modification reduces the complexity from quadratic complexity $O(n_{ci}^2)$ to linear complexity $O(n_{ci})$. A similar method is also found in [32].

As explained in Chapter 2, the CNs are processing the equation (2.5) which has the most computational complexity due to the calculation of the minimum magnitude and the sign for all the neighboring VNs, in order to allow CN processing two non-overlapping layers, the comparator tree have two outputs, one is a pair of first and second minimum from the bottom-half of the tree and the other pair of minimum is select between the top-half tree's and the final stage's, the block diagram of the CN's magnitude comparator tree is shown in figure 5.5.

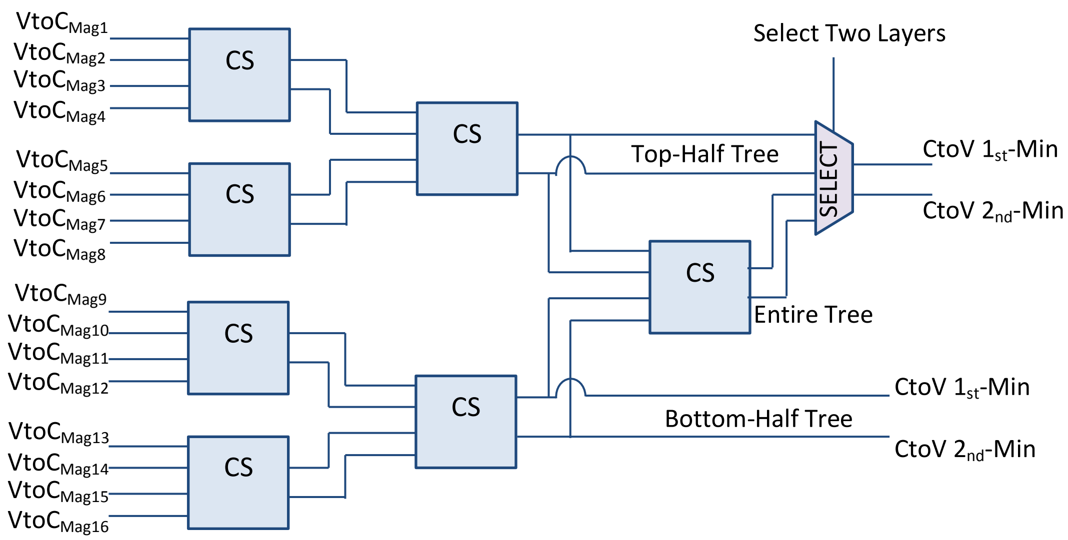


Figure 5.5 Message Magnitude Computation in Check Nodes Unit

The magnitude comparator tree constitute mainly of several compare-select (CS) blocks which compare four inputs and select the first and second minimum value. Figure 5.6 is the graph function of CS blocks in LabVIEW.

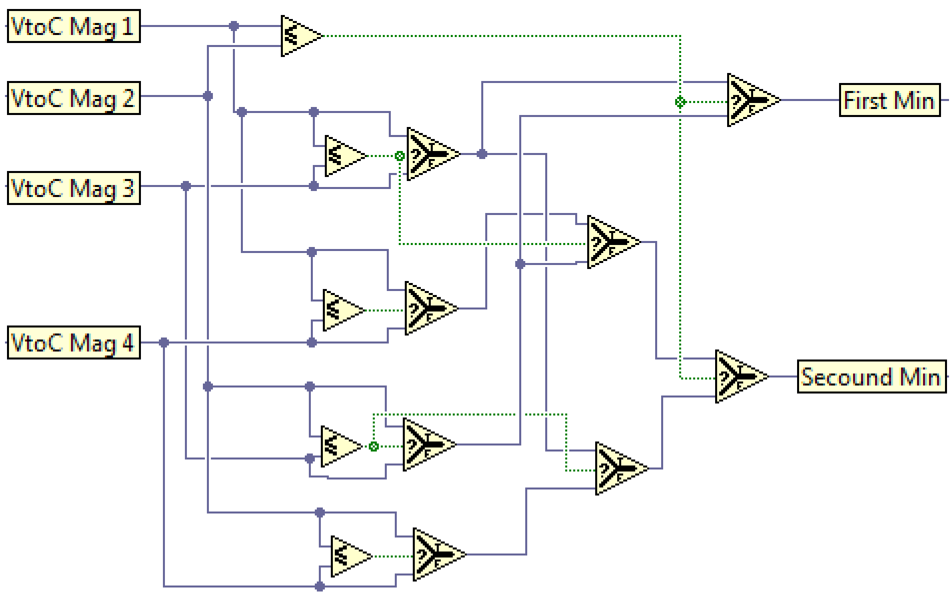


Figure 5.6 Compare-Select Blocks

Another computation tree for the inputs' sign processing the first half part of equation (2.5), this constitute by XOR gates as shown in figure 5.7,

