SYSTEM DEVELOPMENT AND VLSI IMPLEMENTATION OF HIGH THROUGHPUT AND HARDWARE EFFICIENT POLAR CODE DECODER

A Dissertation

by

TIBEN CHE

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,	Gwan S. Choi
Committee Members,	Jiang Hu
	Jim Ji
	Duncan M. (Hank) Walker
Head of Department,	Miroslav M. Begovic

May 2017

Major Subject: Computer Engineering

ABSTRACT

Polar code is the first channel code which is provable to achieve the Shannon capacity. Additionally, it has a very good performance in terms of low error floor. All these merits make it a potential candidate for the future standard of wireless communication or storage system. Polar code is received increasing research interest these years. However, the hardware implementation of hardware decoder still has not meet the expectation of practical applications, no matter from neither throughput aspect nor hardware efficient aspect. This dissertation presents several system development approaches and hardware structures for three widely known decoding algorithms. These algorithms are successive cancellation (SC), list successive cancellation (LSC) and belief propagation (BP). All the efforts are in order to maximize the throughput meanwhile minimize the hardware cost.

Throughput centric successive cancellation (TCSC) decoder is proposed for SC decoding. By introducing the concept of constituent code, the decoding latency is significantly reduced with a negligible decoding performance loss. However, the specifically designed computation unites dramatically increase the hardware cost, and how to handle the conventional polar code sets and constituent codes sets makes the hardware implementation more complicated. By exploiting the natural property of conventional SC decoder, datapaths for decoding constituent codes are compatibly built via computation units sharing technique. This approach does not incur additional hardware cost expect some multiplexer logic, but can significantly increase the decoding throughput. Other techniques such as pre-computing and gate-level optimization are used as well in order to further increase the decoding throughput. A specific designed partial sum generator (PSG) is also investigated in this dissertation. This PSG is hardware efficient and timing compatible with proposed TCSC decoder. Additionally, an polar code construction scheme with constituent codes

optimization is also presents. This construction scheme aims to reduce the constituent codes based SC decoding latency. Results show that, compared with the state-of-art decoder, TCSC can achieve at least 60% latency reduction for the codes with length n = 1024. By using Nangate FreePDK 45nm process, TCSC decoder can reach throughput up to 5.81 Gbps and 2.01 Gbps for (1024, 870) and (1024, 512) polar code, respectively. Besides, with the proposed construction scheme, the TCSC decoder generally is able to further achieve at least around 20% latency deduction with an negligible gain loss. Overlapped List Successive Cancellation (OLSC) is proposed for LSC decoding as a design approach. LSC decoding has a better performance than LS decoding at the cost of hardware consumption. With such approach, the l (l > 1) instances of 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. Meanwhile, approaches to reduce the latency associated with the pipeline scheme are also investigated. Simulation results show that with proposed design approach the hardware efficiency is increased significantly over the recently proposed LSC decoders. Express Journey Belief Propagation (XJBP) is proposed for BP decoding. This idea origins from extending the constituent codes concept from SC to BP decoding. Express journey refers to the datapath of specific constituent codes in the factor graph, which accelerates the belief information propagation speed. The XJBP decoder is able to achieve 40.6% computational complexity reduction with the conventional BP decoding. This enables an energy efficient hardware implementation.

In summary, all the efforts to optimize the polar code decoder are presented in this dissertation, supported by the careful analysis, precise description, extensively numerical simulations, thoughtful discussion and RTL implementation on VLSI design platforms.

DEDICATION

To my wife, my father, my mother and my upcoming baby.

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Gwan Choi, for his financial support, thoughtful guidance and kind encouragement all the way along my Ph.D study. He has always gave me the supports whenever I needed help. I would like to thank Professor Jiang Hu, Professor Jim Ji and Professor Hank Walker for their time in serving in my committee. I appreciate Professor Jiang Hu's introduction and suggestions on polar codes, which opened a new door for my research. Additionally, I took two courses from him. Both of them significantly strengthened my understanding of digital circuit design and inspired me about my research. Professor Jim Ji has been very helpful and he gave me a lot of suggestions after my prelim examination. I would also like to thank Professor Hank Walker. Through his excellent and impressive teaching skills, I was enlightened on the ideas of VLSI testing, which gave me a chance to know more about the IC industry.

I would like to thank all the student's in Professor Choi's group. All the senior students, Ehsan Rohani, Mehnaz Rahman and Jingwei Xu, gave me lots of helps and suggestions at the beginning of my research. In particular, I would like to thank Jingwei Xu. We worked on the same area and had cooperation on several papers. He is very thoughtful and smart, and can always inspired me. Additionally, we had a lot of discussions not only about the research but also about the career path. I also want to thank to my officemates, Honghuang Lin, Jimmy Jin, Qian Wang and Zhiyuan Zheng.

I want to express my gratitude to my lovely wife, Tiantian. She has done so much for me and always believed in me. She quit her job in China and has been to U.S to keep me company. Because of her, I never feel lonely even at my most helpless moment. Last but not least, I would like to thank my parents for their support and encouragement in my whole life.

NOMENCLATURE

ECC	Error Correct Code
LDPC	Low Density Parity Check
RS	Reed-Solomon
BCH	Bose-Chaudhuri-Hochquenghem
BP	Belief Propagation
SC	Successive Cancellation
LSC	List Successive Cancellation
PSG	Partial Sum Generator
SR-PSG	Shift-Register Partial Sum Generator
DM-PSG	Directly Mapped PSG
SR-CB-PSG	Shift-Register Constituent-Code-Based PSG
FPGA	Field Programmable Gate Array

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iv

ACKNOWLEDGMENTS v

NOMENCLATURE vi

TABLE OF CONTENTS vii

LIST OF FIGURES x

LIST OF TABLES xiii

1. INTRODUCTION AND LITERATURE REVIEW 1

    1.1 Research Motivation 1
    1.2 Literature Review 3
    1.3 Contributions of Dissertation Work 6
        1.3.1 Throughput Centric Successive Cancellation 6
        1.3.2 Overlapped List Successive Cancellation Approach 7
        1.3.3 Express Journey Belief Propagation 8
        1.3.4 Asynchronous Circuit Design 8
    1.4 Outline of This Dissertation 9

2. RELEVANT BACKGROUND 10

    2.1 Channel Polarization 10
    2.2 Polar Code Construction and Encoding 13
    2.3 Polar Code Decoding 14
        2.3.1 Successive Cancellation 14
        2.3.2 Constituent Based SC Decoding 16
        2.3.3 List Successive Cancellation Decoding 18
        2.3.4 Belief Propagation Decoding 20

3. THROUGHPUT CENTRIC SUCCESSIVE CANCELLATION 23

    3.1 System Overview 25

3.2 Dataflow, Latency and Flexibility Analysis . . . 25
3.3 Processing Unit . . . 29
3.4 Shift-Register Constituent-Code Based Partial Sum Generator . . . 33
3.4.1 Mathematical Derivation . . . 36
3.4.2 Proposed Architecture . . . 40
3.5 Fix Point Analysis . . . 43
3.6 Implementation Results and Relevant Discussion . . . 43
3.7 Construction Scheme with Constituent Codes Optimization . . . 47
3.7.1 Constituent Codes Optimization . . . 49
3.7.2 Simulation Results . . . 51

4. OVERLAPPED LIST SUCCESSIVE CANCELLATION APPROACH . . . 57

4.1 Path-Overlapping Scheme and Relevant Analysis . . . 59
4.2 Latency Reduction . . . 62
4.2.1 Latency Reduction via Multi-Decision List SC Decoding . . . 62
4.2.2 Latency Reduction via Path-LLR-Compute-Ahead Scheme . . . 64
4.2.3 Latency Reduction via Adaptive LSC Decoding . . . 64
4.3 Performance Simulation and Analysis . . . 65

