ALGORITHM DEVELOPMENT AND VLSI IMPLEMENTATION OF ENERGY EFFICIENT DECODERS OF POLAR CODES

A Dissertation

by

JINGWEI XU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Seong G. Choi
Committee Members,	Eun Kim
	Peng Li
	Narayanan Krishna
Head of Department,	Miroslav M. Begovic

May 2016

Major Subject: Computer Engineering

ABSTRACT

With its low error-floor performance, polar codes attract significant attention as the potential standard error correction code (ECC) for future communication and data storage. However, the VLSI implementation complexity of polar codes decoders is largely influenced by its nature of in-series decoding. This dissertation is dedicated to presenting optimal decoder architectures for polar codes. This dissertation addresses several structural properties of polar codes and key properties of decoding algorithms that are not dealt with in the prior researches. The underlying concept of the proposed architectures is a paradigm that simplifies and schedules the computations such that hardware is simplified, latency is minimized and bandwidth is maximized.

In pursuit of the above, throughput centric successive cancellation (TCSC) and overlapping path list successive cancellation (OPLSC) VLSI architectures and express journey BP (XJBP) decoders for the polar codes are presented.

An arbitrary polar code can be decomposed by a set of shorter polar codes with special characteristics, those shorter polar codes are referred to as constituent polar codes. By exploiting the homogeneousness between decoding processes of different constituent polar codes, TCSC reduces the decoding latency of the SC decoder by 60% for codes with length $n = 1024$. The error correction performance of SC decoding is inferior to that of list successive cancellation decoding. The LSC decoding algorithm delivers the most reliable decoding results; however, it consumes most hardware resources and decoding cycles. Instead of using multiple instances of decoding cores in the LSC decoders, a single SC decoder is used in the OPLSC architecture. The computations of each path in the LSC are arranged to occupy

the decoder hardware stages serially in a streamlined fashion. This yields a significant reduction of hardware complexity. The OPLSC decoder has achieved about 1.4 times hardware efficiency improvement compared with traditional LSC decoders. The hardware efficient VLSI architectures for TCSC and OPLSC polar codes decoders are also introduced.

Decoders based on SC or LSC algorithms suffer from high latency and limited throughput due to their serial decoding natures. An alternative approach to decode the polar codes is belief propagation (BP) based algorithm. In BP algorithm, a graph is set up to guide the beliefs propagated and refined, which is usually referred to as factor graph. BP decoding algorithm allows decoding in parallel to achieve much higher throughput. XJBP decoder facilitates belief propagation by utilizing the specific constituent codes that exist in the conventional factor graph, which results in an express journey (XJ) decoder. Compared with the conventional BP decoding algorithm for polar codes, the proposed decoder reduces the computational complexity by about 40.6%. This enables an energy-efficient hardware implementation. To further explore the hardware consumption of the proposed XJBP decoder, the computations scheduling is modeled and analyzed in this dissertation. With discussions on different hardware scenarios, the optimal scheduling plans are developed. A novel memory-distributed micro-architecture of the XJBP decoder is proposed and analyzed to solve the potential memory access problems of the proposed scheduling strategy. The register-transfer level (RTL) models of the XJBP decoder are set up for comparisons with other state-of-the-art BP decoders. The results show that the power efficiency of BP decoders is improved by about 3 times.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Dr. Gwan Choi, for his financial support and encouragement for my research. He supported me in all the difficult situations where I needed help. I would like to thank Dr. Krishna Narayanan, Dr. Peng Li and Dr. Eun Kim for their time in serving in my committee. I appreciate Dr. Narayanan's introduction and suggestions on polar codes, which made me focus exclusively on polar codes decoders though initially I set out to work on conglomeration of different topics. Dr. Peng Li has been very helpful and he gave me a lot of guidance not only on academic study but also on my career path. In addition, he did provide me with many opportunities to review peers' works, which broadened my horizons and inspired me to think problems in different aspects. I would also like to thank Dr. Eun Kim. Through her excellent and impressive teaching skills, I was enlightened on the ideas and concepts of modern computer architectures which are of the great significance to the fundamental of my research works.

Several other students and people at Texas A&M helped me in my research work. Thanks to Ehsan Rohani for his guidance on standard communication protocols to develop practical hardware architectures. Thanks to Mehnaz Rahman on her working on the Matlab simulation for my architecture on the compressive sensing receiver for MIMO-OFDM cognitive radio. Special thanks to Tiben Che, in particular, for his working on the verification of some of the HDL modules for my architecture on belief propagation decoder. In addition, Tiben spent several weeks with me in working on the paper writing. Thanks to Honghuang Lin, Jimmy Jin, for their helping and guiding me to set up efficient simulation programs by their expertise and carefulness on programming. I would like to thank Mr. Qian Wang for his guidance on the

exploration of hardware architectures.

My lovely wife, Lei, has supported me in much more ways than meets the eye. She did a difficult task of pursuing her Ph.D. degree at Texas A&M University, while taking care of different things at home. Last but not least, I would like to thank my parents for their constant support and encouragement through every major decision in my life.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

This introduction describes the demands of advanced error correction coding techniques for evolving communications and storage systems. With the introduction of the advantages of polar codes over other existing error correction codes, the main contributions of this dissertation is presented. Through the similar hardware development methodology, other contributions made for problems within same mathematical problem scope for advanced communication and storage system are also listed in this dissertation. The outlines of this dissertation is given in the end of this chapter to guide readers follow this dissertation.

1.1 Motivation

The communications markets are continuously facilitated by the diverse and increasing demands of acquiring data and sharing information. With the enormous amount of data swapped, the communication technology is undergoing tremendous rapid escalation from hundred kilobits per second 3G to gigabit per second 5G of mobile telecommunication, from multi-megabits per second to multi-gigabits per second of Ethernet communication, from multi-megabyte portable MP3 player to multi-gigabyte mobile smart phone. The progress and evolution on communication are primarily driven by the discoveries on information theory and advances on the integrated circuits.

Figure 1.1 shows a simplified block diagram of a digital communication system [1]. First, the source data of information such as voice, video is sampled and encoded through the source encoder, so as to be compressed to remove any unnecessary redundancy in the data. Then channel encoder codes the redundancy removed sequences so that it can recover the correct information after passing through the channel. The

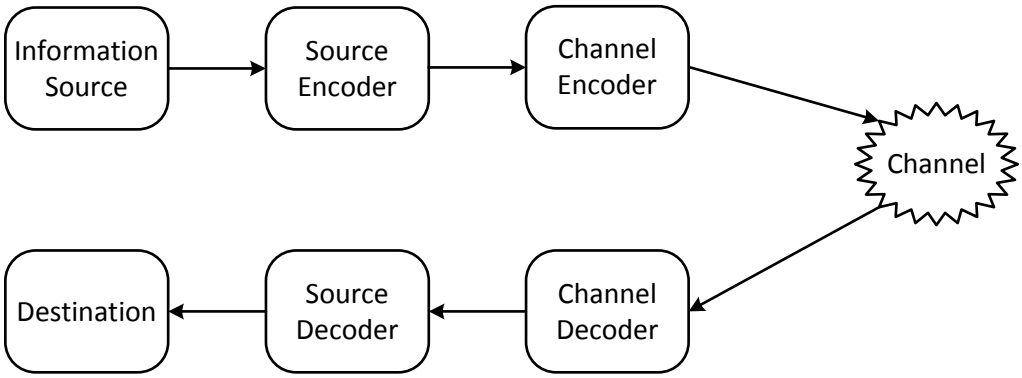


Figure 1.1: Block diagram of communication systems

coded bit sequences of data are transmitted through the channel to receiver. The mediums of channels could be varied as copper (wired communication), air (wireless communication), water (underwater communication), flash memory and so on.

There are two important topics in the communication. They are the source coding and channel coding for efficient compression as and the reliable transmission of the data respectively.

The information is firstly fed through the source encoder or data compression to remove unnecessary redundant data or compact and encapsulate data in smaller sizes. If the data consists of bank records or personal details we cannot afford to lose any information. In such cases, the compression is achieved by exploiting patterns in the data. Before transmission over noisy mediums, the compressed data are coded again by channel encoder, which is the second central topic of information theory. To make communication reliable in the presence of noise, the common procedure is to add redundancy to the data before transmission. The intended receiver only has access to a noisy version of the data. However, if the redundancy is added in a clever way, then it is possible to reconstruct the original data at the receiver. Adding

redundancy for reliable transmission is also referred as error correcting coding (ECC). Coding is a central part of any communication systems; e.g., consider wired phones, mobile phones, or the Internet. Coding is also used for storage on CDs and flash memories to prevent data loss due to scratches or errors during the reading process.

Since the Shannon's work on 1948 [2], research on ECC has developed for several decades. Error correcting codes such as convolutional [3], Turbo [4] and LDPC [5] are typical error correcting codes widely used in protocols. Among them, Turbo [6] and LDPC [7, 8] are proven to be very close to Shannon limit.

Although great success has been achieved by existing near-capacity error correction codes, promoted by the continuously growing demand for reliable big data storage and high-speed communication, the exploration on better correction codes does never stop. In 2008, Polar codes [9] are found as error correcting codes which provable achieve the capacity of symmetric binary-input discrete memoryless channels (B-DMCs).

1.2 Problem Overview

Polar codes proposed by Arikan [9] are not only first provably capacity achieving codes for any B-DMCs, but also have low encoding and decoding complexity. With its low error-floor performance [10] and high regularity in coding structure, polar codes attract a significant attention from the coding theory community [11–28] and have the potential to become a standard ECC for the future communication and data storage systems.

However, the VLSI implementation complexity of polar codes decoder is largely influenced by its nature of in-series decoding. There are three widely-considered approaches to decode polar codes. These are successive cancellation (SC), its variant successive cancellation list (SCL) decoders and belief propagation (BP) algorithms.

The SC algorithm receives more attentions because of its low computational complexity $\mathcal{O}(nlogn)$, where n is the code length. But, decoders based on SC algorithm suffer from the high latency and limited throughput due to their serial decoding natures. Recently several efforts have been taken into reducing the SC decoding latency [29, 30]. Sarkis et al. utilized the constituent codes that exist in the polar codes to significantly reduce the SC decoding latency by avoiding tree traversals [30]. Although the latency of SC algorithm is substantially improved, the time complexity of it is still $\mathcal{O}(n)$. Thus with longer polar codes, SC algorithm is still limited in terms of the throughput. However, polar codes with longer length are more attractive, because the performance of polar codes is superior to other codes at long codeword lengths.

Although polar codes have inherent capacity-achieving property, with small and medium code lengths, the error correction performances of polar codes are inferior to Turbo codes and LDPC. [14] proposes SCL decoder, which is also inherent serial decoding method similar as SC decoder. SCL improves the error correction performance of polar codes, it requires more hardware resource to upgrade the performance as well still suffers from the long latency which is intolerable for real-time transmission.

Another approach to decode the polar codes is belief propagation-based (BP) algorithm, which allows decoding in parallel to achieve much higher throughput in dedicated hardware implementation. Due to its higher computational demand, compared with SC algorithms, BP does not receive much attentions. The first attempt at implementing BP on field programmable gate array (FPGA) is presented by Pamuk in [31], where the message passing functions are approximated by the min-sum (MS) algorithm for efficient hardware design. However, the performance of BP decoding is degraded because of the approximations. Thus, Yuan et al. explored scaled

min-sum (SMS) approximation for message passing functions in [32] to remedy the performance penalty. However, compared with MS algorithm, SMS incurs one extra scaling operations in each message passing. Yuan et al. further improved the efficiency of SMS BP decoders using early termination in [33]. On the other hand, by removing unnecessary computations for frozen bits in polar codes, Zhang et al. reduce the complexity for sum-product (SP) BP decoding in [34] by around 25% without decoding performance degradation.

1.3 Main Contributions

Motivated by the challenges mentioned above on polar codes decoders, a set of methodologies are developed as the guidance to design a hardware-efficient polar codes decoders for next-generation communication and storage systems. Optimization on both algorithm and hardware levels are explored as the core methodology for achieving the objective. In particular, the main contributions of this work are listed in the following:

1. The simplifications on polar codes decoding algorithm which takes the advantage of the existences of the constitute codes of polar codes.

2. The static scheduling computation paradigm by which the proposed advanced belief propagation algorithm is used to reduce the computations, memory and interconnect.

3. The hardware architecture is explored from bottom up to implement the simplifications of polar code decoding algorithms and computation paradigms....

This dissertation summarizes works on two types of hardware-efficiency polar codes decoders based on SC, LSC decoding algorithm, that substantially reduce the

computational complexity over same type of state-of-the-art decoders. The dissertation will emphasize the work on algorithm development on the XJBP and the micro-architecture exploration and implementation for the XJBP decoder.

From the view of algorithm, two novel approaches are devised to achieve the improvements of XJBP decoding process. First approach is to utilize specific constituent codes in the factor graph to reduce the decoding complexity. In this approach, the rules of the belief propagation in each iteration are simplified using the characteristics of the constituent codes. Secondly, unlike conventional BP decoders scheduling mentioned in [31], our approach uses an alternative scheduling method stemming from ideas discussed by Park et al. at [35] and Guo et al. at [11]. In [35], unidirectional scheduling method is employed to reduce memory allocation. In [11], similar ideas are proposed to adapt decoding polar codes concatenated with parity check codes. We describe and compare the two different scheduling methods in this work to show that the alternative scheduling method that we adopt is significantly better than the conventionally used one in terms of decoding efficiency.

We show that along with the novel scheduling method, the XJ-BP MS algorithm yields the same decoding performance of the scaled-min-sum algorithm [32] with 92.8% reduced amount of computations. Compared with the traditional min-sum based BP decoding [31], the proposed method does not only reduce the computations by 90.4% but significantly improves the decoding performance.

As the effort to implement the XJBP decoder, a micro-architecture is presented as the platform to accomplish belief propagation efficiently. Based on the architecture, three different types of scheduling methods are presented in this dissertation to assign computation tasks economically. Scheduling strategy is developed based on different assumptions and scenarios of practical hardware environment. Corresponding hardware modules for different scheduling algorithms are described in RTL model

to estimate the hardware consumption. By simulations and analysis on scheduling algorithms and hardware results, the efficiency of different scheduling methods for XJBP decoder is summarized and compared with other state-of-the-art BP decoders. It is shown that with proper scheduling strategy, the proposed XJBP decoder could improve the energy efficiency of BP decoder by 3X.

1.4 Other Works

Besides works on polar code decoders, other works on low density parity check (LDPC) codes and compressive sensing for advanced communication and storage systems, which inherently have the problem of similar mathematical model and developed in a similar methodology, are also presented in this dissertation.

Before the invention of polar codes, LDPC codes are widely used because of its near-capacity error correction performance. Besides development of polar codes decoders, effort is made on improving energy efficiency of LDPC decoders. For this front, techniques of asynchronous circuits are employed to implement LDPC decoder.

Similar as error correction codes decoding, compressive sensing is another consistent problem in the field of signal processing. Specifically, its reconstruction problem is inherently connected with error correction decoding problem. Both problems are seeking for solutions by given a projection over a matrix. Besides the effort on the hardware setup on error correction decoder, a signal reconstructor is also developed for compressed sensing reconstruction problem. Also a novel framework on application of compressed sensing on MIMO-OFDM cognitive radio is proposed. Both the works on compressed sensing are presented in this dissertation.

1.5 Outline of this Dissertation

The rest of this dissertation is organized as follows: Chapter 2 first introduces the background of polar codes including its construction, principles and typical de-

coding algorithms. After introducing the constituent codes in Chapter 3 which is a phenomenon existing in a polar code coding structure, Chapter 4 presents existing typical polar codes decoder VLSI structures as reference works to compare with our proposed decoders. Chapter 5, 6 and 7 illustrate our works on the developments on three typical polar codes decoders respectively. After the summary of works developed on other relevant topics in Chapter 8, Chapter 9 summarizes my works on polar codes decoders.

2. POLAR CODES AND DECODING

In this chapter, the backgrounds of polar codes are introduced by discussing their construction and underlying concept to achieve the capacity of the B-DMC. Besides the overview of polar codes in [9], the three mainly used decoders are also introduced and discussed.

2.1 Polar Encoder

Polar codes are constructed by taking advantage of the polarization effect to achieve the capacity of symmetric channel. Encoded recursively using the special procedure as discovered in [9], the polar codes polarize the post-decoding reliability of the information bits. An (n, k) polar code is constructed by assigning k information bits and $(n-k)$ '0's at more reliable positions and unreliable positions, respectively. Those fixed '0' bits are usually referred as frozen bits. The n-bit message bits including frozen bits and information bits are denoted as $\mathbf{u}$ in this paper. The n-bit transmitted codeword $\mathbf{x}$ is the product of $\mathbf{u}$ and the generator matrix $\mathbf{G}$

$$\mathbf{G} = \mathbf{F}^{\oplus m}, \mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{2.1}$$

where $\mathbf{F}^{\oplus m}$ is the m-th Kronecker power of and $m = \log_2 n$.

We define the mutual information first before introducing more details of polar codes.

Definition 2.1.1. The mutual information of a B-DMC with input alphabet $\mathcal{X} =$

$0, 1$ is defined as:

$$I(W) \triangleq \frac{1}{2} \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} W(y|x) \log \frac{W(y|x)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1)} \tag{2.2}$$

In the definition, $W(y|x)$ denotes the probability of receiving $y \in \mathcal{Y}$, given that $x \in \mathcal{X}$ sent from the transmitter.

Note that the capacity of a symmetric B-DMC equals the mutual information between the input and output of the channel with uniform distribution on the inputs. $I(W)$ is a measure of rate in a channel, a.k.a. reliable communication is possible over a symmetric B-DMC at any rates up to $I(W)$.

Definition 2.1.2. The Bhattacharyya parameter of a channel is defined as

$$Z(W) \triangleq \sum_{y \in \mathcal{Y}} \sqrt{W(y|0)W(y|1)} \tag{2.3}$$

The Bhattaharyya parameter is a measure of the reliability of a channel since $Z(W)$ is an upper bound on the probability of maximum-likelihood (ML) decision error for uncoded tranmission over channel W.

In the original paper, the encoding complexity is exponential in the block length. To improve the efficency of the encoders of polar codes, construction methods [13, 18, 36] with linear complexity are proposed. In this thesis, we only focus on the efficiency improvement of polar codes decoders. In the following, the polarization phenomenon is introduced to discuss how the polar codes protect information bits.

2.2 Channel Polarization

Channel polarization is an operation that iteratively produces n channels from n independent copies of a B-DMC channel such that n channels are polarized in

terms of their mutual information is either close to 0 (completely noisy channels) or close to 1 (perfectly noiseless channels). As aforementioned, the mutual information represents the reliability of the channel. Thus, the polarization process creates some perfectly noiseless channels out of the identical independent B-DMC channels. The number of noiseless channels is equal the rate the B-DMC channel. Figure 2.1 shows an example case for 2 B-DMC channel polarization. As the figure shows, two separate channels are combined to create a new vector channel of size 2.

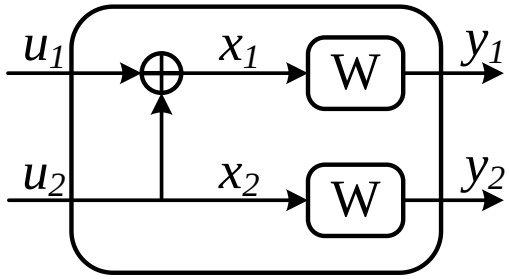


Figure 2.1: Channel polarization example of 2 B-DMC channels

$I(u_1; y_1, y_2)$ is the mutual information of the channel between u_1 and y_1 y_2. Let W^- denote the channel between them as $W^- : 0, 1 \rightarrow \mathcal{Y}^2$. Furthermore, $I(u_2; y_1, y_2, u_1)$ is the mutual information of the enhanced channel between u_2 and the output given that u_1 is known. Let W^+ denote this enhanced channel. The transition probabilities of these two channel can be written as

$$W^-(y_1, y_2|u_1) = \frac{1}{2} \sum_{u_2 \in 0,1} W(y_1|u_1 \oplus u_2) W(y_2|u_2) \tag{2.4}$$

$$W^+(y_1, y_2, u_1|u_2) = \frac{1}{2} W(y_1|u_1 \oplus u_2) W(y_2|u_2) \tag{2.5}$$

The above two equations mean that the combined channel is split into two channels W^+ and W^-. And two created channels are satisified the following properties[9].

$$I(W^+) + I(W^-) = 2I(W) \tag{2.6}$$

$$Z(W^-) \leq 2Z(W) - Z(W)^2 \tag{2.7}$$

$$Z(W^+) = Z(W)^2 \tag{2.8}$$

Through the properties, we can see that $I(W^+)$ and $I(W^-)$ are polarized to the extremes 0 and 1.

In a similar way, the polarization could be further extended to more channels. For n created channels from recursively applying operations 2.42.5, the reliability of n channels are polarized to the extremes.

In transmitting a binary message block of k bits, the k most reliable polarized channels $W^{(i)}$ are used to transmit k information bits. The indexes of the channels are denoted as $i \in \mathbb{I}$. The remaining channels are used to send a fixed binary sequence called frozen bits, which are usually are zeros. In set of indexed where frozen bits are transmitted are indicated as $\mathbb{F}$.

2.3 Successive Cancellation Decoding

One way to decode polar codes is to apply successive cancellation to estimate $\hat{u}_i$ using the channel output $y_1, y_2, ..., y_n$, denoted as y_1^n, and previously estimated $u_1, u_2, ..., u_{i-1}$, denoted as u_1^{i-1} [9]. The determination on $\hat{u}_i$ could be represented as:

$$\hat{u}_i = \begin{cases} 0 & i \in \mathbb{F} \\ \mathcal{H}(y_1^n, \hat{u}_1^{i-1}) & i \in \mathbb{I} \end{cases} \tag{2.9}$$

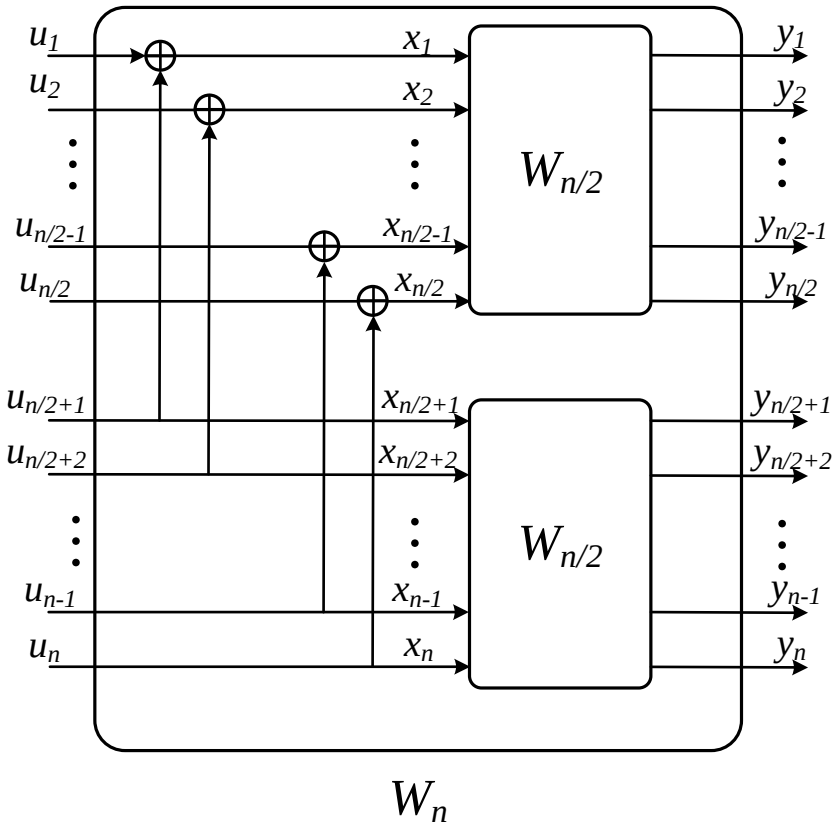


Figure 2.2: Recursive construction of n channel polarization

where

$$\mathcal{H}(y_1^n, \hat{u}_1^{i-1}) = \begin{cases} 1 & \text{if} \quad \frac{W(y_1^n, \hat{u}_1^{i-1}|1)}{W(y_1^n, \hat{u}_1^{i-1}|0)} \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

This approach is naturally represented by a binary tree whose each node corresponds to a decision for each transmitted information bit. For a polar code with code length n, the code tree is a perfect binary tree with depth $n+1$. The depth for a node v is the length of the path from the root. The root has depth of 0. Each level represents the estimation of one information bit transmitted. At each node, the path is selected based on the probability of $W(y_1^n, \hat{u}_1^{i-1}|\hat{u}_i)$. Figure 2.3 shows the

decoding tree for a example of $n = 4$ polar code. As the figure shows, the first bit is decoded only depending the channel outputs y_1^n, while the following bits depend on not only the channel outputs but also the previous decoded bits.

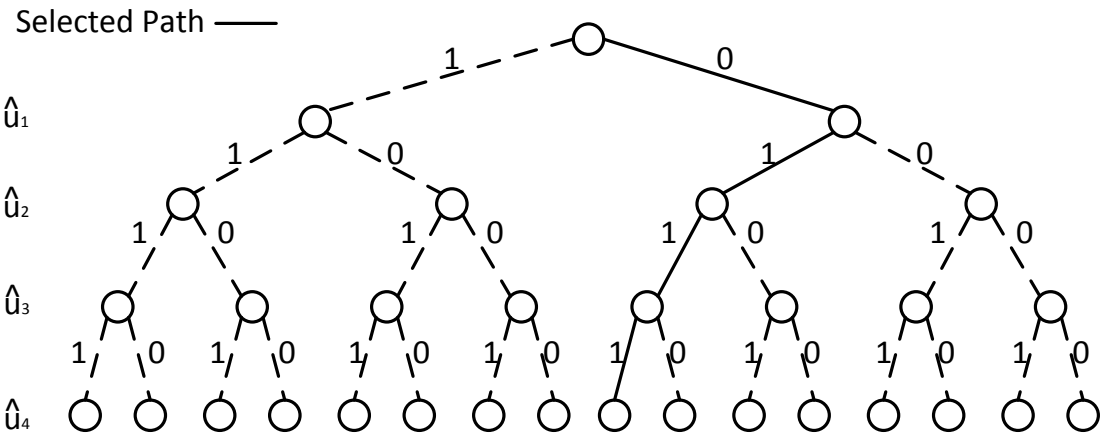


Figure 2.3: Successive cancellation path on decoding tree

2.4 List Successive Cancellation Decoding

The SC decoding algorithm of polar codes could be regarded as a greedy algorithm to estimate the transmitted binary sequences. Between the two path candidates in each level, the selection of path is based on the larger probability of $W(y, u_1^{i-1}|\hat{u}_i)$. However, if any one of the previous decoded bits are not selected, the most a-posterior probability (MAP) will be dismissed. The optimal path as the MAP estimation is defined as:

$$\hat{u}_i^n = argmax(W(y_i^n|\hat{u}_i^n)) \tag{2.11}$$

In this case, if multiple paths are reserved in every layer, the decoding reliability will be enhanced considerably. This fashion of SC decoding with multiple survival paths are referred as LSC, which stands for list successive cancellation decoding [14].

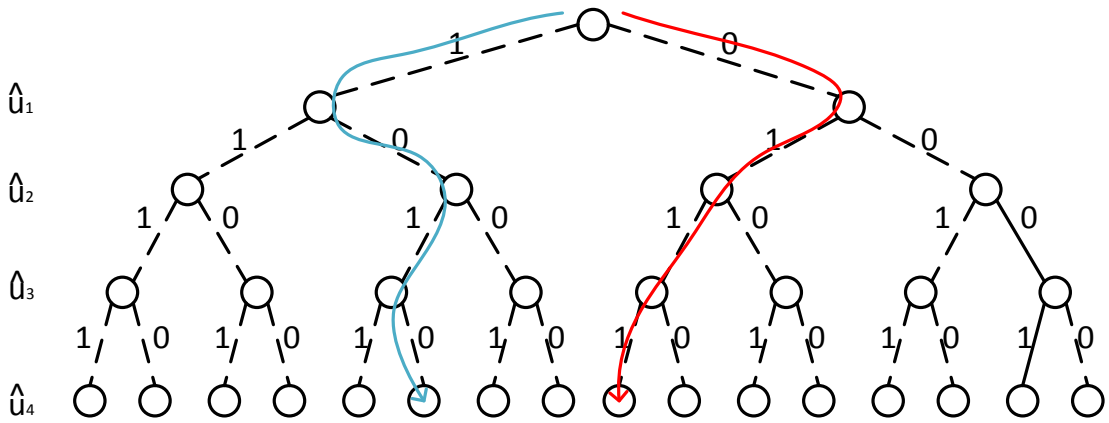


Figure 2.4: List successive cancellation decoding paths on decoding tree

Figure 2.4 shows the two decoding paths reserved by list successive cancellation decoders while decoding through the decoding tree. The size of list in the figure is two. However, the size of the list is variable depending on the reliability requirements of the LSC decoder.

2.5 Belief Propagation Decoding

An alternative way to decode polar codes is based on factor graph representation of the equation 2.1 as described by Arikan [37]. Belief propagation decoding is a message passing algorithm that, through the factor graph, refines the estimations of the codeword $\mathbf{x}$ or message $\mathbf{u}$ in iterations.

The factor graph of a polar code could be represented by the structure of its encoder. An example of factor graph of a polar code with $N = 8$ is given in Figure 2.5. As the figure shows, there are n stages in the factor graph, $n = \log_2(N)$. The bits on the most left column correspond to the message. In the figure, the black nodes and white nodes in the left column are denoted as the frozen bits and the information bits respectively. With recursive encoding by the 2-bit polarization unit through the factor graph, the nodes on the most right column correspond to the codeword. There

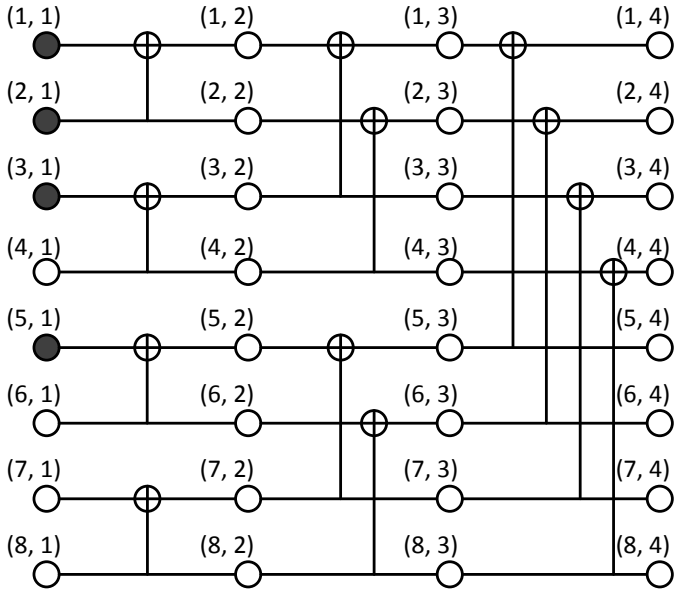


Figure 2.5: Conventional BP factor graph of $N = 8$ polar codes

are two messages passing through each node. The message propagated from right to left through node (i, j) is designated by $L_{i,j}$. The other message passed from the other direction is referred as $R_{i,j}$. Those messages are presented in the log-likelihood ratios (LLRs). Conventionally, those LLRs are updated through a series of check node processing elements (PE) as shown in Figure 2.6. The computations to update LLRs through iterations are written as follows:

$$\begin{aligned} L_{i,j} &= \mathcal{G}(L_{i,j+1}, L_{i+2^{j-1},j+1} + R_{i+2^{j-1},j}) \\ L_{i+2^{j-1},j} &= \mathcal{G}(R_{i,j}, L_{i,j+1}) + L_{i+2^{j-1},j+1} \end{aligned} \tag{2.12}$$

$$\begin{aligned} R_{i,j+1} &= \mathcal{G}(R_{i,j}, L_{i+2^{j-1},j+1} + R_{i+2^{j-1},j}) \\ R_{i+2^{j-1},j+1} &= \mathcal{G}(R_{i,j}, L_{i,j+1}) + R_{i+2^{j-1},j} \end{aligned} \tag{2.13}$$

where $\mathcal{G}(x, y) = \ln((1 + xy)/(x + y))$ is the propagation function to update mes-

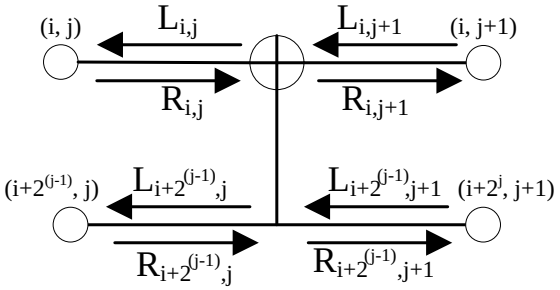


Figure 2.6: Processing element of conventional BP algorithm

sages[38]. In practice, the function $\mathcal{G}$ in Eq. (2.12) and (2.13) needs to be simplified by min-sum approximating $\mathcal{G}(x, y) \approx sign(x)sign(y)min(|x|, |y|)$ or scaled min-sum approximating $\mathcal{G}(x, y) \approx \alpha \cdot sign(x)sign(y)min(|x|, |y|)$, where α is the parameter scaling the $\mathcal{G}$ function.

The messages $L_{i,m+1}$ on the most right column are assigned by LLRs from the channel outputs. The messages $R_{i,1}$ on the first left column are the pre-decoding LLRs of $\hat{\mathbf{u}}$. Decoding starts by assigning ∞ and 0 to the frozen bits and information bits correspondingly. Those nodes on the most left column are also referred as leaf nodes in this paper. The BP decoding is performed by operating processing elements from left to right over and over to refine either $L_{i,1}$ or $R_{i,m+1}$ to estimate the transmitted message $\hat{\mathbf{u}}$ or transmitted codeword $\hat{\mathbf{x}}$ by:

$$LLR_i^{\hat{\mathbf{u}}} = L_{i,1} \tag{2.14}$$

$$LLR_i^{\hat{\mathbf{x}}} = R_{i,m+1} + L_{i,m+1} \tag{2.15}$$

where $LLR_i^{\hat{\mathbf{u}}}$ and $LLR_i^{\hat{\mathbf{x}}}$ are the log-likelihood ratios of the message $\mathbf{u}$ and the trans-

mitted codeword $\mathbf{x}$, respectively. They are defined as:

$$LLR_i^{\hat{\mathbf{u}}} = \ln \frac{P(\mathbf{y}|u_i=0)}{P(\mathbf{y}|u_i=1)}, LLR_i^{\hat{\mathbf{x}}} = \ln \frac{P(\mathbf{y}|x_i=0)}{P(\mathbf{y}|x_i=1)} \tag{2.16}$$

where $P(\mathbf{y}|x)$ represents the probability that $\mathbf{y}$ is received as x is given in the transmitter.

