Waseda University Doctoral Dissertation

Research on High Performance LDPC Decoder

JI, Wen

Graduate School of Information, Production and Systems

Waseda University

February 2011

ABSTRACT

Low-Density Parity-Check (LDPC) code is an error correcting code first discovered in the early 1960's, and rediscovered in 1999. The decoding algorithm of LDPC is inherently parallel and can achieve performance close to the Shannon Limit, thus making LDPC code widely adopted in communication standards, such as WLAN (IEEE 802.11n), WiMAX (IEEE 802.16e) and DVB-S2.

However, it is still a great challenge for the researchers to design LDPC decoders with high performance. As LDPC decoders are mainly used to detect and correct errors in wireless communication devices, there are many aspects which should be concerned in the design, such as Bit Error Rate (BER) performance, hardware cost, throughput etc. As an error correcting technique, the BER performance is always an important metric for the design. The higher the BER performance is, the higher the reliability of the transmission is. In terms of LSI design, the hardware cost of LDPC decoder is considered to be a critical metric for the manufacture process. Also, in communication systems, throughput is an essential metric for LDPC decoders to ensure the real-time transmission.

In this dissertation, we mainly focus on the LDPC decoder design for IEEE 802.11n application and Integrated Services Digital Broadcasting via Satellite - Second Generation (ISDB-S2) application.

In the previous design of IEEE 802.11n LDPC decoder, fully-parallel decoder structures have been implemented aiming at high throughput but with relatively large area. A previous work implementing partially-parallel structure can achieve relatively low hardware cost, but has a low throughput, which is only 54Mbps. However, high throughput as well as low hardware cost is demanded for IEEE 802.11n application.

On the other hand, for ISDB-S2 LDPC decoder, previous decoding algorithms for LDPC decoders can either achieve very high BER performance by implementing very complicated hardware (Belief-Propagation –based algorithms), or achieve small area cost in the sacrifice of degrading BER performance (Min-sum –based algorithms). However, as long as the stable transmission is necessary for satellite broadcasting, very high BER performance is a key request for the design. But as for the manufacture demand, Belief-Propagation –based algorithms with large area cost is not a good option for LDPC decoder design for ISDB-S2.

Based on the above discussion, this dissertation is mainly composed of two issues: the LDPC decoder design for IEEE 802.11n targeting high throughput and low hardware cost; and the LDPC decoder algorithm design for ISDB-S2 targeting high BER performance and low hardware cost.

This dissertation consists of six chapters which are as follows:

Chapter 1 [Introduction] gives a brief introduction of this dissertation. It introduces the basic knowledge of LDPC codes and LDPC decoders. The LDPC decoding algorithms are discussed, followed by the motivation and contribution of this work.

Chapter 2 [A partially-parallel LDPC decoder targeting high throughput] proposed a partially-parallel irregular LDPC decoder for IEEE 802.11n standard targeting high throughput applications.

The proposed decoder has several merits:

(i) The decoder is designed based on a novel delta-value based message passing (DVMP) schedule which facilitates the decoding throughput by redundant computation removal.

(ii) Techniques such as binary sorting, parallel column operation, high performance pipelining are used to further speed up the message-passing procedure.

The implementation is done by using TSMC 0.18μm technology and the synthesis result shows that for (648,324) irregular LDPC code, this decoder can achieve 8 times increase in throughput compared to the previous work targeting the same code, reaching 418 Mbps at the frequency of 200MHz.

Chapter 3 [A partially-parallel LDPC decoder targeting high throughput and low hardware cost] proposed a sum-delta message passing (SDMP) schedule based on the DVMP schedule which can keep the advantages of the DVMP schedule while decrease the area of the decoder.

Registers and memory are optimized to store only the frequently used messages to decrease the hardware cost. An efficient pipeline structure is utilized to boost the total throughput. The synthesis is done by TSMC 0.18μm technology which demonstrates that for (648,324) irregular LDPC code, this decoder achieves 7.5X improvement in throughput compared to the previous work targeting the same code, which reaches 404 Mbps at the frequency of 200MHz. The decoder can also achieve a 11% area reduction compared to the previous work. The backend design of this decoder is also done by Synopsys Astro with ARM's Artisan SAGE-X 0.18μm 1P6M stand-cell library for TSMC, the layout area of this decoder is 13.69mm^2.

Chapter 4 [Self-adjustable offset min-sum algorithm targeting high BER performance and low hardware cost] proposed a novel self-adjustable offset min-sum LDPC decoding algorithm for ISDB-S2 application.

The existing LDPC decoding algorithms can either be categorized into Belief-Propagation (BP) –based algorithms which can achieve good BER performance but with large hardware cost, or Min-sum (MS) –based algorithms which consumes small hardware cost but degrades the BER performance.

In this chapter, a uniform approximation of the check node operation through mathematical induction on Jacobian logarithm is presented for the first time, and theoretically shows that the offset value is mainly dependent on the difference between the two most unreliable inputs from the bit nodes. The algorithm proposed

can adjust the offset value according to the inputs during the iterative decoding procedure. Simulation results for all 11 code rates of ISDB-S2 demonstrate that the proposed method can achieve an average of 0.12dB gain under the same BER performance, compared to the MS based algorithms, and consumes only 1.21% computation complexity compared to BP-based algorithms in the best case.

Chapter 5 [Data conflict resolution for layered schedule targeting high BER performance] proposed a novel selective recalculation method to solve the data conflict problem for applying layered schedule to ISDB-S2 LDPC codes.

The data conflict happens when layered algorithm is directly applied to ISDB-S2 codes. This problem arises as the layered algorithm adopts a parallel computation among a layer of several rows, which ignores the data dependencies of A Posterior Probability (APP). The selective recalculation method proposed in this chapter can determine the inaccurately calculated values based on a recalculation decision rule, and correct them accordingly. By applying this selective recalculation method, the layered algorithm can achieve conflict free BER performance.

Chapter 6 [Conclusion] summarizes the results of this work and discuss the future work.

ACKNOLEDGEMENTS

First of all, I would like to express my sincere gratitude to my advisor, Professor Satoshi Goto at Waseda University, who has constantly guided and supported me during my master and doctor period. I got advices from him in each research meeting and for every work I have done. He inspired my interest in research and also helped me a lot in my life.

I would also like to express my appreciation to Professor Takeshi Yoshimura, Professor Shinji Kimura, at Waseda University for the help and guidance to my research. I learned the basic knowledge of the VLSI design in their courses which are very important in the research. I express my cordial appreciation to thank Dr. Atsuki Inoue in Fujitsu Labs, whose suggestions, comments and guidance were invaluable to the completion of this work.

I am grateful to Professor Takeshi Ikenaga at Waseda University for the valuable knowledge I learned in his courses. It is also a pleasure to thank many of my colleagues who have constantly supported me and made this thesis possible, especially to Dr. Shinpei Hijiya, Mr. Hiroshi Nakayama and Mr. Makoto Hamaminato in Fujitsu Labs. Also, I would like to thank all the students in Goto lab. They gave me a lot of help and advices in my research and made my life in Waseda a wonderful memory.

Finally, I thank my parents, Qihua Ji and Yuqin Li, for their kind support over the years.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

In this chapter, a brief introduction of this thesis is provided. First of all, the basic knowledge of LDPC codes and LDPC decoders are described. Then the state of art LDPC decoder design and design challenges are introduced together with the motivation of this work, and the contribution is summarized, after which the thesis organization is provided.

1.1 LDPC codes and LDPC decoders

Low-Density Parity-Check (LDPC) code is an error correcting code originally proposed by Gallager in 1963[1], but soon forgotten by the scientific world because of the incapability of the microelectronics technology of that time. However, the recent requirement for modern communication systems to operate very close to the Shannon limit of channel capacity, the theoretical maximum [2], led to the rediscovery of the LDPC codes by Mackay and Neal [3] in 1996. After that irregular LDPC codes are constructed enabling data transmission rates close to the Shannon Limit [4], [5]. On the other hand, the LDPC decoding algorithm is inherently parallel and is easier to be implemented than its comparator turbo codes, thus making it more attractive to researchers [5]. As a result, they have been adopted for the 10GBase-T [6], the DVB-S2 [7], WLAN (IEEE802.11n) [17] and WiMAX (IEEE 802.16e) [8].

There are two kinds of LDPC codes: regular codes and irregular codes, as shown in Figure 1.1. The number of ”1” in one row is called the row weight (w_r), the number of ”1” in one column is called the column weight (w_c). If the row weight is the same and the column weight is the same, it is called regular LDPC code, as shown in

Figure 1.1(a). Regular codes are easy to be implemented in hardware while suffering a main problem that the Bit Error Rate (BER) performance is relatively worse than that of irregular codes [5]. As for irregular LDPC code, as shown in Figure. 1.1 (b), the row weight is different and the column weight is different. Although it has good error correction ability, irregularity results in hardware complexity and inefficiency in terms of reusability of functional units.

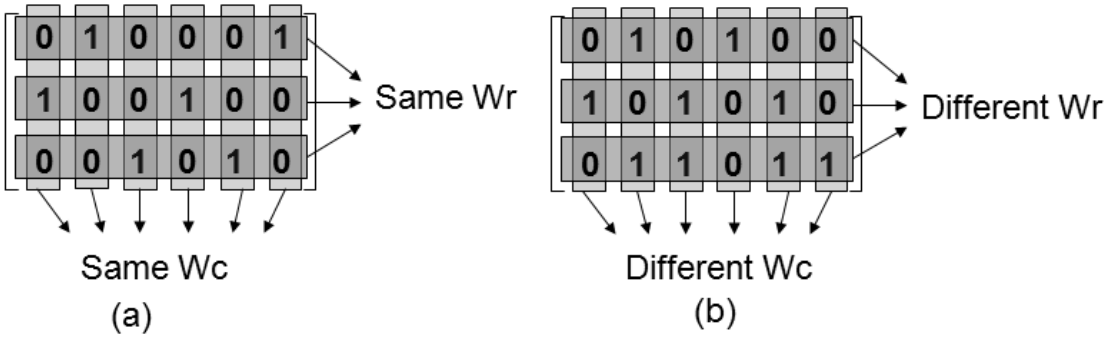
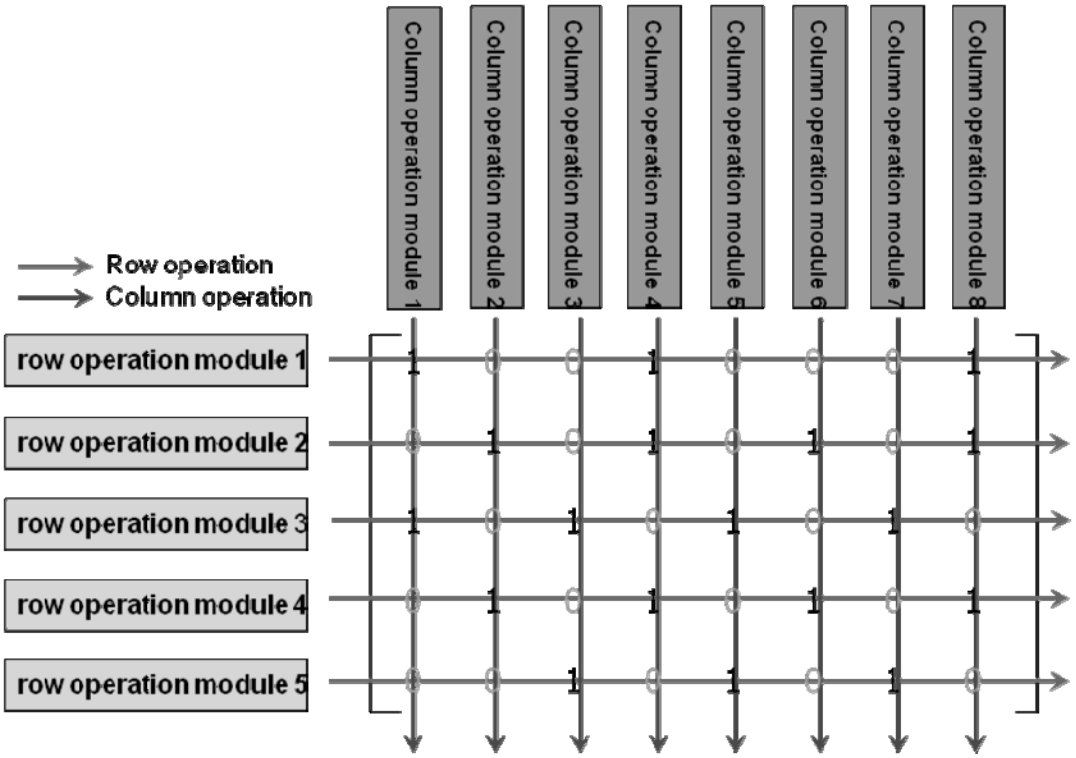
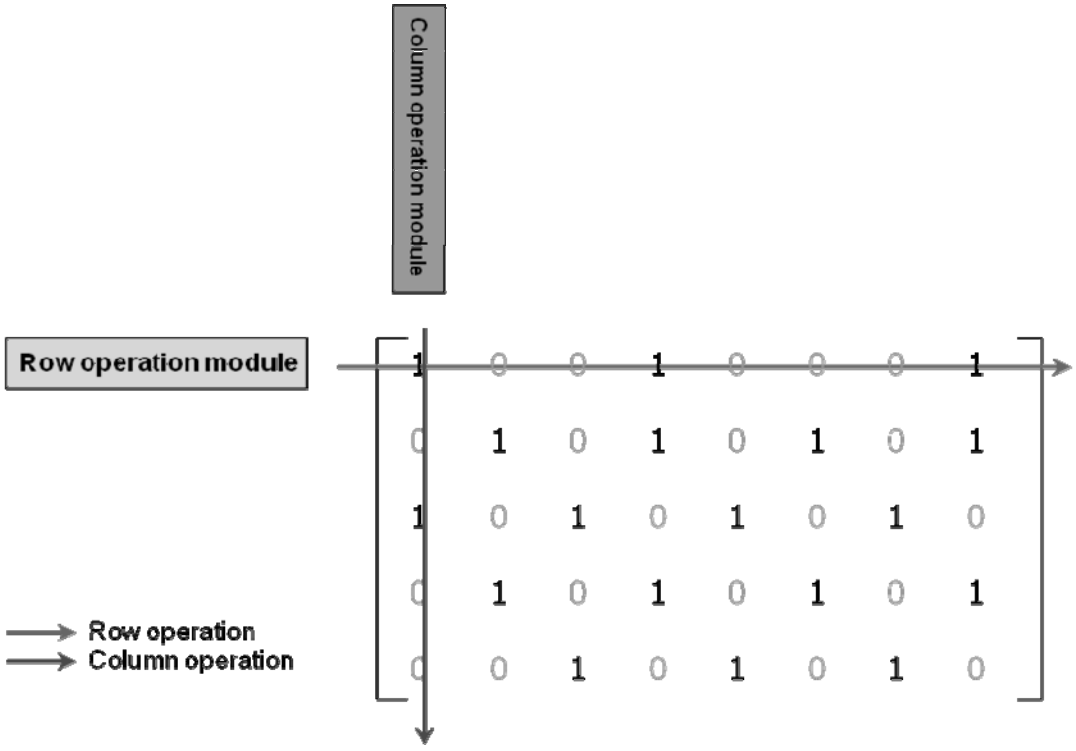


Figure 1.1 LDPC codes: (a) regular LDPC code, and (b) irregular LDPC code

As for the hardware implementation, LDPC decoders can be categorized as fully-parallel decoder and partially-parallel decoder, as shown in Figure 1.2. Fully-parallel LDPC decoder, as shown in Figure 1.2(a), does the row operations and column operations simultaneously for all the rows and columns, which can achieve higher throughput with routing complexity and area overhead [9]. In addition, fully-parallel LDPC decoder requires registers instead of SRAM banks for the row operation modules and the column operation modules. Therefore, the wiring area becomes a significant problem and the hardware architecture is fixed according to the parity check matrix. However, the partially-parallel decoder, as shown in Figure 1.2(b), does the row operations and column operations in groups by reusing the functional units. Figure 1.2(b) shows an example of partially parallel decoder which has one row operation module and one column operation module, respectively. In this way, area and power consumption can be reduced by reusing the functional units but with relatively low throughput [10], [11].



(a)



(b)

Figure 1.2 LDPC decoders: (a) fully-parallel LDPC decoder, and (b) partially-parallel LDPC decoder

1.2 LDPC Code Representation

There are two ways to represent LDPC codes: parity check matrix and tanner graph. Parity Check matrix can provide a direct view of the LDPC code in a mathematical way while through Tanner graph we can clearly see the connection between check nodes and bit nodes.

1.2.1 Parity Check Matrix

The LDPC codes can be defined by a parity check matrix H_{MN} as shown in Equation (1.1), where M and N represent the number of rows and the number of columns in the parity check matrix. Structured LDPC codes can be divided into $B \times D$ sub-blocks, where B and D are the number of row-blocks and column-blocks respectively. Each sub-block matrix is a $b \times b$ square matrix, obtained through right shifting the identity matrix $I_{b \times b}$ by a specific number. The parity check matrix shown in Figure 1.3 is the targeted parity check matrix in Chapter 2 and Chapter 3. It is a 324 × 648 structured matrix (each sub-block is 27bits×27bits) defined in IEEE 802.11n standard [17] with code rate of 1/2 and code length of 648 bits. It has 12 row-blocks and 24 column-blocks. In this figure, each small square represents a 27bits×27bits sub-block and sub-blocks in one row constitute one row-block, sub-blocks in one column constitute one column-block. Blank squares represent for zero sub-blocks while squares with lines are the sub-blocks who are obtained through right shifting the identity matrix.

$$H_{48} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad (1.1)$$

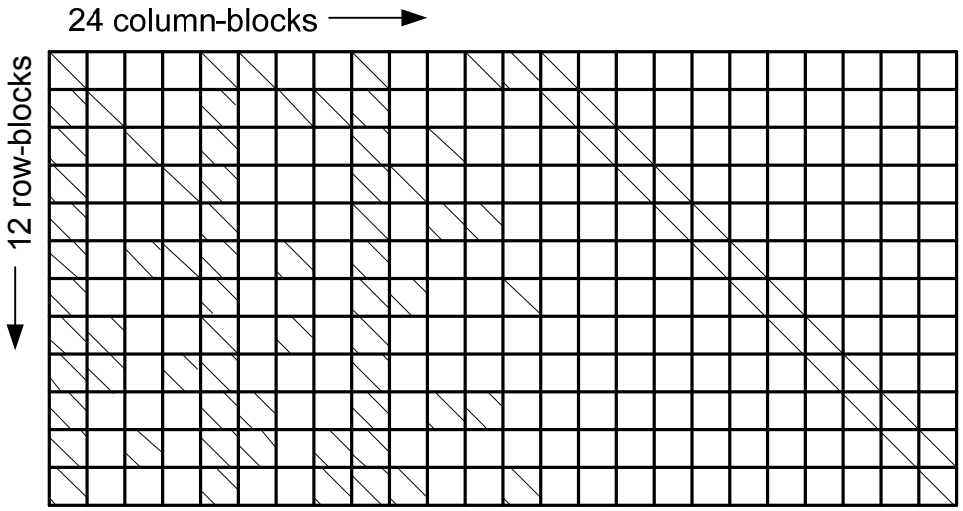


Figure 1.3 IEEE802.11n parity check matrix
(24×12 sub-blocks, each is 27bits×27bits)

1.2.2 Tanner graph

LDPC codes can be represented effectively by a bi-partite graph called a Tanner graph. There are two classes of nodes in a Tanner graph, “Bit Nodes” and “Check Nodes”. The Tanner graph of a code is drawn according to the following rule: Check node c_m (m=1,…, M) is connected to Bit node b_n (n=1,…, N) whenever element h_{mn} in H is “1”. Figure 1.4 shows a Tanner graph and the corresponding parity check matrix is also shown. When there is a “1” in the parity check matrix, there is a connection between the corresponding bit node and check node.

$$H = \begin{array}{c} \begin{matrix} b1 & b2 & b3 & b4 & b5 & b6 & b7 & b8 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} c1 \\ c2 \\ c3 \\ c4 \end{matrix}$$

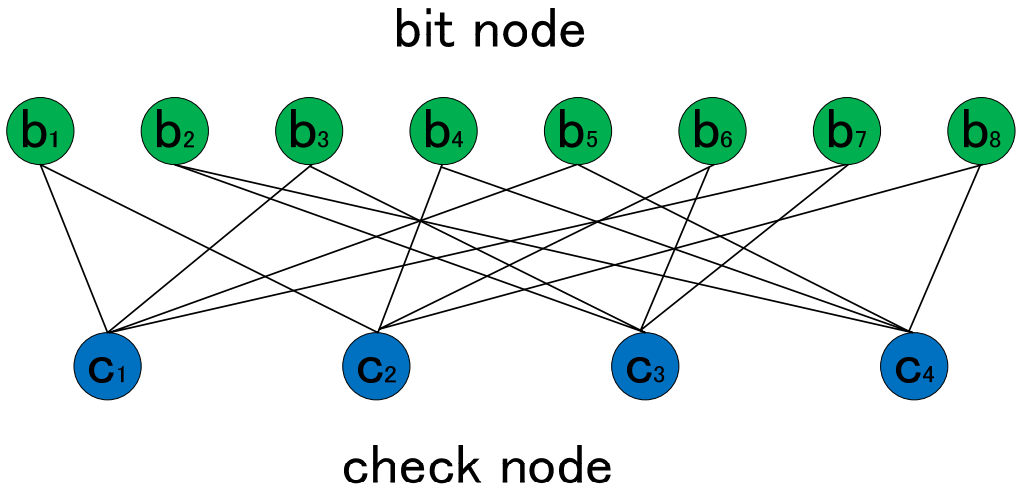


Figure 1.4 Tanner graph of a parity check matrix

1.3 Message passing (MP) decoding algorithm

The model of the communication system using LDPC code is demonstrated in Figure 1.5. Parity data is added to the transmitted data (Tx) to ensure H×Tx =0 in the sender part. The received data (Rx) should be correct if H×Rx=0. However because of the channel noise, H×Rx≠0 often happens. In the receiver, LDPC decoder is used to correct the received data until H×Rx=0.

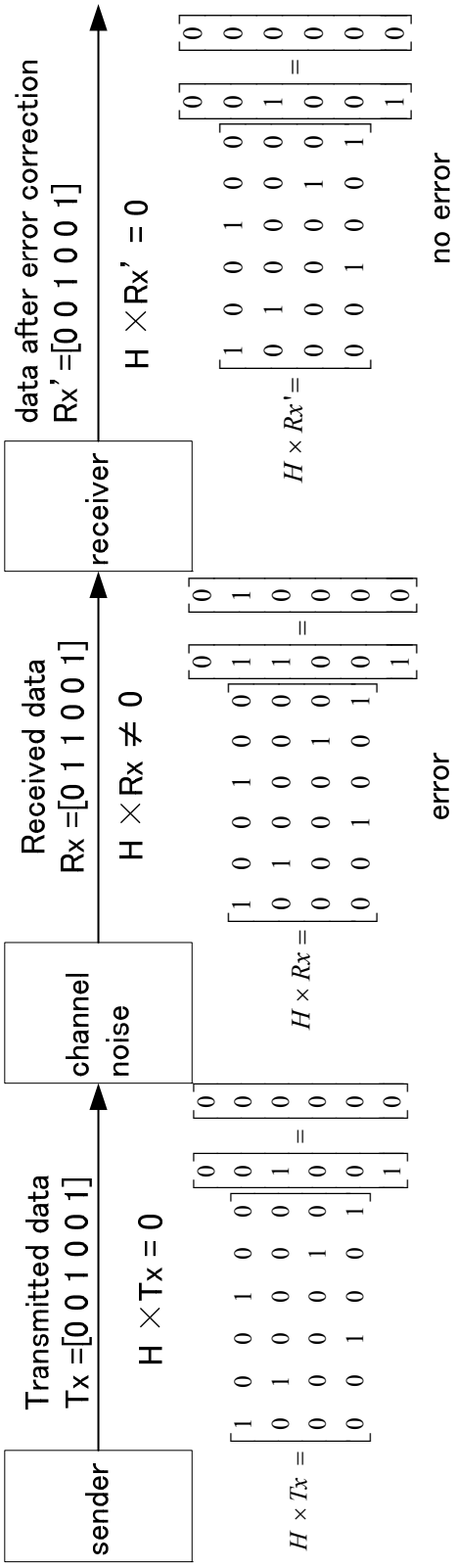


Figure 1.5 Transmission model using LDPC code

The encoding for LDPC code is relatively easy, while the decoding process is complicated and not easy to be implemented in hardware. In this section, basic decoding algorithms for LDPC code are introduced. The decoding algorithm for LDPC code is called message passing algorithm [1] which passes the messages between check nodes and bit nodes by performing row and column operations iteratively. It can also be called Two Phase Message Passing (TPMP) algorithm in contrast to layered algorithm which will be introduced in Section 1.4. According to the different formula used in the message passing algorithm, several kinds of message passing algorithms can be defined.

1.3.1 Belief Propagation (BP) algorithm

We first define the column index sets A(m) and the row index sets B(n)as follows:

$$A(m) \triangleq \{n | H_{mn} = 1\}$$

$$B(n) \triangleq \{m | H_{mn} = 1\}$$

For the AWGN channel with noise variance σ^2 and received signal y_i, the conditional probability of being $x_i = 0$ or $x_i = 1$ is represented as follows:

$$P(y_i \mid x_i = 0) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y_i - 1)^2}{2\sigma^2})$$

$$P(y_i \mid x_i = 1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y_i + 1)^2}{2\sigma^2})$$

The input to the message passing algorithm is initialized as message λ_n. As a Log Likelihood Ratio (LLR), the initial message λ_n for BPSK modulation can be represented as follows:

$$\lambda_n = \ln \frac{P(y_n \mid x_n = 0)}{P(y_n \mid x_n = 1)}$$

$$= \frac{\frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y_n - 1)^2}{2\sigma^2})}{\frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y_n + 1)^2}{2\sigma^2})}$$

$$= -\frac{(y_n - 1)^2}{2\sigma^2} + \frac{(y_n + 1)^2}{2\sigma^2}$$
$$= \frac{2y_n}{\sigma^2}$$

Figure 1.6 shows the flowchart of LDPC decoding algorithm. Following six steps shows the detailed process of message passing algorithm:

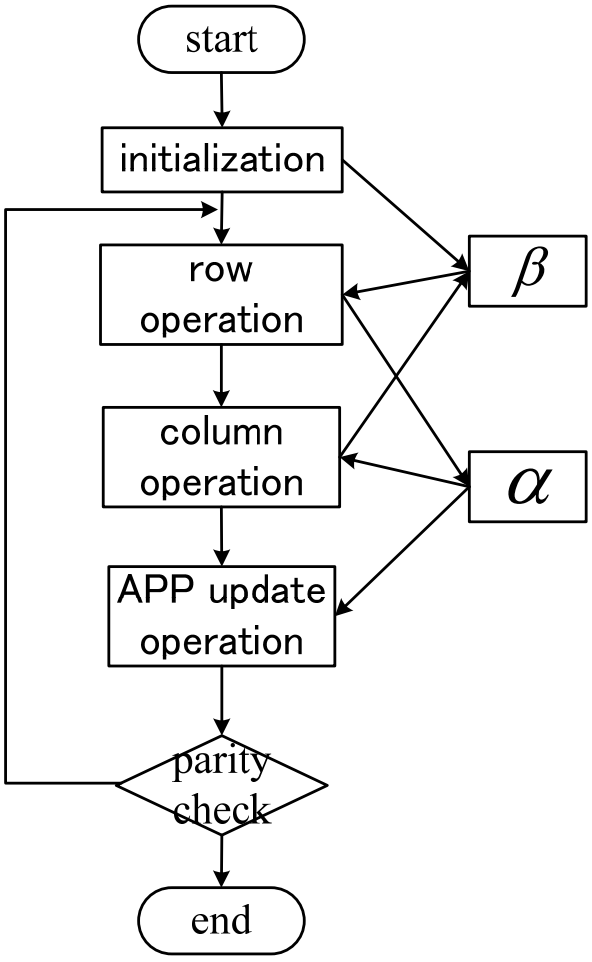


Figure 1.6 Flowchart of LDPC decoding algorithm

Let λ_n denote the Log-Likelihood Ratios (LLR) of the bit node *n* of the received codeword from the channel, α_{mn} be the message sent from check node *m* to bit node *n*, β_{mn} be the message sent from bit node *n* to check node *m*, and sum_n be the A Posteriori Probability (APP) message of the bit node *n* of the codeword.