5. EXPRESS JOURNEY BELIEF PROPAGATION DECODING FOR POLAR CODES . . . 68

5.1 Simplified Belief Propagation Decoding . . . 68
5.1.1 All-Frozen $\mathcal{N}^0$ Codes . . . 69
5.1.2 All-Information $\mathcal{N}^1$ Codes . . . 70
5.1.3 Repetition $\mathcal{N}^{REP}$ Codes . . . 70
5.1.4 Single Parity Check $\mathcal{N}^{SPC}$ Codes . . . 71
5.2 Scheduling . . . 73
5.2.1 Round-Trip BP Updating . . . 73
5.2.2 Early Termination . . . 75
5.3 Simulation and Discussion . . . 75
5.3.1 Decoding Performance . . . 75
5.3.2 Computation Complexity Analysis . . . 77
5.3.3 Discussion . . . 80

6. ASYNCHRONOUS CIRCUIT APPLICATION . . . 81

6.1 Accelerated Dual-Path Asynchronous Circuit . . . 84
6.1.1 Dual-Path Circuit Design . . . 84
6.1.2 Power Cycle Schedule . . . 88
6.1.3 Case Study . . . 89
6.2 Asynchronous Design for Precision-Scaleable Energy-Efficient LDPC Decoder . . . 94

6.2.1 Proposed System 96
6.2.2 Design Details 97
6.2.3 Simulations and Analysis 101

7. SUMMARY 104

REFERENCES 106

LIST OF FIGURES

FIGURE Page

1.1 Diagram for communication system 2

2.1 Channel polarization example of 2 B-DMC channels 11

2.2 Recursive construction of n channel polarization 12

2.3 An example of (8,4) polar code encoder 14

2.4 An example of 8-bit SC decoding via tree presentation 15

2.5 (a) An example of $\mathcal{N}^0$ and $\mathcal{N}^1$ in 8-bit polar code tree, and (b) An example of $\mathcal{N}^{SPC}$ and $\mathcal{N}^{REP}$ in 8-bit polar code tree 18

2.6 List successive cancellation decoding paths on decoding tree 19

2.7 BP factor graph of n=8 polar codes 20

2.8 Basic kernel of BP algorithm 21

3.1 Examples of (a) tree architecture and (b) line architecture of 8-bit polar code decoder 24

3.2 Overview of proposed system when code length = 16 26

3.3 (a) Conventional tree presentation of 32 bits polar code, and (b) Simplified tree presentation of 32 bits polar code at rate 0.3125 and 0.6875. 29

3.4 Design details of memory controller 30

3.5 Design details of PU 31

3.6 Design details of PU_0 33

3.7 The conventional architecture of constituent based PSG 35

3.8 The architecture of SR-PSG 36

3.9 (a) Elements shift in generation matrix, and (b) Diagonal cycle-shift in generation matrix 37

3.10 Overall architecture of SR-CB-PSG 41

3.11 (a) PU tree of SC decoder, (b) PUs and their corresponding register, and (c) Architecture of multiplexing network 42

3.12 An example of (2^m-1) shifter for 16-bit polar code decoder 43

3.13 Effect of quantization on the BER/FER performance of (1024,512) code . 44

3.14 Latency reduction vs. code rate 48

3.15 Examples of constituent codes division optimization 51

3.16 The ber vs E_b/N_0 performance for proposed construction scheme 55

4.1 The conventional architecture of LSC decoder 57

4.2 The architecture of proposed design 59

4.3 Decoding schedule of the path-overlapping scheme for (8,4) polar code with (a) list size = 2 and (b) list size = 4 60

4.4 Latency overhead for different scheme 63

4.5 Decoding schedule of path-LLR-compute-ahead scheme 64

4.6 The improvement of hardware efficiency with proposed design approach . 66

5.1 (a) An example of $\mathcal{N}^0$ codes in shadow and $\mathcal{N}^1$ codes in gray, (b) An example of $\mathcal{N}^{REP}$ codes in shadow and $\mathcal{N}^{SPC}$ codes in gray, and (c) The simplified factor graph for the example of $\mathcal{N}^{REP}$ and $\mathcal{N}^{SPC}$ codes. 69

5.2 (a) Computations scheduled in the conventional BP decoders, and (b) Computations scheduled in a round-trip updating fashion. 74

5.3 Decoding performance of the proposed BP decoding algorithm for (1024, 512) polar code with rate 0.5 and max number of iteration of 60. 76

5.4 Average numbers of iterations of the proposed BP decoding algorithm for (1024, 512) polar code with rate 0.5. 78

5.5 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. 80

6.1 General model of asynchronous circuits 82

6.2 4 phase protocol 83

6.3 Overview of dual-path asynchronous circuit system 85

6.4 Timing of DMR 85

6.5 The details of one asynchronous block 86

6.6 Timing of power cycle scheduling 88

6.7 One transient fault delay example 90

6.8 Histogram of delays caused by fault injections 92

6.9 Delays under difference schemes with different FR 93

6.10 Generic LDPC decoding data flow graph 95

6.11 The overview system flow of proposed LDPC decoder 97

6.12 Asynchronous precision-salable VNU design. 98

6.13 Asynchronous precision-salable CNU design 99

6.14 Proposed asynchronous comparator 100

6.15 Units delays for different bits of precision 101

6.16 Voltage scaling to align processing latency 102

6.17 Normalized power reduction compared with fixed precision LDPC decoder 103

LIST OF TABLES

TABLE Page

3.1 Summary of decoding latency for each constituent polar code 28

3.2 Decoding schedule of pre-computation SC and TCSC for length 32 polar code 29

3.3 Truth table of PTU 32

3.4 Critical path comparison 44

3.5 Decoder latency comparison for length=1024 polar code 45

3.6 Resource comparison 45

3.7 Hardware resource of SR-CB-PSG for 1024 code length polar code decoder 46

3.8 Hardware comparison of different (n,k) SC decoder with q-bit quantization for inner LLRs using tree architecture 46

3.9 Synthesis result for (1024,870) and (1024,512) polar codes 48

3.10 Latency reduction 56

5.1 Number of all constituent codes with different sizes in a (1024, 512) polar code with rate 0.5 72

5.2 Number of computations of reduced-complexity BP algorithm with all polar codes at rate 0.5 78

5.3 Comparison of computations at different code rates 79

6.1 Hardware overhead (unit:μm^2) 94

1. INTRODUCTION AND LITERATURE REVIEW

This introduction first states the research motivation for this polar code related topic, especially for the wireless communication and storage applications. After that, the literature review is presented. Compared with other state of the art polar code coding hardware architectures, the main contributions of this dissertation are introduced. The outlines of this dissertation is given in the end of this chapter to guide readers follow this dissertation.

1.1 Research Motivation

Information plays an important role in the real world. People's activities are all connected by information switching. The technology develops rapidly every day, which makes the information involved in our daily life exponentially increase. Meanwhile, it has been facilitating people's demands for high volume of information as well. This phenomenon also called data explosion. For example, wireless communication standard has been developed from the third generation (3G) to the fourth generation (4G) in the decade, and will release the fifth generation (5G) in the next two years. These changes allow us to have wireless data swapped with the speed from kilobits per seconds to gigabits per seconds. It took several hours to download an MP3 song in the past but now we even are able to enjoy high resolution (4K) video online. Another example of data explosion is that the storage of computer system has changed form megabyte to tri-byte. How to transfer or store information efficiently and reliably are always be the first concern for the real application scenarios.

