HIGH-PERFORMANCE DECODER ARCHITECTURES FOR LOW-DENSITY PARITY-CHECK CODES

by

Kai Zhang

A Dissertation
Submitted to the faculty of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the
Degree of Doctor of Philosophy

in

Electrical and Computer Engineering

by

August, 2011

APPROVED:

Prof. Xinming Huang Thesis Advisor Worcester Polytechnic Institute	Prof. Berk Sunar Thesis Committee Worcester Polytechnic Institute
Prof. James Duckworth Thesis Committee Worcester Polytechnic Institute	Dr. Zhongfeng Wang Thesis Committee Broadcom Corporation

Abstract

The Low-Density Parity-Check (LDPC) codes, which were invented by Gallager back in 1960s, have attracted considerable attentions recently. Compared with other error correction codes, LDPC codes are well suited for wireless, optical, and magnetic recording systems due to their near-Shannon-limit error-correcting capacity, high intrinsic parallelism and high-throughput potentials. With these remarkable characteristics, LDPC codes have been adopted in several recent communication standards such as 802.11n (Wi-Fi), 802.16e (WiMax), 802.15.3c (WPAN), DVB-S2 and CMMB.

This dissertation is devoted to exploring efficient VLSI architectures for high-performance LDPC decoders and LDPC-like detectors in sparse inter-symbol interference (ISI) channels. The performance of an LDPC decoder is mainly evaluated by area efficiency, error-correcting capability, throughput and rate flexibility. With this work we investigate tradeoffs between the four performance aspects and develop several decoder architectures to improve one or several performance aspects while maintaining acceptable values for other aspects.

Firstly, we present a high-throughput decoder design for the Quasi-Cyclic (QC) LDPC codes. Two new techniques are proposed for the first time, including parallel layered decoding architecture (PLDA) and critical path splitting. Parallel layered decoding architecture enables parallel processing for all layers by establishing dedicated message passing paths among them. The decoder avoids crossbar-based large interconnect network. Critical path splitting technique is based on articulate adjustment of the starting point of each layer to maximize the time intervals between adjacent layers, such that the critical path delay can be split into pipeline stages.

Furthermore, min-sum and loosely coupled algorithms are employed for area efficiency. As a case study, a rate-1/2 2304-bit irregular LDPC decoder is implemented using ASIC design in 90 nm CMOS process. The decoder can achieve an input throughput of 1.1 Gbps, that is, 3 or 4 times improvement over state-of-art LDPC decoders, while maintaining a comparable chip size of 2.9 mm^2.

Secondly, we present a high-throughput decoder architecture for rate-compatible (RC) LDPC codes which supports arbitrary code rates between the rate of mother code and 1. While the original PLDA is lack of rate flexibility, the problem is solved gracefully by incorporating the puncturing scheme. Simulation results show that our selected puncturing scheme only introduces the BER performance degradation of less than 0.2dB, compared with the dedicated codes for different rates specified in the IEEE 802.16e (WiMax) standard. Subsequently, PLDA is employed for high throughput decoder design. As a case study, a RC-LDPC decoder based on the rate-1/2 WiMax LDPC code is implemented in CMOS 90 nm process. The decoder can achieve an input throughput of 975 Mbps and supports any rate between 1/2 and 1.

Thirdly, we develop a low-complexity VLSI architecture and implementation for LDPC decoder used in China Multimedia Mobile Broadcasting (CMMB) systems. An area-efficient layered decoding architecture based on min-sum algorithm is incorporated in the design. A novel split-memory architecture is developed to efficiently handle the weight-2 submatrices that are rarely seen in conventional LDPC decoders. In addition, the check-node processing unit is highly optimized to minimize complexity and computing latency while facilitating a reconfigurable decoding core.

Finally, we propose an LDPC-decoder-like channel detector for sparse ISI channels using belief propagation (BP). The BP-based detection computationally depends on the number of nonzero interferers only and are thus more suited for sparse

ISI channels which are characterized by long delay but a small fraction of nonzero interferers. Layered decoding algorithm, which is popular in LDPC decoding, is also adopted in this paper. Simulation results show that the layered decoding doubles the convergence speed of the iterative belief propagation process. Exploring the special structure of the connections between the check nodes and the variable nodes on the factor graph, we propose an effective detector architecture for generic sparse ISI channels to facilitate the practical application of the proposed detection algorithm. The proposed architecture is also reconfigurable in order to switch flexible connections on the factor graph in the time-varying ISI channels.

Acknowledgements

First of all, I wish to thank my advisor, Professor Xinming Huang, for his guidance and support through all stages of my studies and research at the Worcester Polytechnic Institute. I am very grateful for his recognition, his inspiration, and the exposure and opportunities that I have received during the course of my study.

I am also indebted to to Professor Berk Sunar, Professor James Duckworth, and Dr. Zhongfeng Wang for their valuable supports as members of my thesis committee.

I am grateful to my fellow graduate students in the Embedded Computing Lab, Cao Liang, Wenxuan Guo, Yanjie Peng, Chen Shen and Wei Wang for their friendship and support.

I would also like to thank all the staff in ECE department, including Robert Brown, Catherine Emmerton, Colleen Sweeney, Brenda McDonald and Stacie Murray for their kind assistance and coordinations.

This thesis is dedicated to my wonderful family. I am forever grateful to my parents Shanfeng Zhang and Qingfang Zhao, my wife Wanning Jiang, for their love, support, and encouragement. I shall not try to put my appreciation and love for them into words.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter first introduces the history of error correction codes and LDPC codes and then describes the motivation, related works, contributions and outline of this dissertation.

1.1 Background

A major concern in designing reliable data transmission and storage systems is the control of data errors induced by a noisy channel or storage medium. In 1948, Shannon demonstrated that if the information rate is less than the channel capacity, introduced errors can be corrected by proper encoding and decoding of the information [63]. From then on, various types of error correction codes were designed by adding some redundancies to the original data (known as encoding), which the receiver can use to check consistency of the delivered message, and to recover data determined to be erroneous (known as decoding). Classical and widely used error correction codes include Hamming Code (1950), BCH Code (1959) [34, 8], Reed-Solomon Code (1960) [59], Convolutional Code (1955) [20, 74], Turbo Code (1993) [6] and LDPC Code (1962) [24].

Firstly discovered by Gallager in 1962 [24], LDPC codes were a class of Shannon-limit-approaching codes in which two innovative ideas were exploited: iterative decoding and constrained random code construction. However, LDPC codes were mostly neglected by code theorists for more than 30 years due to the tremendous computational efforts in implementing their encoder and decoder and the introduction of Reed-Solomon codes [59]. Shortly after the debut of another Shannon-limit-approaching code, Turbo code [6], in which both the above ideas are explored, LDPC codes were rediscovered by MacKay and Neal in 1996 [50]. LDPC codes have several advantages over Turbo codes: no error floors at high SNRs, more inherent decoding parallelism, higher throughput potentials, lower hardware complexity due to the elimination of long interleavers.

LDPC codes are specified by sparse parity check matrices $\mathbf{H}$ and can be fully represented by bipartite graphs (known as Tanner graph) with two sets of nodes, called the check nodes and the variable nodes. LDPC codes can be effectively decoded using the standard BP algorithm, also called sum-product algorithm (SPA). Two phases of messages, the check-to-variable (CTV) messages and the variable-to-check (VTC) messages, are transmitted along the edges of the Tanner graph [69] to update each other iteratively.

Good LDPC codes that have been found are largely long, random and computer generated codes with BP-based iterative decoding and can achieve an error performance as close as 0.0045 dB away from the theoretical Shannon limit [50, 49, 60, 61, 14]. In despite of excellent error performance, these codes are impractical for hardware implementation due to the following reasons: (1) they are very long (up to 100,000 bits) and thus a large memory is required to store the iterative messages, as well as the configuration of the $\mathbf{H}$ matrix; (2) their encoding circuitry would be very large in order to fulfill the complex matrix and vector

multiplications; (3) their large latency and lack of parallelism when decoding on a random Tanner graph; (4) randomly constructed LDPC codes also require a complex routing network, which not only consumes a large amount of chip area, but also significantly increases the computation delay.

In order to make LDPC codes hardware-friendly, structured LDPC codes that have elegant regularity in the structure of their **H** matrices and can provide comparable or even better error performance were introduced in by algebraic construction [40, 39, 75, 35, 73, 53, 19, 57]. It has been demonstrated in [48] that structured, algebraic-constructed codes can outperform random ones for medium length LDPC codes (up to a few thousand bits). Besides, their encoding can be simply implemented by linear shift registers. That is why LDPC codes are widely adopted by several recent communication standards such as 802.11n (Wi-Fi), 802.15.3c (WPAN) 802.16e (WiMax), DVB-S2, CMMB, 802.3an (10G-BaseT) and etc. Quasi-Cyclic (QC) LDPC codes, a special class of structured LDPC codes, decompose the large **H** matrix into small sub-matrices which are either identity matrix or its permutation [22, 43, 11]. Therefore, QC LDPC codes are best suited for hardware implementation since the regular structures of their **H** matrices make simple message switching and memory accessing possible.

1.2 Related Works

This dissertation is devoted to high-performance VLSI architecture design and implementation for LDPC codes. Generally, state-of-art LDPC decoder architectures can be divided into three categories: fully parallel method, serial method and partly parallel method.

Blanksby et al. [7] directly mapped the BP algorithm into hardware and imple-

mented a fully parallel irregular LDPC decoder. This method, through with high throughput, requires huge amount of interconnections to complete the messages passing within a random Tanner graph, leading to the average net length of 3nm and the total chip area of 52.5 mm^2. On the other side, Yeo et al. [80] designed a serial decoder by sharing the computation units and updating messages in sequential. Serial method results in very simple hardware architecture but low decoding throughput, which is usually unacceptable by real-world applications, due to the large latency of ten-thousand cycles needed for decoding a codeword.

Since structured LDPC codes are architecture-aware, their decoders can be implemented by partly parallel method to explore potential parallelism and improve the throughput [85, 12, 51, 86, 37, 87, 77, 36]. Compared with random LDPC codes, the **H** matrix of structured LDPC can be easily stored by storing only permutation values of each small sub-matrix instead of the entire **H** matrix. Also, regularity in the **H** matrix of structured LDPC codes enables easy control over the read/write operations of VTC and CTV messages when decoding by employing a simple bitwise counter, thus avoiding the huge interconnection network in [7]. QC LDPC decoders proposed in [77, 79, 78, 17, 66, 28, 45, 70, 9, 25, 27, 67] also belong to the partly parallel category.

1.3 Summary of Motivations and Contributions

This research is motivated by desires to explore high-performance LDPC decoder architectures in terms of area efficiency, error-correcting capability, throughput and rate flexibility. Although LDPC decoder has been an extensive research topic for almost a decade, there are still many unresolved problems. With this dissertation, we have considered existing VLSI design issues and developed various decoder ar-

chitectures to improve throughput, enhance flexibility and reduce complexity of an LDPC decoder.

The specific motivations for the work and the corresponding contributions will be summarized as follows.

1.3.1 Ultra-High-Throughput LDPC Decoder Architecture Design

Motivation: With the increasing demand for high-data-rate wireless applications, many recent communication systems employ ultra-high throughput channel codes to match the data-rate requirements. For example, 802.15.3c standard is targeted for the data rate of multi-giga bits per second (Gbps). However, conventional high-throughput LDPC decoders proposed in [7, 46, 54] were implemented by employing more hardware resources to operate at a higher parallelism, which consumed a large amount of chip area. For example, the decoder in [46] achieves an equivalent input throughput of 1 Gbps with a clock rate of only 200 MHz, an undoubtedly high parallelism of 512 and a large chip area of 14.5 mm^2 under 90 nm process. Thus it is desirable to investigate other methods to improve throughput.

Contribution: We proposed a parallel layered decoding architecture to achieve ultra high throughput. Parallel layered decoding architecture not only keeps the advantage of conventional layered decoding algorithm that can reduce the convergence required number of iterations by half and thus double the throughput, but also avoids the serial operations existing in conventional layered decoding algorithm which limits overall throughput. Two other techniques were also applied: critical path splitting and fixed message passing. The former optimizes the decoder's critical path and divides it into several pipelined stages. The latter avoids crossbar-based large interconnect network, which usually bottlenecks an LDPC decoder to achieve

high clock rate. Both techniques improve the clock rate and thus the throughput of an LDPC decoder without increasing parallelism and hardware resources. As a case study, a rate-1/2 2304-bit irregular LDPC decoder was implemented using ASIC design in 90 nm CMOS process. The decoder can achieve an input throughput of 1.1 Gbps, that is, 3 or 4 times improvement over state-of-art LDPC decoders, while maintaining a comparable chip size of 2.9 mm^2.

1.3.2 Flexible-Rate High-Throughput LDPC Decoder Architecture Design

Motivation: In wireless communication systems, it is desirable to adjust ECC code rate according to the channel state information (CSI) to meet various service requirements and channel conditions. For example, WiMax standard provides 6 different LDPC code patterns in 4 different rates (1/2,2/3A,2/3B,3/4A,3/4B,5/6) [4]. Although there have been some research works on the design of flexible rate LDPC decoders [68, 84, 52, 46, 66], none of them can provide arbitrary code rate. Moreover, these decoders usually employ complicated interconnection network to switch between different code patterns [84, 46, 66], such as Benes networks used in [52], which leads to signal congestion problem, large delay and low throughput. How to design a flexible rate LDPC decoder with high throughput remains challenging.

Contribution: With the dissertation we proposed a rate-compatible (RC) LDPC decoder architecture that can provide arbitrary code rate between the rate of the mother code and 1. Codes with flexible rates were generated by puncturing the parity bits of the mother code with negligible error performance degradation. Parallel layered decoding architecture was also adopted to achieve high throughput where complicated interconnect network was replaced by fixed wires. The proposed RC LDPC decoder also helps solve the rate inflexibility problem existing in previous

parallel layered decoding architecture. As a case study, a RC-LDPC decoder based on the rate-1/2 WiMax LDPC code is implemented in CMOS 90 nm process. The decoder can achieve an input throughput of 975 Mbps and supports any rate between 1/2 and 1.

1.3.3 Efficient Decoder Architecture Design for LDPC Codes with Special Structure

Motivation: Although current LDPC decoders can handle QC LPDC codes effectively and efficiently, it is still a problem when dealing with some specially constructed codes, such as the LDPC codes in CMMB standard whose sub-matrices are weigh-2 superimposed matrices instead of identity matrices. The design of an area-efficient LDPC decoder that supports multiple code patterns is essential for wireless applications such as CMMB.

Contribution: We proposed a low-complexity LDPC decoder for CMMB systems. An area-efficient layered decoding architecture based on min-sum algorithm is incorporated in the design. A reconfigurable architecture, which can support dual rate LDPC codes specified in the CMMB standard, is constructed with minimal overhead. A novel split-memory architecture is developed to efficiently handle the weight-2 submatrices that are rarely seen in conventional LDPC decoders. In addition, the check-node processing unit is highly optimized to minimize complexity and computing latency while facilitating a reconfigurable decoding core.

1.3.4 Efficient Architecture Design in LDPC-Like BP-Based Circumstances

Motivation: Besides LDPC decoding, a large variety of algorithms in communication and signal processing can be viewed as instances of BP algorithm, which operates by iterative message passing on a bipartite graph [47]. For example, recent research has focused on the application of the BP algorithm to detecting over a known ISI channel [15, 62, 5]. The computational complexity of the BP-based detection increases exponentially only with the number of the nonzero interferers, with respect to the optimal maximum a posterior (MAP) algorithms or maximum-likelihood (ML) algorithms whose computational complexity are exponential in channel length. BP-based channel detector is needed especially in the sparse ISI channel with long delay spreads and only a few nonzero interferers, such as the underwater acoustic (UWA) channels [62]. Thus it is desirable to investigate LDPC-like VLSI architectures for applications under these circumstances.

Contribution: In this dissertation, we provided a feasible solution of detector architecture for random sparse ISI channels. A reconfigurable architecture was developed in order to switch flexible connections on the factor graph in the time-varying ISI channels. Layered decoding and task rescheduling once used in LDPC decoding were also incorporated to accelerate the iterative process. All of the tasks are managed by the control unit implemented as a finite state machine (FSM). The proposed detector is implemented using ASIC design in 90 nm CMOS process and also prototyped on an FPGA.

1.4 Outline

This dissertation is outlined as follows.

Chapter 2 reviews the fundamentals of LDPC codes and their decoding algorithms that are used throughout this dissertation. Besides standard BP, other decoding algorithms such as loosely coupled algorithm, min-sum algorithm and layered decoding algorithm are also presented.

Chapter 3 develops a ultra-high-throughput LDPC decoder architecture implementation with parallel layered decoding. We first explain how parallel layered decoding works and then presents how it can help split the critical path and eliminate the interconnect network.

In Chapter 4, a puncturing-based RC LDPC decoder architecture that supports any code rate 1/2 to 1 is designed and implemented. First we investigate various puncturing schemes and pick up the optimal one in term of error performance. Then we modify the parallel layered decoding architecture developed in Chapter 3 by adding a control block to carry out puncturing.

Chapter 5 presents a low-complexity LDPC decoder for CMMB systems. Split-memory architecture is proposed to efficiently handle the weight-2 sub-matrices. A reconfigurable architecture, which can support dual rate LDPC codes specified in the CMMB standard, is then constructed with minimal overhead;

Chapter 6 presents VLSI implementation of a BP-based detector for sparse ISI channels, such as underwater acoustic channel. Cache-like architecture is developed by storing only the messages that current node interferes with.

Chapter 7 draws the conclusion.

Chapter 2

LDPC Codes and Decoding Algorithms

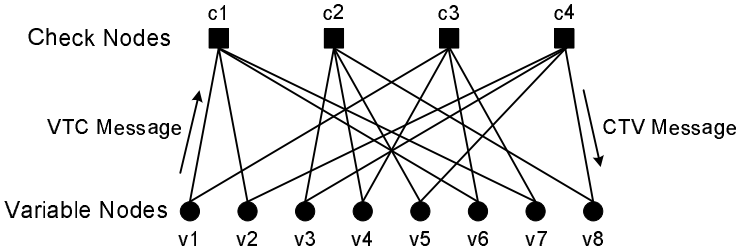
In this Chapter, we reviews some fundamentals of LDPC codes, QC LDPC codes and their decoding algorithms, including standard BP algorithm, min-sum algorithm, loosely coupled algorithm and layered decoding algorithm.

2.1 Introduction of LDPC Codes

The LDPC codes can be described by a $M \times N$ sparse parity check matrix $\mathbf{H}$, in which most of the elements are 0's and only a few are 1's. M denotes the number of parity check equations, that is, number of the check nodes, while N is the block length, that is, the number of variable nodes. Fig. 2.1(a) shows a rate-1/2 4×8 parity check matrix of a regular (8,4) LDPC code. In the decoding aspect, a parity check matrix can be mapped into a bipartite graph, called the Tanner graph, with all of the variable nodes on one side and check nodes on another. Locations of the non-zero elements in the $\mathbf{H}$ matrix indicate the straight connections between variable

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

(a)



(b)

Figure 2.1: Regular (8,4) LDPC codes: (a) Parity-check matrix (b) Tanner graph

nodes and check nodes, as illustrated in Fig. 2.1 (b). These connections can be also considered as the messages (CTV messages and VTC messages) transmitting paths when decoding using the iterative BP algorithm.

2.2 Quasi-Cyclic LDPC Codes

QC LDPC codes are a special class of the LDPC codes with structured $\mathbf{H}$ matrix which can be generated from an $m_b \times n_b$ base matrix $\mathbf{H}_b$.

$$\boldsymbol{H_b} = \begin{bmatrix} \mathbf{P_{0,0}} & \mathbf{P_{0,1}} & \cdots & \mathbf{P_{0,n_b-1}} \\ \mathbf{P_{1,0}} & \mathbf{P_{1,1}} & \cdots & \mathbf{P_{1,n_b-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P_{m_b-1,0}} & \mathbf{P_{m_b-1,1}} & \cdots & \mathbf{P_{m_b-1,n_b-1}} \end{bmatrix} \tag{2.1}$$

Each nonzero element $\mathbf{P_{i,j}}$ in the base matrix is a $z \times z$ submatrix that can be expanded by circularly right-shifting an identity matrix with the shift value defined by $\mathbf{P_{i,j}}$. The structure of the parity check matrix makes it convenient to

determine the locations of the nonzero elements. Random connections between CNUs and VNUs now become well-regulated and easy to handle. Therefore, QC-LDPC codes are welcomed by several advanced communication standards, such as 802.11n, 802.15.3c and 802.16e.

2.3 Belief Propagation Algorithm

The BP algorithm provides an efficient and powerful method to decoding LDPC codes. Standard BP algorithm in [24] is usually transformed into logarithmic domain where additions can be used instead of the complex multiplication. Before presenting the BP algorithm, we first make some definitions as follows: Let c_n denote the n-th bit of a codeword and y_n denote the corresponding received value from the channel. Let $r_{mn}[k]$ ($q_{mn}[k]$) be the CTV (VTC) message from check node m to variable node n at the k-th iteration. Let $N(m)$ denote the set of variables that participate in check m and $M(n)$ denote the set of checks that participate in variable n. The set $N(m)$ without variable n is denoted as $N(m) \setminus n$ and the set $M(n)$ without check m is denoted as $M(n) \setminus m$.

1. **Initialization:**

   Under the assumption of equal priori probability, compute the channel probability p_n (intrinsic information) of the variable node n, by:

   $$p_n = log \frac{P(y_n \mid c_n = 0)}{P(y_n \mid c_n = 1)} \tag{2.2}$$

   The CTV message r_{mn} is set to be zero.

2. **Iterative Decoding:**

   At the k-th iteration, for the variable node n, calculate VTC message $q_{mn}[k]$

by

$$q_{mn}[k] = p_n + \sum_{m' \in \{M(n) \backslash m\}} r_{m'n}[k-1] \tag{2.3}$$

Meanwhile, the decoder can make a hard decision by calculating the APP (a-posterior probability) by

$$\Lambda_n[k] = p_n + \sum_{m' \in M(n)} r_{m'n}[k-1] \tag{2.4}$$

Decide the n-th bit of the decoded codeword $x_n = 0$ if $\Lambda_n > 0$ and $x_n = 1$ otherwise. The decoding process terminates when the entire codeword $x = [x_1, x_2, \cdots\cdots x_N]$ satisfy all of the M parity check equations: $\mathbf{H}x = 0$, or the preset maximum number of iteration is consumed.

If the decoding process does not stop, then, calculate the CTV message r_{mn} for the check node m, by

$$\begin{aligned} r_{mn}[k] &= \prod_{n' \in \{N(m) \backslash n\}} sign\left(q_{mn'}[k]\right) \\ &\quad \times \Psi^{-1}\left(\sum_{n' \in \{N(m) \backslash n\}} \Psi\left(|q_{mn'}[k]|\right)\right) \end{aligned} \tag{2.5}$$

$$\Psi(x) = \Psi^{-1}(x) = log\frac{1+e^{-x}}{1-e^{-x}} \tag{2.6}$$

2.4 Min-Sum Algorithm

The nonlinear $log - tanh$ function in the check node updating step is usually implemented by Look-Up Table (LUT), which seriously increases the complexity and operating latency of the CNU. In paper [78], the author proposed a method to balance the path latency of CNU and BNU by transferring one of the two log-tanh

functions from CNU to BNU. Another effective method is to use the min-sum algorithm to lower the CNU complexity by approximating the CTV message with a minimum operation, as shown in (2.7).

$$r_{mn}[k] = \prod_{n' \in \{N(m) \backslash n\}} sign\left(q_{mn'}[k]\right) \times \left(\min_{n' \in \{N(m) \backslash n\}} \{|q_{mn'}[k]|\} \times \alpha \right) \tag{2.7}$$

Here, a normalized factor α is introduced to compensate for the performance loss exiting in the min-sum algorithm without compensation compared to BP algorithm [23, 26, 10]. In this dissertation, α is set to be 0.75.

Using the min-sum algorithm, the look-up tables (LUTs) which implement the intricate non-linear function in standard BP algorithm are now replaced by rather simple comparators, resulting in simpler computation complexity of CNU. Besides, storage resources can also be reduced, as only the minimum and second minimum value of all of the VTC messages within a check node need to be stored.

2.5 Loosely Coupled Algorithm

In the BP algorithm, messages are transmitted between check nodes and variable nodes iteratively to update each other. VTC messages are renewed by channel probabilities and CTV messages from the check nodes. Then, these renewed messages need to be passed again to update the VTC messages. Every VTC and CTV message should be transmitted immediately after they are renewed along the edges on the Tanner graph. This large amount of transmitted messages causes an interconnection problem. Large chip area is occupied by interconnections and an area-efficient decoder becomes a challenge.