Step 1 [initialization]

Compute the log likelihood ratio (LLR) λ_n for n = 1,2,…,N using the following equation, where σ^2 is variance of the noise generated by the AWGN Channel.

$$\lambda_n = 2y_n / \sigma^2$$

Set $\beta_{mn} = \lambda_n$ for each (m,n) satisfying $H_{mn}=1$. Set the loop counter l = 1, and the maximum number of iterations is set to l_{max}.

Step 2 [row operation]

It can also be called check node operation.

For all the check nodes Cm in the order from m=1,2,…,M, compute the intermediate messages α_{mn} according to the following equation, where each set (m,n) satisfies $H_{mn} = 1$.

$$\alpha_{mn} = 2\tanh^{-1}\left(\prod_{n'\in A(m)\backslash n}\tanh\left(\frac{\beta_{mn'}}{2}\right)\right) \qquad (1.2)$$

Step 3 [column operation]

It can also be called bit node operation.

For all the bit nodes bn in the order from n = 1,2,…,N, compute the message β_{mn} with the following equation, where each set (m,n) satisfies $H_{mn} = 1$.

$$\beta_{mn} = \lambda_n + \sum_{m'\in B(n)\backslash m}\alpha_{m'n} \qquad (1.3)$$

Step 4 [APP update operation]

Compute all the APP data sum_n for n = 1,2,.., N.

$$sum_n = \lambda_n + \sum_{m'\in B(n)}\alpha_{m'n} \qquad (1.4)$$

Step 5 [parity check]

Compute all the tentative LDPC code bits $\hat{y}_n$ for n = 1,2,.., N.

$$\hat{y}_n = \begin{cases} 0, sign(sum_n) = 1 \\ 1, sign(sum_n) = -1 \end{cases}$$

If the tentative code word $(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_N)$ satisfies following equation, output the code word, and terminate the message passing algorithm, otherwise go to Step 6.

$$H \cdot (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_N)^T = 0$$

Step 6 [increment counter]

If $l \le l_{\max}$, set the loop counter $l \leftarrow l+1$ and go to Step 2. Otherwise output the tentative code word $(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_N)$ and end the algorithm.

1.3.2 Min-sum Algorithm

In the row operation in the BP algorithm, there is a non-linear Hyperbolic function (Equation (1.2)), which is hard to be implemented in the hardware. The hardware for the Hyperbolic function can be realized by Look Up Table (LUP) or linear approximation [38][39][40]. A famous approach called min-sum (MS) algorithm which uses Equation (1.5) is widely used in the hardware implementation [18].

$$\alpha_{mn} = \prod_{n' \in A(m) \backslash n} sign(\beta_{mn'}) \times \min_{n' \in A(m) \backslash n} |\beta_{mn'}| \qquad (1.5)$$

The above equation shows that the min-sum algorithm only uses addition, minimization and XOR operation for the procedure, which is suitable for the hardware implementation.

1.3.3 Normalized min-sum (NMS) algorithm and offset min-sum (OMS) algorithm

Although MS algorithm can be easily implemented in hardware, it suffers a large performance degrading which encourages further researches to find better approximation based on the MS algorithm. For example, a normalization factor or offset factor is applied to the MS algorithm, which forms the well-known Normalized Min-sum algorithm (NMS) and Offset Min-sum algorithm (OMS) [19] [20] [21] [22] [23] [24].

The row operation with Equation (1.6) and Equation (1.7) is called normalized min-sum algorithm and offset min-sum algorithm, respectively.

$$\alpha_{mn} = \gamma \prod_{n' \in A(m) \backslash n} sign(\beta_{mn'}) \times \min_{n' \in A(m) \backslash n} | \beta_{mn'} | \tag{1.6}$$

$$\alpha_{mn} = \prod_{n' \in A(m) \backslash n} sign(\beta_{mn'}) \times \max(\min_{n' \in A(m) \backslash n} | \beta_{mn'} | - \varepsilon, 0) \tag{1.7}$$

The above equations also demonstrate an easy implementation in hardware with addition, minimization, XOR and shift operation.

1.4 Layered algorithm

Layered decoding algorithm is another family of LDPC decoding algorithm. It was first proposed in [26] with the name of Turbo Decoding Message Passing (TDMP) algorithm. Recently, because of the layer based operation, it is also called layered decoding algorithm [27]. A layer consists of several rows in the parity check matrix and particularly, the layer for structured parity check matrix can be the row-block of the parity check matrix. Layered decoding algorithm has fast convergence speed compared to message passing algorithm, therefore, the iterations needed to decode the codeword is decreased, thus increasing the throughput and decreasing the power [34].

Figure 1.7 shows the flowchart of layered algorithm. Layered algorithm is repeated for each horizontal layer and the updated A Posteriori Probability (APP) messages are passed between layers. Let λ_n denote the Log-Likelihood Ratios (LLR) of the bit node *n* of the received codeword from the channel, α_{mn} be the message sent from check node *m* to bit node *n*, β_{mn} be the message sent from bit node *n* to check node *m*, and sum_n be the APP message of the bit node *n* of the codeword and be initialized as λ_n. The detailed procedure can be described in the following seven steps:

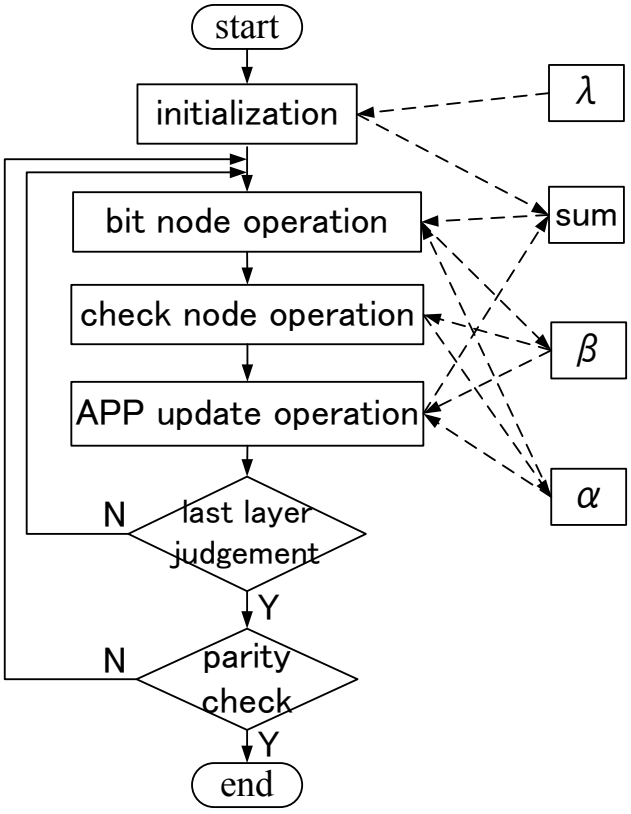


Figure 1.7 Flowchart of layered LDPC decoding algorithm

Step 1 [initialization]

Compute the log likelihood ratio (LLR) λ_n for n = 1,2,…,N using following equation, where σ^2 is variance of the noise generated by the AWGN Channel.

$$\lambda_n = 2y_n / \sigma^2$$

Set $sum_n = \lambda_n$ for each (m,n) satisfying $H_{mn}=1$. Set the loop counter l = 1, and the maximum number of iterations is set to l_{max}.

Step 2 [bit node operation]

For each bit node b_n inside the current layer in the order from n = 1,2,…,N, compute the message β_{mn} with the following equation, where each set (m,n) satisfies H_{mn} = 1.

$$\beta_{mn} = sum_n - \alpha_{mn}$$

Step 3 [check node operation]

For all the check nodes Cm in the current layer in the order from m=1,2,…,M, compute the intermediate messages α_{mn} according to the following equation, where each set (m,n) satisfies $H_{mn} = 1$.

$$\alpha_{mn} = \gamma \prod_{n' \in A(m) \setminus \{n\}} sign(\beta_{mn'}) \times \min_{n' \in A(m) \setminus \{n\}} |\beta_{mn'}|$$

where $A(m) = \{n | H_{mn} = 1\}$.

Step 4 [APP update operation]

The APP messages in the current layer are updated by:

$$sum_n = \beta_{mn} + \alpha_{mn}$$

Step 5 [last layer judgement]

Decide if there is next layer in this iteration. If there is, back to Step 2 to continue to do the operation for the next layer. If there is not, go to step 6.

Step 6 [parity check]

Compute all the tentative LDPC code bits $\hat{y}_n$ for n = 1,2,.., N.

$$\hat{y}_n = \begin{cases} 0, sign(sum_n) = 1 \\ 1, sign(sum_n) = -1 \end{cases}$$

If the tentative code word $(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_N)$ satisfies following equation, output the code word, and terminate the message passing algorithm.

$$H \cdot (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_N)^T = 0$$

Step 7 [increment counter]

If $l \le l_{\max}$, set the loop counter $l \leftarrow l + 1$ and go to Step 2. Otherwise output the tentative code word $(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_N)$ and end the algorithm.

1.5 Design challenges of LDPC decoders

Although many works have been done on LDPC decoder design, there still remains

challenges to design the decoder for specific applications.

1.5.1 BER performance

As an error correcting technique, the BER performance is always an important metric for the design. The higher the BER performance is, the higher the error correcting ability is. As for some communication system which demands very high error correcting ability, the design becomes extremely difficult [42] [43].

Basically, BP algorithm can provide the best BER performance, but it is not hardware-friendly due to the need of implementation of a non-linear Hyperbolic function (Equation (1.2)). Basically it can be implemented by Look Up Table (LUT) or linear approximation [38][39][40]. However, not only the hardware implementation of LUT or linear approximation consumes large hardware cost, but also the BER performance will degrade through the approximations.

On the other hand, MS algorithm, NMS algorithm and OMS algorithm can be implemented in quite simple hardware, but the BER performance is degraded compared to BP algorithm which is certainly not enough for some applications which require very high error correcting performance such as the broadcasting application [18][19].

1.5.2 Hardware cost

Compared to other error correcting codes like convolutional code or turbo code, LDPC code has the best error correcting ability. However, the largest issue for LDPC decoder is the relatively large area for implementing because of the large quantity of message exchanges between the check nodes and bit nodes. The storage used to store the intermediate messages consumes almost half of all the LDPC decoder which is also a main reason for the large hardware cost [14]. The reason why LDPC code has

not completely taken place of all the other error correcting code in the real application is because of the large hardware cost problem. So from this point of view, partially-parallel LDPC decoder which can reuse the function units and has relatively small area has become more and more popular [13] [14] [25].

On the other hand, the row operation of LDPC decoding algorithm performs an important role in the complexity of LDPC decoder. As introduced in Section 1.5.1, BP algorithm can achieve good BER performance but with large hardware cost, and the hardware cost of MS algorithm, NMS algorithm, OMS algorithm is relatively small but the BER performance is also degraded [18] [19]. There is a tradeoff between the BER performance and hardware cost, and it is still a challenge in LDPC decoder design.

1.5.3 Throughput

Because of the growing need of small hardware for manufacture process, partially-parallel LDPC decoder is basically used for modern LDPC decoder design. But the throughput is relatively low compared to the fully-parallel LDPC decoder [10] [11] [14]. However, high throughput is required to ensure the large amount of data transmission in real-time communication systems. There is a tradeoff between hardware cost and throughput, and it is another big challenge in LDPC decoder design.

1.6 Motivation of this work

As introduced in Section 1.3, LDPC code is very suitable for hardware implementation by utilizing a parallel decoding algorithm called Message-Passing (MP) algorithm [1]. In the last few years, researches have been done on designing specific decoder architectures for LDPC implementations, seeking for the best

trade-off between area, power consumption and performance [10] [11] [12] [13] [14]. The LDPC decoders that have been designed share the same primitive processing elements: check functional unit (CFU) performing row operations for check nodes and bit functional unit (BFU) performing column operations for bit nodes. These processing elements are connected according to the Tanner graph, a graph representing the relation between bit nodes and check nodes. The MP algorithm exchanges messages between check nodes and bit nodes by performing row and column operations iteratively.

The authors in [13] proposed an accelerated message-passing schedule, which only performs those column operations whose corresponding check nodes have been updated by the row operations, which is demonstrated to be more efficient than the basic algorithm. A partially-parallel LDPC decoder based on the accelerated MP schedule is introduced in [14] to support for irregular LDPC code for IEEE802.11n application. Although the partially-parallel irregular LDPC decoder proposed in [14] can improve the error correction performance, it suffers a main problem that the overall throughput is relatively low compared with the regular or fully-parallel irregular LDPC decoder (eg. [15], [16]). Considering the fact that throughput is an essential metric in real-time communication systems applying LDPC code for IEEE802.11n application, good tradeoff between hardware cost and throughput for irregular LDPC decoder is required.

On the other hand, LDPC code is also used in the Integrated Services Digital Broadcasting via Satellite - Second Generation (ISDB-S2) application in Japan. Previous decoding algorithms for LDPC decoders can either achieve very high BER performance by implementing very complicated hardware (Belief-Propagation –based algorithms)[29][30], or achieve small area cost in the sacrifice of degrading BER performance (Min-sum –based algorithms)[31][32]. However, as long as the stable transmission is necessary for satellite broadcasting, very high BER performance is a key request for the design. But as for the manufacture demand, Belief-Propagation –based algorithms with large area cost is not a good option for LDPC decoder design for ISDB-S2. So a good tradeoff between BER performance and hardware cost for

LDPC decoding algorithm targeting ISDB-S2 is another topic of this work.

1.7 Contribution

In this dissertation, we mainly focus on two issues: the LDPC decoder design for IEEE 802.11n application targeting high throughput and low hardware cost; an improved LDPC decoding algorithm for ISDB-S2 LDPC decoder targeting high BER performance and low hardware cost.

1.7.1 High throughput partially-parallel irregular LDPC decoder

In this issue, we propose a partially-parallel irregular LDPC decoder for IEEE 802.11n standard targeting high throughput applications. The proposed decoder has several merits:

(i) The decoder is designed based on a novel delta-value based message passing schedule which facilitates the decoding throughput by redundant computation removal.

(ii) Techniques such as binary sorting, parallel column operation, high performance pipelining are used to further speed up the message-passing procedure.

The synthesis result in TSMC 0.18 CMOS technology demonstrates that for (648,324) irregular LDPC code, our decoder can achieve 8 times increase in throughput, reaching 418 Mbps at the frequency of 200MHz.

1.7.2 An improved design for low area cost application

Based on the proposed delta-value based message passing schedule, we propose

another schedule called sum-delta message passing schedule and a partially-parallel irregular LDPC decoder targeting high throughput and small area application is implemented. The design of this decoder is characterized as follows:

(i) Decoding throughput is greatly improved by utilizing the difference value between the updated and the original value to remove redundant computations.

(ii) Registers and memories are optimized to store only the frequently used messages to decrease the hardware cost.

The synthesis result in TSMC 0.18 CMOS technology demonstrates that for (648,324) irregular LDPC code, our decoder achieves 7.5X improvement in throughput, which reaches 404 Mbps at the frequency of 200MHz, with 11% area reduction. The synthesis result also demonstrates the competitiveness to the fully-parallel regular LDPC decoders in terms of the tradeoff between throughput, area and power. The backend design of this decoder is implemented using TSMC 0.18μm technology and the layout area is 13.69mm^2.

1.7.3 Self-adjustable offset min-sum algorithm targeting high BER performance and low hardware cost

A novel self-adjustable offset min-sum LDPC decoding algorithm is proposed for ISDB-S2 (Integrated Services Digital Broadcasting via Satellite - Second Generation) application. We present for the first time a uniform approximation of the check node operation through mathematical induction on Jacobian logarithm. The approximation theoretically shows that the offset value is mainly dependent on the difference between the two most unreliable inputs from the bit nodes and the algorithm proposed can adjust the offset value according to the inputs during the iterative decoding procedure. Simulation results for all 11 code rates of ISDB-S2 demonstrate that the proposed method can achieve an average of 0.12dB gain under the same Bit Error Rate (BER) performance, compared to the Min-sum based algorithms, and can save 98.79% computation complexity compared to BP-based algorithms in the best case.

1.7.4 Data conflict resolution for layered schedule targeting high BER performance

Layered LDPC decoding algorithm is known to achieve high Bit Error Rate (BER) performance and high throughput for LDPC decoders. However, for ISDB-S2 (Integrated Services Digital Broadcasting via Satellite - Second Generation) LDPC decoder, applying layered algorithm directly will result in data conflict problem. In this work, a novel selective recalculation method is proposed to solve the data conflict problem. It determines the inaccurately calculated values based on a recalculation decision rule, and correct them accordingly. By applying this selective recalculation method, the layered algorithm can achieve conflict free BER performance. Simulation results of applying selective recalculation method to the layered schedule of the proposed algorithm introduced in Section 1.7.3 demonstrate that the proposed method can achieve further BER performance improvement, which can achieve an average of 0.2dB gain compared to the MS-based algorithms.

1.8 Organization of the dissertation

The rest of this thesis is organized as follows:

Chapter 2 describes a high throughput partially parallel LDPC decoder based on Delta-Value based Message Passing Schedule. Chapter 3 provides an improved design targeting high throughput and low area cost design based on Sum-Delta Message Passing Schedule. Chapter 4 provides the idea of self-adjustable offset min-sum algorithm and shows the simulation result. Chapter 5 introduces the idea of date conflict resolution by selective recalculation for layered LDPC decoding algorithm. Finally Chapter 6 makes a conclusion over all.

2 A partially-parallel LDPC decoder targeting high throughput

2.1 Introduction

The authors in [13] proposed an accelerated message-passing schedule, which only performs those column operations whose corresponding check nodes have been updated by the row operations, which is demonstrated to be more efficient than the basic message-passing algorithm. A partially-parallel LDPC decoder based on the accelerated MP schedule is introduced in [14] to support for irregular LDPC code. Although the partially-parallel irregular LDPC decoder proposed in [14] can improve the error correction performance, it suffers a main problem that the overall throughput is relatively low compared with the fully-parallel LDPC decoder (eg. [15][16]). Moreover, the throughput of current partially parallel irregular LDPC decoders designed, such as 54Mbps of [14], is not sufficient for most applications of 802.11n standard [17], which requires a throughput of up to 330Mbps. From this observation, there is room for the improvement for partially-parallel irregular LDPC decoders.

In this chapter, we propose an improved MP algorithm and decoder architecture for irregular LDPC decoder, in this paper, to achieve a nearly 8X speedup with almost the same BER performance.

The novelties of this decoder in terms of high throughput are as follows:

• The proposed LDPC decoder is based on a novel delta-value message-passing algorithm suitable for high throughput design.

• An improved binary sorting scheme is designed in row operation to reduce the computation time.

• A parallel structure of bit function unit is designed to speed up the column operation.

• A high performance pipeline structure is used to further speed up the message-passing procedure.

The synthesis result in TSMC 0.18μm CMOS technology demonstrates that for (648,324) irregular LDPC code, our decoder can achieve 8 times increasement in throughput, reaching 418 Mbps at the frequency of 200MHz.

2.2 Proposed delta-value based message passing (DVMP) schedule

In this section, the accelerated message passing algorithm is illustrated through an example and an improved message passing schedule is proposed based on the accelerated message passing algorithm which can achieve high throughput application.

2.2.1 Accelerated message passing algorithm

In this section, we use an example to demonstrate the problem of the accelerated message-passing algorithm.

The accelerated message passing schedule, proposed in [13], is well suitable for irregular LDPC decoder design. The flowchart of the accelerated message passing algorithm is shown in Figure 2.1(a), with the solid line showing the flowchart of the algorithm stages and the dotted line showing the data transmission of the storages. Each iteration of the algorithm is composed of four phases: row operation, column operation, error correction and parity check. The row operation updates message α of all check nodes using message β, and sends the message to bit nodes. The column

operation updates message β of all bit nodes based on message α and initial message λ. The error correction calculates the tentative decision and the parity check operation decides whether another decoding iteration is necessary or not according to the tentative decision.

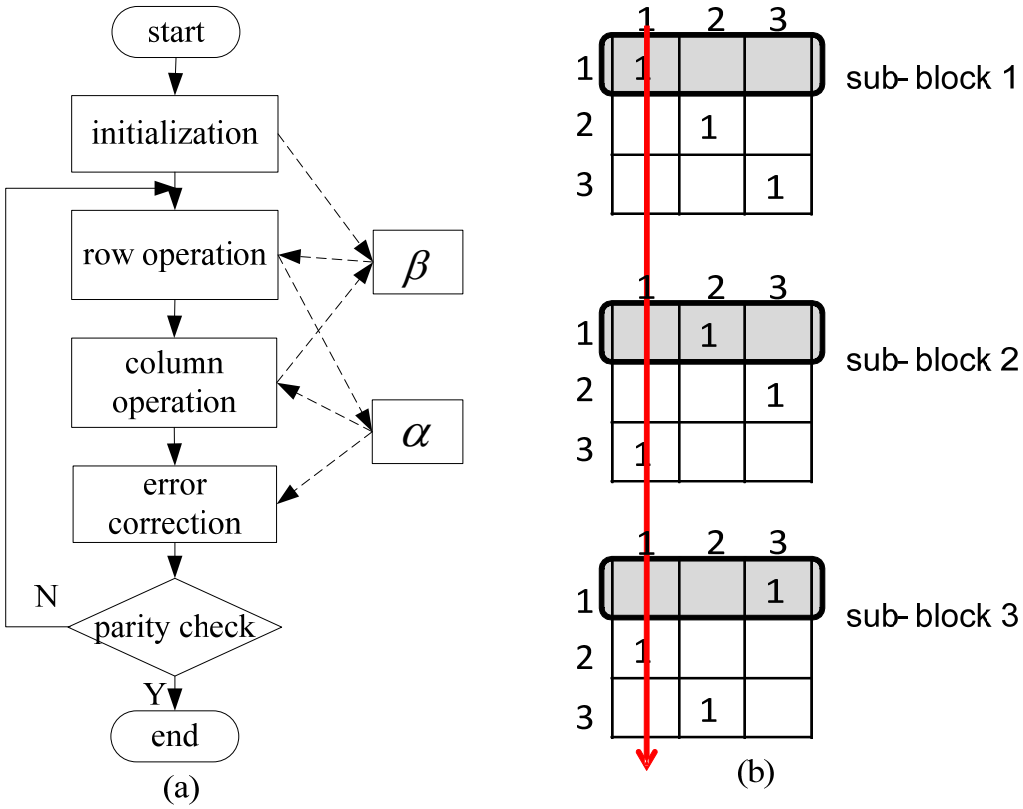


Figure 2.1 Accelerated message passing algorithm: (a) flowchart of accelerated MP algorithm and (b) parity check matrix of motivation example

The idea is illustrated through an example demonstrated in Figure 2.1(b). In a simple 3×1 parity check matrix, of which each sub-block is a 3×3 matrix, the row operations for all the first rows in each sub-block is executed first. Then three α values are updated, which are $\alpha_1(1,1)$, $\alpha_2(1,2)$ and $\alpha_3(1,3)$ respectively. In the accelerated MP algorithm, the calculation of β values will be executed right after the first row operation. The computations of the first column are shown as follows:

$$\beta_1'(1,1) = \lambda(1) + \alpha_2(3,1) + \alpha_3(2,1)$$
$$\beta_2'(3,1) = \lambda(1) + \alpha_1(1,1) + \alpha_3(2,1)$$
$$\beta_3'(2,1) = \lambda(1) + \alpha_1(1,1) + \alpha_2(3,1)$$

2.2.2 Motivation

If we consider these column operations that calculate the updated message β, only a small part of the operands have been changed since last computation of the same message β. For example, only $\alpha_1(1, 1)$ is changed for calculation of $\beta_2(3, 1)$ and $\beta_3(2, 1)$ and no operand is changed for calculation of $\beta_1(1, 1)$. A considerable part of addition operations in column operation, in fact are a repetition of former computations. And for the targeting parity check matrix, when $|B(n)|$ is large, this problem becomes more significant as more useless additions are operated. The above observation demonstrates that accelerated MP scheme still has computational redundancies, which degrade the efficiency of hardware implementation.

2.2.3 Delta-Value Based Message Passing Schedule

In accelerated MP algorithm, the computation of β for columns with a weight of 12 in the targeted parity check matrix ($|B(n)| = 12$), requires the addition of 11 β values and one λ value according to Equation (1.3) [13] [14]. This results in a five-depth addition and thus becomes the bottleneck to get high throughput designs. However, we propose an improved MP schedule called Delta-Value based Message-Passing (DVMP) schedule to help solve this bottleneck problem and thus increases the throughput.

We first introduce the concept of a new message $\Delta\alpha$, as follows.

$$\Delta\alpha(m,n) = \alpha'(m,n) - \alpha(m,n) \tag{2.1}$$

Back to example in Figure 2.1(b), the calculation can be simplified by adding only the updated delta-value of message α shown as follows.

$$\beta_1'(1,1) = \beta_1(1,1)$$
$$\beta_2'(3,1) = \beta_2(3,1) + \Delta\alpha_1(1,1)$$
$$\beta_3'(2,1) = \beta_3(2,1) + \Delta\alpha_1(1,1)$$

This new calculation method of message β can remove redundant computations and reduce the total number of additions in the bottleneck column operation, especially when $|B(n)|$ becomes larger. For the targeted parity check matrix [17], at most two message α are updated after certain row operation. Based on this observation, only a small number of updated α value is sufficient to generate a correct message β. Therefore, we proposed our DVMP algorithm by only calculating the delta value of updated α in the row operation to improve the decoding efficiency. In such scheme, the resulting β can be obtained by at most 2 levels of addition in DVMP rather than 5 levels in the accelerated MP scheme [13], [14].

The proposed DVMP algorithm is shown in Figure 2.2, with the solid line showing the flowchart of the algorithm stages and the dotted line showing the data transmission of the storages. In the initialization step, message β is initialized as message λ. Then the row operation is executed to update message α according to Equation (1.6) and generate corresponding $\Delta\alpha$ according to Equation (2.1) at the meantime. Next, the column operation calculates message β by the updated α values as Equation (2.2). The error correction calculates the tentative decision, based on which a parity check is done after each iteration to determine the termination of the decoding process.

$$\beta'(m,n) = \beta(m,n) + \sum_{m' \in D(n) \setminus m} \Delta\alpha(m',n) \tag{2.2}$$

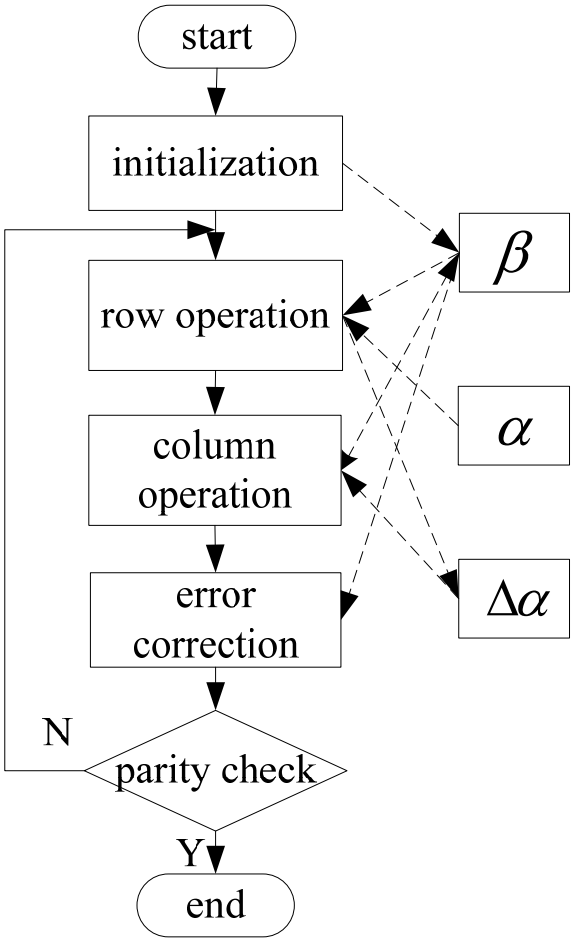


Figure 2.2 Flowchart of proposed delta-value message passing algorithm

In Equation (2.2), D(n) are those row numbers of which the message α has been updated since the last computation of $\alpha(m, n)$ in column n. The total number of possible addition is reduced from $|B(n)|$ to $|D(n)|$. And in the situation of the targeted irregular decoding matrix, this redundancy removal can improve the column operation by a saving of three level additions, which reduce both the computation time and area of hardware implementation.