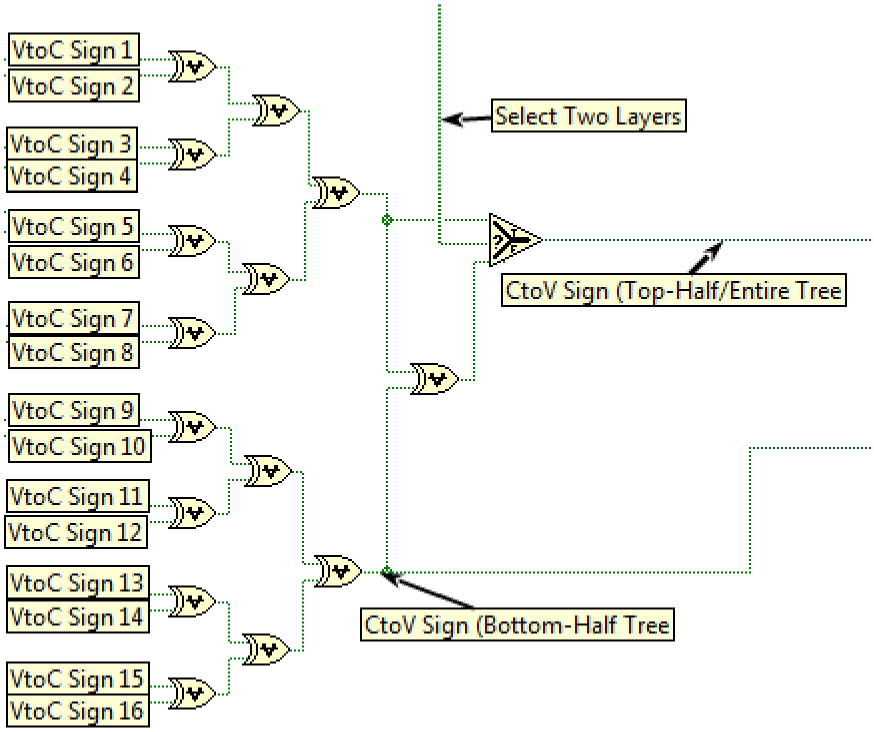


Figure 5.7 Message Sign Computation in Check Nodes Unit

5.3.2 Variable Node Unit

All the VNs received a prior information with both first and second minimum from previous steps, and loads them in accumulation registers, then outputs the variable to check (VtoC) information to all its neighboring CNs in sign-magnitude format. During the performance in VNU, comparing the first minimum magnitude to the VtoC magnitude in the register to marginalize the magnitude. VNU will select the first minimum magnitude in next add tree part if VtoC not equal to first minimum, or will select the second minimum magnitude if they are equal. Let $f^{(t)}$ and $s^{(t)}$ denote the value of first and second minimum at time t respectively, $L_{i'j}^{(t)sel}$ will be the chosen minimum value of $L_{i'j}^{(t)}$ in equation (5.1) of VN Unit.

$$L_{i'j}^{(t)sel} = \begin{cases} s^{(t)}, & \left|L_{i'j}^{(t)}\right| = f^{(t)} \\ f^{(t)}, & else \end{cases} \tag{5.1}$$

Then the value going through the accumulation part as shown in figure 5.8, adding with the priori information together,

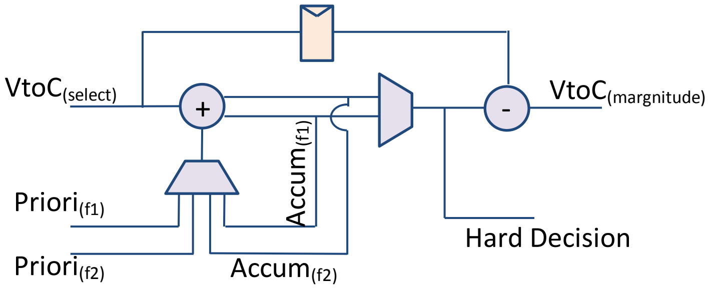


Figure 5.8 Architecture of Magnitude Accumulation in Variable Nodes Unit

5.3.3 Barrel Shifters

In order to simplify and reduce the wiring, VN Units and CN Units connect to barrel shifters to implement the circular shifters required by each sub-matrix,

Barrel shifter rearranges the sub-matrix elements from their original order to a uniform order according to each shifter number it is, it seems like change each sub-matrix to an identity matrix. After computation of all CNs, the barrel shifter changing the order back to the original order for each layer.

5.3.4 VtoC and CtoV Router

Here is a serial processing by each non-overlapping layer, there are z CNs grouped together in a single layer which called Check Node Group (CNG), and all the VNs divided into N groups according to the sub-matrices they belong to. Since the messages passing between CNG and VNG in each non-overlapping layer are coming from at most two layers, the routers between them must ensure the massages from the first layer and second layer go to the top half and the bottom half of each CN block respectively. Each Barrel Shifter connect with z VNs in the same group, and output the massages through VtoC Router to connect each CN correctly, and then back to the VNGs through CtoV Router and Back Shifter, figure 5.9 shows the function in a section of the matrix.

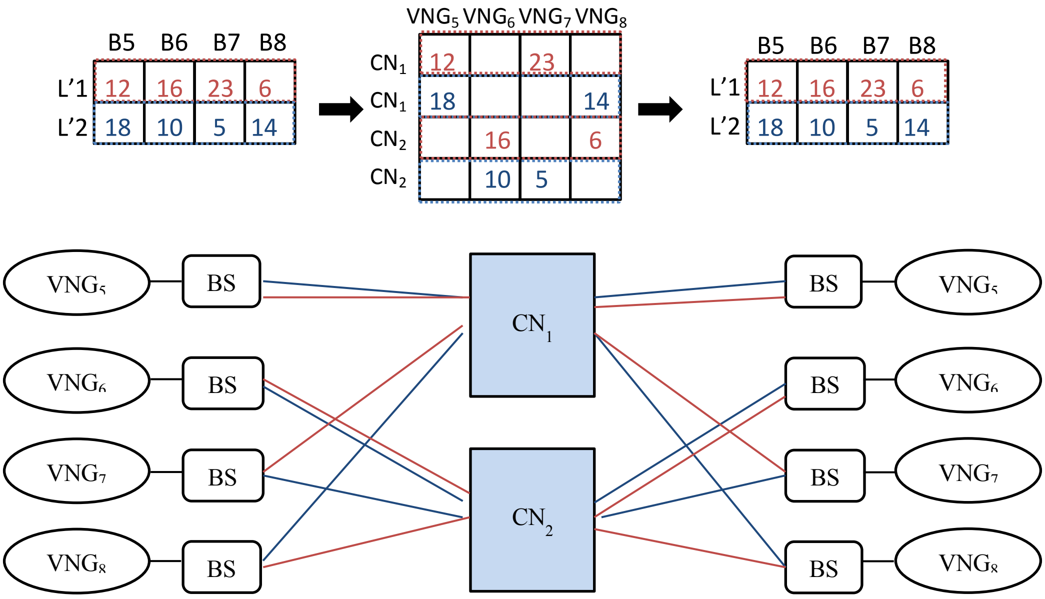


Figure 5.9 VtoC and CtoV Router function

5.3.5 Extrinsic VtoC Updating Unit

This block is outside the component decoder structure which connecting all three decoders, it is only executing when super iteration occurred, all of VNs in each component decoder will exchange the extrinsic information F_l as a priori information through this VtoC exchanging unit, and the new information is going to the CNs in corresponding component decoders.