Loosely coupled algorithm [36] was introduced to solve the complex interconnection problem. The decoder does not exchange the CTV and VTC messages between the check nodes and variable nodes. Instead, it delivers only the check and variable summation Δ_m and Λ_n. At a variable node, given the check summation values Δ_m, a VNU would first recover individual CTV message r_{mn} and then calculate the next variable summation Λ_n which will be transmitted to CNUs. As a result, (2.3) and (2.4) can now be modified as:

$$\begin{aligned} r_{mn}[k-1] &= (sign(\Delta_m[k-1]) \times sign(q_{mn}[k-1])) \\ &\quad \times \Psi^{-1}(\Psi(|\Delta_m[k-1]|) - \Psi(|q_{mn}[k-1]|)) \end{aligned} \tag{2.8}$$

$$\Lambda_n[k] = p_n + \sum_{m' \in M(n)} r_{m'n}[k-1] \tag{2.9}$$

$$q_{mn}[k] = p_n + \sum_{m' \in \{M(n) \backslash m\}} r_{m'n}[k-1] \tag{2.10}$$

Similarly, at a check node, given the variable summation values Λ_n, a CNU would first recover individual VTC message q_{mn} and then calculate the next check summation $\Delta_m$which will be transmitted to VNUs. As a result, (2.5) can now be modified as:

$$q_{mn}[k] = \Lambda_n[k] - r_{mn}[k-1] \tag{2.11}$$

$$\begin{aligned} \Delta_m[k] &= \prod_{n' \in N(m)} sign(q_{mn'}[k]) \\ &\quad \times \Psi^{-1}\left(\sum_{n' \in N(m)} \Psi(|q_{mn'}[k]|)\right) \end{aligned} \tag{2.12}$$

$$r_{mn}[k] = \prod_{n' \in \{N(m) \backslash n\}} sign\left(q_{mn'}[k]\right) \times \Psi^{-1}\left(\sum_{n' \in \{N(m) \backslash n\}} \Psi\left(|q_{mn'}[k]|\right)\right) \tag{2.13}$$

If the min-sum algorithm is also applied, then we only need to transmit the variable summation Λ_n, leading to further simplification in interconnection complexity [70].

2.6 Early Termination Strategy

As presented in the BP algorithm, the decoding process can be terminated by two general conditions: one is the parity check equations $\mathbf{H}x = 0$, another is the actual number of iterations exceeds the predefined maximum number. For regular LDPC code decoding in paper [85], parity check equations become easy to verify, because every clock cycles the decoded bits from the VNUs are within a same check and parity check can be completed immediately after the decoded bits are decided. However, for irregular codes such parity check method would lead to larger hardware resource, longer decoding latency and lower decoding throughput, because extra storage resources and clock cycles are required to save the hard decision bits and to verify to see if the parity check equations are satisfied.

Another conventional termination method is to only set the maximum number of iterations and stop at the preset number without considering if the parity check equations are satisfied. The method will have little hurt to the error-correction performance of the LDPC codes if the maximum number if sufficiently large. In circuit level design, a simple counter will be fine to fully implement this termination method. Thus, less hardware complexity is employed, compared with the parity check equation method. However, the method cannot arbitrarily adjust the num-

ber of iterations and will cause a waste in high SNR channels such as wire-line environments where fewer errors occur.

In order to address the drawbacks of the two existing termination criteria, we propose a more convenient and robust "early termination" strategy [66] to balance the throughput requirements and hardware usage. The early termination strategy can not only dynamically adjust the number of iterations when dealing with communication channels of different SNRs, but also be sufficient for the low-cost and low-power hardware implementation.

The pivot of the early termination strategy lies in that hard decisions from previous iteration are stored and compared with newly generated hard decisions. If all of the decoded bits at current iteration are identical with those of the previous iteration, then the decoder indicates a successful decoding of current codeword and jumps out of the iterative process. Otherwise, the decoding process would continue until two successive hard decisions become the same or the maximum number of iterations is satisfied. As indicated in [66], for the LDPC code used in this paper, we only need to check 1152 information bits instead of 2304 bits, because it is a rate-1/2 systematic linear block code.

Fig. 2.2 shows the hardware architecture of the early termination strategy. It mainly consists of XOR gates, OR gates and registers which are used to save the hard decisions. At every iteration cycle, hard decisions of previous iteration are retrieved from registers and XORed with current decisions to generate an array of intermediate signals $same\_diff$. Then each bits of the $same\_diff$ signal are ORed to generate the $flag$ signal. A high-level $flag$ signal indicates difference of hard decisions between the two successive iterations and then the decoding process will continue if the predefined number of iteration is not satisfied. Otherwise, a low-level $flag$ signal tells identity of the two successive hard decisions and the process

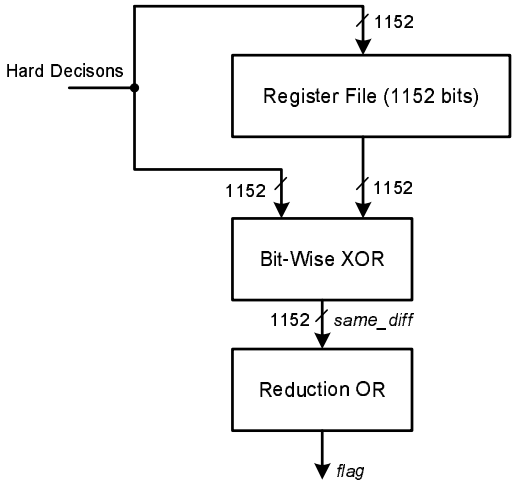


Figure 2.2: Implementation of early termination strategy

will stop.

2.7 Layered Decoding Algorithm

In BP algorithm [24], the two-phase mutual messages, namely VTC messages q_{mn} and CTV messages r_{mn}, are updated by separate processing units and passed to each other iteratively. q_{mn} updates will not start until all of the r_{mn} are prepared and vice versa. In horizontal layered decoding, the CTV message from the current layer will be passed vertically to all other unprocessed layers that belong to the same variable node. In each iteration, the horizontal layers are processed sequentially from the top to the bottom layer.

As an example, the **H** matrix of rate-1/2 LDPC code from 802.16e standard is quasi-cyclic, which consists of sub-matrices that are generated by circularly shifting an identity matrix, as shown in Fig. 2.3. Such structure is well suited for horizontal layered decoding as each sub-matrix has the column weight of one and we can treat each row of the base matrix as a layer. Message passing flow of layered decoding for the selected **H** matrix is illustrated in Fig. 2.4.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1		94	73						55	83			7	0										
2		27				22	79	9				12		0	0									
3				24	22	81		33				0			0	0								
4	61		47						65	25						0	0							
5			39				84			41	72						0	0						
6					46	40		82				79	0					0	0					
7			95	53						14	18								0	0				
8		11	73				2			47										0	0			
9	12				83	24		43				51									0	0		
10						94		59			70	72										0	0	
11			7	65					39	49													0	0
12	43					66		41				26	7											0

Figure 2.3: Parity-check matrix for the selected rate-1/2 LDPC code in 802.16e standard.

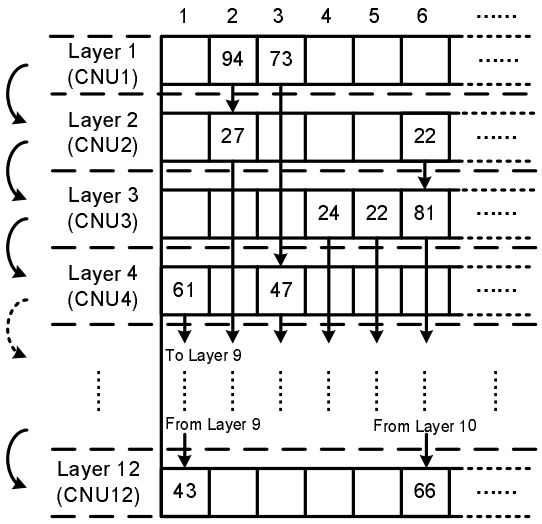


Figure 2.4: Message passing flow in horizontal layered decoding.

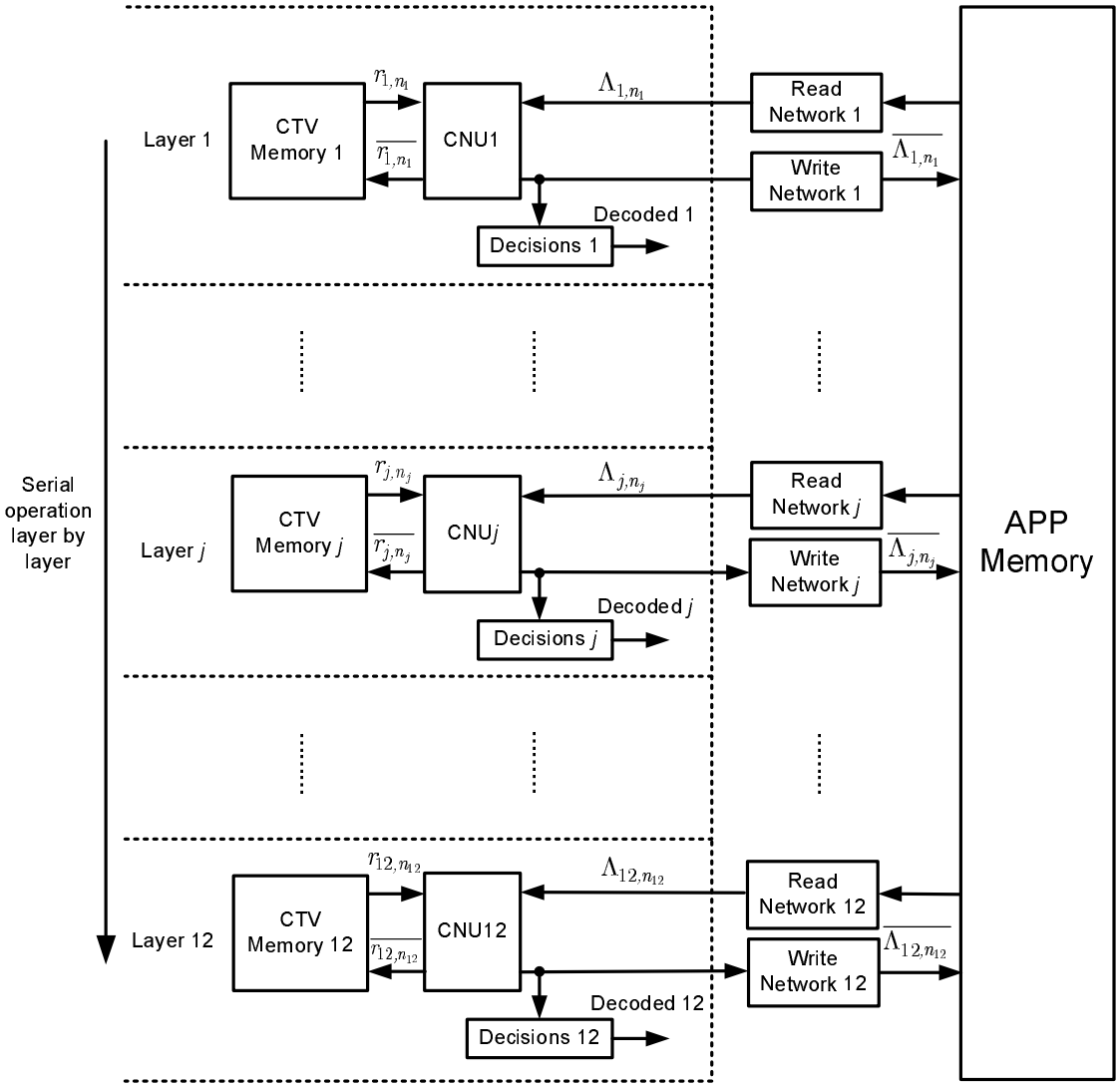


Figure 2.5: Architecture of horizontal layered decoding with loosely coupled algorithm.

Loosely coupled algorithm is also adopted in this work to reduce the interconnection complexity. As illustrated in Fig. 2.5, at the j-th layer, the VTC messages are first recovered from the variable summations Λ_n and the CTV message from memories, as in (6). At the output of CNUj, the updated CTV messages $\left\{r_{j,n_j} \mid n_j \in N(j)\right\}$ in (7) are stored back to the CTV memories. Variable summation $\left\{\Lambda_{n_j} \mid n_j \in N(j)\right\}$ are renewed as in (8) and sent back to the APP memory. The entire flow can be expressed as follows:

$$q_{j,n_j} = \Lambda_{n_j} - r_{j,n_j} \tag{2.14}$$

$$\begin{aligned} \overline{r_{j,n_j}} \;=\; & \left(\prod_{n'_j \in \{N(j) \backslash n_j\}} sign\left(q_{j,n'_j}\right) \right) \\ & \times \left(\min_{n'_j \in \{N(j) \backslash n_j\}} \left\{ \left| q_{j,n_j} \right| \right\} \times \alpha \right) \end{aligned} \tag{2.15}$$

$$\overline{\Lambda_{n_j}} = q_{j,n_j} + \overline{r_{j,n_j}} \tag{2.16}$$

Compared with conventional layered decoding algorithm, it can be observed that loosely coupled algorithm does not require variable node operations. In other words, VNUs can be eliminated since their main function of updating the variable summations Λ_n can be done by CNUs using VTC messages $r_{j,n_j}[k]$ from previous layers, as indicated in (2.16).

Chapter 3

High-Throughput LDPC Decoder Architecture with Parallel Layered Decoding

3.1 High Throughput Strategies

With the increasing demand for high-data-rate wireless applications, many recent communication systems employ ultra-high throughput channel codes to match the data-rate requirements. For example, 802.15.3c standard is targeted for the data rate of multi-giga bits per second (Gbps), thus LDPC codes are preferred compared with convolutional codes and Turbo codes. However, it is a great challenge to design a high-through LDPC decoder due to the complexity of decoding algorithm. Since QC-LDPC codes are increasingly popular in emerging communication standards, we focus on the design and architecture of QC-LDPC decoder in this paper.

The throughput of an LDPC decoder can be calculated as

$$Throughput = \frac{Freq \times Block\,Length}{Cycles\,per\,Iter \times Num\,of\,Iter} \tag{3.1}$$

Therefore, three strategies can be attempted in order to improve the throughput: reducing the number of iterations required for convergence, reducing the decoding latency per iteration and improving the operating frequency. Correspondingly, three architecture-aware schemes are studied in this paper, including layered decoding algorithm, parallel layered decoding architecture, and critical path splitting technique.

First, layered decoding algorithm is adopted to reduce the required number of iterations by a factor of two for any given SNR, compared with the standard BP algorithm. Hence, the decoding throughput is supposed to be doubled without any bit error performance loss. Generally, there are two layered decoding methods: horizontal layered decoding [51, 33, 45, 9, 25, 28] and vertical layered decoding [82]. It has been proved these two methods are theoretically equivalent and both can converge twice as fast as the BP algorithm [65]. In this paper, we employ the horizontal layered decoding strategy because it is favorable for the min-sum algorithm.

Secondly, we propose a novel scheme namely parallel layered decoding architecture that enables concurrent processing among all layers. In traditional layered decoding architecture, the layers are processed sequentially which leads to longer decoding latency per iteration [33]. In parallel layered decoding architecture, precisely scheduled message passing among different layers guarantees that all updated messages are passed to their designated locations in connected layers. The parity-check matrix optimization procedure adds specific offsets to each layer (row) in the base

parity check matrix, making the idle time intervals between connected layers sufficiently large for message passing. Moreover, parallel layered decoding architecture supports the decoding architecture in which check node updates unit (CNU) and variable node updates unit (VNU) are combined into a single functional unit. Thus, no extra clock cycles are needed to complete the variable node update as they have been merged into the CNU. As a result, the number of clock cycles per iteration in parallel layered decoding architecture can be reduced by 75% compared with the existing architectures.

Finally, the technique of reducing critical path delay is proposed to improve the operating frequency at circuit-level. The combination of CNU and VNU results a long critical path in the decoder implementation [66], which limits the maximum clock speed. Fortunately, there are idle time intervals among different layers which allows the iterative messages to be processed and passed to the next layer within several clock cycles. Therefore, we can insert registers to split the CNU (including the VNU) into several pipelined stages. Consequently, the critical path is split into multiple stages and the clock speed can be improved dramatically, on condition that the number of stages does not exceed the idle time intervals. In practice, this critical path splitting method can increase the maximum frequency of the decoder by a factor of 3 or higher.

To demonstrate the aforementioned three techniques, a rate-1/2 2304-bit QC-LDPC code is selected from 802.16e standard as a case study. In addition, existing techniques such as min-sum algorithm and loosely coupled algorithm are also employed to simplify decoding complexity and to reduce the chip area of the decoder design.

3.2 Parallel Layered Decoding Architecture

As discussed above, the essential reason that layered decoding algorithm can reduce the number of iterations is that the latest extrinsic messages are passed to and employed by the subsequent layers within the current iteration. Therefore, layered decoding requires layers to be processed sequentially, which results a large decoding latency per iteration. A method of increasing parallelism inside a layer is proposed in [51, 45], but all layers are still processed in series. Although decoding throughput can be improved, this method introduces crossbar-based interconnection networks that increase the hardware complexity.

Motivated by the partly parallel mechanism mentioned in [85], we propose the parallel layered decoding architecture that allows all layers to be processed concurrently. Each layer generates and sends updated messages and at the same time it also receives the updated messages from other layers. Unlike the method proposed in [51, 45], parallel layered decoding architecture uses parallel processing among all layers and serial processing within each layer. Detailed message processing flow at the j-th layer (CNU) can be summarized as

1. Fetch the corresponding variable summations $\left\{\Lambda_{n_j} \mid n_j \in N(j)\right\}$ from the APP memory and CTV messages $\left\{r_{j,n_j} \mid n_j \in N(j)\right\}$ from local CTV memory.

2. Calculate the VTC messages $\left\{q_{j,n_j} \mid n_j \in N(j)\right\}$ in the same row using (2.14).

3. Calculate horizontally to obtain new CTV messages $\left\{\overline{r_{j,n_j}} \mid n_j \in N(j)\right\}$, as in (2.15).

4. Immediately update the variable summations $\left\{\overline{\Lambda_{n_j}} \mid n_j \in N(j)\right\}$ using (2.16).

5. Deliver the new variable summation $\overline{\Lambda_{n_j}}$ to the same location at another layer.

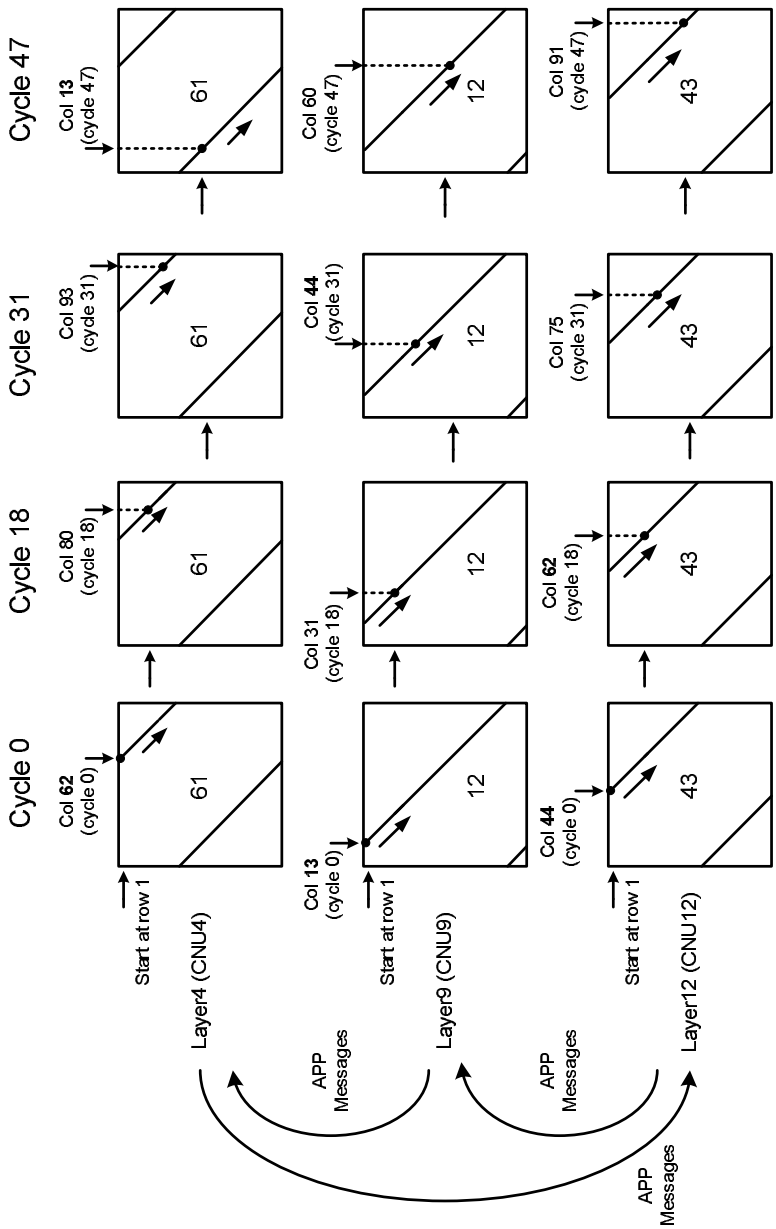


Figure 3.1: Processing status at four different clock cycles.

Hereby, we explain how to pass $\overline{\Lambda_{n_j}}$ messages in parallel layered decoding architecture. Instead of passing $\overline{\Lambda_{n_j}}$ to all unprocessed layers as in conventional layered decoding, parallel layered decoding architecture only sends $\overline{\Lambda_{n_j}}$ to the layer that will use it next. Let us take the first column of the **H** base matrix as an example. None-zero entries are at the 4th, 9th and 12th layer whose permutation numbers are 61, 12 and 43, respectively. Now we suppose that at cycle 0 each of these three corresponding CNUs starts to process from the first row of the sub-matrix, corresponding to the 62th, 13th and 44th column indices. In general, corresponding row and column indices of the entry being processed at cycle l can be calculated as

$$r_{index} = l \tag{3.2}$$

$$c_{index} = mod\left(l + 1 + s\left(i, j\right), 96\right) \tag{3.3}$$

In Fig. 3.1, it illustrates the message passing routes in parallel layered decoding architecture using the the first column of the parity check matrix in Fig. 1 as an example. The operation sequence in time of these three layers is described as the following:

1. At cycle 0, CNUs at layers 4, 9 and12 start to process simultaneously from row 1 of each sub-matrix, corresponding to column 62, column 13 and column 44 according to (3.3).

2. At cycle 18, all three layers are processing at row 19, corresponding to column 80 (in layer 4), column 31 (in layer 9) and column 62 (in layer 12). Layer 12 calls for the latest summation message of column 62 which has already been updated by layer 4. Therefore, the updated variable summations at layer 4 should be sent to layer 12 (Layer 4 →Layer 12).

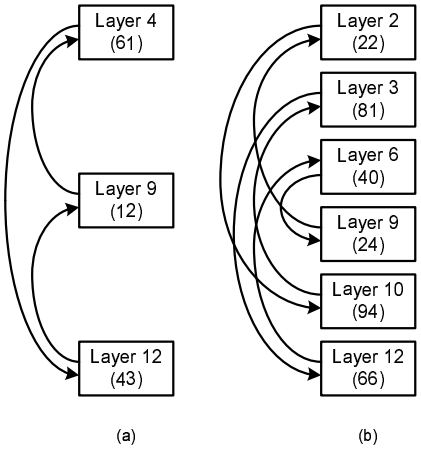


Figure 3.2: Variable summations passing directions of the **H** base matrix: (a) for column 1 (b) for column 6.

3. At cycle 31, all three layers are processing at row 32, corresponding to column 93 (in layer 4), column 44 (in layer 9) and column 75(in layer 12). Layer 9 calls for the latest summation message of column 44 which has already been updated by layer 12. Therefore, the updated variable summations at layer 12 should be sent to layer 9 (Layer 12 →Layer 9).

4. Similarly, at cycle 47, all three layers are processing row 48, corresponding to column 13 (in layer 4), column 60 (in layer 9) and column 91 (in layer 12). Layer 4 calls for latest summation message of column 13 which has already been updated by layer 9. Therefore, the updated variable summations at layer 9 should be sent to layer 4 (Layer 9 →Layer 4).