2.3 Proposed high throughput design

Apart from the high throughput DVMP schedule, we integrated three different strategies into the proposed decoder to further speed up the decoding process. In row operation module, a binary sort scheme is used to shorten the comparison time. In column operation module, parallel DVMP schedule is used to save the processing time. Furthermore, a pipeline structure is utilized to speed up the whole processing procedure.

2.3.1 High throughput row operation module

In the calculation of a value according to Equation (1.6), the minimum and second minimum β value among a total of $|A(m)|$ values in the same row should be obtained through a proper comparison scheme. In previous designs [13], [14], simple sorting algorithms are used to obtain the minimum values, such as the implementation of bubble sort comparison scheme illustrated in Figure 2.3. Eight β values ($|A(m)|$ = 8 for the targeted matrix) are compared serially to the $min1$ and $min2$ to find the minimum value and the second minimum values from the eight β values. $min1$ and $min2$ at the input side of Figure 2.3 are the initial values of the registers used for storing the minimum value and the second minimum value.

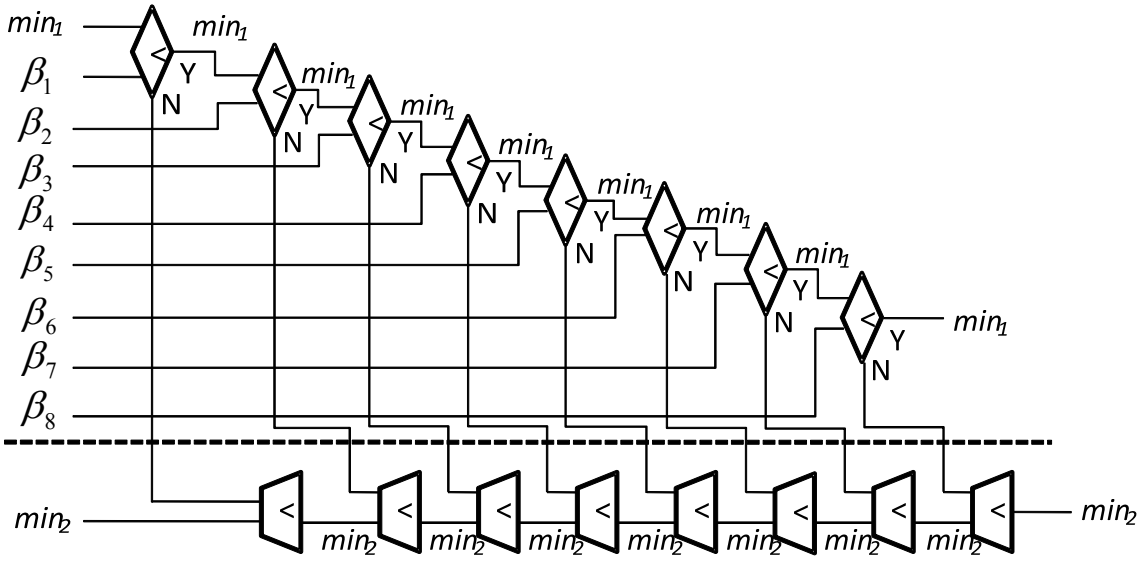


Figure 2.3 Previous comparison scheme in row operation module

The diamond symbol in Figure 2.3 is a decision process which feeds β to different output branches based on the relation between β and $min1$. For example, $\beta 1$ is compared with $min1$ first and the register value of $min1$ is replaced with $\beta 1$ if $\beta 1$ is smaller, or otherwise sent to compare with $min2$. In this way, the total number of clock cycles to get the minimum and second minimum value is 9. These serial comparison steps, however, requires considerable computational time and becomes the most time consuming part in row operation when the number of message β increases greatly in the targeted matrix.

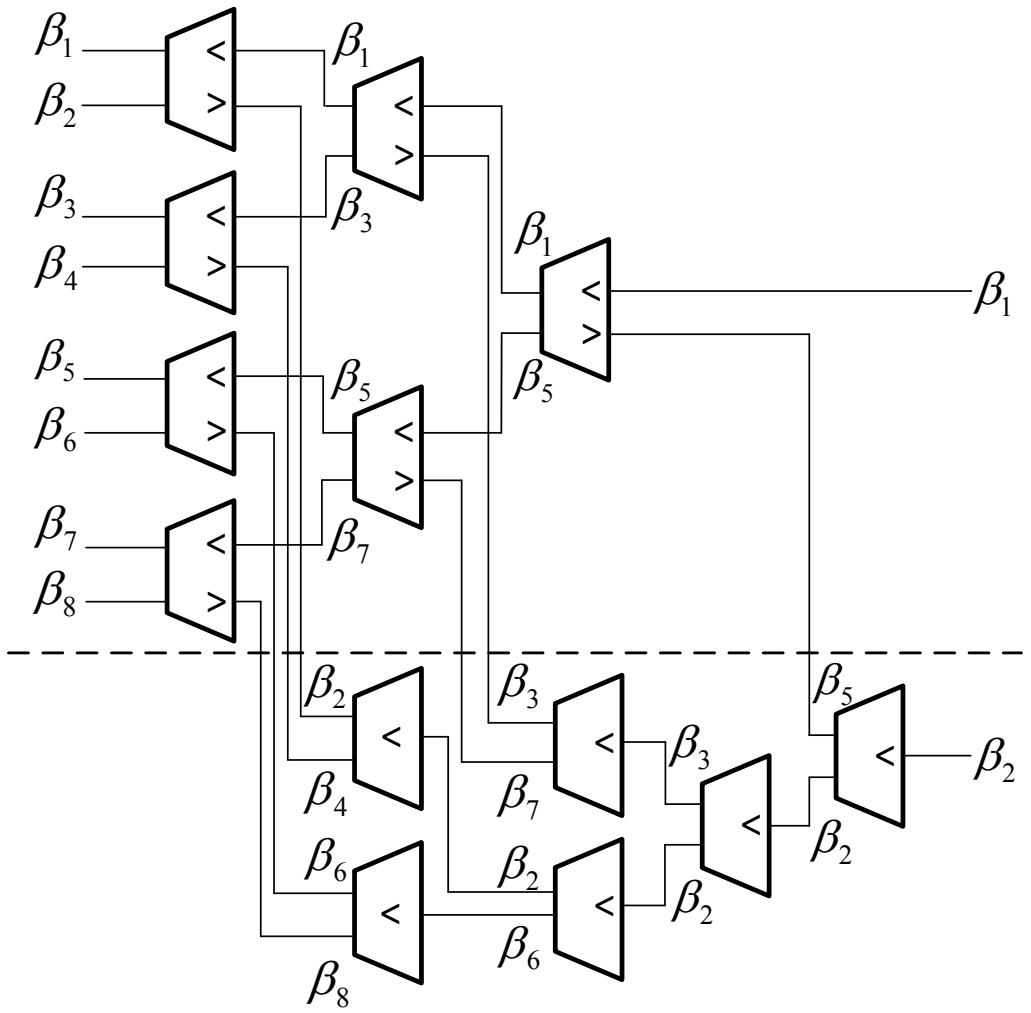


Figure 2.4 Proposed comparison scheme in row operation module

In order to improve the hardware performance, a tree structure is proposed, together with a binary sort to obtain two minimum values in parallel. The modified sorting scheme requires less area and can generate the result in only five comparison steps. The detailed comparison procedure in row operation module is shown in Figure 2.4. The upper tree and the lower tree represent respectively the sorting for the minimum and the second minimum β value among a total of eight values. In the figure, we assume that the values of eight β at the same row ($\beta 1$, $\beta 2$, . . . , $\beta 8$) comply with the relationship of $\beta 1 < \beta 2 < \ldots < \beta 7 < \beta 8$ and the resulting

values are labelled on each data path. In the first step, eight β values are compared in pairs. The smaller one (*e.g.*, $\beta 1$) is remained in the sorting tree for the minimum value while the larger one (*e.g.*, $\beta 2$) is eliminated to the sorting tree for the second minimum value. Then the values in both trees will continue the comparison process in pairs with the larger one in the upper tree eliminated to the lower tree. In this manner, we can get the minimum and the second minimum value at the third and the fifth step of the operation. Compared with a total 9 clock cycles in [14], our proposed scheme can complete the comparison in only 2 clock cycles under the frequency of 200MHz, a 4.5X speedup.

2.3.2 High throughput column operation module

As discussed in Section 2.2.3, the DVMP-based column operation can update β values by the addition of its original β and at most two updated $\Delta\alpha$ values according to Equation (2.2). The number of updated $\Delta\alpha$ values and the exact position of each updated $\Delta\alpha$ in the column operation are determined by the parity check matrix. Therefore, Equation (2.2) is simply achieved by two addition steps during implementation. In the first addition step, at most two updated $\Delta\alpha$ values is added together for the targeting LDPC code in IEEE802.11n [17]. Then we compute the new β value by adding the original one and the sum of $\Delta\alpha$ in the second step. By adapting the DVMP schedule, not only the computation time of each column operation is minimized, but also the area of hardware implementation is reduced. Based on this observation, we can further improve the throughput by applying a parallel column operation here.

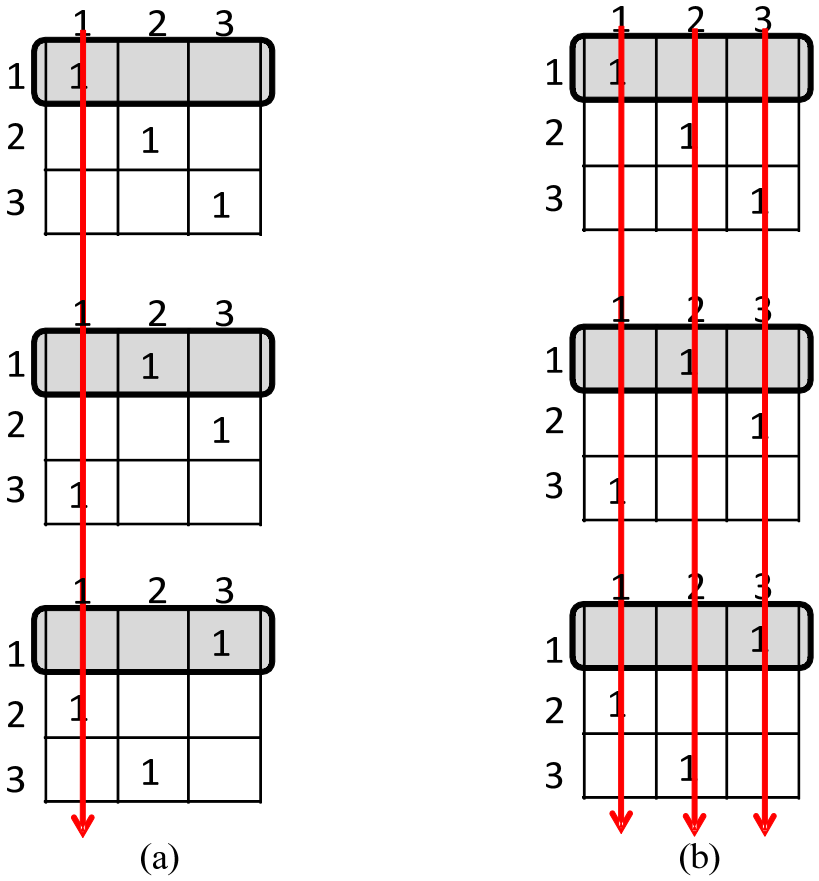


Figure 2.5 Example of column operation: (a) column after column scheme (b) parallel scheme

In previous implementation of the accelerated MP schedule [13], [14], all N column-blocks are computed in parallel while different columns in one column-block are calculated in serial as shown in Figure 2.5(a). After the first row operation is calculated, which is shown as gray rectangulars, column operations are calculated one by one. This processing procedure requires at most 11 clock cycles for column computation in the real implementation of the targeting LDPC code [14]. In the proposed module, message β of all columns, in which there exists an updated α, are computed in parallel, as shown in Figure 2.5(b). A hardware implementation for computing k columns in parallel is shown in Figure 2.6. The resulting parallel DVMP-based column operation can be computed in one clock cycle, which is 11 times faster than the implementation in [14].

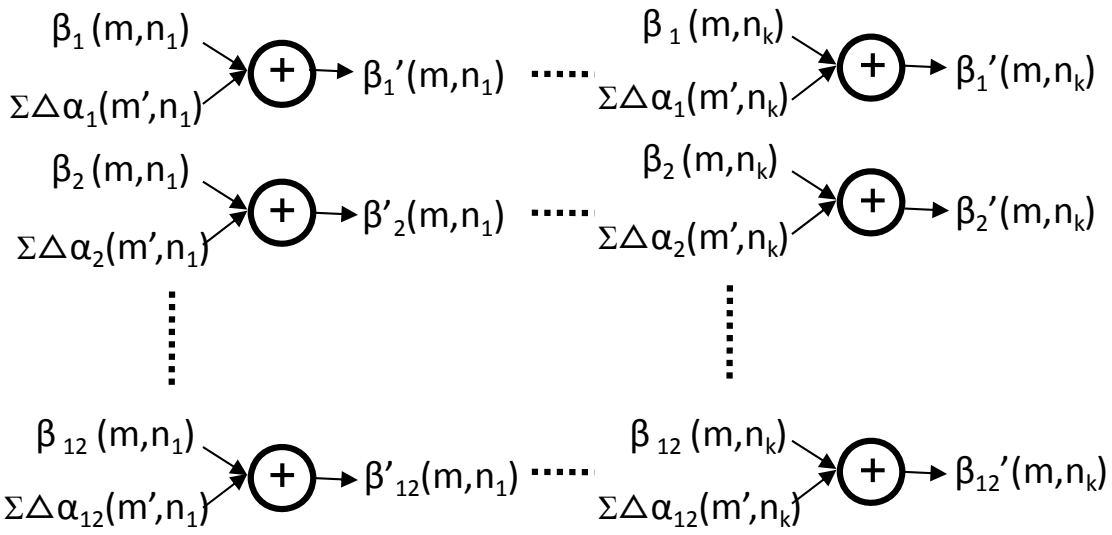


Figure 2.6 Architecture of proposed DVMP-based column operation module

2.3.3 High throughput pipeline schedule

In the proposed decoder, pipeline structure is utilized to achieve further speed up of the procedure, as shown in Figure 2.7. The row operation and column operation (including message read and write) are divided into four and three pipeline stages respectively to balance the computation time of each stage. After a further overlap of the row and column operation based on data dependency information, the message of a particular row operation and corresponding column operation can be updated after four clock cycle.

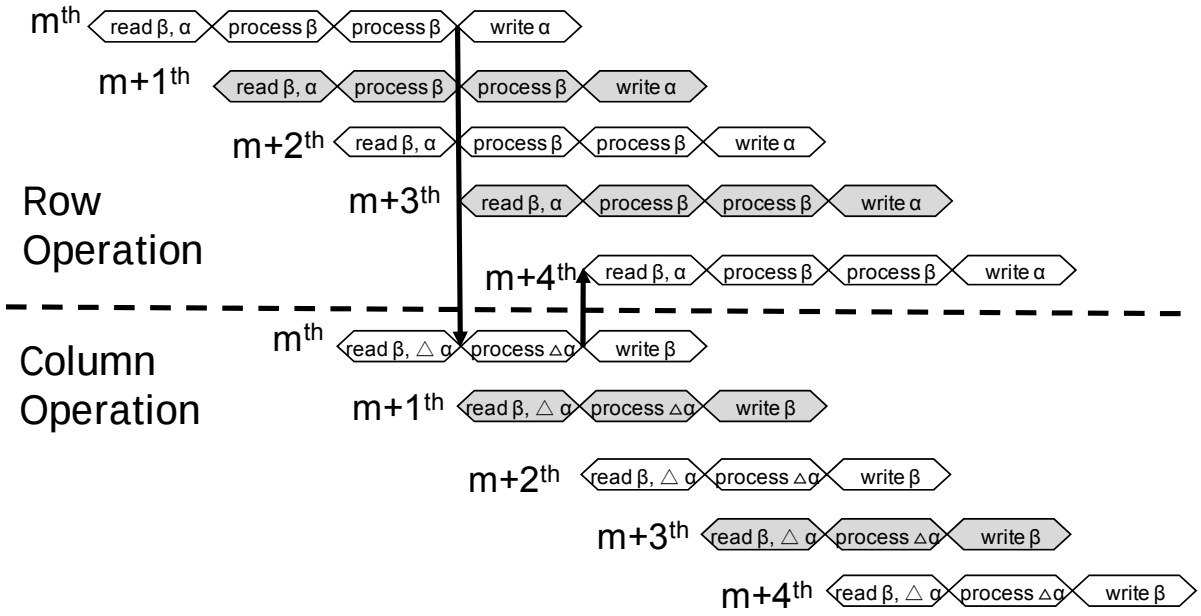


Figure 2.7 Timing schedule of the decoder applying DVMP schedule

During the first clock cycle, α and β values are read in to prepare for a row operation. In the second and third clock cycles, a tree structure comparison is conducted for the minimum and second minimum β value, and corresponding α and $\Delta\alpha$ values are obtained. In the fourth clock cycle, all related column operations are computed based on $\Delta\alpha$ values, and α values are written into memories at meantime.

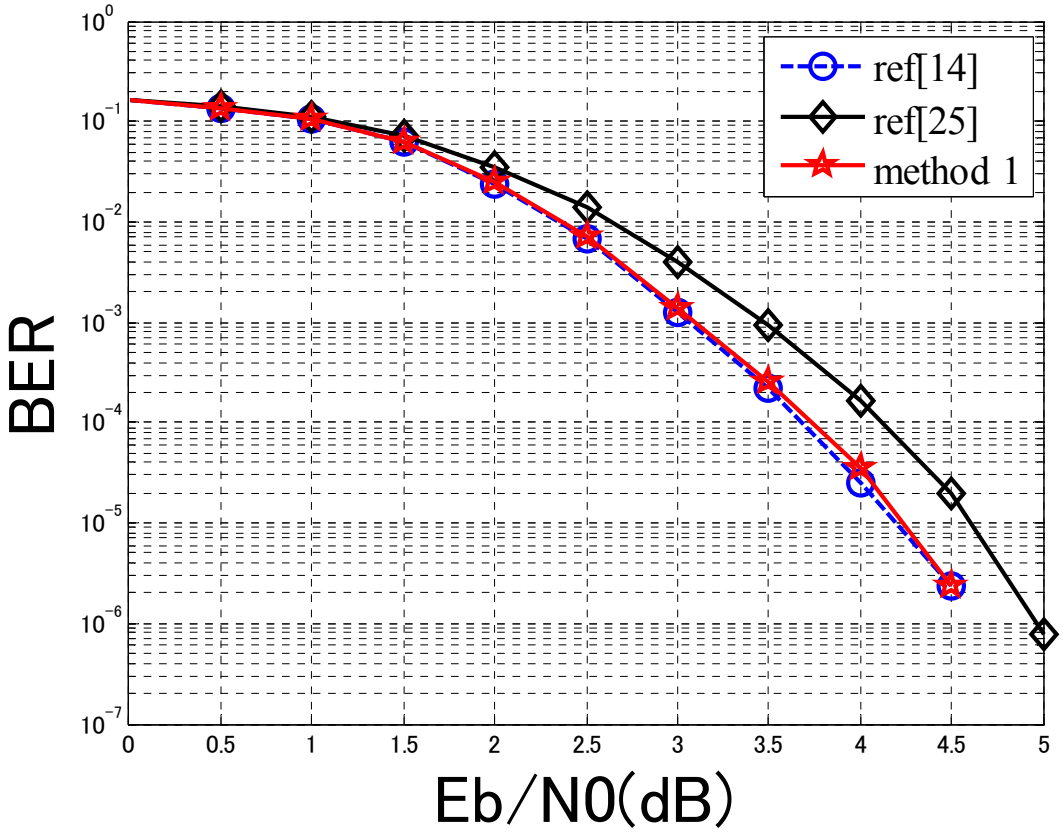


Figure 2.8 Bit error rate performance of the decoder applying DVMP schedule

Based on this proposed pipeline structure, the updated message of a specific row can be used after four clock cycles, in other words, the updated messages of m^{th} row can be used by $(m + 4)^{th}$ row operation. In [25], the updated messages of m^{th} row can be used by $(m+27)^{th}$ row operation. Although in [14], the updated message of the same row is used by $(m+3)^{th}$ row operation, our pipeline structure incurs nearly no performance degrading compared to [14] as illustrated in Figure 2.8. The reason of the gap in the figure is that the updated message of m^{th} row can be used at $(m+4)^{th}$ row other than $(m+3)^{th}$ row in [14], and the sooner the updated messages are used the better the performance will be. Furthermore, the proposed pipeline structure is more compact, which improves both the hardware utilization and throughput.

2.4 Implementation and result

In this section, the hardware implementation of the proposed partially-parallel irregular LDPC decoder and the synthesis results are presented.

2.4.1 Implementation details

The proposed decoder is mainly composed of five parts: row operation modules, column operation modules, parity check, controller and storage parts (memory for α value, registers for β value, $\Delta\alpha$ value and tentative decision value), as shown in Figure 2.9.

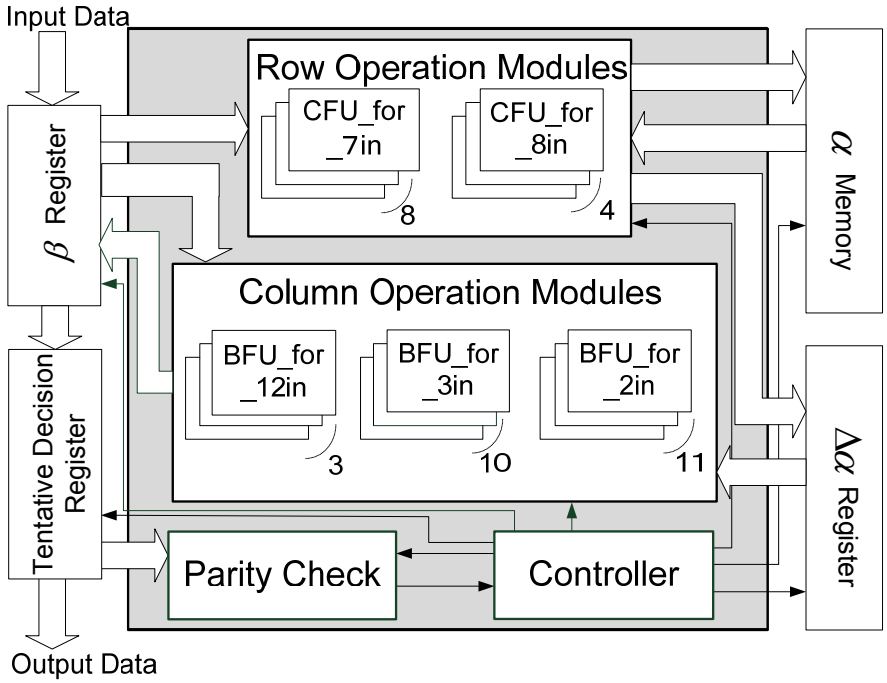


Figure 2.9 Block diagram of the decoder applying DVMP schedule

In the proposed decoder, we design 12 CFUs and 24 BFUs in order to execute the computation of the same row or column in each sub-block in parallel. Since the targeted LDPC decoding matrix is irregular, two different row operation modules (CFU_for_7in and CFU_for_8in), and three different column operation modules (BFU_for_12in, BFU_for_3in, and BFU_for_2in) are designed for rows and columns

with different $|A(m)|$ and $|B(n)|$. The controller module generates the control signals for storage and operation modules, while parity check module does the error correction and the parity check at the end of each iteration. Message α is only used in the row operation and is stored in memory with corresponding row address. Other messages like β, which have multiple-access problem, are stored in registers.

Table 2.1 Composition of the decoder core implementing DVMP schedule

Module	without DVMP		with DVMP	
	Number of Gates	Percentage (%)	Number of Gates	Percentage (%)
Row operation modules	49,902	41.17	53,042	53.00
Column operation modules	60,903	50.24	36,619	36.59
Parity check	9,285	7.66	9,258	9.28
Controller	1,124	0.93	1,124	1.13
LDPC decoder core	121,214	100	100,070	100

The detailed composition of the proposed decoder core is listed in Table 2.1, under the column with DVMP. The result is compared to the implementation without DVMP, and we can see from the table, the former computation-intensive column operation modules are reduced because of the use of DVMP algorithm.

2.4.2 Synthesis result

The proposed decoder is implemented under TSMC 0.18um CMOS technology. The synthesis results are listed in Table 2.2. Ref. [10] is a partially-parallel irregular decoder whose code length is 8088 bit. And we compare our work with [13], [14] under the same LDPC code and the same design rule. Under column *Ref. [14]* is the

synthesis result of a partially-parallel irregular LDPC decoder designed in [14], which is the only design known, targeting the same newly proposed LDPC code in [17]. We also modified the design in [13] to support the same irregular code with detailed synthesis result under column *Ref. [13]**.

Table 2.2 Synthesis result of the LDPC decoder implementing DVMP schedule

	Ref. [10]	Ref. [13]*	Ref. [14]	**Proposed**
Design rule	TI 0.11μm	TSMC 0.18μm		
LDPC code	8088 bit rate 1/2 irregular	802.11n 648 bit rate 1/2 irregular		
LDPC decoder	Partially-parallel			
Throughput	188Mbps (itr=25, SNR=NA)	54Mbps (itr= 5, SNR = 3.0)	54Mbps (itr= 5, SNR = 3.0)	**418Mbps (itr= 5, SNR = 3.0)**
Frequency	212MHz	200MHz	200MHz	**200MHz**
Memory area	407Kgates	708Kgates	502Kgates	**170Kgates**
Area w/o wiring	742Kgates	832Kgates	611Kgates	**423Kgates**
Area w/. wiring	N/A	13,090,549μm^2	9,004,366μm^2	**12,930,433μm^2**
Power(mW) @200MHz,1.6V	N/A	765.85	486.44	**893.18**

Because of the pipeline structure along with the improved row and column operation modules, our proposed decoder requires only 31 clock cycles for a single iteration and five iterations for codeword correction under SNR of 3.0dB, which can achieve 8 times throughput than [13], [14]. And it can also achieve more than twice the throughput than [10] and also requires much smaller gate counts than [10].

2.5 Conclusion

In this chapter, a novel high throughput partially-parallel irregular LDPC decoder is proposed. Row operations and column operations are speeded up by a modified binary searching scheme and delta-value based message-passing schedule respectively. Moreover a pipeline structure is utilized to further compact the procedure. The synthesis result demonstrates that our decoder can achieve a much higher throughput and almost the same bit error rate performance compared to other partially-parallel irregular LDPC decoders.

3 A partially-parallel LDPC decoder targeting high throughput and low hardware cost

3.1 Introduction

Although the decoder proposed in Chapter 2 can achieve a throughput of 418Mbps which meets the requirements for all the applications for IEEE802.11n, the hardware overhead is still a challenge in the design. In this chapter, we propose a partially-parallel irregular LDPC decoder based on the decoder proposed in Chapter 2 and target high throughput and small area applications. The design is based on a novel sum-delta message passing algorithm characterized as follows:

• The decoder designed is based on a novel sum-delta message passing algorithm well suitable for high throughput and low area design.

• The decoding process is further speeded up by an improved binary sorting scheme for row operation and parallel computation for column operation using the proposed sum-delta message passing algorithm. An efficient pipeline structure is utilized to boost message-passing throughput.

• The proposed sum-delta message passing algorithm can effectively reduce the storage area for messages passing through computation units by storing only the frequently used messages, thus saving the total area.

The synthesis result in TSMC 0.18 CMOS technology demonstrates that for

(648,324) irregular LDPC code, our decoder achieves 7.5X improvement in throughput, which reaches 404 Mbps at the frequency of 200MHz, with 11% area reduction. The synthesis result also demonstrates the competitiveness to the fully-parallel regular LDPC decoders in terms of the tradeoff between throughput, area and power.