3. CONSTITUENT CODE PROPERTIES OF POLAR CODES

As mentioned in Chapter 2, the polar codes are encoded recursively through multiple coding stages. Thus, any polar code could be regarded as constituted by two shorter polar codes. For example, in the Figure. 2.5, the polar code of bits $\{(i,4)|i=1,2,...,8\}$ comprises the polar code of bits $\{(i,3)|i=1,2,3,4\}$ and the polar code of $\{(i,3)|i=5,6,7,8\}$ with one more stage polarization. And the polar code of bits $\{(i,3)|i=1,2,3,4\}$ and the polar code of $\{(i,3)|i=5,6,7,8\}$ further consist of shorter polar codes.

Those shorter polar codes which exist in the composition of a polar code are referred as the constituent codes. Some specific constituent codes are discovered in [30] to reduce the latency of SC decoding of polar codes. In this chapter, the constituent codes is briefly introduced. The exploitation of constituent codes is discussed in details in following for simplifying SC BP decoding algorithms.

Here, I present four types of specific constituent codes: all-frozen codes, all-information codes, single parity check codes and repetition codes. They are denoted as $\mathcal{N}^0, \mathcal{N}^1, \mathcal{N}^{SPC}$ and $\mathcal{N}^{REP}$ respectively. Noticeably, all the constituent codes are presented as the distribution of frozen and information bits. Although some throughput improvement could be achieved via constituent codes, which do not alter the location and properties of polar codes.

3.1 All-frozen $\mathcal{N}^0$ Codes and All-information $\mathcal{N}^1$ Codes

For any polar codes with rate of 0, the polar codes only contain frozen bits. It means that the polar codes are not necessary to be decoded, since there is no information bits in the polar codes. This type of polar codes are denoted as $\mathcal{N}^0$ codes in this dissertation. Similarly, if a polar code consists of only information bits

without frozen bits, it is referred as all-information polar codes, denoted as $\mathcal{N}^1$ codes.

3.2 Single Parity Check $\mathcal{N}^{SPC}$ Codes

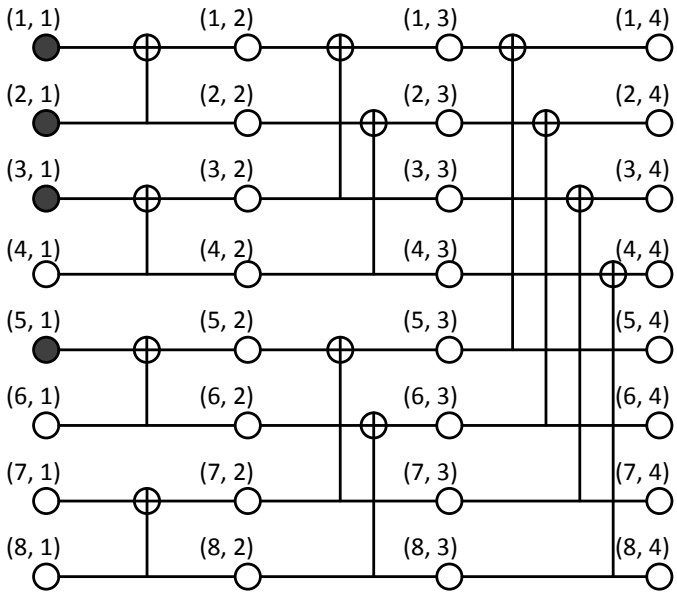


Figure 3.1: An example of $\mathcal{N}^{REP}$ codes and $\mathcal{N}^{SPC}$ codes.

For those polar codes whose rate is $(n-1)/n$ and frozen bit is always the first input bit, u_0. Figure 3.1 shows an example of a $(8, 4)$ polar code, where black nodes in the figure represent the frozen bits. Through the figure, one can observe that the polar constituent code of $\{(5,3),(6,3),(7,3),(8,3)\}$ is a single parity check codes. We denote the single parity check codes as $\mathcal{N}^{SPC}$ codes in the context. For an $\mathcal{N}^{SPC}$ code, it satisfies that

$$\sum_{i=1}^{n} x_i \pmod 2 = 0 \tag{3.1}$$

where x_i is the codewords of $\mathcal{N}^{SPC}$ codes, i.e.

$$x_1^n = \mathbf{G}u_1^n \tag{3.2}$$

For a hard decision decoder, the decoded bits for $\mathcal{N}^{SPC}$ codes could be represented as:

$$\hat{x}_i = \begin{cases} \hat{y}_i \oplus \text{parity} & if i = j \\ \hat{y}_i & otherwise \end{cases} \tag{3.3}$$

where $\hat{y}_i$ is the hard decision of received bits x_i transmitted through the channel and the parity is defined as:

$$parity = \sum_{i=1}^{n} \hat{y}_i \pmod 2 = 0 \tag{3.4}$$

For SC decoder, the specialized SPC decoders expedite the decoding process by utilizing the property of above. Large $\mathcal{N}^{SPC}$ codes are widespread in the relatively long polar codes. As observed by [30], higher-rate polar codes result in more $\mathcal{N}^{SPC}$ constituent codes in the structure. Also longer $\mathcal{N}^{SPC}$ codes are existing with the larger size of polar codes. It makes the constituent codes more attractive and practical, since polar codes can only achieve capacity with considerable long size. [15] shows that the throughput of successive cancellation decoders can be improved by $11 - 14\%$ by using constituent codes in decoding process.

3.3 Repetition $\mathcal{N}^{REP}$ Codes

For those polar codes with rate of $1/n$ and the only information bit u_n transmitted through the most reliable transformed channel, the codeword could be regarded as

a set of copies of the information bit u_n as:

$$x_i = u_n, \quad \text{for} \quad 1 \leq i \leq n \tag{3.5}$$

The type of those polar codes are referred as repetition codes with symbol $\mathcal{N}^{REP}$. The hard decision for such type of polar codes is done by a majority voter of the channel outputs. The only information bit is decoded as the 1 or 0 based on the majority of detected bits of 1s and 0s.

4. VLSI ARCHITECTURE FOR POLAR CODES DECODERS

Besides the appealing theoretic benefits of polar codes, more and more effort have been put on the exploration of the hardware implementation of polar codes decoders. In this section, we will review all existing work on hardware architecture for polar codes decoding.

4.1 Belief Propagation

One approach to decode the polar codes is belief propagation-based (BP) algorithm, which allows decoding in parallel to achieve much higher throughput in dedicated hardware implementation. Due to its higher computational demand, compared with SC algorithms, BP does not receive much attentions. To date, only a few works [10, 11, 31, 33, 35, 37] were reported on the theoretical analysis and hardware implementation.

BP algorithms achieve much higher throughput by utilizing the parallelism in the decoding process. However, compared with SC algorithm that only needs single-direction message passing, the BP algorithm requires propagate LLRs through the factor graph back and forth, which directly results in double consumption on memory. BP algorithms demand multiple round iterations to refine the estimation to achieve the decoding.

The first attempt at implementing BP on field programmable gate array (FPGA) is presented by Pamuk in [31], where the message passing functions are approximated by the min-sum (MS) algorithm for efficient hardware design. However, the performance of BP decoding is degraded because of the approximations. Thus, Yuan et al. explored scaled min-sum (SMS) approximation for message passing functions in [32] to remedy the performance penalty. However, compared with MS algorithm, SMS

incurs one extra scaling operations in each message passing. Yuan et al. further improved the efficiency of SMS BP decoders using early termination in [33]. On the other hand, by removing unnecessary computations for frozen bits in polar codes, Zhang et al. reduce the complexity for sum-product (SP) BP decoding in [34] by around 25% without decoding performance degradation.

There are two common architectures for the BP decoding algorithm. They are array-based architecture and line-based architecture. The basic principles of these two architectures are discussed in following.

4.1.1 Array Architecture

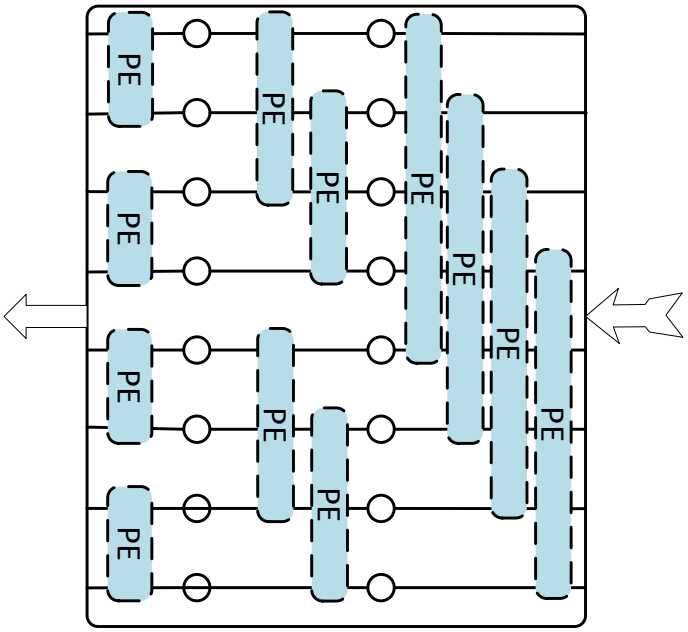


Figure 4.1: The architecture for an array based BP decoder for a $n = 8, rate = 0.5$ polar code

First hardware structure to accomplish computations in BP decoding algorithm

is to directly implement the factor graph in the hardware. Computations in each node are implemented in a single processing element, as shown in Figure 4.1. The LLR from channels are input from right. Belief are updated through the PE networks layer by layer.

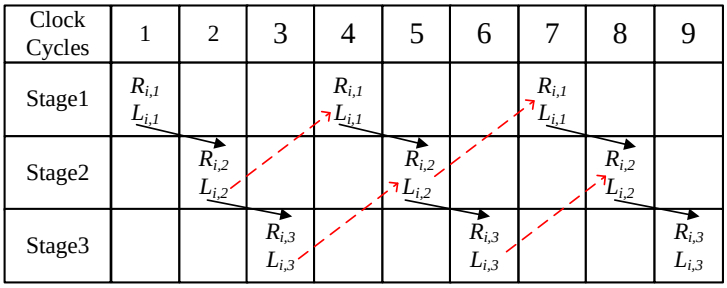


Figure 4.2: The scheduling for an array based BP decoder for a $n = 8, rate = 0.5$ polar code

However, the efficiency of this architecture is highly restricted by the data dependencies. As aforementioned, the belief information has to be updated stage by stage. As Figure 4.2 shows, hardware resources have to left in idle until the previous stage computations finished.

Yuan et al. improved the efficiency of this structure by introducing the folding scheduling technique [33], where the next iterations are designed to be triggered as soon as possible, as shown in Figure 4.3.

4.1.2 Line Architecture

An alternative structure to implement the computations is the line-based architecture. In this architecture, only one stage of processing elements are implemented in hardware.

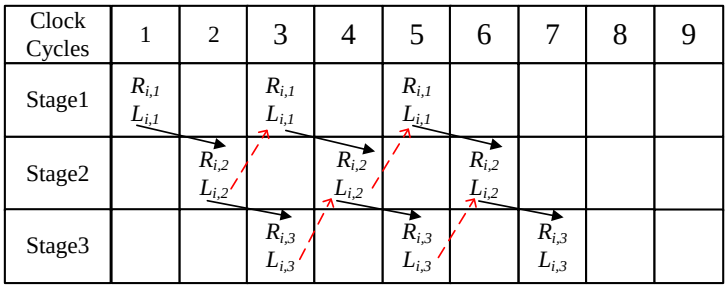


Figure 4.3: The folding scheduling technique for a line based BP decoder for a $n = 8, rate = 0.5$ polar code

Figure 4.4 shows the structure of the line architecture of the BP decoder, where processing elements of only one stage is synthesized. By doing this, the computations efficiency is highly improved. The hardware are always kept busy during the decoding process. However, additional hardware resources such as the interleaver, additional memory are consumed to arrange the intermediate temporary data.

4.2 Successive Cancellation

Different with BP decoders, SC decoder is attractive for its low computational complexity of $\mathcal{O}(NlogN)$, where N is the length of the code. Thus, many relevant hardware designs are proposed [39–41].

However, algorithmically, SC decoder suffers from high latency. Typically, for conventional SC decoder, its latency ($2N2$) increases linearly with respect to the code length. This is a significant challenge since polar codes work well only at very long code lengths. A lot of works have been done to reduce the latency of SC decoder from both hardware and algorithm aspects. In [42], a pre-computation method is used to reduce decoding latency from $2N2$ to $N1$. In [29], three approaches, the dedicated 2-bit decoder for the last stage of SC decoding, overlapped-scheduling and lookahead techniques are applied, which eventually results in a $3N/41$ latency. In [15, 30], by

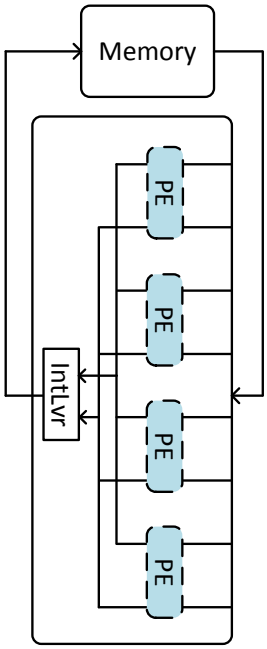


Figure 4.4: The architecture for a line based BP decoder for a $n = 8, rate = 0.5$ polar code

observing the tree architecture of SC decoding, certain patterns of constituent codes are found. These constituent codes can feed back the hard decision information immediately without traversal, which can significantly reduce the latency of decoding some polar codes with a given architecture. This approach is refer to as fastSSC decoder. Moreover, a processors-array based structure for FPGA implementation is also proposed in [30].

As introduced in [40], tree architecture or line architecture for SC decoder is the most common. Line architecture has a higher hardware utilization but needs increased complexity in control module and memory access. Figure (TODO: make figures show tree and line architecture) shows Processing unit (PU) performs the f and g functions in Eq. (1) and Eq. (2), respectively, and its arithmetic part is used to decode N SPC and N REP as well.

[43] presents an architecture to unify the computations required for both conventional SC and special constituent codes.

4.3 List Successive Cancellation

Decoding performances of both SC and BP are inferior to that of low density parity check (LDPC) codes. In order to make polar codes more competitive, the list SC (LSC) decoding algorithm is presented in [14]. By exploiting a larger range in the codeword tree, LSC significantly improves the decoding performance.

Attracted by the potentials of LSC, a number of relevant hardware designs have been explored. In [44], hardware LSC architectures of list sizes two and four are proposed with pointer memory technique, which can avoid the high complexity of likelihood copying. In [45], a hardware efficient architecture of LSC concatenated with cyclic redundancy check (CRC) is presented. In [46], a hardware architecture of sub-optimal version of LSC decoding is introduced. In [47], a LSC with multi-bit decision is discussed, which significantly reduces the decoding latency, and the corresponding hardware architecture is presented. All of aforementioned designs are using l duplicates of SC decoder for LSC decoder with list size l. Consequently, compared with SC decoder, the complexity of LSC increases from nlogn to $ln \log n$, where n and l are the length of codeword and list size, respectively. However, such complexity increasing makes all current existing LSC architectures are impractical for decoders with large list size.

For the LSC algorithm, every information bit can derive two candidate paths, which are used to represent the decision of bit as 0 or 1. Each path has its own path metric which is corresponding to its survival probability. When performing the LSC decoding, l paths are expanded to $2l$ paths for each estimated information bits. Then the metrics of $2l$ paths are calculated to decide the l survivals. All the corresponding

inner log likelihood ratios (LLRs) and partial sum of the reserved paths need to be kept along with l paths as well. Finally, the l paths are fed back to SC decoders and do all the steps again and again until the last information bit is decoded. Although all the LSC designs mentioned above have differences at some details, the main architecture are similar. Typically, for a LSC decoder, it has l copies of SC decoders and one metrics computation units (MCU), one sorting module and three memory banks with respect to path metrics, current survival paths and LLRs and partial sums. The SC decoder consists of multiple processing units (PUs) with a tree architecture which consumes most of hardware resources. Such duplicates of SC decoder yield a significant hardware redundancy of LSC decoder design. An efficient scheduling method is proposed in [48] to overlap decoding processes for two consequent paths, which results in a dramatic reduction of the hardware complexity without any decoding performance loss. Also multiple-bits decisions technique is applied to reduce the latency associated with the pipeline scheme. Simulation results show that with proposed design approach the hardware efficiency is increased significantly over other state-of-the-art LSC decoders.

5. THROUGHPUT CENTRIC SUCCESSIVE CANCELLATION DECODER

Polar codes can be decoded by recursively applying successive cancellation to estimate ui using the channel output y_0^{N-1} and the previously estimated bits u_0^{i-1}. This approach is naturally represented by a binary tree whose each node corresponds to a constituent code. The number of bits in one constituent node in stage $m(m = 0, 1, 2...)$ is equal to 2^m.

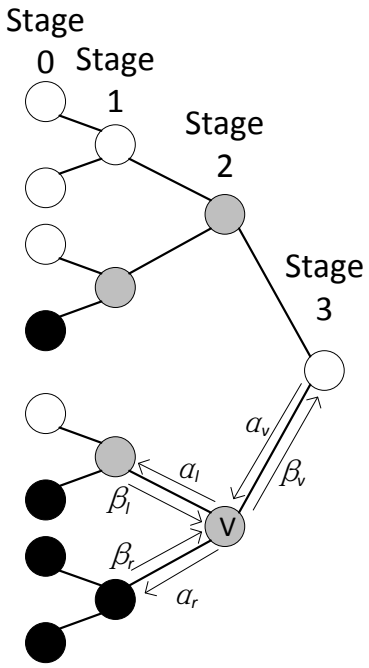


Figure 5.1: Tree presentation of SC decoding processes for a $(8, 4)$ polar code.

Figure 5.1 shows an example of (8, 4) polar code. α stands for the soft reliability value, typically is log-likelihood ratio (LLR), and β stands for the hard decision. α_l and α_r are the message passing from parent node to left and right child, and can be

computed by the following equations:

$$\alpha_l[i] = sign(\alpha_v[i])sign(\alpha_v[i + N^m/2]) \cdot \min(|\alpha_v[i]|, |\alpha_v[i + N^m/2]|) \tag{5.1}$$

$$\alpha_r[i] = (-1)^{\beta_l[i-N^m/2]} \cdot \alpha_v[i - N^m/2] + \alpha_v[i] \tag{5.2}$$

At stage 0, β_v of a frozen node is always zero, and for information bit its value is calculated by threshold detection of the soft reliability according to

$$\beta_v = \begin{cases} 0 & \text{if } \alpha_v \geq 0 \\ 1 & \text{otherwise} \end{cases} \tag{5.3}$$

At intermediate stages, β_v can be recursively calculated by

$$\beta_v[i] = \begin{cases} \beta_l[i] \oplus \beta_r[i] & \text{if } i \leq N^m/2 \\ \beta_r[i - N^m/2] & \text{otherwise} \end{cases} \tag{5.4}$$

5.1 Fast SC Decoder

The main idea of fast-SSC algorithm is illustrated in [15, 30, 42]. By identifying certain pattern of constituent polar codes, the hard decision β_v of each constituent node can be determined immediately, without traversing the entire subtree. Such arrangement significantly reduces the decoding latency. For a length N constituent code in non-systematic polar codes, $\hat{u}_N$ is calculated by

$$\hat{u}_N = \beta_{vN} \cdot G_N \tag{5.5}$$

where G_N is the generator matrix for length N polar code. There are $\mathcal{N}^0$, $\mathcal{N}^1$, $\mathcal{N}^{SPC}$ and $mathcal N^{REP}$ codes which can be utilized to expedite the estimation of β. $\mathcal{N}^0$ and $\mathcal{N}^1$ codes are referred to those constituent codes which only contain frozen bits or information bits, respectively. For $\mathcal{N}^0$ codes, we can set β_v to 0 immediately. For $\mathcal{N}^1$ node, β_v can be directly determined via threshold detection Equation 5.3. $\mathcal{N}^{SPC}$ and $\mathcal{N}^{REP}$ are two kinds constituent codes containing both frozen bits and information bits. In a length N $\mathcal{N}^{SPC}$ codes, only the first bit is frozen. It renders the constituent codes as a rate $(N1)/N$ single parity check (SPC) code. This code can be decoded by performing parity check with the least reliable bit which has the minimum absolute value of LLR. First, get the hard decision HD_v of β_v via threshold detection. Then, calculated the parity by

$$\text{parity} = \sum_{i=1}^{N^m} \oplus HD_v[i] \tag{5.6}$$

and, find the index of the least reliable bit

$$j = arg \min_i |\alpha_v[i]| \tag{5.7}$$

Eventually, β_v is decided by

$$\beta_v[i] = \begin{cases} HD_v[i] \oplus \text{parity} & \text{when } i = j \\ HD_v[i] & \text{otherwise} \end{cases} \tag{5.8}$$

In a length N $\mathcal{N}^{REP}$ codes, only the last bit is information bit. In this case, all the $\beta_v[i]$ should be the same and are duplicates of the information contained in the only one information bit. Thus, the decoding algorithm starts by summing all input

LLRs and β_v is calculated as

$$\beta_v[i] = \begin{cases} 0 & \text{when } \sum \alpha_v[i] \geq 0 \\ 1 & \text{otherwise} \end{cases} \tag{5.9}$$

Figure 5.2 gives the examples of tree presentations of these four kinds constituent

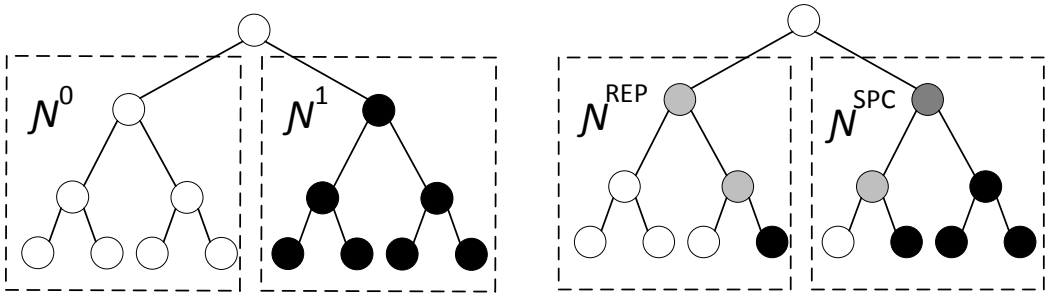


Figure 5.2: (a) An example of $\mathcal{N}^0$ and $\mathcal{N}^1$ codes in a (8, 4) polar code tree, and (b) an example of $\mathcal{N}^{REP}$ and $\mathcal{N}^{SPC}$ in a (8, 4) polar code tree.

polar codes. These four types of constituent polar codes are adopted in the SC decoder to expedite the processing, therefore which is referred as fast SC decoder.

5.2 VLSI Architecture

To fully utilize the advantage of constituent codes in the polar codes tree, a novel hardware implementation of fastSSC decoder is presented in [43]. For a polar code with a given length, different code rate yields different distribution of constituent polar codes. A thoughtfully-composed architecture should have the capability and flexibility to deal with different rates. Thus, by exploiting the homogeneousness between the decoding processes of fast constituent polar codes and regular polar

codes, our design supports a variety of rates. Additionally, an approach is proposed in this section for sharing and reusing computational elements to achieve higher hardware efficiency.

As introduced in [39], tree architecture or line architecture for SC decoder is the most common. Line architecture has a higher hardware utilization but needs increased complexity in control module and memory access. Tree architecture is employed in the TCSC decoder for control simplification.

Figure 5.3 shows an overview of proposed system when code length = 16. P rocessing unit(PU) performs the functions in Equations 5.1 and 5.2, respectively, and its arithmetic part is used to decode $\mathcal{N}^{SPC}$ and $\mathcal{N}^{REP}$ as well. Pre-computation technique is also used, which allows the f and g functions update in the same clock cycle. The PU used in stage 0 has a slight difference with ordinary PU. We denote it with PU_0 in the figure. According to Equation 5.7, the minimum LLR value needs to be found. The comparator tree is used to perform this since it inherently exists in the tree architecture of PUs. A judicious scheduling permits obtaining the minimum value at stage 0 and recording the choice of smaller input for each PU at each stage. After that, a backward operation implemented by a series of parity transmit unit (PTU) can help to locate the minimum one among the length N $\mathcal{N}^{SPC}$ polar codes. Design details are illustrated in following. The estimation of current bit in SC decoding is based on the information of previous decoded bits (β). This information is also called partial sum. Thus, a partial sum generator (PSG) which can co-operate with decoding pipeline is also needed.

5.3 Dataflow, Latency and Flexibility Analysis

In terms of tree presentation, SC decoder conventionally process one node in each clock cycle. Traversal of a subtree contained N leaf nodes needs $2N2$ clock cycles.

By using pre-computation as introduced in [42], which calculates the functions 5.1 and 5.2 in the same clock cycle, the latency can be reduced to $N1$. In TCSC design, if this subtree is belong to fast constituent polar codes, the latency can be further reduced.

For $\mathcal{N}^0$ codes, the β_v are all set to 0. For $\mathcal{N}^1$, the the β_v are determined by hard decision of input LLRs. Both of the two computations need only one clock cycle after they are activated. For $\mathcal{N}^{SPC}$ codes , according to Equations 5.6, 5.7 and 5.8, only three operations needed. Finding the minimum LLR can be done by a comparator tree, which naturally exists in SC decoder with tree architecture since every PU has a comparator for Equation 5.1. For N LLRs, finding the smallest one takes $\log(2N)$ clock cycles. Meanwhile, we can obtain the parity bit when the minimum LLR is found, which will be explained in the next subsection. After that, one more clock cycle is need for signal parity check which is done by a XOR gate. Thus, totally, decoding a length N $\mathcal{N}^{SPC}$ constituent polar codes need $\log(2N) + 1$ clock cycles. For $\mathcal{N}^{REP}$ codes, according to Equation 5.9, an accumulation operation is needed. Similar to the comparator tree, an adder tree also exists in SC decoder within the tree architecture since every PU has an adder for Equation 5.2. For a length N $\mathcal{N}^{REP}$ constituent polar code, it needs $\log(2N)$ clock cycles to decode.

To briefly sum up, $\mathcal{N}^0$ and $\mathcal{N}^1$ have time complexity $\mathcal{O}(1)$ and $\mathcal{N}^{SPC}$ and $\mathcal{N}^{REP}$ have time complexity $\mathcal{O}(log2N)$. Compared with commonly discussed SC architecture in [5], [7] and [8], which all have linear time complexity $\mathcal{O}(N)$, the latency can be significantly reduced from the TCSC micro-architecture and scheduling with very large codelength N. The details of micro-architecture and scheduling are given in the following.

5.4 Unified Computational Unit

Figure 5.4 shows design details of PU. A single PU can perform Equations 5.1 and 5.2. Also a PU tree can help to find the minimum values or do accumulation for multiple inputs. In the figure, S stands for signed magnitude number and C stands for 2's complement number. Unlike the PU design in [29], in which data are initially stored as signed magnitude form, our design use 2's complement as initial form. We do this for two reasons.

1. According to synthesis result, the critical path of PU is along with the Equation 5.2 path. By moving number system convert modules to the Equation 5.1 path, which means using 2's complement as initial data form, the critical path is still along with g function path, but delay on critical path is significantly reduced.

2. Compared with four number system convert modules are used in [29], only three are used if use 2's complement number. This is more hardware efficient. The benefits of this modification can be seen in later.

Table 5.1: Truth table of PTU

PCB	SS	O1	O2	PCB	SS	O1	O2
0	0	0	0	1	0	1	0
0	1	0	0	1	0	0	1

For each PU, two LLRs are fed simultaneously. Since we use the pre-computation technique, equations 5.1 and 5.2 are calculated at the same time, and which one needs to be output is determined by *modeselect*2. According to Equation 5.2, there are

only two types of possible results, sum or difference. Its final result depends on the corresponding partial sum. So two registers are used here to hold the most recently computed values until the corresponding partial sum is calculated. When it calculates the sum for decoding $\mathcal{N}^{REP}$, only additions are needed. The datapath is decided by $Modeselect1$ signal. When f function is performed, according to Equations 5.1, both 2 inputs are divided into two parts: sign bit and unsigned number. Each part is processed separately first, and results of two parts are combined together to obtain the updated value. C to S and S to C modules are needed before and after comparisons, respectively. When it deals with $\mathcal{N}^{SPC}$, the result of comparison should be recorded using a register as the select signal for PTU. Since the processing of searching minimum value lasts several clock cycles, there should be a feedback of the register to hold this value for the later clock cycles. The input source is chosen by $Modeselect3$ signal. Since every PU does exclusive-or operation to the sign bit of two inputs, according to Equations5.8, the sign bit of the final value in stage 0 should be equal to the parity. Equation5.6 can be performed using an XOR gate. The PU that contains the minimum LLR receives the parity check bit and the others receive 0's. The transmission of parity check bit is done by the PTU which is a two-input two-output module. One input is the parity check bit(PCB) and the other is the select signal (SS). The parity check bit is transmitted via *output* 1 (O1) or *output* 2 (O2) based on the values of SS. Table 5.1 shows the truth table of PTU. We can obtain the logic expression of O1 and O2 as: $O1 = PCB \,\&\, SS$, $O2 = PCB \,\&\, SS$. This can be done by two *and* gates and one $Inverter$.

The PU in $stage0$, as shown in Figure 5.3, has a simpler architecture compared with PU. Figure 5.5 shows the design details of PU_0. Since only one more clock cycle need for single parity check, there is no feed back to this register. Furthermore, $\mathcal{N}^{SPC}$ cannot exist in stage0. So top part in Figure 5.4, which is relative to single parity

Table 5.2: Hardware comparison of different (n, k) SC decoder with q-bit quantization for inner LLRs using tree architecture

Hardware Type	[42]	[39]	[29]	Proposed Design
# of PU	$n-1$	$n-1$	$n-1$	$n-1$
# of PTU	0	0	0	$2/n-1$
# of 1 bit REG	$\approx 3qn$	$\approx qn$	$\approx 3qn$	$\approx (3q+1)n$
HC	1.3	1	1.3	1.31
Latency (clock cycle)	$n-1$	$2n-2$	$0.75n-1$	$\approx (0.1 \sim 0.3)n$
Throughput	2	1	2.67	$\approx 6.69 \sim 22.26$
Throughput/HC	1.53	1	1.74	$5.1 \sim 16.99$

check can be removed. For Equation 5.2 and $\mathcal{N}^{REP}$, the output of Equations 5.1 can be fed back to it immediately, and the sign bit of the result of adding is the partial sum for $\mathcal{N}^{SPC}$.

5.5 Hardware Performance

5.5.1 Fixed Point Analysis

Before the discussion on hardware consumption, we quantized the simulation to figure out how many bits need to be assigned to make sure the reliability of hardware implementation. Figure 5.6 shows the effect of quantization on the (1024, 512) polar code. For channel outputs and inner LLRs, we use separate quantization schemes. The quantization schemes are shown in (C, L, F) format, where C, L and F are the number of bits used for presenting channel output, inner LLRs and fraction parts of both channel output and LLRs, respectively. As the result of the trade-off between hardware efficiency and decoding performance, (4, 5, 0) quantization scheme is chosen for TCSC design.

5.5.2 Hardware Comparison with Other State-of-the-art SC Decoders

Table. 5.2 shows the hardware comparisons between proposed design and other state-of-the-art designs. All the candidates are (n, k) SC decoder with tree architectures, and they all use q-bit quantization for inner LLRs. All the throughputs and hardware complexity (HC) are normalized to the SC decoder in [39], and the hardware complexity is estimated based on the synthesis results. The latency for proposed design is a range with respect to the code rates change from 0.05 to 0.95. From this table, we can see that our TCSC design achieves the highest throughput per unit of hardware complexity. The exact latency depends on the code rate. Figure. 5.7 shows the latency reduction of the proposed design along with code rates from 0.05 to 0.95. The reduction is relative to the 2b-SC-Precomputation decoder which so far is known to be the fastest. The figure shows at least 60% latency reduction can be achieved by our proposed design. This is very promising for many applications where high rate channel codes are needed, such as for data storage system.

Table 5.3: Synthesis result for $(1024, 870)$ and $(1024, 512)$ polar codes

Silicon Area (μm^2)	275899
Max Frequency (GHz)	1.04
Latency (1024,870) (clock cycle)	156
Throughtput(1024,870) (Gbps)	5.81
Latency (1024,512) (clock cycle)	266
Throughtput(1024,512) (Gbps)	2.01

Additionally, we implemented the proposed design with *Verilog* for the polar code with length=1024 and synthesized it using *Nangate FreePDK* 45*nm* process with *Synopsys Design Complier*. We calculated the throughput for $(1024, 870)$

and $(1024, 512)$ polar codes. Table 5.3 shows the synthesis result for $(1024, 870)$ and $(1024, 512)$ polar codes. Notice that the maximum frequency is higher than that reported in [29] which use the same process as our design. TCSC design in theory should have a lower maximum frequency since it has one more Mux delay for regular and fast constituent polar codes. However, thanks to the introduction of fast constituent codes in the decoding process, the number of cycles dedicated for each decoding is reduced so as that the hardware performance is improved.

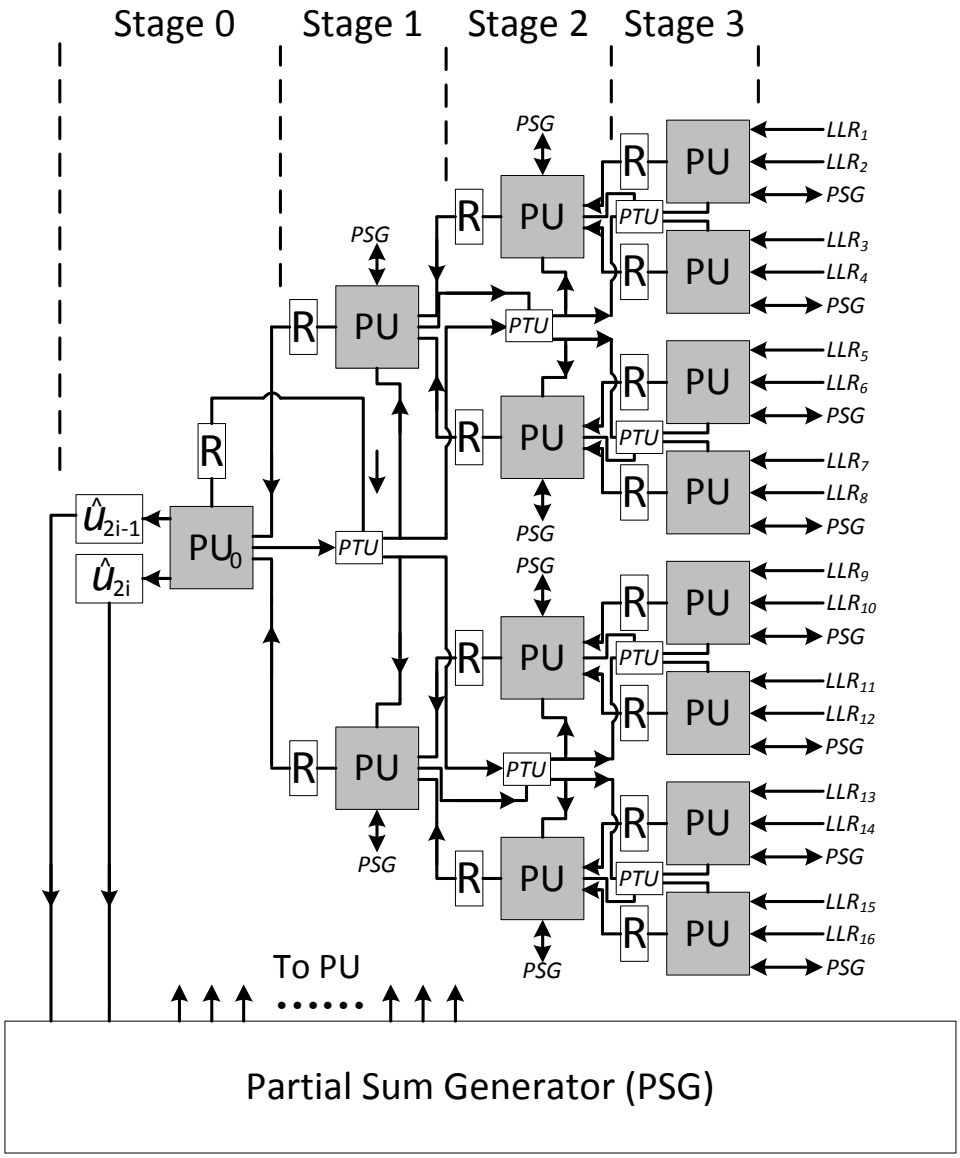


Figure 5.3: Overview architecture of TCSC decoder for a $(8, 4)$ polar code.