Based on the description above, message passing routes for column 1 of base matrix is shown in Fig. 3.2(a). An additional example is illustrated in Fig. 3.2(b) for column 6 of the base matrix. In general, we can determine the message passing routes among layers based on the their permutation values. For each column, we can sort the permutation values of all layers in descending order. Each layer then passes messages to the next layer with a smaller permutation value. Moreover, the

Layer	Offset	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	+0		94	73						55	83			7	0										
2	+8		35				30	87	17				20		8	8									
3	+0				24	22	81		33				0			0	0								
4	+12	73		59						77	37						12	12							
5	+84			27				72			29	60						84	84						
6	+0					46	40		82				79	0					0	0					
7	+88			87	45						6	10								88	88				
8	+8		27	81				10			55										8	8			
9	+0	12				83	24		43				51									0	0		
10	+16						14		75			86	88										16	16	
11	+0			7	65					39	49													0	0
12	+80	27					50		25				10	87											80

Figure 3.3: Offset-modified parity-check matrix for rate-1/2 LDPC code in 802.16e.

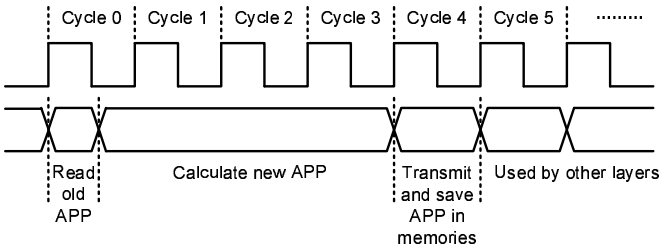


Figure 3.4: Timing diagram for parallel layered decoding architecture.

layer with the smallest permutation value loops back and connects to the layer with the largest value. This message passing scheme guarantees the updated messages among all layered are processed progressively within each iteration.

3.3 Critical Path Splitting

According to the message passing mechanism explained above, we suppose CNUs in all layers start at row 1 of each sub-matrix. Actually, the CNUs can start to operate from different rows, which is equivalent to adding an offset to the permutation value of each layer. For example, Fig. **??** shows a modified **H** base matrix with a set of offset values added for different layers. The offset values are carefully selected, such that the difference of modified permutation values between any two layers is

at least 5. It means each layer needs to read, update, store and pass the message to the next connected layer with 5 clock cycles. The detail steps for the processing in a layer are shown in in Fig. 3.4. In other words, each CNU has a period of 4 cycles to complete the task of reading old message and update new message, as indicated in (2.14) to (2.16). The message passing rule remains the same, except that the updated permutation values should be used to determine the connections in PLDA.

From Fig. 3.3, we can see that the row weight of each layer is either 6 or 7, which requires 6 or 7 messages to be compared in the CNU. Combined with other necessary functions such as adding, rounding and memory read/write, CNU becomes the critical path that limits the maximum operating frequency of the decoder. Taking advantage of the 4-cycle time intervals, we can split the critical path of CNU into 4 pipelined stages by inserting 3 levels of registers. An optimal splitting yields balanced delays among the pipeline stages. The implementation of critical path splitting will be given in Section 3.4.2.

3.4 Proposed Decoder Architecture

In order to prove the concept of our proposed parallel layered decoding architecture and high-throughput strategies, a rate-1/2 2304-bit QC-LDPC code selected from 802.16e standard is designed based on the offset-modified **H** matrix. The size of each sub-matrix is chosen to be 96×96. Before constructing the functional units of the decoder, we first perform fixed-point analysis to quantize the word-length of messages. In fact, word-length quantization is a tradeoff between memory resources and bit-error-rate performance. Fig. 3.5 shows the BER performance of the selected rate-1/2 LDPC code using message word-length $(3, 2)$. As demonstrated in [**?**], 5 bits (3 bits for the integer part and 2 bits for the fractional part) are adequate

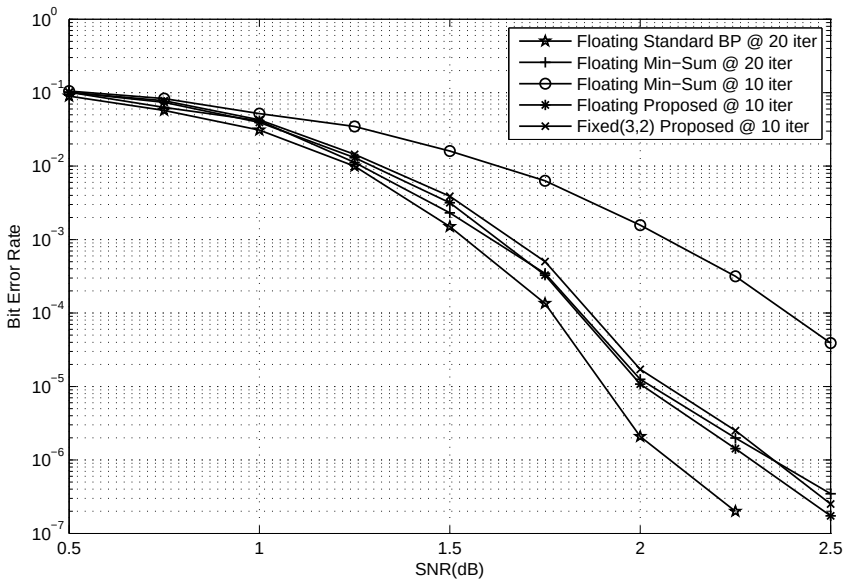


Figure 3.5: BER performance comparison of different decoding algorithms.

for representing the absolute values of the extrinsic messages. Considering one additional sign bit, we choose 6 bits fixed-point representation for messages in this implementation.

3.4.1 Overall Decoder Architecture

The overall architecture of the QC-LDPC decoder is shown in Fig. 3.6. It consists of check node processing units (CNUs), APP memory banks, CTV memory banks and hard-decision units. The entire **H** matrix is divided into 12 layers and each layer employs a dedicated CNU. APP memory banks and CTV memory banks are used to store variable summations and CTV messages. Each non-zero sub-matrix corresponds to one APP memory unit and one CTV memory unit. In layered decoding, each APP memory exports a summation message to the CNU and imports a new summation message from the CNU of another layer. Similarly, each CTV memory exports a CTV message to the CNU and imports a new CTV message from the same CNU for each layer. As a result, single-port memories that support

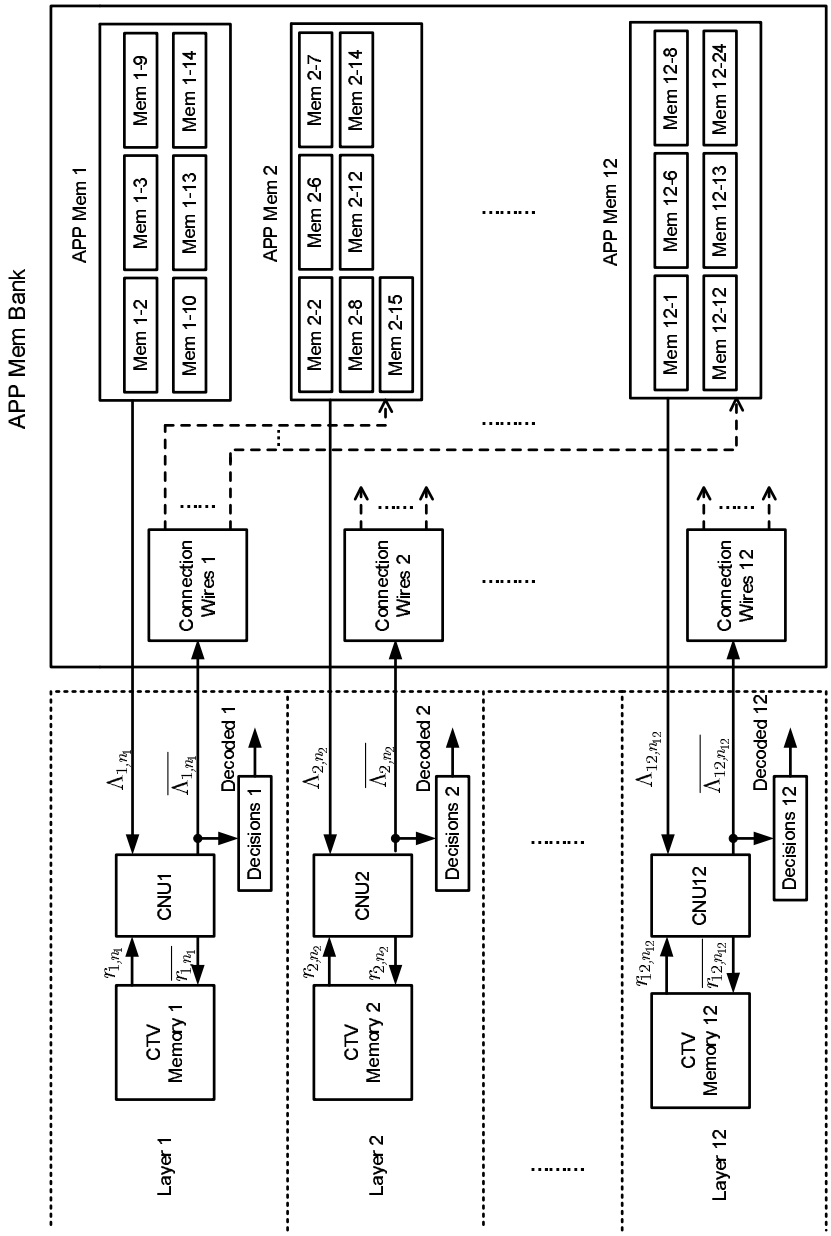


Figure 3.6: Overall parallel layered decoding architecture for QC-LDPC codes.

concurrent read and write operations are used in the design. Since there are 76 submatrices in total, the APP memory bank and the CTV memory bank each consists of 76 small single-port memory units.

Each layer in the parallel layered decoding architecture corresponds to 6 or 7 APP memory units, which means that the updated variable summations of the j-th layer $\overline{\Lambda_{j,n_j}}$ will be delivered to APP memory units in other layers based on the message passing scheme described in Section **??** and Fig. 3.2. Therefore, these variable summations are passed to their destinations through fixed connection wires instead of crossbar-based networks.

3.4.2 Pipelined Architecture for CNU

For check node update, each layer employs a CNU to perform a series of functions including subtraction, comparison and addition. Consequently, the CNU becomes the longest delay path in timing. In min-sum algorithm, a CNU needs to compare 6 or 7 numbers to find the minimum value and its location as well as the second minimum value. Thus, the comparator becomes a key component in a CNU. A 2-input comparator consists of an adder and a multiplexer. A 3-input comparator can be implemented with three adders, five multiplexers and some basic logic gates. Based on comparison units of 2-input and 3-input comparators, 6-input or 7-input comparator can be constructed by combining three levels of 2-input or 3-input comparators, as shown in Fig. 3.7.

Fig. 3.7 shows detailed architecture of CNU and its functional block inside. The *subtractor* and *adder* blocks fulfill the functions defined in (2.14) and (2.16), respectively. Two *quantizers* are inserted to prevent the CTV and variable summations from overflow during computation. The *abs* block calculates the absolute values of VTC messages and *the compare & select* block determines the values of CTV

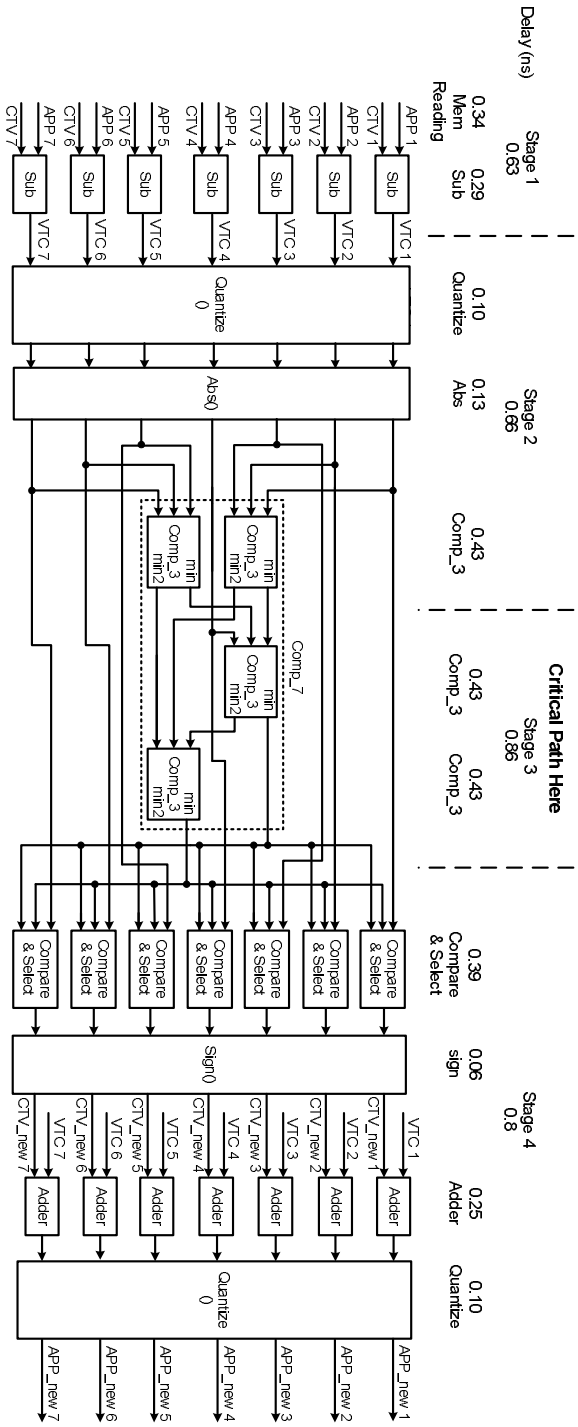


Figure 3.7: CNU architecture with critical path splitting into 4 pipelined stages.

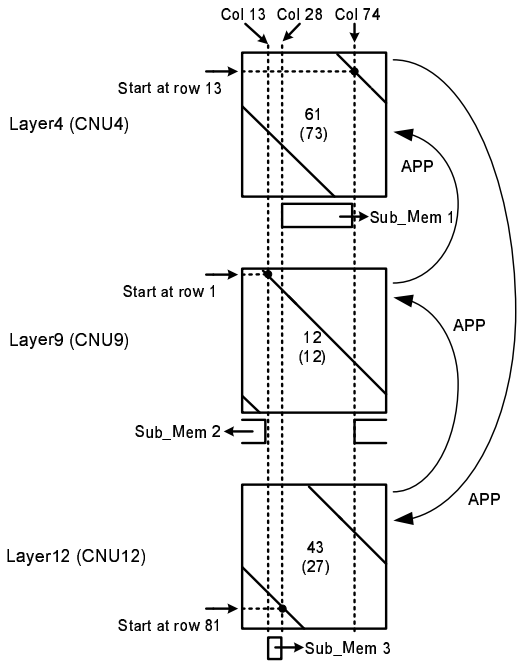


Figure 3.8: Register allocation in each section of the hard decisions.

messages.

As explained in Section 3.3, in order to reduce the critical path delay and improve clock speed, a CNU can be split into 4 pipelined stages by inserting 3 set of registers. Registers are carefully inserted such that all 4 pipeline stages have approximately the same delay, since maximum frequency is determined by the longest delay path. Delays of each functional unit and the pipeline stages are also shown in Fig. 3.7.

3.4.3 Decision Units

The output bits of the LDPC decoder are decided by the signs of the variable summations. In traditional horizontal layered decoding, decisions can be made during the variable node processing at the bottom layer. Parallel layered decoding architecture operates in a slightly different way in which every layer generates variable summations during each iteration, as CNUs in different layers are working concur-

rently. As illustrated in Fig. 3.8, let us take the first column of the modified **H** base matrix with offset as an example. CNU 4, CNU 9 and CNU 12 start to operate from row 13, row 1 and row 81, respectively, as defined by the offset values shown in Fig. 3.3, which correspond to column 74, column 13 and column 28, respectively. According to the message passing rule derived in Section 3.4.2, variable summations passing routes of the first column is Layer 4→Layer 12→Layer 9→Layer 4, also shown in Fig. 3.8. After completion of each iteration, the final variable summations for columns 13 through 27 are stored at layer 12. Variable summations for columns 28 through 73 are stored at layer 4. Similarly, variable summations for column 74 through 12 are stored at layer 9. Therefore, the hard decision bits for columns 1 through 96 are stored in the memories distributed in 3 different layers.

In this design, we employ a convenient and robust "early termination" strategy, similar to [64, 66]. The pivot of the early termination strategy lies in that the hard decision bits from previous iteration are stored and compared with the decision bits from current iteration. If all decoded bits are identical, then the decoder indicates successful decoding of a codeword and the iterative decoding terminates. Otherwise, the decoding process continues until the maximum number of iterations is reached.

3.5 Implementation Results

In order to evaluate the performance of our proposed parallel layered decoding architecture, we implement a rate-1/2 2304-bit QC-LDPC decoder in TSMC 90nm 1.0V CMOS technology with 8-layer metals. We complete synthesis and core area place and route using Synopsys tools.

Implementation results show that the decoder can operate at a maximum frequency of 950MHz after synthesis, corresponding to 2.2Gbps decoding throughput

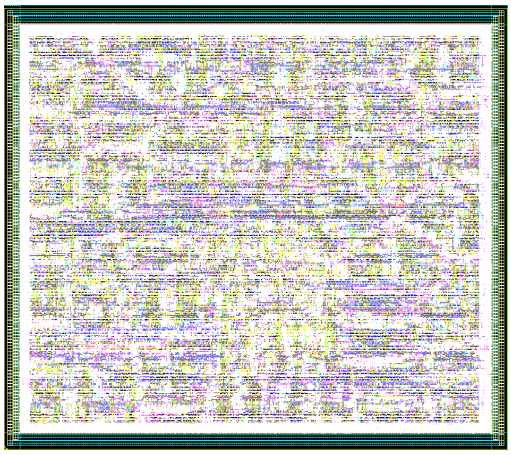


Figure 3.9: Layout of the decoder core area.

using 10 iterations. A total of 152 single-port memory units each with size of 96×6 bits (including one sign bit for each message) are employed, which sums up to $87,752$ bits of memory use and occupies more than 75% of the core area. Parallel layered decoding architecture only needs $(p+3) \times Iter$ clock cycles for the decoding process and p is the size of the sub-matrix. In [66] and [70], this number rises to $p \times (4 \times Iter + 1)$ and $p \times (5 \times Iter) + 12$, respectively. Hence, under the same number of iterations, parallel layered decoding architecture could reduce the decoding latency by approximately 75%. Moreover, the implementation results show that the maximum operating frequencies with and without critical path splitting method are 950MHz and 305MHz, respectively, which demonstrates an improvement of the decoder speed by a factor of 3. Combined with layered decoding algorithm that doubles the convergence speed, our proposed architecture can significantly improve the throughput of the QC-LDPC decoder up to multi-Gbps.

Fig. 3.9 shows the layout view of the decoder core area of $1.8mm \times 1.6mm$ and the logic density of 70%. The design does not have a crossbar interconnection network, since parallel layered decoding architecture employs fixed message passing paths Single-port memories are generated by Synopsys DesignWare tool and thus

Table 3.1: Overall comparison between proposed decoder and other existing LDPC decoders.

	C. Liu [45]	X. Shih [66]	T. Brack [9]	G. Gentile [25]	Y. Ueng [70]	M. Karkooti [?]	Proposed Decoder
Code Length	576~2304	576~2304	576~2304	576~2304	2304	1944	2304
Frequency	150MHz	83.3MHz	333MHz	400MHz	200MHz	412MHz	950MHz
Iterations	20	2~8	10, 15	15	4.6 (average)	15	10
Throughput	105Mbps	60~220Mbps	133-928Mbps	128-746Mbps	106Mbps	736MHz	2.2Gbps
Technology	90nm	130nm	130nm	65nm	180nm	130nm	90nm
Area	$6.25mm^2$	$8.29mm^2$	$3.83mm^2$	$0.59mm^2$	-	$2.4mm^2$	$2.9mm^2$
Power	264mW	52mW	-	-	-	502mW	870mW

flattened during synthesis and place and route design flow. The core area of the decoder is $2.9mm^2$ and the estimated power consumption is 870 mW.

Table 3.1 shows the decoder implementation results compared with the existing QC-LDPC decoders. Note that the throughput values from [9, 25] are recalculated based on the decoded bits for fair comparison to other implementations listed in Table 3.1. We show that the proposed decoder can achieve higher decoding throughput with comparable or smaller chip area. The decoder consumes more power mainly due to its high operating frequency.

3.6 Summary

LDPC codes are widely used in recent communication systems due to their superior error-correction performance. In this chapter, we proposed a new architecture to improve the throughput of QC-LDPC decoders. With parallel layered decoding architecture and critical path splitting technique, the decoder implementation, which uses TSMC 90nm technology, can achieve 2.2 Gbps decoding throughput for selected rate-1/2 irregular QC-LDPC codes. In addition, min-sum and loosely coupled algorithms are employed for area efficiency and the core size is $2.9mm^2$. The proposed parallel layered decoding architecture is suitable for the near-capacity channel codes and to meet the increasing demand of high data rate communication systems.

Chapter 4

High-Throughput Rate-Compatible LDPC Decoder Architecture

4.1 Introduction

Recently, there is a growing interest in rate-compatible (RC) LDPC codes and their applications. On time-varying channels, it is desirable to adjust the FEC code rate according to the channel state information (CSI). Rate compatibility is also desirable for communications in the type-II hybrid automatic-repeat-request (ARQ) protocols [42]. The concept of RC LDPC codes was first proposed in [32, 30, 31] by puncturing the parity bits of a low-rate randomly constructed LDPC code (called the mother code). The punctured codes showed good error performance, but the complicated puncturing algorithm and large block size were not practical for hardware implementation. Later, finite-length puncturing patterns were proposed in [29] by introducing the definition of k-step recovery (k-SR) variable node. Based on the k-SR theory, several studies were presented on puncturing schemes for QC LDPC codes with dual-diagonal parity structure [13, 58]. Nevertheless, the hardware design

of RC LDPC decoder has not yet been well studied.

From the hardware implementation prospective, there are many existing research work on VLSI implementation of LDPC decoders for single-rate codes [85, 12, 51, 86, 77, 17, 16] and multi-rate codes [52, 44, 81, 68, 84, 46, 66]. With the increasing demand for high-data-rate wireless applications, high-throughput LDPC decoder architectures are in need. In our previous work presented in Chapter 3 (also see [83]), we proposed a parallel layered decoding architecture that can provide up to 1 Gbps input throughput owing to the concurrent processing of all layers and the split critical path resulting much faster clock speed. However, the parallel layered decoding architecture has a *major drawback*. Because it uses fixed connections among layers, the custom designed decoder only fits one specific code. That provides the motivation for us to conduct further research to solve the flexibility problem.

In this paper, we investigate the puncturing schemes for rate-compatible LDPC codes and their hardware implementations. For QC LDPC codes with dual-diagonal parity structure, an efficient puncturing scheme is selected which includes the weight-3 vertical sub-matrix in the puncturing block. As a case study using the rate-1/2 WiMax LDPC code, we show that the selected puncturing scheme results in the bit-error-rate (BER) performance degradation of less than 0.2dB compared with dedicated WiMax codes at four different code rates. Subsequently, we incorporate the puncturing scheme into the parallel-layered-decoding based architecture for the design of a RC LDPC decoder. The decoder is implemented in CMOS 90 nm process and can achieve an input throughput of 975 Mbps at 10 iterations. It supports any arbitrary rates between rate of the mother code and 1.

The rest of this chapter is organized as follows. Section 4.2 introduces the background of RC LDPC codes and a comparison of different puncturing schemes. The high-throughput RC LDPC decoder architecture is presented in Section 4.3. Sec-

tion 4.4 gives the implementation results by comparing with some existing WiMax LDPC decoders, followed by summary and future work in Section 4.5.