3.2 Proposed sum-delta message passing (SDMP) schedule

Through observing the accelerated message passing algorithm, we found in Equation (1.3) where message β is obtained by the addition of λ and all connected message α of the same column except the one at the same row. Similar computations exist among the calculation of different message β at the same column, such as $\beta_1(1, 1)$, $\beta_2(3, 1)$ and $\beta_3(2, 1)$ in the example in Figure 2.1(b).

Based on DVMP schedule, we present our SDMP schedule in this section to address the efficiency issues. The proposed algorithm can improve the overall throughput and save storage area. We first introduce the concept of a new message *sum*, as follows.

$$sum(n) = \lambda(n) + \sum_{m \in B(n)} \alpha(m,n) \tag{3.1}$$

Compared with Equation (1.3), β value can be computed as follows.

$$\beta(m,n) = sum(n) - \alpha(m,n) \tag{3.2}$$

Then message *sum* instead of message β is stored and passed through different computation modules to form a new message passing scheme and save computation cost. The corresponding storage is also reduced from at most 12 message β to a single message *sum*. We further simplify the computation of message *sum* by

introducing $\Delta\alpha$ values. In Figure 3.1(a), let us revisit the example in Figure 2.1(b), as can be noticed from the example that during the computation of message *sum* using Equation (3.1), only one α ($\alpha_1(1, 1)$) out of three has been updated. Therefore, only those updated delta value of message α is enough to calculate the correct message *sum*.

$$sum'(n) = sum(n) + \sum_{m \in D(n)} \Delta\alpha(m, n) \tag{3.3}$$

In Equation (3.3), $D(n)$ are those row numbers of which the message α has been updated since the last computation of $\alpha(m, n)$ in column n. The total number of possible addition is reduced from $|B(n)|$ to $|D(n)|$.

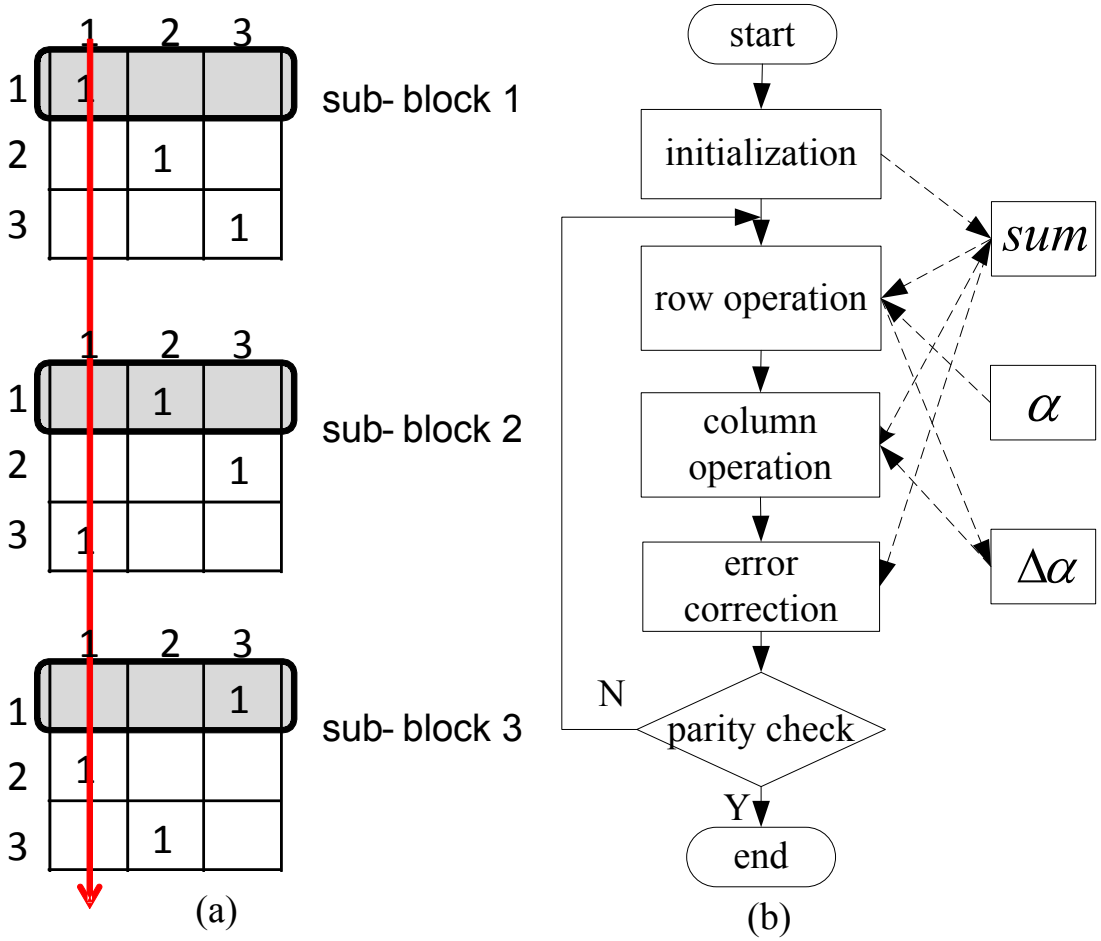


Figure 3.1 Sum-delta message passing schedule: (a) parity check matrix of motivation example, and (b) flowchart of proposed sum-delta message passing schedule

The proposed SDMP schedule, which is shown in Figure 3.1(b), takes fully advantage of the improvement compared to DVMP schedule. In the initialization step, message *sum* is initialized as message λ. Then the row operation is executed to update message α according to Equation (1.6), where β value is obtained through Equation (3.2) using message *sum* and α, and corresponding $\Delta\alpha$ is generated at the meantime. After that, the column operation calculates message *sum* by the updated $\Delta\alpha$ values as Equation (3.3). The error correction calculates the tentative decision, based on which a parity check is done after each iteration to determine the termination of the decoding process.

Back to the example in Figure 3.1(a), for the computation of β value in this example using SDMP schedule, we update *sum* value using delta-value of message α first and each β value can be calculated using Equation (3.2) respectively.

$$\begin{aligned} sum'(1) &= sum(1) + \Delta\alpha_1(1,1) \\ \beta_1'(1,1) &= sum'(1) - \alpha_1(1,1) \\ \beta_2'(3,1) &= sum'(1) - \alpha_2(3,1) \\ \beta_3'(2,1) &= sum'(1) - \alpha_3(2,1) \end{aligned}$$

In this example, the addition depth only decreases from two to one, this is because the parity check matrix is so small that the advantage of SDMP schedule is not obvious. But as for the real implementation of the targeted matrix in 802.11n, the addition depth is decreased significantly using the SDMP schedule. By comparing the proposed algorithm and the accelerated algorithm, we can see that the benefit of this SDMP schedule is twofold. First, this calculation method of newly introduced message *sum* based on updated message $\Delta\alpha$ only, can remove the redundant computations and reduce the total number of addition depth in the column operation, especially when $|B(n)|$ becomes larger. For the targeted parity check matrix, at most two messages α are updated after certain row operation ($|D(n)| = 2$). This will save the computational depth from 5 levels in the accelerated MP scheme to at most 2 levels, which improve decoding throughput. On the other hand, by adapting this novel

decoding scheme, only one message *sum* rather than at most 12 message β of each column is required for storage, which helps reduce area cost. The detailed area saving strategies are discussed in section 3.4.

3.3 High throughput design

In this section, strategies other than SDMP schedule to achieve high throughput design in described.

3.3.1 SDMP based column operation module

As discussed in the previous section, the SDMP-based column operation can update message *sum* by addition of its original value and at most two updated $\Delta\alpha$ values using Equation (3.3). The number of updated $\Delta\alpha$ values and the exact position of each updated $\Delta\alpha$ in the column operation are determined by the parity check matrix. By adapting the SDMP schedule, not only the computation time of each column operation is minimized, but also the area of hardware implementation is reduced. Based on this observation, we can further improve the throughput by applying a parallel column operation here.

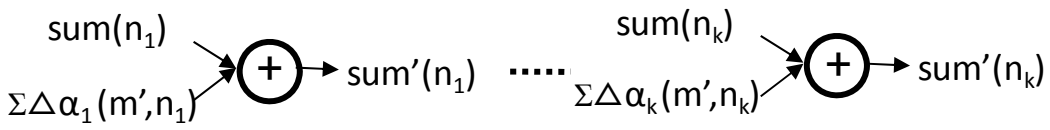


Figure 3.2 Architecture of proposed SDMP-based column operation module

In previous implementation of the accelerated MP schedule [13], [14], all N column-blocks are computed in parallel while different columns in one column-block are calculated in serial, which may require at most 11 clock cycles for column computation. In the proposed module, message *sum* of all columns, in which there exists an updated α, are computed in parallel. A hardware implementation for computing k columns in parallel is shown in Figure 3.2. The resulting parallel

SDMP-based column operation can be computed in one clock cycle.

3.3.2 Pipeline schedule

In the proposed decoder, a pipeline structure is utilized to achieve further speed-up of the procedure, as shown in Figure 3.3. Compared to the pipeline schedule in Chapter 2, this design needs one more clock cycle to calculate β from *sum* and α. The row operation and column operation are divided into five and three pipeline stages respectively to balance the computation time of each stage.

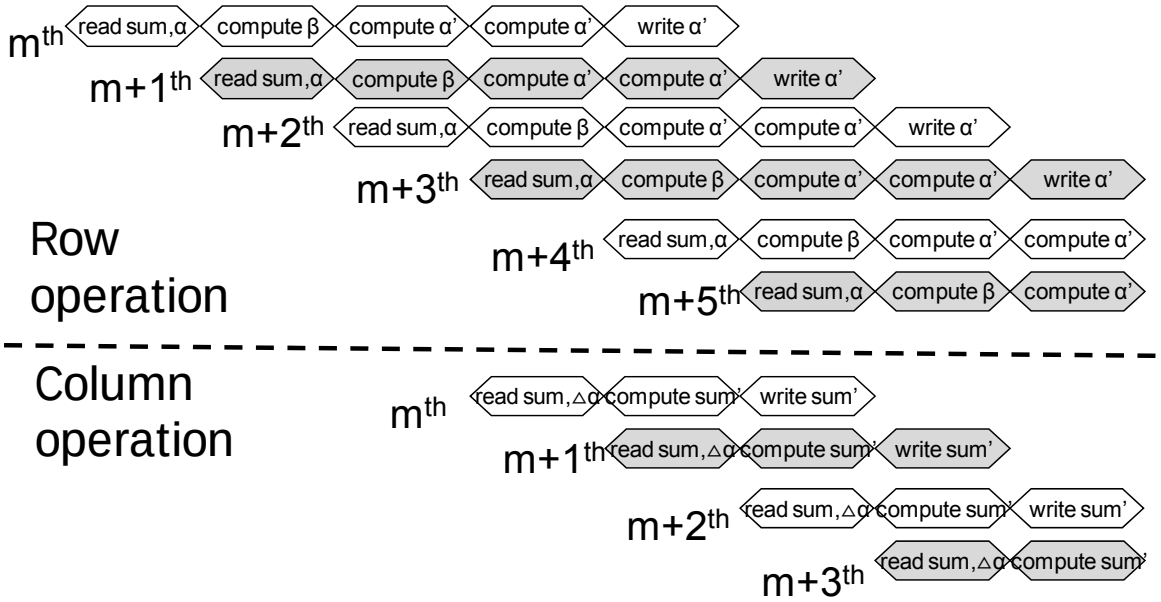


Figure 3.3 Timing schedule of the decoder applying SDMP schedule

During the first clock cycle, α and *sum* values are read in to prepare for a row operation. In the second clock cycle, β value is obtained from α and *sum* values. In the third and fourth clock cycles, a tree structure comparison is conducted for the minimum and second minimum β value, and corresponding α and $\Delta\alpha$ values

are obtained. In the fifth clock cycle, all related column operations are computed based on *sum* and $\Delta\alpha$ values, and α values are written into memories at meantime. Then the updated α and *sum* messages will be read in and used by $(m + 5)^{th}$ row operation. Based on this proposed pipeline structure, the BER performance comparison with ref. [14] and the proposed decoder in Chapter 2 is illustrated in Figure 3.4 under the same iteration (itr=5). Method 1 represents the design in Chapter 2 using DVMP schedule and method 2 is the design proposed in this chapter. This pipeline structure incurs nearly no performance degrading compared to [14]. The reason of the gap of the BER performance of method 2 and ref. [14] is that the updated message of m^{th} row can be used at $(m+5)^{th}$ row other than $(m+3)^{th}$ row in [14], and the sooner the updated messages are used the better the performance will be. Besides that, the proposed pipeline structure is more compact, which improves both the hardware utilization and throughput. The number of clock cycles for each iteration decreased from 256 to 32.

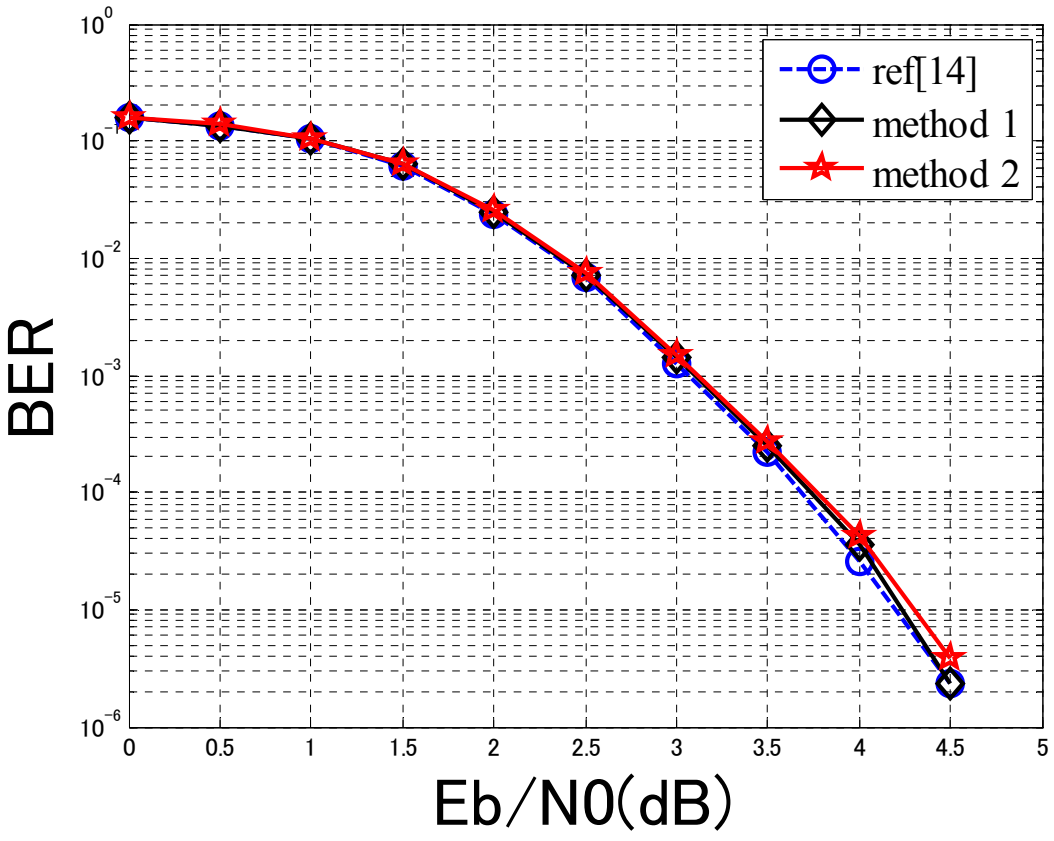


Figure 3.4 Bit error rate performance of the decoder applying SDMP schedule

3.4 Low area design

By adapting the novel SDMP schedule described in Section 3.2, we can further reduce the memory allocation and save chip area. Let us revisit the motivation example shown in Figure 3.1(a). Since β values $\beta_1(1, 1)$, $\beta_2(3, 1)$, $\beta_3(2, 1)$ is obtained and used only in row operation, the storage of message β is no longer required. Instead, only one message *sum*(1) for the first column is stored for further computation. In columns with a weight of 12 in the targeted parity check matrix ($|B(n)| = 12$), this strategy will save the area by 11/12 at the best case.

The implementation result demonstrates that using the storage for *sum* instead of β in SDMP schedule can save 82% ((1,949,487-347,036)/1,949,487) of the storage area, as listed in Table 3.1.

Furthermore, in the proposed SDMP schedule, α value is only required in row operation for the calculation of $\Delta\alpha$, while in former MP scheme, message α is accessed at both row and column operation. In other words, in former MP scheme, message α should be accessible by both row and column address, while in SDMP, a row of message α can be stored in a single memory line, thus saving control logic and area. Additionally, in SDMP schedule, λ is no longer used in each column operation, which further saves the storage area. The detailed saving for message α and λ are also demonstrated in Table 3.1. As we can see from the table, the proposed method can save a total 60% storage compared to [14].

Table 3.1 Comparison of storage area (μm^2)for messages in TSMC 0.18μm

	Ref. [14]	Proposed	**Total saving**
β	1,949,487	0	
Sum	0	347,036	
α	1,949,487	1,695,501	
Λ	1,301,845	0	
$\Delta\alpha$	0	40,395	
Total	**5,200,819**	**2,082,932**	**3,117,887**

3.5 Implementation and result

In this section, we present the hardware implementation of the proposed partially-parallel irregular LDPC decoder, using TSMC 0.18μm CMOS technology and the synthesis result of the proposed decoder.

3.5.1 Implementation details

The proposed decoder is mainly composed of five parts: row operation, column operation, parity check, controller and storage parts (memory for α value, registers for *sum* value, $\Delta\alpha$ value), as shown in Figure 3.5.

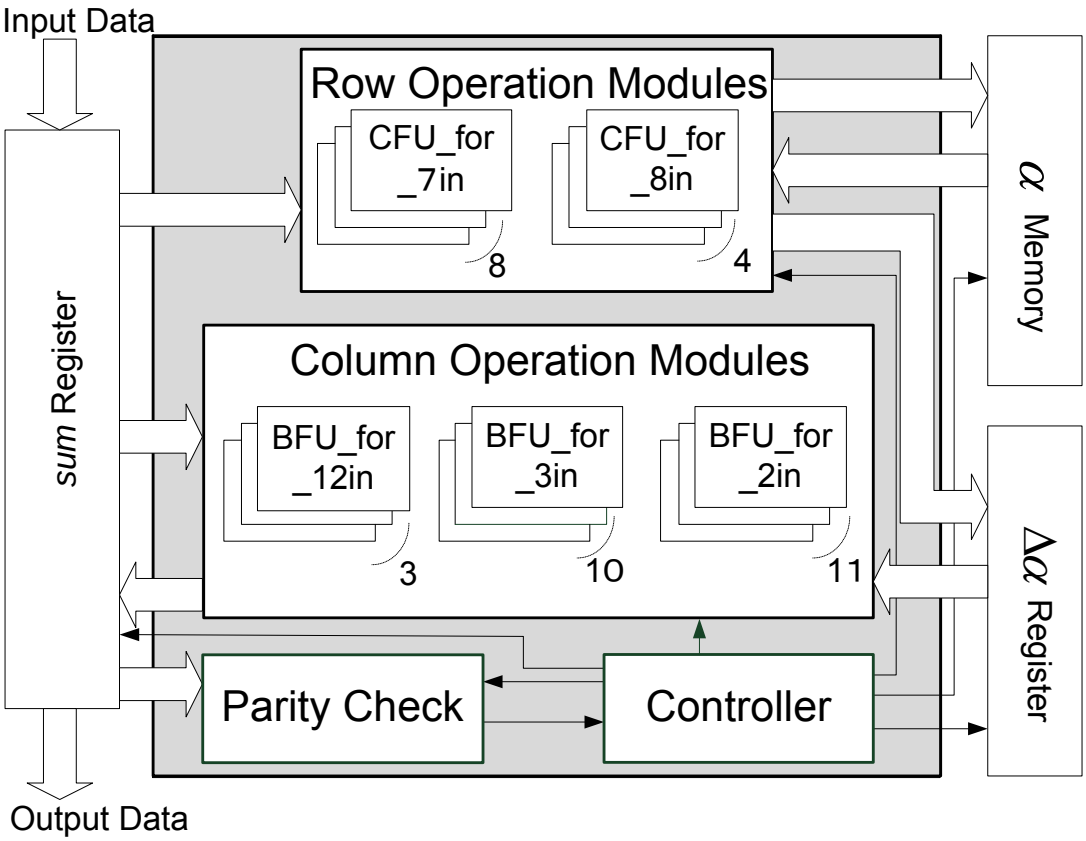


Figure 3.5 Block diagram of the decoder applying SDMP schedule

In the proposed decoder, we design 12 CFUs and 24 BFUs in order to execute the computation of the same row or column in each sub-block in parallel. Since the targeted LDPC decoding matrix is irregular, two different row operation modules (CFU_for_7in and CFU_for_8in), and three different column operation modules (BFU_for_12in, BFU_for_3in, and BFU_for_2in) are designed for rows and columns with different $|A(m)|$ and $|B(n)|$. The controller module generates the control signals for storage and operation modules, while parity check module does the error correction and the parity check at the end of each iteration. Message α is only used in the row operation and stored in memory with corresponding row address. Other messages like *sum*, which have multiple-access problem, are stored in registers.

The detailed composition of the proposed decoder core is listed in Table 3.2. As we can see from the table, the former computation-intensive column operation modules are greatly reduced because of the use of SDMP schedule.

Table 3.2 Composition of the decoder core implementing SDMP schedule

Module	Number of Gates	Percentage (%)
Row Operation Modules	81,813	82.48
Column Operation Modules	6,669	6.73
Parity Check	9,258	9.36
Controller	1,419	1.43
LDPC decoder core	99,186	100

3.5.2 Synthesis result

The synthesis results of the proposed decoder are listed in Table 3.3. We compare our work with typical decoder designs in [9], [10] under the different specifications, and with [13], [14] under the same LDPC code and the same design rule. The design in [8] is an irregular partially-parallel decoder targeting a different LDPC code while the design in [9] is a fully-parallel regular decoder. Under column *Ref. [14]* is the synthesis result of a partially-parallel irregular LDPC decoder designed in [14], which is the only design known, targeting the same newly proposed LDPC code in [17]. We also modified the design in [13] to support the same irregular code with detailed synthesis result under column *Ref. [13]**. Ref. [25] is the design targeting different irregular LDPC code in 802.11n. The decoder under column Method 1 is the decoder introduced in Chapter 2. The decoder under column Method 2 is the decoder introduced in this chapter.

Table 3.3 Synthesis result of the LDPC decoder implementing SDMP schedule

	Ref. [9]	Ref. [10]	Ref.[25]
Design rule	0.16μm	TI 0.11μm	TSMC 0.18μm
LDPC code	1024 bit rate 1/2 regular	8088 bit rate 1/2 irregular	802.11n 1296bit rate:1/2 irregular
LDPC decoder	Fully-parallel	Partially-parallel	Partially-parallel
Throughput	1Gbps✕1/2 (itr=54, SNR = 3.0dB)	188Mbps (itr=25, SNR=NA)	1Gbps (Itr= 5)
Frequency	64MHz	212NHz	200MHz
Memory area	N/A	407Kgates	No use
Area w/o wiring	1750Kgates	742Kgates	520Kgates
Area w/. wiring	N/A	N/A	23,528,130μm^2
Power(mW)	690 (@64MHz, 1.5V)	N/A	755
Chip area(mm^2)	N/A	N/A	N/A

Table 3.4 Synthesis result of the LDPC decoder implementing SDMP schedule (continued)

	Ref. [13]*	Ref. [14]	**Method 1**	**Method 2**
Design rule	TSMC 0.18μm			
LDPC code	802.11n 648 bit rate 1/2 irregular			
LDPC decoder	Partially-parallel			
Throughput	54Mbps (itr= 5, SNR = 3.0)	54Mbps (itr= 5, SNR = 3.0)	**418Mbps (itr= 5, SNR = 3.0)**	**404Mbps (itr= 5, SNR = 3.0)**
Frequency	200MHz	200MHz	**200MHz**	**200MHz**
Memory area	708Kgates	502Kgates	**170Kgates**	**170Kgates**
Area w/o wiring	832Kgates	611Kgates	**423Kgates**	**313Kgates**
Area w/. wiring	13,090,549μm^2	9,004,366μm^2	**12,930,433μm^2**	**8,012,999μm^2**
Power(mW)	765.85	486.44	**893.18**	**712.38**
Chip area(mm^2)	N/A	N/A	**N/A**	**13.69 (layout)**

Because of the novel SDMP schedule along with the improved row and column operation modules, our proposed decoder requires only 32 clock cycles for a single iteration and five iterations for codeword correction under Signal-to-Noise Ratio (SNR) of 3.0dB. In contrast to the partially-parallel decoders in [10], [13], [14], the proposed decoder achieves much highest throughput for partially-parallel irregular LDPC decoder. Because of the optimization of storage, the proposed decoder achieves a gate count reduction of 39% and 11% compared with [13], [14] respectively. Although it can not achieve as high as the throughput compared with [25], the proposed decoder consumes less power and area. It also shows the advantage in the area and power when it is compared to the design introduced in Chapter 2.

When it is compared to the fully-parallel decoder in [9], although it consumes a

little more power than [9], the proposed decoder requires only about 1/5 the gate count of that of [9].

The backend design of the proposed decoder is implemented by Synopsys Astro with ARM's Artisan SAGE-X 0.18μm 1P6M stand-cell library for TSMC. The layout design includes the design setup, floor planning, timing setup, placement, CTS and routing. The layout is provided in Figure 3.6, where modules are labeled in the figure and memories on the boundaries are for message α. The size of the chip core after layout is 13.69mm2(3.7mm × 3.7mm).

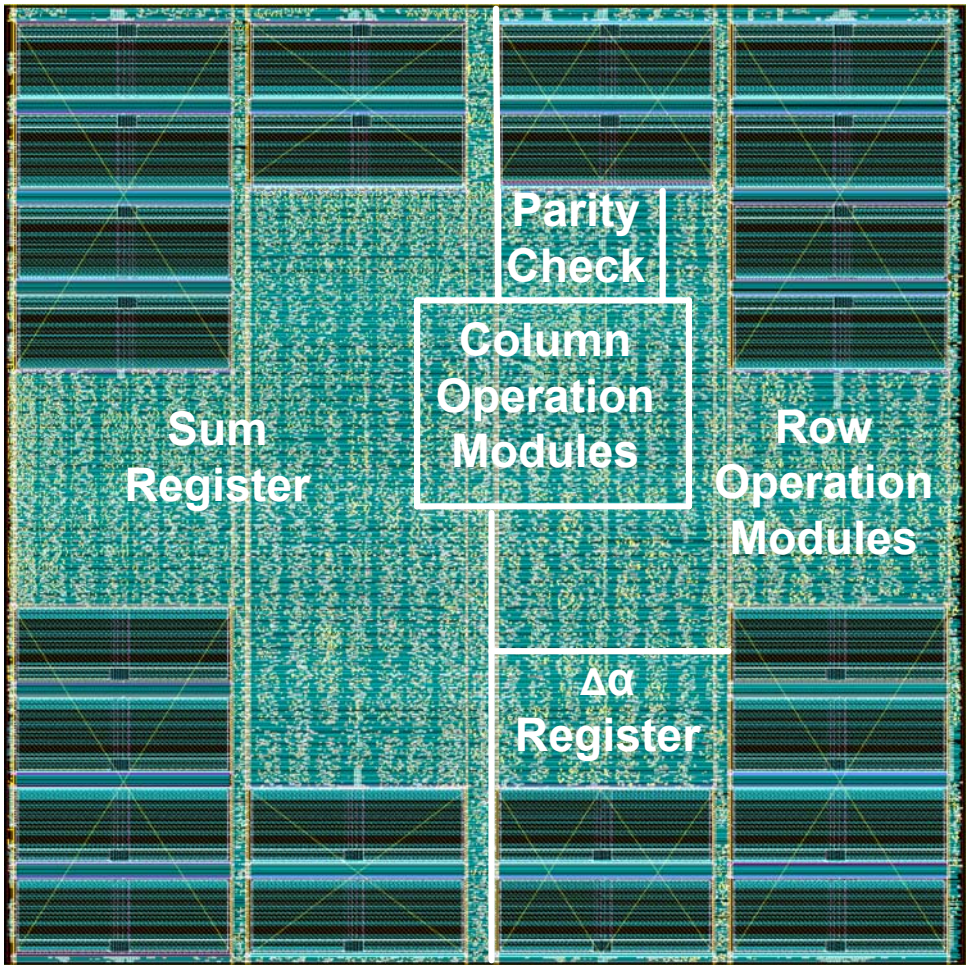


Figure 3.6 Layout of the decoder

3.6 Conclusion

In this chapter, a novel high throughput low area cost partially-parallel irregular LDPC decoder is proposed based on the one proposed in Chapter 2. Row and column operations are speeded up by a modified binary searching scheme and SDMP schedule respectively. Area cost is decreased because of the use of the proposed SDMP schedule. The synthesis result demonstrates that our decoder can achieve a much higher throughput and almost the same bit error rate performance with less area cost compared to other partially-parallel irregular LDPC decoders. It is also a better design in terms of tradeoff between throughput, area and power compared to fully-parallel regular LDPC decoders.