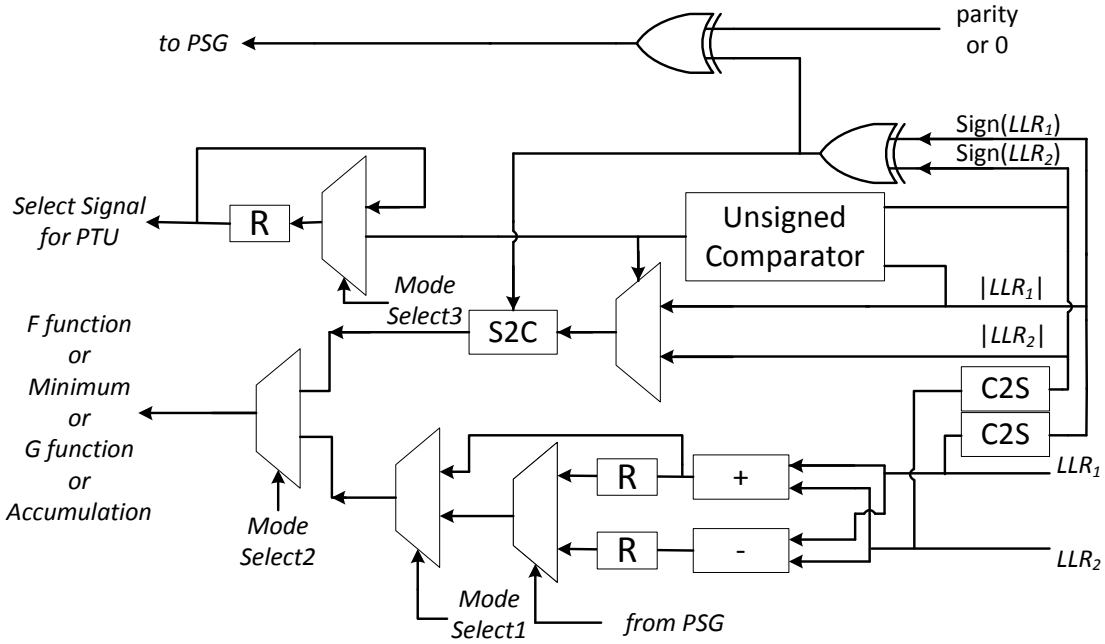


Figure 5.4: The design details of PU in TCSC decoder.

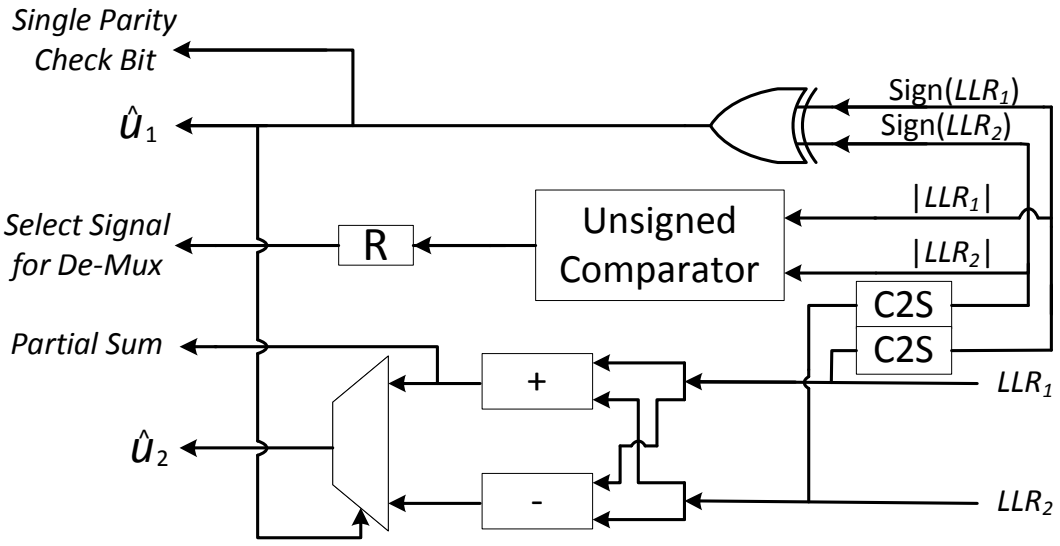


Figure 5.5: The design details of PU0 in TCSC decoder.

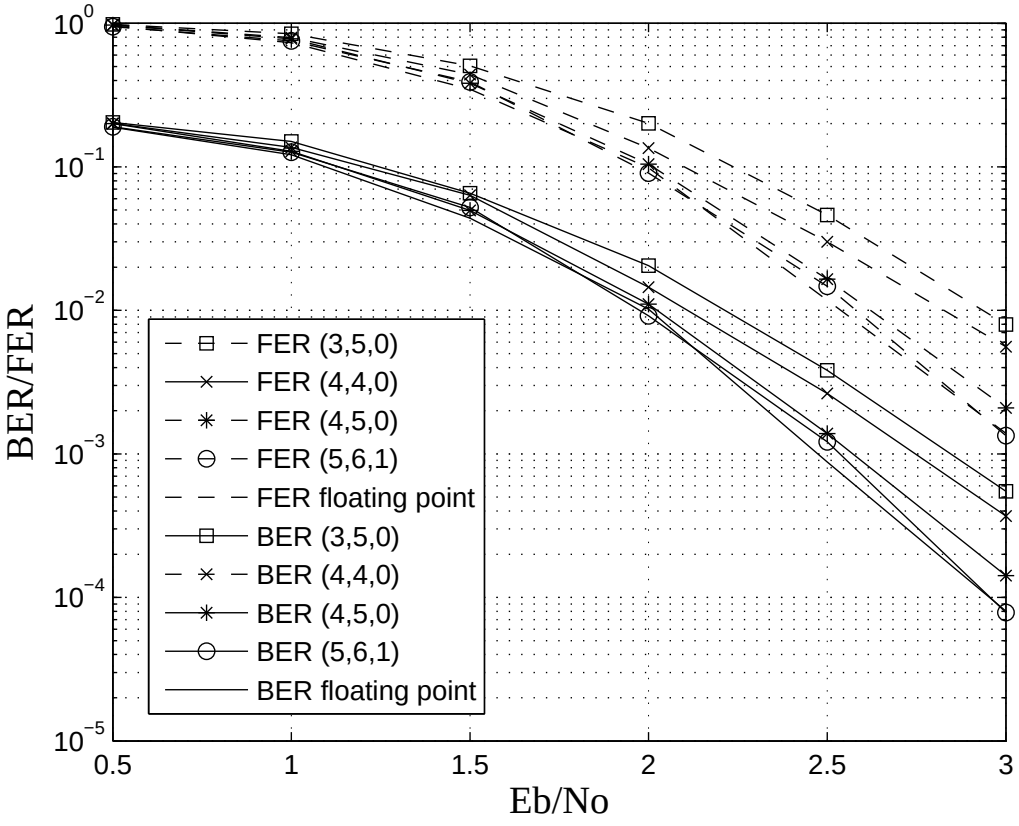


Figure 5.6: Effect of quantization on the BER/FER performance of a (1024, 512) polar code.

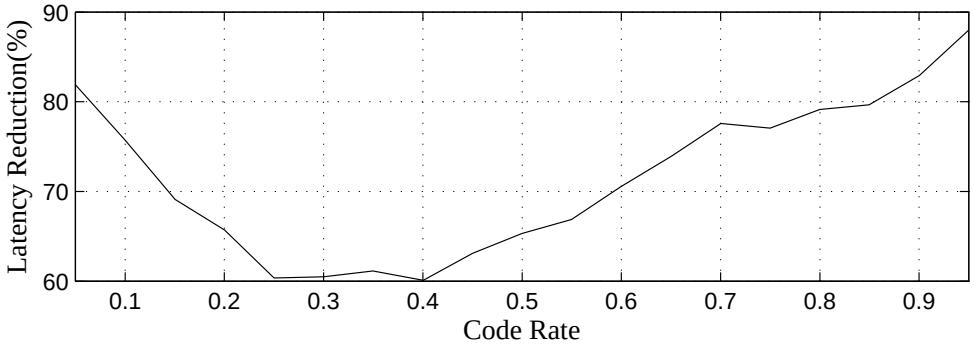


Figure 5.7: The trend of latency reductions on code rates.

6. OVERLAPPING-PATH LIST SUCCESSIVE CANCELLATION DECODER

In order to make polar codes more competitive, the list SC (LSC) decoding algorithm is presented in [14]. As aforementioned, by overlapping the decoding paths on a single tree and applying the multiple-bit decision, the overlapping-path list successive cancellation decoder significantly improve the LSC decoder efficiency.

Following the details of the overlapping-path list successive cancellation (OPLSC) is presented.

6.1 VLSI Architecture

Figure 6.1 shows the architecture of proposed approach and the example of the modified architecture of SC decoders associated with the list sizes two and four. Since the duplicates of SC decoder involve the most hardware complexity, only one instance of SC decoder is left in the architecture.

In the OPLSC, every path is computed simultaneously in the decoding threads by judiciously utilizing the decoder hardware as follows: The computation of each path is overlapped with others' in the pipeline arrangement. The architecture of SC decoder is modified to support this paradigm. Since modifications are made only on architecture and scheduling plan, no decoding performance gain loss or change is incurred. The sorting module, metrics computation units (MCU), and related memory components are compatible with other LSC decoders, and the partial sum generator is scheduled in a similar way to be compatible with the path-overlapping SC decoder. In the following sections, the details of the scheme and the specific LSC decoder are discussed.

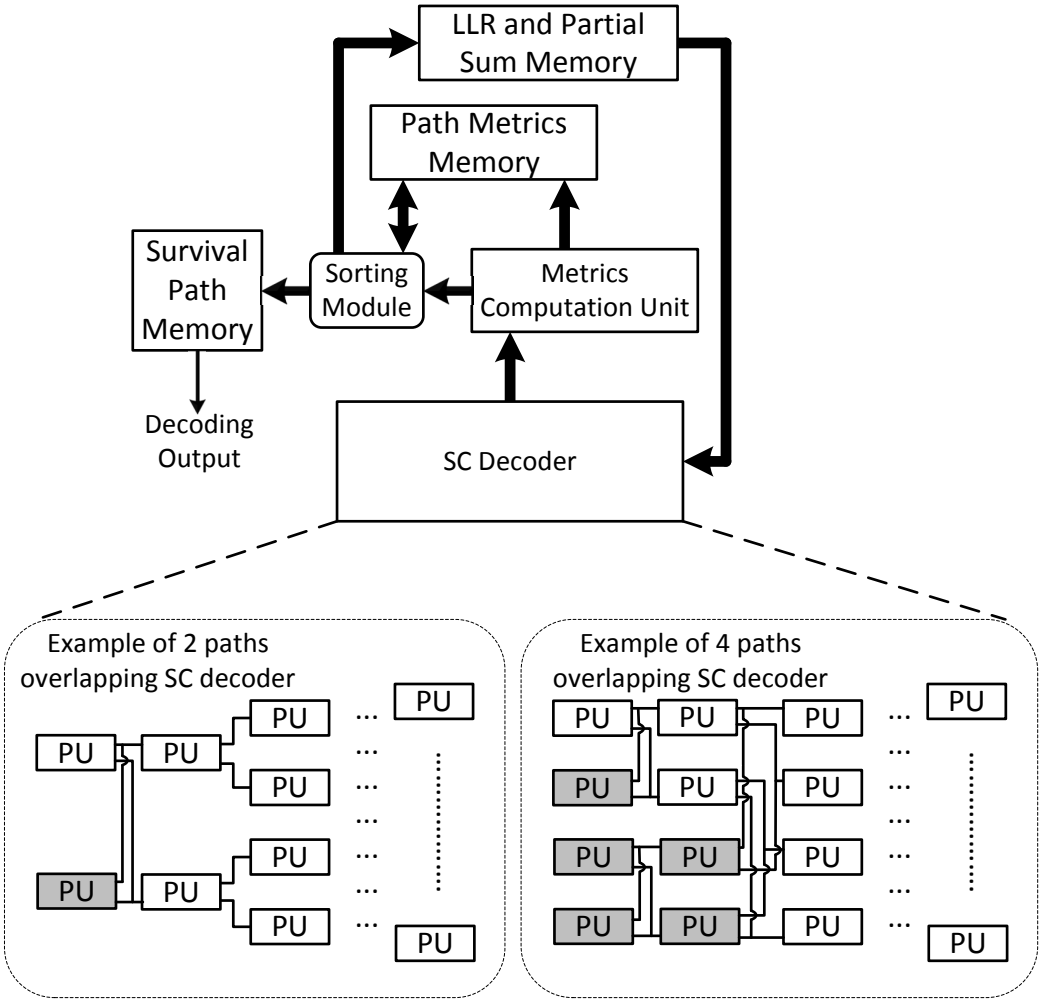
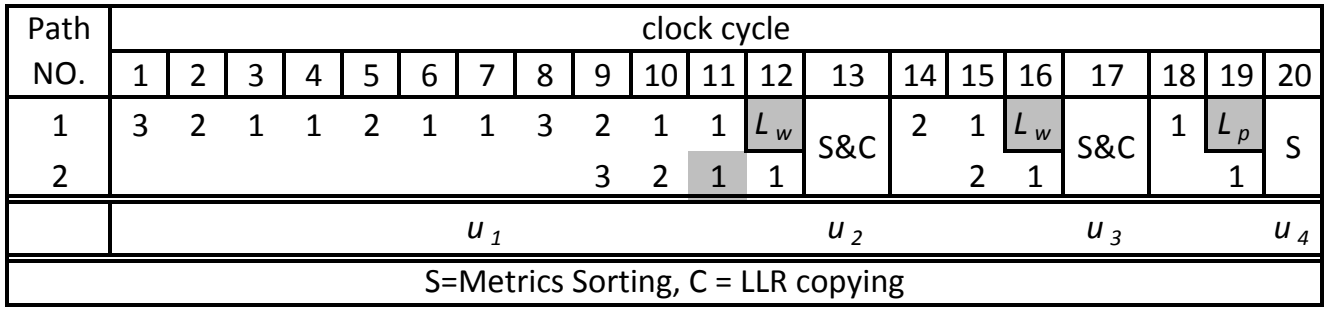


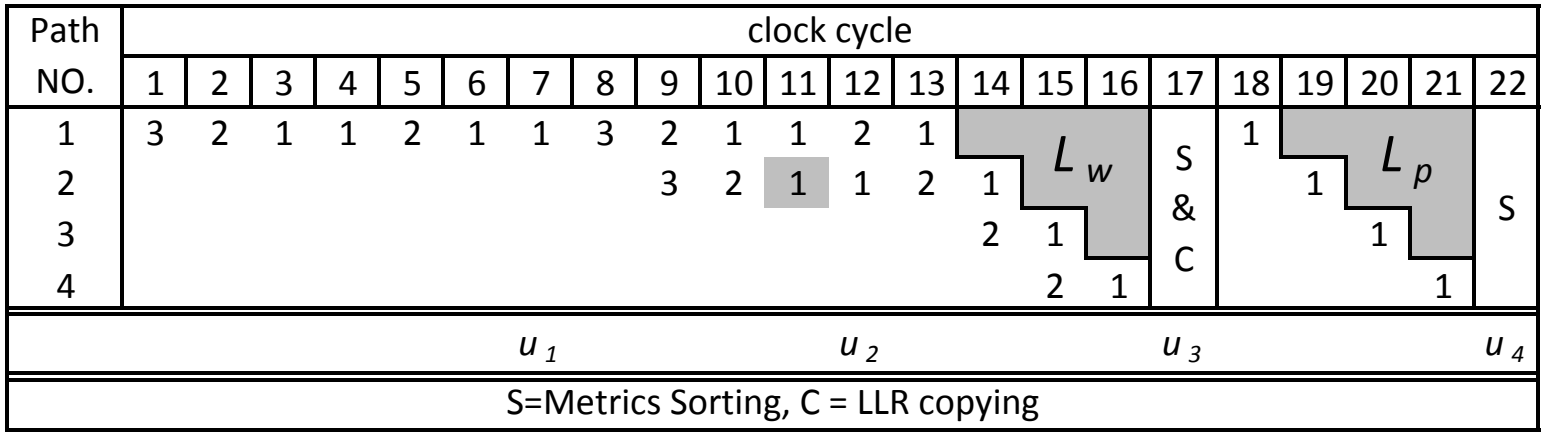
Figure 6.1: The overall architecture of overlapping-path list successive cancellation decoder.

6.2 Path-Overlapping Scheme

Simultaneous processing approach is already presented in some SC decoders, where multiple frames are computed in an overlapped fashion so as to increase the throughput [49]. The SC decoder with tree architecture consists of multiple processing unites (PU) arranged like a binary tree. For every clock cycle, only one stage of PUs in the tree is activated. The basic idea of simultaneous processing approach is



(a)



(b)

Figure 6.2: Decoding schedule of the path-overlapping scheme for 2 lists (a) and 4 lists (b).

to activate multiple decoding stages in one clock cycle by feeding in several frames in pipeline. This means that each frame comes into the decoder with one clock cycle delay.

Similar as the idea for SC decoder, it is found that the duplicates of SC decoder in conventional LSC decoder is unnecessary, since all the paths can be fed into the same decoder in pipelined fashion. Different stages in the single SC decoder can process different paths simultaneously. Computations of successive paths are overlapped in temporal with only one clock cycle delay. However, the decoding scheme is not exactly the same as multiple frames overlapping SC decoder. Figure 6.2a and 6.2b show the decoding schedules of two and four path-overlapping scheme, respectively. The number means current activated stages, and the duplicated stage is marked with gray. If a SC decoder is with l path-overlapping scheme, where $l(2^i1)$, it can be constructed by duplicating $(2^{i1}1)$ stages, where the index starts from the information bits side with respect to the tree architecture. The duplication plan is also presented in Figure 6.1. Noticeably in Figure 6.2b there is only one duplication of stage one, which is not the same as what presented in Figure 6.1. This is because the number of copies in Figure 6.1 are the minimum requirement for all the case. The actual requirement is decided by the code length and rate. Figure 6.2b is just a specific case where only one stage duplication is needed for four path-overlapping scheme.

Such architecture significantly reduces hardware complexity. Another advantage of OPLSC is that it reduces the critical path length of decoder. Usually, the critical path lies in the sorting block. For a conventional LSC decoder, the sorting block is composed of staged combination logic. Even for very small list size, e. g. list $= 4$, the critical path is much longer than any other module. With proposed approach, since each path metrics comes with pipeline arrangement, naturally, the sorting block is designed as a pipeline module which has a shorter critical path than that of com-

bination logic for the same list size. This means, by applying proposed approach, OPLSC decoder can run at a much higher frequency.

Although proposed approach can achieve a higher frequency compared with the conventional LSC decoder, there are some additional clock cycles introduced. These consist of two parts. The first part is the path pipeline latency L_p. Since all paths are fed into decoder with one clock cycle delay, for the OPLSC with list size l, $L_p = (l-1)$. The second part is path waiting latency L_w. After the number of path extending to the maximum, the pipeline processing has to suspend when estimating the newly generated information bit since the decoder needs to wait for all the paths to finish before commencing metric sorting and LLR copying. This waiting period is referred to as pipeline stalling. The waiting time is equal to L_p. Thus, for the list size l LSC with respect to (n, k) polar code, $L_w = (k - \log_2 l - 1) \cdot (l-1)$. Thus, the total latency overhead introduced by path-overlapping scheme L_m can be calculated by:

$$L_m = L_w + L_p = (k - \log_2 l) \cdot (l-1). \tag{6.1}$$

Although the hardware complexity for LSC is significantly reduced by OPLSC, and it incurs few additional clock cycles to achieve the improvement. Thus, it is difficult to evaluate such design approach merely in term of the usage of hardware resource or the latency. Thus we introduce the hardware efficiency (HE) metric which is noted as e to measure the performance of proposed approach[50]. The e is defined as: $e = Throughput/Area$.

From Eq. (6.1), it is shown that the latency overhead would significantly aggregate with either list size or code rate, which can significantly diminish the e. In order to achieve a high e with OPLSC architecture, the latency overhead must be reduced to an acceptable level. In the next sections, we will present three approaches aimed at

decreasing the latency overhead.

6.2.1 Latency Reduction via Multi-Decision List SC Decoding

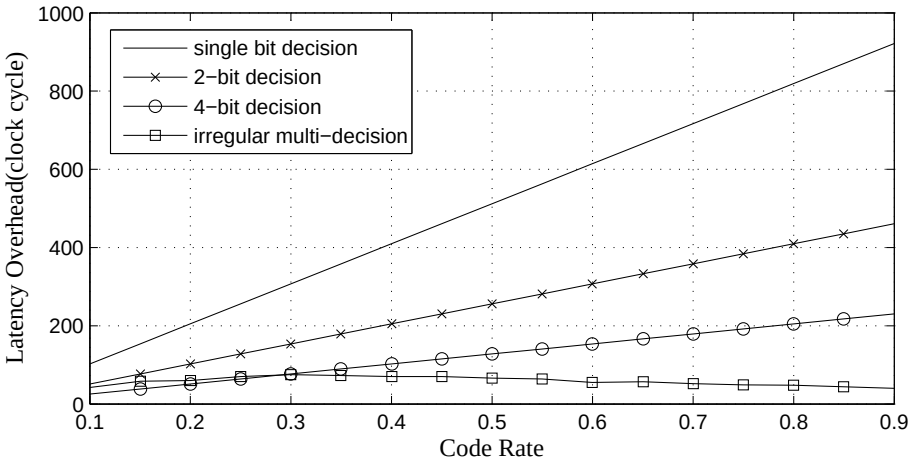


Figure 6.3: The total latency overhead versus the polar codes rates.

The first part of Equation (6.1) corresponds to the path waiting latency. For every instance of estimating an information bit, the pipeline processing has to suspend until all the paths finish calculations. This provides an observation that if the times of estimating the information bit can be reduced, the L_w will decrease significantly.

Multi-decision is an approach of estimating m bits ($m > 1$) instead of just one at the same time. It helps to reduce the number of estimations. Many approaches can be regarded as multi-decision [43, 47, 51, 52]. Generally, they can be classified into two types. The first type is referred to as regular mutil-decision decoder; it estimates m bits ($m > 0$) every time. Most of current multi-decision decoders belong to this type [47,51]. The second type is called irregular mutil-decision decoders; the number of bits estimated every time is not fixed. Currently, only the list fast-SSC decoder [52]

belong to this type. It simplifies the SC decoding by utilizing constituent codes. The number of bits estimated every time is corresponding to the size of constituent code. Besides, the distribution of constituent codes irregularly change along with code rate.

For path-overlapping LSC decoder with mutil-decision, L_m can be further reduced to $L_m = \alpha \cdot (l-1)$. For m bits regular mutil-decision, $\alpha = \lceil (k - \log_2 l)/m \rceil$. For irregular mutil-decision, $\alpha = S - \log_2 l$ where S is the total number of constituent codes which irregularly changes along with code rate. Figure 6.3 shows the latency overhead of different schemes for LSC decoder with code length $n = 1024$ and list size $l = 4$. We can see that all the mutil-decision schemes can significantly reduce latency overhead, and as increasing of code rate, the irregular mutil-decision scheme can still keep a very low latency overhead.

6.2.2 Latency Reduction via Path-LLR-Compute-Ahead Scheme

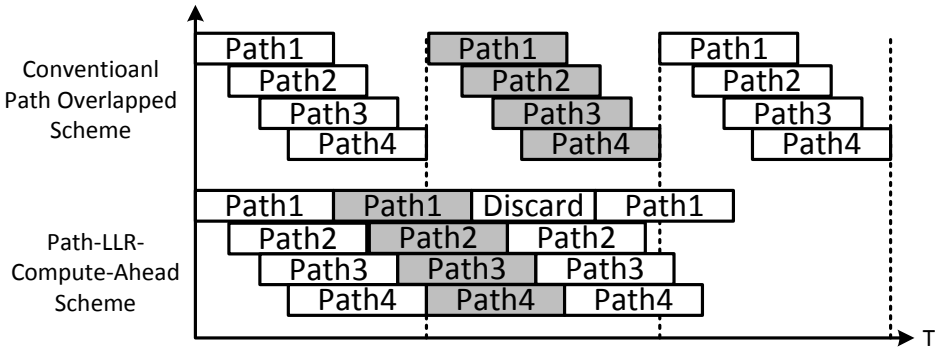


Figure 6.4: Decoding schedule of OPLSC.

Besides reducing the number of estimations, the other approach to decrease latency overhead is by avoiding the pipeline stalling, which can be achieved by overlapping multiple frames during the OPLSC decoding. Figure 6.4 shows this decoding

schedule. A single bar means the decoding process between estimations of two successive information bits. When pipeline stalling happens in one path, instead of waiting, current path can do a pre-estimated between two candidates (0 and 1) which it solely generates without suspension. The pipeline processing continues with the one with larger metrics and keeps the other to compared with the next coming paths. If more suitable paths are found later, the previous computed ones are discarded. With this scheme, the L_m for the best case is equal to pipeline latency L_p, which means the entire processing is handled without any stalling, and the L_m for the worst case is equal to simple path-overlapping scheme.

6.2.3 Latency Reduction via Adaptive LSC Decoding

In Equation 6.1, the second part of the formulation is equal to the L_p. It is determined by the number of paths existing in the pipeline. This makes the latency overhead increase linearly with respect to the list size l. If we can decrease the list size, the latency overhead can be significantly reduced. Typically, L_p is fixed for a LSC with given length. However, by applying adaptive LSC algorithm [53], the L_p is allowed to change on the fly according to current metrics of each path. The list size would decrease along the decoding processing, which also means the latency overhead would get reduction.

The summary of advantages of such techniques applied in OPLSC will be given in the following section.

6.3 Performance and Analysis

Figure 6.5 shows the improvement of e with proposed design approach for widely proposed LSC decoders with code length $n = 1024$ and list size $l = 4$. The x-axis is the rate of polar codes, and the y-axis is the ratio of e with proposed approach over e with original LSC decoders. OPLSC architecture is applied to four types of LSC

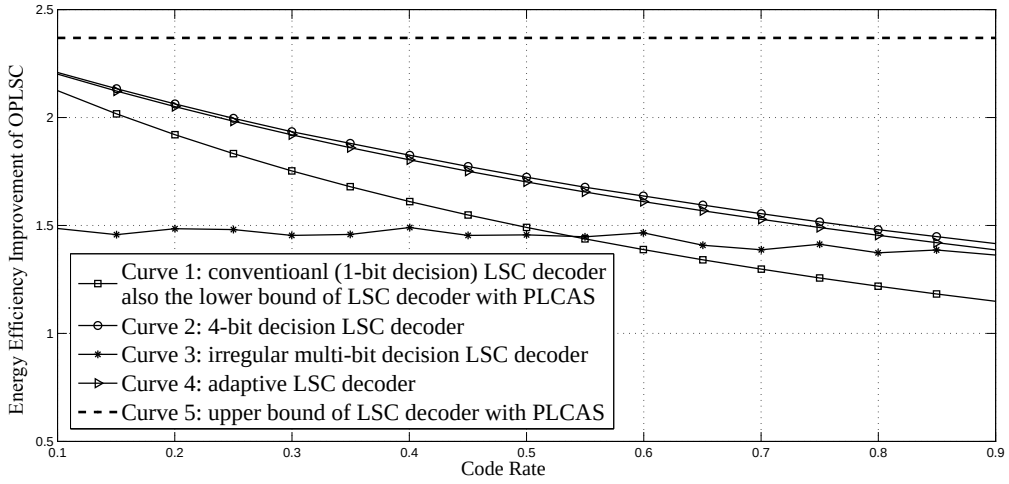


Figure 6.5: The improvement of energy efficiency of OPLSC.

decoder. They are conventional LSC decoder which also is regarded as 1-bit decision LSC decoder, 4-bit decision LSC decoder, irregular multi-bit decision decoder and the adaptive LSC decoder. We also calculated the upper and lower bound of the e improvement with PLCAS. These simulations are based on the decoders described in [43, 44, 47, 52].

In Figure 6.5, all the curves are beyond the ratio of one, which means with the proposed approach, all the decoders are able to achieve a better hardware efficiency. According to curve 1 and curve 2, the hardware efficiency of regular decision decoder, 1-bit and 4-bit decision decoder, is decreasing alone with the code rate increasing. This is because the latency overhead is larger at higher code rate. Besides, the regular multi-bit (4-bit) decoder achieves more improvement of e than that of conventional (1-bit) decoder, which is due to the latency reduction as we described in section 6.2.1. This can easily derive that for $n-bit$-decision regular decoder, the bigger the n, the more the improvement of e can be achieved with proposed approach. Curve 1 and 5 indicate the range of the e improvement with PLCAS. The actual value depends on

the channel outputs and channel quality.

According to curve 4 and curve 1, we can tell that the adaptive LSC help proposed approach to dramatically increase the hardware efficiency. Another very interesting phenomenon is about the improvement of irregular multi-bit decision (list fast-SSC decoder). The gain of e does not change too much with code rate varying. This is because the latency overhead of irregular multi-bit decision decoder does not linearly change along with coder rate. The average improvement of irregular multi-bit decision is less than that of regular one. This is due to the inherent latency of irregular LSC decoder is already very low [43].

Noticeably all the improvements are calculated based on the assumption that the maximum frequency of decoder with proposed approach or ordinary approach are the same. However, according to the analysis in Section 6.1, the maximum frequency of decoder with OPLSC approach should be higher, which indicates that the improvements of e in Figure 6.5 should be even more in practice. Additionally, all the approaches mentioned above are not conflicting with each other. Using multiple approaches together can further increase the hardware efficiency. The above mentioned properties indicate that proposed approach can measurably contain the hardware complexity associated with large scale LSC decoder implementation.

6.4 Summary

This chapter introduces overlapping path LSC decoder to improve the hardware efficiency of LSC decoder via path-overlapping scheme. The details of design approach and three strategies to reduce the latency overhead are also presented. The numerical results show that the conventionally used LSC decoders can significantly achieve a higher hardware efficiency by equipping OPLSC architecture.

7. EXPRESS JOURNEY BELIEF PROPAGATION DECODER*

In [54], different types of constituent codes are introduced to reduce the complexity of BP decoding algorithm. In this chapter, the simplifications on constituent codes are discussed with the details on scheduling techniques, which are designed to optimizing the hardware efficiency. To be specific, different scheduling techniques are discussed to reduce the decoding latency with constrained hardware resources. However, the specific scheduling techniques impose the difficulties on memory access of XJBP decoder. With specific scheduling techniques, VLSI architectures of the XJBP decoders are also presented which can accommodate the benefits of XJBP decoding algorithm as well as maintaining the feasibility of the XJBP decoder.

7.1 Algorithm Simplification

The general idea of XJBP decoding algorithm is to refine the estimation of the transmitted codeword $\hat{\mathbf{x}}$ without traversing the entire factor graph in each iteration. The various constituent codes are studied in this section to simplify the factor graph so as to reduce the decoding the complexity.

7.1.1 All-frozen $\mathcal{N}^0$ Codes

First type of the useful constituent codes are the codes whose left leaf nodes are all frozen bits. These codes are referred as $\mathcal{N}^0$ codes. Figure 7.1 shows an example of $\mathcal{N}^0$ code, where the shadowed nodes of $\{(1,2),(2,2)\}$ compose a $\mathcal{N}^0$ code. For those codes, there is no necessity to compute their LLRs, since the codeword is fixed by the frozen bits already. If the frozen bits are set to 0, the nodes of $\mathcal{N}^0$ codes are also

*Reprinted, with permission, from J. Xu T. Che G. Choi, XJ-BP: Express Journey Belief Propagation Decoding for Polar Codes, 2015 IEEE Global Communications Conference (GLOBECOM), and Dec. 2015. © 2015 IEEE.

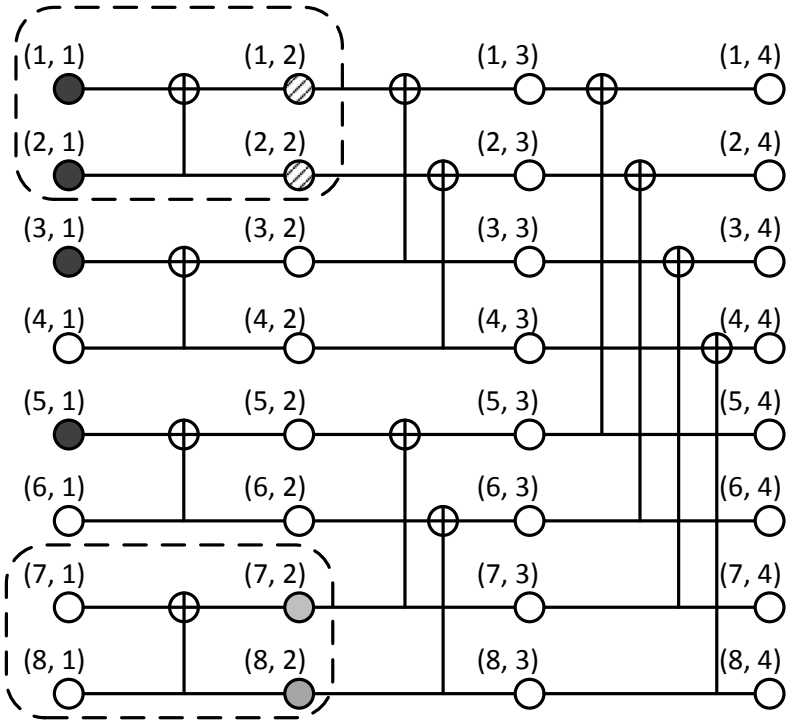


Figure 7.1: An example of $\mathcal{N}^0$ codes in shadow and $\mathcal{N}^1$ codes in gray.

0 in the encoding factor graph. Thus, by setting messages $R_{i,j}$ of nodes $\mathcal{N}^0$ codes as ∞ before the decoding, the decoding can be performed in each iteration without operating redundant processing elements left to the $\mathcal{N}^0$ codes.