4.2 Puncturing Schemes for Rate-Compatible LDPC Codes

4.2.1 Quasi-Cyclic LDPC Codes with Dual-Diagonal Parity Structure

As introduced in Section 2.2, QC LDPC codes are a special class of structured LDPC codes which are well suited for hardware implementation. In [56], a special class of systematic codes is defined based on QC LDPC codes. The base matrix $\mathbf{H}_b$ can be partitioned into two parts, the systematic bits matrix $\mathbf{H}_s$ on the left and the parity bits matrix $\mathbf{H}_p$ on the right, such that $\mathbf{H}_b = \left[(\mathbf{H}_s)_{m_b \times k_b} \vdots (\mathbf{H}_p)_{m_b \times m_b}\right]$, where $k_b = n_b - m_b$. The parity bits matrix $\mathbf{H}_p$ can be further partitioned into two sections: the left most column $\mathbf{H}_o$ is a weigh-3 matrix and the remaining columns $\mathbf{H}_d$ is a dual-diagonal matrix.

$$\boldsymbol{H}_{\mathbf{p}} = \left[\begin{array}{c|cccccc} & H_o & & & H_d & \\ \mathbf{P_{b0}} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P_{b1}} & \mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \mathbf{0} & \mathbf{P_{b2}} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{P_p} & \vdots & \vdots & \cdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{P_{b_{m_b-2}}} & \mathbf{I} \\ \mathbf{P_q} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{P_{b_{m_b-1}}} \end{array}\right] \tag{4.1}$$

We refer such structure as "dual-diagonal parity structure". The original purpose of this structure as in [56] was intended for fast encoding because it guarantees linear time encoding efficiency. Consequently, the dual-diagonal parity structure has been adopted in many LDPC codes including those in the WiMax standard [4]. In WiMax codes, the secondary diagonal elements $\mathbf{P}_{\mathbf{b}_1}, \mathbf{P}_{\mathbf{b}_2}, \cdots \mathbf{P}_{\mathbf{b}_{m_b-1}}$ are also identity matrices. Then the connections between the parity bit nodes and the check bit nodes become zigzag edges. In this paper, we focus on the puncturing schemes and decoder design for rate-compatible LDPC codes with such dual-diagonal parity structures.

4.2.2 Rate-Compatible LDPC Codes

Rate compatibility can be achieved by puncturing the parity bits of the mother code. If belief-propagation (BP) algorithm [24] is employed for decoding, puncturing can be carried out by simply setting the log-likelihood ratios (LLRs) of the punctured bits to zeros (in logarithmic domain). Several puncturing schemes have been proposed for randomly constructed LDPC codes [32, 30, 31] and for short-length LDPC codes [29]. The punctured codes, though at different code rates, are originated from the same mother code. Therefore, only one encoder and decoder is needed.

A major advantage of using rate-compatible codes is to reduce the storage memory. For instance, the WiMax standard lists six different LDPC codes for four different rates (1/2,2/3A,2/3B,3/4A,3/4B,5/6). A typical WiMax LDPC encoder/decoder design [9, 25, 66, 45] must store the parity-check matrices of all six codes. In addition, it also requires a complicated switching network that can change the connections among VNUs and CNUs for each different code. Rate-compatible codes avoid the storage overhead as well as the potential latency problem caused by the large switching network. In communication systems, rate-compatible codes are also

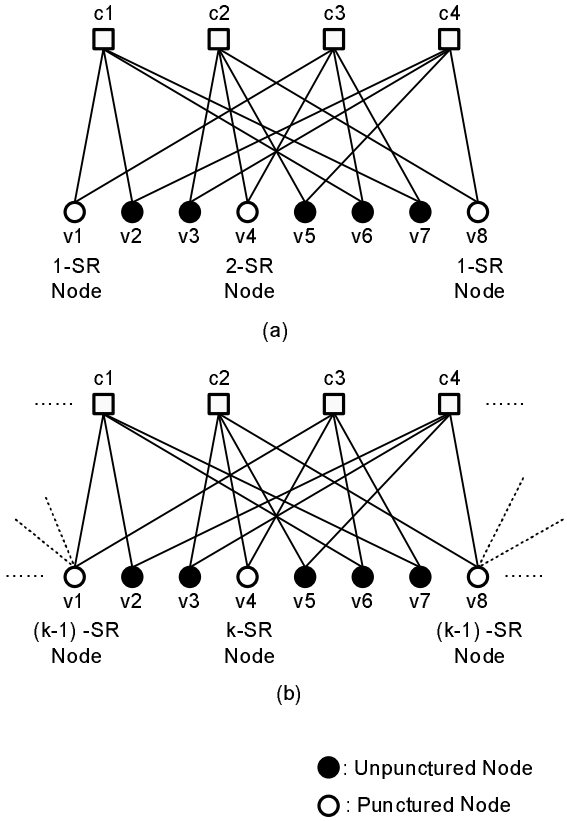


Figure 4.1: Description of 1-SR node and k-SR node.

suitable for the hybrid ARQ protocols [42], because the transmitter can add redundancies progressively by puncturing the mother code based on the CSI.

Since rate-compatible codes have many advantages listed above, a prominent question is: can they provide comparable error performance with the unpunctured codes? For fair comparison, the unpunctured codes are referred to dedicated codes with the same length of the punctured codes. Before presenting the detail puncturing schemes, we first introduce the definition of the k-SR variable node [29].

A punctured variable node v is called 1-step recoverable (1-SR) if there is at least one connected check node c, called survived check node, such that all other variable nodes connected to c are not punctured except for v. It is called 1-SR because such a punctured variable node can be recovered in one iteration on binary erasure channel

(BEC). An example of 1-SR node is shown in Fig. 4.1(a). The punctured variable node $v1$ is connected to two check nodes, i.e. node $c1$ and $c3$. One of these two check nodes, node $c1$, has four neighboring variable nodes, namely $v1$, $v2$, $v6$ and $v7$. Except for node $v1$, all other variable nodes, $v2$, $v6$ and $v7$, are not punctured. Therefore, variable node $v1$ is called 1-SR node.

The definition of k-SR node can be extended from 1-SR node such that at least one connected check node c, called the survived check node, contains one or more $(k-1)$-SR variable nodes while others are m-SR node, where $0 \leq m < k-1$ [29]. On BEC, a k-SR node can be recovered in kth iteration. Obviously, a k_1-SR node is more reliable than a k_2-SR node if $k_1 < k_2$. An illustration of k-SR node is shown in Fig. 4.1(b).

4.2.3 Selected Puncturing Scheme

Since a k_1-SR node is more reliable than a k_2-SR node if $k_1 < k_2$, we first maximize the size of the 1-SR group G_1 in order to minimize the error performance loss of the punctured codes. Then we try to maximize the size of the 2-SR group G_2, and so on. In other words, for the groups $\{G_1, G_2 \cdots G_n\}$ the punctured scheme will puncture the low indexed group first.

The puncturing procedure can be divided into two steps: (1) the punctured block selection and (2) the punctured bits selection inside a block.

First, we will investigate how to select punctured blocks. As an example, we present an efficient puncturing scheme by puncturing the rate-1/2 2304-bit mother code from WiMax standard [4], as shown in Fig. 4.2. It includes 1152 parity bits which are partitioned into 12 blocks with the block size of 96 bits. Due to the zigzag pattern of the parity structure, the grouping of k-SR node becomes easy to handle [13]. For example, if a rate-2/3 LDPC code is obtained by puncturing the mother

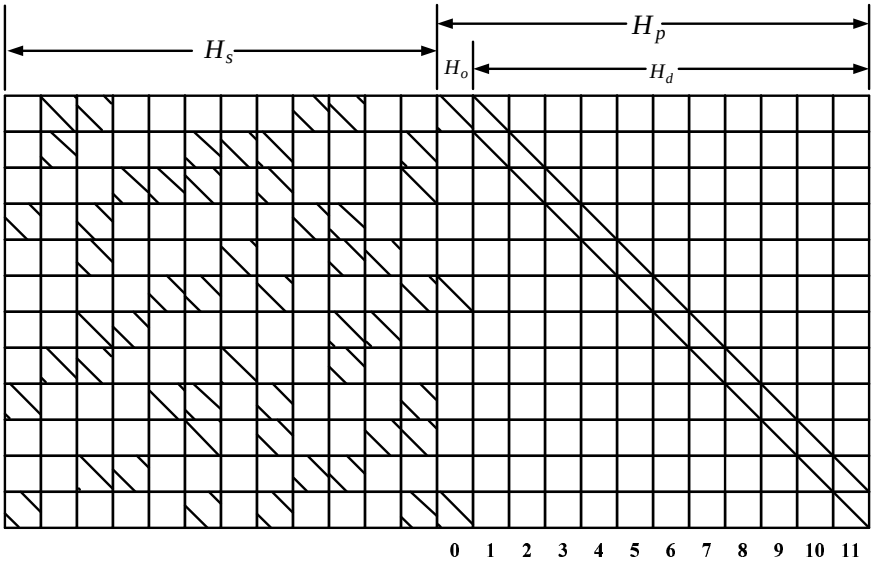


Figure 4.2: Parity-check matrix for the selected rate-1/2 LDPC code in WiMax.

code, half of the parity bits, i.e., 6 blocks (or 576 bits) are punctured. Table I shows three puncturing schemes in which the 6 punctured blocks are all 1-SR nodes. PBI denotes the punctured block index and SC denotes the number of survived checks corresponding to each punctured block.

Despite of the same number of 1-SR nodes, the error performances of the three schemes still differ from each other, as shown in Fig. 4.3. The number of survived checks, as listed in Table 4.1, has an impact on the error performance of the punctured code. Scheme 1 and scheme 3 both have more survived checks than scheme 2. Therefore, the BER performances of scheme 1 and scheme 3 are better than that of scheme 2.

Scheme 1 is the puncturing scheme proposed in [58], in which the weight-3 block, i.e., $\mathbf{H}_o$, is not punctured. In contrary, scheme 3 selects the weight-3 block for puncturing. Simulation results in Fig. 4.3 show that scheme 3 has better error performance than scheme 1 over AWGN channels. This is largely because the punctured weight-3 variable nodes have more neighboring checks which can provide more in-

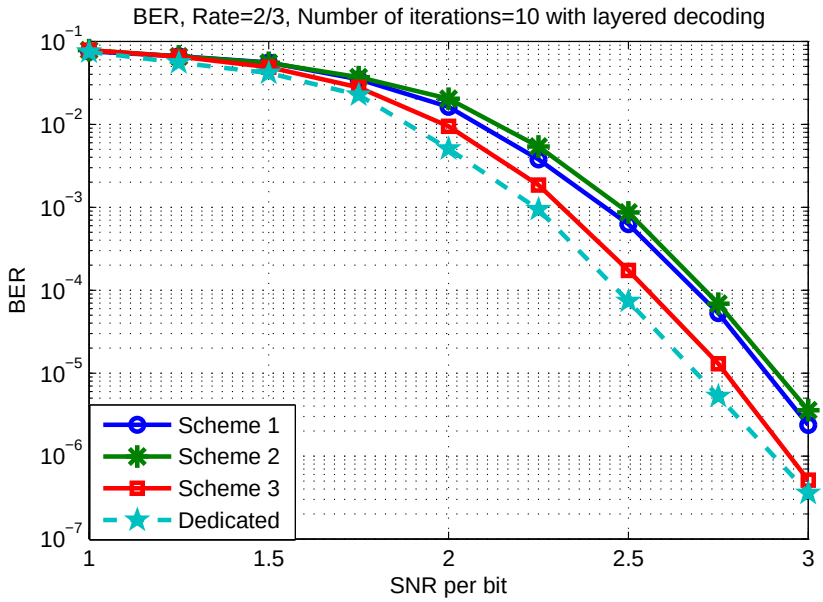


Figure 4.3: BERs of the three punctured codes and the dedicated code at rate 2/3 over AWGN channels.

Table 4.1: Three puncturing schemes for achieving rate 2/3 from rate 1/2 mother code

Scheme 1		Scheme 2		Scheme 3	
PBI	SC	PBI	SC	PBI	SC
1	2	0	1	0	2
3	2	1	1	2	2
5	2	3	1	4	2
7	2	4	1	6	1
9	2	6	1	8	2
11	2	7	1	10	2
Total SC	12		6		11

Table 4.2: Index of punctured blocks at different desired rates

Num	Rate	Punc Bits	Punc Blk Idx (PBI)	Surv Blk Idx (SBI)
1	1/2 - 12/23	1 - 96	*0*	*0*,1,2,3,4,5,6,7,8,9,10,11
2	12/23 - 6/11	97 - 192	0, *10*	1,2,3,4,5,6,7,8,9,*10*,11
3	6/11 - 4/7	193 - 288	0 2, *10*	1,3,4,5,6,7,8,9,*10*,11
4	4/7 - 3/5	289 - 384	0,2,8,*10*	1,3,4,5,6,7,9,*10*,11
5	3/5 - 12/19	385 - 480	0,2,4,8,*10*	1,3,5,6,7,9,*10*,11
6	12/19 - 2/3	481 - 576	0,2,4,6,8,*10*	1,3,5,7,9,*10*,11
7	2/3 - 12/17	577 - 672	0,2,4,6,8,*9*,10	1,3,5,7,*9*,11
8	12/17 - 3/4	673 - 768	0,*1*,2,4,6,8,9,10	*1*,3,5,7,11
9	3/4 - 4/5	769 - 864	0,1,2,4,*5*,6,8,9,10	3,*5*,7,11
10	4/5 - 6/7	865 - 960	0,1,2,4,5,6,*7*,8,9,10	3,*7*,11
11	6/7 - 12/13	961 - 1056	0,1,2,*3*,4,5,6,7,8,9,10	*3*,11
12	12/13 - 1	1057 - 1152	0,1,2,3,4,5,6,7,8,9,10,*11*	*11*

formation during the decoding iterations. Thus, puncturing the weight-3 block as in scheme 3 is recommended. Table 4.2 shows the selected puncturing scheme when 1, 2, ... 12 blocks are punctured, which corresponds to a group of code rates of 12/23, 12/22, ... 12/12. Here we name this group of rates as "block rate".

If the desired rate is between two consecutive block rates, we can puncture a block partially by taking some bits out of a block. In "Punc Blk Idx (PBI)" column of Table 4.2, the normal numbers indicate the entire blocks are punctured and an italic number indicates the designated block which may be punctured partially if needed. The punctured bits within that block are selected based on the following procedure [58]. In order to disperse the punctured bits within a block, a special sequence u_z is generated recursively from u_1 as in (4.2) through (4.3) and (4.4), where z is the size of the submatrix.

$$u_1 = \{0\}\,, \tag{4.2}$$

$$u_{2k} = \{u_k(0),\ u_k(0)+k,\ u_k(1),\ u_k(1)+k,\ \cdots,\ u_k(k-1),\ u_k(k-1)+k\}, \tag{4.3}$$

$$u_{2k+1} = \{k,\ u_k(0),\ u_k(0)+k+1,\ \cdots,\ u(k-1),\ u_k(k-1)+k+1\}, \tag{4.4}$$

Next, the actual punctured bit sequence u'_z is adjusted from u_z based on the values of b_0, l and q using (4.5), where l is the row number of $\mathbf{P_p}$, b_0 is the permutation value of $\mathbf{P_{b0}}$ and q is the permutation value of $\mathbf{P_q}$ in the weight-3 submatrix $\mathbf{H_o}$ from (4.1). For the selected rate-1/2 WiMax code, $z = 96$, $q = 7$ and $l = 6$.

$$u'_z = \begin{cases} mod(b_0 u_z,\ z) & (if\ PBI \leq l) \\ mod((z-q)\,u_z,\ z) & (if\ PBI > l) \end{cases} \tag{4.5}$$

Note that u'_z is a sequence with length of z, and each element $u'_z(i)$, $i = 0, 1, ..., z-1$, indicates the column index of the bit to be punctured within the block. In practice, the sequence is computed and stored in a LUT. As soon as the number of punctured bits is calculated from the given rate, the indices for the punctured blocks and the bits of the partially punctured block can be looked up instantly. More details will be discussed in the decoder implementation.

4.3 High-Throughput Rate-Compatible LDPC Decoder Architecture

4.3.1 Summary of the Parallel Layered Decoding Architecture

LDPC codes can be effectively decoded using belief-propagation (BP) algorithm [24]. Two phases of messages, check-to-variable (CTV) messages and variable-to-check (VTC) messages, are transmitted along the edges of Tanner graph to update each other iteratively. Min-sum algorithm and modified min-sum algorithm [23, 26, 10] have been introduced to reduce the complexity of CTV message updating.

Layered decoding algorithm [51, 9, 25, 45, 33, 28] has been adopted to reduce the number of iterations by a factor of two, compared with the standard BP algorithm. Hence, the decoding throughput can be improved without any bit error performance loss. In BP algorithm, VTC updates do not start until all of the CTV messages are received and vice verse. In horizontal layered decoding algorithm, the updated CTV messages from the current layer are passed vertically to all layers below for the same variable node. In each iteration, the horizontal layers are processed sequentially from the top to the bottom layers.

The overall decoding procedure for a $m \times n$ parity check matrix with min-sum and layered decoding algorithm is summarized as follows:

$$q_{j,n_j}[k] = \Lambda_{n_j}[k] - r_{j,n_j}[k] \quad (n_j \in N(j)) \tag{4.6}$$

$$r_{j,n_j}[k] = \left(\prod_{n'_j \in \{N(j) \setminus n_j\}} sign\left(q_{j,n'_j}[k]\right) \right) \times \left(\min_{n'_j \in \{N(j) \setminus n_j\}} \left\{ \left| q_{j,n_j}[k] \right| \right\} \times \alpha \right) \quad (4.7)$$

$$\overline{\Lambda_{n_j}}[k] = q_{j,n_j}[k] + r_{j,n_j}[k] \quad (4.8)$$

The details about layer decoding algorithm can be referred to [51, 9, 25, 45, 33, 28]. However, traditional layered decoding algorithm processes layers in sequential order, which results in a large decoding latency per iteration. A method of increasing parallelism inside a layer is proposed in [51, 45], but all layers are still processed in series. Although decoding throughput can be improved, this method introduces crossbar-based interconnection networks that increase the hardware complexity.

In Chapter 3 (also see [83]), a parallel layered decoding architecture was proposed which allows all layers to be processed in parallel. Each layer has an individual CNU which generates and sends updated messages and at the same time also receives the updated messages from other layers. Unlike the method proposed in [51, 45], parallel layered decoding architecture uses parallel processing among all layers and serial processing within each layer. In parallel layered decoding, the message passing routes among layers are based on their permutation values as in the parity-check matrix. Fig. 3.3 shows the values in the parity check of the rate 1/2 WiMax codes. For each vertical block, we first sort the permutation values of all layers in descending order. Subsequently, we designate each layer to pass its message to another layer which has the next smaller permutation value in the same column. Finally, the layer with the smallest permutation value loops back and connects to the layer with the largest value. This message passing scheme guarantees the updated messages

among all layered are processed progressively within each iteration. This designated message passing scheme works well, because the rows within each layer are processed sequentially and the updated messages are passed only to the layer who is going to process the same column next.

It is worth mentioning that we have tentatively added an offset to the permutation values for each layers as in Fig. 3.3, which is equivalent to show that the CNUs start to process from different rows (instead of already start from row 1). From the decoding prospective, changing the processing order in each layer does not affect the performance nor throughput of a decoder. In fact, the offset values are carefully selected such that the difference of the modified permutation values between any two layers is at least 5. It means that each layer (or CNU) has a time span of 5 clock cycles to read, update, store and then pass the message to the next connected layer. The main advantage is that we can design the CNUs, which are usually the critical path in decoder design, into 5-stage pipeline architecture. This is called critical path splitting technique in 3, which reduce the latency of the critical path and thus improve the clock speed and decoding throughput.

The major issue of the aforementioned parallel layered decoding architecture is that the concurrent message passing routes among all layers are fixed and optimized for the specific code. As mentioned in Section 4.1, rate compatible LDPC codes or at least multiple rates are desirable for communication systems on time-varying channels. Thus, we investigate various puncturing schemes and incorporate the rate compatibility into parallel layered decoding architecture to provide the much needed flexibility, without sacrificing its advantage of high throughput.

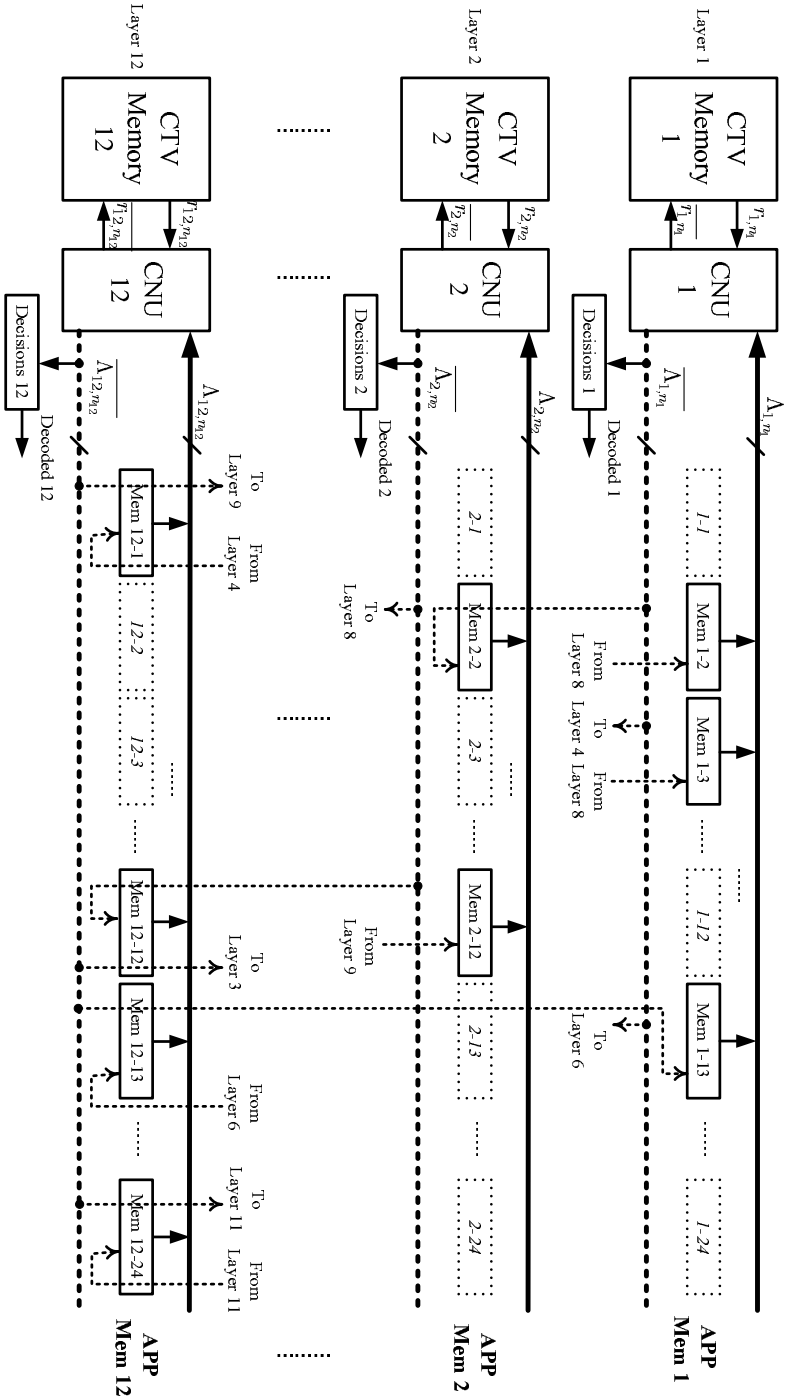


Figure 4.4: Overall architecture of a rate-compatible LDPC decoder.

4.3.2 RC LDPC Decoder Design

The overall architecture of the RC LDPC decoder is shown in Fig. 4.4. It consists of CNUs, LLR initialization block, APP memory banks, CTV memory banks and bit decision units. It is similar to a regular PLDA design except for the initialization of the APP memory. Using the rate-1/2 WiMax code as an example, the entire $\mathbf{H_b}$ matrix is divided into 12 layers and each layer has a dedicated CNU. APP memory banks and CTV memory banks are used to store APP messages and CTV messages. Each nonzero elements in the base parity check matrix corresponds to one APP memory unit and one CTV memory unit. Each APP memory exports APP messages to the CNU and each CTV memory also exports CTV messages to the CNU. The CNU first calculates the VTC messages (as indicated in (4.6)), then calculate the updated set of CTV and APP messages (as indicated in (4.7) and (4.8)), and finally imports updated CTV message to CTV memory banks and updated APP messages to APP memory banks. Therefore, the CNU becomes the critical path of the PLDA which limits the maximum frequency.

However, as indicated in Section 4.3.1, an interval of 5 clock cycles is available for the APP message passing from one layer to another if offset-modified parity-check matrix in Fig. 3.3 is used. Considering the memory writing operation which will cost one clock cycle, a split CNU with 4 pipeline stages can be designed by inserting some registers to the original CNU, bringing in reduced critical path delay and improved maximum frequency.

The puncturing scheme is implemented by patching the LLRs of the punctured parity bits to be 0's, which is called LLR initialization. The length of the actual received codeword is smaller than that of the original mother code because of the punctured bits. For the selected puncturing scheme, the punctured blocks and bits listed in Table II should be initialized to zeros.