4 Self-adjustable offset min-sum algorithm targeting high BER performance and low hardware cost

4.1 Introduction

In Japan, a next generation satellite broadcasting system named "Integrated Services Digital Broadcasting via Satellite - Second Generation (ISDB-S2)" was proposed by NHK (Japan Broadcasting Corporation), and is currently under the examination of Association of Radio Industries and Businesses (ARIB) [28]. To ensure the transmission quality and high error correction capability, LDPC code is selected as the error correction code for ISDB-S2 and is expected to achieve a Bit Error Rate (BER) of 10^{-11}.

Defined by ISDB-S2, the parity check matrices targeted in this work are 11 different codes with code rate ranging from 1/4 to 9/10. The numbers of columns (N) for all 11 codes are fixed as 44,880 and the numbers of rows (M) are related to the code rate. And all the codes are structured LDPC codes, with a sub-block size of 374 $\times$ 374.

LDPC code can be efficiently decoded through messages exchange between check nodes and bit nodes by performing check node and bit node operations iteratively. Among decoding algorithms, Belief Propagation (BP) algorithm, also known as Sum Product algorithm, is well known for its good error correcting performance. However it is not hardware-friendly due to the necessity of implementing Hyperbolic functions

[1]. Min-sum (MS) algorithm approximates BP algorithm with easy hardware implementation but greatly degrades the error correcting performance [18]. Recently, many approaches have been proposed to trade off between the BER performance and hardware complexity. These approaches can be categorized as two kinds of schemes: MS-based schemes and BP-based schemes. The MS-based schemes aim at improving the error correcting performance of the MS algorithm by introducing a multiplied or additive factor, *i.e.*, Normalized Min-sum (NMS) algorithm and Offset Min-sum (OMS) algorithm [19]. Later on, some further derivatives of OMS algorithm appear, such as the Degree-Matched Min-sum (DMMS) algorithm [29] which associates the offset with the degree of the check node, and the Adaptive Offset Min-sum (AOMS) algorithm [30] which adapts the offset according to the most unreliable information sent from the bit nodes. BP-based schemes, on the other hand, approximate the BP algorithm by calculating the Hyperbolic function term by term using Jacobian logarithm, such as Modified Min-sum (MMS) algorithm and Delta Min (DM) algorithm[31][32].

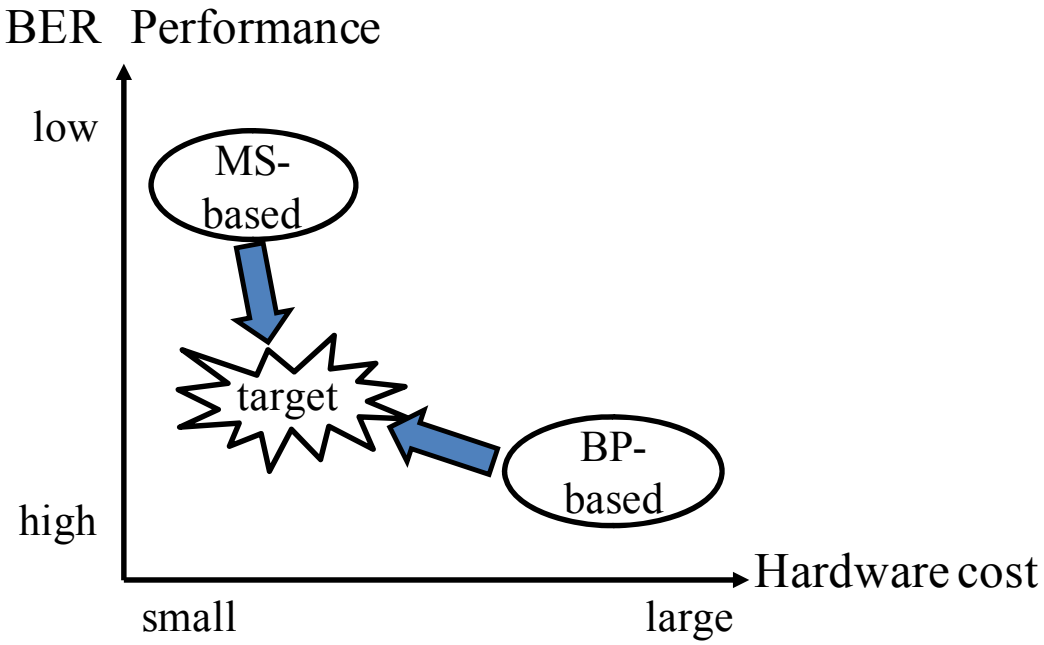


Figure 4.1 Requirement for ISDB-S2 LDPC decoder

Generally, BP-based algorithms outperform MS-based algorithms in BER

performance, but they require larger hardware cost due to the iterative term-based implementation, as shown in Figure 4.1. Specifically, for the LDPC codes in ISDB-S2, a maximum of 90 times of computation complexity is introduced compared to MS algorithm, which directly increases the hardware overhead and power consumption. As far as the high BER performance requirement of practical ISDB-S2 application is concerned, MS-based algorithms are not competent enough. On the other hand, the hardware and power overhead of BP-based algorithms also limit their practical usage for the highly parallel implementation of ISDB-S2 LDPC decoder. Therefore, a decoding scheme which can achieve a similar BER performance as BP-based algorithms while maintaining the low hardware cost, will become the trend of future LDPC decoder design for next generation satellite applications.

Motivated by this challenging design task, we proposed a hybrid decoding scheme as an initial attempt for both high BER performance and low hardware cost design. The algorithm improves the OMS algorithm by a uniform approximation to the check node computation while the approximation is derived through mathematical induction on Jacobian logarithm, adopted widely by BP-based algorithms. It utilizes a self-adjustable offset based on the difference of the two most unreliable input values from the bit nodes. The simulation results further demonstrate that the proposed method can not only improve the BER performance compared to the MS-based schemes with nearly no overhead in hardware cost, but also consumes far less hardware than the BP-based schemes.

4.2 MS-based approximation and BP-based approximation

Let λn denote the Log-Likelihood Ratios (LLR) of the bit node n of the received codeword from the channel, α_{mn} be the message sent from check node m to bit node n, β_{mn} be the message sent from bit node n to check node m, and sum_n be the A Posteriori

Probability (APP) message of the bit node n of the codeword. The check node operation, bit node operation and APP update operation of BP algorithm can be expressed as Equation (4.1), Equation (4.2) and Equation (4.3), respectively. Note that $A(m)$ and $B(n)$ are defined as $A(m) = \{n|Hmn = 1\}$ and $B(n) = \{m|Hmn = 1\}$.

$$\alpha_{mn} = 2\tanh^{-1}\left(\prod_{n'\in A(m)\backslash n}\tanh\left(\frac{\beta_{mn'}}{2}\right)\right) \tag{4.1}$$

$$\beta_{mn} = \lambda_n + \sum_{m'\in B(n)\backslash m}\alpha_{m'n} \tag{4.2}$$

$$sum_n = \lambda_n + \sum_{m'\in B(n)}\alpha_{m'n} \tag{4.3}$$

Since the check node function of BP algorithm is not hardware friendly, varies researches have been done to approximate the BP algorithm for better hardware implementation.

4.2.1 MS-based approximation

A simple approximation to Equation (4.1) is called Min-Sum algorithm which uses the minimum magnitude of input β as a replacement of the Hyperbolic functions, as shown in Equation (4.4).

$$\alpha_{mn} = \min_{n'\in A(m)\backslash n}|\beta_{mn'}| \tag{4.4}$$

Although MS algorithm can be easily implemented in hardware, it suffers a large performance degrading which encourages further researches to find better approximation based on the MS algorithm. For instance, a normalization factor or offset factor is applied to the MS algorithm, which forms the well-known Normalized MS algorithm and Offset MS algorithm, as shown in Equation (4.5) and (4.6) [19].

$$\alpha_{mn} = \gamma\prod_{n'\in A(m)\backslash n} sign(\beta_{mn'})\times\min_{n'\in A(m)\backslash n}|\beta_{mn'}| \tag{4.5}$$

$$\alpha_{mn} = \prod_{n'\in A(m)\backslash n} sign(\beta_{mn'})\times\max\left(\min_{n'\in A(m)\backslash n}|\beta_{mn'}| - \varepsilon, 0\right) \tag{4.6}$$

Note that the normalization factor γ and offset factor ε is not subject to change during the decoding procedure. Some recent progress claims that techniques to adjust the offset factor according to either the degree of the check node (DMMS algorithm [29]) or the minimum output data from the check node (AOMS algorithm[30]) can achieve better performance. However, DMMS requires significant computation power to determine the offset factor while the AOMS lacks sufficient theoretical evidence to support its approximation.

4.2.2 BP-based approximation

We first denote a basic computation in the check node operation of BP algorithm (Equation (4.2)) as function ⊗:

$$\alpha = 2\tanh^{-1}(\tanh(\frac{\beta_1}{2})\tanh(\frac{\beta_2}{2})) = \beta_1 \otimes \beta_2 \tag{4.7}$$

Therefore, Equation (4.2) can be simplified as Equation (4.8).

$$\begin{aligned}\alpha_n &= 2\tanh^{-1}(\tanh(\frac{\beta_1}{2})\tanh(\frac{\beta_2}{2})...\tanh(\frac{\beta_n}{2})) \\ &= \underbrace{\beta_1 \otimes \beta_2 \otimes ... \otimes \beta_n}_{|A(m)\backslash n|}\end{aligned} \tag{4.8}$$

Equation (4.7), the primitive form of Equation (4.8), can be expanded using Jacobian Logarithm ($\ln(e^a+e^b)=\max(a,b)+\ln(1+e^{-|a-b|})$) twice as follows[31]:

$$\begin{aligned}\alpha &= 2\tanh^{-1}(\tanh(\frac{\beta_1}{2})\tanh(\frac{\beta_2}{2})) \\ &= \ln\frac{1+e^{\beta 1+\beta 2}}{e^{\beta 1}+e^{\beta 2}} \\ &= \ln(1+e^{\beta 1+\beta 2}) - \ln(e^{\beta 1}+e^{\beta 2}) \\ &= \max(0,\beta 1+\beta 2) + \ln(1+e^{-|\beta 1+\beta 2|}) - \max(\beta 1,\beta 2) - \ln(1+e^{-|\beta 1-\beta 2|}) \\ &= sign(\beta 1)sign(\beta 2)\cdot(\min(|\beta 1|,|\beta 2|) + \ln(1+e^{-||\beta 1|+|\beta 2||}) - \ln(1+e^{-||\beta 1|-|\beta 2||}) \\ &= sign(\beta 1)sign(\beta 2)\cdot(\min(|\beta 1|,|\beta 2|) + f(|\beta 1|+|\beta 2|) - f(|\beta 1|-|\beta 2|))\end{aligned} \tag{4.9}$$

where function f(x) is defined as $f(x)=\ln(1+e^{-|x|})$ as shown in Figure 4.2.

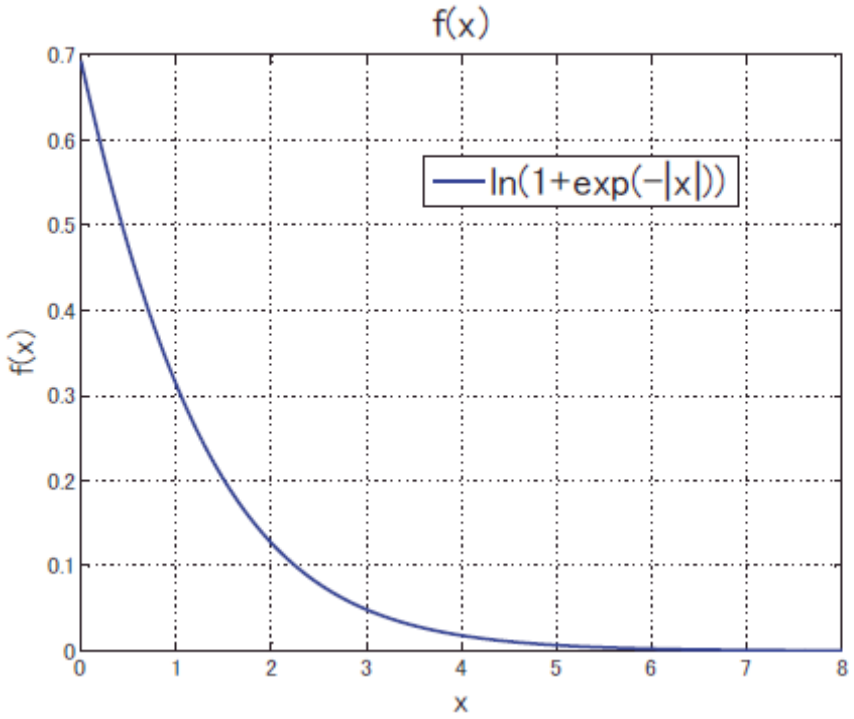


Figure 4.2 f(x)=ln(1+e$^{-|x|}$)

Since f(x) is not hardware friendly, several works focus on the approximation of Equation (4.9). An MMS algorithm is proposed in [31] with Equation (4.10) as a substitution of Equation (4.9). Similarly, a DM algorithm is proposed in [32] using Equation (4.11) to calculate the parameter D in Equation (4.10).

$$\beta 1 \otimes \beta 2 = sign(\beta 1)sign(\beta 2) \cdot (\max(\min(|\beta 1|,|\beta 2|) - D,0))$$

$$\text{where } D = \begin{cases} 0.5 & |\beta 1 + \beta 2| \le 1 \,\&\, |\beta 1 - \beta 2| > 1 \\ -0.5 & |\beta 1 + \beta 2| \le 1 \,\&\, |\beta 1 - \beta 2| > 1 \\ 0 & else \end{cases} \tag{4.10}$$

$$D = \max((0.9 - \frac{\delta}{2}),0) where\ \delta = ||\beta 1| - |\beta 2|| \tag{4.11}$$

Equation (4.10) and Equation (4.11) are then applied iteratively for the check node operation (Equation 2). Figure 4.2 demonstrates this iterative computation

process for message α_{m1}. In each iteration, the $\otimes$ function of the intermediate result and a β message is calculated. Therefore, for each α value, a total of (|A(m)|-2) $\otimes$ computations are required. Since altogether there are |A(m)| α values to be calculated in one row, the computation complexity of the check node operation is proportional to |A(m)|×(|A(m)|-2), which is relatively large for some codes in ISDB-S2.

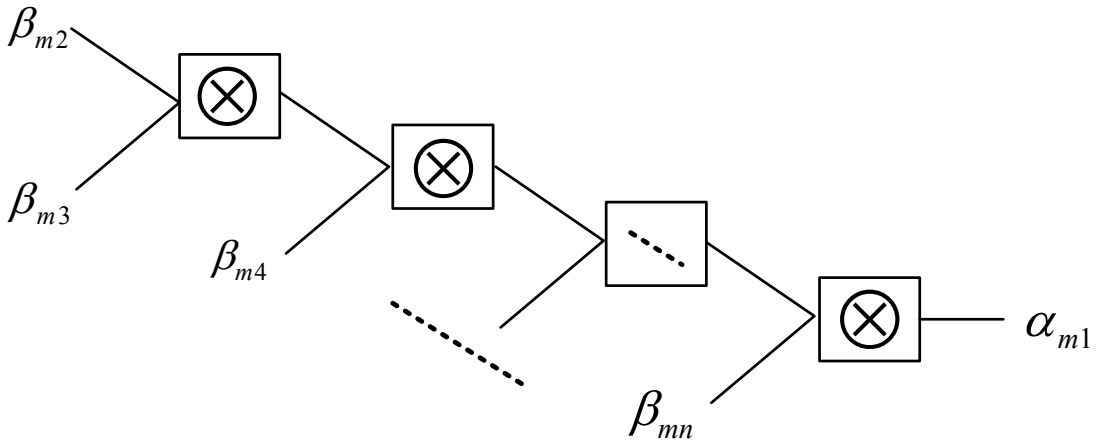


Figure 4.3 Iterative calculation for row operation using BP-based scheme

4.3 Proposed self-adjustable offset min-sum algorithm

In this section, a novel self-adjustable offset min-sum algorithm is proposed, in which a uniform approximation for the check node operation of the BP algorithm is developed through mathematical induction on Jacobian logarithm. The effectiveness of the proposed approximation is demonstrated by the simulation results of all the 11 parity check matrices in ISDB-S2, showing a better BER performance than MS-based schemes. The computation complexity and area cost are also analyzed to further exhibit that the proposed algorithm has much smaller hardware cost than the BP-based schemes.

4.3.1 Proposed approximation of BP algorithm

In order to reduce the computation complexity of check node operation, we first consider a general case as shown in Equation (4.12). Note that the general case is targeted here by considering $n' \in A(m)$ rather than $n' \in A(m) \backslash n$ in Equation (4.2). The exact calculation of α_{mn} will be explained after the uniform approximation is derived.

$$2\tanh^{-1}\left(\prod_{n'\in A(m)}\tanh\left(\frac{\beta_{mn'}}{2}\right)\right) = \underbrace{\beta_{m1}\otimes\beta_{m2}\otimes\cdots\beta_{mn'}}_{n'\in A(m)} \tag{4.12}$$

Since function $\otimes$ holds commutative law, we can fairly assume that $|\beta m1| < |\beta m2| < \ldots < |\beta mn'|$. Under this assumption, Equation (4.12) can be further expanded as Equation (4.13) through a mathematical induction based on Equation (4.9).

$$\begin{aligned}
&2\tanh^{-1}\left(\tanh\left(\frac{\beta_{m1}}{2}\right)\cdots\tanh\left(\frac{\beta_{mn'}}{2}\right)\right)\\
&\cong sign(\beta_{m1})sign(\beta_{m2})\cdots sign(\beta_{mn'})(\min(|\beta_{m1}|,|\beta_{m2}|,\cdots,|\beta_{mn'}|)\\
&-f(|\beta_{m2}|-|\beta_{m1}|)-f(|\beta_{m3}|-|\beta_{m1}|)-\cdots-f(|\beta_{mn'}|-|\beta_{m1}|)\\
&+f(|\beta_{m2}|+|\beta_{m1}|)+f(|\beta_{m3}|+|\beta_{m1}|)+\cdots+f(|\beta_{mn'}|+|\beta_{m1}|))
\end{aligned} \tag{4.13}$$

The detailed proof of Equation (4.13) is listed below.

---

(1)The condition of $n'= 2$ is already proved in Section 4.2.2

(2)Suppose $n'= k$ is correct, consider the situation of $n'=k+1$

if

$$\begin{aligned}
&2\tanh^{-1}\left(\tanh\left(\frac{\beta_{m1}}{2}\right)\cdots\tanh\left(\frac{\beta_{mk}}{2}\right)\right)\\
&\cong sign(\beta_{m1})\cdots sign(\beta_{mk})(\min(|\beta_{m1}|,\cdots,|\beta_{mk}|)\\
&\quad -f(|\beta_{m2}|-|\beta_{m1}|)-\cdots-f(|\beta_{mk}|-|\beta_{m1}|))\\
&\quad +f(|\beta_{m2}|+|\beta_{m1}|)+\cdots+f(|\beta_{mk}|+|\beta_{m1}|))
\end{aligned}$$

is true

$$2\tanh^{-1}(\tanh(\frac{\beta_{m1}}{2})\cdots\tanh(\frac{\beta_{mk}}{2})\tanh(\frac{\beta_{m(k+1)}}{2}))$$
$$\cong sign(\beta_{m1})\cdots sign(\beta_{mk})sign(\beta_{m(k+1)})(\min(|\beta_{1m1}|,\cdots,|\beta_{mk}|,|\beta_{m(k+1)})|)$$
$$-f(|\beta_{m2}|-|\beta_{m1}|)-\cdots-f(|\beta_{mk}|-|\beta_{m1}|)-f(|\beta_{m(k+1)}|-|\beta_{m1}|))$$
$$+f(|\beta_{m2}|+|\beta_{m1}|)+\cdots+f(|\beta_{mk}|+|\beta_{1m}|)+f(|\beta_{m(k+1)}|+|\beta_{m1}|))$$

is also true

proof:

$$\tanh(\frac{\beta_{m1}}{2})\cdots\tanh(\frac{\beta_{mk}}{2})=\mathrm{X}$$
$$2\tanh^{-1}(\tanh(\frac{\beta_{m1}}{2})\cdots\tanh(\frac{\beta_{mk}}{2}))=\mathrm{Y}$$
$$\Rightarrow \mathrm{X}=\tanh\frac{\mathrm{Y}}{2}$$

$$2\tanh^{-1}(\tanh(\frac{\beta_{m1}}{2})\cdots\tanh(\frac{\beta_{mk}}{2})\tanh(\frac{\beta_{m(k+1)}}{2}))$$
$$=2\tanh^{-1}(\mathrm{X}\cdot\tanh(\frac{\beta_{m(k+1)}}{2}))$$
$$=2\tanh^{-1}(\tanh(\frac{\mathrm{Y}}{2})\tanh(\frac{\beta_{m(k+1)}}{2}))$$
$$=sign(\mathrm{Y})sign(\beta_{m(k+1)})(\min(|\mathrm{Y}|,|\beta_{m(k+1)}|)+f(|Y|+|\beta_{m(k+1)}|)-f(|\beta_{m(k+1)}|-|\mathrm{Y}|))$$

$$=sign(\beta_{m1})\cdots sign(\beta_{mk})sign(\beta_{m(k+1)})(\min(\min(|\beta_{m1}|,\cdots,|\beta_{mk}|)-f(|\beta_{m2}|-|\beta_{m1}|)-\cdots f(|\beta_{mk}|-|\beta_{m1}|)$$
$$+f(|\beta_{m2}|+|\beta_{m1}|)-\cdots f(|\beta_{mk}|+|\beta_{m1}|),|\beta_{m(k+1)}|)-f(|\beta_{m(k+1)}|-|\mathrm{Y}|)+f(|\beta_{m(k+1)}|+|\mathrm{Y}|))$$

$$\because|Y|\cong\min(|\beta_{m1}|,\cdots,|\beta_{mk}|)=|\beta_{m1}|$$

$$\therefore 2\tanh^{-1}(\tanh(\frac{\beta_{m1}}{2})\cdots\tanh(\frac{\beta_{mk}}{2})\tanh(\frac{\beta_{m(k+1)}}{2}))$$
$$\cong sign(\beta_{m1})\cdots sign(\beta_{mn})sign(\beta_{m(k+1)})(\min(\min(|\beta_{m1}|,\cdots,|\beta_{mk}|)-f(|\beta_{m2}|-|\beta_{m1}|)-\cdots f(|\beta_{mk}|-|\beta_{m1}|)$$
$$+f(|\beta_{m2}|+|\beta_{m1}|)+\cdots f(|\beta_{mk}|+|\beta_{m1}|),|\beta_{m(k+1)}|)$$
$$-f(|\beta_{m(k+1)}|-\min(|\beta_{m1}|,\cdots,|\beta_{mk}|))+f(|\beta_{m(k+1)}|+\min(|\beta_{1}|,\cdots,|\beta_{k}|)))$$
$$\cong sign(\beta_{m1})\cdots sign(\beta_{mk})sign(\beta_{m(k+1)})(\min(|\beta_{m1}|,\cdots,|\beta_{mk}|,|\beta_{m(k+1)}|)$$
$$-f(|\beta_{m2}|-|\beta_{m1}|)-\cdots f(|\beta_{mk}|-|\beta_{m1}|)-f(|\beta_{m(k+1)}|-|\beta_{m1}|)$$
$$+f(|\beta_{m2}|+|\beta_{m1}|)+\cdots f(|\beta_{mk}|+|\beta_{m1}|)+f(|\beta_{m(k+1)}|+|\beta_{m1}|))$$

Since $|A(m)|$ is usually a large number for ISDB-S2, the implementation of Equation (4.13) requires a large amount of hardware resources. Hence an efficient approximation to the equation to reduce hardware cost is a necessity. Based on the characteristics of function f(x), as shown in Figure 4.3, we find out that f(x) is a

monotonically decreasing function with $f(x) \fallingdotseq 0$ when $x > 2.5$. Because of the relationships among $|\beta m1|,\ldots, |\beta mn|$, we can derive that $|\beta m2| - |\beta m1|$ is the smallest one among all the arguments of $f(x)$ in the equation, thus $-f(|\beta m2| - |\beta m1|)$ becomes the dominant term of all the function $f(x)$ terms. We can easily figure out, through the above derivation, the offset term is mainly dependent on the two most unreliable inputs from the bit nodes which are denoted as $\beta min1$ and $\beta min2$ from now on. However, simply keeping the dominant term and ignoring all the other ones degrades the precision of computation. Therefore, we further approximate all the other ones by multiplying a normalization factor or adding an offset factor to the dominant term $-f(\beta min2 - \beta min1)$. In this work, we use the normalization factor γ' and obtain Equation (4.14) as an approximation to Equation (4.13).

$$\alpha_{mn} = \prod_{n' \in A(m)\backslash n} \operatorname{sgn}(\beta_{mn'}) \left(\min_{n' \in A(m)\backslash n} |\beta_{mn'}| - \gamma' f(\beta_{\min 2} - \beta_{\min 1}) \right) \quad (4.14)$$

As can be seen from Equation (4.2), the computation of αmn is based on $\beta mn'$ values with $n' \in A(m)\backslash n$. However, Equation (4.14) is derived considering the $\beta mn'$ values with $n' \in A(m)$. In the following parts, we will discuss, in three different cases, how we derive the proposed approximation of Equation (4.2) from Equation (4.14).

Case 1: $|\beta mn|$ is $|\beta min1|$, the smallest one among all absolute β values. In this case, $|\beta min1|$ should not be included in the computation of αmn. Therefore, $|\beta min2|$ and $|\beta min3|$ become the minimum value and second minimum value among all $\beta mn' (n' \in A(m)\backslash n)$. Hence,

$$\alpha_{mn} = \prod_{n' \in A(m)\backslash n} \operatorname{sgn}(\beta_{mn'}) (\min 2 - \gamma' f(|\beta_{\min 3}| - |\beta_{\min 2}|))$$

Case 2: $|\beta mn|$ is $|\beta min2|$, the second minimum value among all absolute β values. In this case, $|\beta min1|$ and $|\beta min3|$ become the minimum value and second minimum value among all $\beta mn' (n' \in A(m)\backslash n)$. Therefore,

$$\alpha_{mn} = \prod_{n' \in A(m)\backslash n} \operatorname{sgn}(\beta_{mn'}) (\min 2 - \gamma' f(|\beta_{\min 3}| - |\beta_{\min 1}|))$$

Case 3: $|\beta mn|$ is neither $|\beta min1|$ nor $|\beta min2|$. In this case $|\beta min1|$and $|\beta min2|$ are still the minimum value and second minimum value among all $\beta mn' (n' \in A(m)\backslash n)$. Therefore,

$$\alpha_{mn} = \prod_{n' \in A(m) \backslash n} \operatorname{sgn}(\beta_{mn'})(\min 2 - \gamma' f(|\beta \min 2| - |\beta \min 1|))$$

So altogether three cases should be considered to implement Equation (14), which gives rise to additional design overhead. To solve the problem, we further simplify the check node operation. Through simulation, we notice that the computation of $|\beta min3| \dashv \beta min2|$ in Case 1 can be approximated as $|\beta min2| \dashv \beta min1|$, and using $-\gamma' f(|\beta min2| \dashv \beta min1|)$ instead of $-\gamma' f(|\beta min3| \dashv \beta min1|)$ for Case 2 incurs nearly no performance degrading. Hence, we combine three cases into one uniform expression shown in Equation (4.15), which greatly reduces the hardware implementation cost.

$$\alpha_{mn} = \prod_{n' \in A(m) \backslash n} \operatorname{sgn}(\beta_{mn'})(\min_{n' \in A(m) \backslash n} |\beta_{mn'}| - \gamma' f(|\beta_{\min 2}| - |\beta_{\min 1}|)) \qquad (4.15)$$

From Equation (4.15), we can see that the offset factor is self adjustable, during the iterative decoding, according to the difference of the two most unreliable inputs from the bit nodes. Such adjustable scheme precisely models the variations of bit node messages, hence enhances the decoding efficiency.