7.1.2 All-information $\mathcal{N}^1$ Codes

As the counterpart of the $\mathcal{N}^0$ codes, $\mathcal{N}^1$ codes have their all leaf nodes of information bits. An existence of $\mathcal{N}^1$ code in the $n = 8$ polar code example is given in Figure 7.1. In the figure, the grayed codeword $\{(7, 2), (8, 2)\}$ is a $\mathcal{N}^1$ code whose leaf nodes are all information bits.

From the aspect of the factor graph, the refinement does originate from checking information provided by the frozen bits on leaf nodes. Since there is no frozen bits on the leaf nodes, it is implied that the messages do not get refined by further message

passing through $\mathcal{N}^1$ codes. From the Eq. (2.12) and (2.13), it also shows that the $R_{i,j+1}$ and $R_{i+2^{j-1},j+1}$ are not updated with consistent zeros of $R_{i,j}$ and $R_{i+2^{j-1},j}$. Thus the computations for $\mathcal{N}^1$ codes could be removed through BP decoding.

7.1.3 Repetition $\mathcal{N}^{REP}$ Codes

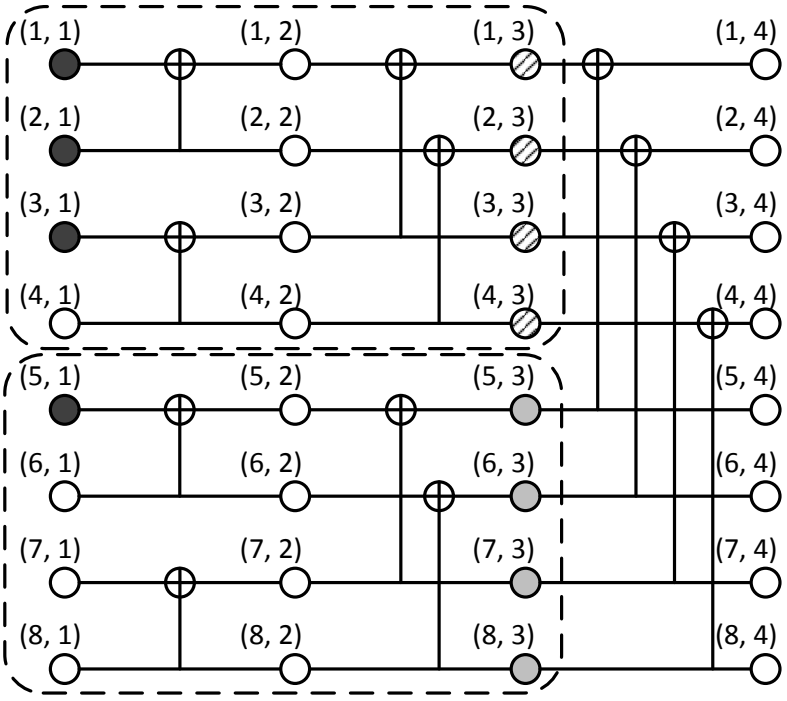


Figure 7.2: An example of $\mathcal{N}^{REP}$ codes in shadow and $\mathcal{N}^{SPC}$ codes in gray.

Another observation from the factor graph is that there exist considerable amount of constituent codes which only have a single information bit on the last leaf nodes. Those codes duplicate the only information bit by multiple times to construct the codeword. The repetition codes are referred as $\mathcal{N}^{REP}$ codes. The example given in Figure 2.5 does contain a $\mathcal{N}^{REP}$ code as shows in Figure 7.2, where the shadowed nodes $\{(1,3),(2,3),(3,3),(4,3)\}$ constitute a $\mathcal{N}^{REP}$ code.

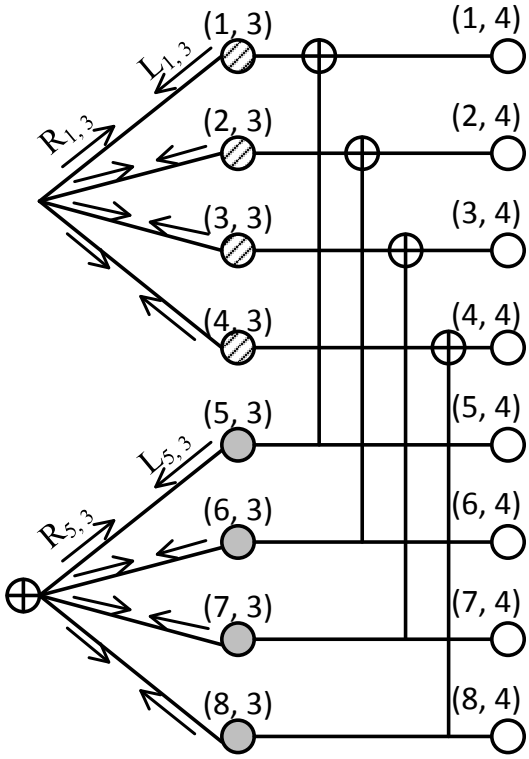


Figure 7.3: The simplified factor graph for the example of $\mathcal{N}^{REP}$ and $\mathcal{N}^{SPC}$ codes.

Since we already know that $\mathcal{N}^{REP}$ codes are formed by duplication, the conventional factor graph can be simplified so as to avoid message passing through multiple message stages. The corresponding example of the factor graph of the $\mathcal{N}^{REP}$ code is given in the Figure 7.3, where the top 4 shadowed nodes constitute a repetition code. Since each node is a duplication of others, they share the belief messages with others in the factor graph. The message passing rule of the $\mathcal{N}^{REP}$ codes follows the theory of factor graph [38] as:

$$R_{i,j} = \sum_{k \neq i} L_{k,j} \tag{7.1}$$

For a repetition code with length l, the complexity of conventional BP is $\mathcal{O}(l \log l)$. Whereas the complexity of the proposed updating rule is $\mathcal{O}(l)$. Specifically, the

proposed algorithm for length-l $\mathcal{N}^{REP}$ codes takes $(2l-1)$ two-input additions. Indiscriminately treating nodes of $\mathcal{N}^{REP}$ codes as normal nodes by using conventional BP consumes $(2l\log_2 l)$ comparisons operations and same amount of additions.

7.1.4 Single Parity Check $\mathcal{N}^{SPC}$ Codes

The other type of constituent codes exists in polar codes is the single parity check code. For those constituent codes that only have a single frozen bit on the first leaf node, the codewords are actually single parity check (SPC) codes, the sums of whose codewords are always zero in binary field. The SPC codes are also referred as $\mathcal{N}^{SPC}$.

As Figure 7.2 shows, the leaf nodes of the grayed constituent codeword with nodes: $\{(5,3),(6,3),(7,3),(8,3)\}$ are all information bits except the first one. Similar to $\mathcal{N}^{REP}$ codes, it is unnecessary to evaluate through all conventional computations to update the messages R of those nodes. Since the codeword is a SPC code, the factor graph of the $\mathcal{N}^{SPC}$ codes could be modeled as a parity check node connected with all bits of the codeword. The modified factor graph of the $\mathcal{N}^{SPC}$ code in the example is shown in Figure 7.3. In the figure, an additional parity check nodes is added to propagate the belief information among the nodes. With the consistency on using min-sum algorithm, the parity check update is written as:

$$R_{i,j} = \prod_{k\neq i} sgn(L_{k,j}) \cdot \min_{k\neq i} |L_{k,j}| \tag{7.2}$$

Similar as the repetition codes, the complexity of the modified message passing algorithm is $\mathcal{O}(l)$ for length-l single parity check code which is superior to the complexity of the conventional algorithm, $\mathcal{O}(l\log l)$. Thus with longer constituent codes, more computation are saved with the proposed algorithm.

Noticeably, the $\mathcal{N}^0$ and $\mathcal{N}^1$ codes are not usually included in $\mathcal{N}^{SPC}$ and $\mathcal{N}^{REP}$

codes in reality. Simplifications of message passing on those four different types of constituent codes are all applied simultaneously. The distributions of exclusive constituent codes in a $(1024, 512)$ are shown in Table 7.1. As the table shows, there are considerable amount of constituent codes in the polar code. There are more number of $\mathcal{N}^{REP}$ and $\mathcal{N}^{SPC}$ codes than $\mathcal{N}^0$ and $\mathcal{N}^1$ codes. Thus an efficient BP algorithm design for the $\mathcal{N}^{REP}$ and $\mathcal{N}^{SPC}$ codes could substantially further reduce the BP decoding complexity. Also notice that the distribution of the constituent codes does also depend on the code rate and polar codes with rate of 0.5 contain relatively less number of constituent codes. With higher code rate, it is more attractive to apply the proposed methods to simplify the message passing. The details of complexity analysis will be presented in Section 7.6.

Table 7.1: Number of all constituent codes with different sizes in a (1024, 512) polar code with rate of 0.5

	Constituent codes sizes						All
	4	8	16	32	64	128	
$\mathcal{N}^0$	3	3	2	2	0	1	11
$\mathcal{N}^1$	3	3	2	1	0	0	9
$\mathcal{N}^{REP}$	16	8	4	1	1	1	31
$\mathcal{N}^{SPC}$	15	5	3	1	1	0	25

With the constituent codes applied to reduce computations, the journey for message passing is simplified so that the LLRs of $\hat{\mathbf{u}}$ are not immediately available from BP iterations. Thus in the proposed algorithm, we focus on refining the estimations of transmitted codeword $\hat{\mathbf{x}}$ instead of messages $\hat{\mathbf{u}}$. The estimated LLRs of $\hat{\mathbf{x}}$, the soft estimations of transmitted codeword $\mathbf{x}$ in log likelihood ratio, are represented by Eq. (2.15). As aforementioned, $L_{i,m+1}$ are LLRs from the channel outputs. So

in our algorithm, $R_{i,m+1}$ is refined in iterations to accomplish decoding. The details how the computations are scheduled to accommodate the simplification is presented in the next section.

7.2 Early Termination

Before discussion the performance of proposed algorithm, an early termination technique is presented here, which is important to introduce for the efficiency measurements of XJBP decoding.

In this work, we apply early termination technique to determine whether the decoding is successfully done or not. Polar codes belong to the block codes. In [33], three different early termination techniques are proposed based on the usage of estimation of transmitted message $\hat{\mathbf{u}}$. However, in our proposed algorithm, $\hat{\mathbf{u}}$ is not available because of the simplified BP graph. In the absence of $\hat{\mathbf{u}}$, we employ a more straight-forward method based on the estimation of transmitted codeword $\hat{\mathbf{x}}$.

Noticeably, it is claimed in [33] that BP decoders do not output $\hat{\mathbf{x}}$. However, actually BP decoder does provide with information $\hat{\mathbf{x}}$.

For block codes, H matrix could be used for codeword detection. According to the coding theory [3], the parity check matrix H could be derived given generator matrix G'. Here G' is a $k \times n$ matrix consisting rows of matrix $\mathbf{G}$ corresponding to the positions of the information bits. Then the termination of a decoding is indicated by the equation:

$$\hat{\mathbf{x}}H = \mathbf{0} \tag{7.3}$$

where $\hat{\mathbf{x}}$ is the hard decision of the transmitted codeword estimations, i.e.

$$\hat{\mathbf{x}}_i = \begin{cases} 0, & LLR_i^{\hat{\mathbf{x}}} > 0 \\ 1, & otherwise \end{cases} \tag{7.4}$$

Noticeably, the early termination technique proposed here is not only specific to the proposed decoding algorithm, but it can be used in any other BP decoders.

7.3 VLSI Architecture and Resource Assignments

Overall architecture of proposed decoder is given in Figure 7.4. The proposed architecture mainly contains three processing units, array of conventional processing elements (PE), array of adders and the array of comparators. The tasks are assigned and distributed by a controller, which coordinates the data flow and control signals of the three units to perform XJ BP decoding.

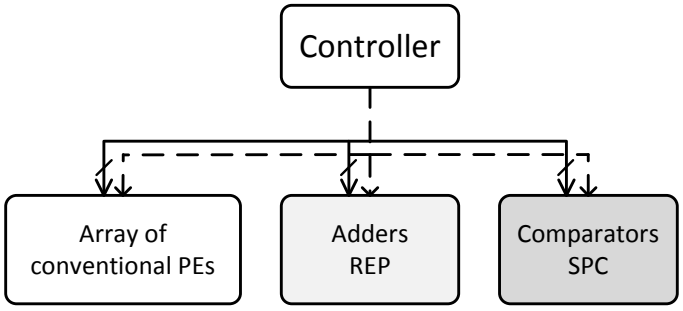


Figure 7.4: An overview of XJBP decoder.

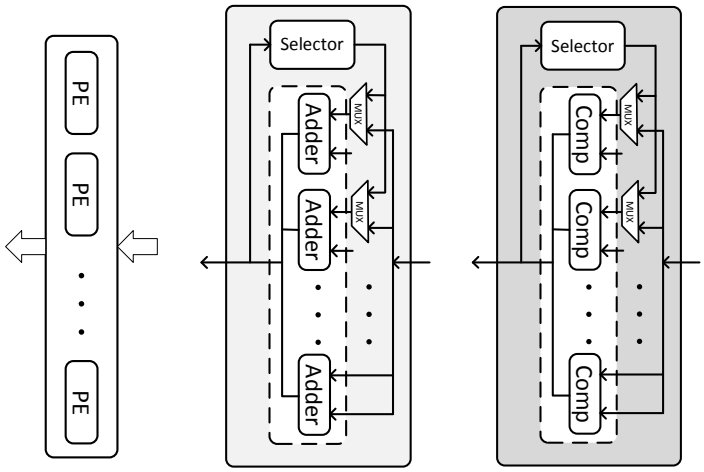
The overviews of the three types of processors, the PE array, adder array and comparator array, are given in Figure 7.5. The PE array consists of a set of processing elements (PEs). Each PE handles Equation 2.12 or Equation 2.13 depending on the

computation demand. For $\mathcal{N}^R EP$, the belief propagation updates are based on Equation 7.1, which can be written as:

$$R_{i,j} = \sum L_{k,j} - L_{i,j} \tag{7.5}$$

This indicates that the repetition codes updates can be accomplished by deducting the sum of right-to-left LLRs of the repetition codes by the corresponding right-to-left LLR for each input. Thus, the processes can be implemented as multiple-stage adder to sum all inputs, as the output of each port is the difference between the sum and corresponding input. Similarly, the belief propagation updates of $\mathcal{N}^{SPC}$ codes can be simplified by observing Equation 7.2. For a l long $\mathcal{N}^{SPC}$ codes, instead of looking for minimum of $l-1$ elements for each entry, alternatively we are going to search the two minimums out of all inputs. But for each output, one of the two minimums is selected based on if corresponding input is equal to the other minimum. Therefore, the implementations for the $\mathcal{N}^{SPC}$ codes consist of a set of $4:2$ sorters. Each sorters get the two minimums out of the four inputs. For a l long $\mathcal{N}^{SPC}$ codes, by iteratively applying the set of sorters, two minimum values are able to get acquired from l inputs. Then the output will be acquired after a further comparator to check if the corresponding input is equal to one of the two minimum, the other minimum will be assigned to the output.

To fully utilize the timing, the delay of one cycle of repetition codes updates and single parity check code updates should be close to that of the normal processing elements. Thus, $4:2$ sorters and four-input adders are applied in the context of this architecture. In the following, the early termination techniques and scheduling strategies are discussed to explore the advantages of the proposed architecture.



(a) (b) (c)

Figure 7.5: (a) Structure of array of processing elements. (b) Array of adders to decoder repetition codes. (c) Array of comparators to decoder SPC codes.

7.4 XJBP Scheduling

In this section, the details on how the computations are scheduled in the proposed architecture are discussed.

7.4.1 Round-trip Scheduling

First of all, let us recall the conventional scheduling of belief propagation decoders. As Figure 4.2 shows, the conventional scheduling strategy propagates the LLR always in a single direction from right to left. For a long n polar code, according to the Figure 4.2, each iteration consists of m stages of computations, where $m = log2(n)$ is the number of stages in the factor graph. For each stage, the messages of both direction $R_{i+1;j}$ and $L_{i,j}$ of each stage are computed. And the computations are repeated in one-way direction from left to right iteratively.

However, naturally the belief are propagated in a back-and-forth fashion. Another way to schedule the computations is to separately update right-to-left mes-

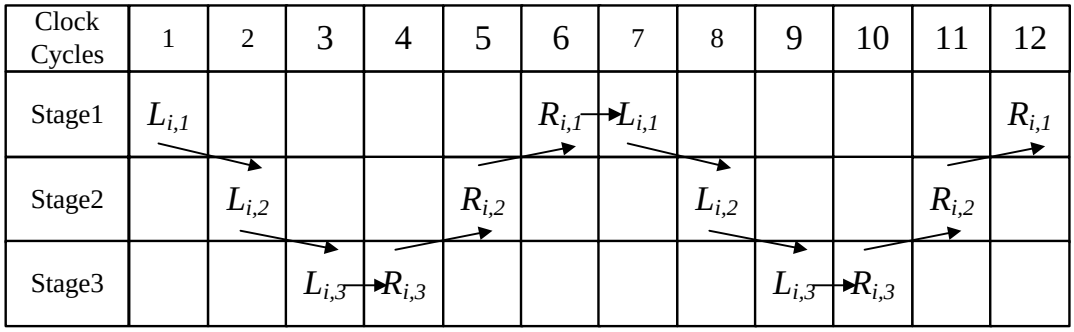


Figure 7.6: The computation scheduling and dependency for round trip BP of a $n = 8, r = 0.5$ polar code.

sages and left-to-right messages as mentioned in [35]. An example of dependency of a $n = 8, r = 0.5$ polar code is presented in the Figure 7.6. As the figure shows, the computations of each iteration are separated to two parts. In the first part, the right-to-left $L_{i,j}$ messages are updated from column m+1 to the most left nodes existing in the factor graph. The second is following to update the other direction messages $R_{i,j}$ from leftto the column $m + 1$. Since in each iteration there is a round trip through the factor graph, this scheduling scheme is referred as round-trip scheduling. Though each iteration of this modified scheduling contains a longer round-trip visit of nodes instead of one-way traverse, the amount of computations is same as that of the conventional scheduling, because only half of messages, either $L_{i,j}$ or $R_{i,j}$, are updated in each direction.

Furthermore, with XJBP decoding, the propagation path could be simplified. Figure 7.7 shows the express journey propagation path for an example of a $n = 8, r = 0.5$ polar code. For XJBP decoding, the propagation does not even reach the most right bits, because of the existences of constituent codes. By Figure 7.7, the advantages of XJBP decoders are shown more clearly, since the decoding cycles are reduced more substantially in each iteration.

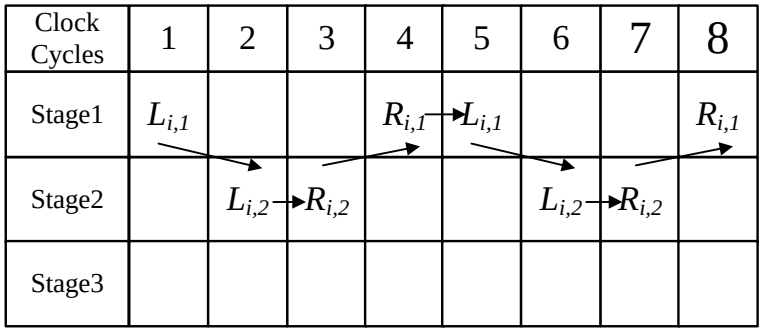


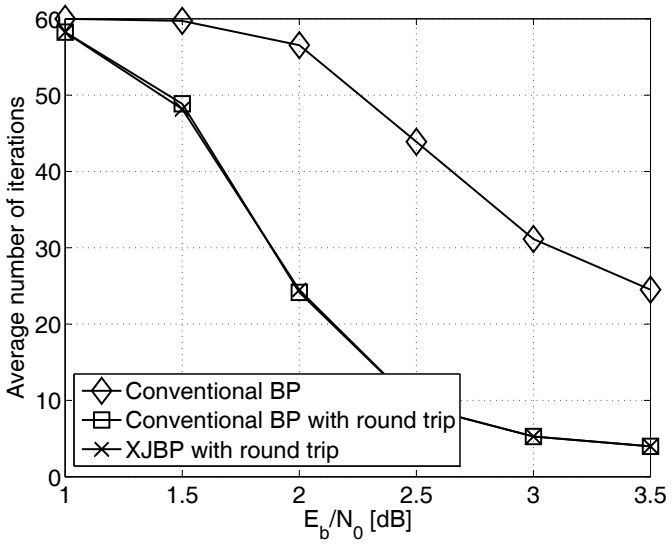
Figure 7.7: The computation scheduling and dependency for round trip XJBP of a $n = 8, r = 0.5$ polar code.

To verify the reliability of the round trip scheduling and XJBP decoding, Figure. 7.8 shows the performances and efficiency of round-trip scheduling strategies as well as equipped with XJBP decoding. In the experiments, the maximum number of iterations of both decoders are set to 60. From Figure.7.8b, we can see that the performance of round-trip scheduling strategy outperforms the conventional BP decoding, while the XJBP decoding algorithm does not impose any degradation on the belief propagation decoding. In terms of efficiency, through Figure.7.8a, the numbers of iterations of round-trip scheduling is much less than that of normal scheduling technique.

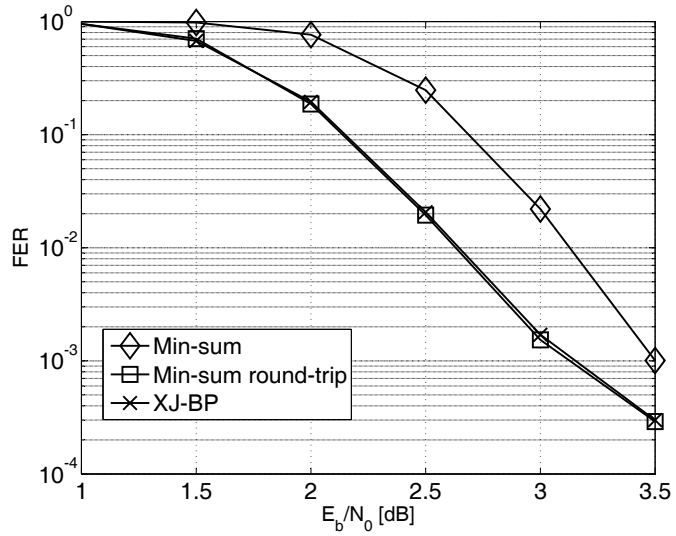
Table 7.2: This is a comparison of number of iterations in different BP decoders

	Traditional[31]	5-stage SMS[32]	5-stage Folding[33]	XJBP
number of iterations	40	21.9	24.5	4.1
Performance Difference	-0.3dB	0	-0.3dB	0

Furthermore, we compare the number of iterations of our proposed method with



(a)



(b)

Figure 7.8: The comparison between roundtrip scheduling and conventional scheduling in terms of efficiency (number of iterations) (a) and performance (Frame error rate) (b).

state-of-the-art BP decoders in Table 7.2. By using the round trip scheduling, the efficiency of BP decoding is improved significantly. Furthermore, compared with other-of-the-art BP basded decoder, the round trip did not impose any performance degra-

dation. Specifically, the Min-Sum algorithm based round trip scheduling achieves the same performance as scaled min-sum algorithm but with much one sixth iterations.

By studying the case of the $n = 8$ polar code, advantages of XJBP decoding and round-trip scheduling are presented. However, for large scale polar code, the scheduling for XJBP will be difficult. In the following, the dependencies and scheduling are studies in the context for a more general case.

7.4.2 Dependency

Before we start trying to scheduling all computations, we first look into the dependencies graph of the XJBP decoding. As aforementioned, each iteration of decoding process contains two parts of propagation, first part is propagate the belief information from the channel input to the constraints (From right to left in the context figures). The second one is sent back the refined belief information from left to right. In the first phase, each stage breaks down a longer polar code to two constituent codes. And the constituent codes might have some special properties like $\mathcal{N}^{REP}$ or $\mathcal{N}^{SPC}$.

Figure 7.9 shows the process of LLR propagation in the first phase of XJBP decoding for a 128 long polar codes with rate of 0.5. As the figure shows, the dependency of the first phase is similar as a tree, where the data in each node depends on its parent node. Every stage breaks down a polar code into two smaller pieces. The root node is a conventional 128 long polar code, referred as C node in the figure. Every C node has two child nodes, which are generated by the a set of Equation 2.12 and Equations 2.13. In the third stage, first special constituent codes appear. There are a 32 long repetition code and a 32 long single parity check code, which are referred as R and S nodes in the figure, respectively. With deeper the position, a node represents a shorter length constituent code. The leave nodes in

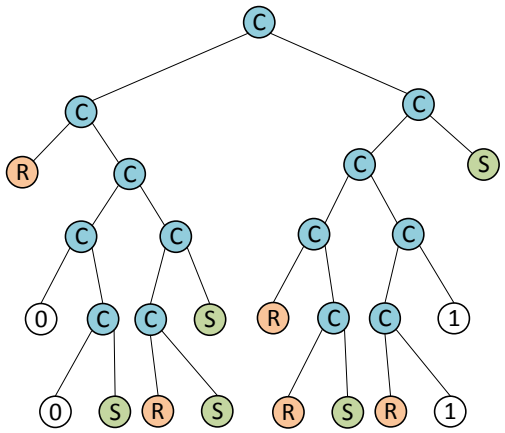


Figure 7.9: The dependency for belief propagation of a $n = 128, r = 0.5$ polar code.

the dependencies trees must be a special constituent code, which falls in the set of $\mathcal{N}^0, \mathcal{N}^1, \mathcal{N}^{REP}, \mathcal{N}^{SPC}$

To explore a more accurate data dependency between different computation, the data flow graph for an entire iteration of the 128 long polar code case is presented in Figure 7.10. In that figure, each iteration is accomplished by collecting left-to-right LLRs $R_{:,n}$ at the most right layer in the coding structure. In the bottom half of the dependency graph, nodes are presenting the LLR propagation from left to right. Every stage increase the size of constituent codes by two. For each special constituent codes, they are fed into right-to-left LLRs and output left-to-right LLRs. Thus, in the middle of the data flow, nodes are all special constituent codes. Noticeably, although only one node is used in the graph to present each stage processing, the scales of the processes of nodes are different. In the top half, with the decreasing size of polar codes, deeper nodes have relatively smaller scale of processes. However for the bottom half, nodes in deeper stages involve more parallel computations.

By the proposed the architecture mentioned above with unlimited resources, each

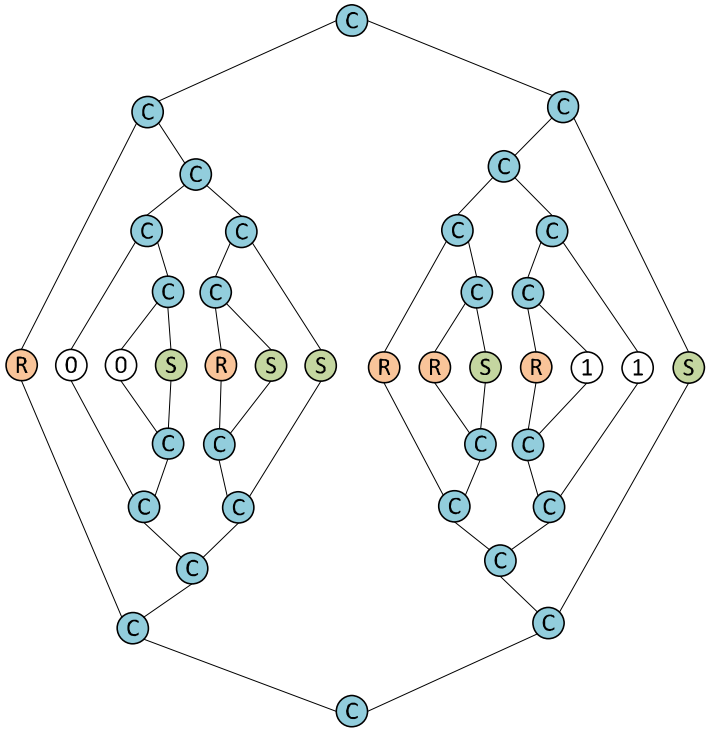


Figure 7.10: The dependency for an iteration of BP decoding of a $n = 128, r = 0.5$ polar code.

C node consumes one cycle to accomplish processing. However, R nodes and C nodes need iteratively using the adders and comparators to acquire the sum and minimum respectively. And the delays of R and C nodes depend on the scales of the corresponding repetition codes and single parity check codes. For a long n $\mathcal{N}^{SPC}$, it takes $\log_2(n) + 1$ to finish updating LLRs, while for a long n $\mathcal{N}^{REP}$, it takes $\lceil \log_4(n) \rceil + 1$ to update LLRs.

Based on the dependency, different scheduling strategies are explored to utilize the resource efficiently to schedule all computation. However, the problem of deciding whether a schedule exists for a set of dependent tasks and a given deadline is NP-complete problem [55]. To solve the scheduling problem, strategies employed in this

thesis to make the problem easier are described in the following.

7.4.2.1 Add resource constraints

As mentioned above, the XJBP consists of different hardware units to delivery the final results. For a certain hardware implementation, the resource is fixed. So an alternative exploration can be done through scheduling optimization for a certain hardware resource, which simplifies the problem substantially.

7.4.2.2 Split problems into static and dynamic part

Another idea to simplify the scheduling problem in this thesis is that we split the problem into Static and Dynamic parts. By doing this, more efforts are paid for a specific polar code size and length. A better optimal schedule is explored in off-line to minimize the decisions to be taken at run-time.

7.4.2.3 Use scheduling algorithms from high-level synthesis

To statically schedule tasks, different parameters could be utilized to explore the optimal schedule. In this thesis, according to proposed architectures, different assumptions are made with corresponding algorithms to schedule computations to improve the hardware efficiency.

In the following, based on the simplification above, different situations are given to discuss the optimal solutions for computations scheduling.

7.4.3 Scheduling Without Priority

A straight scheduling method is to handle all ready tasks as soon as possible without considering the priority among different tasks. Figure 7.11 shows the idea of this straight

Besides the straight forwards scheduling algorithms, a more sophisticated scheduling strategy based on ASAP/ALAP methods called list scheduling (LS) is mentioned

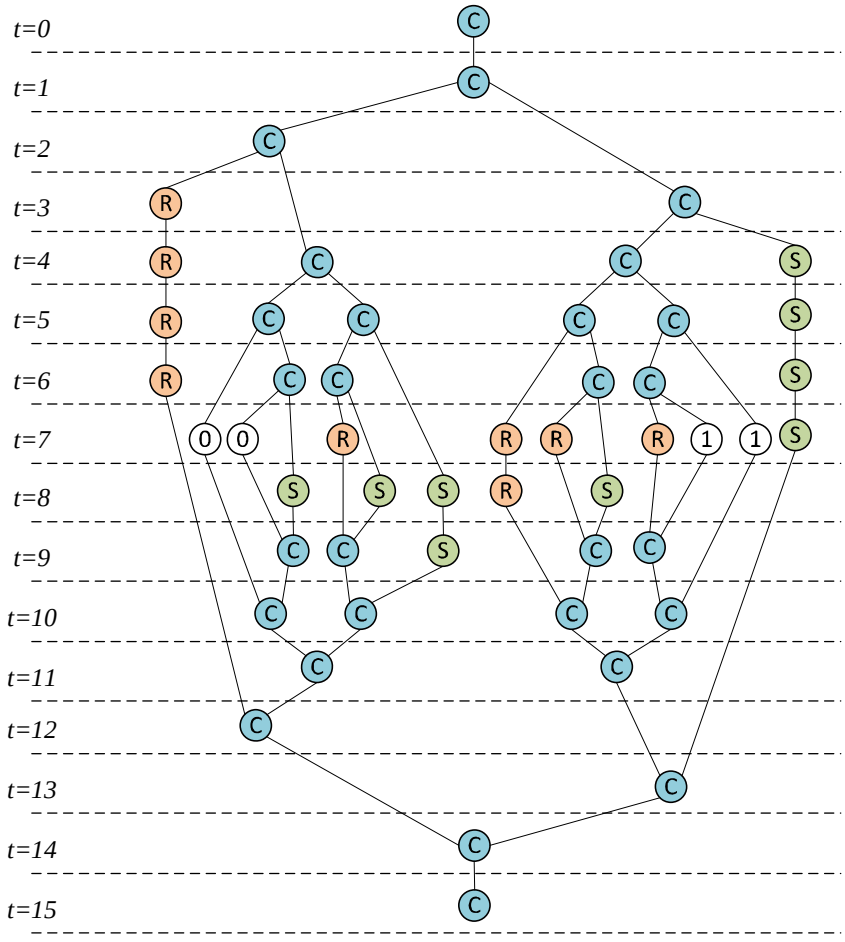


Figure 7.11: An example of ASAP scheduling for XJBP decoding of a $n = 128, r = 0.5$ polar code.

[56]. The idea of list scheduling is to take more factors into considerations to improve the efficiency. Following gives more discussion with priorities to schedule all computations events.

7.4.4 Scheduling With Priority

Recall the dependency figure 7.10, tasks have different-length paths to final results in the end. Thus different path has different urgency to be processed. Longer path is more urgent to process. In this thesis, we recognize the tasks on longer path has

higher priorities. Based on this assumption, different scheduling algorithms can be developed to schedule all tasks.

As aforementioned, all C nodes have one cycle delay to process the LLRs, while S and R nodes have multiple cycle to accomplish the belief propagation update. Thus, the question arises first here is whether the hardware has the ability to interrupt the tasks and resume them later. If the hardware allows SEP and REP tasks interrupted, we refer the corresponding scheduling algorithm as preemptive, non-preemptive otherwise. The answer to the question depends on the micro-architecture applied in the practice. In following, two scheduling cases of both preemption and non-preemption are discussed.

7.4.4.1 Scheduling with preemption

If the hardware implementation allows the interruptions of multiple-cycles tasks, a simple modification on the ASAP scheduling algorithm can be made to apply the idea of priority. Instead of processing the assigned tasks till they finish, a task which arrives later but with higher priority occupies the hardware resource by interrupting the unfinished tasks.

Figure 7.12 presents an example of computations for a $n = 128, r = 0.5$ polar code scheduled by the proposed preemptive scheduling method. In the example, we assume that there are constrained resource to compute 64 LLRs associated with conventional polar codes, repetition constituent codes and single parity check codes inclusively.

Noticeably, as mentioned above, nodes with different layers have different scales of complexity. Before the propagate the LLRs from left to right, the scales of nodes get decreased by half each at each dependency progress. Thus it is assumed that the hardware can handle more nodes with smaller scales.