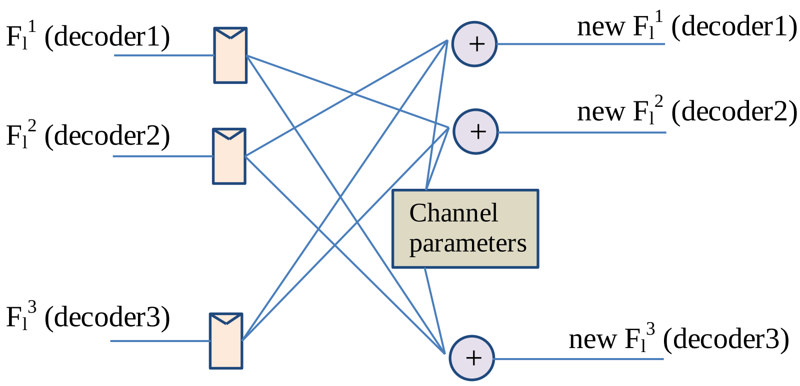


Figure 5.10 Structure of Extrinsic VtoC Updating Unit

5.4 Pipeline Architecture

As we discussed the non-overlapping layers before, all the CNs in a group are processed in parallel to compute one non-overlapping layer. And for each layer it can be only processed in serial since they are dependent on each other for some block columns, it requires enough clock cycles (as same as the number of non-overlapping layers) to time multiplex the serial CNG. Each VN needs to receive message from all layers in its column to compute VtoC extrinsic message for next iteration, it means that until the last message from the last layer to be received and stored into memory in VN, it is impossible to calculate the new message to CNG for the first block row operating, it will results in a delay for waiting all layers executed and output the final message before starting a new loop. The non-pipelined case in figure 5.11(A) shows clearly that, dependent layers limit the degree of parallelization and the idle clock cycles impact on achieving high throughput directly.

To maximize the effectiveness of the component decoder, let us first understand the layer-to-layer independence condition. Within a non-overlapped super-layer, when VNs process messages for the block b'(u,v), the CNs can process messages for the block b'(u+1,v) at the same time, u={1,2,…,L'}, and v={1,2,…,N}. The figure 5.11(B) shows how the pipeline works only in the local iteration, each β (u,w) block including all the b'(u,v) blocks in the same super-layer which execute in parallel, w is the number of local iteration. Compare to the non-pipeline one, the pipeline architecture reduces three clock cycles per iteration.

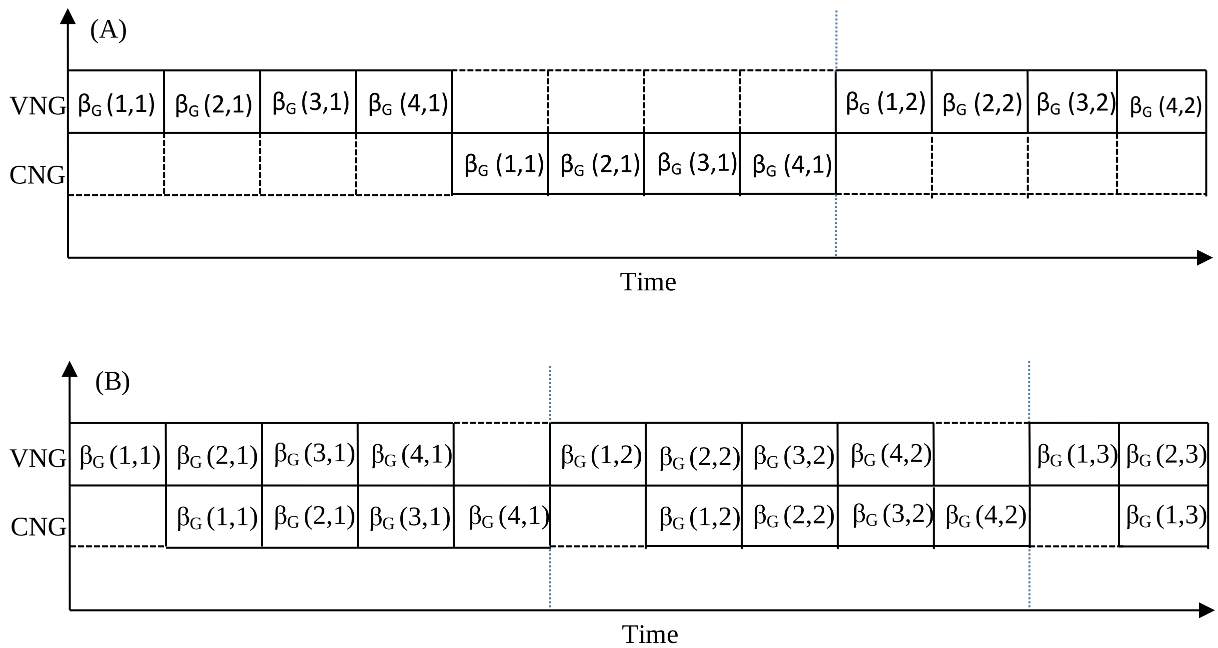


Figure 5.11 Pipeline timing diagram in local-iteration

(A) Non-pipeline case of one component decoder. (B) Pipeline case of one component decoder.