4.4 Simulation result

Software simulation of the proposed decoding algorithm has been conducted for all 11 parity check matrices used in ISDB-S2. The QPSK modulation and AWGN channel is modeled in the simulation. A total of 10,771,200 input bits are used for simulation. The maximum number of iteration is set to 50, and the simulation program terminates when the decoded codeword is a valid one or the maximum iteration times are achieved.

Figure 4.4 and Figure 4.5 illustrate the simulation result of the BER performance of BP, NMS, OMS, DMMS, AOMS, MMS, DM and the proposed decoding algorithm for rate 3/5 and 3/4 , which will be mainly used in ISDB-S2 service. Except BP algorithm is simulated using floating values, all the intermediate messages of simulations for the other algorithms are coded in 6 bit sign-magnitude format and the APP message is realized in an 8 bit sign-magnitude format to avoid overflow. The

parameters of all algorithms are chosen to optimize both the BER performance and hardware implementation as γ =0.875 for NMS (Equation (4.5)), ε = 0.125 for OMS (Equation (4.6)), and γ '= 0.125 for the proposed method (Equation (4.15)). Also, for simple hardware implementation, we use the same Δ function $\Delta(x)$ = max(5/8 − |x|/4, 0) as [26] for approximation of function $f(x)$ for the proposed algorithm in this work. The approximation is illustrated in Figure 4.6. It can be observed from the figure that the proposed algorithm achieves an average of 0.2dB gain compared to the MS-based algorithms, and sometimes even outperforms BP-based algorithms.

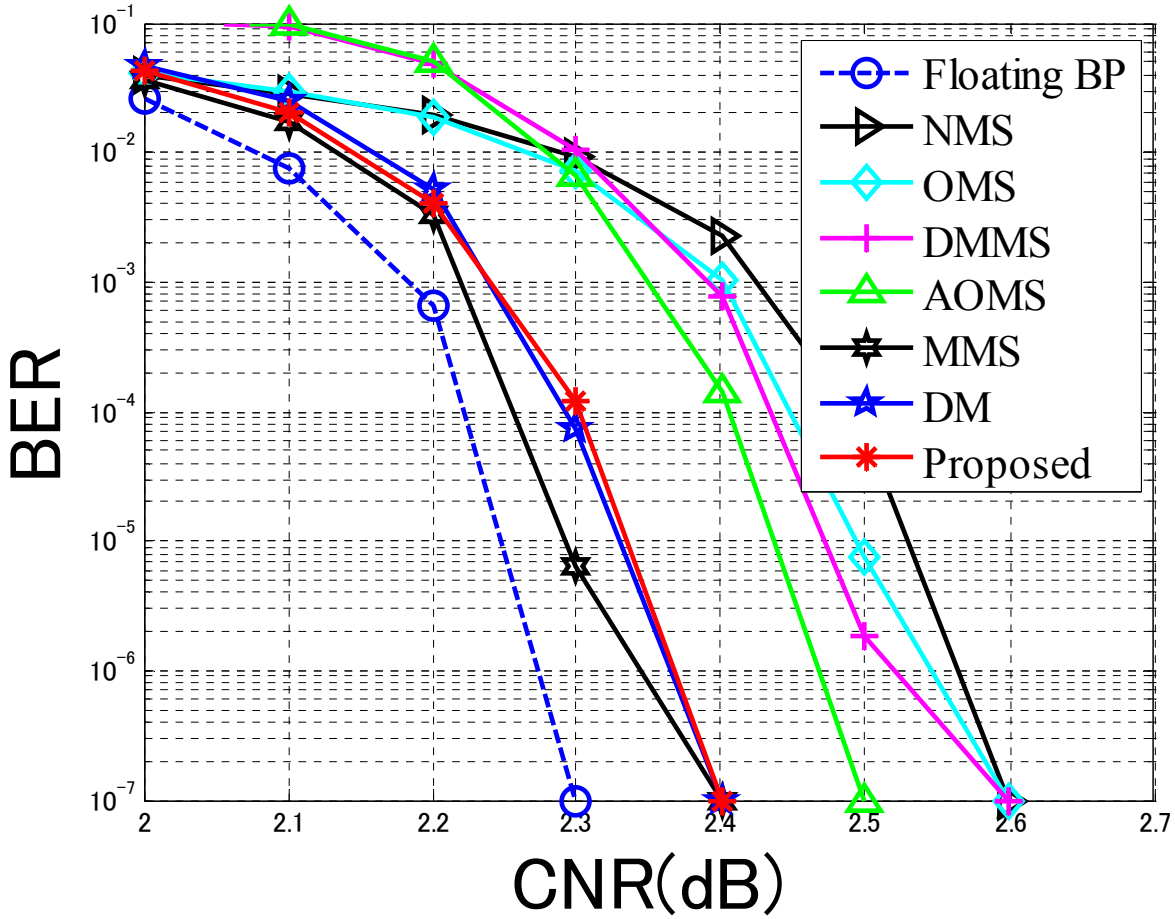


Figure 4.4 BER performance comparison for rate 3/5

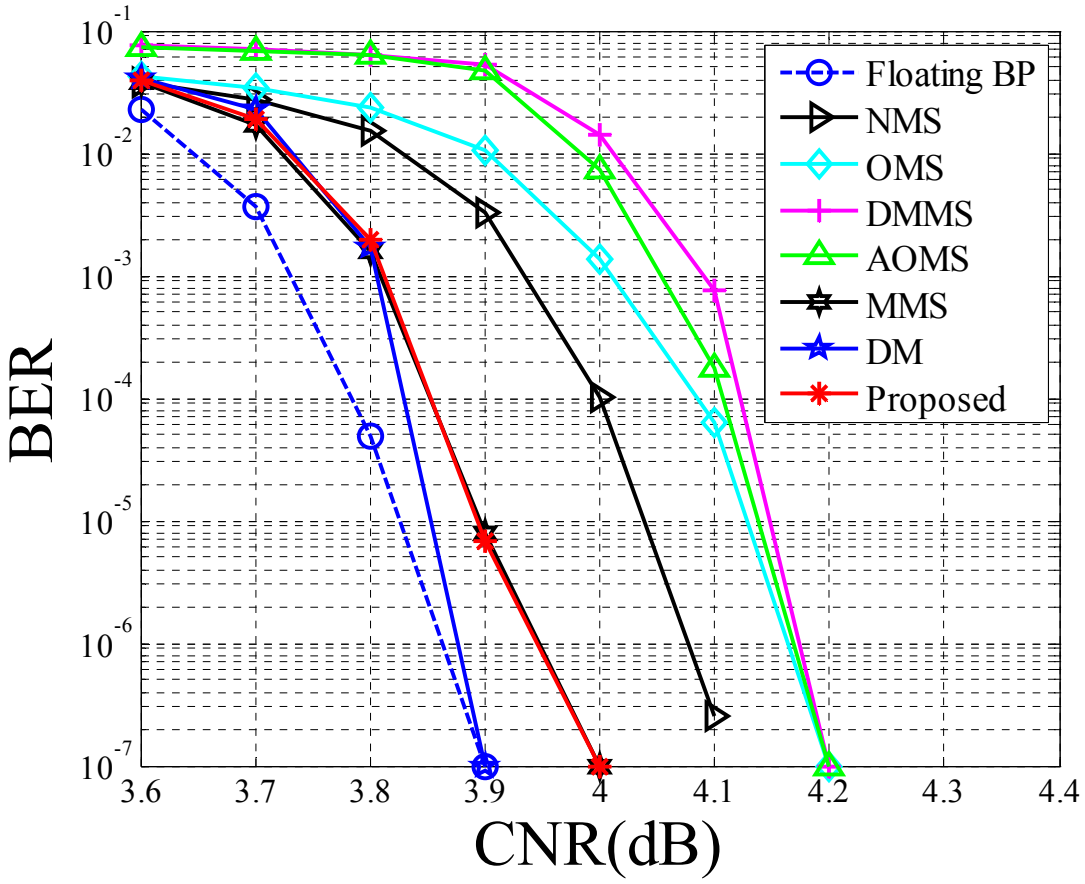


Figure 4.5 BER performance comparison for rate 3/4

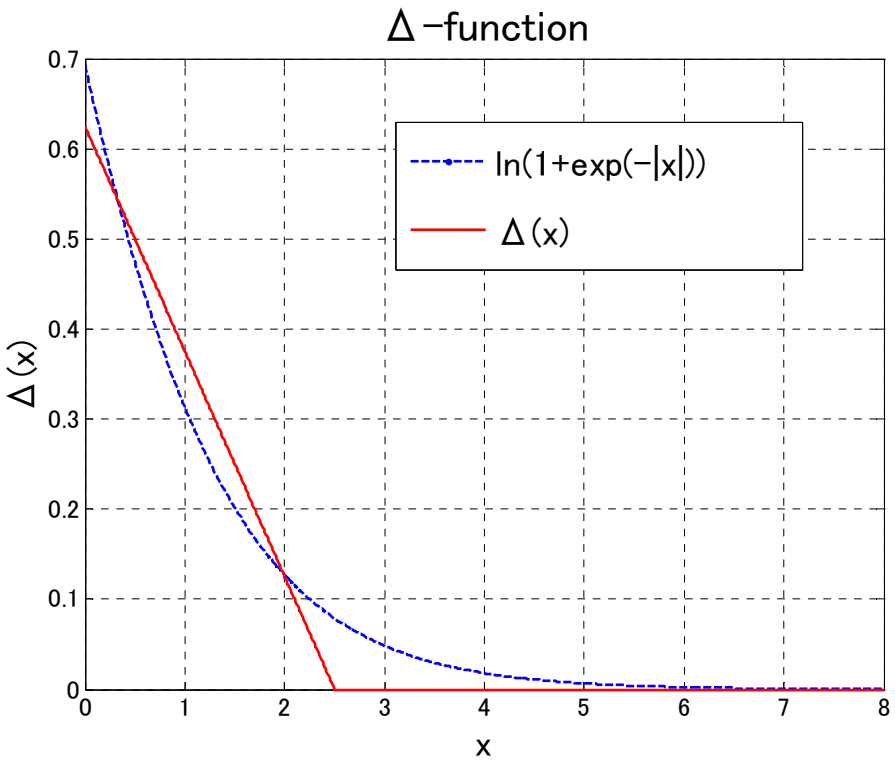


Figure 4.6 f(x) and its approximation function Δ(x)

4.5 Comparison of required CNR

In order to further analyze the efficiency of the proposed algorithm and its suitability to all the LDPC codes in ISDB-S2, we use a metric, called the required CNR. The required CNR is defined as the carrier-to-noise ratio when the BER exceeds 10^{-11} for ISDB-S2 [33]. Because of the error floor free performance of ISDB-S2 code and relatively long computer simulation time to evaluate the BER downto the range of 10^{-11}, in this work, we use the same evaluation method as [33], namely extrapolation, to calculate the required CNR. The simulation uses 10^7 input data, if no error can be found in the simulation, it is fair to say that this point is free of error at $BER = 10^{-7}$. We call this point "BER = 0 Observation Point", as shown in Figure 4.7. In this figure, P1 and P2 are simulation points obtained from the computer simulation result. P3 is the BER=0 Observation Point and P4 is the point with the required CNR (CNR4). We calculate CNR4 as shown in Equation (4.16) using the

extrapolation technique.

$$CNR4 = 2 \cdot \frac{\log(10^{-11}) - \log(BER3)}{\frac{\log(BER2) - \log(BER1)}{CNR2 - CNR1} + \frac{\log(BER3) - \log(BER2)}{CNR3 - CNR2}} + CNR3 \quad (4.16)$$

The results of the required CNR are listed in Table 4.1 for BP algorithm [33], and all other algorithms discussed in this chapter. Except the result of BP algorithm which we include from [33], the results for other algorithms discussed in this paper are obtained through simulation. In this table, the result for BP algorithm uses floating simulation and the results for other algorithms include a 6-bit quantization. The row Δ BP(TPMP) in the table indicates the average differences of required CNR for all the code rates compared to the BP algorithm in [33]. As can be seen from Table 4.1, the proposed algorithm is only 0.226dB away from the standard BP algorithm [33], and is about 0.12dB better than the MS-based algorithms in average.

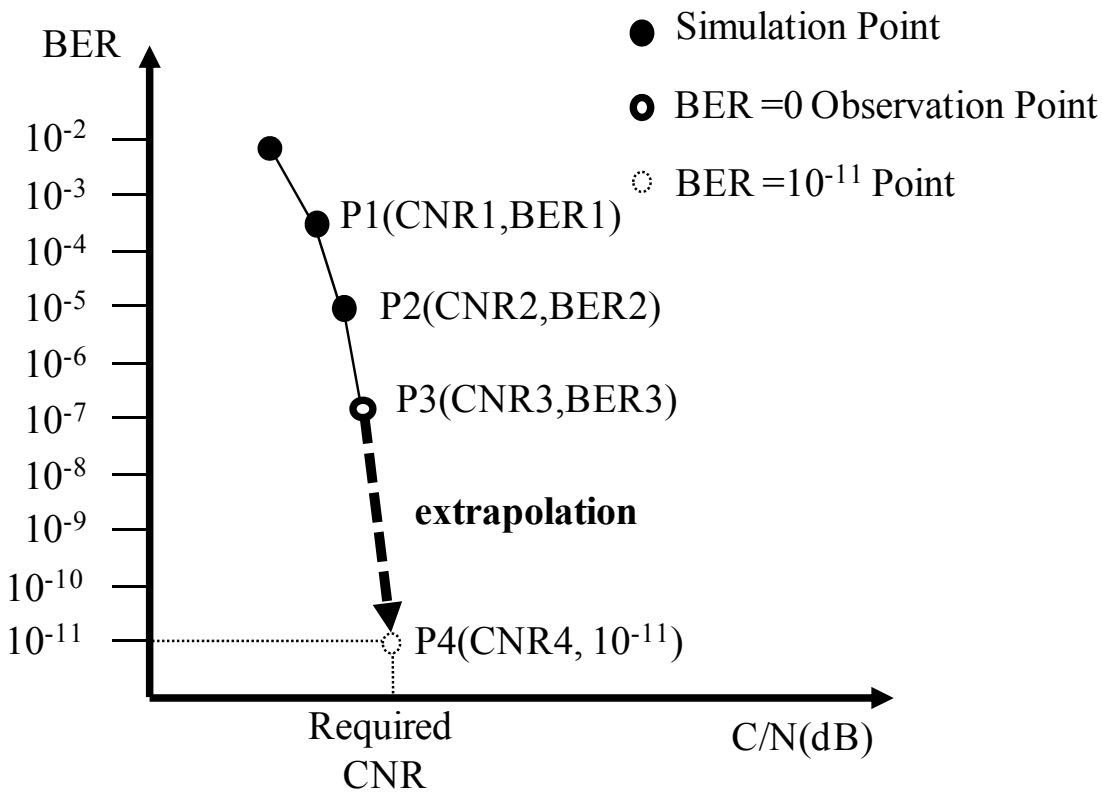


Figure 4.7 Required CNR calculation using extrapolation

Table 4.1 Comparison of required CNR (dB)

Rate	BP (floating point)[33]	MS-based				BP-based		Proposed
		NMS	OMS	DMMS	AOMS	MMS	DM	
1/4	-2.1	-1.43	-1.29	-1.38	-1.35	-1.55	-1.44	-1.41
1/3	-1.0	-0.61	-0.50	-0.52	-0.41	-0.83	-0.73	-0.51
2/5	0.0	0.36	0.34	0.48	0.54	0.15	0.24	0.37
1/2	1.2	1.51	1.46	1.57	1.56	1.43	1.37	1.41
3/5	2.5	2.79	2.81	2.66	2.66	2.58	2.57	2.57
2/3	3.3	3.59	3.58	3.46	3.46	3.38	3.37	3.37
3/4	4.0	4.31	4.39	4.36	4.36	4.19	4.05	4.19
4/5	5.0	5.28	5.37	5.28	5.15	5.08	5.07	5.07
5/6	5.5	5.79	6.00	5.66	5.66	5.58	5.46	5.58
7/8	5.9	6.29	6.07	6.28	6.28	6.08	6.09	6.07
9/10	6.8	7.08	6.88	7.00	7.00	6.78	6.77	6.88
Δ BP	0	0.351	0.365	0.341	0.346	0.161	0.156	0.226

4.6 Comparison of computation complexity and hardware cost

Although the BER performance of BP-based algorithms outperforms the proposed algorithm, their computation complexity and hardware cost can not be neglected. The comparison of computation complexity and the hardware cost of the check node

operation for one row (exclude the sign computation) are listed in Table 4.2.

Table 4.2 Comparison of computation complexity and hardware cost

	Computation Complexity	Hardware Cost
NMS	2×|A(m)|×[comp]+2×[shift]	2×[adder5]
OMS	2×|A(m)|×[comp]+2×[add]	4×[adder5]
DMMS	(2×|A(m)|+4)×[comp]+3×[shift] +7×[add]	5×[adder5]+ 6×[adder6]
AOMS	2×|A(m)|×[comp]+1×[shift] +2×[add]	4×[adder5]+8word×3bit[LUT]
MMS	|A(m)|×(|A(m)|-2) ×(3×[comp]+3×[add])	|A(m)|×(4×[adder6]+2×[adder5])
DM	|A(m)|×(|A(m)|-2) ×(1×[comp]+3×[add] +1×[shift])	|A(m)|×(4×[adder5])
Proposed	2×|A(m)|×[comp] +4×[add] +2×[shift]	6×[adder5]

Under column *computation complexity*, [*comp*], [*add*], [*shift*] indicate the computation complexity of comparison operation, addition or subtraction operation, and shift operation, respectively. For MS-based algorithm, $|A(m)|$ items are compared serially to get the minimum and the second minimum value, so $2\times|A(m)|\times[comp]$ is needed. After that, normalization factor or offset factor is applied to minimum and second minimum value, so additional calculations for the normalization factor γ and offset factor ε in Equation (4.5) and Equation (4.6) are needed. For the proposed algorithm, after the minimum value and the second minimum value are found, according to Equation (4.15), we require two more subtraction, two more shift operation and 2 subtraction for offset. For BP-based algorithm, Equation (4.10) or Equation (11) is invoked $(|A(m)|-2)$ times for each $n(n\in A(m))$ and a total of $|A(m)|$ different n values, thus requiring $|A(m)|\times(|A(m)|-2)$ times of Equation (10) or

Equation (4.11). For rate 9/10 with the biggest row weight $|A(m)|$ of 32 among all the parity check matrices in ISDB-S2, the computation complexity relation between NMS, OMS, DMMS, AOMS, MMS, DM and the proposed method is 1.03 : 1.03 : 1.22 : 1.05 : 90 : 75 : 1.09. The computation complexity for the proposed algorithm is similar to MS-based algorithms, and much smaller compared to BP-based algorithms. Figure 4.8 shows the relation of average computation complexity and average required CNR for all the rates in ISDB-S2. From the figure, we can see that the proposed algorithm consumes much less computation complexity compared to the BP-based algorithms but can achieve much better error correcting performance compared to MS-based algorithm with almost the same computation complexity.

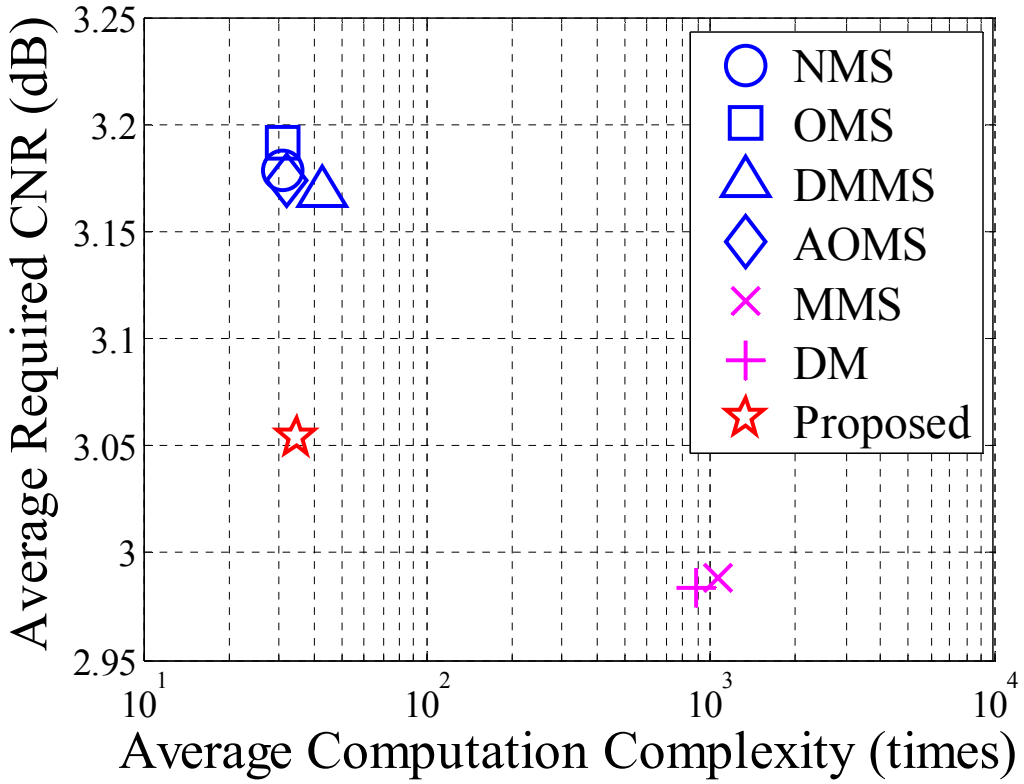


Figure 4.8 Average required CNR vs. average computation complexity

We also estimate the hardware cost for one check node operation (exclude the sign operation) using gate counts. The estimation results are listed under column *hardware cost* with [*adder*5] and [*adder*6] indicating the cost of an adder or subtractor for 5 bits and 6 bits. Note that we assume a comparator shares a similar cost with an

adder, and we neglected the cost for shifter. To keep almost the same clock cycles for one check node operation for all algorithms, MMS and DM require a parallel implementation of comparison, thus making the hardware cost almost $|A(m)|$ times as the other algorithms. In Figure 4.9, we show the relation of area cost and average required CNR for all the rates in ISDB-S2. The adder is estimated as 6 gates per bit and the LUT is estimated as 10 gates per bit. The figure demonstrates a similar trend as Figure 4.8 that the proposed algorithm greatly reduces the area compared to the BP-based algorithms while achieves much better error correcting performance than MS-based algorithms.

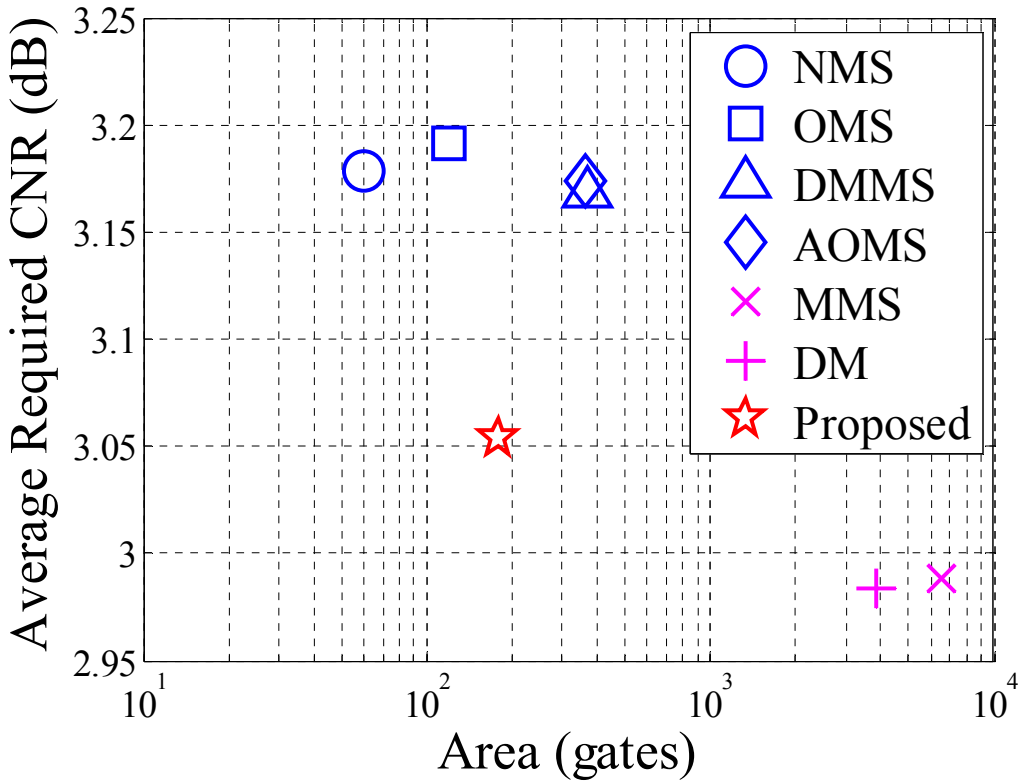


Figure 4.9 Average required CNR vs. area

4.7 Conclusion

In this chapter, in order to achieve high BER performance for satellite transmission services, a novel self-adjustable offset min-sum algorithm is proposed

with the check node operation approximating BP algorithm. The correctness of the approximation is proved by mathematical induction through using Jacobian logarithm iteratively. The proposed algorithm is hardware-friendly compared to the BP-based algorithms and the simulation results show that the proposed algorithm can achieve an average of 0.12dB gain compared to Min-sum based algorithms.

5 Data conflict resolution for layered schedule targeting high BER performance

5.1 Introduction

LDPC code can be efficiently decoded by two phase message passing (TPMP) algorithm [1] introduced in Section 1.3, which can correct errors through messages exchange between check nodes and bit nodes by performing check node and bit node operations iteratively. In terms of different implementation of the check node operation, TPMP algorithm can be categorized as Belief Propagation (BP) algorithm [1], Min-sum (MS) algorithm [18], Normalized Min-sum (NMS) algorithm [19] and Offset Min-sum (OMS) algorithm [19], etc. BP algorithm has the best error correcting performance, yet not hardware-friendly due to the implementation of Hyperbolic functions. MS algorithm uses the minimum magnitude of inputs from the bit nodes as a replacement of the Hyperbolic functions, but incurs great performance degrading. In chapter 4, we proposed a Self-adjustable Offset Min-sum (SOMS) algorithm which can adjust the offset value according to the inputs during the iterative decoding procedure and achieves a 0.12dB improvement of BER performance compared to MS-based algorithms.

On the other hand, in 2003, a new family of decoding algorithm called layered algorithm or layered schedule is proposed by Mansour [26]. Because layered algorithm shares the same check node operation as TPMP algorithm, the decoding methods for the check nodes in TPMP can be applied to layered algorithm, forming

the new Layered BP (LBP) algorithm, Layered Min-sum (LMS) algorithm, Layered Normalized Min-sum (LNMS) algorithm and Layered Offset Min-sum (LOMS) algorithm. And they maintain a similar pros and cons as in TPMP algorithms. Of course, the Self-adjustable Offset Min-sum (SOMS) algorithm proposed in Chapter 4 for ISDB-S2 application can also be applied to layered algorithm to achieve further high BER performance.