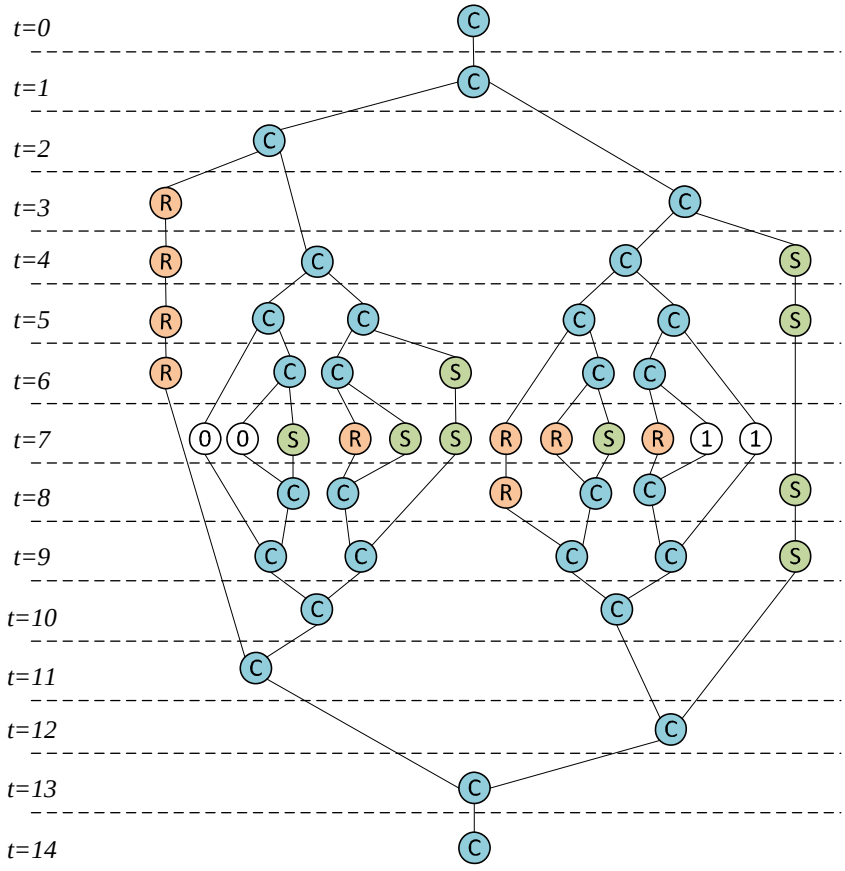


Figure 7.12: An example of preemptive scheduling with priority for XJBP decoding of a $n = 128, r = 0.5$ polar code.

From the comparison between Figure 7.12 and Figure 7.11, we can see that the more urgent paths with more nodes receive more consideration to schedule as soon as possible. Although the less urgent tasks are ready more earlier because of their early executed predecessors, less urgent tasks are paused when tasks on critical paths arrive. In this study case, the preemptive scheduling algorithm with consideration of priority does not impose any delays on the critical path. Without instantiating any new hardware resource, the schedule reduces the number of cycles by 2, which is around 11% of total delays in the ASAP scheduling algorithm. However, the

deduction on the computation delays also depend on the code length, which will be discussed later in this section.

Algorithm 1 Preemptive Scheduling Algorithm for XJBP with priority

```
initialization
set time t = 0
computer priorities of all nodes;
while Any nodes are not scheduled do
    Pick tasks with highest priority from the ready tasks group c_i ∈ C
    if Uc_i is occupied then
        Pause the task in the Uc_i
        Push the paused task into ready tasks group
    end
    Schedule task c_i on resource Uc_i at timej;
    Update candidate tasks C
    set time t++
end
return schedule;
```

Algorithm 1 shows the details of the scheduling process. In this static scheduling, the ready tasks are pulled from container one by one, if the hardware resource are available to new tasks. The tasks are scheduling at the time. While if all hardware are busy at the moment, the priorities between the ready tasks and undergoing tasks are compared to determine if the undergoing tasks should be paused to allow more urgent tasks get processed.

It is noticeable that the deductions on the decoding latency depend on the code sizes and code rates. Given the resources of hardware, Table 7.3 shows amounts of delays of conventional BP decoding, XJBP decoding with ASAP scheduling and XJBP decoding with preemptive scheduling based on priority. In the entries of the first column of the table, numbers stand for the hardware resource allocate for normal constituent codes, $\mathcal{N}^{REP}$ codes and $\mathcal{N}^{SPC}$ codes. For example, for 1024 long polar

Table 7.3: Decoding delays of XJBP with different code sizes

		code sizes n				
Hardware resource	scheduling	1024	2048	4096	8192	16k
C Units: n/4	regularBP	40	44	48	52	56
S Units: n/32	XJBP-ASAP	34	35	43	45	48
R Units: n/32	XJBP-LS	27	29	35	35	40
C Units: n/8	regularBP	80	88	96	104	112
S Units: n/64	XJBP-ASAP	64	62	79	83	90
R Units: n/64	XJBP-LS	50	52	62	62	73

codes, $(1/2, 1/8, 1/8)$ means that at each cycle, there are most a 512 long normal constituent, a 128 long repetition code and 128 long single parity check code could be processed by the hardware. Because that there exist much more conventional constituent polar codes in the data flow, there are more resource of normal PE units allocated than resource for the special constituent codes. From the table, we can tell that with any long polar codes, the XJBP decoding algorithm significantly reduces the number of cycles by around 35% for each iteration with introducing a overhead on the hardware resource for special constituent codes.

However, the scheduling algorithm discussed in this subsection is based on assumption that the tasks could be interrupted on the fly. This results in a extra memory spaces to store the interrupted tasks and its status. An alternative way to optimize the schedules could be made without this assumption, which is referred as non-preemptive scheduling. Following the discussions on the non-preemptive scheduling are given.

7.4.4.2 Scheduling without preemption

Different with preemptive scheduling, non-preemptive schedule does not interrupt on on-going tasks. This will results in idle cycles of processors even when there are

ready tasks for the processor.

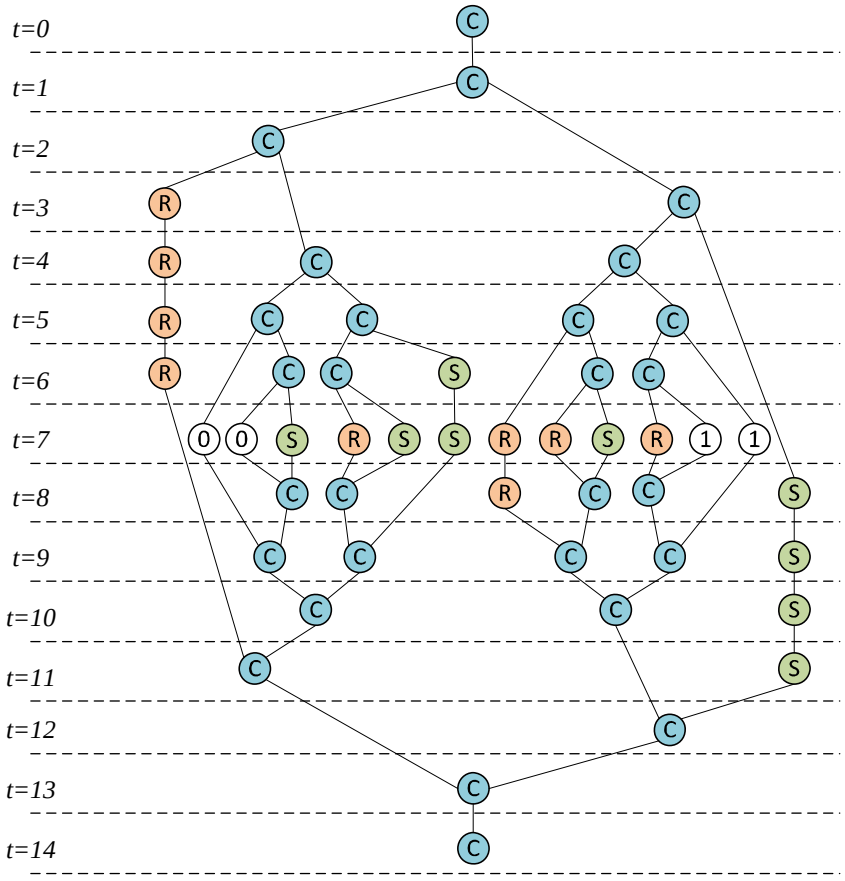


Figure 7.13: An example of non preemptive scheduling with priority for XJBP decoding of a $n = 128, r = 0.5$ polar code.

Figure 7.13 shows the scheduled tasks for the same example as above. In this schedule, continuous multiple-cycle processes of repetition and single parity check codes updates are not interrupted during the running. From the figure, we can see that processor does always execute computations even when the corresponding tasks are ready. This is because that during the time executing low-priority tasks, higher-priority tasks may arrive to request processing. Compared with schedule of ASAP

without priority in Figure 7.11, the non-preemptive scheduling algorithm can still reduce the processing delays, although the improvement is relatively less significant than preemptive scheduling.

Algorithm 2 Scheduling Algorithm for Static Non-preemptive

initialization
computer priorities of all nodes;
while *Any nodes are not scheduled* **do**
  Pick the task with the highest priority $c_i \in \mathcal{C}$
  $j = 0$
  while j++ **do**
    if $|c_i^j| + |g^j| \leq B$ *and Not Violating Dependency* **then**
      break
    end
    Schedule task c_i at $timej$;
  end
  Update candidate tasks $\mathcal{C}$
end
return schedule;

Algorithm 2 gives heuristic static scheduling algorithm for the proposed non-preemptive situations. Instead of scheduling ready tasks at a time, only the tasks satisfied by the following condition will be scheduled. The condition is that if the scheduled task does not conflict with other same-type higher-priority tasks which are scheduled based on ASAP after the task scheduled. In other word, if the schedule results in potential higher-priority tasks conflicting, the schedule is not accepted.

The non-preemptive scheduling algorithm has the most complexity in running time. And the delay of this algorithm stands intermediate position between ASAP scheduling and preemptive priority scheduling algorithm.

7.5 Memory Access

After discussing about the scheduling problem of XJBP, the details of hardware implementation are given in this section. Because of the introductions of sophisticated scheduling algorithms, the memory access become more complicated.

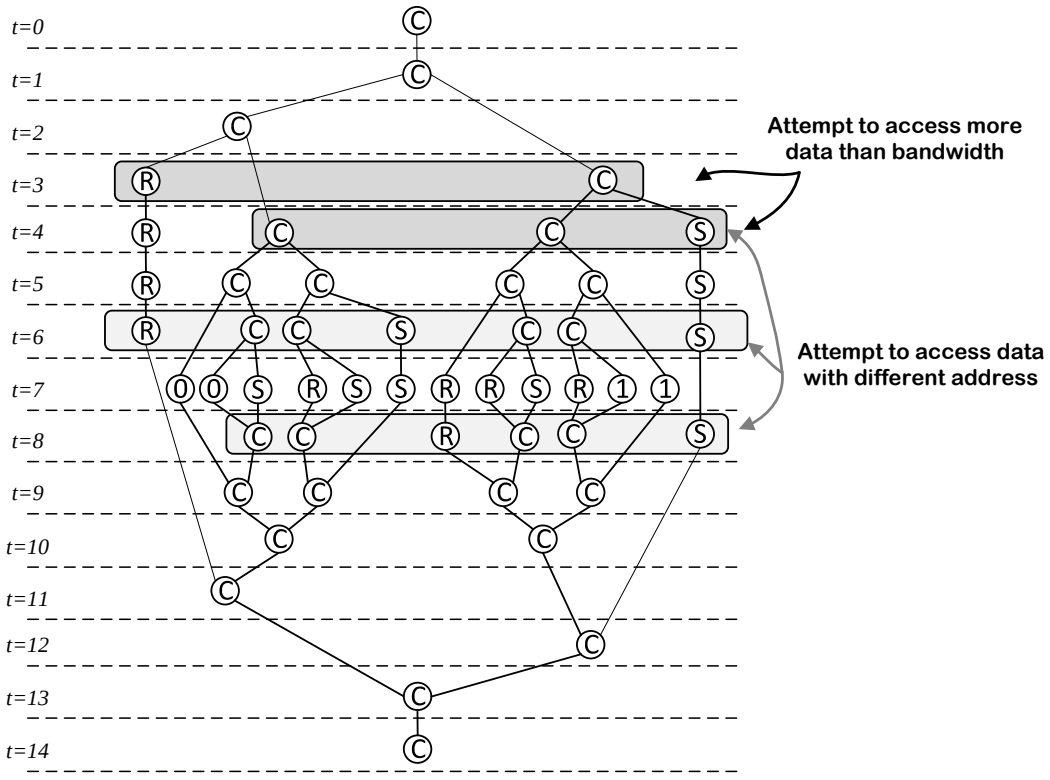
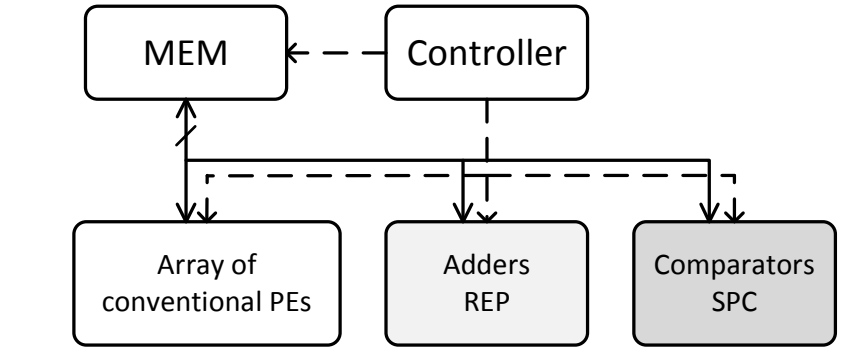


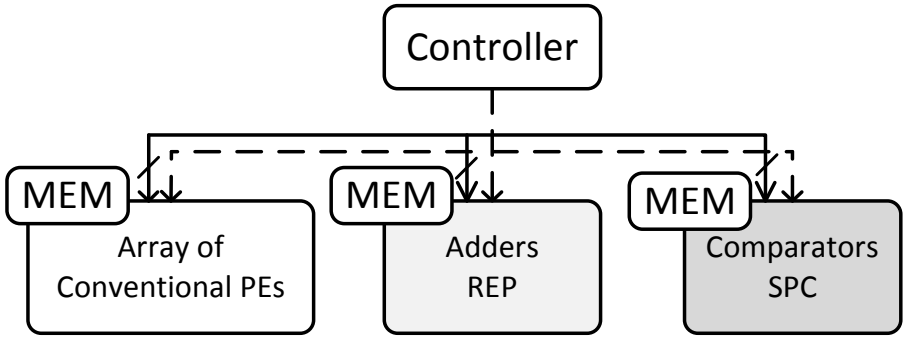
Figure 7.14: An example of memory accesses for XJBP decoding of a $n = 128, r = 0.5$ polar code.

Figure 7.14 shows an example of memory access problems in the context. In the figure, there show two potential memory access problem with the proposed scheduling algorithm in a traditional share-memory architecture. First of all, with a traditional shared memory architecture, the bandwidth is limited. However, in the proposed

method, different units may request accessing memory at same time, which result in requirement on higher memory bandwidth than the conventional one. Furthermore, the problem is not just about the bandwidth. With the design of different processing units for conventional BP nodes, single parity check nodes, repetition codes nodes, it is possible that different units attempt to access data which do not belong to a single line in the traditional memory.



(a)



(b)

Figure 7.15: The comparison between shared memory micro-architecture (a) and distributed memory micro-architecture (b).

Thus instead of using a shared-memory architecture, we distributed the memory into each local processing units. Figure 7.15 gives the traditional shared memory

access BP decoder and proposed distributed memory access. In the proposed method as shown in Figure 7.15b, each of those distributed memory is dedicated to store associated data so as to solve the problems mentioned above. In the following, the details of the each processor along with its memory are discussed.

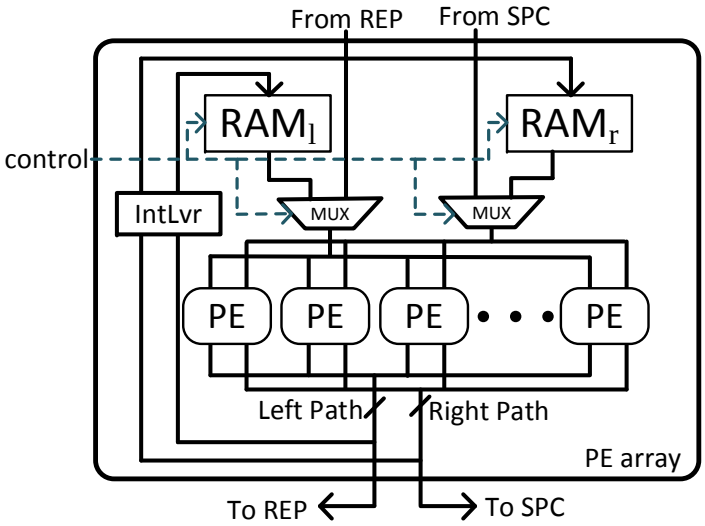


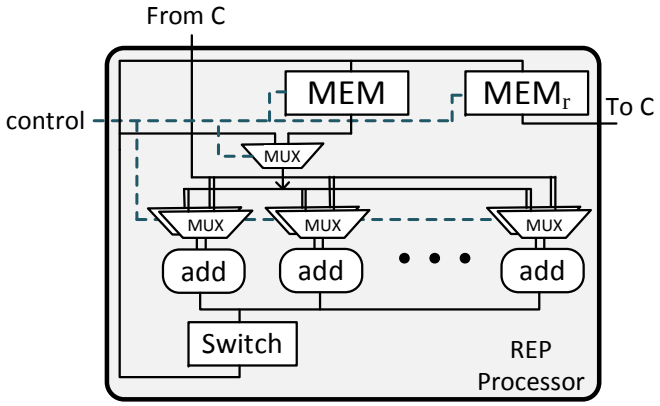
Figure 7.16: Implementation of a conventional processing element of XJBP decoder.

Figure 7.16 shows the internal structure of the array of conventional processing elements. Each processing element is design to handle the classic BP decoding equations2.12 and 2.13. MEM$_l$ and MEM$_r$ are used to store data which are fed into the processing elements. For each PE, there are four inputs, two of which are given by MEM$_l$ and the other two given by MEM$_r$. The observation is made that in the scheduling figures 7.11, 7.12 and 7.13, the C nodes have two child nodes in the top half and two parent nodes in the bottom half. By further observation, we can tell following conclusions:

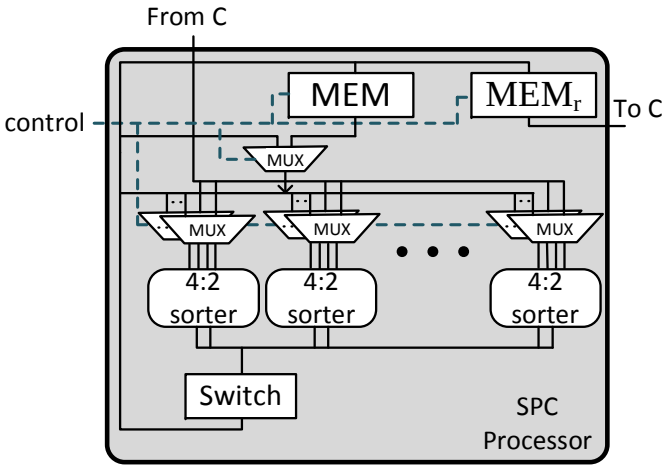
- In the top half data graph, the outputs of C nodes can be split into two parts as forwarding to right or left.
- The inputs of repetition codes must come from the left-part outputs of C nodes.
- The inputs of single parity check codes must come from the right-part outputs of C nodes.
- In the bottom half data graph, the inputs of C nodes can be split into two parts as forwarded from right or left.
- The outputs of repetition codes must be fed into the left-part inputs of C nodes.
- The outputs of single parity check codes must come from the right-part inputs of C nodes.

Guided by the above observations, the micro-architecture of processing elements can be explored easily. First of all, the outputs of the array of PEs are split into two parts as shown in the data flow. If the current tasks of C node have repetition codes as left child node, left-part data are forwarded to REP codes unit. Otherwise, the left-part data are fed back to memory for next conventional codes computation. Similarly, depending on the type of right child node type, the right-part data can be forwarded to SPC codes units or fed back to memory of PE array itself.

On the propagation way back from left to right, the inputs of the array of processing elements can be split into two parts, from left predecessor or from right predecessor. As the observation, we can tell that the inputs from left predecessor can be from either the PE array itself or from the repetition codes processors. Thus multiplexers are given in this proposed architecture to allow the data to be directly fed by the repetition codes units. Vice versa, the from-right-predecessor data also have options of directly from SPC units or from PE array itself.



(a)



(b)

Figure 7.17: The implementation details of REP processor(a) and SPC processor (b).

Once we have the micro-architecture of C nodes, the processors of REP and SPC codes can be correspondingly designed, because REP and SPC processors have to follow the interaction with the conventional PE array. Following gives the details about the REP and SPC processors.

Now, let us look into repetition and single parity check codes processors in the preemptive scheduling. Figure 7.17 shows the internal structures of $\mathcal{N}^{REP}$ and $\mathcal{N}^{SPC}$

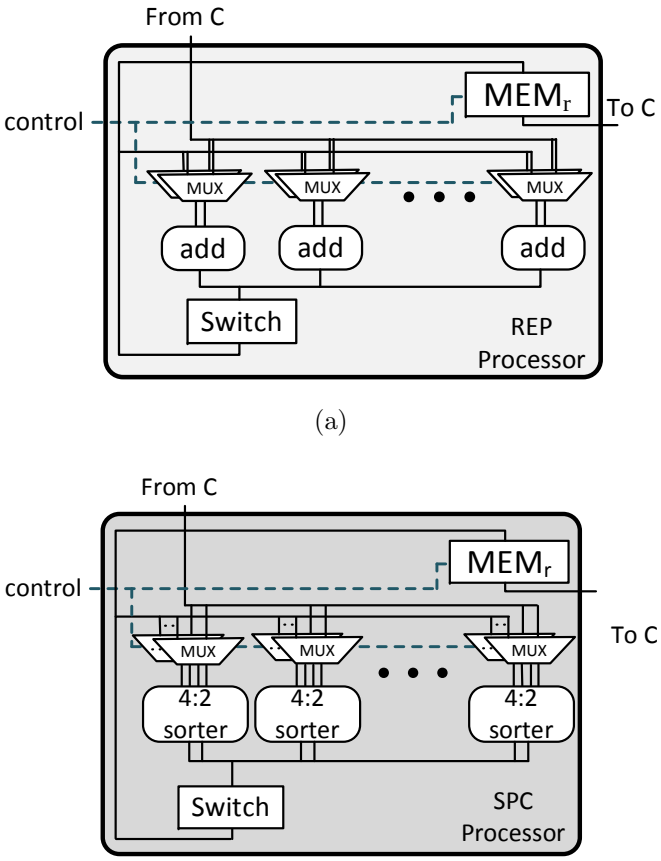


Figure 7.18: The implementation details of non-preemptive REP processor(a) and SPC processor(b).

codes processors. With the micro architecture design of PE array, the repetition codes processor is much more straight forward. First of all, the inputs of the processor must be from the C nodes as PE array. The outputs of the processor must feed PE array back. The internal structure of REP processor are consisting of multiple adders. In our design, $4:1$ adders, which have four inputs and sum as output, are used to get a comparable delay with the conventional PE.

For a larger scale repetition code, multiple cycles are needed to accomplish updates. An additional memory is allocated to allow the interruption in the preemptive

scheduling algorithm. The calculated data are saved in a memory until accessed by the C node. Similarly, single parity check code processors have same architecture but replacing the adders with $4:2$ sorter to select the two minimum values of the input vector.

After talking with preemptive units, the structures of non-preemptive $\mathcal{N}^{REP}$ and $\mathcal{N}^{SPC}$ codes processors are given in the Figure 7.18. The structures have only one difference between preemptive and non-preemptive structure. The non-preemptive hardware solutions get rid off the additional memory, since there does not exist data which need store temporarily.

Following gives the discussion about hardware consumption of the proposed micro-architecture and comparisons with other state-of-the-art BP decoders.

7.6 Results and Discussion

In this section, the hardware performance is presented. The complexity of XJBP decoder is analyzed and compared with other state-of-the-art BP decoders without consideration of implementation details. Furthermore, a fixed-point analysis on XJBP decoder is given to guarantee the feasibility of the XJBP decoder in practice. In the end, the hardware synthesis results are given and discussed.

7.6.1 Complexity Estimation and Analysis

In this subsection, we set up experiments to verify the efficiency and analyze the complexity of the XJBP decoding algorithm. As an example, $(1024, 512)$ polar code is used to emulate the proposed decoder to estimate the complexity with max number of iterations of 60.

First of all, the average numbers of iterations of those algorithms are summarized in the Figure 7.8a. It is shown in the figure that with the round-trip scheduling computations, the efficiency of the BP algorithm is significantly increased. Noticeably

scaled min-sum BP algorithm reduces the number of iterations. However the reduction is at the cost of the additional scaling computation in each node update. The interesting phenomenon from this experiment is that the round-trip scheduling significantly improves the iteration efficiency without the additional computational complexity cost. Under the condition of high $E_b/N_0 = 3.5$, the round-trip BP scheduling only takes 3.98 average iterations to complete decoding. As mentioned above, the amounts of computations for conventional scheduling and round-trip scheduling in each iteration are the same. Compared with 24.5 average number of iterations consumed by the conventional MS BP decoding, the decoding efficiency is immediately improved by 83.7% without considering the simplification on factor graph yet. Also, it is addressed that the proposed XJ-BP algorithm does not reduce the number of iterations compared with the traditional BP but with round-trip scheduling.

Secondly, we evaluate the reduction of computations in each iteration resulting from the proposed XJ approach for message passing. As mentioned above, computations for nodes of $\mathcal{N}^0$ and $\mathcal{N}^1$ codes could be removed directly. The computations of $\mathcal{N}^{REP}$ and $\mathcal{N}^{SPC}$ codes are reduced by XJ-BP. The numbers of total operations (2-input addition or 2-input comparison) are shown in the Table. 7.4. In the table, polar codes are set at rate = 0.5 and the channel polarization is done under the binary erasure channel (BEC) model with erasure ratio of 0.3. It is shown that the total number of computations could be reduced by about 40% in each iteration using the proposed simplified BP algorithm. And we found that this ratio is kept at about 40% even with significantly longer code length. In another word, the proposed simplification saves around 40% amount of computations regardless of lengths of the polar codes.

Another factor that affects the simplification is the code rate. Table. 7.5 shows the number of computations for proposed algorithm decoding a polar code of length

Table 7.4: Number of computations of XJ-BP algorithm with all polar codes at rate = 0.5

	Polar code sizes				
	128	256	512	1024	2048
Conventional BP	1792	4096	9216	20480	45056
XJ-BP	1040	2488	5536	12160	27304
Ratios [%]	58.0%	60.9%	60.1%	59.4%	60.6%

1024 at different typical code rates. As the table shows, the proposed algorithm saves more computation resource to decode polar code with higher code rates. This is because that more constituent codes exist in the factor graph with more unbalanced number of frozen bits and information bits.

Table 7.5: Computations of XJ-BP algorithm in each iteration at different code rates

	Code Rates				
	1/2	2/3	3/4	5/6	7/8
conventional BP	20480	20480	20480	20480	20480
XJ-BP	12160	11488	10680	9376	8936
Ratios [%]	59.4%	56.1%	52.3%	45.8%	44.6%

Finally, the overall complexity reduction is evaluated by considering both the reduced number of iterations and simplified computations in each iteration. Take the (1024, 512) codes as an example, Figure 7.19 shows the average numbers of computations to decode one codeword at different levels of E_b/N_0. Due to the extra scaling operations, SMS consumes around 34% more computations over the conventional MS decoding algorithm, although SMS outperforms conventional BP in terms of decoding performance. Compared with conventional BP decoding, the round-trip scheduling reduces the number of computations by 83.7% at $E_b/N_0 = 3.5$ resulting

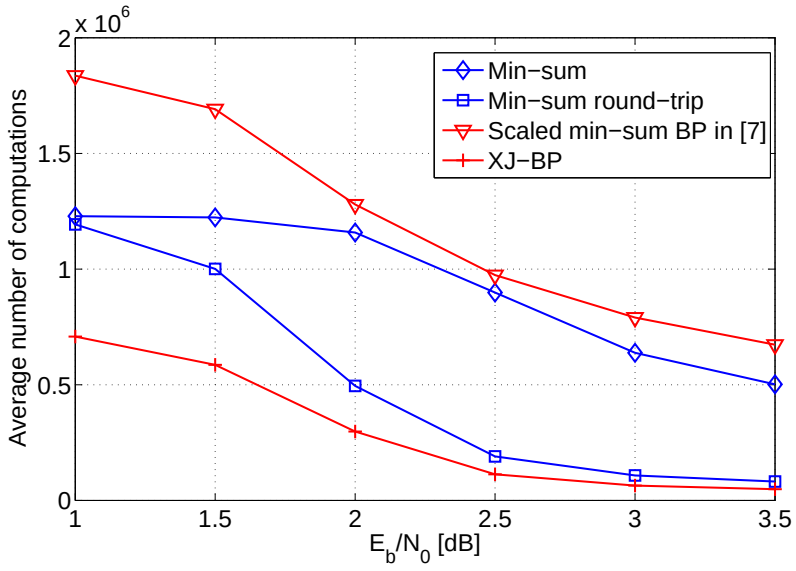


Figure 7.19: Average numbers of computations consumed to decode each codeword of by the proposed BP decoding algorithm for $(1024, 512)$ polar code with rate $= 0.5$.

from the reduced number of iterations. Based on round-trip scheduling, the proposed method does not yield any further improvement on number of necessary iterations. However the XJ-BP decoding simplifies factor graph so as to reduce the computations in each iteration by 40.6%. As a results, the overall complexity is reduced by 90.4% using XJ-BP, compared with conventional BP decoding.

7.6.1.1 Discussion

From the aspect of practical implementation, the conventional BP processing element symmetrically computes updates for messages $R_{i,j}$ and $L_{i,j}$. Traditional computations for $R_{i,j}$ as shown in Equation (2.13) are as same as those for $L_{i,j}$ in Equation (2.12). In practical implementation for the proposed algorithm, the processing elements should be designed as only to deal with functions $\mathcal{G}(x, y + z)$ and $\mathcal{G}(x, y) + z$ to satisfy only one-direction message computations. Thus, one should be aware of that the processing element for regular codes in this thesis consume only

half of the resource consumed by "Processing Elements" in other state-of-the-art BP decoders.

7.6.2 Fixed-point Analysis

Before showing the synthesis results, a fixed-point performance of the proposed architecture is analyzed to determine how many bits are necessarily to assign on the belief propagation so that the hardware performance could be promised.

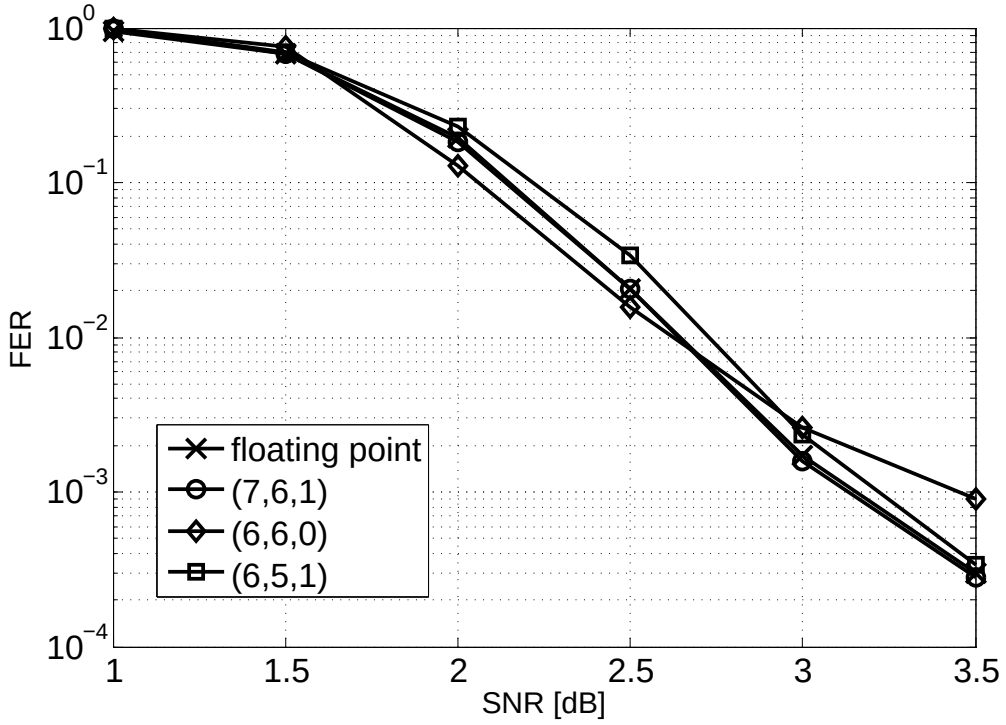


Figure 7.20: Quantized error correction performance of XJBP.

The results of the proposed architecture are briefly summarized in Table 7.6. The table shows the area and power consumption of all units in this proposed micro architecture. Since there are only comparison and addition involved in the computations of the proposed algorithms, we apply same quantization scheme on both

channel input LLRs and internal propagation LLRs. We use three number (C, I, F) to represent quantization scheme, where C, I and F stand for the total number of bits assigned on each value, number of bits assigned on integer part and number of bits assigned on fraction parts respectively. Figure 7.20 shows the performances of the floating-point model and fixed-point model at different quantization schemes. It is clearly shown in the figure that with scheme $(7, 6, 1)$, there is no degradation from the floating point model. With one bit reduction on the fraction part, scheme $(6, 6, 0)$ only results in a negligible degradation on the error correction performance. However, if we reduce one bit on the integer part, a error floor appears and results in unexpected performance degradation on the error correction capability. Thus, we adopt quantization scheme $(7, 6, 1)$ to implement our XJBP decoder.

7.6.3 Synthesis Results and Discussion

Table 7.6: This is a logic consumption table for XJBP

	normal codes 256 PE	REP codes 32 adder	SPC codes 32 adder
equivalent gate count	85.2k	10.5k	26.9k
Delay(ns)	1.88	1.1	1.78
Power(mW)(@ 500MHz)	28.2	4.74	10.3

To estimate the power consumption of the proposed deocder, RTL model for a (1024, 512) polar XJBP decoder is developed in Verilog HDL. The consumption on logic and memory are sperated estimated to deliver a clear picture on power consumption on proposed decoder. The design is synthesized by *SynopsysDesignCompiler* with FreePDK CMOS 45nm library. The supply voltage is set to 1.1 volts with typical timing model at 27 C.

Table 7.6 shows the hardware consumption on computation units, which includes 256 conventional processing elements, 32 repetition codes and single parity check codes processing units. It is shown in the table that the two new introduced units for repetition codes and single parity check decoding does yield less latency than the conventional PE. It means that the proposed idea does not reduce the clock frequency of traditional BP decoders designs. Besides the timing information, it can be seen in the table that the proposed new processing units consume less hardware resources than normal PUs in terms of gate counts. As mentioned above, this is because that the special computation resource are demanded less in the factor graph and they are more efficient to delivery updates. Also the power consumption of all units based on clock frequency at $500MHz$ are estimated in the synthesis report. Similar as the gate counts, REP and SPC units consumes much less power. Specifically, REP codes consumes more less power, since they have simpler computations of additions instead of $4:2$ sorting in the SPC PUs.