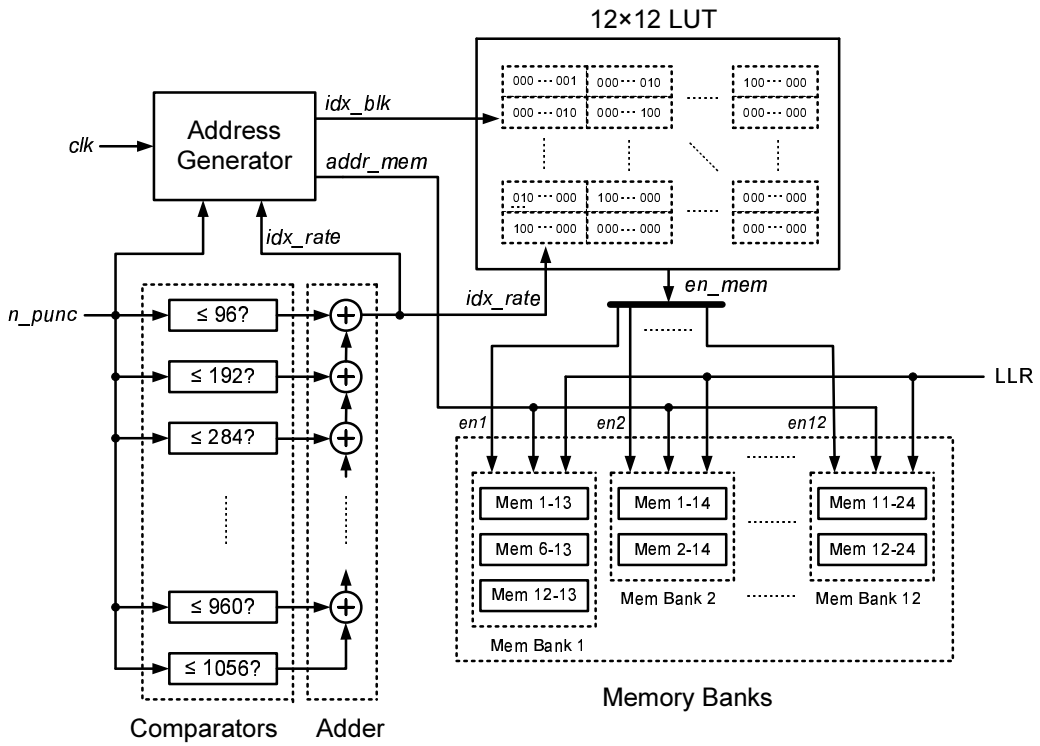


Figure 4.5: Architecture of LLR Initialization Block.

Fig. 4.5 shows the architecture of the LLR Initialization Block, including a group of *Comparators*, a *Decoder*, an *Address Generator* and a 12×12 look-up table (LUT). The LLRs from the channel are stored in assigned memory banks and the others are set to be 0's. Based on the length of the punctured bits (n_punc), the rate of the code can be deduced using a group of *Comparators* and an *Adder*. The thresholds of the comparators are set to be the multiples of the sub-matrix size z, which is 96 for the mother code. Each comparator compares n_punc with one threshold and returns 1 if n_punc is greater and 0 otherwise. Totally there are 11 comparison results. These results are sent to a decoder to determine the range of the punctured code. Here the decoder is simply composed of a group of adders which add all of the comparison results to get the idx_rate signal. For example, if the length of the punctured bits is smaller than 96, then each comparator returns a

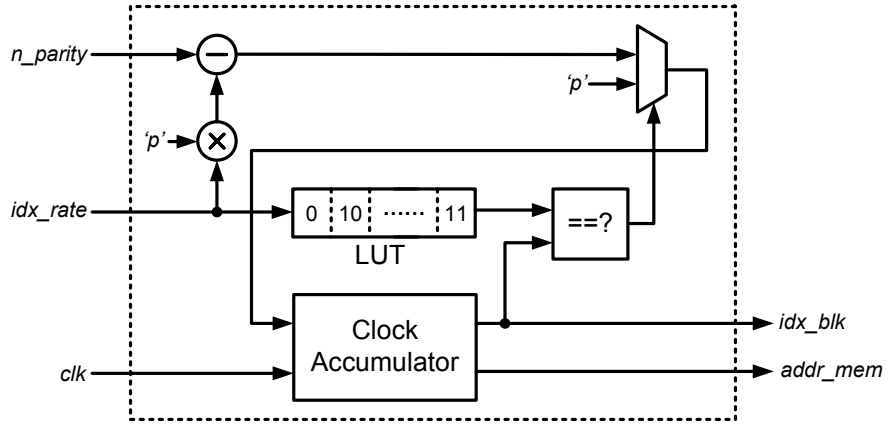


Figure 4.6: Architecture of the Address Generator.

0 and *idx_ rate* is therefore 0. This corresponds to row 1 in Table 4.2 and the rate lies between 1/2 and 12/23. The *Address Generator* generates addresses for the memory banks, as well as the 12×12 LUT. Signal *idx_ blk* denotes which memory bank is being written with the LLRs at a time instant. The 12×12 LUT stores the enable signal for the memory banks to decide which memory should be written by LLRs at a time constant. Thus, the contents of the 12×12 LUT represent the puncturing bit selection as in Table 4.2.

The detail design of the *Address Generator* is shown in Fig. 4.6. The core component is an accumulator which accumulates every clock cycle and outputs two signals *idx_ blk* and *addr_ mem*, one for the 12×12 LUT to select the enable signal and another for the memory banks as the write address. However, it can be observed from Table II that for every rate range, only one vertical block is partially punctured and the rest are entirely punctured. In other words, the clock accumulator will accumulate every 96 cycles unless for the partially punctured block that it accumulates at a smaller number of cycles which is determined by the number of punctured bits in this block. Therefore, a LUT is used here to store the index of the partially punctured block based on Table II, in order to indicate the clock accumu-

lator to accumulate at a different step when meeting with the partially punctured block. The content of the LUT is exactly the italic numbers in Table 4.2.

4.4 Experimental Results

For experimental study, we implement the selected puncturing scheme for the WiMax LDPC codes. We choose the rate-1/2 LDPC code in the WiMax standard as the mother code. Numerical simulations are performed to verify the BER performance between the punctured codes and the dedicated codes at three different rates. Furthermore, the rate compatible LDPC decoder are developed based on the parallel layered decoding architecture and then implemented using standard cell ASIC design flow.

4.4.1 Simulation Results for Punctured WiMax Codes

The BER performance of a group of punctured LDPC codes are presented in Fig. 4.7. The rate of mother code is 1/2 and the code length is 2304 bits. Five different rates are generated using the selected punctured schemes, i.e., 3/5, 2/3, 3/4, 5/6 and 6/7. The corresponding number of punctured bits are 384, 576, 768, 922 and 960. Thus the code length of the punctured codes are 1920, 1728, 1536, 1382 and 1344, respectively.

To verify the selected puncturing scheme, dedicated LDPC codes at rate 2/3, 3/4 and 5/6 from WiMax standard are simulated to compare their performance with the punctured codes, also shown in Fig. 4.7. At each rate, a specific mode is selected from the 19 modes of each WiMax code to make the selected code lengths of the dedicated codes equal or similar to those of the punctured codes. The code lengths of the dedicated codes at rate 2/3, 3/4 and 5/6 are adjusted to 1728, 1536

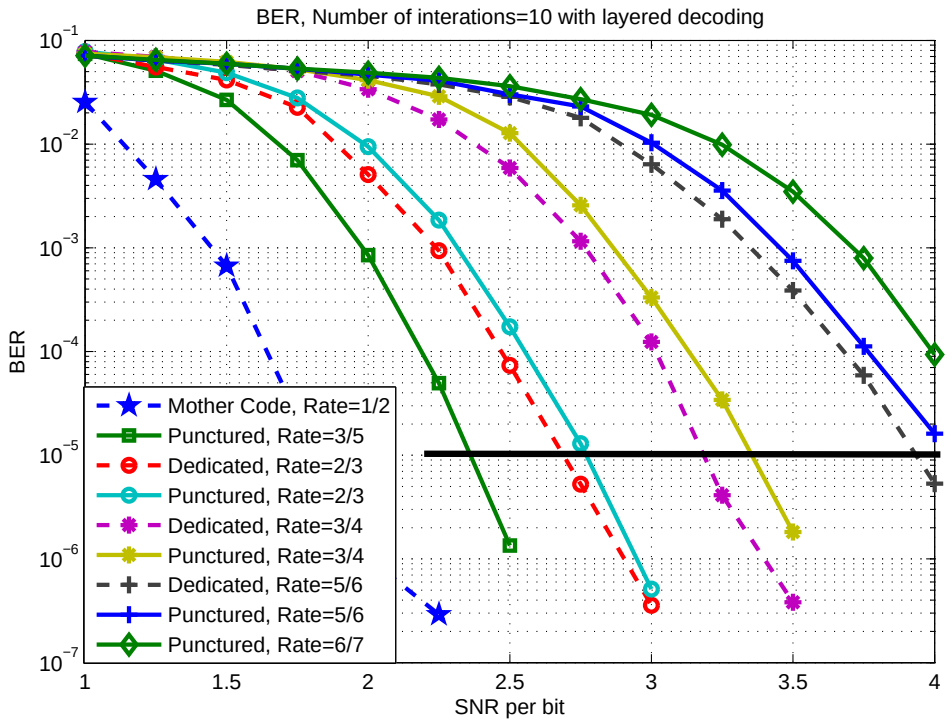


Figure 4.7: BERs of the punctured LDPC codes over AWGN channels.

and 1344, equivalent or similar to those of the corresponding punctured codes whose code lengths are 1728, 1536 and 1382. Fig. 4.7 shows that the BER of the punctured code is very close to the dedicated code, with less than 0.2dB performance loss at BER of 10^{-5}.

4.4.2 Hardware Implementation Results

In order to demonstrate the combined system performance of the selected rate compatible LDPC codes and the PLDA architecture, we implement the rate LDPC decoder in TSMC 90 nm technology with 8 layers. We complete the synthesis and core area place and route using the standard Synopsys tools.

Implementation results show that the decoder can operate at a maximum frequency of 838 MHz after synthesis, which corresponds to a constant input throughput of 975 Mbps for all code rates. Fig. 4.8 shows the layout view of the decoder

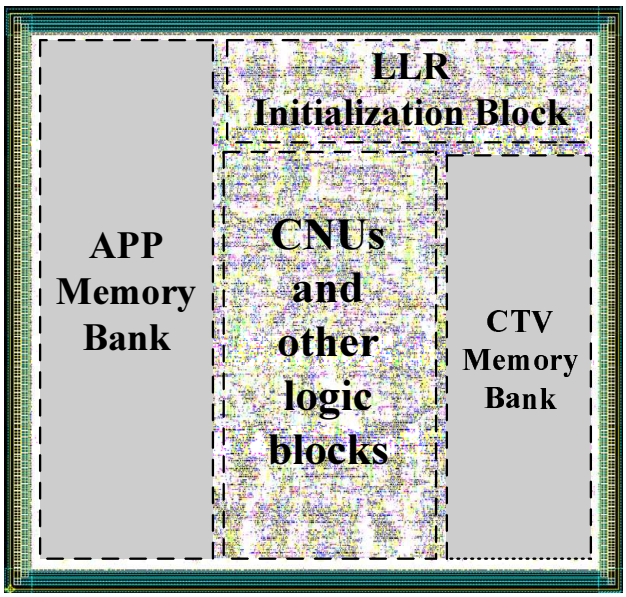


Figure 4.8: Layout of the proposed decoder chip.

with the core area of 1.96 mm^2 and the logic density of 70%. Read/write memories are generated by Synopsys DesignWare tool and thus flattened during synthesis and place and route process. The estimated power consumption of the decoder core is 650 mW running at 838 MHz clock frequency.

We also compare the proposed RC LDPC decoder design with several other existing WiMax LDPC decoder implementations, as listed in Table 4.3. For fair comparison, we first scale all the decoders to 65 nm technology node. Then, a metric called the throughput-to-area ratio (TAR) is introduced show how much throughput a decoder can achieve per area unit. Table 4.3 shows that the proposed decoder can provide higher throughput using smaller chip area. More interestingly, the proposed decoder design can provide any arbitrary code rate between 1/2 and 1 as opposed to only 4 selected rates in the existing WiMax LDPC decoders.

Table 4.3: Overall comparison between proposed decoder and other existing WiMax LDPC decoders

	C. Liu [44]	T. Brack [9]	X. Shih [66]	C. Liu [45]	Proposed
Supported Rates	1/2, 2/3, 3/4, 5/6	1/2, 2/3, 3/4, 5/6	1/2, 2/3, 3/4, 5/6	1/2, 2/3, 3/4, 5/6	Any rate between 1/2 and 1
Frequency (MHz)	300	333	83.3	150	1100
Iterations	20	10, 15	2~8	20	10
Throughput (Mbps)	212	83-155	60~220	105	1280
Technology (nm)	90	130	130	90	65
Area (mm^2)	6.22	3.83	8.29	6.25	1.96
Area scaled to 65 nm (mm^2)	3.24	0.96	2.07	3.26	1.96
TAR ($Mb \cdot s^{-1}mm^{-2}$)	65.4	86.5~161.5	29.0~106.3	32.2	653.1

4.5 Summary

This paper presents the algorithm, design and implementation of a rate-compatible LDPC decoder. Using the selected puncturing scheme, the BER performances of the punctured codes are comparable with the dedicated codes with less than 0.2 dB performance degradation in simulation results. In addition, rate compatible LDPC codes provide an ideal solution to the flexibility problem of the parallel layered decoding architecture. Considering the WiMax standard, a rate compatible LDPC decoder is designed using the rate-1/2 code as mother code. The hardware implementation shows the maximum input throughput of 975 Mbps. Comparing to a multi-rate LDPC decoder, the rate compatible design can eliminate the memory to store multiple codes and the network to switch among them. Therefore, rate-compatible LDPC coder are highly desirable for advanced wireless communication systems.

Chapter 5

Low-Complexity LDPC Decoder Architecture for CMMB Systems

5.1 Introduction

Mobile digital broadcasting TV is an emerging area for next-generation multimedia communications and entertainment service. Several communication standards have already taken place worldwide, including DVB-H [21] in Europe, T-DMB [2] in Korea, and ISDB-T [3] in Japan. The China Multimedia Mobile Broadcasting (CMMB) standard [1, 76], ratified in 2006, is the mobile television and multimedia standard developed and specified by the State Administration of Radio, Film, and Television (SARFT) in China. Both high data rate and high reliability are desirable for broadcasting networks, which requires high-throughput forward error correction (FEC) codes with excellent error correcting performance.

LDPC code is therefore an ideal candidate due to its near-shannon-limit error performance and inherent parallelism for parallel implementation. They have been chosen as the ECC for CMMB together with Reed-Solomon (RS) codes. The mo-

bility requirement of CMMB standard demands for an area efficient and low-power LDPC decoder. In recent years, many research projects have been focused on reducing memory size and complexities of node processing units and interconnection network in LDPC decoders [85, 51, 36, 78].

With the increasing popularity of mobile handheld devices, it demands for a low-complexity FEC decoder that is both area efficient and low power. In recent years, many research projects have been focused on the reduction of memory size and simplifying the complexity of interconnection network in an LDPC decoder. Memory-efficient architectures are constructed by employing min-sum algorithm [78] and memory-aware architecture [51]. Complicated interconnects can be reduced by the partially parallel architectures [85, 51] and further optimized by the loosely coupled algorithm [36]. While layered decoding architectures addressing weight-1 matrices are presented in [45], the novelty of this paper lies in the low-complexity architecture design for layered decoding with weight-2 matrices.

In this chapter, we present the architecture and implementation of an LDPC decoder for CMMB standard. The main contributions of this paper are listed as follows: (1) A reconfigurable architecture, which can support dual rate LDPC codes specified in the CMMB standard, is constructed with minimal overhead; (2) Split-memory architecture is proposed to efficiently handle the weight-2 superimposed sub-matrices. The proposed techniques takes advantages of the regular structure in both rate 1/2 and 3/4 codes in CMMB standard and they apply well to the codes with a few weight-2 sub-matrices.

The rest of this chapter is organized as follows. The QC-LDPC code structure in the CMMB standard and its decoding algorithms are introduced in Section 5.2. The reconfigurable architecture of the dual-rate LDPC decoder design is presented in Section 5.3. Memory reduction technique and simplified read/write networks are

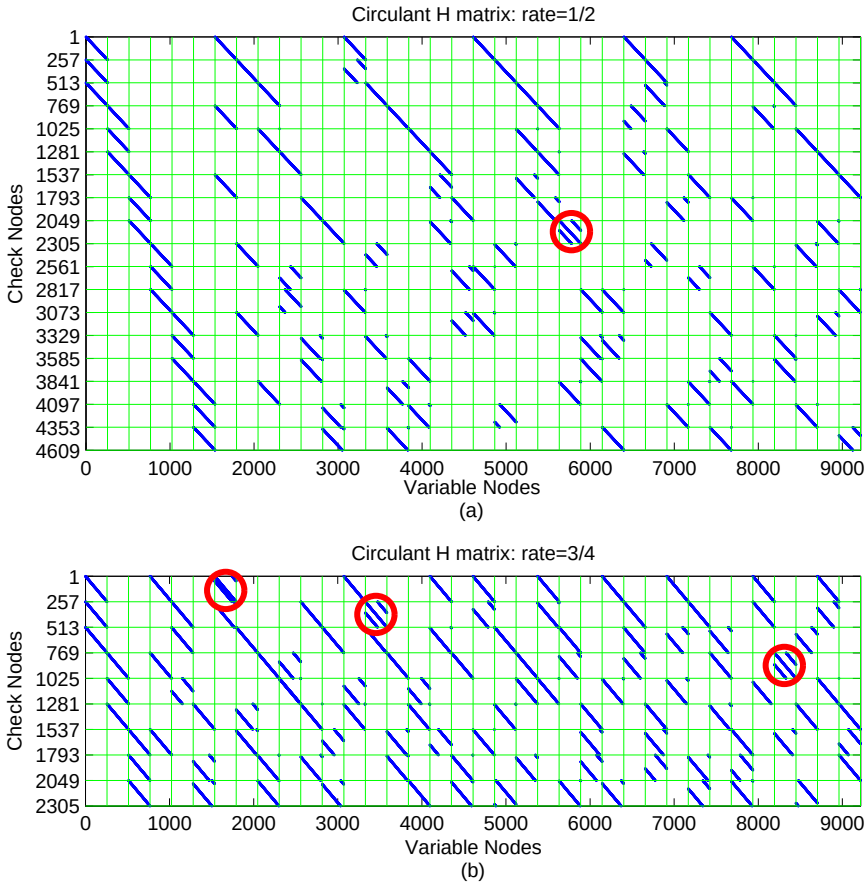


Figure 5.1: Structure of the parity check matrix for LDPC codes in CMMB standard: (a) rate-1/2; (b) rate-3/4.

discussed in Section 5.4. Section 5.5 shows the ASIC implementation of the decoder and its performance results, followed by the summary in Section 5.6.

5.2 QC-LDPC Codes in CMMB Standard

Fig. 5.1 shows the structures of the base matrix $\mathbf{H}$ for the LDPC codes defined in the CMMB standard, including the rate-1/2 code and the rate-3/4 code. They belong to the class of QC-LDPC codes that can be expanded from the base matrix, and the size of each sub-matrix is 256×256. In fact, the original parity check matrix in the CMMB standard is regular but not well structured. We obtain an equivalent

QC form as shown in Fig. 5.1 through proper column permutations. The size of the base matrix $\mathbf{H}$ is 18×36 for the rate-1/2 code and is 9×36 for the rate-3/4 code. A blank element can be expanded as a 256×256 all zero sub-matrix. A non-blank element can generally be expanded as a 256×256 circularly shifted identity matrix, also called weight-1 sub-matrix, except for a few weight-2 sub-matrices. As highlighted in Fig. 5.1, there is one weight-2 sub-matrix in the rate-1/2 code and three of them in the rate-3/4 code. To the best of our knowledge, the QC-LDPC decoder with weight-2 sub-matrices has not been much studied. Most of the existing works were focused on QC codes with all weight-1 sub-matrices, e.g. [45].

Both rate-1/2 and 3/4 codes in CMMB standard are regular LDPC codes with code length of 9216 bits and column weight of 3. Row weight of the rate-3/4 code is 12, twice of the row weight of the rate-1/2 code which is 6. Therefore, the total number of none-zero elements in both codes are exactly the same, which is a unique property for the design of a unified decoder that can support dual rates as specified in CMMB standard.

5.3 Dual-rate Decoder Design

Before constructing the functional units of the decoder, we first perform fixed-point simulations to quantize the word-length of messages. In fact, word-length quantization is a tradeoff between memory size and bit-error-rate performance. Fig. 5.2 shows the BER performance of the LDPC codes in CMMB with CTV message representations in floating point and word-length $(3, 1)$ which corresponds to 3 bits of integer part and 1 bit of the fractional part. Considering a sign bit, we choose 5-bit fixed-point representation for the CTV messages in the design. APP messages are 6-bit representations in our design, including a sign bit, 4 bits of integer part and 1

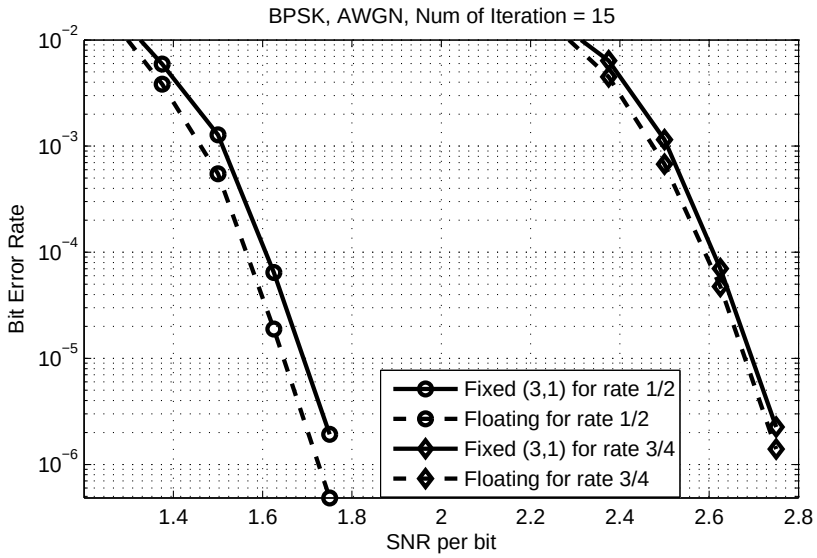


Figure 5.2: BER performance for different rates and quantization schemes.

bit of fractional part.

5.3.1 Overall Architecture

Fig. 5.3 shows the overall decoder architecture for CMMB LDPC codes. The *CTV memory* is used to store the CTV messages corresponding to one row in the **H** matrix of the QC-LDPC code. Instead of storing 6 CTV messages in each row (because the rate 1/2 codes have the row weight of 6), a concatenated message is stored consisting of four elements: the minimum magnitude; the second minimum magnitude; the relative location index of the minimum magnitude; and the signs of all messages in this row. Since there are 6 elements in each row for rate-1/2 code, only 3 bits are needed to represent the relative location index. The *recovery unit* is to recover the individual CTV message from its compressed form. The *APP memory bank* stores the APP messages for each column and it loads the intrinsic messages from the channel at initialization. The *read network* fetches the APP messages from APP memory bank according to the locations of nonzero elements in each row.

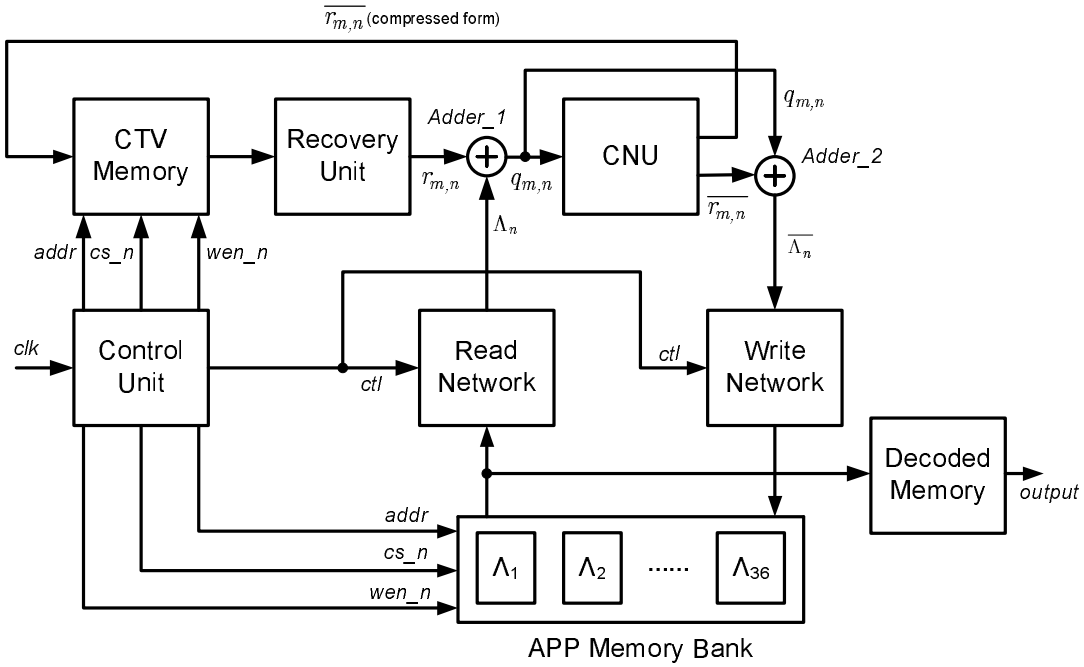


Figure 5.3: Overall architecture of the CMMB LDPC decoder.