No matter the data transferring or storage, the process can be abstracted as shown in Fig. 1.1. Consider that we have source information to send, such as video. First, we need to compress it to save the bandwidth cost. This is usually done by removing redundancy information existed in the original information. The encoding approach which aims to

compress the original data called source encoding. After that, some carefully selected redundant information is attached to it. This is for increasing the reliability of the communication, which is referred as channel encoding. While the information carried by the channel, due to noisy introduced by the channel, the information may incur some unexpected error, such as bit flipping or erasing. After receiving those information, those errors could be fixed via the redundant information which are previously added. This process is also called channel decoding. After that, all the re-corrected information is decompressed at the destination side. Similarly, the decompressing process is called source decoding.

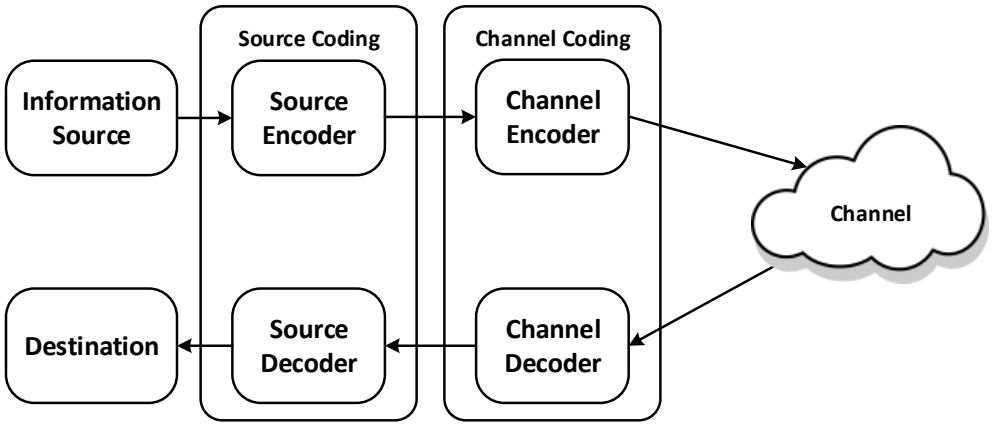


Figure 1.1: Diagram for communication system

Above description stresses two very important concepts in the communication system; source coding and channel coding. The first is trying to increase the channel utilization and the second one is trying to increase the channel robustness. Both of them are essential to a communication system. The codes used for channel coding are called error corrected coding (ECC). This dissertation focuses on the hardware architecture of one kind of channel coding called polar code.

In 1948, Claude E. Shannon [1] pointed out that, for a particular noise level, there is a theoretical maximum information transfer rate of the channel. This maximum rate is referred as Shannon capacity. In the next decades, researchers have been working on searching ECC scheme that is able to achieve Shannon capacity. Lots of channel coding scheme were proposed, such as convolutional code [2], BCH code [3], ReedSolomon (RS) code [4], turbo code [5] and low density parity check (LDPC) code [6]. Among those, turbo code and LDPC code are able to achieve the performance which is very close to Shannon capacity. They have been selected as the standard of LTE and ViMax, respectively. Recently, a new kind of EEC code called polar code invented since 2009 attracts lot of research interest. This dissertation work mainly focus on the hardware implementation of polar codes decoder.

1.2 Literature Review

Although both turbo and LDPC have a very good performance, they never can reach the Shannon capacity. Besides, they all suffer the error floor problems. In 2009, polar code was invented by E. Arikan [7]. It is the first channel code which provably is able to achieve the Shannon capacity. Additionally, its error floor performance is better than any other existed channel code [8]. Thus, polar codes is receiving increasing research attention these years. A lot of works have been done to theoretically evaluate and improve the performance of polar code [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]. Polar code is also regarded as a potential candidate for future standard of wireless communication and storage systems. Its encoding scheme is very straightforward and simply, which is also one of the attractions for real applications. However, the decoding scheme is more sophisticate and challenging.

Mainly, all the popular decoding scheme can be divided into three categories. The first one is called successive cancellation (SC) [7]. It decodes the codewords by recur-

sively applying the bit cancellation along the decoding factor graph. The second method is called list successive cancellation (LSC) [10]. It can be regarded as an extension of SC by exploiting more possibilities in codewords set. The third approach is called belief propagation (BP). This method estimates the correct codewords via a iterative message passing approach. All of those could directly map to a very straightforward hardware architecture. However, they suffers from either latency or computation complexity issues. Thus, lots of works are stressed on an efficient hardware implementation of polar code decoder.

C. Leroux [27] proposed both tree and line architecture of SC decoder. For length n polar code, it takes (2n-2) clock cycles to decode length n polar code. This is the first paper to systematically illustrate and evaluated the hardware architecture of SC decoder. Later on, he [28] proposed a semi-parallel architecture for both tree and line SC decoder, which gives a good inspiration about the trade-off between hardware complexity and latency. C. Zhang [29] proposed a low latency SC decoder with pre-computing and overlapped architecture. Pre-computing technology reduces the latency to (n-1), and the overlap scheme significantly increase the throughput for multiple frames situation. B. Yuan [30] proposed an architecture with applying the 2-bit decoding at the last stage and some gate level optimization. This further reduce the latency to (3n/4-1). B. Yuan [31] also proposed an architecture for LSC using multi-bit decision. A. Mishra [32] proposed the ASIC design for 1024-bit polar code with 180nm technology. Balatsoukas-Stimming [33, 34] provided the hardware architecture of LSC and LLR based LSC algorithm as well. J. Lin [35] proposed high throughput architecture for LSC with a reduced latency list decoding (RLLD) algorithm. T. Che [36] proposed a path overlapped design approach which enhances the hardware efficiency of LSC decoder. Alamdar-Yazdi [37] proposed the simplified SC (SSC) which can reduce the latency by finding some certain pattern sub-codewords. These kinds of sub-codewords are also called constituent codes. G. Sarkis [38] proposed the fast-SSC which can further reduce the latency by exploring more kinds of constituent codes. T.

Che [39] proposed the hardware architecture for constituent based SC decoder. This constituent concept can also be extended to LSC decoding. G. Sarkis [40] presented the constituent codes based LSC decoding. SC decoding is based on the feedback from decoded codewords which is also called partial sum. A partial sum generator (PSG) is needed for any kind of SC and LSC decoder. The partial sum needs to be calculated at the same clock cycle when the codewords are determined. Thus, it is on the critical path of the decoding and can affect the maximum frequency of the decoder. Some works have been done for a good PSG design. C. Leroux [28] proposed an indicator function based PSG (IF-PSG). C. Zhang [29] proposed a PSG with feedback part (FB-PSG). J. Lin [41] proposed a hybrid PSG for LSC. G. Berhault proposed a shift-register-based PSG (SR-PSG) [42, 43], which is able to increasing the timing performance and reduce the hardware complexity. Y. Fan [44] proposed a similar architecture with SR-PSG however with higher level simplification. T. Che [45] proposed a efficient constituent codes based PSG.

There are also many works have been done on BP decoder implementations. Compared with SC decoding, BP suffers from high computation complexity issues. Thus, it does not receive as much attentions as SC does. However, there are still some valuable researches have been done on that. A. Pamuk [46] presented a field programmable gate array (FPGA) implementation of BP decoder with an approximation of message passing function. Such approximation is done by the min-sum (MS) algorithm which degraded the decoding performance. B. Yuan [47] explored scaled min-sum (SMS) approximation for message passing functions remedy the performance penalty at the expense of extra scaling operations in each message passing. Later on, he [48] further improved the efficiency of SMS BP decoders using early termination in. C. Zhang [49] proposed a complexity reduced sum-product (SP) BP decoding by removing unnecessary computations for frozen bits in polar codes, which results in around 25% complexity reduction without any decoding performance degradation. J. Xu [50] proposed the express journey BP (XJ-BP).