Table 7.7: This is a memory consumption table for XJBP

	Normal	P-REP	NP-REP	P-SPC	NP-SPC
bits	71260	3416	1708	5824	2912
gate count (k)	470	22.5	11.3	38.4	19.2
Power (mW) (@500MHz)	271.2	13	6.5	22.2	11.1

For BP decoder, all LLRs in the factor graph need to store in memory for access during the propagation. Although XJBP decoder eliminate a portion of LLRs by simplifying the factor using constituent codes, the space to store LLR is still considerable. To estimate the memory consumption, the numbers of bits necessary to store in each unit are first examined in simulation. Along with the synthesis results on

memory space, the numbers of bits for each module are presented in the Table 7.7. In the table, the memory consumption of different processing units are listed. Noticeably, to support preemptive scheduling method, the hardware architecture needs extra memory space to save the paused tasks. Therefore, the memories consumed by preemptive PUs and non-preemptive PUs are separated. From the table, we can see that conventional BP PUs consume most amount of memories because of the abundant amount conventional tasks. Similar as the results in logic consumption, the memory consumption of SPC is higher that of REP. This is because of the longer delays of SPC codes updating, which results in more data temporarily saved in SPC local memory.

Table 7.8: This is a total hardware consumption table for XJBP

		Normal PUs	SPC PUs	REP PUs	total
Preemptive	gate count (k)	555.2	65.3	33	653.5
	power (mW) (@500MHz)	299.4	32.5	17.7	349.6
Non-Preemptive	gate count (k)	555.2	46.1	21.8	623.1
	power (mW) (@500MHz)	299.4	21.4	11.24	332

To give a more clear picture on hardware resource consumption, Table 7.8 presents total consumption from both memory and logic. Conventional processing elements consumes most of total hardware resource (approximately 90%). This implies that the introduction of SPC and REP units does not results in much overhead but significantly reduce the number of cycles in each iteration. From the table, it is also shown that because of the the large portion contribution of conventional processing elements, the difference of hardware resources between non-preemptive and preemptive

situation is limited. It indicates that the advantages of non-preemptive scheduling will be eliminated on cases when non-preemptive scheduling results in more latency than preemptive scheduling.

To give a more comprehensive answer to this question and compare with other state-of-the-art BP decoders, a detailed comparison is given in Table 7.9. In Table 7.9, polar code $(1024, 512)$ is taken as a case to study the hardware performance of different BP decoders.

The number of iterations used in decoding in each frame is based on BPSK channel with $SNR = 0.5$. All hardware results are based on FreePDK CMOS 45nm library with supply voltage of 1.1 volts and typical timing model at 27 C. The efficiency is analyzed based on both power and area[50]. The power efficiency is defined as:

$$\text{Power Efficiency} = \frac{\text{Clock Frequency} \times n}{\text{power} \times \text{latency}} \tag{7.6}$$

where n is the code length of polar codes. The definition turns out the results which stand for the number of bits decoded by each unit amount of energy. In addition The area efficiency is defined as:

$$\text{Area Efficiency} = \frac{\text{Throughput}}{\text{Gate count}} \tag{7.7}$$

which gives the throughput achieved by each unit amount of hardware resources in terms of gate counts. To be more accurate and fair on hardware capacity, the throughput in the definition is derived based on the maximum clock frequency which can be achieved:

$$\text{Throughput} = \frac{\text{Max Frequency} \times k}{\text{Decoding latency}} \tag{7.8}$$

where k is the number of information bits in each codeword of polar codes.

			Traditional [31]	5-stage SMS [33]	5-stage MS[32]	ASAP	Preemptive	NP
number of iterations			40	21.9	24.5	4	4	4
average cycles per iteration			20*P	2.13	2.13	34	27	34
Logic Consumption	#PE	C	1024/P	5120	5120	256	256	256
		SPC	n.a.	n.a.	n.a.	32	32	32
		REP	n.a.	n.a.	n.a.	32	32	32
	delay(ns)		1.88	1.93	1.45	1.88	1.88	1.88
Memory (bits)			157,696	788,480	788,480	75,880	80,500	75,880
Total Gate Count (100K)			3.4/P	19.2	15.7	6.2	6.5	6.2
Throughput (@Max Freq)			340/P Mbps @ 532MHz	5.69Gbps @518MHz	6.77Gbps @690MHz	2.00Gbps @532MHz	2.52Gbps @532MHz	2.00Gbps @532MHz
Power (mW) @500MHz			/	/	/	332	349.6	332
Energy Efficiency (bit/nJ)			3.05	4.55	4.97	11.3	13.6	11.3
Area Efficiency (Mbps/Kgates)			1	2.96	4.31	3.26	3.87	3.26
Performance Difference			-0.3dB	0dB	-0.3dB	0dB	0dB	0dB

Table 7.9: This is a hardware comparison table between XJBP and other state-of-the-art BP decoders

The performances among proposed different scheduling strategies, ASAP, Preemptive and Non Preemptive are presented in the most right columns. As shown in the table, the preemptive scheduling takes less number of iterations because of consideration of tasks priorities as analyzed in the Section 7.4 but with more hardware consumption on extra memory space in the REP and SPC PUs. Furthermore, non-preemptive scheduling does also consider tasks priorities, however, for the $(1024, 512)$ polar codes, the scheduling does not help reduce the amount of cycles in each iteration. On the hardware consumption side, compared with preemptive scheduling, the extra memory saved by non-preemptive scheduling contribute a limited portion of the total power consumption of XJBP decoder. That results in the hardware efficiency non-preemptive scheduling inferior to that of preemptive scheduling in practice.

Compared with other state-of-the-art BP decoders, the numbers of iterations are substantially reduced by proposed XJBP decoder. Array-architecture BP decoders in [32, 33] significantly reduce the cycles for each iteration. Along with optimization on gate level design in [32], they turn out a considerable area efficient design on BP decoder. However, much more hardware resource are consumed in [32, 33] as the cost to achieve the small number of cycles for each iteration. As a results, the power efficiency of our proposed XJBP decoder substantially outperform the multi-stage folding BP decoders. Noticeably, although the proposed XJBP decoder has much higher power efficiency, the area efficiency among XJBP decoder and the state-of-the-art decoders are comparable. This is because of two factors. First is the area efficiency is derived based on the maximum frequency. Due to the optimization on gate level design in [32, 33], BP decoders of those achieve a higher clock frequency. Second reason is that in XJBP decoders, multiple processors are introduced to reduce the number of cycle in each iteration but which does also result in some idle cycles of some processors due to the dependency problem.

To give a more comprehensive comparison among implementations of different algorithms, the results of state-of-the-art SC decoders are also included in Table 7.10. As the results shown in table, we can see that the SC decoders consumes much less area because of its advantages on computations complexity. However, the serial decoding nature impose the difficulty for SC based decoder to achieve higher throughput. As a result, proposed BP decoder outcomes better performances than SC-based decoder as well as state-of-the-art BP based decoder in terms of energy and area efficiency.

Table 7.10: Hardware consumption comparisons among SC and BP decoders

	[39]	[29]	[33]	XJBP
algorithm	SC	SC	BP-SMS	BP-MS
Total Gate Count (100K)	3.4	3.4	19.2	6.5
Throughput (Gbps) @ Max Freq	0.3	0.8	5.69	2.52
Latency (cycles)	2046	767	47	136
Power (mW) @ 500MHz	84	84	/	349.6
Energy Efficiency (bit/nJ)	2.98	7.94	4.55	11.3
Area Efficiency (Mbps/Kgates)	0.88	2.35	2.96	3.87

To sum up, by using static scheduling techniques on XJBP decoder, the number of cycles per iteration is substantially reduced which turns out much higher energy efficiency performance on hardware implementation. XJBP makes the MS decoders to achieve same performance as SMS decoders without introducing extra scaling hardware computations. To compare with same error correction ability BP decoder of [33], the proposed XJBP decoder enhances power efficiency and area efficiency by $3X$ and $1.3X$, respectively.

8. OTHER WORKS*

Before the invention of polar codes, LDPC codes are applied widely because its near-capacity error correction performance. Besides development of polar codes decoders, I also put my effort on improving energy efficiency of LDPC decoders. For this target, techniques of asynchronous circuits are employed to implement LDPC decoder. This chapter will present my contributions to the asynchronous LDPC decoders.

Similar as error correction codes decoding, compressive sensing is another interesting problem in the field of signal processing, whose reconstruction problem is similar as decoding problem which searches for a solution by given a projection over a matrix. This chapter also shows my contributions on the compressive sensing techniques.

8.1 Asynchronous Design for Precision-Scaleable LDPC Decoder

Low-density parity-check codes [5, 7] are powerful error correction codes that perform very close to the Shannon limit and are used in many communication standards such as IEEE 802.16e (WiMAX) [57], DVB-S2 and IEEE 802.11n (WiFi) [58]. LDPC performance is significantly affected by the decoding algorithm. Excellent LDPC performance is achieved by soft decoding typically with the sum-product (SP) algorithm. The algorithm operates by iteratively passing a posteriori probability or

*Reprinted, with permission, from J. Xu T. Che E. Rohani G.Choi, Asynchronous Design for Precisionscaleable Energy Efficient LDPC Decoder, Signals, Systems and Computers, 2014 48th Asilomar Conference on, and Nov. 2014. © 2014 IEEE. Reprinted, with permission, from J. Xu E. Rohani M. Rahman G. Choi, Signal Reconstruction Processor Design for Compressive Sensing, Circuits and Systems (ISCAS), 2014 IEEE International, and Jun. 2014. © 2014 IEEE. Reprinted, with permission, from J. Xu G. Choi, Compressive Sensing and Reception for MIMO-OFDM Based Cognitive Radio, Computing, Networking and Communications (ICNC), 2015 International Conference on, and Feb. 2015. © 2015 IEEE.

log-likelihood ratio (LLR) messages along the edges of a factor graph [59], which involves check-node update and variable-node update. In practice, the variants of SP algorithm such as min-sum (MS), offset min-sum (OMS) algorithms are used to be implemented for avoiding overflow and efficient hardware implementation.

LDPC decoder is first implemented in hardware using fully-parallel architecture by [60]. However, fully-parallel LDPC decoder suffers from a lot of aspects such as complex interconnect issues to access the memory, inefficient logic utilization. To efficiently address those problems, [50] proposes an improved architecture which utilizes the value-reuse property of the OMS algorithm with layer decoding. For an exploration of the low-power design on the existing LDPC architectures, [61] proposes LDPC decoder equipped with the dynamic voltage and frequency scaling (DVFS) technique. In [61], the decoder is designed to run at minimum clock frequency and supply voltage to meet latency requirements by estimating the necessary number of iterations. The synchronous circuits have to assure that the critical path in the design does not violate the time limitation imposed by the clock frequency. Although DVFS technique enables the low-power operation of the LDPC decoder, it requires the strictly accurate voltage and frequency controls to make sure that the completion time of synchronous circuits do not violate the limitation of the clock frequency. [62] proposes a design with an adaptive wordwidth mechanism to reduce the power consumption by optimizing the datapath in the hardware. Without dynamically scaling voltage and frequency, the power reduction is relatively moderate in [62]. Scaling voltage and frequency helps significantly reduce the power consumption though, it introduces difficulties on the coordination between voltage and scaling to avoid time violation.

Fortunately, the natures of asynchronous circuits overcome the problems mentioned in DVFS technique and make asynchronous circuits a efficient method to

do low-power design [63]. Asynchronous circuits has recently received increasing attentions also because of its high operation speed, omission of clock distribution related problems, and robustness with respect to variations in supply voltage [64]. An asynchronous baseband processor framework design for satellite communication is proposed in [65] to mitigate the radioactive interference. [66] proposes a fully parallel LDPC decoder with the asynchronous circuits. Since the check nodes in a fully-parallel have substantially various transmission delays at each iteration, [66] proposes to utilize asynchronous techniques to speed up the interleaver of the LDPC decoder so as to enhance the logic utilization. In [67], we present a precision-scalable LDPC decoder on asynchronous circutis techniques.

8.1.1 Proposed System

Figure 8.1 shows the overview system for the proposed LDPC decoder. The LDPC decoder is composed by the check nodes units (CNU) and variable nodes units (VNU) which iteratively refine the received bit stream. In the proposed system, the LDPC decoder consists of precision-scaleable units which are designed in asynchronous circuits as delay variously depending on the precision of the calculation in use. The control model is introduced here to determine the pair of necessary computation precision and voltage supply, for a given decoding fidelity, according to the interference strength. Different with the conventional synchronous precision-scalable design, event-driven asynchronous circuits allow not to dynamically scale the operating frequency along with voltage scaling. To accommodate the precision scalability, the modifications of CNU and VNU are presented in this following.

The basic data flow chart of LDPC decoder is given in Figure 8.2. There are different algorithms on how to propagate the belief through the check nodes and variable nodes. In practice, min-sum algorithm and its variants are popular for

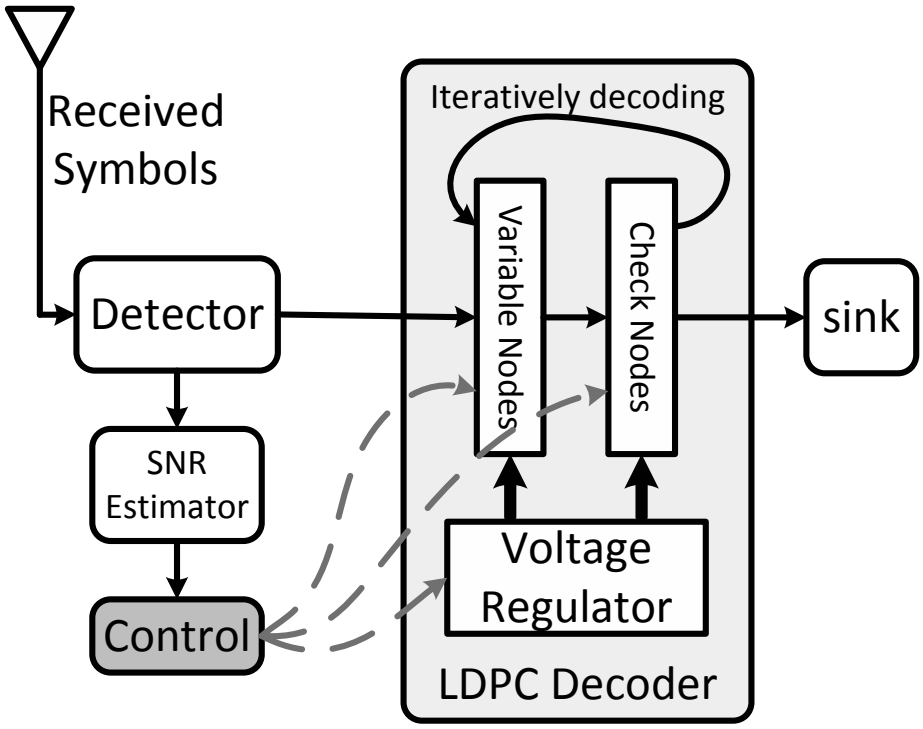


Figure 8.1: The overview system flow proposed LDPC decoder.

its simplified operation on belief propagation. In this paper, min-sum algorithm is taken as the example to be applied on our precision-adaptive decoder design in asynchronous circuits. Noticeably, although min-sum algorithm is taken to be implemented in our paper, the design flow could be extended to apply other BP LDPC decoding algorithm.

When to update the check nodes at ith iteration, for each variable node V_n and the set of its neighboring variable nodes $m \in M(n)$, the message sent from variable nodes to check nodes, Q^i_{nm}, could be presented as:

$$Q^i_{nm} = y_n + \sum_{m' \in M(n) \backslash m} R^{(i)}_{mn} \tag{8.1}$$

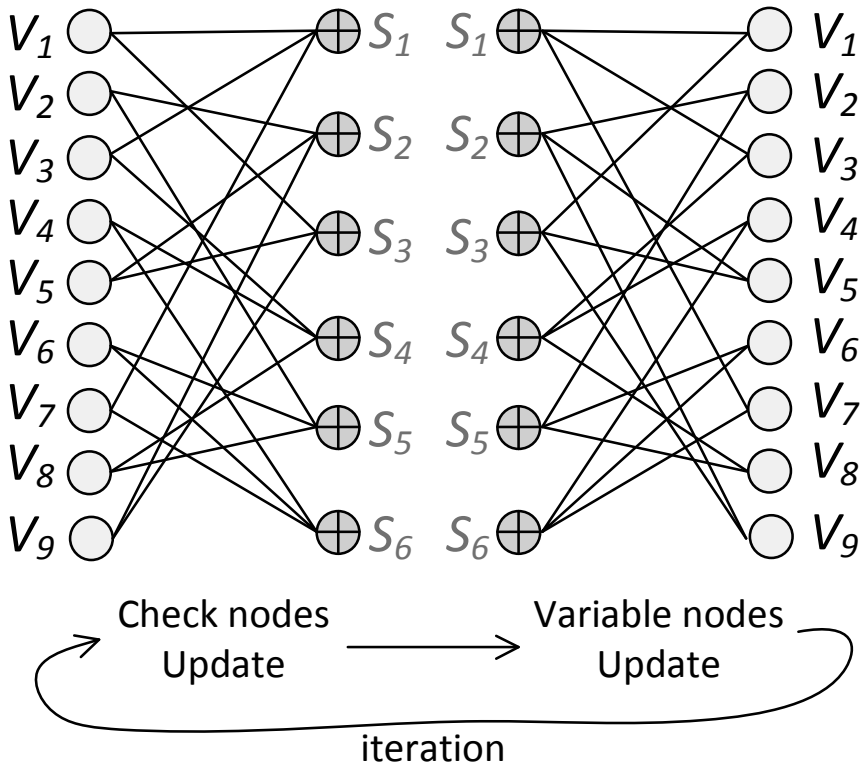


Figure 8.2: Generic LDPC decoding data flow graph.

$R_{mn}^{(i)}$ is the message sent from check nodes S_m to variable node V_n. It is updated in the variable nodes update phase by the following equation:

$$R_{mn}^{(i)} = \Big(\prod_{n' \in N(m) \backslash m} sign(Q_{n'm}^{i})\Big) \min_{n' \in N(m) \backslash m} (Q_{n'm}^{i}) \tag{8.2}$$

In the proposed asynchronous circuits, those computations above are implemented in a way that the precision is reconfigurable so that minimum necessary precision will be used to do computation on the fly. The details of the implementation will be presented in the next subsection.

Asynchronous circuits are unlike the synchronous circuits that consist of registers and combination logic, driven by a global clock. On the contrary, in asynchronous

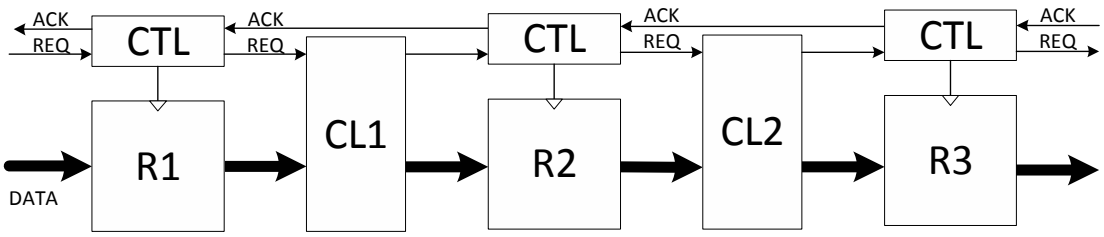


Figure 8.3: Asynchronous circuits data path model.

circuits, data forwarding from the input to the output ports are controlled by handshaking between receiving and forwarding units via acknowledgement signals. The processing-flow model of asynchronous circuits is shown in Figure 8.3. The REQ signal from previous stage informs the next stage that the new data is ready. Then the combinational logic (CL) circuit commences to compute. . After the combinational computing, the control module (CTL) generates an ACK signal and send it back to the previous module to inform that current computing cycle is complete and it is ready to receive new data. In this paper, the combinational logic for the VNU and CNU are designed with the scalability of the precision. Under different configurations of the precision, the combinational logic costs different amount of resources in terms of voltage or time to complete computations. Due to the nature of the clock-free asynchronous circuits, the voltage could be scaled down for low-precision calculations so as to reduce the power consumption. The details on how the precision impacts the decoding performance and precision-scalable units implementation are discussed in the following.

8.1.2 Implementation of the Proposed System

As aforementioned, the decoding procedure is operated by the iterations between two basic computation units, VNUs and CNUs. Here, the two units with the precision

reconfigurablility are implemented as asynchronous circuits.

8.1.2.1 Variable node units

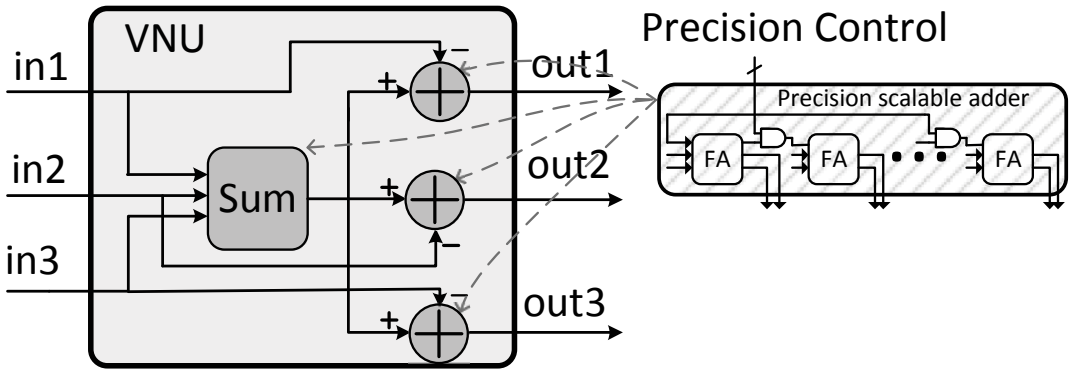


Figure 8.4: Asynchronous precision-salable VNU design.

The variable node units with precision-scalability is presented first. Each VNU has multiple inputs and the same number of outputs. Without losing the generality, Figure 8.4 shows a VNU with 3 inputs and outputs. The inputs are fed the values from the outputs of the CNUs. The VNU generates the outputs by the summations of the input values except the value from the corresponding input with same index, as described in Equation8.1. The number of inputs is as many as 7 in the standard LDPC matrix. To reduce the complexity of the VNU, usually the VNU is designed to sum all inputs together first and then subtract the input value from the summation for the corresponding output. This algorithm is implemented by a adder tree which sums all inputs first and multiple adders to subtract the inputs.

The precision scalability is augmented by designing a precision-reconfigurable adder. As Figure 8.4shows, the carry-in for each full adder is gated by a control signal. The control signal could gate each specific carry-in signal. Therefore, the precision

of adder is adjusted by enabling the specific number of full adders by the control signals. Thus the critical path is reduced for less precision requirement operation. In the asynchronous fashion, there is no clock to constraint the completion. Each stages passes the data to the neighbor by the asynchronous protocol. With different precisions, the delays of the unit are adjusted without any overhead on the clock configuration. Therefore, asynchronous circuits offer more feasibility of the scalability than conventional synchronous circuits.

8.1.2.2 Check node units

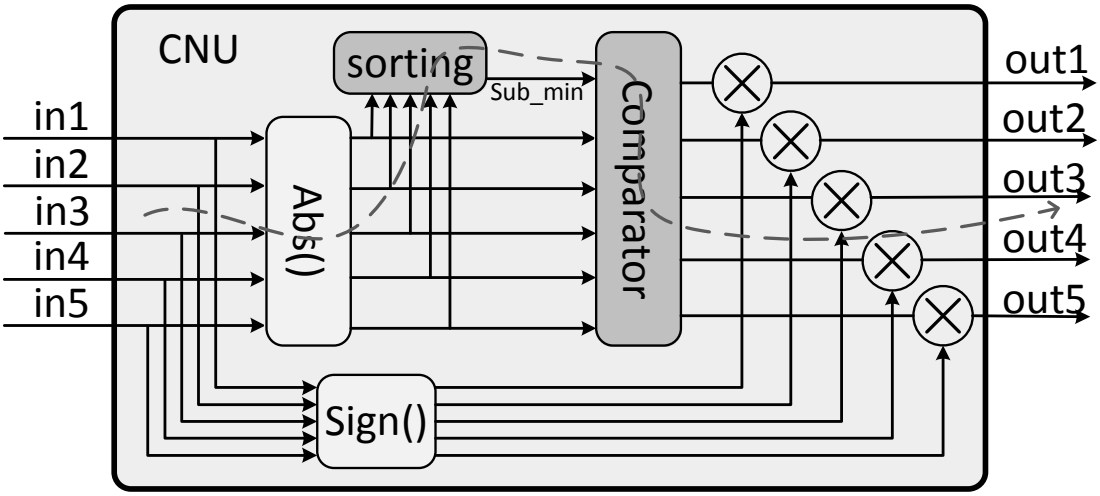


Figure 8.5: Asynchronous precision-salable CNU design

Compared with the VNU, CNU is of greater complexity to execute Equation 8.2. Figure 8.5 shows the architecture of the CNU. The critical path of the CNU is marked by the dashed line in the figure, which consists of the absolute value calculation, a sorting unit and a comparator. In the first operation, the 2's complementary number is translated to sign-magnitude number. Then the magnitude values are

passed to the sorting units to get the minimum and sub-minimum values out of all of the magnitudes. Finally, the minimum and sub-minimum values are selected in the comparator for each output. From the figure, it is shown that the critical path mainly contains the sorting units and a comparator. And sorting unit could be made by multiple comparators. Therefore, to reduce the completion time of the CNU in lower precision, we proposed the precision-scalable comparator for the comparator as well as the sorting operator.

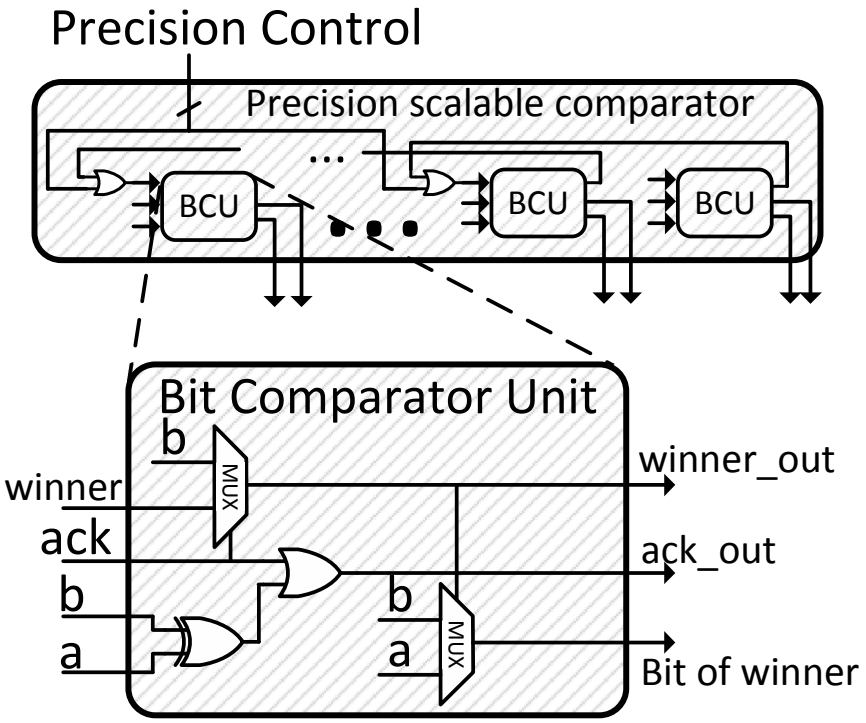


Figure 8.6: Proposed asynchronous comparator

To accommodate the scalability of the precision, the bit comparator units (BCUs) is introduced as the single units in the comparators. As Figure 8.6 shows, the proposed comparator is composed by multiple BCUs which are concatenated together.

The most right BCU corresponds to the MSB, while the left most one is for LSB. For a certain number of precision, the necessary number of BCUs are clipped out by the precision control signal. The critical path of the proposed comparator starts from the right to the left. Since the values after absolute operation are unsigned, each BCU plays role on determine which value of the inputs is greater by checking if two bits are same are not. Also the BCU needs notify the BCU in the next level if the comparison is done. If the comparison is done by the previous stage BCU, the subsequent BCUs output the results correspondingly. The details of the BCU is also given in the figure. Inputs a and b are the two bits of the operators. They are compared by a XOR gate to determine if comparison between the two bits is done in this stage. The ack input is used to acknowledge the BCU that the comparison is done by the previous BCU. The winner input indicates which number has the greater value. Distinct with conventional comparator, since we utilize the asynchronous technique here, the comparator proposed here is not only equipped with ability of precision reconfiguration, it is also designed to get the result as soon as possible by checking from MSB to LSB. If the comparison is done earlier in more significant bits, the critical path for the following BCUs is reduced, so that the computation could be completed earlier without the clock constraints. In the following, the evaluations of the proposed method in terms of computation latency, voltage scaling and power reduction are given.

8.1.3 Simulations and Analysis

To evaluate the proposed method, we first synthesize the precision-scalable computation units by Synopsys Design Compiler on Nangate FreePDK 45nm library. The original design without precision reduction is referenced as the baseline design. For timing analysis of the proposed method, the critical paths under different precision

configurations are extracted by Synopsys Primetime and evaluated by the Synopsys Hspice.

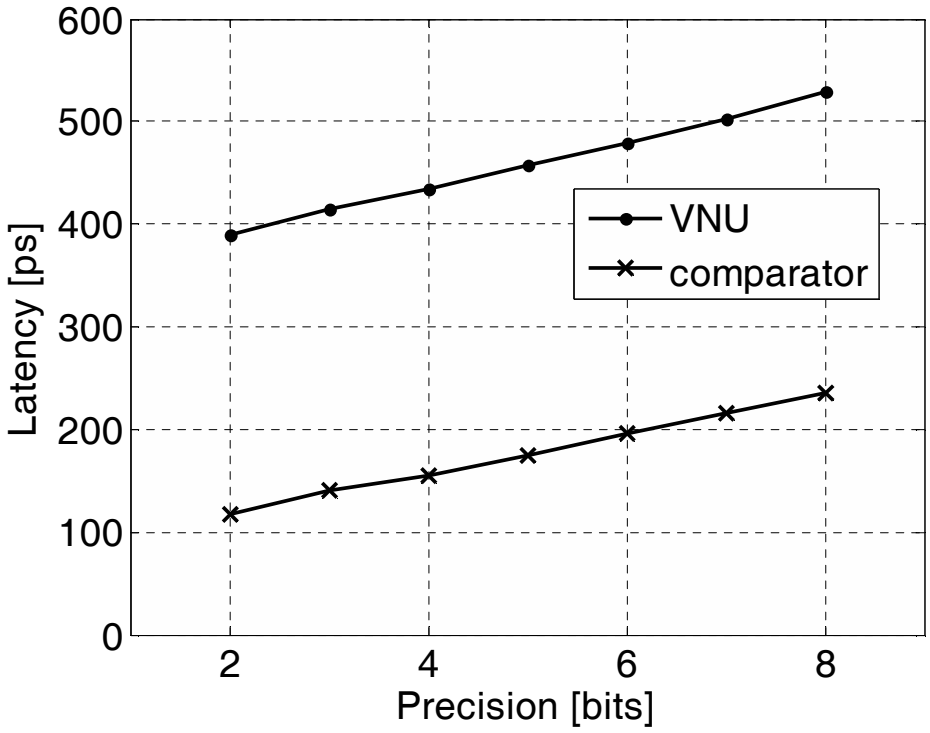


Figure 8.7: Units delays for different bits of precision

First of all, the delays of the proposed units with different precision configurations are evaluated by the SPICE simulation. And the results are shown in the Figure 8.7. As the figure shows, the latency of the computation units are substantially affected by the number of bits involved in the computation. To utilize the time reduction of the lower-precision computation in high-SNR situations, the supply voltage could be tuned down without losing throughput. Because the asynchronous circuits applied, overhead of frequency control is dismissed as we mentioned above.

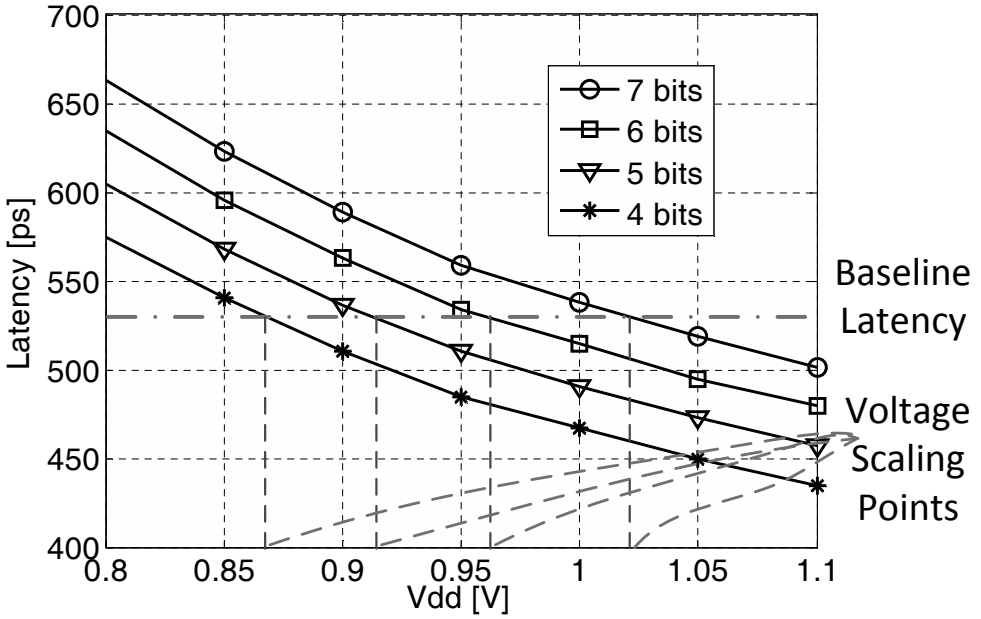


Figure 8.8: Voltage scaling to align processing latency

Here we examined the necessary voltage supplies for computations under different precision without losing the throughput. VNU is taken as an example here to illustrate the voltage scaling. Figure 8.8 shows the latency over the supply voltage of VNU with different precisions. The baseline latency is referenced as the 8-bit full precision. To achieve the same latency, lower voltage supplies are sufficient for those lower-precision computations according to the curves. And the voltage scaling points are indicated as the necessary voltage supply for different bits of precision.

According to the supply voltages reductions examined above, we derived the normalized power reduction to the LDPC decoder running at full precision by the power, $P \propto V_{dd}^2$. Figure 8.9 shows the relatively power consumption compared with full 8-bit precision LDPC decoder. With high SNR environments, 4-bit precision LDPC decoder could be taken to do decoding for a given target transmission reliability with only cost of 49% power consumption as the full-precision running decoder.