Adder_1 recovers the VTC message as in (??) and then sends it to the CNU. The *CNU* reads the VTC messages from adder_1 and updates the CTV messages. The compressed form of the updated CTV messages is saved in the CTV memory, while in original form they are sent to *adder_2* for updating the APP messages as defined in (??). Subsequently, the *write network* stores the updated APP messages to the APP memory bank. The *decoded memory* stores the output bits after the decision.

5.3.2 Dual-Rate CNU Design

Before describing the dual-rate CNU architecture, we first elaborate CNU design for the rate-1/2 codes. Fig. 5.4 can be viewed as two rate-1/2 CNUs in parallel. For implementation of the min-sum algorithm, the CNU requires a 6-number comparator to find the minimum and second minimum values of the VTC messages in each row. Here we employ a pseudo-rank order filter (PROF) based design derived from a

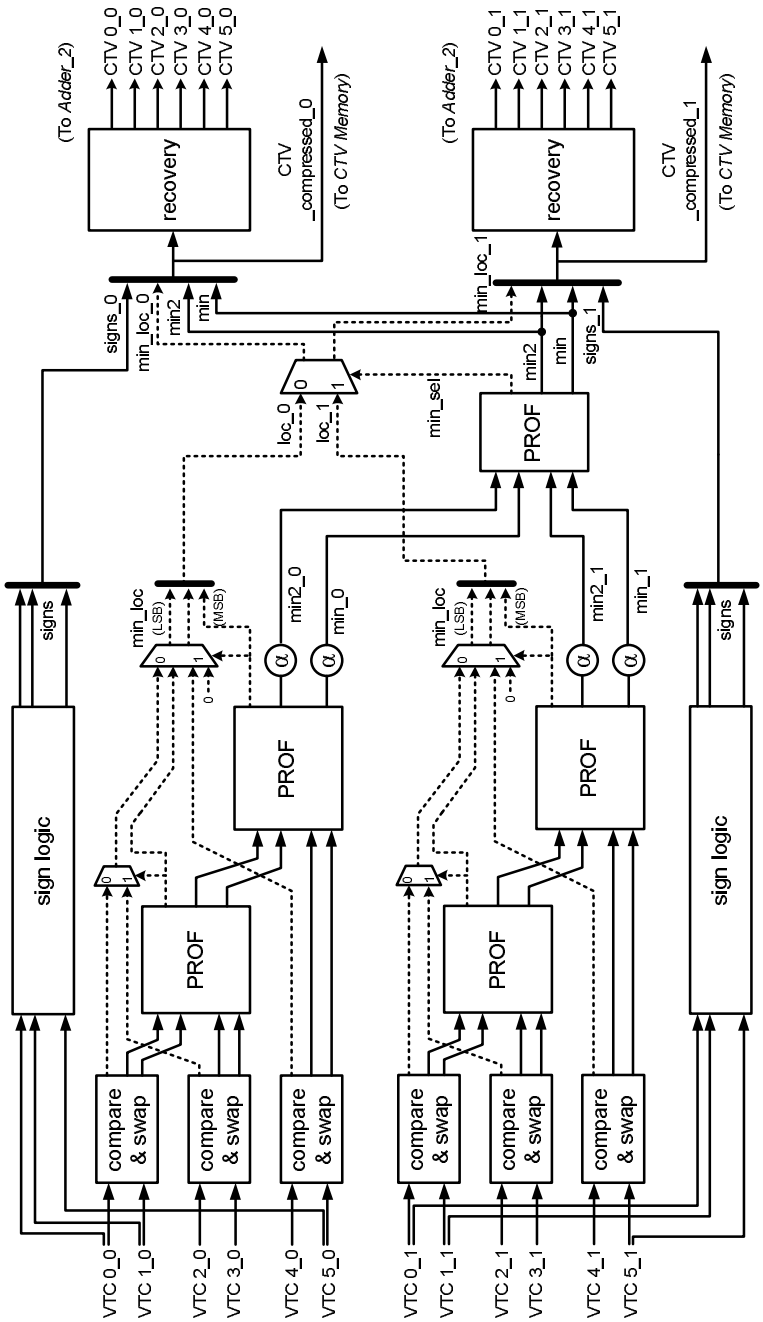


Figure 5.4: Architecture of CNU for dual-rate (1/2, 3/4) CMMB LDPC codes.

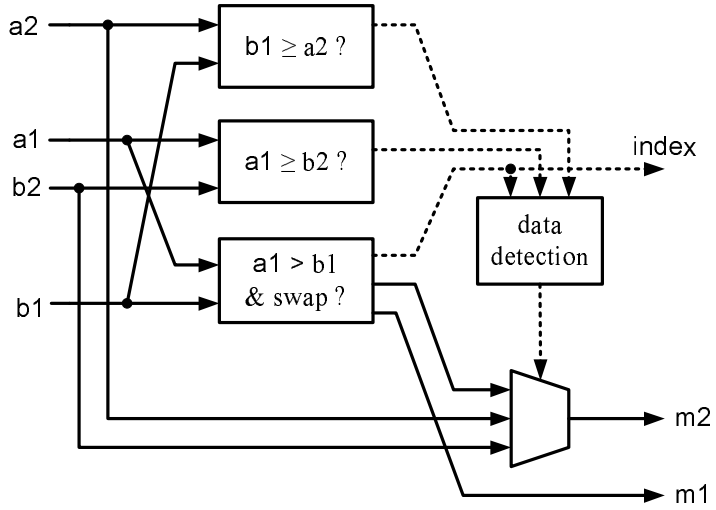


Figure 5.5: The design of an PROF-based comparator .

previous work [78], as shown in Fig. 5.3.2. The *PROF* can output the minimum and second minimum magnitudes of two pairs of inputs. For each pair, two elements are compared and the one with larger value is placed on the top. Thus, the *compare & swap* blocks are added before the *PROF* blocks to sort the magnitudes of each pair of inputs. To assist recording the positions of the minimum values, the compare & swap block outputs a bit value 0 if two inputs are swapped. Otherwise, it outputs 1 as shown in the dotted line. Similarly, the *PROF* block also indicates the location of the minimum magnitude by setting an output 0 if the minimum value belongs to the upper pair and 1 otherwise.

In order to achieve a fully functional LDPC decoder for CMMB standard, the proposed architecture must support both code rates of 1/2 and 3/4. Instead of introducing a large switch network as in other multi-rate LDPC decoder which increases the complexity and area, here we innovatively extend the rate-1/2 CNU design and make it compatible with rate-3/4 by adding a few extra multiplexers together with minor modification of several functional units. With minimum overhead, we can achieve a reconfigurable architecture for dual-rate decoding.

For the rate-3/4 implementation, the CNU has to deal with the row weight of 12, twice that of the rate-1/2 codes. The idea is to combine two rate-1/2 CNUs, namely CNU0 and CNU1, into a single rate-3/4 CNU design as shown in Fig. 5.4. Here we briefly elaborate the process of constructing the dual-rate CNU. For rate-3/4 codes, a 12-number comparator is required. Therefore, a row is divided into two sets, each of which has a weight of 6. Then, two CNUs each with a 6-number comparator are assigned to process two sets of CTV messages in parallel. As expected, each set produces the minimum and the second minimum magnitudes. Now we introduce an additional *PROF* block to compare these two sets of minimum magnitudes that belong to the same row (denoted as signal *min* and *min*2). Signal *min_sel* is set to 0 if the minimum value from CNU0 is selected and set to 1 if the minimum value from CNU1 is selected. The locations of these two sets of minimum values are denoted by signals *loc_0* and *loc_1*, respectively.

An additional multiplexer is introduced to assign the index location of the minimum value. If *min_sel* is set to 0, which indicates the minimum value of the row belongs to CNU0, the 3-bit location signal *loc_0* is passed to the upper recovery block through signal *min_loc_0*. The location signal of the lower recovery block is assigned with idle value 111, which indicates that the minimum magnitude is not contained in CNU1. Similarly, the location index is passed to the lower recovery block if CNU1 contains the minimum magnitude of the row.

As we can see from Fig. 5.4, the overall operations of the reconfigurable CNU remain the same. The introduced overhead is minimum and does not affect the decoder throughput or frequency of the circuits. Therefore, this is a very efficient approach to design a reconfigurable dual-rate decoder.

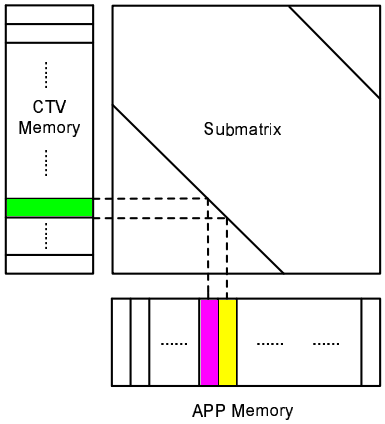


Figure 5.6: Element correspondence relations between CTV memory and APP memory.

5.3.3 Memory Access for Partially Parallel Layered Decoding

To increase the decoding throughput, multiple rows can be processed in parallel which is usually referred as partially parallel layered decoding. For parallelism factor p, it requires access of the CTV messages for p consecutive rows in one clock cycle. Therefore, we group p rows of CTV messages into a single entry in the CTV memory. Similarly, p columns of APP messages can also be grouped into a single entry in the APP memory. However, the alignment of row and column-based memory partitions may not match exactly, depending on the permutation value of the sub-matrix. As shows in Fig. 5.6, the p CTV messages from an entry of the CTV memory correspond to p APP messages that are stored in 2 adjacent locations in the APP memory. Therefore, a shift register can be used for storing and selection of the appropriate APP messages read from the memory.

The parallelism factor p is usually chosen as a number that divides l, where l is the size of the sub-matrix ($l = 256$ for CMMB codes). Practically, two APP memory locations can be read in 2 consecutive clock cycles and the data are stored in a shift register with the size of $2p$. The permutation value s_{ij} of the sub-matrix determines which p messages are to be selected. It can be proved that the corresponded p

messages are located between $(s\%p+1)$ and $(p-1+s_{ij}\%p)$ in the shift register, where the shift control signal assigned by $(s_{ij}\%p)$ has the bit width of $\lceil log_2(p) \rceil$.

5.3.4 Split-Memory Architecture for Weight-2 Sub-matrices

A major challenge of decoder design in CMMB standard is to handle the weight-2 sub-matrices in the **H** base matrix highlighted in Fig. 5.1. Each weight-2 sub-matrix is composed of 2 superimposed cyclic-shifted identity matrices, as opposed to 1 shifted identity matrix for weight-1 sub-matrix. Considering layered decoding, in [55], a simplified layered decoding algorithm is proposed to deal with such superposed sub-matrices without APP message passing inside each sub-matrix. To fulfill a complete layered decoding algorithm with weight-2 sub-matrices, the CNU needs to read two APP messages from two different memory locations at the same time. Similarly, two updated APP messages must be written back to memory in the same clock cycle. Therefore, a dual-read, dual-write memory is needed to support the operations. However, dual-port memories are not desirable for implementation because of increased area and power consumption.

Hereby, we propose a novel split-memory architecture that consists of two single-port memories to handle the weight-2 sub-matrix. Fig. 5.7 shows the memory structure. A weight-2 sub-matrix is decomposed into two weight-1 sub-matrices s_0 and s_1, and each uses a single-port memory to store the APP messages. One memory stores APP messages corresponding to permutation value s_0 and another to permutation value s_1. Without loss of generality, we suppose $s_0 < s_1$. Each row i corresponds to two columns j_0 and j_1. Two APP messages from column j_0 and j_1 are sent to the CNU through the read network, while the updated APP messages $\overline{\Lambda_{j_0}}$ and $\overline{\Lambda_{j_1}}$ are interchanged before being stored in the other memory, as shown in Fig. 5.7(a). In this case, it can be guaranteed that the CNU receives

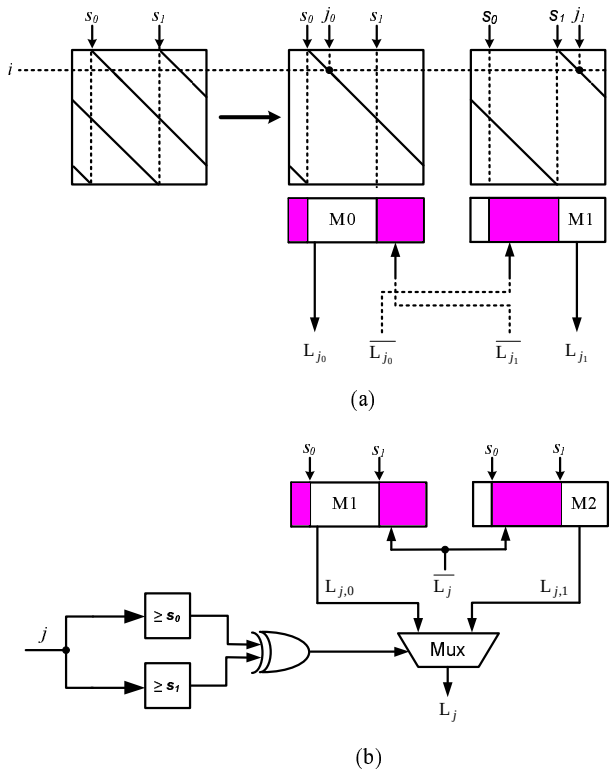


Figure 5.7: Split-memory design for handling the weight-2 sub-matrices.

the most updated APP messages within the same iteration. More specifically, the processing of columns $[s_1, l-1]$ and $[0, s_0-1]$ for sub-matrix s_0 incorporates the newest APP messages that have just been updated as in the sub-matrix s_1. Similarly, the processing of columns $[s_0, s_1-1]$ for sub-matrix s_1 involves the newest APP messages updated as in the sub-matrix s_0. Upon completion of the processing of the weight-2 sub-matrix, the newest APP messages corresponding to columns $[s_1, l-1] \cup [0, s_0-1]$ are stored in memory $M0$ and the newest APP messages corresponding to columns $[s_0, s_1-1]$ are stored in memory $M1$. When processing many other regular weight-1 sub-matrices, a multiplexer is used to read the correct APP messages for the columns involved with the weight-2 sub-matrices. As in Fig. 5.7(b), the APP messages from memory $M0$ is selected if the column index $j \in [s_1, l-1] \cup [0, s_0-1]$. Otherwise, APP messages from $M1$ is selected if $j \in [s_0, s_1-1]$.

The key advantage of the proposed split-memory architecture is that it makes the processing of weight-2 sub-matrices the same as that of regular weight-1 sub-matrices. It does not introduce additional delays to slow down the decoding throughput. Moreover, there is no performance lost for layered decoding using the proposed split-memory architecture.

5.3.5 Number of Pipeline Stages

For the overall decoder architecture shown in Fig. 5.3, there is a long data path beginning from the APP memory, through read network, CNU, two adders, write network, and returning to the APP memory. In order to improve the overall clock frequency, this path is split into several pipeline stages. After reading the APP message from memory, it should allow multiple clock cycles to complete the CNU processing, update the APP messages for the current layer and store them back to

the memory. In fact, it is not necessary to immediately write back the updated APP messages unless the same columns are used again during the processing of following layers. However, the number of the pipeline stages can not be arbitrarily determined because of the constraint brought by the weight-2 sub-matrices during layered decoding.

The weight-2 sub-matrices are composed of two overlapped weight-1 sub-matrices with the permutation values of s_0 and s_1. Without loss of generality, we suppose that $s_1 \geq s_0$. At row i, it corresponds to two columns and the difference of two column indices is calculated as

$$\triangle_i = \begin{cases} s_1 - s_0, Fo & i \in [0,\, l - s_1 - 1] \cup [l - s_0 - 1,\, l - 1] \\ l - (s_1 - s_0) & i \in [l - s_1,\, l - s_0 - 2] \end{cases} \tag{5.1}$$

The minimal value of $\triangle_i$ is denoted as $\triangle_{min}$. For each weight-2 sub-matrix, two columns are processed currently. As described in the split-memory architecture in Section 4.2, the columns that are processed by sub-matrix s_0 first must be stored back to the APP memory before sub-matrix s_1 starts to process the same column again, and vice verse for the columns processed by $s_1$first. Considering the partially parallel architecture that processes p rows each time, the property of the weight-2 sub-matrix requires APP message to be updated and stored back within $\lfloor \triangle_{min}/p \rfloor$ clock cycles. In CMMB LDPC codes, there are four weight-2 sub-matrices, including 1 in the rate-1/2 codes and 3 in the rate-3/4 codes as shown in Fig. 5.1. The permutation values of the four sub-matrices are (0, 146), (0, 208), (0, 145), and (0, 138), respectively. Thus $\triangle_{min}$ is 48, which corresponds to the sub-matrix at row 0 column 7 in the **H** base matrix of the rate-3/4 codes. If we choose the parallelism factor as 8, then the number of pipeline stages is 6. The decoder implementation with 6-stage pipeline shows that it can reduce the critical path delay and results in

high clock frequency.

5.4 Area-Efficient Design Techniques

5.4.1 Memory Reduction

Memory usually dominates the overall chip area and power consumption in an LDPC decoder [51] because iterative messages at each check nodes and variable nodes must be stored for the processing in subsequent iterations. Min-sum algorithm simplifies the storage requirement at the check node. As mentioned earlier, the CTV messages for a row are saved in a compressed form that includes the minimum magnitude, the second minimum magnitude, the relative location index of the minimum magnitude and the signs of the CTV messages in this row. As an example, the row weight of rate-1/2 codes is 6 and each CTV message is quantized into 5 bits. For m rows, the size of the CTV memories can be reduced from $30m$ bits to $17m$ bits, which is about 43% memory reduction.

In addition, the proposed layered decoding algorithm only requires the storage of APP messages instead of the actual VTC messages. It reduces the memory size since each column only needs to store one APP message. For example, the LDPC codes in CMMB has a column weight of 3, which indicates that the required memory to store variable node information is reduced by 66.7%.

5.4.2 Read/Write Networks

As the key part of the decoder, read and write networks control the messages passing between APP memory and CNU. In general, a structural (m, n) code such as the QC-LDPC code in CMMB can reduce a large $m \times n$ read and write networks to a

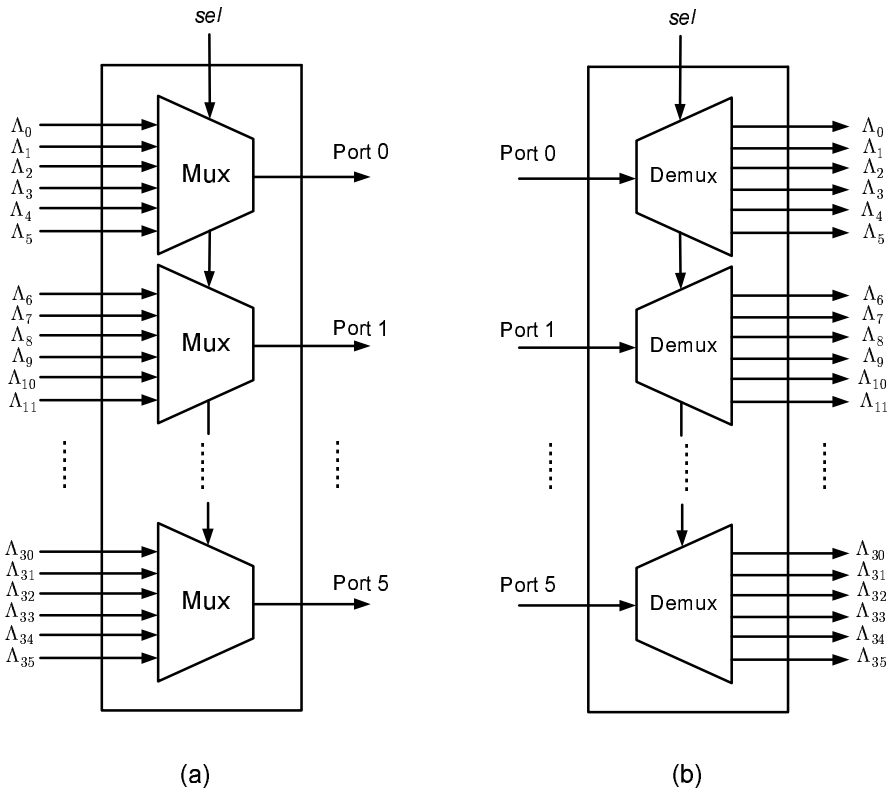


Figure 5.8: For rate-1/2 codes (a) Architecture of read network; (b) Architecture of write network.

Table 5.1: Connections of input and output ports in the read network for the rate-1/2 codes

Input Ports	Output Ports
Λ_0, Λ_1, Λ_2, Λ_3, Λ_4, Λ_5	Port 0
Λ_6, Λ_7, Λ_8, Λ_9, Λ_{10}, Λ_{11}	Port 1
Λ_{12}, Λ_{13}, Λ_{14}, Λ_{15}, Λ_{16}, Λ_{17}	Port 2
Λ_{18}, Λ_{19}, Λ_{20}, Λ_{21}, Λ_{22}, Λ_{23}, Λ_{24}	Port 3
Λ_{22}, Λ_{24}, Λ_{25}, Λ_{26}, Λ_{27}, Λ_{28}, Λ_{29}	Port 4
Λ_{30}, Λ_{31}, Λ_{32}, Λ_{33}, Λ_{34}, Λ_{35}	Port 5

Table 5.2: Connections of input and output ports in the read network for the rate-3/4 codes

Input Ports	Output Ports
Λ_0, Λ_1, Λ_2	Port 0
Λ_3, Λ_4, Λ_5	Port 1
Λ_6, Λ_7, Λ_8, Λ_9	Port 2
Λ_6, Λ_9, Λ_{10}, Λ_{11}, Λ_{12}	Port 3
Λ_{12}, Λ_{13}, Λ_{14}, Λ_{15}	Port 4
Λ_{13}, Λ_{15}, Λ_{16}, Λ_{17}, Λ_{20}	Port 5
Λ_{18}, Λ_{19}, Λ_{20}	Port 6
Λ_{21}, Λ_{22}, Λ_{23}, Λ_{24}	Port 7
Λ_{24}, Λ_{25}, Λ_{26}	Port 8
Λ_{27}, Λ_{28}, Λ_{29}, Λ_{30}	Port 9
Λ_{29}, Λ_{30}, Λ_{31}, Λ_{32}, Λ_{33}	Port 10
Λ_{32}, Λ_{33}, Λ_{34}, Λ_{35}	Port 11

much smaller $m_b \times n_b$ switching network, which is 18×36 for rate-1/2 CMMB codes. The size of the network can be further reduced by exploiting the structure of the QC codes. As seen from Fig. 5.1, the row weight of rate-1/2 code is limited to 6, which simplifies the number of cyclic shift patterns in a horizontal block. Therefore, the network can be further reduced to the size of 36×6. Moreover, the output ports of the read network do not need to connect all 36 input ports. From Fig. 5.1, we can draw that the first output port of the read network only needs to be connected with the APP memories Λ_0, Λ_1, Λ_2, Λ_3, Λ_4 and Λ_5, as the first 6 columns in the base matrix. The connections between the input and output ports of the read network for the rate-1/2 codes are listed in Table 5.4.2. Therefore, only four 6×1 multiplexers and two 7×1 multiplexers are required to implement the read network. Similarly, four 1×6 de-multiplexers and two 1×7 de-multiplexers are used to implement the write network. The architectures of the read network and write network for rate-1/2 codes are shown in Fig. 5.8. Applying the same simplification technique, the connections of the input and the output ports of the read network

for the rate-3/4 codes are listed in Table 5.4.2.