For MPCGC decoder, exchanging the messages output from VNG is required during the super-iteration, each component decoder passing the posterior probability after decision step to the other decoders, and get back the updated message which is going to CNG. The extrinsic VtoC updating unit cannot compute the new messages until received messages from all component decoders, this also brings a large amount of idle cycles when super-iteration occurred, shown as Figure 5.12 (A). The pipeline technique here is similar to the inner pipeline of component decoder, all the b'(u,v) blocks in one super-layer pass VtoC messages to the updating unit in parallel, it is same for messages from the other two component decoders and the updated VtoC messages $newF_l^1$ $newF_l^2$ and $newF_l^3$ are send back to CNG respectively in the next clock cycle. Figure 5.12 (B) illustrate that pipeline reduces half the clock cycles per super-iteration.

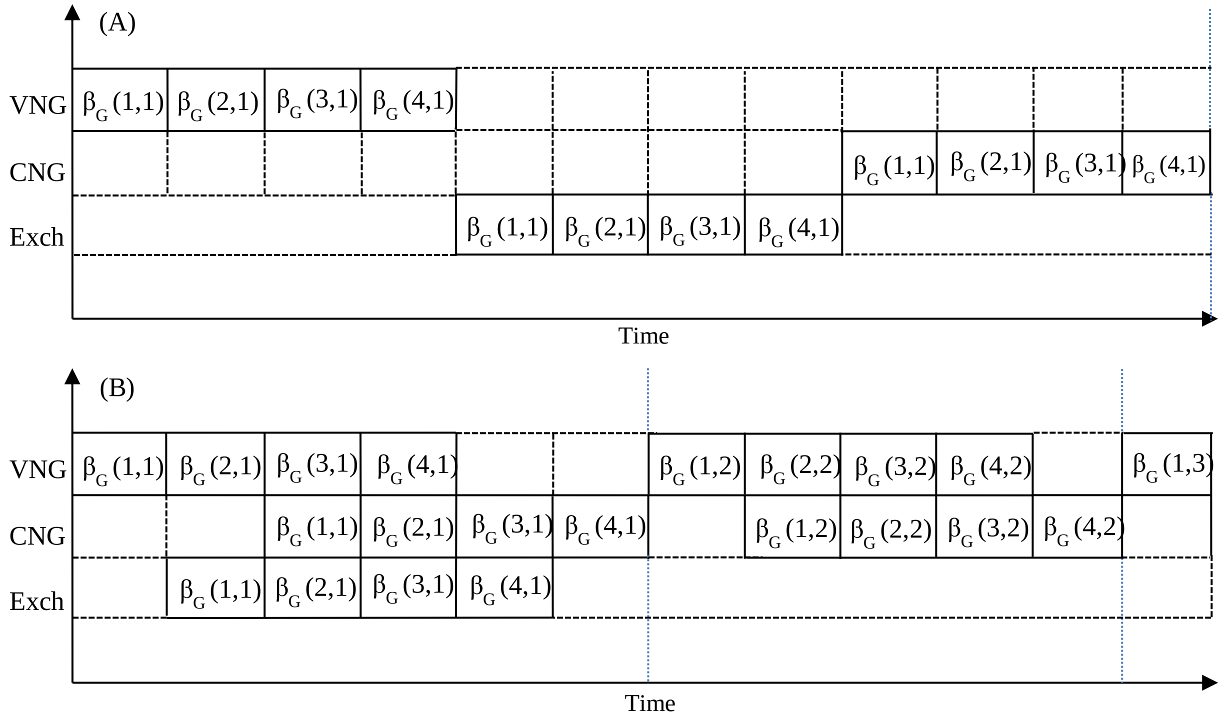


Figure 5.12 Pipeline timing diagram in super-iteration

(A) Non-pipeline case of one component decoder. (B) Pipeline case of one component decoder.

To make sure this pipeline design for super-iteration works well, the dimension and MCW of parity check matrix in each component decoder will be adapted, especially for the size of sub-matrix and the arrangement, so this pipeline design did not suitable for the parity check matrices in different number of super-layers.

The pipeline design here allowing us to overcome the layer dependency and improve throughput, although there are some idle cycles at the end of each iteration, the number of the clock cycles are much less than the non-pipeline structure.