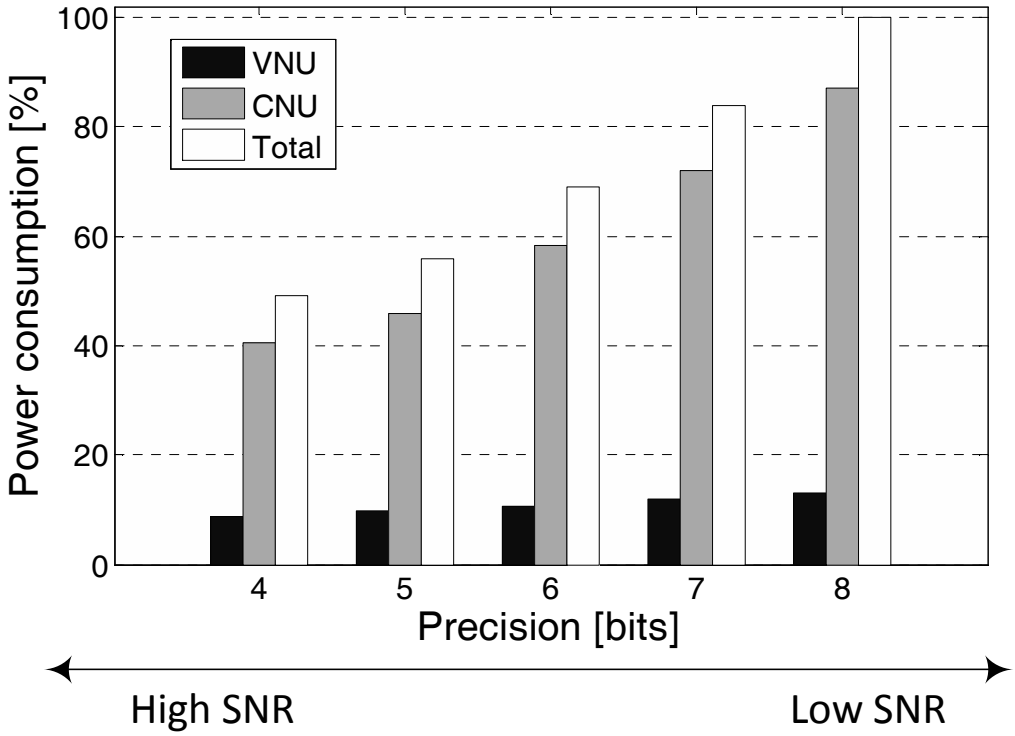


Figure 8.9: Normalized power reduction compared with fixed precision LDPC decoder

8.2 Reconstruction of Compressive Sensing

The essence of decoding for block error correction codes is to search for a original vector given its projection over a tall matrix. Another similar topic based on the problem is compressive sensing reconstruction problem [68].

Conventional sampling theory requires sampling rate be faster than Nyquist sampling rate to avoid aliasing. Compressive sensing (CS) [69, 70] is a technique which allows sampling of sparse or compressible signals efficiently below the Nyquist rate. According to CS theories, a sparse time discrete sparse signal can be exactly recovered by some of its projections over a random basis. The CS theory allows sensor to operate at a much lower sampling frequency than the Nyquist rate when the signal

is sparse enough. Thus CS theory could be used as a framework to reduce sampling rate for ultra-wideband spectrum sensor [71], power-sensitive wireless sensor networks[72] and transmission bandwidth limited sensor terminals. Recent years, efforts have been made on bringing the CS theory to practical implementation. In[72], a circuit is designed to acquire multi-channel data with a single ADC by utilizing CS theory. [73] proposed a circuit implementation of the analog front-end based on CS theory. Although the feasibility of CS was proven by the above circuit design and implementation, few attention has been paid on digital design of CS signal reconstruction. An efficient CS reconstruction design is necessary for real-time applications and power-sensitive back-end processing. Thus a design that can easily instantiate a VLSI implementation for compressive sensing signal reconstruction is desired.

In [74], we emphasizes architectural composition that lends to efficient data path and control for real-time application as well as for low-power implementation approaches.

8.2.1 Compressive Sensing Model

Given a length N vector of time-discrete signal $\tilde{x} \in \tilde{R}^N$, if it can be presented as $\vec{x} = \sum_{i=1}^{N} \psi_i \mathrm{a}_{[i]}$ where ψ_i is i^{th} column of known orthogonal basis and $\vec{a}$ has only K non-zero elements, $\vec{x}$ is called a K-sparse signal. In this paper, we define the occupancy of the signal by the ratio K/N. According to CS theories, $\vec{x}$ could be exactly recovered from M measurements of projections of $\vec{x}$ over a random basis, Φ. The M-long measurements could be written as $\tilde{y} = \Phi\Psi\,\tilde{a}$. The spreading matrix Φ is a random rectangle matrix which has M rows and N columns. The binary elements of ± 1 is usually taken in Φ for easy signal acquisition in reality.

To reconstruct $\vec{x}$ from $\vec{y}$, the K-nonzero-elements vector $\vec{a}$ needs to be estimated

from the following equation:

$$\hat{\tilde{a}} = \arg\, min \|\tilde{a}\|_0 \quad s.t.\ \tilde{y} = \Phi\Psi\tilde{a} \tag{8.3}$$

By $\tilde{x} = \Psi\tilde{a}$, $\vec{x}$ could be reconstructed from the estimated $\hat{a}$.

8.2.2 Reconstruction Algorithm

Given a compressively sensed signal vector $\vec{y}$, the recovery performance does highly depend on the selection of reconstruction algorithms. The literature describes several approaches to solve (8.3). Those approaches can fall into three general categories: convex relaxation, combinational algorithm and greedy pursuits. Each approach has certain advantages as well as inherent shortcomings. Combinational algorithm can swiftly reconstruct data, while it needs unusual structured samples which may not be easy to acquire in reality. Convex relaxation reconstruction can succeed with a small number of measurements. But it tend to be computationally burdensome. Greedy pursuits stand the intermediate position between the two other algorithms in terms of sampling efficiency and reconstruction complexity. However, algorithms of greedy pursuits require a matrix-inverse operation in each iteration, which is expensive in terms of hardware cost.

Iterative hard thresholding (IHT) [70, 75], the algorithm selected in our design, belongs to the set of convex relaxation algorithms. Although IHT has its native bottleneck of computation complexity to perfectly recover sparse signal. With the careful selection on transform matrix Ψ and number of iterations, IHT is a good candidate for hardware implementation, which we will discuss in detail in following.

IHT is a simple and efficient algorithm that iteratively approaches the solution of

(8.3). For a given measurement vector $\tilde{y}$, $\hat{\vec{a}}$ could be found by the following iterations:

$$\vec{a}_{[i+1]} = \mathrm{H}_K\left(\vec{a}_{[i]} + \mathbf{B}^T(\vec{y} - \mathbf{B}\vec{a}_i)\right) \tag{8.4}$$

where $\mathbf{B} = \Phi\Psi$ and $\mathrm{H}_K(\vec{z})$ is a non-linear operation which set all elements except the largest K ones of vector $\vec{z}$ to zero. The computational bottleneck of IHT is at the operator matrix multiplication $\mathbf{B}$ and $\mathbf{B}^T$. At first, the two matrix operations at each iteration appears too complicated to be a good choice for hardware implementation. However, a closer examination of the equations reveals that each matrix multiplication could be decomposed into two easy hardware-friendly computations:

$$\mathbf{B} = \Phi\Psi, \mathbf{B}^T = \Psi^T\Phi^T \tag{8.5}$$

If Ψ is selected as a structured operator such as Fourier transform, $\Psi\tilde{a}$ could be computed as fast Fourier transform (FFT) in an efficient manner in hardware implementation. And Ψ^T operation is nothing but the inverse fast Fourier transform (IFFT) with a constant scale. The binary matrix Φ multiplication could be easily implemented by some multiple-stage adders.

8.2.3 Modification on IHT

As discussed above, the IHT algorithm is suitable for efficient hardware implementation due to its simple straightforward computation in each iteration. As shown in Eq. (8.4), the largest K elements need to be picked out in each iteration. While in realistic situation, the number of K is unknown at first. It implies that an estimation of occupancy is necessary for IHT reconstruction, as well as for greedy search algorithms.

To eliminate dependence on knowledge of sparsity, here we propose a modifi-

cation to IHT. In our modified-IHT (MIHT) algorithm, the sparse signal can be reconstructed without knowing the sparsity. IHT iteration function (8.4) is modified as follows:

$$\vec{a}_{[i+1]} = \tilde{\mathrm{H}}_{r_{[i]}} \left(\vec{a}_{[i]} + \mathbf{B}^T (\vec{y} - \mathbf{B}\vec{a}_{[i]}) \right) \tag{8.6}$$

where $\tilde{\mathrm{H}}_{r_{[i]}}(\vec{z})$ reserve all elements whose Norm-2 values are not less than $r_{[i]} \cdot \max \|z_j\|$, $z_j \in \vec{z}$:

$$\text{if } \tilde{p} = \tilde{\mathrm{H}}_{r_{[i]}}(\tilde{z}), \quad p_j = \begin{cases} 0 & \|z_j\| < r_{[i]} \cdot \max \|z_j\| \\ z_j & else \end{cases} \tag{8.7}$$

where j is the index of the element. $r_{[i]}$ is fraction number between 0 and 1 . The value of $r_{[i]}$ initially is 1 and does decrease monotonously with the number of iterations. The reason we propose this strategy based on observation on IHT simulation experiments: If we define the estimation error as $err = \mathbf{B}\vec{a}_{[i]} - \vec{y}$, each iteration does nothing but reducing the estimation error by zero-forcing $-err$ on $\vec{a}$ itself. In the proposed algorithm, initially only frequency components with significant contribution are considered, i.e. $r_{[i]}$ is close to one. With additional iterations, the $\vec{a}$ starts to become increasingly sparse. Meanwhile, the value of threshold ratio of $r_{[i]}$ is sweeping down to select more and more components. To acquire an efficient convergence, our threshold function is selected based on an insight drawn from the results reported in [76]: the upper error bound of IHT, $\|\vec{a} - \hat{\vec{a}}\|$, does decrease exponentially with number of iteration i. Following exponential function is taken as our threshold function:

$$r_{[i]} = t_0^{-i} \tag{8.8}$$

To evaluate reconstruction fidelity of the modified algorithm, we introduce the

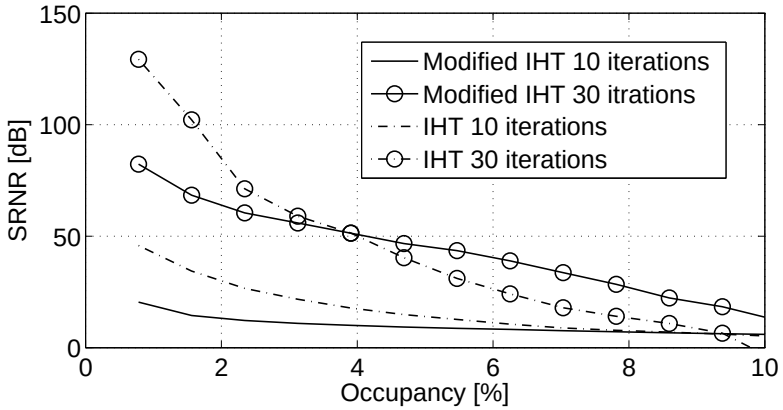


Figure 8.10: Comparison between modified IHT with $t_0 = 0.82$ and original IHT at 25% Nyquist sampling rate

definition of signal-to-reconstruction-noise ratio (SRNR) here:

$$\text{SRNR}\left(\vec{x}, \hat{\vec{x}}\right) = 20\log_{10}\left(\frac{\|\vec{x}\|}{\|\vec{x} - \hat{\vec{x}}\|}\right) \tag{8.9}$$

where $\vec{x}$ is the input signal vector and $\hat{x}$ is the recovered data. The recovery fidelity of the modified algorithm with comparison of original IHT algorithm is shown in Figure 8.10. In the figure, the input signals contain K frequency tones out of 256-point Fourier transform frequency components. The position of K active tones are randomly chosen. The amplitude of the K active frequencies are i.i.d. Gaussian. As Figure 8.10 shows, the modified algorithm is working but the convergence speed of modified one is slower than the original IHT algorithm. The selections of coefficient t_0 and number of iterations are discussed at the following Section, where the implementation details are also discussed.

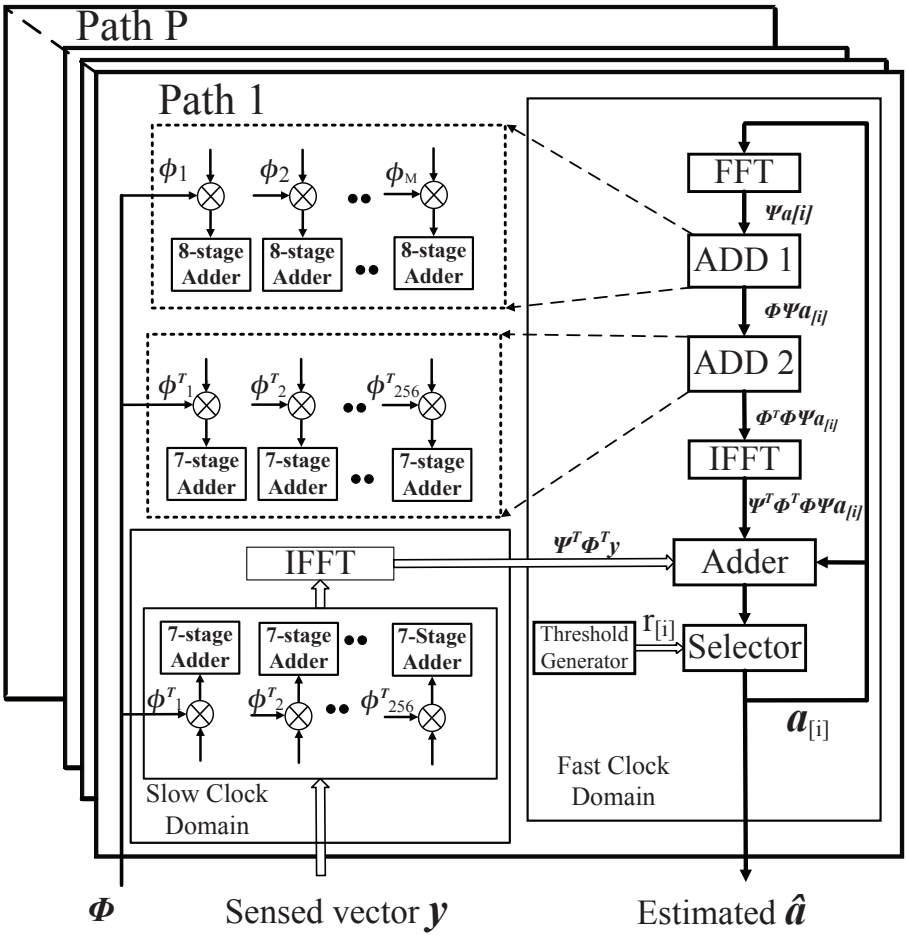


Figure 8.11: Parallel architecture for compressive sensing reconstruction

8.2.4 Implementation of Proposed Algorithm

The modified reconstruction algorithm could be in implemented in a parallel architecture as shown in Figure 8.11. Each path processes the problem (8.6) in an iterative fashion. The problem (8.6) could be rewritten as:

$$\tilde{a}_{[i+1]} = \tilde{\mathrm{H}}_{r_{[i]}} \left(\Psi^T \Phi^T \tilde{y} + \tilde{a}_{[i]} - \Psi^T \Phi^T \Phi \Psi \tilde{a}_{[i]} \right) \tag{8.10}$$

where $\Psi^T\Phi^T\Phi\Psi\tilde{a}_{[i]}$ is calculated in each iteration. As Figure 8.11 shows, each path starts the computation by feeding $\vec{a}_{[i]}$ into FFT block. For frequency-domain sparse signal, $\Psi\vec{a}_{[i]}$ and $\Psi^T\vec{a}_{[i]}$ is implemented efficiently by FFT and IFFT blocks. Φ^T and Φ are random matrix with binary content(1 or -1), which could be implemented by multiple stage adders. Thus the FFT is followed by M 8-stage adders, the ADD1 block shown in the Figure 8.11, to compute $\Phi\Psi\tilde{a}_{[i]}$. M is the number of the measurements in compressive sensing. The m^{th} adder calculates sum of $\sum_{j=1}^{256}\phi_{mj}d_j$, where d_j is the j^{th} elements of the input and ϕ_{mj} is the binary elements on m^{th} row and j^{th} column. The multiplications with -1 is carried out by two's complementary operations. Subsequently, $\Psi^T\Phi^T\Phi\Psi\tilde{a}_{[i]}$ is computed by another multiple-stage adders and an IFFT block.

Although $\vec{a}$ is updated in each iteration, the term $\Psi^T\Phi^T\tilde{y}$ is calculated once for each reconstruction. So the $\Psi^T\Phi^T\tilde{y}$ could be derived in a slow clock speed as shown in Figure 8.11 for low-power optimization.

Multiple three-input adders are used to sum $\Psi\vec{a}_{[i]}$, $\Psi^T\Phi^T\tilde{y}$ and $\Psi^T\Phi^T\Phi\Psi\tilde{a}_{[i]}$ together. The threshold ratio is generated by the threshold generator. For exponential function, a shift adder is used in the generator as the accumulative multiplier. The Selector block in Figure 8.11 generate $\Psi\vec{a}_{[i+1]}$ by resetting all elements less than the threshold.

If the signal is sparse enough, IHT is able to do exact data recovery with sufficient iterations[76]. The modified algorithm shows the similar property in our experiment. However, the value of t_0 in threshold function affect the fidelity considerably. Here, we show how values of t_0 is related with the number of iterations. And a set of reasonable parameters is selected for target specifications.

Figure 8.12 shows, with 20 iterations, different values of factor t_0 result in different reliability on reconstruction. $t_0 = 0.75$ is shown as the coefficient which provides the

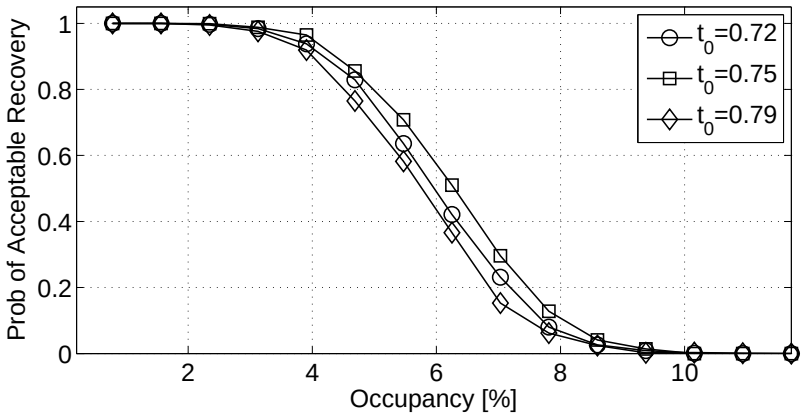


Figure 8.12: Successful recovery rate with different threshold function coefficient t_0 at 20 iterations

best fidelity out of all three t_0 values shown in the figure. We could also see that, given a certain combination of iterations and t_0, the proposed algorithm could reconstruct signals with unknown and vary sparsity which is less than a certain threshold.

For the efficiency of the proposed design, a fine search on number of iterations is performed to look for the minimum number of iterations that yield maximum SRNR of 44 dB SRNR (7 ENOBs) at target occupancy, 4%. We set up the experiments for each pair of iterations and t_0 to explore the relationship between recovery reliability and number of iterations. Figure 8.13 shows how t_0 affects the recovery fidelity. We see that, basically raising the threshold vale t_0 could result in a higher SRNR, while also leading to have more number of iterations. To optimize our design, given a certain number of iterations, t_0 is chosen as the one who can achieve highest SRNR. For example, when 18 iterations are taken to reconstruct data, the best coefficient t_0 should be selected as 0.79, which is able to provide SRNR of 38.67. The extensive characterization on number of iterations is summarized in Table 8.1. To be aligned with the front-end design [73], we target our SRNR as 44 dB. As Table 8.1 shows,

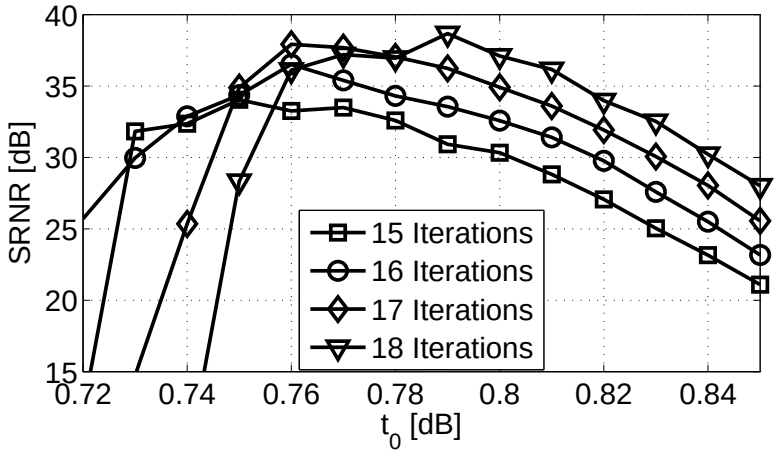


Figure 8.13: SRNR with different threshold function coefficient t_0

22 is the least number of iterations to reconstruct the signal to our expected SRNR level. Thus 22 iterations along with $t_0 = 0.80$ is applied in our design.

Table 8.1: SRNR with different numbers of iterations

Iterations	t_0	SRNR (dB)	Iterations	t_0	SRNR (dB)
15	0.75	34.04	19	0.78	41.07
16	0.76	36.61	20	0.79	43.04
17	0.76	37.91	21	0.79	42.98
18	0.79	38.67	22	0.80	44.91

8.2.5 Simulations and Analysis

To implement the architecture with above parameters, a fixed-point model is first set up in Matlab. Corresponding register-transistor-level (RTL) design is implemented in Verilog. RTL implementation is functionally verified with output from that of fixed-point model. Then RTL is synthesized using *Synopsys Design Compiler* with *TSMC* 45*nm* technology standard cell library.

Table 8.2: CS reconstruction design power consumption

Power consumption @ 88 MHz	FFT	41.08 mW
	ADD1	30.3 mW
	ADD2	51.28 mW
	IFFT	36.48 mW
	Adder + Selector	5.647 mW
	Overall	165 mW

To determine the parallelism and the frequency, one single path of the proposed architecture is implemented and synthesized. The critical path has the delay of 4.05 ns. To target the equivalent sampling speed of 1 GSPS, the parallelism is determined by:

$$P > \frac{\mathrm{n} \cdot \mathrm{Throughput}}{N \cdot f_f} \tag{8.11}$$

where n is the number of iterations necessary for reconstruct sparse signal. Power consumption of the proposed system is summarized in Table 8.2. Our design consumes 165 mW at 88 MHz. It takes 22 cycles, latency of $0.25\mu s$ to reconstruct one set of samples.

In this work, an exploration on digital circuits design of compressive sensing reconstruction is provided. An modification to a hardware-friendly algorithm is made to adapt unknown and varying degree of sparsity of signals. A corresponding iterative architecture is given for the modified algorithm. An existing analog front-end compressive sensor is referenced as interface specification for designing implementation parameters. The design parameters are studied empirically. We implemented the fixed-point model in RTL coding, which can reconstruct compressive sensed 4%-occupancy sparse signal of 7 ENOBs by SRNR of 30 dB with consuming 165 mW as equivalent Nyquist Sampling rate of 1 GSPS.

Current effort is being paid on several questions such as higher reconstruction

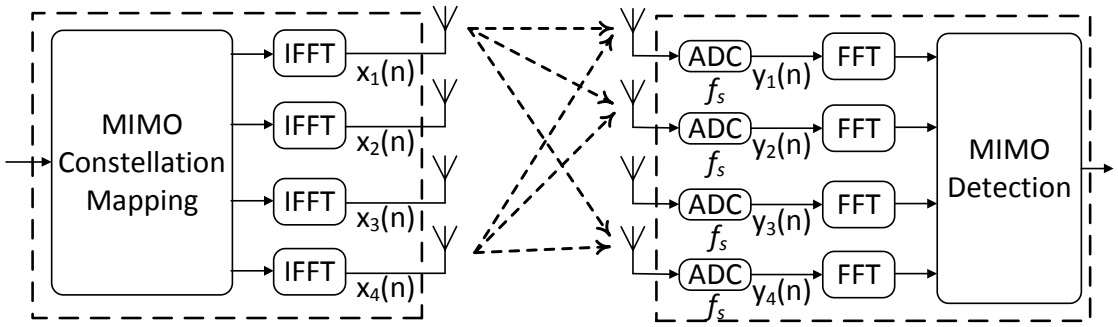
fidelity, higher recoverable occupancy and low-power exploration on circuits implementation to make proposed system applicable to commercial devices.

8.3 Compressive Sensing on MIMO-OFDM Cognitive Radio

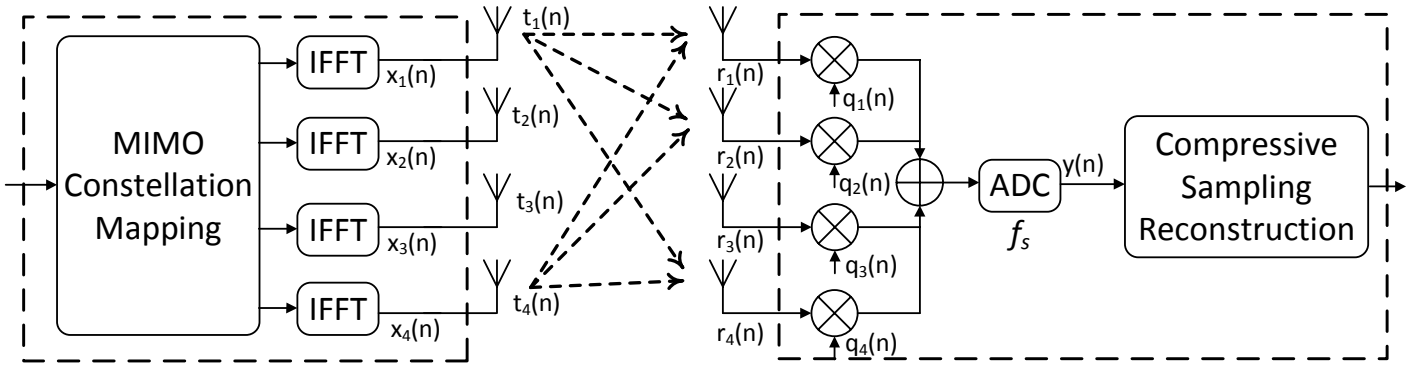
According to CS theories, a time-discrete sparse signal can be exactly recovered by some of its projections over a random basis. Thus CS theory could be utilized to reduce sampling rate for ultra-wideband spectrum sensor [71], power-sensitive wireless sensor networks and transmission bandwidth limited sensor terminals. In [77–79], applications of CS on spectrum sensing and reception for conventional CR network are explored. While, to the best of our acknowledgement, no attention has been paid on the application of CS on CR system with MIMO, which is an enabled technique in most current current and emerging wireless standard. From the aspect the development of CS technique, recently more and more effort has been paid on bringing CS from theory to practical circuit implementation. [72] proposed a efficient CS-based analog to digital converter (ADC) for multi-channel signals sampling. In reference to the multi-channel ADC [72], the question we address in this paper is whether or not we can reduce the hardware cost of MIMO-OFDM based CR receivers with a new architecture based on CS theory.

8.3.1 Proposed System

Figure 8.14 shows the both structures for the ordinary and proposed system of 4×4 MIMO scheme, while the number of antennas could be scaled. The overall architecture of the proposed MIMO-OFDM based CR receiver exploiting CS is shown in Figure 8.14b. To compare with the conventional one, the equivalent traditional MIMO-OFDM transceiver is also illustrated in Figure 8.14a. Assuming the MIMO-OFDM based CR signal in each antenna is sparse, which means only limited number of subcarriers are in the use, the signal is able to be received by the



(a)



(b)

Figure 8.14: Structures of conventional (a) and proposed (b) MIMO-OFDM transmission system.

specific novel reception architecture. Different from the traditional MIMO-OFDM receiver, the proposed one first randomly modulates and sums the signals together. Then the signal is sampled by a single ADC, who has the same rate same as the ADCs in the ordinary architecture. The DSP block for the proposed system replaces the conventional DSP blocks consisting of FFT units for each channel and MIMO detector. Instead of transforming the signals on timing domain to Fourier domain, the CS reconstruction algorithm directly separates and recoveries ths signals on the frequency domain where the signal represents sparsity. The mathematics model of the proposed system and solution will be discussed in the next subsection.

8.3.1.1 Modulating signals with random sequences

The value of the pseudo-random sequence for modulation should be selected to be hardware-friendly. In this paper, we choose $q_i(n)$ as random binary value (± 1) to modulate signals. In practice, the pseudo-random random sequence could be generated by linear feedback shift registers (LFSR).

8.3.1.2 DSP block

In the proposed system, the DSP block plays the role on separating the compressively sampled data. In conventional CS reconstruction problem, there are multiple algorithms proposed for solving this problem such as l_1 linear/convex optimization [74], greedy algorithms [80] and approximate message passing (AMP) [81]. Without knowing any spectrum utilization information such like the occupancy and locations of the occupied frequency points, those ordinary algorithms suffer from their burdensome computational complexity and undetermined number of iterations. However, in the proposed OFDM based communication system, the receiver could easily have the knowledge of utilization of spectra by information synchronization before the transmission or spectrum sensing by FFT. Thus here we give a straightfor-

ward algorithm utilizing the information about the occupancy of the OFDM channel to separate and reconstruct signals in following.

8.3.2 Sparse Signal Model

In this section, the model of the proposed system is derived. The background of conventional MIMO wireless communication is first breifly introduced. The proposed mathematical model is derived in following.

8.3.2.1 Conventional MIMO-OFDM model

Figure 8.14a shows a equivalent baseband model for a MIMO-OFDM system. Assuming there are N_f subcarriers in each channle, the nth output of the IFFT on the ith antenna is $x_i(n)$ could be expressed as:

$$x_i(n) = \frac{1}{\sqrt{N_f}} \sum_{m=0}^{N_f-1} a_i(m) \exp(j2\pi nm/N_f) \tag{8.12}$$

, where $a_i(m)$ is the complex modulated signals on subcarrier on ith transmit antenna. When a user tries to access a MIMO-OFDM channel, some of N_f subcarriers would be assigned to the user for the duration of usage. The transmitted signal could also be described in a vector-matrix format as follows:

$$\boldsymbol{x}_i = \boldsymbol{F}_{N_f}^H \boldsymbol{a_i} \tag{8.13}$$

Then the signals are transmitted over multiple antennas. Let us denote the channel impulse response (CIR) vector $\boldsymbol{h_{ij}}$ between the ith transmit antenna and jth receive antenna as $\boldsymbol{h_{ij}}$. Suppose that there are L multipath between the transmitter and

receiver, $\boldsymbol{h_{ij}}$ could be presented as:

$$\boldsymbol{h}_{ij} = \left[h_{ij,0}, \ldots, h_{ij,L-1} \right]^T \tag{8.14}$$

The complex baseband equivalent received signal is the transmitted signal convoluted by the CIR. To protect from the intersymbol interference (ISI), the cyclic prefix (CP) is added before each transmit vector $\boldsymbol{x_i}$. Thus the received signals on jth receive antenna from ith transmit antenna could be expressed as vector-matrix format as:

$$\boldsymbol{y}_{i \to j} = \overline{\boldsymbol{H}}_{ij} \boldsymbol{x}_i \tag{8.15}$$

where $\boldsymbol{y}_{i \to j}$ is the signals received by jth receive antenna from ith transmit antenna and $\overline{\boldsymbol{H}}_{ij}$ is a matrix of the operation of the cyclic convolution with the CIR vector $\boldsymbol{h_{ij}}$ as shown below:

$$\begin{bmatrix} h_{ij,0} & \ldots & 0 & 0 & \ldots & h_{ij,1} \\ \vdots & \ddots & 0 & \vdots & 0 & h_{ij,L-1} \\ h_{ij,L-1} & \ldots & h_{ij,0} & 0 & \vdots & 0 \\ 0 & \ddots & \ldots & h_{ij,0} & \vdots & \vdots \\ \vdots & & \ddots & \vdots & \ddots & 0 \\ 0 & \ldots & 0 & h_{ij,L-1} & \ldots & h_{ij,0} \end{bmatrix}$$

For a $N_t \times N_r$ MIMO-OFDM system (N_t and N_r are the number of transmit and receive antennas correspondingly), the receive vectors could be expressed as below:

$$\begin{bmatrix} \boldsymbol{y_1} \\ \vdots \\ \boldsymbol{y_{N_r}} \end{bmatrix} = \begin{bmatrix} \overline{\boldsymbol{H}}_{11} & \ldots & \overline{\boldsymbol{H}}_{N_t 1} \\ \vdots & \ddots & \vdots \\ \overline{\boldsymbol{H}}_{1 N_r} & \ldots & \overline{\boldsymbol{H}}_{N_t N_r} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_1} \\ \vdots \\ \boldsymbol{x_{N_t}} \end{bmatrix} + \boldsymbol{V} \tag{8.16}$$

where $\boldsymbol{V}$ is the additive white noise. For the convenience, we denote the above channel matrix as $\boldsymbol{H}$. The above equation could be represented as, $\boldsymbol{y} = \boldsymbol{Hx} + \boldsymbol{V}$. To measure the performance of our proposed work, in this paper, the definition of signal-to-noise ratio (SNR) for the wireless channel is defined as:

$$\text{SNR} = 20\log_{10}\frac{\parallel \boldsymbol{Hx} \parallel_2}{\parallel \boldsymbol{V} \parallel_2} \quad \text{dB} \tag{8.17}$$

In the conventional MIMO-OFDM problem, instead of solving (8.16), the detection will be accomplished by first applying FFT to transform received signals on Fourier domain to simplify as a diagonal matrix. Here we denote $\boldsymbol{h}'_m$ as the channel gain matrix for the symbols on mth subcarrier, whose element, $h'_{ij,m}$, is the gain between jth transmit antenna and ith receive antenna. Then the conventional MIMO detector demodulates the transmitted symbols by solving the following equations:

$$\begin{bmatrix} y'_1(m) \\ \vdots \\ y'_{N_r}(m) \end{bmatrix} = \begin{bmatrix} h'_{11,m} & \dots & h'_{N_t1,m} \\ \vdots & \ddots & \vdots \\ h'_{1N_r,m} & \dots & h'_{N_tN_r,m} \end{bmatrix} \begin{bmatrix} a'_1(m) \\ \vdots \\ a'_{N_t}(m) \end{bmatrix} + \boldsymbol{V'} \tag{8.18}$$

where $\boldsymbol{V'}$ is the corresponding additive noise on the subcarrier, $y'_1(m)$ is the signals after FFT on the mth subcarrier. An exhaustive search on all possibilities of $\boldsymbol{a}\,(m)$ could find $\hat{\boldsymbol{a}}(m)$, which minimizes $\|\boldsymbol{y'}(m) - \boldsymbol{h}'_m\hat{\boldsymbol{a}}(m)\|$. This solution $\hat{\boldsymbol{a}}(m)$ is also called maximum likelyhood (ML) solution for MIMO detection. However, ML suffers from its high computation complexity. Instead of using ML, zero-forcing (ZF) algorithm or other trade-off tree-search algorithms [82] are used in practice.

Instead of using the conventional method, the detection problem and method to the problem of the proposed system is described in the following.

8.3.2.2 Sparse signal model for the proposed system

As aforementioned, the baseband signal sent on each antenna is a superposition of a number of subcarriers, which are column vectors of the N_f-point DFT matrix $\boldsymbol{F}_{N_f}$ as shown at (8.13).