5.5 Implementation results

To evaluate the performance of the proposed architectures, we implemented the dual-rate 9216-bit LDPC decoder in 90nm 1.0V CMOS technology with 8-layer metals. Synthesis results show that the decoder can operate at a maximal frequency of 431 MHz. The parallelism factor is 8 and the number of pipeline stages is 6. Because there are 18 horizontal blocks in the **H** matrix and each block can be expanded to a 256×256 sub-matrix, the number of clock cycles per iteration is $18 \times 256/8 + 5 = 581$. The decoder is capable to process the two rates specified in CMMB standard with a maximum decoding throughput of 456 Mbps at 15 iterations. If calculated by the information bits, the effective throughput is 228 Mbps for the rate-1/2 code and 342 Mbps for the rate-3/4 code.

The decoder chip consumes a core area of $2.1mm \times 2.1mm$ and power consumption of 115mW. SRAMs are generated by Synopsys DesignWare tool and thus flattened during synthesis and chip layout. In practice, we can operate the decoder at a reduced clock frequency while significantly lowering the supply voltage, which takes advantage of high throughput of current design. Thus the resultant design should be more power efficient.

Table 5.5 shows the decoder implementation results compared with other LDPC decoders with long code length, such as LDPC codes in DVB-S2 standard. In the proposed decoder, over 75% of the core area is consumed by SRAMs, which are synthesized memory blocks. While in [72, 71], the memories are customized memory modules which are area-efficient and low-power. But still, the proposed decoder is able to achieve higher decoding throughput with comparable chip area.

Table 5.3: Overall comparison between proposed decoder and other existing irregular decoders

	F. Kienle [38]	J. Dielessen [18]	P. Urard [72]	P. Urard [71]	Proposed Decoder
Code Length	64800	64800	16200/64800	16200/64800	9216
Frequency	270MHz	-	300MHz	174MHz	431MHz
Iterations	30	30	programmable	programmable	15
Throughput	255Mbps	90Mbps	135Mbps	105Mbps	228Mbps or 342Mbps
Technology	130nm	90nm	90nm	65nm	90nm
Memory	-	-	2.832M bits	3.18M bits	170K bits
Area	$22.7mm^2$	$4.1mm^2$	$15.8mm^2$	$6.07mm^2$	$4.4mm^2$

5.6 Summary

In this paper, we present a low-complexity QC-LDPC decoder implementation to support dual-rate LDPC codes specified by the CMMB standard. Various optimizations are incorporated in the design to reduce complexity and latency of computational units, to minimize memory usage, and to simplify the switching network. Min-Sum based layered decoding is employed to achieve effective high decoding throughput with low computation complexity. A split-memory technique is proposed to efficiently handle exceptional weight-2 sub-matrices. Reconfiguration for supporting dual-rate LDPC codes is enabled with minimal hardware overhead.

Chapter 6

Design of Belief-Propagation Based Detectors for Sparse ISI Channels

6.1 Introduction

Sparse channels are most often found in underwater acoustic communications (UWA) and ultra-wideband communications (UWB) both of which have attracted lots of interests in recent years. For instance, the underwater acoustic communication systems have large delay spreading and multipath propagation. The channel is usually modeled with large delay spread and high sparsity. It is a challenge to design a low-complexity detector with acceptable error performance for sparse channel. Traditional approaches such as Viterbi algorithm is optimal but the complexity is too high for long channel length. In recent years, a group of researchers in signal processing and communications proposed to use the BP algorithm for symbol detection over a known ISI channel [41, 62]. The computational complexity of the BP-based detection is solely determined by the number of the nonzero interferers. On the contrary, the complexities of the optimal maximum a posterior (MAP) algorithms

or maximum-likelihood (ML) algorithms increase exponentially as of the channel length. Therefore, BP-based channel detector is considered in the sparse ISI channel with long delay spreads and only a few nonzero interferers, such as the underwater acoustic channels [62].

The BP algorithm was popularly used to represent the iterative decoding of LDPC codes [24] on a factor graph (also called Tanner graph). While applying the BP algorithm to detect symbols over a known ISI channel, the input and output symbols of the sparse channel are described as the variable nodes and check nodes on the factor graph, respectively.

Architecture of the BP-based detector can be referred to the architecture of an LDPC decoder [85, 83]. The main functional blocks include memories for storing the iterative messages, node processing units (including check node units (CNUs) and variable node units (VNUs)), connection networks and other control blocks. However, the LDPC codes are usually designed to have some special structure to facilitate parallel node processing and reduce the complexity of controlling and message passing, such as QC LDPC codes [83]. Although we can model the sparse channel using similar factor graph structure, the major challenge of a BP-based detector design lies in that the channel structure is lack of regularity due to the random locations of the interferers in a time-varying ISI channel.

In this chapter, we provide a feasible and efficient solution for such type of random ISI channels. A reconfigurable architecture of the BP-based detector is design, which can change the connections in the factor graph for time-varying ISI channels. The channel state information (CSI) including the connections between CNUs and VNUs are stored in registers which can be updated if the CSI changes. Layered decoding algorithm is also incorporated to accelerate the iterative latency. All of the tasks are managed by the control unit implemented as a finite state

machine (FSM). The proposed detector is implemented using ASIC design in TSMC 90 nm CMOS process and also prototyped on an FPGA.

The rest of this chapter is organized as follows. In section 6.2, the BP algorithm and layered decoding algorithm for the ISI channels is introduced. Section 6.3 presents the the simulation results. The detector architecture is presented in Section ?? and the implementation results is presented in 5. Finally, Section 6 draws the conclusion.

6.2 Channel Model and Decoding Algorithms

6.2.1 Channel Model and Factor Graph Representation

In a wireless communication system through a discrete-time ISI channel with noise, the information bits d is first encoded by the forward error correction (FEC) encoder to add redundant information to the stream of information bits in a way that allows errors which are introduced by the noise in the channel to be corrected. The coded bits c are then enter the modulator to be converted into signals appropriate for transmission over the channel. For simplicity, the binary phase-shift keying is used in this paper. The modulated symbols, i.e., the input of the channel x, are corrupted by interference and noise in the channel, which can be written as

$$y_n = \sum_{l=0}^{L-1} f_l x_{n-l} + \omega_n \tag{6.1}$$

where L is the channel length and $\boldsymbol{F} = \{f_0, f_1, \cdots\cdots f_{L-1}\}$ denotes the equivalent impulse response of the ISI channel. Here we assume $\boldsymbol{F}$ is known to the detector already. The number of nonzero elements in $\boldsymbol{F}$ is actually the number of the nonzero interferers, denoted by D.

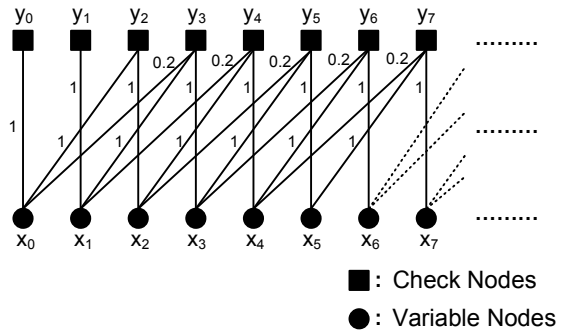


Figure 6.1: Factor graph of an ISI channel.

The factor graph representation of an ISI channel is shown in Fig. 6.1 with $L = 4$ and $\boldsymbol{F} = \{1, 0, 1, 0.2\}$. The input and the output of the channel are denoted as variable nodes and check nodes, respectively. The connections between the check nodes and variable nodes represent the dependencies of the output on the channel input symbols. The number on the edge denotes the amplitude of each tap in $\boldsymbol{F}$.

6.2.2 Belief-Propagation Algorithm

Before presenting the BP algorithm [62], we first give some definitions as follows: Let $r_{n \to n-j}[k]$ be the CTV message from check node n to variable node $n-j$ during the k-th iteration. Let $q_{n \to n+j}[k]$ be the VTC message from variable node n to check node $n+j$.

1. **Initialization:**

   Under the assumption of equal priori probability, compute the channel log-likelihood ratio (LLR) p_n (intrinsic information) of the variable node n, by:

$$p_n = log \frac{P(y_n \mid x_n = +1)}{P(y_n \mid x_n = -1)} \tag{6.2}$$

   For uncoded systems, $p_n = 0$. The CTV message $r_{n \to n-j}$ is initially set to zero and the APP messages Λ_n is initially set to be p_n.

2. **Calculating the VTC messages:**

   At the k-th iteration, for variable node n, calculate VTC message $q_{n\to n+j}[k]$ by

$$q_{n\to n+j}[k] = \Lambda_n[k-1] - r_{n+j\to n}[k-1] \tag{6.3}$$

3. **Calculating the CTV messages:**

   Calculate the CTV message $r_{n\to n-j}$ by (6.4). Here $b_n = (d_n+1)/2$.

$$\begin{aligned} r_{n\to n-j}[k] \;=\; & \max_{\forall x:\, x_{n-j}=+1} \left(\frac{-\left| y_n - \sum_{l=0}^{L} f_l x_{n-l} \right|^2}{2\sigma^2} + \sum_{i=0, i\neq j}^{L} b_{n-i} \cdot q_{n-i\to n}[k] \right) \\ & - \max_{\forall x:\, x_{n-j}=-1} \left(\frac{-\left| y_n - \sum_{l=0}^{L} f_l x_{n-l} \right|^2}{2\sigma^2} + \sum_{i=0, i\neq j}^{L} b_{n-i} \cdot q_{n-i\to n}[k] \right) \end{aligned} \tag{6.4}$$

4. **Renew the APP messages:**

   Then the APP (a-posterior probability) messages are renewed as follows

$$\Lambda_n[k] = q_{n\to n+j}[k] + r_{n+i\to n}[k] \tag{6.5}$$

5. **Deciding hard bits:**

   If the preset maximum number of iterations is reached, decide the n-th bit of the decoded codeword $x_n = +1$ if $\Lambda_n \geq 0$ and $x_n = -1$ otherwise.

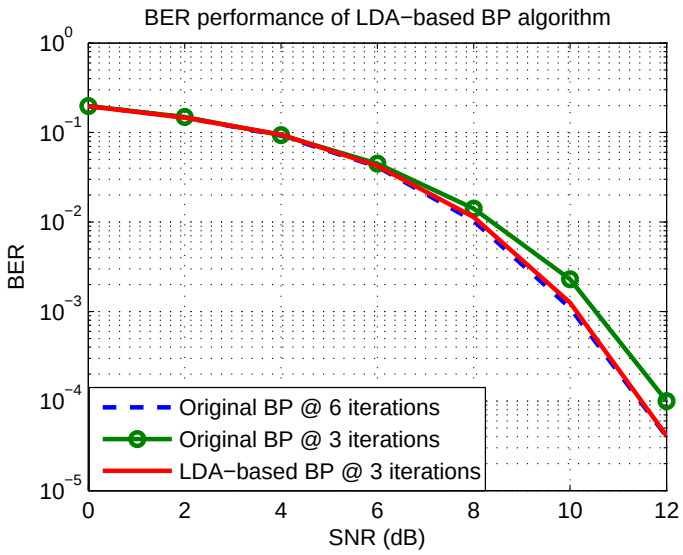


Figure 6.2: BER comparison between original BP algorithm and LDA-based BP algorithm.

6.2.3 Layered Decoding Algorithm

In order to alleviate message dependency and reduce decoding latency, we borrow the layered decoding algorithm from LDPC decoding to use it here in the BP-based detector design. Layered decoding is performed in a way that each check node is treated as a layer and within an iteration, renewed CTV messages from current horizontal layer will be transmitted vertically to other unprocessed layers that belong to the same variable nodes as the newly updated messages. In this way, layered decoding is able to reduce the required number of iterations by half for a given SNR, as shown in Fig. 6.2. In this figure, the sparse ISI channel is characterized by $L = 6$ and $\boldsymbol{F} = \{0.408, 0, 0, 0, 0.816, 0.408\}$.

6.3 Simulation Results

In this section, we evaluate the performance of the proposed algorithm by Matlab simulation in term of bit error rate (BER) versus signal-to-nose ratio (SNR) per bit

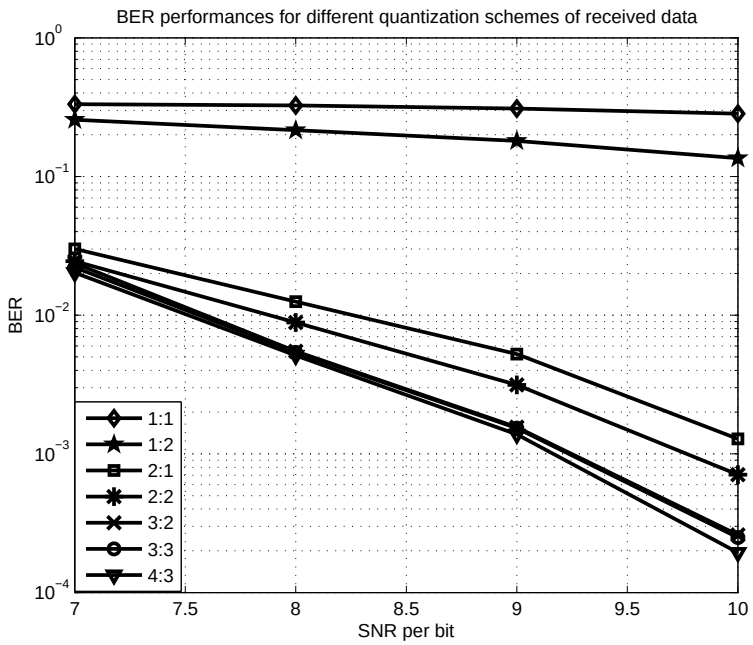


Figure 6.3: BER performance for various quantization schemes of received data.

E_b/N_0. The finite word length effect on the detector performance is also analyzed to minimize the required word length of every kind of message. In this paper, the channel model is characterized as a 64-tap channel ($L = 64$) with typically 4 nonzero interferers ($D = 4$). The data are modulated by BPSK and transferred over AWGN channel.

Let us first examine the quantization of the received data y_n in (6.4). A short word length would result in poor BER performance. Hereby through simulation we try to find an optimal quantization scheme with the shortest word length as well as no significant BER performance loss.

The quantization scheme can be express as $i : f$, in which the received data are quantized to i integer bits and f fractional bits. Various quantization schemes for the received data are investigated from $1 : 1$ to $4 : 3$. Error performances for some typical quantization schemes, such as $1 : 1$, $1 : 2$, $2 : 1$, $2 : 2$, $3 : 2$, $3 : 3$, and

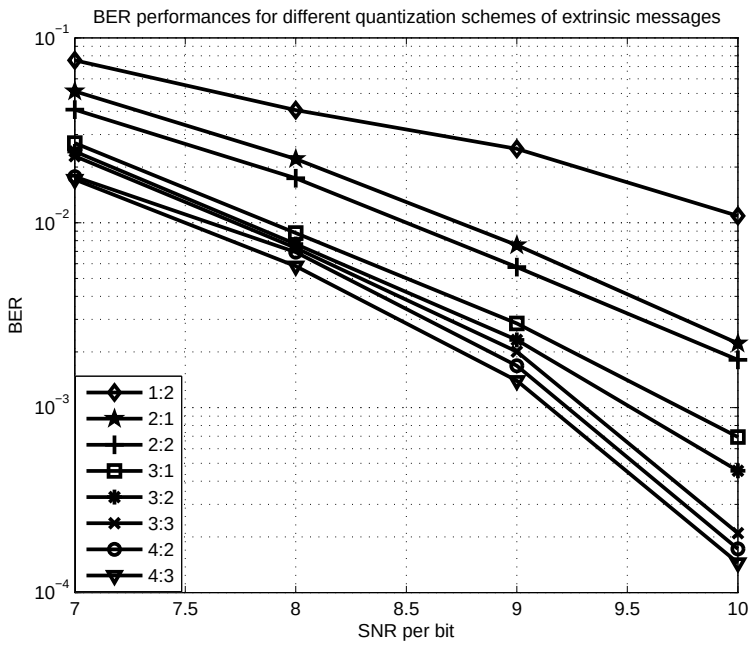


Figure 6.4: BER performance for various quantization schemes of extrinsic messages.

$4:3$, are depicted in Fig. 6.3. It is observed that the difference between $3:2$ and $3:3$ is negligible within a wide range of BER, while the difference $3:2$ and $2:2$ is significant. Thus it turns out that using the $3:2$ scheme for the received data seems to be the optimal tradeoff between hardware complexity and decoding performance.

Similarly, the quantization for the extrinsic messages are also analyzed. Error performances for some typical quantization schemes, such as $1:2$, $2:1$, $2:2$, $3:1$, $3:2$, $3:3$, $4:2$, and $4:3$ for the CTV and VTC messages, are depicted in Fig. 6.4. Also, as the APP message is the sum of several CTV messages, the quantization scheme for an APP message is $(i+2):f$ if the corresponding CTV quantization scheme is denoted as $i:f$. It is observed that the difference between $4:2$ and $4:3$ is negligible within a wide range of BER, while the difference $4:2$ and $3:2$ is significant. Since scheme $4:2$ and $3:3$ have the same length, $4:2$ is preferred due to better BER performance. Thus it turns out that using the $4:2$ scheme for the

extrinsic messages (6 : 2 for the APP messages) seems to be the optimal tradeoff between hardware complexity and decoding performance.

6.4 Detector Architecture Design

6.4.1 Overall Architecture

In this section, we propose an effective architecture for the BP-based detector with layered decoding. Overall architecture is shown in Fig. 6.5. In this section, the channel model is characterized as a 64-tap channel ($L = 64$) with typically 6 nonzero interferers ($D = 4$). The length of each frame of data is 2048 bits. The detector mainly consists of two memories which store the extrinsic messages, a cache which is used to store extrinsic messages for check node currently being processed, a CNU responsible for the check node update shown in (6.4) and a control unit which switches from one state to another. The LDA is implemented by processing each check node (layer) one by one and the processing of each check node can be divided into several states. The proposed architecture is also reconfigurable in order to switch flexible connections on the factor graph in the time-varying ISI channels. Below are the detailed descriptions of each state and how to control them.

1. The APP messages are initialized by channel LLRs and stored in *APP Memory* with the size of $n \times m_{app}$. n is the length of a frame of data steam and m_{app} denotes the length of the quantized APP messages. Here $n = 2048 = 2^{11}$ and $m_{app} = 6 + 2 = 8$ according to the word length analysis presented in Section 3.

2. An APP message is read from the *APP Memory* and written to the *Cache*. A *PC* (program counter) is applied to indicate the index of currently processed

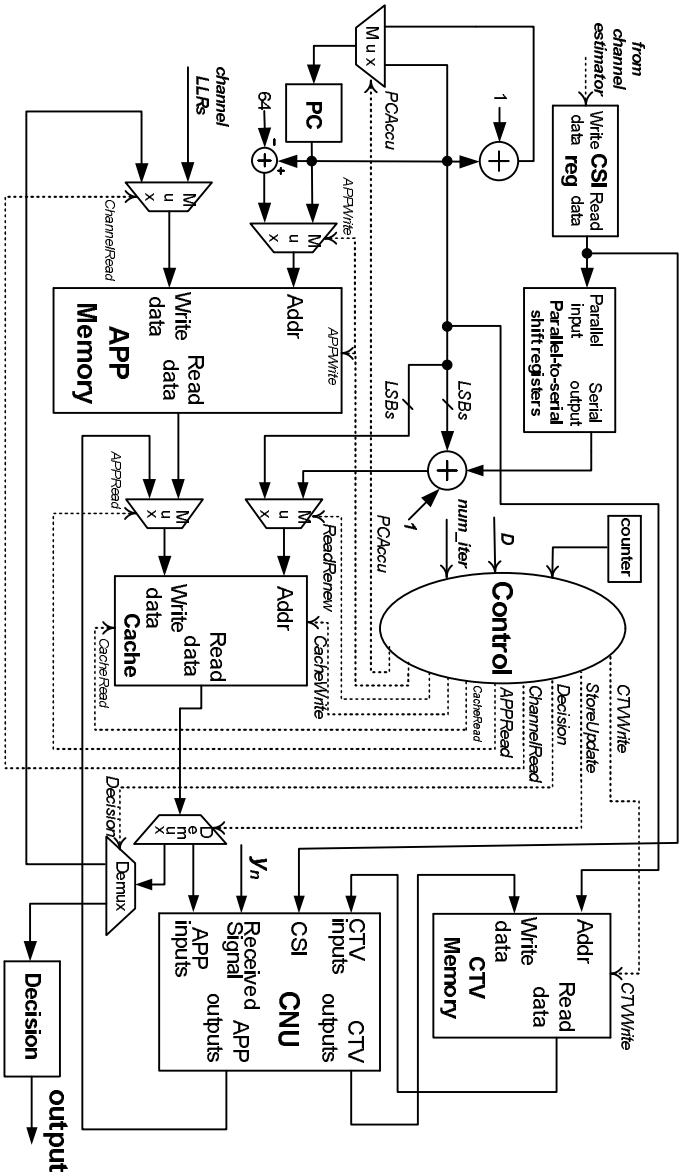


Figure 6.5: Overall detector architecture.

layer (ICL), which is actually the address of the *APP memory* and the *CTV memory*. The size of the *Cache* is $L \times m_{app}$ and $L = 64 = 2^6$ is the channel length ($L \ll n$). When the detector starts to process layer $j+1$ after layer j, only one new APP message (Λ_{j+1}) is needed and others are either replaced by new APP messages from *CNU* or remain unchanged.

3. APP messages corresponding to current layer are read from *Cache*. The read addresses are from the *CSI reg* which stores CSI including the relative interferer locations (RIL) and interferer coefficients. CSI is obtained from the channel estimator and is flexible depending on the time-varying channel. Meanwhile, CTV messages are also fetched and the CNU complete the task defined in (6.3), (6.4) and (6.5).

4. Renewed CTV messages are stored back into *CTV Memory* and renewed APP messages are stored back to *Cache*. Signal *APPRead* is used to switch between writing a new APP message from the *APP Memory* or writing renewed APP messages from the *CNU*.

5. When the detector starts to process layer $j+1$, variable nodes in the range $[j-L+2,\ j+1]$ are all potential interferers. Hereby, variable node $j-L+1$ will never be used again within this iteration and corresponding APP message Λ_{j-L+1} must be stored back to the *APP memory* and prepared for the use in the next iteration. Signal *StoreUpdate* decides where the output of the *Cache* should go: the *CNU* or the *APP memory*? Also, at the input the *APP memory*, signal *ChannelRead* controls what to be saved in the *APP memory*: channel LLRs or the APP messages.

6. Particularly, if the preset number of iterations is reached, the APP messages do not need to store back to the *APP memory* and are controlled by signal

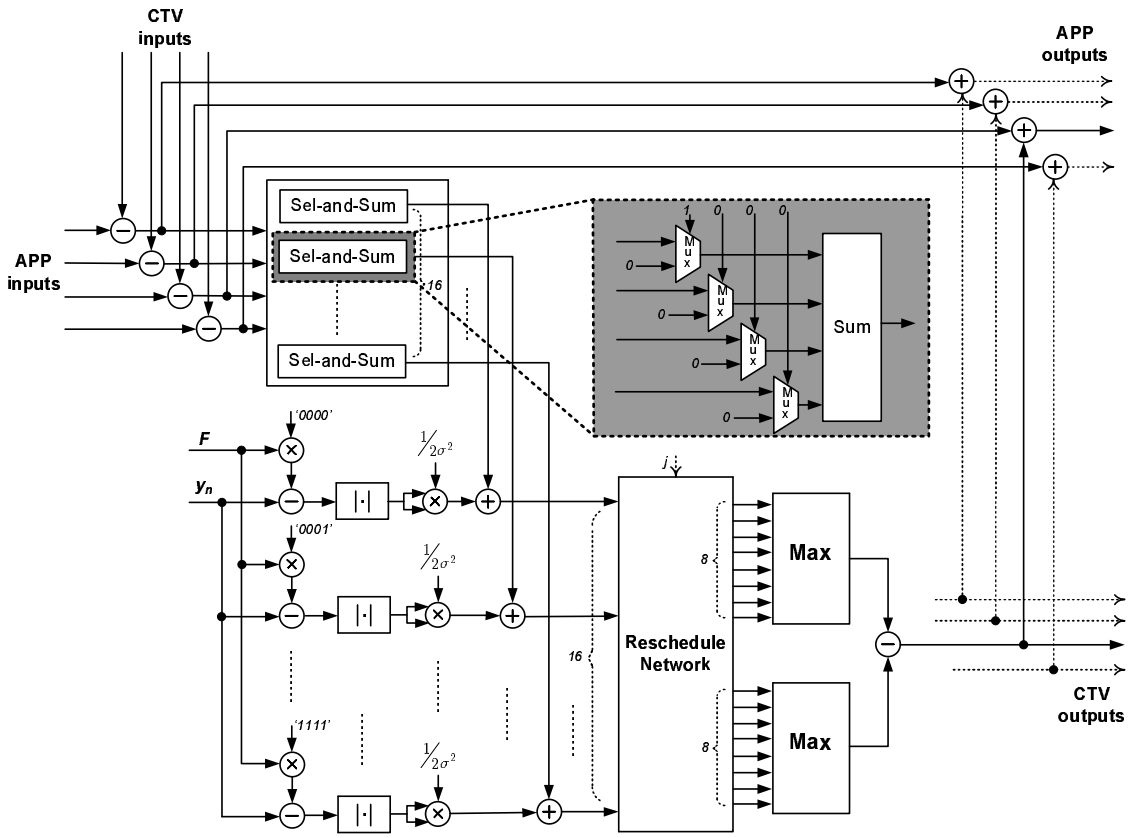


Figure 6.6: Architecture of the CNU.