XJ-BP introduce the concept of constituent code into BP decoding and also proposes a new scheduling scheme, which significant reduce the computation complexity.

1.3 Contributions of Dissertation Work

The mainly contribution of my dissertation works has four parts. They are the throughput centric successive cancellation (TCSC) decoder, overlapped design approach for LSC decoder, the express journey belief propagation (XJBP) decoder and research on the asynchronous circuit design for channel coding system.

1.3.1 Throughput Centric Successive Cancellation

This work presents hardware architecture of constituent code based successive cancellation algorithm for polar codes, which significantly reduces the decoding latency and dramatically increases the throughput. Constituent codes are those subset of the codeword with specific patterns. They are used to accelerate the successive cancellation decoding process of polar code without any performance degradation. Algorithmically, constituent code based SC algorithm suffers from the fact that its decoder scheduling and the consequent architecture depends on the code rate; this is a challenge for rate-compatible system. However, by exploiting the homogeneousness between the decoding processes of constituent polar codes and regular polar codes, the presented design is compatible with any rate. The scheduling plan and the intendedly designed processing core are also described. Additionally, a specifically designed partial sum generator (PSG) which is compatible with constituent code based SC decoder is proposed as well. We derive the mathematical presentation with the partial sums set which is corresponding to each constituent code. From this, we concoct a shift-register based PSG from . Results show that, compared with the state-of-art decoder, our design can achieve at least 60% latency reduction for the codes with length n = 1024. This design is validated via ASIC design with Nangate FreePDK 45nm process.

Besides, a polar code construction scheme that reduces constituent-code supplemented decoding latency is also presented. We modify the traditional construction approach to yield increased number of desired constituent codes that speeds the decoding process. For (n, k) polar code, instead of directly setting the k best and (n-k) worst bits to the information bits and frozen bits, respectively, we thoughtfully swap the locations of some information and frozen bits according to the quality of their equivalent channels. The algorithm of constituent codes division optimization is presented. We conducted the simulation of 1024 and 2048 length polar codes with multiple rates and analyzed the decoding latency for various length codes. The numerical results show that the proposed construction scheme generally is able to achieve at least around 20% latency deduction with an negligible gain loss with carefully selected optimization threshold.

1.3.2 Overlapped List Successive Cancellation Approach

This work presents an efficient hardware design approach for list successive cancellation (LSC) decoding of polar codes. By applying path-overlapping 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. The architecture of SC decoder is modified to support this new paradigm as well. Since modifications are made only on architecture and scheduling plan, no decoding performance gain loss or change is incurred. Three approaches, multi-decision list SC decoding, path-LLR-compute-ahead scheme and adaptive LSC decoding, to reduce the latency associated with the pipeline scheme are presented and evaluated as well. Simulation results show that with proposed design approach the hardware efficiency is increased significantly over the recently proposed LSC decoders.

1.3.3 Express Journey Belief Propagation

This work presents a novel constituent code based belief propagation decoding algorithm for polar codes. 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 information to propagate in each decoding iteration. In addition, this XJ-BP decoder employs a novel round-trip message passing scheduling method for the increased efficiency. The proposed method simplifies min-sum (MS) BP decoder by 40.6Along with the round-trip scheduling, the XJ-BP algorithm reduces the computational complexity of MS BP decoding by 90.4%; this enables an energy-efficient hardware implementation of BP decoding in practice.

1.3.4 Asynchronous Circuit Design

We are not only pursuing a decent design of polar code decoder via the architecture level, but also enhancing the entire system via the bottom circuit level. Asynchronous circuit is a technique can yield a good power and reliability performance, which is suitable for channel coding system. It has recently received increasing attention due to its low power consumption, high operation speed, less emission of elector-magnetic noise, better modularity, omission of clock distribution related problems, and robustness with respect to variations in supply voltage, temperature and fabrication process parameters [51]. I applied asynchronous circuit technology to two applications. Although both of them use LDPC decoder as a case study, they can be easily extended to polar code scenarios.

The first one is accelerated dual-path asynchronous circuit architecture. The dual-path architecture accelerates the asynchronous circuit system by circumventing transient faults caused delay. Specifically, dual-path asynchronous circuit design and associated arbiter are developed. Asynchronous circuit inherently tolerates transient errors however by incurring additional delay. This in turn can debilitate the circuit to suspension in an envi-

ronment where fault rate (FR) is excessively high. Dual-path design approach eliminates this problem by using whichever output or outcome combination that becomes valid first in each combination stage. The design approach is illustrated with an LDPC decoder architecture that must exhibit high degree of reliability in error-prone operating conditions. Results show that the decoder can achieve 11.3% and 39.5% speed up for average and maximum case, respectively, while the arbiter introduces only about 2% area overhead.

The second one is asynchronous precision-scaleable energy-efficient LDPC decoder.The decoder configures the computation precision to minimize circuit-level switching necessary for given target biterror rate (FER). The asynchronous circuit approach guarantees the completion of each compute-and-forward phase at necessary voltage levels. The voltage level is scheduled to ensure completion of minimum necessary decoding iterations. The proposed scheme is studied for the specific application of IEEE 802.16e to reduce the power consumption at a given target FER. The proposed design is evaluated on Nangate 45nm library. The results show that the proposed asynchronous design results in 51% reduction in terms of power consumption compared with full-precision decoding mode.

1.4 Outline of This Dissertation

The rest of this dissertation is organized as follows: Section 2 introduces the background of polar codes including its construction, principles and typical decoding algorithms. Section 3, 4, and 5 presents three polar code decoder designs, throughput centric successive cancellation, overlapped list successive cancellation approach and express journey belief propagation, respectively. Section 6 introduce my work on asynchronous design. Section 7 summarizes my dissertation works.

2. RELEVANT BACKGROUND

In this section, we first introduce the concept of channel polarization. Based on that, the construction and encoding scheme of polar code coding are presented. After that, three widely known kinds of decoding algorithms are described. The constituent based decoding is introduced as well.

2.1 Channel Polarization

Channel polarization is the key concept to understand that how polar code works. Polar code is first introduced for binary-input discrete memoryless channel (B-DMC). For a given B-DMC, there are two channel parameter, the symmetric capacity and the Bhattacharyya parameter, need to be defined.

Definition 2.1.1. The symmetric capacity of a B-DMC with input alphabet $\mathcal{X} = 0, 1$ is defined as:

$$I(W)\frac{1}{2}\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} W(y|x)\log\frac{W(y|x)}{\frac{1}{2}W(y|0)+\frac{1}{2}W(y|1)} \tag{2.1}$$

and

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

$$Z(W) \triangleq \sum_{y\in\mathcal{Y}}\sqrt{W(y|0)W(y|1)} \tag{2.2}$$

where $W(y|x)$ denotes the probability of receiving $y \in \mathcal{Y}$, given that $x \in \mathcal{X}$ sent from the transmitter. Those two parameters are used to described the rate and reliability of the B-DMC,respectively. Noticeably, the capacity of a symmetric B-DMC actually equals the mutual information between the input and output of the channel with uniform distribution on the inputs. It denotes that the reliable communication is possible over a

symmetric B-DMC at any rates up to $I(W)$. $Z(W)$ is an upper bound on the probability of maximum-likelihood (ML) decision error for $\{0,1\}$ tranmission over channel W.