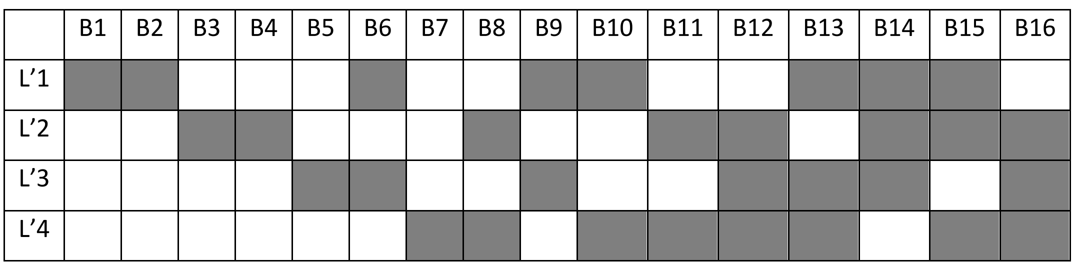
Chapter 6

LabVIEW^{TM} Simulation and Compilation

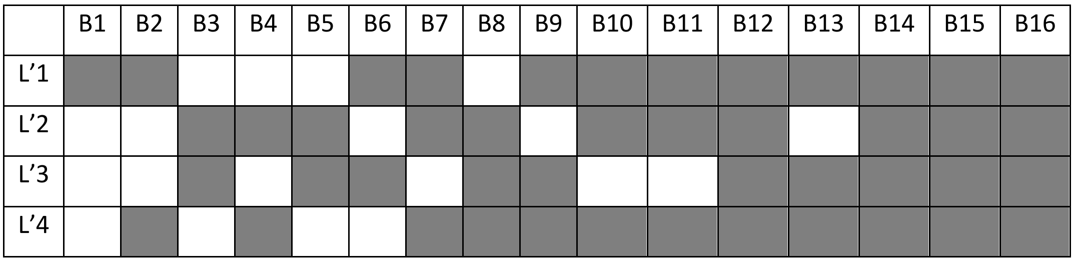
As a graphical programming language, National Instruments' LabVIEW with its Modulation Toolkit and virtual instruments (VIs) makes it easy to design and produce a flexible and powerful communication test system, especially for simple applications, such as quick lab tests or monitoring applications. It is in contrast to the sequential logic of most text-based programming languages that LabVIEW stands out, the executability and execution of graphical block dependents on the dataflow architecture, which means that the output data sending to all other connected blocks while all input data are available. And for large development project, it is more intuitive to plan the project and check the error with graphical functions in LabVIEW.

6.1 Simulation of BER Performance

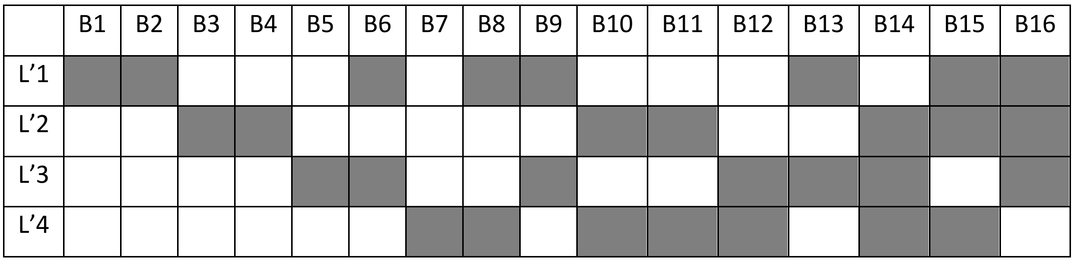
The decoder implementation based on the pipelined architecture which mentioned in chapter 5, each component decoder operating for 192*384 parity check matrix with z=24 sub-matrix and code rate R=1/2. The MCWs of each matrix are different and controlled by the sub-matrix arrangement, we select a combination of MCW which has a good performance in test before, MCW1=1.94, MCW2=2.81 and MCW3=1.81, the sub-matrix arrangement is shown in figure 6.1 that the grey boxes are Quasi-Cyclic matrix in different shift amount.



(a) MCW1=1.94



(b) MCW2=2.81



(c) MCW3=1.81

Figure 6.1 Sub-matrix layout in parity check matrix

Compare to the Bit Error Rate performance of MPCGC (with three component decoders) in different parameters. We select several sets with different number of local- and super-iteration to compare in LabVIEW, the performance in Figure 6.2 shows that the BER performance is more based on the number of super-iteration, one more super-iteration would have much more improvement. As shown in Figure 6.3 there is slightly different with blue line (8 local-iteration, 3 super-iteration) and red line (13 local-iteration, 3 super-iteration), the total iteration number for red line is 13*3=39, to compare with it, a 8 local-iteration and 5 super-iteration (white) case which have nearly same iteration number 8*5=40 been chosen, it shows better performance to red line. The performance of green line in Figure 6.3 is better than others but it needs much more clock cycles for the huge number of iteration, compare to the total number of clock cycles they needed, we choose the blue line case here which needs less iteration number for the relatively acceptable BER performance and compile it in LabVIEW which discussed in next paragraph.

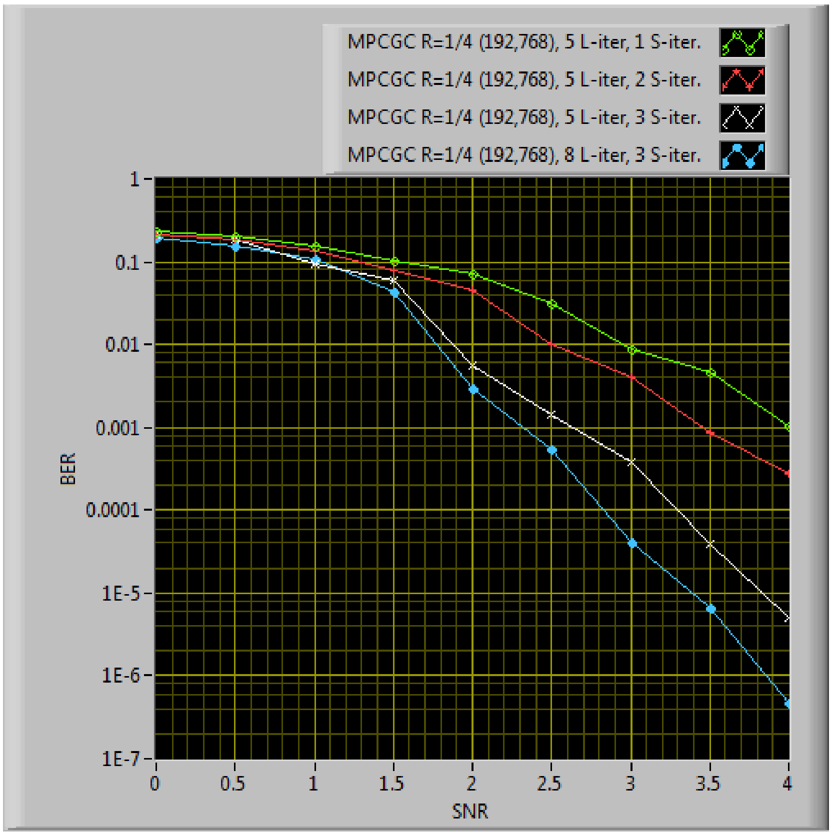


Figure 6.2 MPCGC Bit Error Rate (BER) performance comparison between 5 local-iteration, 1 super-iteration (green); 5 local-iteration, 2 super-iteration (red); 5 local-iteration, 3 super-iteration (white); 8 local-iteration, 3 super-iteration (blue).