Subsequently, the received signals $\boldsymbol{y}_j$ could be presented as (8.16). In our proposed system, the received signals are modulated with random sequences $\boldsymbol{q}_j$ again and summed together by a single ADC, the sampled data $\boldsymbol{y}_{cs}$ could be expressed as:

$$\boldsymbol{y}_{cs} = \sum_{j=1}^{N_r} \boldsymbol{q}_j \boldsymbol{r}_j \tag{8.19}$$

where $\boldsymbol{q}_j$ is a vector, whose elements are random binary number, ± 1. q_i is used to modulate the signal from the ith receive antenna. This equation could also be presented in a vector-matrix format as below:

$$\boldsymbol{y}_{cs} = \begin{bmatrix} \overline{\boldsymbol{Q}}_1 & \overline{\boldsymbol{Q}}_2 & \dots & \overline{\boldsymbol{Q}}_{N_r} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_{N_r} \end{bmatrix} \tag{8.20}$$

where $\overline{\boldsymbol{Q}}_j$ is a diagonal matrix with the random binary number on the diagonal. As (8.20) shows, using the flat matrix consisting of multiple diagonal matrixs, the total number of samples acquired by the receiver is reduced by the factor of the number of receive antennas. Here we denote the above flat matrix as $\boldsymbol{Q}$. Thus the sampled

data could be represented as:

$$\begin{aligned} \boldsymbol{y}_{cs} &= \boldsymbol{QH} \begin{bmatrix} \boldsymbol{F}_{N_f}^H & \dots & \boldsymbol{0} \\ \vdots & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \dots & \boldsymbol{F}_{N_f}^H \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_1 \\ \vdots \\ \boldsymbol{a}_{N_r} \end{bmatrix} + \boldsymbol{n}_0 \\ &= \boldsymbol{QHF} \cdot \boldsymbol{\alpha} + \boldsymbol{n}_0 \end{aligned} \tag{8.21}$$

where $\boldsymbol{a}_i$ is the $N_f \times 1$ OFDM vector on the ith transmit antenna and $\boldsymbol{a}_i$ is sparse when the subcarriers are rarely occupied. $\boldsymbol{F}_{N_f}$ is the FFT matrix as aforementioned. Furthermore, we define $\boldsymbol{B}$ as $\boldsymbol{QHF}$ and $\boldsymbol{\alpha}$ as the concatenated vector from $\boldsymbol{a}_i$. $\boldsymbol{n}_0$ is the equivalent noise after randomly modulation and mixture. Now we can arrive the question to our model that detecting MIMO-OFDM symbols $\boldsymbol{\alpha}$ by much less samples:

$$\hat{\boldsymbol{\alpha}} = \arg\min \parallel \boldsymbol{y}_{cs} - \boldsymbol{B\alpha} \parallel, \text{ s.t. } \alpha_i = 0, \ i \in \Pi_{idle} \tag{8.22}$$

where a_i is the ith element of the vector $\boldsymbol{\alpha}$, Π_{idle} is the set of indices where the subcarriers are not occupied. Compared with conventional receivers, the mixed-signal block is relaxed by the reduction on number of signal acquisitions. For the DSP part, the traditional detector units are replaced by a unit could solve the equation (8.22). Different with ordinary CS problem, (8.22) provides additional constraint on the positions of idle subcarriers. In [83], we also propose a straightforward algorithm to reconstruct the data with utilization of the additional constraints. The reconstruction algorithm is given in the following subsection.

8.3.3 Reconstruction

In the proposed architecture, the randomly modulated signals are mixed together and sampled. Given a compressed signal vector $\boldsymbol{y}_{cs}$, $\boldsymbol{\alpha}$ needs to be recovered where the

information is carried. To solve question (8.22), $\hat{\boldsymbol{a}}$ is intuitively found by the linear combination of some column vectors out of $\boldsymbol{B}$ as close as possible to $\boldsymbol{y}_{cs}$. Without effective occupancy information, traditional CS algorithms suffer from undetermined number of iterations and excessive complexity. Since in the proposed scenario, the band usage of the communication channel could be acquired by pre-information or FFT for OFDM-based channels. In this paper, we try to utilize the occupancy information in order to simplify the reconstruction processes compared with those ordinary algorithms.

First of all, we denote Π_{busy} as the set of all indices, where subcarriers are occupied, e.g. $\boldsymbol{\alpha}_i \neq 0,\ i \in \Pi_{busy}$. And also $\boldsymbol{b}_i$ is defined as the ith column vector of $\boldsymbol{B}$. As aforementioned, to solve the question (8.22) is essential to find the closest linear combination of the some column vectors to $\boldsymbol{y}_{cs}$. In the scenario in this paper the set of active subcarriers, Π_{busy}, is known. Thus we pick those columns $\boldsymbol{b}_i,\ i \in \Pi_{busy}$ from $\boldsymbol{B}$ which are corresponding the positions of the busy subcarriers indices to form a new matrix. Noticeably, because that the position of non-zero entries are unknown in the conventional CS sensor, the traditional CS reconstructors such as orthogonal matching pursuit (OMP) have to iteratively pursue those bases which contribute the sensed signal. Thus the iteration does considerably aggregates the complexity in the ordinary reconstruction algorithms. In our proposed reconstruction, $\boldsymbol{B}_\Pi$ is introduced as a matrix, whose columns are consisting of $\boldsymbol{b}_i,\ i \in \Pi_{busy}$. Then the pseudo-inverse of $\boldsymbol{B}_\Pi$, is calculated and applied to project the sampled data $\boldsymbol{y}_{cs}$ to $\hat{\boldsymbol{\alpha}}_\Pi$ as below:

$$\hat{\boldsymbol{\alpha}}_\Pi = \boldsymbol{B}_\Pi^\dagger \cdot \boldsymbol{y}_{cs} \quad \text{s.t. } \boldsymbol{B}_\Pi^\dagger = (\boldsymbol{B}_\Pi^T \boldsymbol{B}_\Pi)^{-1} \boldsymbol{B}_\Pi^T \tag{8.23}$$

where $\hat{\boldsymbol{\alpha}}_\Pi$ is the reconstructed signals on those busy subcarriers. The length of the vector $\hat{\boldsymbol{\alpha}}_\Pi$ is equal to the number of elements in Π_{busy}. The values of elements in

$\hat{\boldsymbol{\alpha}}_{\Pi}$ are mapped to the elements of $\hat{\boldsymbol{\alpha}}$ with corresponding indices in Π_{busy}. By slicing the reconstructed symbols, the sent bit stream could be demodulated out. The performance based on the proposed architecture is discussed in the following section.

8.3.4 Setup and Simulation

To show the performance of our proposed system structure, we set up the simulation for our proposed model in MATLAB. In the simulation, we suppose that there are $N_f = 256$ subcarriers for users over the given channel bandwidth. Different MIMO schemes of 2×2, 3×3 and 4×4 are simulated. For each scheme, simulations are run for different occupancy of the OFDM channel. The performance is evaluated by the successful reconstruction rate (SRR) and the bit error rate (BER). The performance comparisons with the conventional MIMO detectors such as ML and ZF detector are also presented. In our simulation, we also assume that there are three fading paths between transmitter and receiver and the channel state information (CSI) of the channel is perfectly known.

8.3.4.1 Reconstruction fidelity

First of all, the sparse reconstruction algorithm introduced mentioned above is tested for different occupancy. SRR is defined as ratio of the number of correctly reconstructed symbols over the total number of symbols transmitted. Figure 8.15 in general shows how the occupancy of the OFDM wireless channel affects SRR. In the experiment, the symbols are modulated by 16QAM and the SNR is 15 dB. As shown in the figure, the successful reconstruction rate is degraded as more number of subcarriers are occupied. We also scale up MIMO system from 2×2 to 4×4 in order to evaluate the scaling effect. We find that the MIMO system of the moderate scale is more resilient to the increase of occupancy. For example, less than 44 subcarriers out of 256 total subcarriers could be achieved over 95% SRR at 2×2 MIMO scheme.

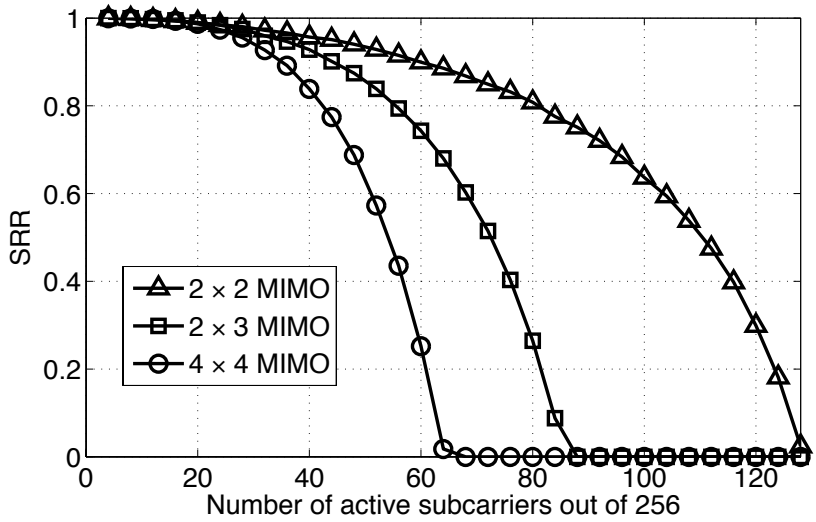


Figure 8.15: Successful reconstruction rate at 15 dB SNR

However, to target same SRR, only 28 active tones could be carried at 4×4 MIMO scheme. This is because that we mix signals from all receive antennas together. At same level of the channel occupancy, larger scale MIMO increases the crowdedness of the mixed signal.

To further evaluate the fidelity of detection in wireless channel, Figure 8.16 shows the detection capability of the proposed architecture for 4×4 MIMO scenario. In the figure, the numbers in legend represent the amount of active subcarriers. It is shown in Figure 8.16 that less occupancy signals could be demodulated with higher reliability. For instance, to achieve BER of 10^{-3}, the MIMO-OFDM signals with 4 active subcarriers outperforms the one with 8 active subcarriers by 3 dB. Moreover, with the occupancy of the channel increasing, the BER gap between signals of different occupancy is reducing. For example, to target same BER level, there is only 1 dB gap between the signal with 20 and 24 subcarriers.

From Figure 8.16, we also show that the proposed architecture can successfully

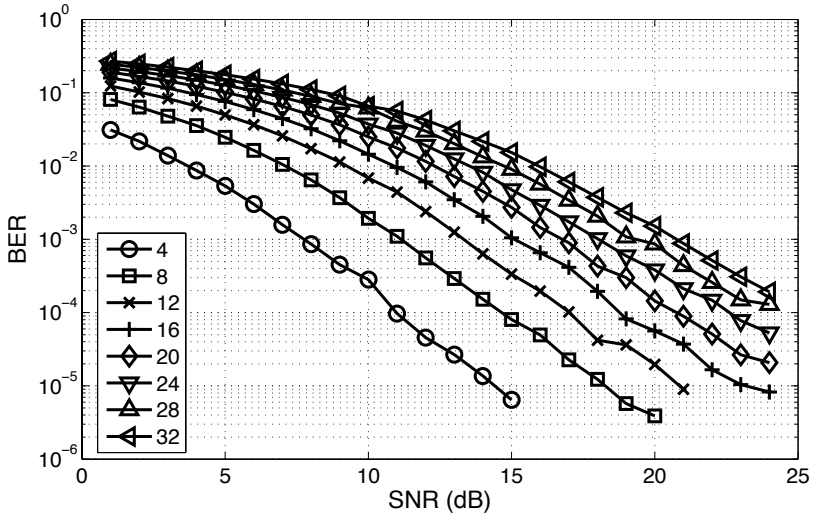


Figure 8.16: Detection performance of the proposed system at 4×4 MIMO

detect signals which has 32 subcarriers at 4×4 MIMO system.

8.3.4.2 Different scales of MIMO system

To examine the impact of the scale of MIMO on the detection fidelity, the detection reliability over different scales of MIMO is simulated. Figure 8.17 shows the reconstructed BER in relationship with the number of busy subcarriers in the channel over different MIMO schemes. From the figure, we show that in very limited occupied OFDM channel, larger scale of MIMO has the diversity to provide higher wireless communication fidelity. But as the occupancy of the OFDM channel increasing, the detection ability is gradually degraded. Larger scale of MIMO has more diversity over the MIMO channel, however, where the sampled signals are more crowded since signals from all receive antennas are summed together. Thus at some point, the diversity of MIMO is beat by the crowd, where larger scale of MIMO fails to provide transmission as reliable as smaller scale does.

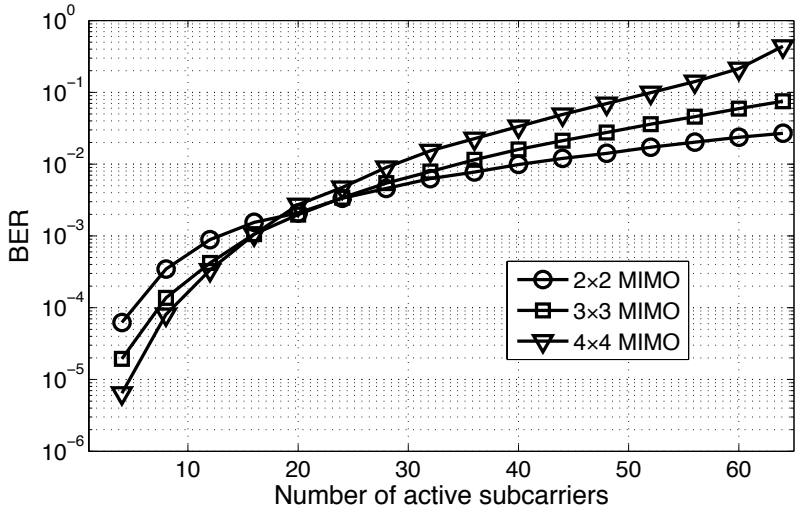


Figure 8.17: Detection performance of different MIMO scales at 15 dB SNR

8.3.4.3 Comparisons with conventional MIMO receiver

To compare our proposed CS-based detector with the conventional MIMO detectors, we present the ML and ZF detection performance with ours in Figure 8.18, where 3×3 MIMO is applied. The legends of the figure show the number of occupied subcarriers as well as the detection method, where CS, ML and ZF indicate the proposed receiver, conventional ML detection and ZF detection respectively. As shown in the figure, the proposed receiver outperforms the ZF detector substantially. And with lightly occupied MIMO-OFDM channel, the proposed architecture performance is close to that of the ML detection. When the number of active subcarriers increases, the proposed performance is degraded more dramatically than the other two. However, with 32 active subcarriers, the proposed detector still outperform the ZF detector. This phenomenon is explained before as that increasing occupancy on the channel will results in more crowded compressive sensed signals which is more difficult to reconstruct.

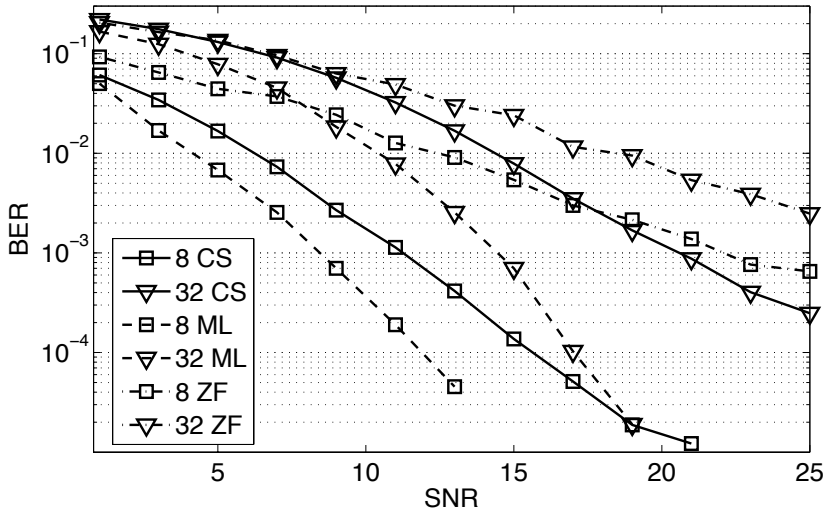


Figure 8.18: The performances comparisons with conventional MIMO detection

In this work, an exploration on application of compressive sensing technique on MIMO-OFDM based cognitive radio is discussed. In the proposed architecture, the signals from multiple channels are summed together to reduce the samples by N_r times, where N_r is the number of receive antennas. To recover the compressively sensed signal, the simplified reconstruction method is also presented. A multitude of MATLAB simulations are conducted to analyze the reception fidelity of the proposed receiver over the wireless communication. The simulation results show that the proposed architecture is able to efficiently detect MIMO-OFDM based CR signals. And the detection performance does depend on the occupancy of OFDM channel and the scale of the MIMO scheme. Moreover, it is also shown that the proposed receiver substantially outperforms the conventional ZF detector.

8.4 Summary

In this chapter, three works on related signal processing field are discussed. As near-capacity error correction codes already applied in many protocols, an hard-

ware efficient LDPC decoders based on asynchronous circuits techniques is explored and discussed. In following, compressive sensing which inherently contains similar mathematical problem as error correction codes are discussed and analyzed in two dimensions exploration as of both hardware reconstruction and application on practical communication problem.

9. SUMMARY

9.1 Contributions

In this dissertation, hardware efficiency improvements on different types of polar codes decoders are explored in algorithm and VLSI architecture level.

A hardware architecture of fastSSC algorithm for successive cancellation (SC) polar code decoders is presented. By exploiting the similarity between the decoding processing of fast constituent and regular polar codes, an unified hardware is presented to overcome the disadvantage of fast-SSC decoder that lacking decoding flexibility with respect to multiple code rates. Corresponding scheduling plan and the intendedly designed PU are also described. Compared with other state-of-art SC decoders, result shows that throughput centric successive cancellation decoder significantly increases the decoding throughput of polar codes by at least 2.5X depending on polar code rates.

The OPLSC hardware design approach for LSC decoding of polar codes is also introduced. By applying overlapping-path scheme, the l instances of $(l > 1)$ successive cancellation (SC) decoder for LSC with list size l can be cut down to only one. This results in a dramatic reduction of the hardware complexity without any decoding performance loss. Simulation results show that with proposed design approach the hardware efficiency is improved significantly over the other state-of-the-art LSC decoders.

To improve the hardware efficiency of BP polar codes decoder, the express journey belief propagation (XJBP) decoding algorithm is presented. The proposed algorithm facilitates belief propagation by utilizing the specific constituent codes that exist in the factor graph, which results in an express journey (XJ) for belief propagation in

each decoding iteration. In addition, this XJBP decoder employs a round-trip message passing scheduling method for the increased efficiency. The proposed method simplifies min-sum (MS) BP decoder by 40.6%. Along with the round-trip scheduling, the XJBP algorithm reduces the computational complexity of MS BP decoding by 90.4%; this enables an energy-efficient hardware implementation of BP decoding in practice.

As a co-design part for XJBP algorithm, the corresponding hardware architecture is set up to practice XJBP algorithm. The feasibility and efficiency are studied in scheduling problem first. By using different strategies of scheduling algorithm, it is shown that with constrained hardware resource, the proposed scheduling strategies save up to 40% decoding latency. With the proposed static scheduling strategy, the practical memory access problem is identified. Furthermore, the memory access problem is solved by the introduction of distributed memory architecture. The final results shows the proposed XJBP hardware implementation can achieve 3X power efficiency improvement without any performance loss.

9.2 Future Work

Although polar codes received numerous attentions and gained considerable success since it is invented, there are still many opportunities and challenges for further exploration and research to adopt polar codes in practice. The future works to achieve this objective are listed below.

First, polar codes achieve the channel capacity with very large code sizes, however large-size polar codes decoders is constrained by the limitation of hardware resources. To date, works on concatenated polar codes show that concatenated shorter polar codes are able to outperform a long polar code in terms of error correction performance. Future effort will be paid on energy efficient decoders which uses the special

property of concatenated polar codes.

Second, the simplifications presented in this dissertation for polar codes are based on constituent codes, however, which depend on the distributions of frozen bits in the coding structure. While the selections of positions of frozen bits are determined by the channel models, future work will be spent on a more flexible decoder for polar codes.

Third, although in terms of error correction capability, LSC decoders outperform SC and BP based decoders, LSC decoders consume too much hardware resource to be competitive at resource-limited scenarios. Different from SC decoder, which has theoretical bottle neck of limited search space, BP decoder has the potential to achieve the comparable performance as the LSC decoder. Future investigation will be given on the variances of BP decoding algorithm to improve error correction performance of BP decoders.

REFERENCES

[1] Marvin K Simon and Mohamed-Slim Alouini. *Digital communication over fading channels*, volume 95. John Wiley & Sons, 2005.

[2] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[3] Todd K Moon. Error correction coding. *Mathematical Methods and Algorithms. Jhon Wiley and Son*, 2005.

[4] Claude Berrou and Alain Glavieux. Near optimum error correcting coding and decoding: Turbo-codes. *Communications, IEEE Transactions on*, 44(10):1261–1271, 1996.

[5] Robert G Gallager. Low-density parity-check codes. *Information Theory, IRE Transactions on*, 8(1):21–28, 1962.

[6] C Nerrou, Alain Glavieux, and Punya Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-Codes (1). In *Proc. IEEE Int. Conf. Commun*, pages 1064–1070, 1993.

[7] David JC MacKay and Radford M Neal. Near Shannon limit performance of low density parity check codes. *Electronics letters*, 32(18):1645–1646, 1996.

[8] Thomas J Richardson, M Amin Shokrollahi, and Rüdiger L Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *Information Theory, IEEE Transactions on*, 47(2):619–637, 2001.

[9] Erdal Arikan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *Information Theory, IEEE Transactions on*, 55(7):3051–3073, 2009.

[10] A Eslami and H Pishro-Nik. On bit error rate performance of polar codes in finite regime. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 188–194. IEEE, 2010.

[11] Jing Guo, Minghai Qin, Albert Guillen i Fabregas, and Paul H Siegel. Enhanced belief propagation decoding of polar codes through concatenation. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 2987–2991. IEEE, 2014.

[12] Kai Niu and Kai Chen. CRC-aided decoding of polar codes. *Communications Letters, IEEE*, 16(10):1668–1671, 2012.

[13] Irina Tal and Alexander Vardy. How to construct polar codes. *Information Theory, IEEE Transactions on*, 59(10):6562–6582, 2013.

[14] Ido Tal and Alexander Vardy. List decoding of polar codes. *Information Theory, IEEE Transactions on*, 61(5):2213–2226, 2015.

[15] Amin Alamdar-Yazdi and Frank R Kschischang. A simplified successive-cancellation decoder for polar codes. *IEEE communications letters*, 15(12):1378–1380, 2011.

[16] K Niu and K Chen. Stack decoding of polar codes. *Electronics letters*, 48(12):695–697, 2012.

[17] Kai Chen, Kai Niu, and Jiaru Lin. Improved successive cancellation decoding of polar codes. *Communications, IEEE Transactions on*, 61(8):3100–3107, 2013.

[18] Ryuhei Mori and Toshiyuki Tanaka. Performance of polar codes with the construction using density evolution. *Communications Letters, IEEE*, 13(7):519–521, 2009.

[19] Satish Babu Korada, Eren Şaşoğlu, and Rüdiger Urbanke. Polar codes: Characterization of exponent, bounds, and constructions. *Information Theory, IEEE Transactions on*, 56(12):6253–6264, 2010.

[20] Nadine Hussami, Satish Babu Korada, and Rüdiger Urbanke. Performance of polar codes for channel and source coding. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pages 1488–1492. IEEE, 2009.

[21] Ying Wang and Krishna R Narayanan. Concatenations of polar codes with outer BCH codes and convolutional codes. In *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*, pages 813–819. IEEE, 2014.

[22] Eran Hof and Shlomo Shamai. Secrecy-achieving polar-coding. In *2010 IEEE Information Theory Workshop*, 2010.

[23] O Ozan Koyluoglu and Hesham El Gamal. Polar coding for secure transmission and key agreement. *Information Forensics and Security, IEEE Transactions on*, 7(5):1472–1483, 2012.

[24] Dong-Min Shin, Seung-Chan Lim, and Kyeongcheol Yang. Mapping Selection and Code Construction for 2ˆ m-ary Polar-Coded Modulation. *Communications Letters, IEEE*, 16(6):905–908, 2012.

[25] Naveen Goela, Satish Babu Korada, and Michael Gastpar. On LP decoding of polar codes. In *Information Theory Workshop (ITW), 2010 IEEE*, pages 1–5. IEEE, 2010.

[26] Emmanuel Abbe and Emre Telatar. MAC polar codes and matroids. In *Information Theory and Applications Workshop (ITA), 2010*, pages 1–8. IEEE, 2010.

[27] Erdal Arıkan. Systematic polar coding. *IEEE Commun. Lett*, 15(8):860–862, 2011.

[28] Bin Li, Hui Shen, and David Tse. An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check. *Communications Letters, IEEE*, 16(12):2044–2047, 2012.

[29] Bo Yuan and Keshab K Parhi. Low-latency successive-cancellation polar decoder architectures using 2-bit decoding. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 61(4):1241–1254, 2014.

[30] Gabi Sarkis, Pascal Giard, Alexander Vardy, Claude Thibeault, and Warren J Gross. Fast polar decoders: Algorithm and implementation. *Selected Areas in Communications, IEEE Journal on*, 32(5):946–957, 2014.

[31] Alptekin Pamuk. An FPGA implementation architecture for decoding of polar codes. In *Wireless Communication Systems (ISWCS), 2011 8th International Symposium on*, pages 437–441. IEEE, 2011.

[32] Bo Yuan and Keshab K Parhi. Architecture optimizations for BP polar decoders. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 2654–2658. IEEE, 2013.

[33] Bo Yuan and K.K. Parhi. Early Stopping Criteria for Energy-Efficient Low-Latency Belief-Propagation Polar Code Decoders. *Signal Processing, IEEE Transactions on*, 62(24):6496–6506, Dec 2014.

[34] Yingxian Zhang, Qingshuang Zhang, Xiaofei Pan, Zhan Ye, and Chao Gong. A simplified belief propagation decoder for polar codes. In *Wireless Symposium (IWS), 2014 IEEE International*, pages 1–4. IEEE, 2014.

[35] Youn Sung Park, Yaoyu Tao, Shuanghong Sun, and Zhengya Zhang. A 4.68 Gb/s belief propagation polar decoder with bit-splitting register file. In *VLSI Circuits Digest of Technical Papers, 2014 Symposium on*, pages 1–2. IEEE, 2014.

[36] Ramtin Pedarsani, S Hamed Hassani, Ido Tal, and Emre Telatar. On the construction of polar codes. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 11–15. IEEE, 2011.

[37] Erdal Arıkan et al. A performance comparison of polar codes and reed-muller codes. *IEEE Commun. Lett*, 12(6):447–449, 2008.

[38] Tom Richardson and Ruediger Urbanke. *Modern coding theory*. Cambridge University Press, 2008.

[39] Camille Leroux, Ido Tal, Alexander Vardy, and Warren J Gross. Hardware architectures for successive cancellation decoding of polar codes. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 1665–1668. IEEE, 2011.

[40] Camille Leroux, Alexandre J Raymond, Gabi Sarkis, and Warren J Gross. A semi-parallel successive-cancellation decoder for polar codes. *Signal Processing, IEEE Transactions on*, 61(2):289–299, 2013.

[41] Anadi Mishra, Alexandre J Raymond, Luca Gaetano Amaru, Gabi Sarkis, Camille Leroux, Pascal Meinerzhagen, Andreas Burg, and Warren J Gross. A successive cancellation decoder ASIC for a 1024-bit polar code in 180nm CMOS. In *Solid State Circuits Conference (A-SSCC), 2012 IEEE Asian*, pages 205–208. IEEE, 2012.

[42] Chuan Zhang, Bo Yuan, and Keshab K Parhi. Reduced-latency sc polar decoder architectures. In *Communications (ICC), 2012 IEEE International Conference*

on, pages 3471–3475. IEEE, 2012.

[43] Tiben Che, Jingwei Xu, and Gwan Choi. TC: Throughput Centric Successive Cancellation Decoder Hardware Implementation for Polar Codes. *arXiv preprint arXiv:1504.06247*, 2015.

[44] Alexios Balatsoukas-Stimming, Alexandre J Raymond, Warren J Gross, and Andreas Burg. Hardware architecture for list successive cancellation decoding of polar codes. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 61(8):609–613, 2014.

[45] Jun Lin and Zhiyuan Yan. An efficient list decoder architecture for polar codes. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 23(11):2508–2518, 2015.

[46] Chuan Zhang, Xiaohu You, and Jin Sha. Hardware architecture for list successive cancellation polar decoder. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pages 209–212. IEEE, 2014.

[47] Bo Yuan and Keshab K Parhi. Low-latency successive-cancellation list decoders for polar codes with multibit decision. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 23(10):2268–2280, 2015.

[48] Tiben Che, Jingwei Xu, and Gwan Choi. Overlapped List Successive Cancellation Approach for Hardware Efficient Polar Code Decoder. *arXiv preprint arXiv:1511.00577*, 2015.

[49] Chenghui Zhang and Keshab Parhi. Low-latency sequential and overlapped architectures for successive cancellation polar decoder. *Signal Processing, IEEE Transactions on*, 61(10):2429–2441, 2013.

[50] Kiran K Gunnam, Gwan S Choi, Mark B Yeary, and Mohammed Atiquzzaman. VLSI architectures for layered decoding for irregular LDPC codes of WiMax. In *Communications, 2007. ICC'07. IEEE International Conference on*, pages 4542–4547. IEEE, 2007.

[51] Chenrong Xiong, Jun Lin, and Zhiyuan Yan. Symbol-based successive cancellation list decoder for polar codes. In *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*, pages 1–6. IEEE, 2014.

[52] Gabi Sarkis, Pascal Giard, Alexander Vardy, Claude Thibeault, and Warren J Gross. Unrolled Polar Decoders, Part II: Fast List Decoders. *arXiv preprint arXiv:1505.01466*, 2015.

[53] Chuan Zhang, Zhongfeng Wang, Xiaohu You, and Bo Yuan. Efficient adaptive list successive cancellation decoder for polar codes. In *Signals, Systems and Computers, 2014 48th Asilomar Conference on*, pages 126–130. IEEE, 2014.

[54] J. Xu, T. Che, and G. Choi. XJ-BP: Express Journey Belief Propagation Decoding for Polar Codes. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2015.

[55] Michael R Garey, David S. Johnson, and Larry Stockmeyer. Some simplified NP-complete graph problems. *Theoretical computer science*, 1(3):237–267, 1976.

[56] Jürgen Teich. *Digitale Hardware/Software-Systeme: Synthese und Optimierung*. Springer-Verlag, 2013.

[57] IEEE 802.16 Working Group et al. IEEE standard for local and metropolitan area networks. part 16: Air interface for fixed broadband wireless access systems. *IEEE Std*, 802:16–2004, 2004.

[58] Brian P Crow, Indra Widjaja, Jeong Geun Kim, and Prescott T Sakai. IEEE 802.11 wireless local area networks. *Communications Magazine, IEEE*, 35(9):116–126, 1997.

[59] Frank R Kschischang, Brendan J Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.

[60] Andrew J Blanksby and Chris J Howland. A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder. *Solid-State Circuits, IEEE Journal of*, 37(3):404–412, 2002.

[61] Weihuang Wang, Euncheol Kim, Kiran K Gunnam, and Gwan S Choi. Low-Power VLSI Design of LDPC Decoder Using Dynamic Voltage and Frequency Scaling for Additive White Gaussian Noise Channels. *Journal of Low Power Electronics*, 5(3):303–312, 2009.

[62] Tinoosh Mohsenin, Houshmand Shirani-mehr, and Bevan M Baas. LDPC decoder with an adaptive wordwidth datapath for energy and BER co-optimization. *VLSI Design*, 2013:7, 2013.

[63] Tong Lin, Kwen-Siong Chong, Joseph S Chang, and Bah-Hwee Gwee. An ultra-low power asynchronous-logic in-situ self-adaptive system for wireless sensor networks. *Solid-State Circuits, IEEE Journal of*, 48(2):573–586, 2013.

[64] Jens Spars and Steve Furber. *Principles Asynchronous Circuit Design*. Springer, 2002.

[65] Ehsan Rohani, Jingwei Xu, Tiben Che, Mosaddequr Rahman, Gwan Choi, and Mi Lu. Asynchronous baseband processor design for cooperative mimo satellite

communication. In *Circuits and Systems (MWSCAS), 2014 IEEE 57th International Midwest Symposium on*, pages 833–836. IEEE, 2014.

[66] Naoya Onizawa, Vincent C Gaudet, and Takahiro Hanyu. Low-energy asynchronous interleaver for clockless fully parallel LDPC decoding. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 58(8):1933–1943, 2011.

[67] Jingwei Xu, Tiben Che, Ehsan Rohani, and Gwan Choi. Asynchronous design for precision-scaleable energy-efficient LDPC decoder. In *Signals, Systems and Computers, 2014 48th Asilomar Conference on*, pages 136–140. IEEE, 2014.

[68] Fan Zhang. *LDPC Codes over Large Alphabets and Their Applications to Compressed Sensing and Flash Memory*. PhD thesis, Texas A&M University, 2010.

[69] David L Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[70] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, 2006.

[71] Zhi Tian and Georgios B Giannakis. Compressed sensing for wideband cognitive radios. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–1357. IEEE, 2007.

[72] Youngchun Kim, Wenjuan Guo, B Vikrham Gowreesunker, Nan Sun, and Ahmed H Tewfik. Multi-channel sparse data conversion with a single analog-to-digital converter. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 2(3):470–481, 2012.

[73] Xi Chen, Ehab Ahmed Sobhy, Zhuizhuan Yu, Sebastian Hoyos, Jose Silva-Martinez, Samuel Palermo, and Brian M Sadler. A sub-nyquist rate compressive

sensing data acquisition front-end. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 2(3):542–551, 2012.

[74] Jingwei Xu, Ehsan Rohani, Mosaddequr Rahman, and Gwan Choi. Signal reconstruction processor design for compressive sensing. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pages 2539–2542. IEEE, 2014.

[75] Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.

[76] Thomas Blumensath and Mike E Davies. Normalized iterative hard thresholding: Guaranteed stability and performance. *Selected Topics in Signal Processing, IEEE Journal of*, 4(2):298–309, 2010.

[77] Zhuizhuan Yu, Sebastian Hoyos, and Brian M Sadler. Mixed-signal parallel compressed sensing and reception for cognitive radio. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 3861–3864. IEEE, 2008.

[78] Marco F Duarte and Yonina C Eldar. Structured compressed sensing: From theory to applications. *Signal Processing, IEEE Transactions on*, 59(9):4053–4085, 2011.

[79] Jia Meng, Wotao Yin, Husheng Li, Ekram Hossain, and Zhu Han. Collaborative spectrum sensing from sparse observations in cognitive radio networks. *Selected Areas in Communications, IEEE Journal on*, 29(2):327–337, 2011.

[80] Deanna Needell and Joel A Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[81] David L Donoho, Arian Maleki, and Andrea Montanari. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences*, 106(45):18914–18919, 2009.

[82] Pankaj Bhagawat, Rajballav Dash, and Gwan Choi. Array like runtime reconfigurable MIMO detectors for 802.11 n WLAN: a design case study. In *Proceedings of the 2009 Asia and South Pacific Design Automation Conference*, pages 751–756. IEEE Press, 2009.

[83] Jingwei Xu and Gwan Choi. Compressive sensing and reception for MIMO-OFDM based cognitive radio. In *Computing, Networking and Communications (ICNC), 2015 International Conference on*, pages 884–888. IEEE, 2015.