Instead of using A Posteriori Probability (APP) message at the end of each iteration in traditional TPMP algorithm, layered algorithm uses the intermediate APP results between layers within iterations, and converges two times faster than TPMP algorithm [34]. Therefore, layered algorithm becomes more suitable for high BER performance and high throughput design for the satellite transmission services.

However data conflict problem happens when layered algorithm is directly applied to ISDB-S2 codes. This problem arises as the layered algorithm adopts a parallel computation among a layer of several rows, which ignores the data dependencies of APP messages, thus degrading BER performance. To solve the data conflict problem in layered algorithm, authors in [35] and [36] tried to split the layers through memory mapping and scheduling the matrices for DVB-S2 and DVB-T2 application, but these methods cannot eliminate all the conflicts and they are limited when they are used to ISDB-S2 codes because of a different code design. Ref. [37] proposed a method to approximate the APP value targeting DVB-S2, and authors in [36] also proposed to add dummy bit nodes for DVB-T2 to leverage the performance degrading. But neither of these methods can achieve conflict free performance and they introduce additional computations and storages.

In this work, we proposed a selective recalculation method to achieve conflict free performance by recalculating the inaccurately calculated values. This method enables a parallel implementation of the whole layer and can correct the inaccurate values after the parallel implementation based on the decision of a recalculation rule. The simulation results show that the proposed method can achieve better BER performance than the previous data conflict strategy in [37].

5.2 Data conflict

In this section, data conflict problem of layered algorithm is introduced. The reason for data conflict is explained and the characteristics of the LDPC parity check matrices which have data conflict problem is introduced through an example. The previous strategies to solve data conflict problem are discussed, and the proposed data conflict resolution by selective recalculation is introduced.

5.2.1 Reason for data conflict

As discussed in Section 5.1, layered algorithm is usually favored for high BER performance application. However, directly applying layered algorithm to ISDB-S2 LDPC codes will lead to data conflict. This is mainly because of the data dependency during the parallel execution within each layer. Here LNMS algorithm is used as an example to introduce the data conflict problem and the correspondent conflict resolutions, other MS-based algorithms share the same characteristics.

The bit node operation, check node operation and APP update operation of the LNMS algorithm can be expressed as Equation (5.1), Equation (5.2) and Equation (5.3), respectively.

$$\beta_{mn} = sum_n - \alpha_{mn} \tag{5.1}$$

$$\alpha_{mn} = \gamma \prod_{n' \in N(m) \setminus \{n\}} sign(\beta_{mn'}) \times \min_{n' \in N(m) \setminus \{n\}} |\beta_{mn'}| \tag{5.2}$$

$$sum_n = \beta_{mn} + \alpha_{mn} \tag{5.3}$$

Figure 5.1 shows an example of a layer with two sub-blocks (size b = 3). Each element "1" in the matrix indicates a corresponding α and β message, and there is an APP data sum_n associated with each column n. Take column 1 as an example, if a sequential decoding process is applied, the bit node operation, check node operation and APP update operation for column 1 are listed as Equation (5.4):

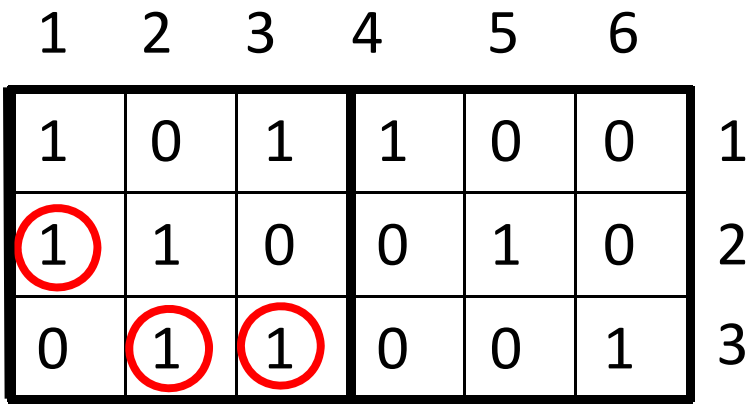


Figure 5.1 Example of data conflict in layered algorithm

$$
\begin{aligned}
&\beta_{11} = sum_1 - \alpha_{11} \\
&\alpha_{11} = \gamma sign(\beta_{13}) sign(\beta_{14}) \times \min(|\beta_{13}|, |\beta_{14}|) \\
&sum_1(1) = \beta_{11} + \alpha_{11} \\
&\beta_{21} = sum_1(1) - \alpha_{21} \\
&\alpha_{21} = \gamma sign(\beta_{22}) sign(\beta_{25}) \times \min(|\beta_{22}|, |\beta_{25}|) \\
&sum_1(2) = \beta_{21} + \alpha_{21}
\end{aligned}
\tag{5.4}
$$

Here, β_{11} and α_{11}, for example, stand for the message β and message α at row 1 and column 1, and sum_1 represents the APP value of column 1. $sum_1(1)$ and $sum_1(2)$ are the APP value updated for the first time and the second time. Note that, in the sequential decoding process, the calculation of message β_{21} uses the updated APP value $sum_1(1)$, and the last updated APP value ($sum_1(2)$) is passed to the next layer as the initial APP value for the next layer. If a layer of b (b = 3) rows are processed in parallel instead, the bit node operation, check node operation and APP update operation of this parallel calculation for column 1 are listed as Equation (5.5):

$$
\begin{aligned}
&\beta_{11} = sum_1 - \alpha_{11} \\
&\beta_{21} = sum_1 - \alpha_{21} \\
&\alpha_{11} = \gamma sign(\beta_{13}) sign(\beta_{14}) \times \min(|\beta_{13}|, |\beta_{14}|) \\
&\alpha_{21} = \gamma sign(\beta_{22}) sign(\beta_{25}) \times \min(|\beta_{22}|, |\beta_{25}|) \\
&sum_1(1) = \beta_{11} + \alpha_{11} \\
&sum_1(2) = \beta_{21} + \alpha_{21}
\end{aligned}
\tag{5.5}
$$

As can be seen, the calculation for β_{21} uses the old APP value sum_1 other than the updated data $sum_1(1)$. The incorrect calculation of β_{21} will not only result in the incorrect result of $sum_1(2)$ but also make the calculation of α_{22} and α_{25} inaccurate either. Such data dependency problem in layered algorithm is called data conflict, and will degrade the decoding performance. In this example, all the "1"'s with red circle have data conflict problem. In general, when applying layered decoding algorithm to structured LDPC codes, data conflicts happens when there are more than two "1"'s in one column within the layer.

Simply ignoring the data conflict will affect the performance, which makes layered schedule less effective than TPMP schedule. After a complete simulation of all the parity check matrices in ISDB-S2, we find out that the worst case happens for rate 7/8 with 9.85% of the sub-blocks having data conflict problems, and an average of 4.67% for all the codes. The small number of conflicts encourages us to apply layered schedule with proper remedy for data conflict to achieve better performance.

5.2.2 Previous data conflict resolution

To solve the data conflict problem, [35] proposed to use group splitting through memory mapping for DVB-S2, while [36] considered to achieve group splitting through reconstructing the matrices by scheduling for DVB-T2. However these two methods cannot eliminate all the data conflicts and they are limited when they are used for ISDB-S2 matrices because that the size of the layer of ISDB-S2 (b = 374) has

less factors than that of DVB-S2 or DVB-T2 (b = 360). Ref. [36] also introduced a method to add dummy bit nodes and check nodes to reconstruct the matrices which can help leverage the performance degrading, but introduced additional computations and storages. Also, it cannot achieve conflict free performance.

Table 5.1 shows the simulation result of the decoding performance of several layer sizes using the proposed algorithm in Chapter 4. The required CNR is evaluated using extrapolation as introduced in Section 4.4 [33]. Here we mainly consider those layer sizes whose value is the factor of 374 to simplify the decoding control logic. The results of required CNR metric of these layer sizes for different code rates are demonstrated in Table 5.1. Data under column *374P* is the result of computing 374 rows in parallel (*i.e.*, layer size of 374). It can be interpreted as simply ignoring all the data conflicts. Column *1P* is the result of scheduling with layer size of 1, which is the simulation result of conflict free schedule. Columns *187P*, *34P* and *22P* illustrate the simulation results for layer size of 187, 34 and 22 respectively. As can be observed from the table, layer size of both 34 and 22 achieve better performance than TPMP but has performance loss compared to the conflict free one, not to mention the low throughput due to procedure based on small layer sizes.

Ref. [37] proposed a method to approximate the APP value which enables the parallel implementation for 360 rows for DVB-S2. The method in [37] can be introduced based on the example in Figure 5.1. The authors in [37] analyzed the relation between the intermediate APP values ($sum_1(1)$ and $sum_1(2)$) and the initial APP value (sum_1) for the current layer, and after the parallel calculation shown in Equation (5.5), Equation (5.6) is calculated to approximate the updated APP value for current layer (sum'_1) which will be used as the initial APP value for the next layer.

$$sum'_1 = sum_1(1) + sum_1(2) - sum_1 \qquad (5.6)$$

Table 5.1 Required CNR for different layer sizes (dB)

Rate	TPMP	374p	187p	34p	22p	1p
1/4	-1.41	-0.91	-1.15	-1.55	-1.55	-1.55
1/3	-0.51	-0.49	-0.49	-0.52	-0.56	-0.67
2/5	0.37	0.44	0.38	0.38	0.38	0.34
1/2	1.41	1.4	1.4	1.38	1.38	1.38
3/5	2.57	2.86	2.58	2.46	2.46	2.45
2/3	3.37	3.47	3.36	3.25	3.25	3.25
3/4	4.19	4.61	4.28	4.05	4.05	4.05
4/5	5.07	5.44	5.18	5.07	5.08	5.07
5/6	5.58	6.07	5.67	5.57	5.57	5.45
7/8	6.07	6.39	6.17	6.07	6.07	6.07
9/10	6.88	7.45	7.11	6.88	6.88	6.88
Average	3.054	3.339	3.135	3.004	3.001	2.975

However, the APP approximation method cannot achieve the conflict free BER performance and it introduces additional storages for the intermediate APP values. In this work, we proposed for the first time a method called selective recalculating to enable a parallel implementation of 374 rows for ISDB-S2 to achieve conflict free BER performance.

To further explain the efficiency of these layered schedules, we compare, in Figure 5.2, the simulation results with the method in [37] for code rate 7/8, the code with the most conflicts in ISDB-S2. As can be seen in the figure, schedules with the layer size of 187, 34 and 22 all achieve better performance than [37]. However, none of these data conflict resolutions can achieve conflict free BER performance.

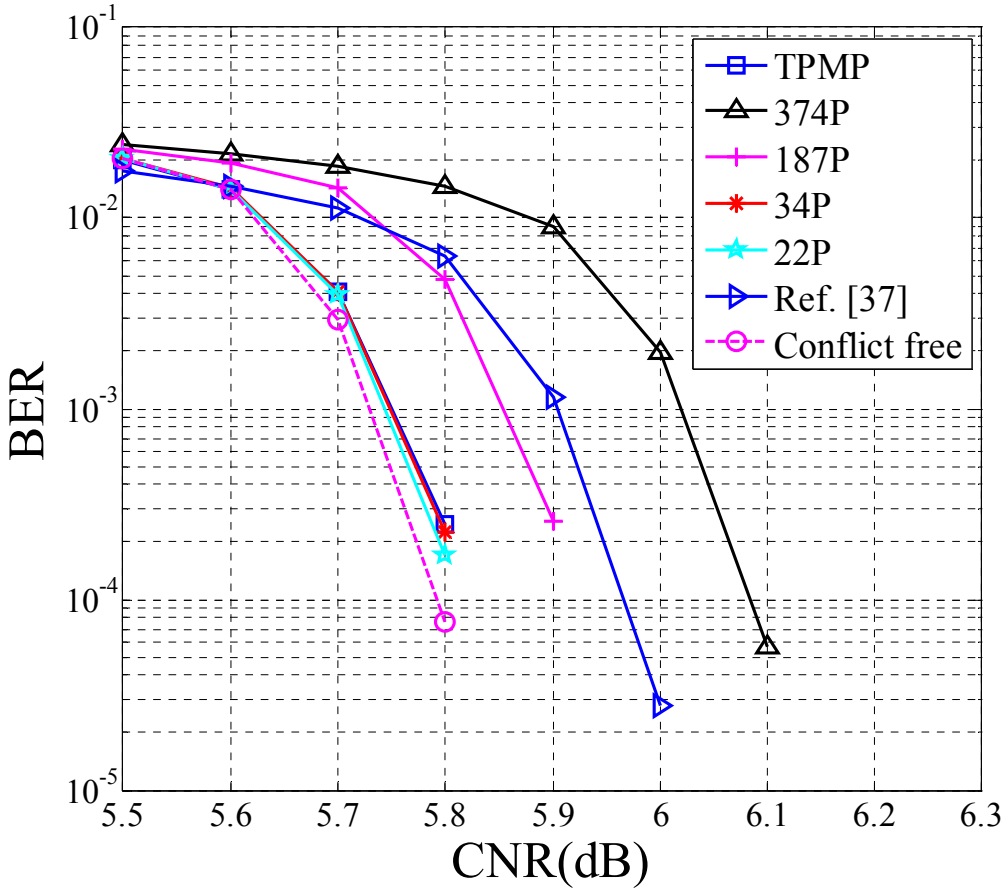


Figure 5.2 BER performance comparison of different solving strategies for data conflict problem for rate 7/8

5.3 Proposed data conflict resolution by selective recalculation

In this section, the proposed data conflict resolution scheme which can achieve conflict free BER performance is introduced. A detailed memory-saving strategy is also discussed to realize the proposed data conflict resolution scheme.

5.3.1 Selective recalculation scheme

In the example shown in Figure 5.1, because of the parallel calculation of a layer,

β_{21} is incorrectly calculated which will further result in the incorrect calculation of α_{22} and α_{25} by using Equation (5.2). The error will propagate to sum_1, sum_2, sum_5 based on Equation (5.3). In order to compensate for the error, we considered to recalculate the problematic β values and the related α and *sum* values after the computation of each layer.

However, according to Equation (5.2), not all the α and *sum* values related to the recalculated β values need to be adjusted. To demonstrate that, we divide Equation (5.2) into two parts, the calculation of the sign of α_{mn} (Equation (5.7)) and the calculation of the absolute value of α_{mn} (Equation (5.8)).

$$sign(\alpha_{mn}) = \prod_{n' \in A(m) \backslash n} sign(\beta_{mn'}) \qquad (5.7)$$

$$|\alpha_{mn}| = \gamma \times \min_{n' \in A(m) \backslash n} |\beta_{mn'}| \qquad (5.8)$$

It can be observed in Equation (5.8) that the absolute values of α_{mn} have only two values: either the minimum β values from a total of $|A(m)|$ β_{mn} values or the second minimum in the case that the corresponding input β value happens to be the minimum one. So Equation (5.8) can be rewritten as Equation (5.9). Here β_{min1} and β_{min2} indicate the minimum and second minimum β values among all β_{mn} values. The condition $pos(\beta_{mn}) = pos(\beta_{min1})$ indicates the position of β_{mn} under calculation is the position of β_{min1} and only in this condition the absolute value of β_{mn} equals $\gamma \times |\beta_{\min 2}|$.

$$|\alpha_{mn}| = \begin{cases} \gamma \times |\beta_{\min 2}| & pos(\alpha_{mn}) = pos(\beta_{\min 1}) \\ \gamma \times |\beta_{\min 1}| & otherwise \end{cases} \qquad (5.9)$$

From Equation (5.7) and Equation (5.9), it is apparent that the sign of α_{mn} is only related to the sign of β values input from the bit nodes, and the absolute value of αmn is only related to the minimum and second minimum β values input from the bit nodes and the position of the minimum β value. So the calculation of Equation (5.2) requires recording the signs of all β values in row *m* and tracking the minimum and second minimum β values. However, the inaccurately calculated β_{mn} values should be excluded from the calculation of minimum and second minimum β values. We denote the minimum and second minimum β values

without considering the inaccurate β_{mn} values in row m as $min1\_m$ and $min2\_m$ from now on and provides the recalculation decision rule in Figure 5.3.

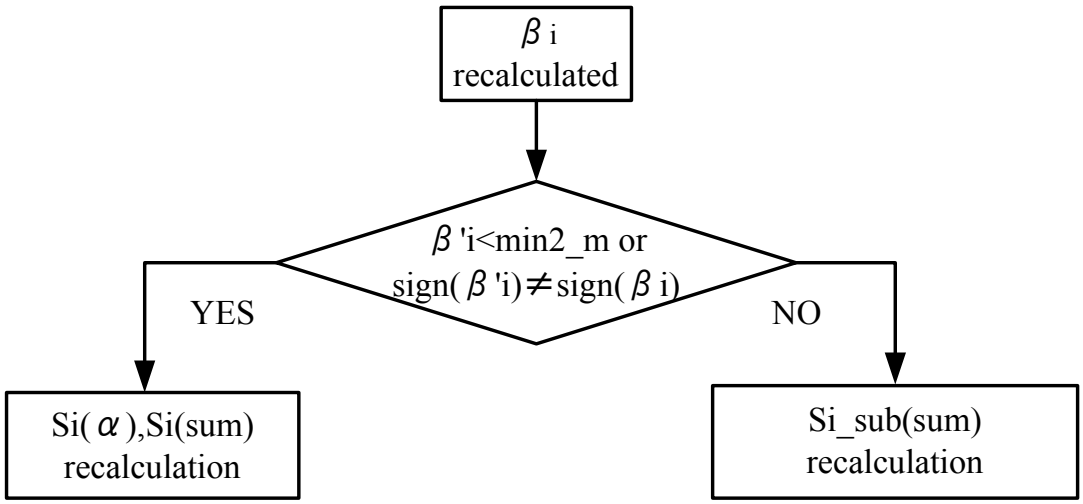


Figure 5.3 Recalculation decision rule for $S_i(\alpha)$ and $S_i(sum)$

In Figure 5.3, we denote the β values which should be recalculated as set $S(\beta)$, β_i as the i^{th} β_{mn} value in set $S(\beta)$, and β_i' as the i^{th} β_{mn} value after recalculation. We also denote the α values and *sum* values which should be recalculated because of the recalculation of β_i as $Si(\alpha)$ and $Si(sum)$. When condition $\beta_i' < min2\_m$ or $sign(\beta_i') \neq sign(\beta_i)$ satisfies, according to Equation (5.2), $S_i(\alpha)$ should be recalculated because of the alteration of input variables in Equation (4), and $S_i(sum)$ should also be recalculated because of the recalculation of $S_i(\alpha)$ according to Equation (5.3). When condition $\beta_i' < min2\_m$ or $sign(\beta_i') \neq sign(\beta_i)$ does not satisfy, the results of Equation (5.2) are not changed, so only a subset of $S_i(sum)$ (sum_n) needs to be recalculated because of the recalculated β_i' (Equation (5.3)).

Since usually $|A(m)|$ for ISDB-S2 is a large number, the probability for $\beta_i < min2\_m$ almost equals to $2/|A(m)|$ which is quite small. Moreover, as the sign for β_i is not altered frequently, the recalculation decision rule is rarely satisfied. Even when the recalculation decision rule satisfies, the recalculations of $S_i(\alpha)$ and $S_i(sum)$ are not complicated and the detailed recalculation techniques will be introduced later.

Based on the above exploration, we finally get our selective recalculation scheme

by using the recalculation decision rule, as shown in Figure 5.4. Here X represents all the "1" in the layer, and $S_m(\beta)$ indicates the subset of $S(\beta)$ for the β in the same row m. Note that the recalculation for rows can also be computed in parallel as long as there are no further data conflicts during the recalculation, *i.e.*, there are no two "1"'s in the same column of these simultaneously calculated rows. Basically, the proposed selective recalculating method is effective for any layered algorithm based on MS algorithm, such as LMS algorithm, LNMS algorithm and LOMS algorithm and the proposed algorithm in Chapter 4. Also, the proposed method can be used in any structured LDPC parity check matrix, not limited by ISDB-S2. Figure 5.5 shows how our selective recalculation works for the code in Figure 5.1.

Algorithm 1 Selective recalculation scheme

initialization $sum_n = \lambda_n$
for each layer **do**
  bit node, check node, APP update operation for set X
  if data conflict happens **then**
    for each subset of $S_m(\beta)$ in $S(\beta)$ **do**
      recalculate $S_m(\beta)$
      for each β_i' **do**
        if recalculation selecting rule satisfied **then**
          recalculate $S_i(\alpha)$
          recalculate $S_i(sum)$
        else
          recalculate $S_i\_sub(sum)$
        end if
      end for
    end for
  end if
end for

Figure 5.4 Selective recalculation scheme

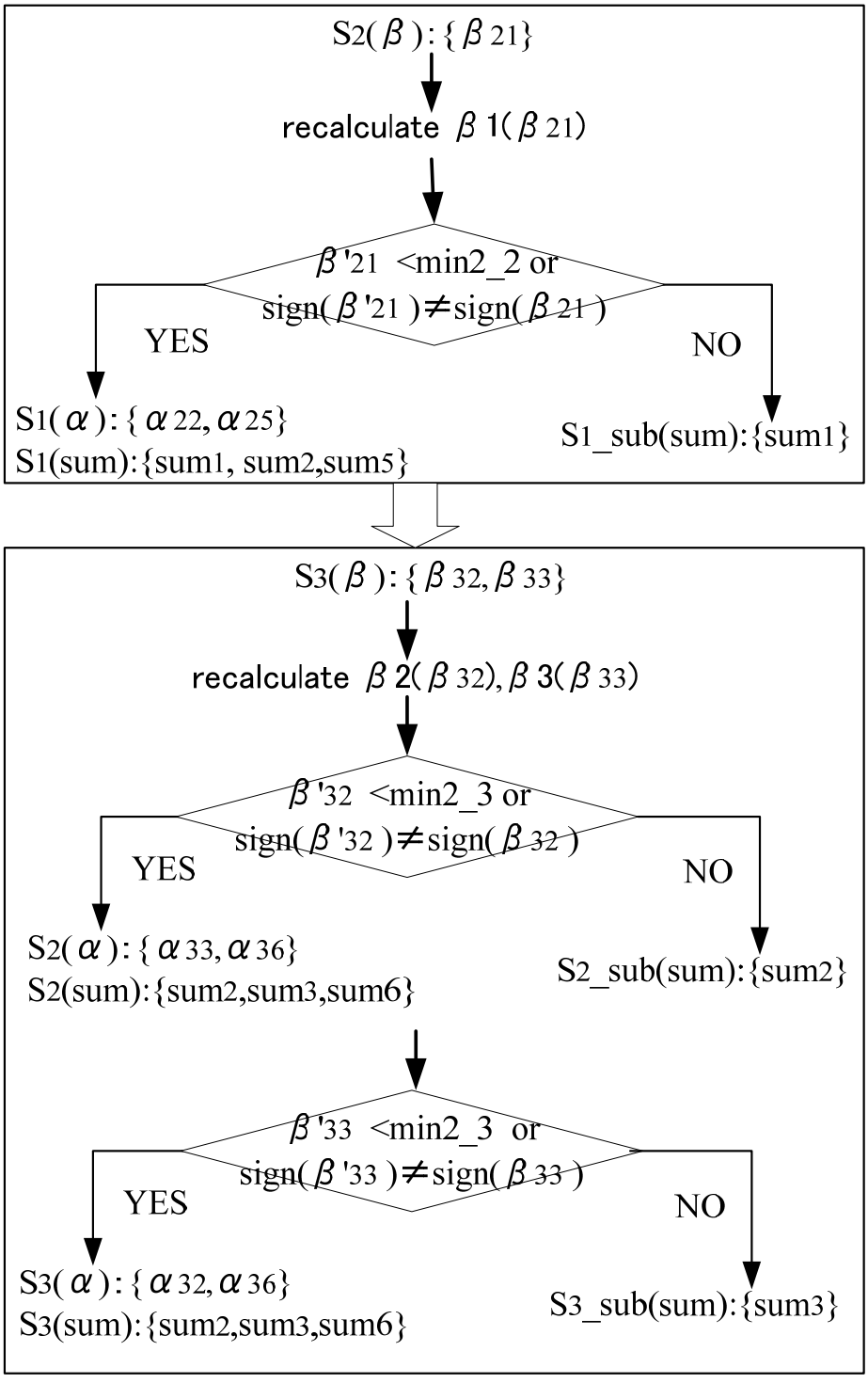


Figure 5.5 Example of data conflict resolution by selective recalculation

5.3.2 Realization of selective recalculation scheme

If the recalculation selection rule in Figure 5.3 satisfies, $Si(\alpha)$ and $Si(sum)$ need to be recalculated. As can be seen in Equation (5.2) and Equation (5.3), the recalculation for the check node operation is comparably complicated and time consuming. However, the recalculation can be easily carried out by favoring a memory-saving strategy which can not only save the memory for message α but also accelerate the procedure for the recalculation of check node operation.

As explained in Section 5.3.1, the result of α_{mn} is only related to the minimum and the second minimum β messages sent from bit nodes, the position of the minimum β message and the sign of all the β messages. Hence the storage of α values can be divided into its absolute value and sign bit. In this design, we favor a memory saving strategy based on this special characteristic. Instead of storing all the α messages in the memory, we only store $min1\_m$, $min2\_m$, the position of $min1\_m$ and the signs of α messages in the form of a vector as [$min1\_m$, $min2\_m$, pos_min1_m, $\{sign(\alpha_{mn})|n \in A(m)\}$][35]. With this form of representation, each α value can be easily retrieved through Equation (5.7) and Equation (5.9). This method can help save memories, specifically for rate 9/10 which has the largest row weight $|A(m)|$ of 32 in ISDB-S2, whose α memory is only 24.5% of that of storing all α messages.

As shown in Figure 5.3, $S(\alpha_i)$ and $S(sum_i)$ will be recalculated only if the recalculated β_{mn}' is smaller than the $min2\_m$ or the sign of the recalculated β_{mn}' is different from that of β_{mn}. In the following parts, we will discuss, in five different cases, how we can easily derive the recalculated value[$min1'\_m$, $min2'\_m$, $pos\_min1'\_m$, $\{sign(\alpha'_{mn})|n \in A(m)\}$] using this vector representation.

- Case 1: $\beta_{mn}' < min1\_m$ and $sign(\beta_{mn}') = sign(\beta_{mn})$.

In this case, the minimum value, the second minimum value and the position of the minimum value are all changed. Therefore, $min1'\_m = |\beta_{mn}'|$, $min2'\_m = min1\_m$ and $pos\_min1'\_m = pos(\beta_{mn}')$.

- Case 2: $\beta_{mn}' < min1\_m$ and $sign(\beta_{mn}') \neq sign(\beta_{mn})$

In this case, the minimum value , the second minimum value, the position of the minimum value and the signs of α are changed. Therefore, $min1'\_m = |\beta_{mn}'|$, $min2'\_m = min1\_m$ and pos_min1'_m = $pos(\beta_{mn}')$, and the signs of α messages are all changed except that of α_{mn}.

- Case 3: $min1\_m < \beta_{mn}' < min2\_m$ and $sign(\beta_{mn}') = sign(\beta_{mn})$.

In this case, the second minimum value is changed. Therefore $min2\_m' = |\beta_{mn}'|$.

- Case 4: $min1\_m < \beta_{mn}' < min2\_m$ and $sign(\beta_{mn}') \neq sign(\beta_{mn})$

In this case, the second minimum value and the signs of α are changed. Therefore, $min2\_m' = |\beta_{mn}'|$, and the signs of α messages are all changed except that of α_{mn}.

- Case 5: $\beta_{mn}' > min2\_m$ and and $sign(\beta_{mn}') \neq sign(\beta_{mn})$.

In this case, the signs of α messages are all changed except that of α_{mn}.

We can see that the recalculation of $S(\alpha_i)$ can be done without calculating Equation (5.2) again. Such scheme can not only save the memory but also expedite the recalculation procedure.

5.4 Simulation result

In this section, the performance of the proposed selective recalculation is presented compared to the result in Chapter 4.