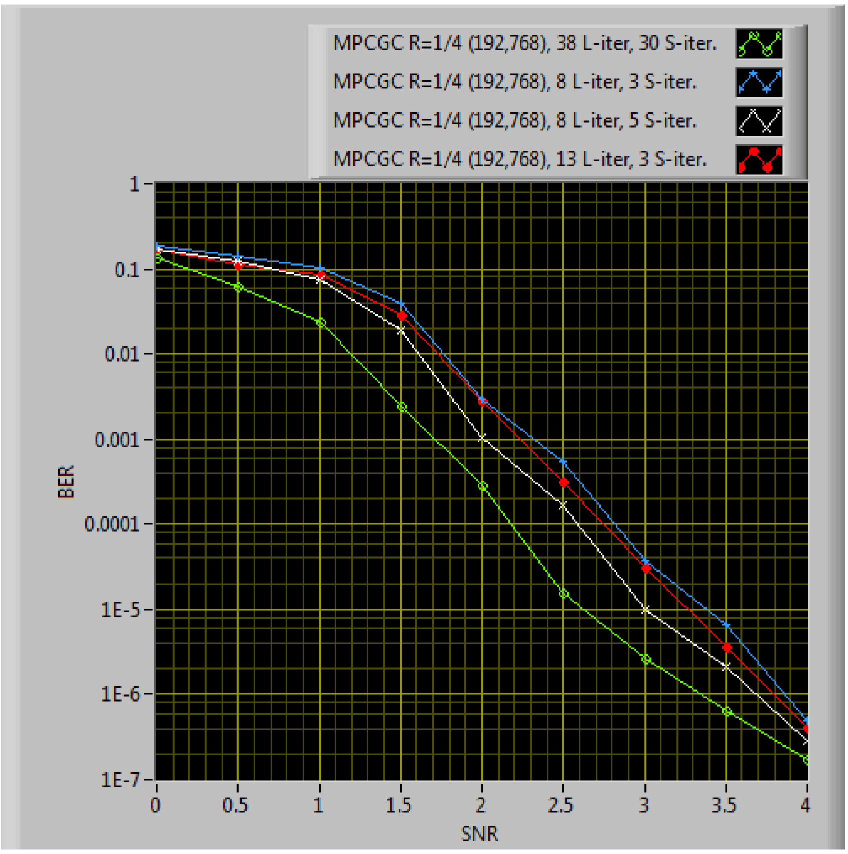


Figure 6.3 MPCGC Bit Error Rate (BER) performance comparison between 8 local-iteration, 3 super-iteration (blue); 8 local-iteration, 5 super-iteration (white); 13 local-iteration, 3 super-iteration (red); 38 local-iteration, 30 super-iteration (green).

6.2 LabVIEW™ FPGA Compilation

In previous works, the IEEE802.11ad standards required multi-gigabit LDPC decoders, in order to achieve high throughput and low power consumption, many types of architecture are used for the decoder. Because of the MPCGC decoder is combined with multi LDPC decoders, to find an optimal architecture for LDPC decoder which can achieve a good performance is helpful to the optimization of MPCGC decoder.

As the LabVIEW™ FPGA and can compile the design in a very fast way with a high-level language graphical code in it instead of the hardware description language, while the real-time operating system increases the PC's usefulness by limiting the variation in latency. Here compiling one component decoder on the Xilinx Kintex-7K160T FPGA which available on the NI cRIO-9033 FPGA board, it can estimate the FPGA resource utilization very quickly, here is only 39min. And the decoding core achieves throughput of at an operating frequency 120MHz. Table 6.1 shows the resource utilization for only one component decoder, for the whole decoding system there are three decoders processing in parallel, the resource may triple than it due to the Extrinsic VtoC Updating Unit do not take many logic gates and slices. The throughput achieved for one component decoder is T=576Mb/s.

Device	**Kintex-7K160T**	
Clock Frequency	120MHz	
Time Compile	39min	
	usage amount	(percentage of total)
Total Slice	5754	(25.1%)
Slice Register	46122	(22.7%)
Slice LUTs	39855	(39.3%)
Block RAMs	32	(9.8%)
DSP48s	41	(6.8%)

Table 6.1 MPCGC component decoder IP FPGA resource utilization on Kintex-7K160T

6.3 Memory Cost Estimation

The MPCGC decoder will store three type of messages during the decoding process, the initial LLR values with calculated from the channel parameters, the extrinsic CtoV messages which are updated in each iteration including the super iterations, the hard decision messages which used for the early convergence.

The bit width for one LLR value is 6 bits, the memory size of the initial variable messages and hard decision messages are 6912 here (384*6*3=6912). But the memory size of extrinsic CtoV messages are different for different decoding methods and structures, here we discuss the total memory size of extrinsic messages in this scenario.

	MPCGC			**LDPC**		
Rate	**Z**	**L(Com)**	**Memory**	**Z**	**L**	**Memory**
1/4	24	8(3)	27,648	24	24	55,296
1/3	24	8(2)	18,432	24	16	39,168
2/5	24	4(3)	10,368	24	12	31.680
1/2	16	4(3)	9,216	16	12	13,824
3/4	16	2(2)	8,064	16	4	3,072
8/9	4	2(3)	7,200	8	3	2,592

Table 6.2 Comparison of the memory size with MPCGC and LDPC

It is obviously in Table 6.2 that both the memory size is reducing while the code rate increasing, the MPCGC have an advantage in the lower code rate especially when the rate is less than 1/2, that because MPCGC just divides the systematic information into several parts for each component decoders while remaining the redundant information. The proportion of parity information in the total information is rising while the increasing of code rate, then the memory size gap between MPCGC and LDPC will increase highly with the high code rate. The MPCGC code is more suitable for lower code rate situation such as the rate 1/4 in DVB-S2 standard, it has less advantage on higher code rate especially higher than 1/2.