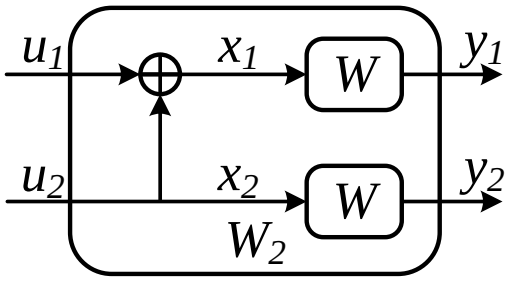


Figure 2.1: Channel polarization example of 2 B-DMC channels

Channel polarization is done by channel combining and channel splitting. Fig. 2.1 shows an example case for 2 B-DMC channel polarization. As the figure shows, two separate single bit channels $W : \mathcal{X} \to \mathcal{Y}$ are combined to create a new 2-bit channel $W_2 : \mathcal{X}^2 \to \mathcal{Y}^2$. The transition probability of new W_2 is

$$W_2(y_1, y_2|u_1, u_2) = W(y_1|u_1 \oplus u_2)W(y_2|u_2) \tag{2.3}$$

Then, we split the W_2 channel into two new single bit channel, $W^-(y_1, y_2|u_1)$ and $W^+(y_1, y_2, u_1|u_2)$. Their transition probabilities are calculated as

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

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

Thus, the symmetric capacity of the two new splitting channel are

$$I(W(y_1, y_2|u_1)) = I(W)^2 \tag{2.6}$$

$$I(W(y_1, y_2, u_1|u_2)) = 2I(W) - I(W)^2 \tag{2.7}$$

If we have single $I(w) = 0.5$, then $(W(y_1, y_2|u_1)) = 0.25$ and $I(W(y_1, y_2, u_1|u_2)) = 0.75$. Thus, we get two split channels with relatively worst and better channel qualities, respectively.

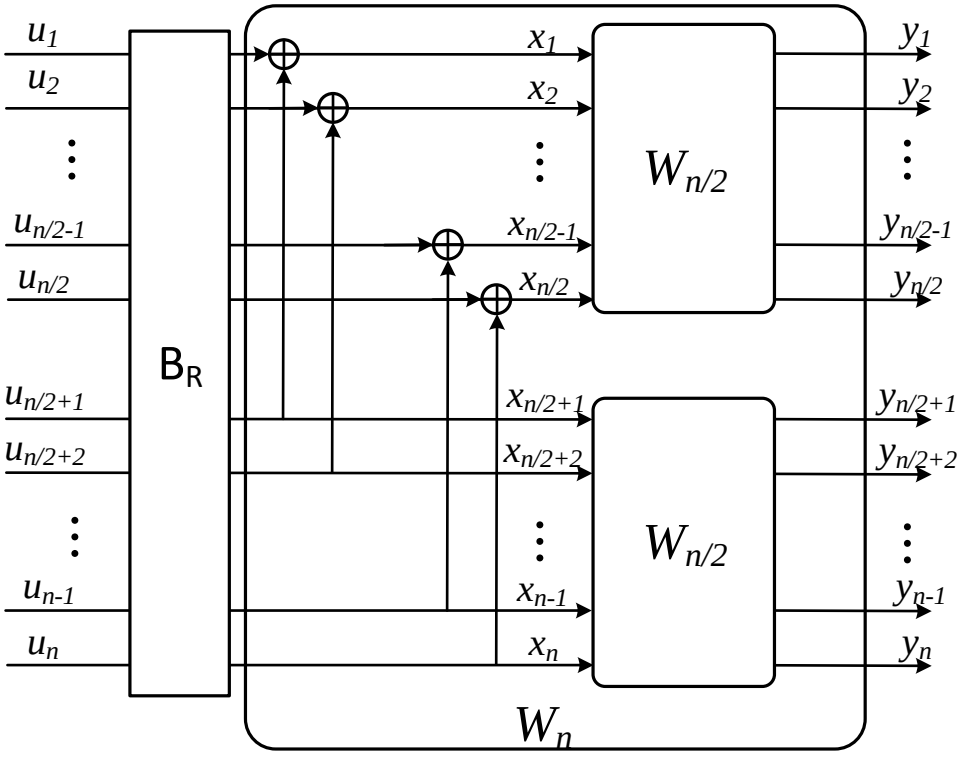


Figure 2.2: Recursive construction of n channel polarization

Inspired by this idea, if we keep recursively apply such polarize operation, all the symmetric capacities of original channel would be polarized to either 1 or 0. Fig. 2.2

shows an example of the recursively polarization of n channels. B_R stands for a bit-reverse network. The $\{I(W_N^{(i)})\}$ can be calculated by the recursive computing

$$I(W_N^{(2i-1)}) = I(W_{N/2}^{(i)})^2 \tag{2.8}$$

$$I(W_N^{(2i)}) = 2I(W_{N/2}^{(i)})^2 - I(W_{N/2}^{(i)})^2 \tag{2.9}$$

According to [7], for a B-DMC with error probability ε, if we apply Eq. (2.8) to n channels, as the n goes infinity, there are ε and $1-\varepsilon$ of n channels' capacities would to polarized to 0 and 1, respectively. This makes polar code the first channel code that is able to achieve the channel capacity.

2.2 Polar Code Construction and Encoding

As introduced in [7], a polar code is constructed by successively performing channel polarization. Mathematically, polar codes are linear block codes of length n = 2^m. The transmitted codeword $\boldsymbol{x} \triangleq (x_1, x_2, \cdots, x_m)$ is computed by $\boldsymbol{x} = \boldsymbol{uG}$ where $\boldsymbol{G} = \boldsymbol{F}^{\otimes \boldsymbol{m}}$, and $\boldsymbol{F}^{\otimes \boldsymbol{m}}$ is the *m*-th Kronecker power of $\boldsymbol{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. Each row of *G* is corresponding to an equivalent polarizing channel. For an (n,k) polar code, k bits that carry source information in $\boldsymbol{u}$ are called information bits. They are transmitted via the most k reliable channels. While the rest n-k bits, called frozen bits, are set to zeros and are placed at the least n-k reliable channels. Determining the location of the information and frozen bits depends on the channel model and the channel quality is investigated in [9]. Fig. 2.3 shows an example of (8,4) polar code encoder, where the black and white nodes stand for the information bits and frozen bits, respectively.

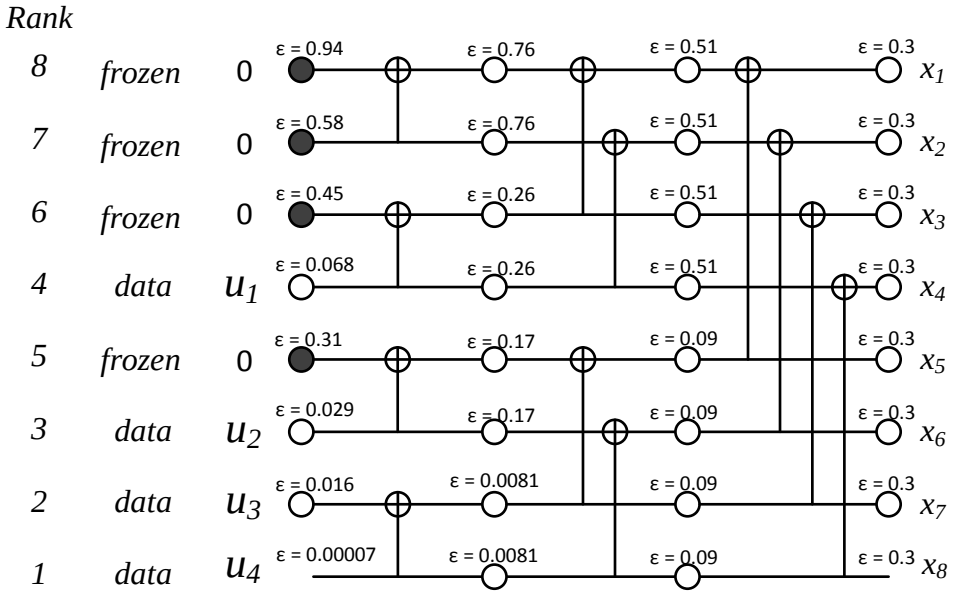


Figure 2.3: An example of (8,4) polar code encoder

2.3 Polar Code Decoding

2.3.1 Successive Cancellation

Polar codes can be decoded by recursively applying successive cancellation to estimate $\hat{u}_i$ using the channel output y_0^{N-1} and the previously estimated bits $\hat{u}_0^{i-1}$. Typically, the channel output are represented by log-likelihood ratios (LLR). LLR are defined as:

$$LLR_i^{\hat{\mathbf{u}}} = \ln \frac{P(\mathbf{y}|u_i = 0)}{P(\mathbf{y}|u_i = 1)} \tag{2.10}$$

where $P(\mathbf{y}|x)$ represents the probability that **y** is received as x is given in the transmitter. The estimated bits are obtained at the left side. The entire decoding process is running stage by stage. For length n polar code, there are $m = \log_2 n$ stages. Each stage consists of several computation unite with different functions. The functions are decided by their locations.

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. Fig. 2.4 shows an example of 8-bit SC decoding. $\boldsymbol{\alpha}$ stands for the soft

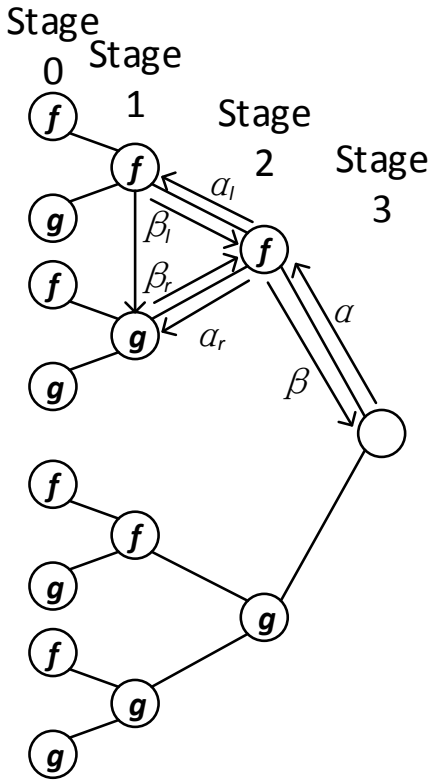


Figure 2.4: An example of 8-bit SC decoding via tree presentation

reliability value, typically is log-likelihood ratio (LLR), and $\boldsymbol{\beta}$ stands for the hard decision. $\boldsymbol{\alpha_l}$ and $\boldsymbol{\alpha_r}$ are the message passing from parent node to left and right child, and can be

computed according to Eq. (2.11) and Eq. (2.12), respectively.

$$\begin{aligned}\alpha_l[i] &= f(\alpha_v[i], \alpha_v[i + N^m/2]) \\ &= sign(\alpha_v[i])sign(\alpha_v[i + N^m/2]) \\ &\cdot\ min(|\alpha_v[i]|, |\alpha_v[i + N^m/2]|)\end{aligned} \tag{2.11}$$

$$\begin{aligned}\alpha_r[i] &= g(\beta_l[i - N^m/2], \alpha_v[i], \alpha_v[i - N^m/2]) \\ &= (-1)^{\beta_l[i-N^m/2]} \cdot \alpha_v[i - N^m/2] + \alpha_v[i]\end{aligned} \tag{2.12}$$

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

$$\beta_v = h(\alpha_v) = \begin{cases} 0,\ if\ \alpha_v \geqslant 0 \\ 1,\ otherwise \end{cases} \tag{2.13}$$

At intermediate stages, $\boldsymbol{\beta_v}$ can be recursively calculated by

$$\beta_v[i] = \begin{cases} \beta_l[i] \oplus \beta_r[i]\ if\ i \leq\ N^m/2 \\ \beta_r[i - N^m/2]\ otherwise \end{cases} \tag{2.14}$$

2.3.2 Constituent Based SC Decoding

SC decoding generally suffers from the high latency due to its inherent serial property. The processing of getting the partial sum from each note significantly constrain the decoding speed. Thus, in order to reduce the latency for calculating partial sum, constituent based SC decoding has been proposed [37], [38]. By finding some certain patterns in the source code, some part of the codeword and their corresponding partial sums can be estimated immediately without traversal. This method significantly reduces the partial-

sum-constrained latency. $\mathcal{N}^0$, $\mathcal{N}^1$, $\mathcal{N}^{SPC}$ and $\mathcal{N}^{REP}$ are the four commonest constituent code.

$\mathcal{N}^0$ and $\mathcal{N}^1$ are refer to those constituent codes which only contain frozen bits or information bits, respectively. For $\mathcal{N}^0$ codes, we can set $\boldsymbol{\beta_v}$ to 0 immediately. For $\mathcal{N}^1$ node, $\boldsymbol{\beta_v}$ can be directly determined via threshold detection Eq. (2.13). $\mathcal{N}^{SPC}$ 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 (n-1)/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 $\boldsymbol{\beta_v}$ via threshold detection. Then, calculated the parity by

$$parity = \sum_{i=1}^{N^m} \oplus HD_v[i]. \tag{2.15}$$

and, find the index of the least reliable bit via

$$j = arg \min_i |\alpha_v[i]|. \tag{2.16}$$

Eventually, $\boldsymbol{\beta_v}$ is decided by

$$\beta_v[i] = \begin{cases} HD_v[i] \oplus parity, \ when \ i = j \\ HD_v[i], \ otherwise \end{cases} \tag{2.17}$$

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

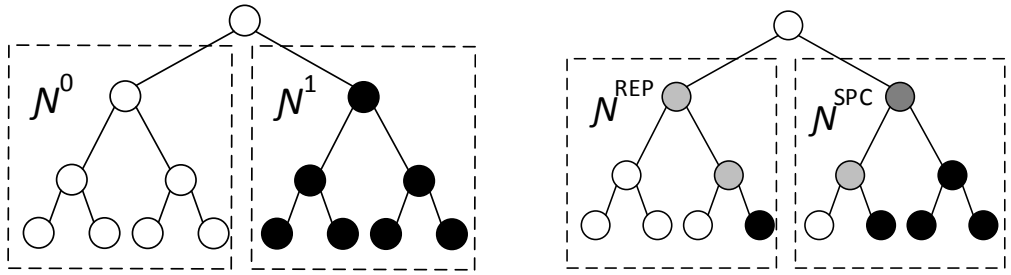


Figure 2.5: (a) An example of $\mathcal{N}^0$ and $\mathcal{N}^1$ in 8-bit polar code tree, and (b) An example of $\mathcal{N}^{SPC}$ and $\mathcal{N}^{REP}$ in 8-bit polar code tree

calculated as

$$\beta_v[i] = \begin{cases} 0, \ when \ \sum \alpha_v[i] \geqslant 0; \\ 1, \ otherwise \end{cases} \tag{2.18}$$

Fig. 2.5 shows an example of constituent codes in tree presentation.