5.4.1 BER performance

Software simulation of the proposed conflict resolution scheme @has been conducted for all 11 parity check matrices used in ISDB-S2 using the algorithm proposed in Chapter 4. The QPSK modulation and AWGN channel is modeled in the simulation. The maximum number of iteration is set to 50, and the simulation program terminates when the decoded codeword is valid or the iteration upper bound is

reached. Figure 5.6 and Figure 5.7 illustrate the simulation result of the BER performance of rate 3/5 and rate 3/4 which will be mainly used in ISDB-S2 service. Except BP algorithm is simulated using floating values, all the intermediate messages of simulations for the other algorithms are coded in 6 bit sign-magnitude format and the APP message is realized in an 8 bit sign-magnitude format to avoid overflow. The parameters of all algorithms are chosen to optimize both the BER performance and hardware implementation as γ =0.875 for NMS (Equation (4.5)), ε = 0.125 for OMS (Equation (4.6)), and γ' = 0.125 for the proposed method (Equation (4.15)). Also, for simple hardware implementation, we use the same Δ function $\Delta(x)$ = max(5/8 − |x|/4, 0) as [26] for approximation of function $f(x)$ for the proposed algorithm in this work. The line labeled "proposed (layered)" indicates the simulation result of proposed algorithm in Chapter 4 after applying layered schedule and the data conflict is solved by selective recalculation proposed in this chapter.

As can be observed from Figure 5.6 and Figure 5.7, after applying the layered schedule to the proposed algorithm in Chapter 4, the proposed algorithm can achieve further BER improvement and performs better than BP-based algorithms.

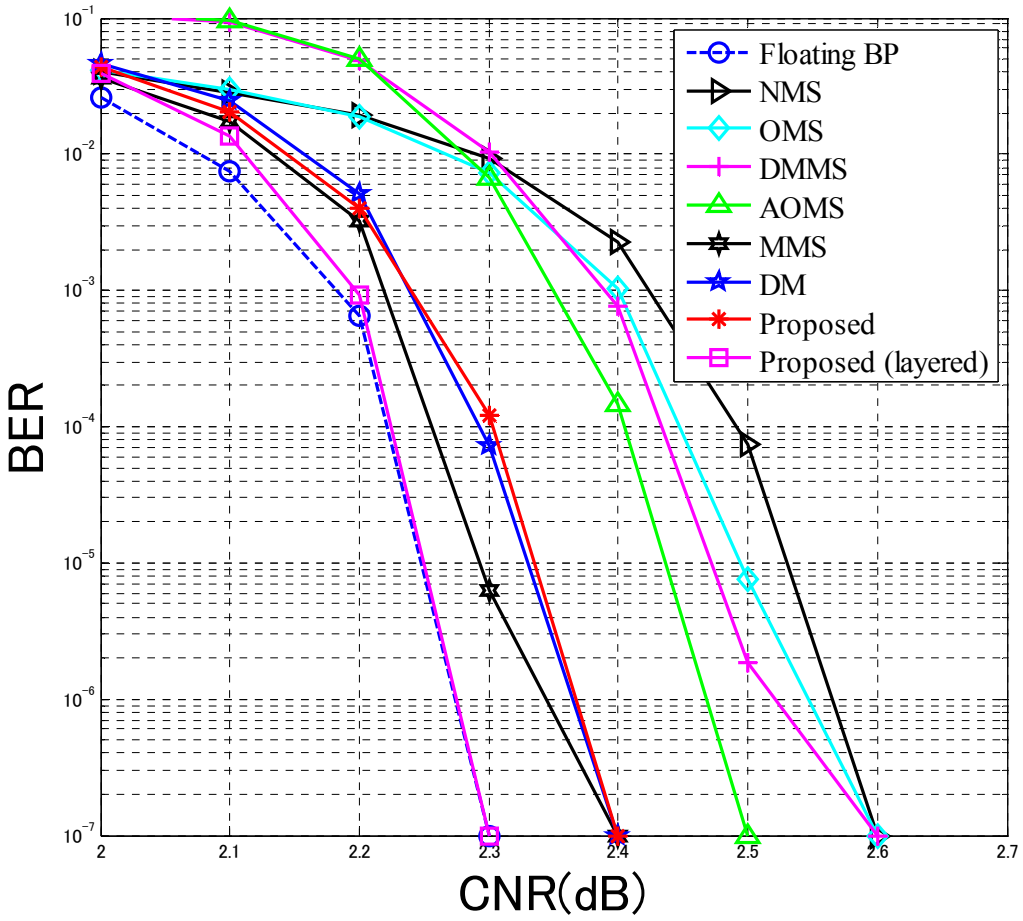


Figure 5.6 BER performance comparison for rate 3/5 (including layered schedule)

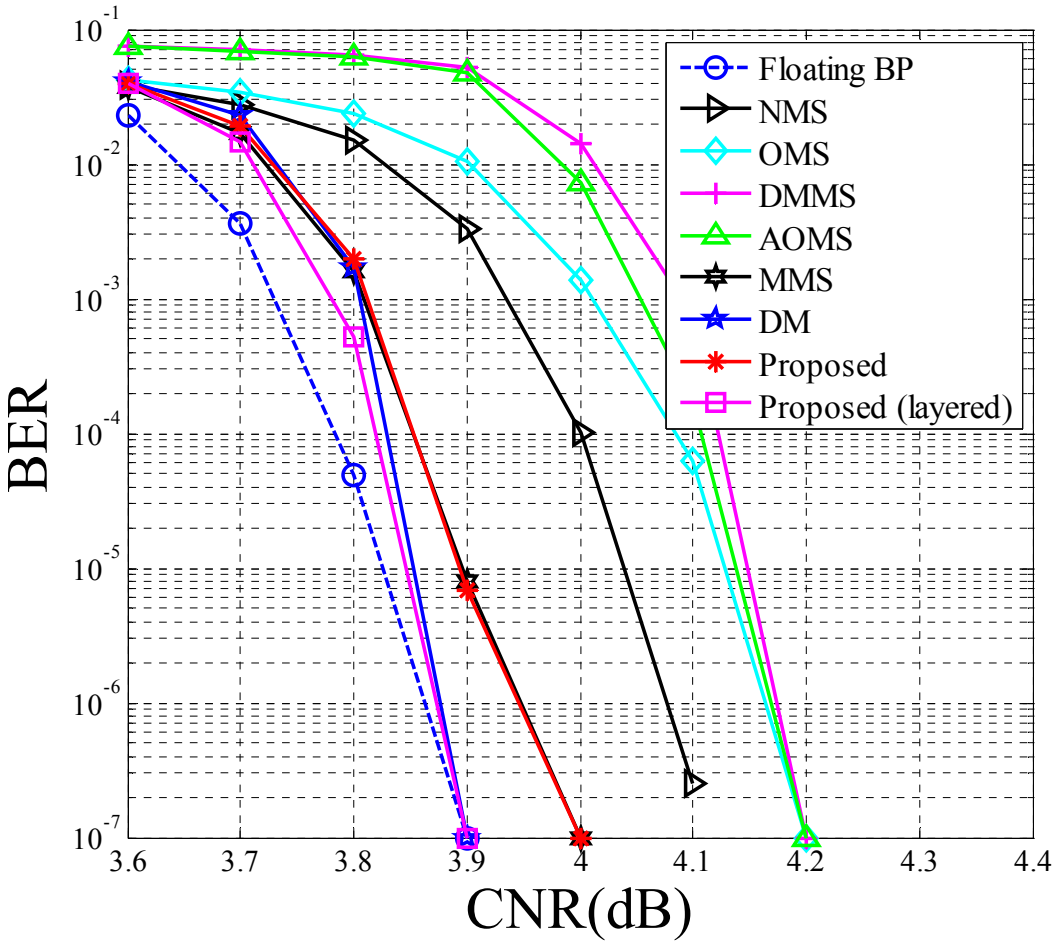


Figure 5.7 BER performance comparison for rate 3/4 (including layered schedule)

5.4.2 Comparison of required CNR

In order to further analyze the efficiency of the proposed selective recalculation and its suitability to all the LDPC codes in ISDB-S2, we use the same metric as Section 4.5, required CNR, to evaluate the performance. The required CNR is defined as the carrier-to-noise ratio when the BER exceeds 10^{-11} for ISDB-S2 [33]. Because of the error floor free performance of ISDB-S2 code and relatively long computer simulation time to evaluate the BER down to the range of 10^{-11}, in this work, we use the same evaluation method as in Chapter 4, namely extrapolation, to calculate the required CNR.

The results of the required CNR of all the codes in ISDB-S2 are listed in Table

5.2 for BP algorithm [33], NMS algorithm, OMS algorithm, DMMS algorithm, AOMS algorithm, MMS algorithm, DM algorithm and the proposed algorithm with TPMP schedule. The result of the proposed algorithm of layered schedule with selective recalculation scheme to solve the data conflict problem is also listed in this table. Except the result of BP algorithm which we include from [33], the results for others are obtained through simulation. In this table, the result for BP algorithm uses floating point simulation and the results for others include a 6-bit quantization process. The row ΔBP in the table indicates the average differences of required CNR for all the code rates compared to the BP algorithm in [33]. As can be seen in this table, after applying layered schedule to the proposed algorithm, it can achieve 0.2dB BER improvement compared to MS-based algorithms.

Table 5.2 Comparison of required CNR after applying layered schedule (dB)

Rate	BP (floating point)[33]	MS-based				BP-based		Proposed	
		NMS	OMS	DMMS	AOMS	MMS	DM	TPMP	layered
1/4	-2.1	-1.43	-1.29	-1.38	-1.35	-1.55	-1.44	-1.41	-1.55
1/3	-1.0	-0.61	-0.50	-0.52	-0.41	-0.83	-0.73	-0.51	-0.67
2/5	0.0	0.36	0.34	0.48	0.54	0.15	0.24	0.37	0.34
1/2	1.2	1.51	1.46	1.57	1.56	1.43	1.37	1.41	1.38
3/5	2.5	2.79	2.81	2.66	2.66	2.58	2.57	2.57	2.45
2/3	3.3	3.59	3.58	3.46	3.46	3.38	3.37	3.37	3.25
3/4	4.0	4.31	4.39	4.36	4.36	4.19	4.05	4.19	4.05
4/5	5.0	5.28	5.37	5.28	5.15	5.08	5.07	5.07	5.07
5/6	5.5	5.79	6.00	5.66	5.66	5.58	5.46	5.58	5.45
7/8	5.9	6.29	6.07	6.28	6.28	6.08	6.09	6.07	6.07
9/10	6.8	7.08	6.88	7.00	7.00	6.78	6.77	6.88	6.88
Δ BP	0	0.351	0.365	0.341	0.346	0.161	0.156	0.226	0.148

5.5 Conclusion

In this chapter, in order to further improve the BER performance for satellite transmission services, layered algorithm is applied to the proposed algorithm in Chapter 4 and the data conflict problem is completely solved through a selective recalculation method. After applying selective recalculation to layered algorithm, it can achieve conflict free performance. The simulation results of applying selective recalculation method to the proposed algorithm in Chapter 4 using layered schedule also demonstrate that the proposed method can achieve a further BER performance improvement, which can achieve 0.2dB gain under the same BER performance than MS-based algorithms.

6 Conclusion

6.1 Summary of results

In order to solve the low throughput problem for partially-parallel LDPC decoder for IEEE 802.11n application, two methods are proposed:

The first schedule is named Delta-Value based Message Passing (DVMP) schedule (proposed in Chapter 2). In this design, row operation is speeded up by a modified binary searching scheme and column operation is speeded up by DVMP schedule, in which the redundant computations are removed through using the difference between the updated value and the original value. Moreover a pipeline structure is utilized to further compact the procedure. The synthesis result demonstrates that our decoder can achieve a much higher throughput and almost the same bit error rate performance compared to other partially-parallel irregular LDPC decoders.

The other schedule is proposed based on the DVMP schedule proposed in Chapter 2, whose name is Sum-Delta Message Passing (SDMP) schedule (proposed in Chapter 3). In this design, the decoding throughput is greatly improved by utilizing the difference value between the updated and the original value to remove redundant computation. Registers ad memory are optimized to store only the frequently used messages to decrease the hardware cost. The synthesis result shows that our decoder can achieve much higher throughput and almost the same bit error rate performance with less area cost to other partially-parallel irregular LDPC decoders. The backend design of this decoder is implemented by Synopsys Astro with ARM's Artisan SAGE-X 0.18μm 1P6M stand-cell library for TSMC and the layout area is 13.69mm^2.

In order to achieve high BER performance for ISDB-S2 application, two techniques are also proposed to improve the BER performance:

The first technique is introduced in Chapter 4. In order to achieve high BER performance for satellite transmission services, a novel self-adjustable offset min-sum algorithm is proposed with the check node operation approximating BP algorithm. The correctness of the approximation is proved by mathematical induction through using Jacobian logarithm iteratively. The proposed algorithm is hardware-friendly compared to the BP-based algorithms and the simulation results show that the proposed algorithm can achieve an average of 0.12dB gain compared to Min-sum -based algorithms.

The other technique is introduced in Chapter 5 to further apply layered schedule to ISDB-S2 codes. In order to achieve high BER performance for satellite transmission services, layered algorithm is applied to the proposed algorithm in Chapter 4 to ISDB-S2 LDPC decoder and the data conflict problem is completely solved through a selective recalculation method. After applying selective recalculation to the proposed algorithm in Chapter 4 using layered schedule, it can achieve 0.2dB gain under the same BER performance compared to Min-sum -based algorithms.

6.2 Future work

As long LDPC code is adopted in satellite transmission to ensure the transmission quality, the hardware design for long code becomes a challenge because the hardware cost is large [35][37][41][42][43]. A hardware architecture which is suitable for the implementation of long code and multi-rate code is a main future work for us.

On the other hand, the LDPC code design is an important topic in this field. A good LDPC code can achieve good error correcting performance as well as small hardware cost. There are already some methods for designing LDPC codes which proves to be efficient, but the design should be vary for different application and requirement [44][45][46].

Also, for wide application, multi-standard LDPC decoder design (for example, IEEE 802.11 and IEEE 802.13) or flexible decoder architecture for multiple error correction codes (for example, LDPC code and turbo code) is also a good topic for research in LDPC field [47][48].

REFERENCE

[1] Gallager, R. G., “Low-Density Parity-Check Codes”, MIT Press, Cambridge, MA , 1963

[2] Shannon, C., “A mathematical theory of communication”, *Bell Syst. Tech. J.*, Vol.27,pp.379–423,623–656, 1948

[3] MacKay, D.J.C., “Good error-correcting codes based on very sparse matrices”, *IEEE Trans. Inform. Theory*, Vol.45, No.2, pp.399–431, 2001

[4] Richardson, T. J., Sholrollahi, M. A. and Urbanke, R. L., “Design of capacity approaching low-density parity-check codes”, *IEEE Trans. Inform. Theory*, Vol.47, No.2, pp.619–637, 2001

[5] S.Y., Forney, G.D., Richardson, T.J. and Urbanke, R.L., “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, 2001

[6] *World Wide Web, IEEE 802.3an Task Force*, http://www.ieee802.org/3/an/index.html, 2004

[7] ETSI, D. V.B.: *Second generation framing structure, channel coding and modulation systems for Broadcasting, Interative Services, News Gathering and other broadband satellite applications*, draft EN 302 307 V1.1.1 edition, 2004

[8] *IEEE Standard for Local and Metropolitan Area Networks, IEEE 802.16e Standard*, http://standards.ieee.org/getieee 802/download/802.16e-2005.pdf, 2006

[9] Blanksby, A. and Howland, C., “A 690-mW 1-Gbps 1024-b, rate-1/2 low-density parity-check code decoder”, *J. Solid State Circuits*, Vol.37, No.3, pp.404–412, 2002

[10] Chen, Y. and Hocevar, D., “A FPGA and ASIC implementation of rate 1/2 8088-b irregular low density parity check decoder”, *IEEE Global Telecommunications Conf.*, pp.113–117, 2003

[11] Mansour, M. and Shanbhag, N., “Low power VLSI decoder architectures for LDPC codes”, *Proc. Int. Symp. Low Power Electronics & Design*, pp.284–289, 2002

[12] Liao, E., Yeo, E. and Nikolic, B., “Low-density parity-check code constructions for hardware implementation”, *Proc. IEEE Conf. Communications*, pp.2573–2577, 2004

[13] Shimizu, K., Ishikawa, T., Togawa, N., Ikenaga, T. and Goto, S., “Power-efficient LDPC decoder architecture based on accelerated message-passing schedule”, *IEICE Trans. Fundamentals*, Vol.E89-A, No.12, pp.3602–3612, 2006

[14] Li, X., Abe, Y., Shimizu, K., Qiu, Z., Ikenaga, T. and Goto, S., “Cost-efficient parallel irregular LDPC decoder with message passing schedule”, *Int. Symp. Integrated Circuits*, pp.548–551, 2007

[15] Chien, Y.H. and Ku, M.K., “A high throughput H-QC LDPC decoder”, *IEEE Int. Symp. Circuits & System*, pp.1648–1652, 2007

[16] Wang, Q., Shimizu, K., Ikenaga, T. and Goto, S., “A power-saved 1Gbps irregular LDPC decoder based on simplified min-sum algorithm”, *VLSI Design, Automation and Test (VLSI-DAT)*, pp.95–98, 2007

[17] *IEEE P802.11 Wireless LANs Joint Proposal: High throughput extension to the 802.11 Standard: PHY*, IEEE 802.11-05/1102r4, Jan. 2006

[18] M. Fossorier, M. Mihaljevic and H. Imai, “Reduced Complexity Iterative Decoding of Low Density Parity Check Codes Based on Belief Propogation”, *IEEE Trans. Commun.*, vol. 47, pp. 673-680, May 1999

[19] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossoeier and X.-Y. Hu, “Reduced-Complexity Decoding of LDPC Codes”, *IEEE Trans. Commun.*, vol. 53, pp.1288-1299, Aug. 2005

[20] J. Chen and M. Fossorier, “Density evolution for two improved BP-based decoding algorithms of LDPC codes”, *IEEE Commun. Lett.*, vol. 6, pp. 208-210, May 2002

[21] Zarkeshvari, F. and Banihashemi, A.H., “On implementation of min-sum algorithm for decoding low-density parity-check (LDPC) codes”, *Global Telecommunications Conference, 2002. (GLOBECOM ’02)*, vol. 2, pp. 1349-1353, Nov. 2002

[22] Jinghu Chen and Marc P. C. Fossorier, “Near optimum universal belief propagation based decoding of low-density parity check codes”, *IEEE Trans. commun.*, vol. 50, no. 3, pp. 406-414, March 2002

[23] Jinghu Chen, Tanner, T.M. Jones, C. and Yan Li, “Improved min-sum decoding algorithms for irregular LDPC codes”, *IEEE proc. International Symposium on Information Theory (ISIT’05)*, pp. 49-53, Sept. 2005

[24] Darabiha, A., Carusone, A.C. and Kschischang, F.R., “A bit-serial approximate min-sum LDPC decoder and FPGA implementation”, *IEEE Proc. International Symposium on Circuits and Systems, 2006. (ISCAS’06)*, pp. 149-152, May 2006

[25] Yuta Abe, X. Li, K. Shimizu, T. Ikenaga and S. Goto, "High-Throughput Design of High-Efficiency Message Passing partially parallel irregular-LDPC decoder based on 802.11n", *The 2008 International Conference on Embedded Systems and Intelligent Technology (ICESIT2008)* , Thailand, 2008

[26] M. Mansour and N.R. Shanbhag, “High-Throughput LDPC Decoders”, *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 11(6): 976-996, Dec. 2003

[27] D. E. Hocevar, “A reduced complexity decoder architecture via layered decoding of LDPC codes”, *IEEE Workshop on Signal Processing Systems(SiPS’04)*, pp 107-112, Austin, USA, Oct. 2004

[28] A. Hashimoto and Y. Suzuki, “A new transmission system for the advanced satellite broadcast”, *IEEE Trans. Consum. Electron.*, vol. 54, Issue 2, pp. 353-360, May 2008

[29] S. L. Howard, C. Schlegel, and V. C. Gaudet, “Degree-matched check node decoding for regular and irregular LDPCs”, *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 10, pp. 1054-1058, Oct. 2006

[30] M. Jiang, C. Zhao, L. Zhang, and E. Xu, “Adaptive offset min-sum algorithm for low-density parity check codes”, *IEEE Commun. Lett.*, vol. 10, no. 6, pp. 483-485, June 2006

[31] A. Anastasopoulos, “A comparison between the sum-product and the min-sum iterative detection algorithms based on density evolution”, *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 2, pp. 1021-1025, Nov. 2001

[32] L. Sakai, W. Matsumoto, and H. Yoshida, “Reduced complexity decoding based on approximation of update function for low-density parity-check codes”, *IEICE Trans. Fundamentals*, vol. J90-A, no. 2, pp. 83-91, Feb. 2007

[33] Y. Suzuki, A. Hashimoto, M. Kojima, S. Tanaka, T. Kimura, and T. Saito, “LDPC codes for the advanced digital satellite broadcasting system”, *IEICE Technical Report*, vol. 109, no. 212, pp. 19-24, Sep. 2009

[34] P. Radosavljevic, A. D. Baynast, and J. R. Cavallaro, "Optimized message passing schedules for LDPC decoding", *Asilomar Conference on Signals, Systems and Computers 2005 (ACSSC)*, pp. 591-595, Oct. 2009

[35] J. Dielissen, A. Hekstra, and V. Berg, "Low cost LDPC decoder for DVB-S2", *Design, Automation and Test in Europe* (*DATE 2006)*, pp. 130-135, March 2006

[36] C. Marchand, J. B. Dore, L. C. Canencia, and E. Boutillon, "Conflict resolution by matrix reordering for DVB-T2 LDPC decoders", *IEEE Global Telecommunications Conference (GLOBECOM)*, Oct. 2009

[37] A. Segard, F. Verdier, D. Declercq, and P. Urard, "A DVB-S2 compliant LDPC decoder integrating the horizontal shuffle scheduling", *International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pp. 1013-1016, Dec. 2006

[38] A. Blad and O. Gustafsson, "Energy-efficient data representation in LDPC decoders", *IET Journals , Electronics Letters*, Vol. 42, Issue 18, pp. 1051-1052, 2006

[39] T. Theocharides, G. Link, E.Swankoski, N. Vijaykrishnan, M.J. Irwin and H. Schmit, "Evaluating alternative implementations for LDPC decoder check node function", *IEEE Computer Society Annual Symposium on VLSI*, pp. 77-82, 2004

[40] Tong Zhang, Zhongfeng Wang and Keshab K. Parhi, "On finite precision implementation of low density parity check codes decoder", *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 202-205, vol. 4, 2001

[41] Macro Gomes, Gabriel Falcao, Vitor Silva, Vitor Ferreira, Alaxandre Sengo and Miguel Falcao, "Flexible parallel architecture for DVB-S2 LDPC decoders", *Global Telecommunications Conference (GLOBECOM'07)*, pp. 3265-3269, Nov. 2007

[42] P. Urard, E. Yeo, L. Paumier, P. Georgelin, T. Michel, V. Lebars, E. Lantreibecq, B. Gupta, "A 135Mb/s DVB-S2 compliant codec based on 64800b

LDPC and BCH codes", *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 446-609 vol. 1, Feb. 2005

[43] P. Urard, L. Paumier, V. Heinrich, N. Raina, N. Chawla, "A 360mW 105MB/s DVB-S2 compliant codec based on 64800b LDPC and BCH codes enabling satellite-transmission portable devices", *IEEE International Solid-State Circuits Conference(ISSCC)*, pp. 310-311, Feb. 2008

[44] Yu Kou, Shu Lin, Marc P. C. Fossorier, "Low density parity check codes based on finite geometries: a rediscovery and new results ", *IEEE Trans. on Information Theory*, vol 47, issue 7, pp. 2711-2736, 2001

[45] Norifumi Kamiya, "High-rate quasi-cyclic low-density parity-check codes derived from finite affine planes", *IEEE Trans. on Information Theory*, vol. 53, no. 4, pp. 1444-1459, 2007

[46] Zhengang Chen, Tyler L. Brandon, Duncan G. Elliott, Stephen Bates, Witold A. Krzymien, and Bruce F. Cockburn, "Jointly designed architecture-aware LDPC convolutional codes and high-throughput parallel encoders/decoders", *IEEE Trans. on Circuits and Systems*, vol. 57, issue 4, pp. 836-849, 2010

[47] F. Naessens, V. Derudder, H. Cappelle, L. Hollevoet, P. Raghavan, M. Desmet, A. M. AbdelHamid, I. Vos, L. Folens, S. O'Loughlin, S. Singirikonda, S. Dupont, J.-W. Weijers, A. Dejonghe, L. Van der Perre, "A 10.37mm2 675mW reconfigurable LDPC and Turbo encoder and decoder for 802.11n, 802.16e and 3GPP-LTE", *2010 IEEE Symposium on VLSI Circuits (VLSIC)*, pp. 213-214, June 2010

[48] T. S. V. Gautham, Andrew Thangaraj, Devendra Jalihal, "Common architecture for decoding turbo and LDPC codes", *2010 National Conference on Communications (NCC)*, pp.1-5, Jan. 2010

LIST OF PUBLICATIONS

Journal

1. Wen Ji, Yuta Abe, Takeshi Ikenaga, Satoshi Goto, "A High Performance Partially-Parallel Irregular LDPC Decoder Based on Sum-Delta Message Passing Schedule", *IEICE Trans. Fundamentals*, Vol.E91-A, No.12 pp.3622-3629, Dec. 2008

2. Wen Ji, Xing Li, Takeshi Ikenaga, Satoshi Goto, "A High Throughput LDPC Decoder Design Based on Novel Delat-value Message-passing Schedule", *IPSJ Transactions on System LSI Design Methodology*, Vol. 2, pp 122-130, Feb. 2009

3. Wen Ji, Makoto Hamaminato, Hiroshi Nakayama and Satoshi Goto, "Self-adjustable offset min-sum algorithm for ISDB-S2 LDPC decoder", *IEICE Electron. Express*, Vol. 7, No. 17, pp.1283-1289, 2010

International Conference

1. Wen Ji, Xing Li, Takeshi Ikenaga, Satoshi Goto, "High Throughput Partially-Parallel Irregular LDPC Decoder Based on Delta-Value Message-Passing Schedule ", *VLSI Design, Automation and Test (VLSI-DAT)*, Hsinchu, Taiwan, pp.220-223, Apr. 2008

2. Wen Ji, Yuta Abe, Takeshi Ikenaga, Satoshi Goto, "A Cost-Efficient Partially-Parallel Irregular LDPC Decoder Based on Sum-Delta Message Passing Algorithm", *Great Lakes Symp. VLSI (GLSVLSI)*, Florida, USA, pp.207-212, May. 2008

3. Wenming Tang, Wen Ji, Xianghui Wei, Takeshi Ikenaga, Satoshi Goto, "A Power-saving 1Gbps Irregular LDPC Decoder based on High-efficiency Message Passing", *The 23rd International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, pp. 193-196, Shimonoseki, Japan, Jul.2008

4. Wen Ji, Yuta Abe, Takeshi Ikenaga, Satoshi Goto, "A High Performance LDPC Decoder for IEEE802.11n Standard", *14th Asia and South Pacific Design Automation Conference (ASP-DAC2009)*, Japan, pp. 127-128, Jan. 2009.

5. Wen Ji, Makoto Hamaminato, Hiroshi Nakayama, Satoshi Goto, "A Novel Hardware-friendly Self-adjustable Offset Min-sum Algorithm for ISDB-S2 LDPC Decoder", *2010 European Signal Processing Conference (EUSIPCO)*, Aalborg, Denmark, Aug. 2010

6. Wen Ji, Makoto Hamaminato, Hiroshi Nakayama, Satoshi Goto, "Data Conflict Resolution for Layered LDPC Decoding Algorithm by Selective Recalculation", *The 3rd International Congress on Image and Signal Processing (CISP)* , Yantai, China, Oct. 2010

Domestic Conference

1. Wen Ji, Yuta Abe, Takeshi Ikenaga, Satoshi Goto, "High Throughput Rate-1/2 Partially-Parallel Irregular LDPC Decoder", *2008 IEICE General Conference*, pp. 123, Kitakyushu, Japan, March. 2008

2. Wen Ji, Makoto Hamaminato, Hiroshi Nakayama, and Satoshi Goto, "An

Efficient Decoding Algorithm for ISDB-S2", *IEICE LDPC workshop 2009*, Tokyo, Japan, pp. 13-17, Sep. 2009