Chapter 7

Conclusion

7.1 Advances

This thesis was concerned with investigating a new approach to coding and decoding LDPC codes based on Parallel Concatenated Gallager Code (PCGCs) using multiple constituent codes (MPCGC). These are a class of concatenated codes built from the direct parallel concatenation of LDPC codes without interleavers. They are characterized by a competitive BER performance while still maintaining the low complexity and flexibility attributes.

Encoding and decoding techniques for these codes have been considered and BER results obtained confirming the improved performance of these codes. Further analysis in terms of the number of constituent codes and complexity has also been given.

A key novelty of the work in this thesis is to propose a highly-parallelized and pipelined MPCGC decoder based on the QC-LDPC structure, which is suitable for multi-Gb/s wireless network, this design takes advantage of layered structure to intelligently distribute the hardware resources, and also fully pipelined. The simulation result supported by preliminary FPGA implementation of the constituent sub-blocks using LabVIEW FPGA IP tools indicate that the proposed coding structure supports high throughput with slight sacrifice in the resources consumption. This, to my knowledge, is the first work to consider implementation issues of MPCGC codes.

7.2 Future work

Although this work proposed an architecture which is a near-optimal design, the test of actual application with this structure is still required. Firstly, completing the whole MPCGCs decoding system to find how much improvements in throughput and hardware resource utilization.

1. To meet the requirement for the development of wireless communication, high-throughput coding method received attention in the research community, then increasing

the operating frequency and pipeline depth to process extra frames simultaneously becomes crucial in next work.

2. The amount of previous works shown the good performance of non-binary LDPC codes in wireless channels compare with the binary codes [33], such as the multiple-input multiple-output (MIMO) transmission which is one of the most practical methods to increase the capacity and against fading effects of wireless channels [34]. Because of non-binary LDPC codes can efficiently combat mixed types of noise and perform well in the presence of burst errors [35], next work can focus on the non-binary GF(q) (q>2) MPCGCs to increase the performance on error correcting.
3. As MPCGCs breaking a long complex code into multiple small codes that only breaking the redundant information of original long code, then it makes the message with relatively low code rate take more advantage of applying MPCGCs coding method. Apply the MPCGCs coding scheme to DVB-S2 with code rate 1/3 and 1/4 or even other popular IEEE standards which accept low code rate (less than 1/2) can be attempted in the future works.

References

[1] H. M. Behairy, and S. C. Chang, “Parallel concatenated Gallager codes,” Electronics Letters, vol. 36, no. 24, pp. 2025-2026, Nov. 2000

[2] IEEE Standard. (2010, May). IEEE draft standard for information technology-wireless LANs-part 21: mmWave PHY specification [Online]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=5407676.html

[3] Agilent Technologies. (2013, May). Wireless Lan at 60 GHz-IEEE 802.11ad Explained [Online]. Available: http://cp.literature.agilent.com/litweb/pdf/5990-9697EN.pdf

[4] IEEE Standard. (2009). IEEE standard for information technology-local and metropolitan area networks-part 12.2: Amendment: SC mmWave PHY mode-IEEE Std. 802.15.3c [Online]. Available: https://standards.ieee.org/findstds/standard/802.15.3c-2009.html

[5] DVB Blue Book. (2015, March). Digital Video Broadcasting (DVB) Implementation guidelines for the second generation system for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications [Online]. Available: https://www.dvb.org/resources/public/standards/a171-1_s2_guide.pdf

[6] R. W. Hamming, “Error Detecting and Error Correcting Codes,” Bell System Technical Journal, vol. 29, no. 2, Apr. 1950.

[7] R. G. Gallager, “Low-Density Parity-Check Codes,” Ph.D. thesis, Dept. of Electrical Engineering, Massachusetts Institute of technology, Cambridge, 1962.

[8] R. M. Tanner, “A recursive approach to low-complexity codes,” IEEE Transactions on Information Theory, vol. 27, no. 5, pp. 533-547, Sep. 1981.

[9] D. J. C. MacKay, and R. M. Neal, “Near Shannon Limit Performance of Low Density Parity Check Codes,” Electron. Lett, vol. 32, no. 18, pp. 1645-1646, 1996.

[10] D. J. C. MacKay, “Gallager Codes That Are Better Than Turbo Codes,” Proc. 36th Allerton Conf. Communication, control and Computing, Monticello, Sep. 1998.

[11] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, “Improved Low-Density Parity-Check Codes Using Irregular Graphs,” IEEE TRANSACTIONS ON INFORMATION THEORY, vol. 47, no. 2, pp. 585-598, Feb. 2001.

[12] S. Lin, and D. J. Costello, “Error Control Coding: Fundamentals and Applications,” Second Edition. Upper Saddle River. USA: Prentice Hall, May. 2004.

[13] M. P. C. Fossorier, “Quasi-Cyclic Low-Density Parity-Check Codes From Circulant Permutation Matrices,” IEEE TRANSACTIONS ON INFORMATION THEORY, vol. 50, no. 8, pp. 1788-1793, Aug. 2004.

[14] Z. Li, L. Chen, L. Zeng, S. Lin, and W. H. Fong, “Efficient Encoding of Quasi-Cyclic Low-Density Parity-Check Codes”, IEEE Transactions on Communications, vol. 54, no. 1, pp. 71-81, Jan. 2006.

[15] R. Echard, and C. Shih-Chun, “The π-rotation low-density parity check codes”, IEEE Globecom 2001, vol. 2, pp. 980–984. 25-29 Nov. 2001.

[16] S. Myung, K. Yang, and J. Kim, “Quasi-Cyclic LDPC Codes for Fast Encoding”, IEEE Transactions on Information Theory, vol. 51, no. 8, pp. 2894-2901, Aug. 2005.