2.3.3 List Successive Cancellation Decoding

The SC decoding algorithm of polar codes could be regarded as a greedy algorithm to estimate the transmitted binary sequences. For each bit, there are two candidates, namely, 0 and 1. The selection of the candidate is based on their probability $W(y, u_1^{i-1}|\hat{u}_i)$. This probability is calculated based on the previously estimated bit. This yields a very low robustness of entire decoding process since the most a-posterior probability (MAP) will be missed when any one of the previous decoded bits is not correct. This gives us a hint that if we can keep more than one path along with the decoding tree, there is a higher chance we are able to hit the global optimization. The final optimal path as the MAP estimation is selected by:

$$\hat{u}_i^n = argmax(W(y_i^n|\hat{u}_i^n)) \tag{2.19}$$

The approach by revering multiple paths during SC decoding called list successive

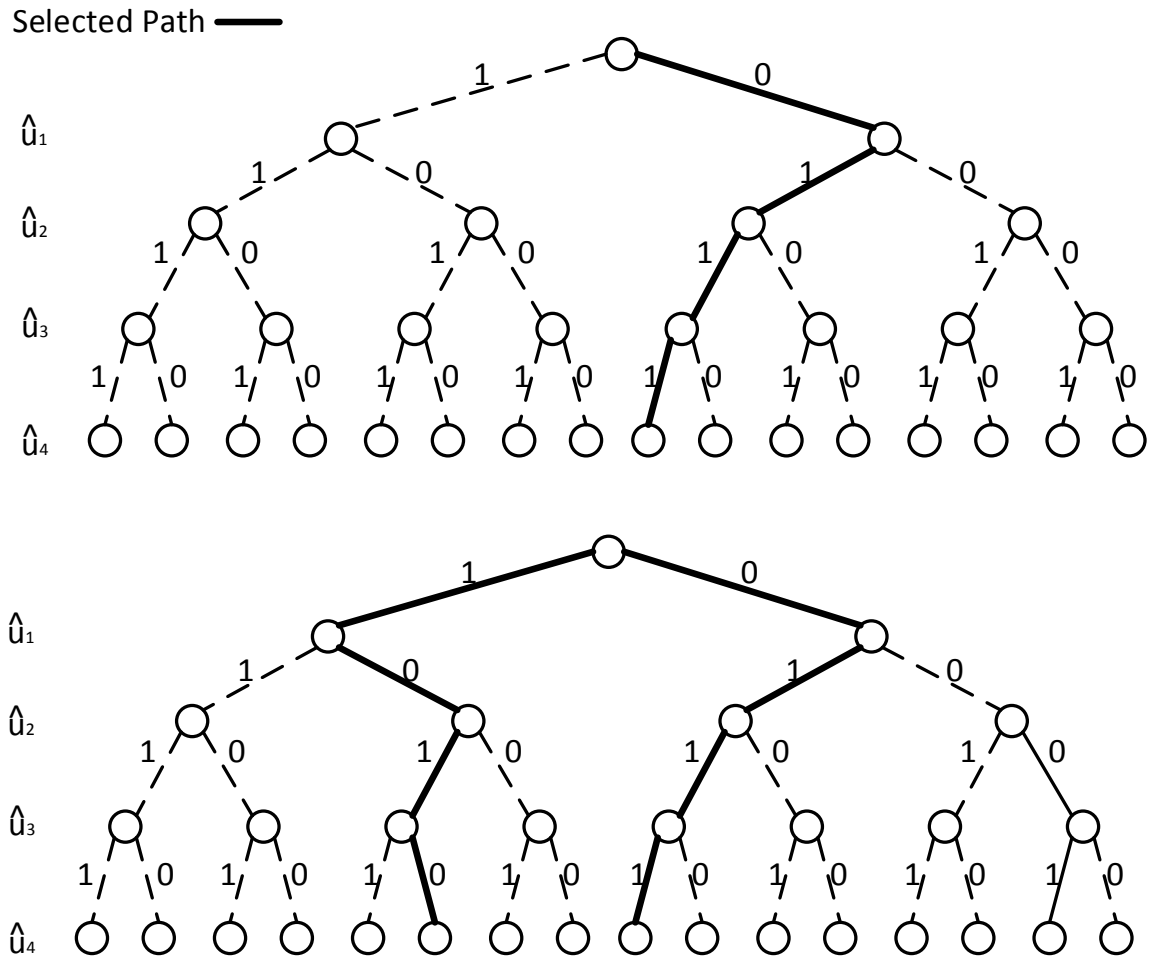


Figure 2.6: List successive cancellation decoding paths on decoding tree

cancellation. Actually, SC can be regarded as a special case of LSC. Fig. 2.6 shows the two relationship between LS and LSC. The upper figure is the decoding path of SC, we can see only one path survival at each layer. The bottom one describe the decoding path of LSC with list size two, which means two paths are reserved at each layer. The optimized decoded codeword is selected according to Eq. (2.19)

2.3.4 Belief Propagation Decoding

Both SC and LSC are decoding the codewords in a serial fashion. There is another approach which is able to perform decoding in parallel referred as belief propagation (BP) [52]. BP is a message passing algorithm which is capable of iteratively refining the codewrod information along the factor graph. The factor graph of polar code BP decoding is based on the encoding architecture. Fig. 2.7 shows an example of factor graph of a polar code with length n=8.

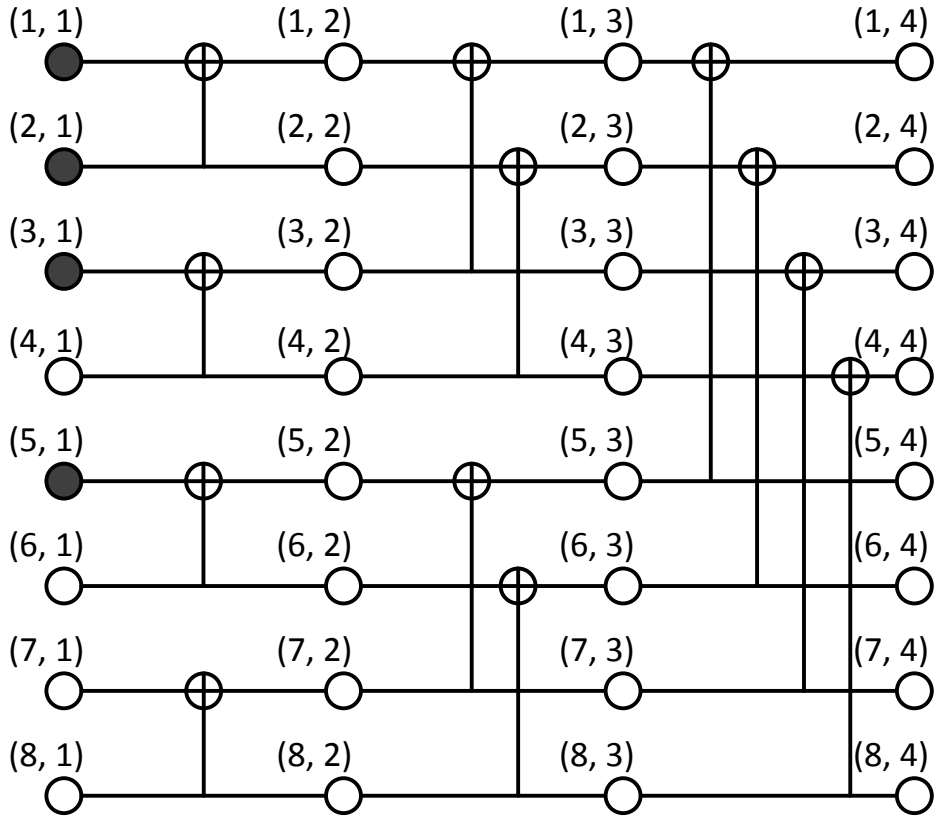


Figure 2.7: BP factor graph of n=8 polar codes

As Fig. 2.7 shows, it's very similar to the encoding architecture. For length n polar

code, there are m stages in the factor graph, $m = \log_2(n)$. The bits on the most right column contain the information of the channel output. The bits on the most left column contain the information for the estimated codeword. For the locations corresponded to the frozen and information bits, they are feeding ∞ (strong 0) and 0 (the most uncertain) to the factor graph, respectively. The messages are iteratively refining between channel output and estimated codeword until reaching the maximum iteration number or meeting early-terminate requirement.