Decision and delivered directly to the *Decision* unit to decide the hard bits.

6.4.2 Architecture of CNU

The CNU architecture is illustrated in Fig. 6.6. Its functionality covers equation (6.3), (6.4) and (6.5). As here the number of nonzero interferers D is 4, totally $2^D = 16$ values of the term $\left(\frac{-\left| y_n - \sum_{l=0}^{L} f_l x_{n-l} \right|^2}{2\sigma^2} + \sum_{i=0, i\neq j}^{L} b_{n-i} \cdot q_{n-i\to n}[k] \right)$ should be calculated, as indicated in (6.4). Among these 16 values, 8 will be used to calculate the minuend term (first maximum function) in (6.4) and another 8 values are for the subtrahend term (second maximum function). Parallel architecture is constructed to calculated the 16 values concurrently and a *Reschedule Network* is designed to

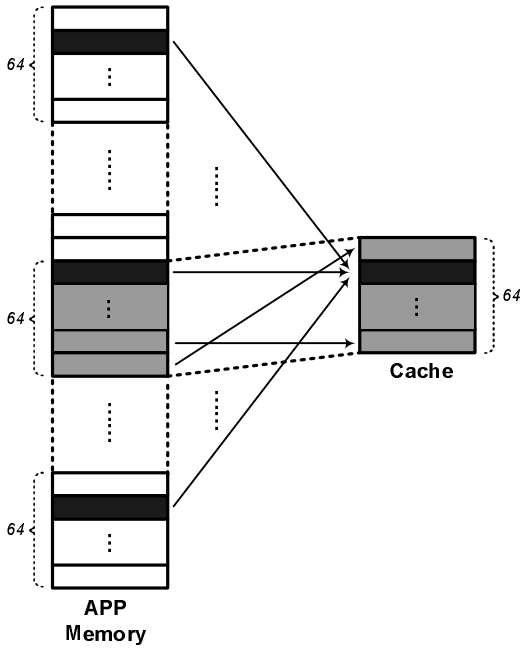


Figure 6.7: Direct-mapped Cache Architecture

partition the 16 values into two groups, one for the minuend and another for the subtrahend.

6.4.3 Cache-Like Architecture

The LLRs from the channel and the APP messages from the belief propagation will be stored in the *APP Memory* during the iterative process. Layered decoding algorithm is implemented by processing the check nodes one by one and each check node is considered as one layer. However, at each check node (layer), only D variable nodes out of previous L nodes are connected with current check node. In other words, D VTC messages from previous L variable nodes are selected and sent to *CNU* for iterative processing. Therefore, as shown in Fig. 6.5, a cache-like architecture is developed in which a *Cache* is used to read L APP messages from the main memory (the *APP Memory*) and then provide D useful APP messages for the *CNU*.

Fig. 6.7 shows the direct mapping from the APP Memory to the Cache in which each APP message is mapped to exact one location in the Cache. Since $n = 2048$ and $L = 64$, in this paper a $32 : 1$ direct-mapped cache is used. L consecutive APP messages are copied from the APP Memory to the Cache, but the start-point of these L consecutive messages may not be stored in the first location of the Cache. Actually the L messages are stored in a circulant shifted format with the shift value determined by the least significant bits (LSBs) of the ICL, as shown in Fig 6.5. Below are the detailed conclusions on how to set the addresses for the APP Memory and the Cache:

1. When storing the updated APP message back from the Cache to the APP Memory, the address for the Cache is the LSBs of the ICL and the address for the APP memory equals to $ICL - 64$.

2. When reading new APP message from the APP Memory to the Cache, the address for the Cache is the LSBs of the ICL and the address for the APP memory equals to ICL.

3. When reading $D = 4$ APP messages from the Cache, the address for the Cache is the sum of the RIL and the shift value of the 64 consecutive messages which equals to the LSBs of the ICL plus 1, as depicted in Fig. 6.5.

6.5 Implementation Results

To evaluate the performance of our proposed schemes, a BP-based detector with the frame size of 2048 bits is synthesized and implemented on both FPGA and ASIC platforms. Implemented on Xilinx XC2VP30 device, the maximum frequency is 56.2 MHz which corresponding to the throughput of 1.44 Mbps with 3 iterations.

The same architecture is implemented and synthesized using TSMC 90nm ASIC technology. The synthesized detector can achieve a maximum throughput of 3.48 Mbps for 3 decoding iterations. The core area of the detector is $1.8mm^2$ and the power consumption is 54 mW at the maximum frequency of 136 MHz.

6.6 Summary

In this chapter, a low-complexity detector design was presented for sparse ISI channels using the belief-propagation algorithm. The layered decoding algorithm is also employed in the design. Simulation results show the 3 : 2 quantization scheme for the received data and 4 : 2 scheme for the extrinsic messages cause negligible BER performance loss but would reduce the hardware complexity significantly. The proposed detector for sparse ISI channels is implemented on both FPGA and ASIC platforms, which demonstrate the real-time performance of 3.48 Mbps and 1.44 Mbps throughput, respectively.

Chapter 7

Conclusions

This dissertation investigated several VLSI design issues for iterative belief propagation (BP) algorithm, including high-performance LDPC decoders and ISI detectors with the goal to reduce chip area and/or achieve high-throughput and/or rate flexible implementations.

A parallel layered decoding LDPC decoder architecture was proposed to achieve ultra high throughput. Sequential operations on layers in traditional layered decoding architecture was replaced by concurrent message calculations and transmissions among all layers according to certain rules that won't change the nature of layered decoding algorithm. Besides concurrent processing, this architecture also enables pipelined critical path and avoids the complicated interconnection network which is usually considered as the barrier for high-throughput LDPC decoders. All above methods contribute to a high-throughput LDPC decoder and push the input throughput to more than 1 Gbps.

Then, we extended our proposed parallel layered decoding architecture to punctured LDPC codes and developed a decoder architecture with both high throughput and rate flexibility. We studied several puncturing schemes and picked up one that

has the best error performance and least BER degradation compared with dedicated codes. A corresponding implementation of the selected puncturing scheme was designed and added to the parallel layered decoding architecture. The proposed LDPC decoder can achieve an input throughput of 975 Mbps and supports any rate between 1/2 and 1.

Furthermore, we proposed a low-complexity, area-efficient LDPC decoder architecture that is compatible with China Multimedia Mobile Broadcasting (CMMB) standard. Using resource-sharing, the check node unit in this architecture can switch between the two required code rates in CMMB. Split-memory architecture enables effective implementation of layered decoding algorithm on weigh-2 superimposed matrices.

Finally, we developed a detector architecture for sparse ISI channels which is also based on sparse matrix and BP algorithm, like the LDPC decoder. Bipartite graph, min-sum algorithm and layered decoding algorithm, which are popular in LPDC decoding, are also applied here. Unlike the LDPC codes, for detecting in sparse ISI channels, variable nodes are possibly connected to part, not all, of the check nodes. Hence, a cache-based detector architecture was proposed to use a cache to store messages from/to check nodes that interfere with current variable node. Also, the detector architecture is reconfigurable to support any possible connections between variable nodes and check nodes. To our best knowledge, this is the first VLSI realization of BP-based detector based upon the available literature resources.

Bibliography

[1] *Mobile Multimedia Broadcasting (P. R. China) Part 1: Framing Structure, Channel Coding and Modulation for Broadcasting Channel.*

[2] *[Onleine]: http://eng.t-dmb.org/.*

[3] *[Online]: http://www.dibeg.org/techp/techp.htm.*

[4] *[Online]: http://www.ieee802.org/16/tge.*

[5] E. Aktas, "Belief Propagation with Gaussian Priors for Pilot-Assisted Communication over Fading ISI Channels," *IEEE Trans. Wireless Commun.*, vol. 8, no. 4, pp. 2056–2066, Apr. 2009.

[6] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes," in *Proc. IEEE Intl. Conf. Commun. (ICC 1993)*, vol. 2, May 1993, pp. 1064–1070.

[7] A. Blanksby and C. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 404–412, Mar 2002.

[8] R. Bose and R.-C. D.K., "On a Class of Error Correction Binary Group Codes," *Inform. Control*, vol. 3, pp. 68–79, Mar. 1960.

[9] T. Brack, M. Alles, F. Kienle, and N. Wehn, "A Synthesizable IP Core for WiMAX 802.16E LDPC Code Decoding," in *Proc. IEEE 17th Int. Symp. Personal, Indoor and Mobile Radio Communications*, Sep. 2006, pp. 1–5.

[10] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-Complexity Decoding of LDPC Codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[11] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near-Shannon-Limit Quasi-Cyclic Low-Density Parity-Check Codes," *IEEE Trans. Commun.*, vol. 52, no. 7, pp. 1038–1042, Jul. 2004.

[12] Y. Chen and D. Hocevar, "A FPGA and ASIC Implementation of Rate 1/2, 8088-b Irregular Low Density Parity Check Decoder," in *Proc. IEEE GLOBECOM 2003*, vol. 1, Dec. 2003, pp. 113–117.

[13] E. Choi, S. Suh, and J. Kim, "Rate-Compatible Puncturing for Low-Density Parity-Check Codes with Dual-Diagonal Parity Structure," in *Proc. IEEE Symp. Person. Indoor Mobile Radio Commun. (PIMRC)*, vol. 4, Sep. 2005, pp. 2642–2646.

[14] S. Y. Chung, G. Forney, T. Richardson, and R. Urbanke, "On the Design of Low-Density Parity-Check Codes within 0.0045 dB of the Shannon Limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

[15] G. Colavolpe and G. Germi, "On the Application of Factor Graphs and the Sum-Product Algorithm to ISI Channels," *IEEE Trans. Commun.*, vol. 53, no. 4, p. 746, Apr. 2005.

[16] O. Daesun and K. Parhi, "Min-Sum Decoder Architectures with Reduced Word Length for LDPC Codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 1, pp. 105 –115, Jan. 2010.

[17] Y. Dai, N. Chen, and Z. Yan, "Memory Efficient Decoder Architectures for Quasi-Cyclic LDPC Codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 9, pp. 2898–2911, Oct. 2008.

[18] J. Dielissen, A. Hekstra, and V. Berg, "Low Cost LDPC Decoder for DVB-S2," in *Proc. Design, Autom. Test Eur.*, 2006, pp. 130–135.

[19] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, "A Class of Low-Density Parity-Check Codes Constructed Based on Reed-Solomon Codes with Two Information Symbols," *IEEE Commun. Lett.*, vol. 7, no. 7, pp. 317–319, Jul. 2003.

[20] P. Elias, "Coding for Noisy Channels," *IRE Conv. Rec.*, vol. 4, pp. 37–47, 1955.

[21] European Telecommunications Standards Institude (ETSI), "Digital Video Broadcasting (DVB): Transmission System for Handheld Terminals," *EN 302 307 V1.1.1. http://www.dvb.org.*

[22] M. Fossorier, "Quasicyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.

[23] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity-check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.

[24] R. Gallager, "Low-Density Parity-Check Codes," *IRE Tans. Inf. Theory*, vol. 7, pp. 21–28, 1962.

[25] G. Gentile, M. Rovini, and L. Fanucci, "Low-Complexity Architectures of A Decoder for IEEE 802.16e LDPC Codes," in *Proc. Euromicro Conf. Digital System Design (DSD)*, Aug. 2007, pp. 369–375.

[26] E. B. Guilloud and J. Danger, "λ-Min Decoding Algorithm of Regular and Irregular LDPC codes," in *Proc. 3rd Int. Symp. Turbo Codes and Related Topics*, Sep. 2003, pp. 451–454.

[27] K. Gunnam, G. Choi, W. Wang, and M. Yeary, "Multi-Rate Layered Decoder Architecture for Block LDPC Codes of the IEEE 802.11n Wireless Standard," in *IEEE ISCAS 2007*, May 2007, pp. 1645–1648.

[28] K. Gunnam, G. Choi, M. Yeary, and M. Atiquzzaman, "VLSI architectures for layered decoding for irregular LDPC codes of WiMax," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2007, pp. 4542–4547.

[29] J. Ha, J. Kim, D. Klinc, and S. McLaughlin, "Rate-Compatible Punctured Low-Density Parity-Check Codes with Short Block Lengths," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 728–738, Feb. 2006.

[30] J. Ha, J. Kim, and S. McLaughlin, "Puncturing for Finite Length Low-Density Parity-Check Codes," in *Proc. Inter. Symp. Inform. Theory (ISIT)*, Jun. 2004, p. 151.

[31] ——, "Rate-Compatible Puncturing of Low-Density Parity-Check Codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2824–2836, Nov. 2004.

[32] J. Ha and S. McLaughlin, "Optimal Puncturing Distributions for Rate-Compatible Low-Density Parity-Check Codes," in *Proc. Inter. Symp. Inform. Theory (ISIT)*, Jun. 2003, p. 233.

[33] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop on Signal Process. Syst. (SiPS)*, Oct. 2004, pp. 107–112.

[34] A. Hocquenghem, "Codes corecteurs d'erreurs," *Chiffres*, vol. 2, pp. 147–156, 1959.

[35] S. J. Johnson and S. R.Weller, "Codes for Iterative Decoding from Partial Geometries," in *Proc. IEEE Intl. Symp. Inform. Theory*, Jul. 2002, p. 310.

[36] S.-H. Kang and I.-C. Park, "Loosely Coupled Memory-Based Decoding Architecture for Low Density Parity Check Codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 5, pp. 1045–1056, May 2006.

[37] M. Karkooti and J. Cavallaro, "Semi-Parallel Reconfigurable Architectures for Real-Time LDPC Decoding," in *Proc. ITCC¡¯2004*, vol. 1, Apr. 2004, pp. 579–585.

[38] F. Kienle, T. Brack, and N. Wehn, "A Synthesizable IP Core for DVB-S2 LDPC Code Decoding," in *Proc. Design, Autom. Test Eur.*, vol. 3, Mar. 2005, pp. 100–105.

[39] Y. Kou, S. Lin, and M. Fossorier, "Low Density Parity Check Codes Based on Finite Geometries: A Rediscovery and New Results," *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.

[40] ——, "Low Density Parity Check Codes Based on Finite Geometries: A Rediscovery," in *Proc. IEEE Intl. Symp. Inform. Theory*, Jun. 2000, p. 200.

[41] B. Kurkoski, P. Siegel, and J. Wolf, "Joint Message-Passing Decoding of LDPC Codes and Partial-Response Channels," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1410–1422, Jun. 2002.

[42] J. Li and K. Narayanan, "Rate-Compatible Low Density Parity Check Codes for Capacity-Approaching ARQ Schemes in Packet Data Communications," in *Proc. Int. Conf. on Comm., Internet, and Info. Tech.(CIIT)*, Nov. 2002.

[43] Z. Li and B. Kumar, "A Class of Good Quasi-Cyclic Low-Density Parity Check Codes Based on Progressive Edge Growth Graph," in *Proc. 38th Asilomar Conf. Signals, Syst. Comput.*, vol. 2, Nov. 2004, pp. 1990–1994.

[44] C. Liu, C. Lin, S. Yen, C. Chen, H. Chang, C. Lee, Y. Hsu, and S. Jou, "Design of A Multimode QC-LDPC Decoder Based on Shift-Routing Network," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 9, pp. 734 –738, Sep. 2009.

[45] C.-H. Liu, S.-W. Yen, C.-L. Chen, H.-C. Chang, C.-Y. Lee, Y.-S. Hsu, and S.-J. Jou, "An LDPC Decoder Chip Based on Self-Routing Network for IEEE 802.16e Applications," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 684–694, Mar. 2008.

[46] L. Liu and C.-J. Shi, "Sliced Message Passing: High Throughput Overlapped Decoding of High-Rate Low-Density Parity-Check Codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 11, pp. 3697–3710, Dec. 2008.

[47] H.-A. Loeliger, "An Introduction to Factor Graphs," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 28–41, Jan. 2004.

[48] R. Lucas, M. Fossorier, Y. Kou, and S. Lin, "Iterative Decoding of One-Step Majority Logic Deductible Codes Based on Belief Propagation," *IEEE Trans. Commun.*, vol. 48, no. 6, pp. 931–937, Jun. 2000.

[49] D. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[50] D. MacKay and R. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes," *Electron. Lett.*, vol. 32, no. 18, p. 1645, Aug. 1996.

[51] M. Mansour and N. Shanbhag, "High-throughput LDPC decoders," *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*, vol. 11, no. 6, pp. 976–996, Dec. 2003.

[52] G. Masera, F. Quaglio, and F. Vacca, "Implementation of A Flexible LDPC Decoder," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 6, pp. 542–546, Jun. 2007.

[53] T. Mittleholzer, "Efficient Encoding and Minimum Distance Bounds of Reed-Solomon-Type Array Codes," in *Proc. IEEE Intl. Symp. Inform. Theory*, Jul. 2002, p. 282.

[54] T. Mohsenin and B. Baas, "High-throughput ldpc decoders using a multiple split-row method," in *Proc. ICASSP 2007*, vol. 2, Apr. 2007, pp. II–13–II–16.

[55] S. Muller, M. Schreger, M. Kabutz, M. Alles, F. Kienle, and N. Wehn, "A novel LDPC decoder for DVB-S2 IP," in *Proc. Design, Automation and Test in Europe, (DATE'09)*, April 2009, pp. 1308–1313.

[56] S. Myung, K. Yang, and J. Kim, "Quasi-Cyclic LDPC Codes for Fast Encoding," *IEEE Trans. Inf. Theory*, vol. 51, no. 8, pp. 2894–2901, Aug. 2005.

[57] T. Okamura, "Designing LDPC Codes Using Cyclic Shifts," in *Proc. IEEE Intl. Symp. Inform. Theory*, Jun. 2003, p. 151.

[58] H. Park, K. Kim, D. Kim, and K. Whang, "Structured Puncturing for Rate-Compatible B-LDPC Codes with Dual-Diagonal Parity Structure," *IEEE Trans. Wireless Commun.*, vol. 7, no. 10, pp. 3692–3696, Oct. 2008.

[59] R. S. Reed and S. G., "Polynomial Codes over Certain Fields," *J. Soc. Ind. Appl. Math.*, vol. 8, pp. 300–304, Jun. 1960.

[60] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[61] T. Richardson and R. Urbanke, "The Capacity of Low-Density Parity-Check Codes under Message-Passing Decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[62] S. Roy, T. Duman, and V. McDonald, "Error rate improvement in underwater mimo communications using sparse partial response equalization," *IEEE J. Ocean. Eng.*, vol. 34, no. 2, pp. 181–201, Apr. 2009.

[63] C. E. Shannon, "A Mathematical Theory of Communications," *Bell Syst. Tech. J.*, pp. 379–423, Jul. 1948.

[64] R. Shao, S. Lin, and M. Fossorier, "Two Simple Stopping Criteria for Turbo Decoding," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1117–1120, Aug. 1999.

[65] E. Sharon, S. Litsyn, and J. Goldberger, "Efficient Serial Message-Passing Schedules for LDPC Decoding," *IEEE Trans. Inf. Theory*, vol. 53, no. 11, pp. 4076–4091, Nov. 2007.

[66] X.-Y. Shih, C.-Z. Zhan, C.-H. Lin, and A.-Y. Wu, "An 8.29 mm^2 52 mW Multi-Mode LDPC Decoder Design for Mobile WiMAX System in 0.13 μm CMOS Process," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 672–683, Mar. 2008.

[67] C. Studer, N. Preyss, C. Roth, and A. Burg, "Configurable High-Throughput Decoder Architecture for Quasi-Cyclic LDPC codes," in *Proc. 42th Asilomar Conf. Signals, Syst., Comput.*, Oct. 2008, pp. 1137–1142.

[68] Y. Sun and J. Cavallaro, "A Low-Power 1-Gbps Reconfigurable LDPC Decoder Design for Multiple 4G Wireless Standards," in *Proc. 2008 IEEE Int. SOC Conf.*, Sep. 2008, pp. 367–370.

[69] R. Tanner, "A Recursive Approach to Low Complexity Codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.

[70] Y.-L. Ueng, C.-J. Yang, Z.-C. Wu, C.-E. Wu, and Y.-L. Wang, "VLSI Decoding Architecture with Improved Convergence Speed and Reduced Decoding Latency for Irregular LDPC Codes in WiMAX," in *Proc. IEEE ISCAS 2008*, May 2008, pp. 520–523.

[71] P. Urard, L. Paumier, V. Heinrich, N. Raina, and N. Chawla, "A 360mW 105Mb/s DVB-S2 Compliant Codec based on 64800b LDPC and BCH Codes Enabling Satellite-Transmission Portable Devices," *IEEE ISSCC Dig. Tech. Papers*, pp. 310–311, Feb. 2008.

[72] P. Urard, E. Yeo, L. Paumier, P. Georgelin, T. Michel, V. Lebars, E. Lantreibecq, and B. Gupta, "A 135Mb/s DVB-S2 Compliant Codec Based on 64800b LDPC and BCH Codes," *IEEE ISSCC Dig. Tech. Papers*, pp. 446–609, Feb. 2005.

[73] B. Vasic, "Combinatorial Constructions of Low-Density Parity-Check Codes for Iterative Decoding," in *Proc. IEEE Intl. Symp. Inform. Theory*, Jul. 2002, p. 312.

[74] A. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.

[75] P. O. Vontobel and R. M. Tanner, "Construction of Codes Based on Finite Generalized Quadrangles for Iterative Decoding," in *Proc. IEEE Intl. Symp. Inform. Theory*, Jun. 2001, p. 223.

[76] P. Wang and Y. Chen, "Low-Complexity Real-Time LDPC Encoder Design for CMMB," in *Proc. IIHMSP '08*, Aug. 2008, pp. 1209–1212.

[77] Z. Wang and Z. Cui, "A Memory Efficient Partially Parallel Decoder Architecture for QC-LDPC Codes," in *Proc. 39th Asilomar Conf. Signals, Syst., Comput.*, Nov. 2005, pp. 729–733.

[78] ——, "Low-Complexity High-Speed Decoder Design for Quasi-Cyclic LDPC Codes," *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*, vol. 15, no. 1, pp. 104–114, Jan. 2007.

[79] Z. Wang and Q. wei Jia, "Low Complexity, High Speed Decoder Architecture for Quasi-Cyclic LDPC Codes," in *Proc. IEEE ISCAS 2005*, May 2005, pp. 5786–5789.

[80] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam, "VLSI architectures for iterative decoders in magnetic recording channels," *IEEE Trans. Magn.*, vol. 37, no. 2, pp. 748–755, Mar. 2001.

[81] C. Zhang, Z. Wang, J. Sha, L. Li, and J. Lin, "Flexible LDPC Decoder Design for Multigigabit-per-Second Applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 1, pp. 116–124, Jan. 2010.

[82] J. Zhang and M. Fossorier, "Shuffled iterative decoding," *IEEE Trans. Commun.*, vol. 53, no. 2, pp. 209–213, Feb. 2005.

[83] K. Zhang and X. Huang, "High-Throughput Layered Decoder Implementation for Quasi-Cyclic LDPC Codes," *IEEE J. Select. Areas Commun.*, vol. 27, no. 6, pp. 985–994, Aug. 2009.

[84] L. Zhang, L. Gui, Y. Xu, and W. Zhang, "Configurable Multi-Rate Decoder Architecture for QC-LDPC Codes Based Broadband Broadcasting System," *IEEE Trans. Broadcast.*, vol. 54, no. 2, pp. 226–235, Jun. 2008.

[85] T. Zhang and K. Parhi, "VLSI implementation-oriented (3,k)-regular low-density parity-check codes," in *Proc. IEEE Workshop on Signal Process. Syst. (SiPS)*, 2001, pp. 25–36.

[86] H. Zhong and T. Zhang, "Design of VLSI Implementation-Oriented LDPC Codes," in *Proc. IEEE VTC 2003*, vol. 1, Oct. 2003, pp. 670–673.

[87] ——, "Block-LDPC: A Practical LDPC Coding System Design Approach," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 4, pp. 766–775, Apr. 2005.