[17] L. Lan, L. Zeng, Y. Y. Tai, L. Chen, S. Lin, and K. A. Ghaffar, ”Construction of QC LDPC Codes for AWGN and Binary Erasure Channels: A Finite Field Approach”, IEEE Transactions on information theory, vol. 53, no. 7, Jul. 2007.

[18] H. M. Behairy, “Dual Mode PCGCs For Advanced Wireless Communications Networks,” Proceedings of IEEE International Design and Test Workshop, IDT 2007, Dec. 16-18, 2007.

[19] C. E. Shannon, “A Mathematical Theory of Communication,” Bell System Technical journal, Jul. 1948.

[20] C. E. Shannon, “Certain results in coding theory for noisy channels,” Information and Control, vol. 1, pp. 6-25, Sep. 1957.

[21] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, “Near-Shannon-limit Quasi-Cyclic Low-Density Parity-Check Codes,” IEEE Transaction on Communications, vol. 52, no. 7, pp. 1038-1042, Jul. 2004.

[22] M. C. Davey and D. J. C. MacKay, “Low-density parity check codes over GF(q),” IEEE commun, Lett., vol. 2, no. 6, pp. 165-167, Jun. 1998.

[23] C. Spagnol and W. Marnane, “A class of quasi-cyclic LDPC codes over GF(2m),” IEEE Transactions on Communications, vol. 57, no. 9, pp. 2524-2527, Sep. 2009.

[24] W. Chang, and J.R. Cruz, “Performance and decoding complexity of nonbinary LDPC codes for magnetic recording,” IEEE Trans. Magn., vol. 44, no. 1, pp. 211-216, Jan. 2008.

[25] R. M. Tanner, “A recursive approach to low complexity codes,” IEEE Transactions on Information Theory, vol. 27, pp. 533-547, Sep. 1981.

[26] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.–Y. Hu, “Reduced-Complexity Decoding of LDPC Codes,” IEEE Transactions on Communications, vol. 53, no. 8, pp. 1288-1299, Sep. 2005.

[27] H. M. Behairy, and M. Benaissa, “Multiple Parallel Concatenated Gallager Codes: Code Design and Decoding Techniques,” IEEE Journal of Research, vol. 59, no. 6, Nov. 2013, pp. 659-664.

[28] H. Behairy, “Parallel Concatenated Gallager Codes and Their Applications in CDMA Wireless Networks,” George Mason University, Summer. 2002.

[29] D. Divsalar, S. Dolinar, and F. Pollara, “Iterative Turbo Decoder Analysis Based on Density Evolution,” IEEE Journal on Selected Areas in Communications, vol. 19, no. 5, pp. 891-907, May. 2001.

[30] S. Mhaske, H. Kee, T. Ly, A. Aziz, and P. Spasojevic, “Strategies for high-Throughput FPGA-based QC-LDPC Decoder Architecture,” May. 2015.

[31] S. Mhaske, D. Uliana, H. Kee, T. Ly, “A 2.48Gb/s FPGA-based QC-LDPC Decoder: An Algorithmic Compiler Implementation,” 2015 36th IEEE Samoff Symposium. Sep. 2015, pp. 88-93.

[32] Yang Sun, and J. R. Cavallaro, “VLSI Architecture for Layered Decoding of QC-LDPC Codes With High Circulant Weight,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 10, Oct. 2013, pp. 1960-1964.

[33] Y. L. Ueng, C. y. Leong, C. J. Yang, C. C. Cheng, K. H. Liao, and S. W. Chen, “An Efficient Layered Decoding Architecture for Nonbinary QC-LDPC codes,” IEEE Transaction on Circuits and Systems, vol. 59, no. 2, pp. 385-398, Feb. 2012.

[34] R. Peng, and R. R. Chen, “Application of Nonbinary LDPC Cycle Codes to MIMO Channels,” IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, vol. 7, no. 6, pp. 2020-2026, Jun. 2008.

[35] W. Chang and J. R. Cruz, “Performance and decoding complexity of nonbinary LDPC codes for magnetic recording,” IEEE Trans. Magn., vol. 44, no. 1, pp. 211–216, Jan. 2008

[36] E. Eleftheriou and S. Olcer, “Low-density parity-check codes for multilevel modulation,” in Proc. IEEE Int. Symp. Information Theory (ISIT2002), Lausanne, Switzerland, pp. 442, Jun./ Jul. 2002.

[37] C. Zhang, Z. Xiao, H. Wu, L. Zeng, and D. Jin. “Performance Analysis on the OFDM PHY of IEEE 802.11ad Standard,” IEEE Computational Problem-Solving (ICCP), pp. 708-713, Oct. 2011.

[38] X. Yang, L. Wang, and Y. Li. “Performance of Q-ary PCGC based on PEG algorithm,” Proceedings of IEEE Intelligent Signal Processing and Communication Systems, pp. 1-4, Nov. 28-Dec. 2007.

[39] A. Balatsoukas-Stimming, N. Preyss, A. Cevrero, and A. Burg, "A parallelized layered QC-LDPC decoder for IEEE 802.11ad," IEEE New Circuits and Systems Conference (NEWCAS), pp. 1-4, Jun. 2013.

[40] K. K. Gunnam, G. S. Choi, W. Wang, and M. B. Yeary, "A Parallel VLSI Architecture for Layered Decoding," 20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID'07), pp. 738-743, Jan. 2007.

[41] Y. Sun, G. Wang, and J. R. Cavallaro, "Multi-Layer Parallel Decoding Algorithm and VLSI Architecture for Quasi-Cyclic LDPC Codes," 2011 IEEE International Symposium of Circuits and Systems (ISCAS), pp. 1776-1779, May. 2011.

[42] C. Marchand, L. Conde-Canencia, and E. Boutillon, "Architecture and Finite Precision Optimization for Layered LDPC Decoders," Journal of Signal Processing Systems, vol. 65, pp. 185-197, Nov. 2011.