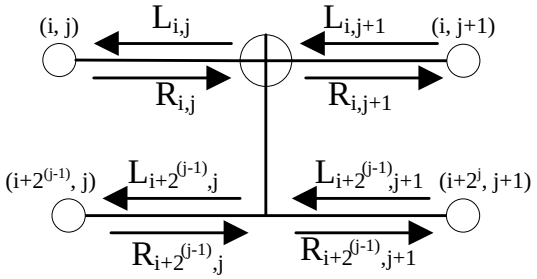


Figure 2.8: Basic kernel of BP algorithm

The messages are updated with the following rules. Fig. 2.8 shows the basic kernel of the BP decoding. For each node, there are one incoming message and one outcoming messages. 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 $R_{i,j}$. Both of them are presented in LLR form. The outcoming messages is updated based on the incoming

message and outcoming messages of other nodes via follows rules:

$$L_{i,j} = \mathcal{G}(L_{i,j+1}, L_{i+2^{j-1},j+1} + R_{i+2^{j-1},j}) \tag{2.20}$$

$$L_{i+2^{j-1},j} = \mathcal{G}(R_{i,j}, L_{i,j+1}) + L_{i+2^{j-1},j+1} \tag{2.21}$$

$$R_{i,j+1} = \mathcal{G}(R_{i,j}, L_{i+2^{j-1},j+1} + R_{i+2^{j-1},j}) \tag{2.22}$$

$$R_{i+2^{j-1},j+1} = \mathcal{G}(R_{i,j}, L_{i,j+1}) + R_{i+2^{j-1},j} \tag{2.23}$$

where $\mathcal{G}(x, y) = \ln((1 + xy)/(x + y))$ is the propagation function to update messages[53]. For a simplified hardware design, the function $\mathcal{G}$ typically is replaced by min-sum approximating $\mathcal{G}(x, y) \approx sign(x)sign(y)min(|x|, |y|)$.

3. THROUGHPUT CENTRIC SUCCESSIVE CANCELLATION [1]

This section we are going to talk about the hardware design of throughput centric successive cancellation (TCSC) decoder and an decoding-latency-reduced construction scheme with constituent codes optimization. Before that, let's start with some review of conventional SC decoder.

The general architecture of SC decoder was firstly evaluated in [27]. It categorizes the SC decoder into two types. Tree architecture and line architecture. Fig. 3.1 shows examples of both architectures. From this figure, we can tell both of them are very straightforward in term of decoding algorithm. The only difference the how to arrange the processing unit (PU). Tree architecture instantiates each stage with separate PUs. The interconnections among them are very simple. However, since only one stage is activated every clock cycle during decoding. This architecture seems not hardware efficient. Line architecture is designed to reduce the number of PUs. It only needs to instantiate the stage with maximum number of PUs. Then, the rest stages can be implemented by reusing thoes PUs. This approach is able to reduce half number of PUs compared with tree architecture. However, this is resulting from more complex routing logic. This cost becomes more significant when code size increases. Besides, tree architecture potentially has the merit that high throughput friendly if pipeline fashion involved for multiple frames scenario [29]. Thus, this dissertation work focus on the tree architecture. Some improvement is able to benefit line architecture as well.

Ideally, for length n polar code, 2n-2 clock cycle latency is need to do decoding. Many efforts have been done to reduce the latency. To the best of knowledge, (3/4n-1) [30] is the

[1] Part of this section is reprinted with permission from "Tc: throughput centric successive cancellation decoder hardware implementation for polar codes" by Tiben Che, Jingwei Xu and Gwan Choi, 2016. *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Page 991-995, ©2016 IEEE.

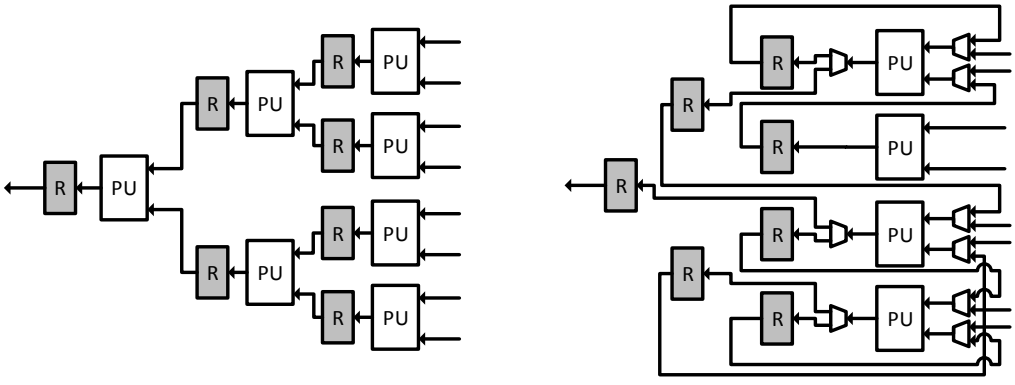


Figure 3.1: Examples of (a) tree architecture and (b) line architecture of 8-bit polar code decoder

faster speed which the conventional constituent code can achieve. However, it is still unacceptable especially for very large n. Constituent codes based [37, 38] SC can significantly reduce the decoding latency. This section presents a throughput centirc hardware architecture of it. Algorithmically, constituent code based successive cancellation algorithm suffers from the fact that its decoder scheduling and the consequent architecture depends on the code rate; this is a challenge for rate-compatible system. However, by exploiting the homogeneousness between the decoding processes of fast constituent polar codes and regular polar codes, the presented design is compatible with any rate. The scheduling plan and the intendedly designed processing core are also described. Additionally, a specifically designed partial sum generator (PSG) which is compatible with consituent code based SC decoder is proposed as well. We derive the mathematical presentation with the partial sums set which is corresponding to each constituent codes. From this, we concoct a shift-register based PSG from. Besides, a polar code construction scheme that reduces constituent-code supplemented decoding latency is also presented. We modify the traditional construction approach to yield increased number of desired constituent codes that

speeds the decoding process. For (n, k) polar code, instead of directly setting the k best and (n-k) worst bits to the information bits and frozen bits, respectively, we thoughtfully swap the locations of some information and frozen bits according to the quality of their equivalent channels. The algorithm of constituent codes division optimization is presented. All the relevant implementation and simulation results are presented in this section as well.

3.1 System Overview

Fig. 3.2 shows an overview of proposed system when code length = 16. Processing unit (PU) performs the f and g functions in Eq. (2.11) and Eq. (2.12), 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 Eq. (2.16), 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 $\mathcal{N}^{SPC}$ constituent polar codes. Design details are illustrated in section 3.3. The estimation of current bit in SC decoding is bases on the information of previous decoded bits ($\boldsymbol{\beta}$). This information is also called partial sum. Thus, a partial sum generator (PSG) which can co-operate with decoding pipeline is also needed. The details the PSG design are presented in Sec. 3.4.

3.2 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 2n-2 clock cycles. By using pre-computation as introduced in [29], which calculate all the possible value in

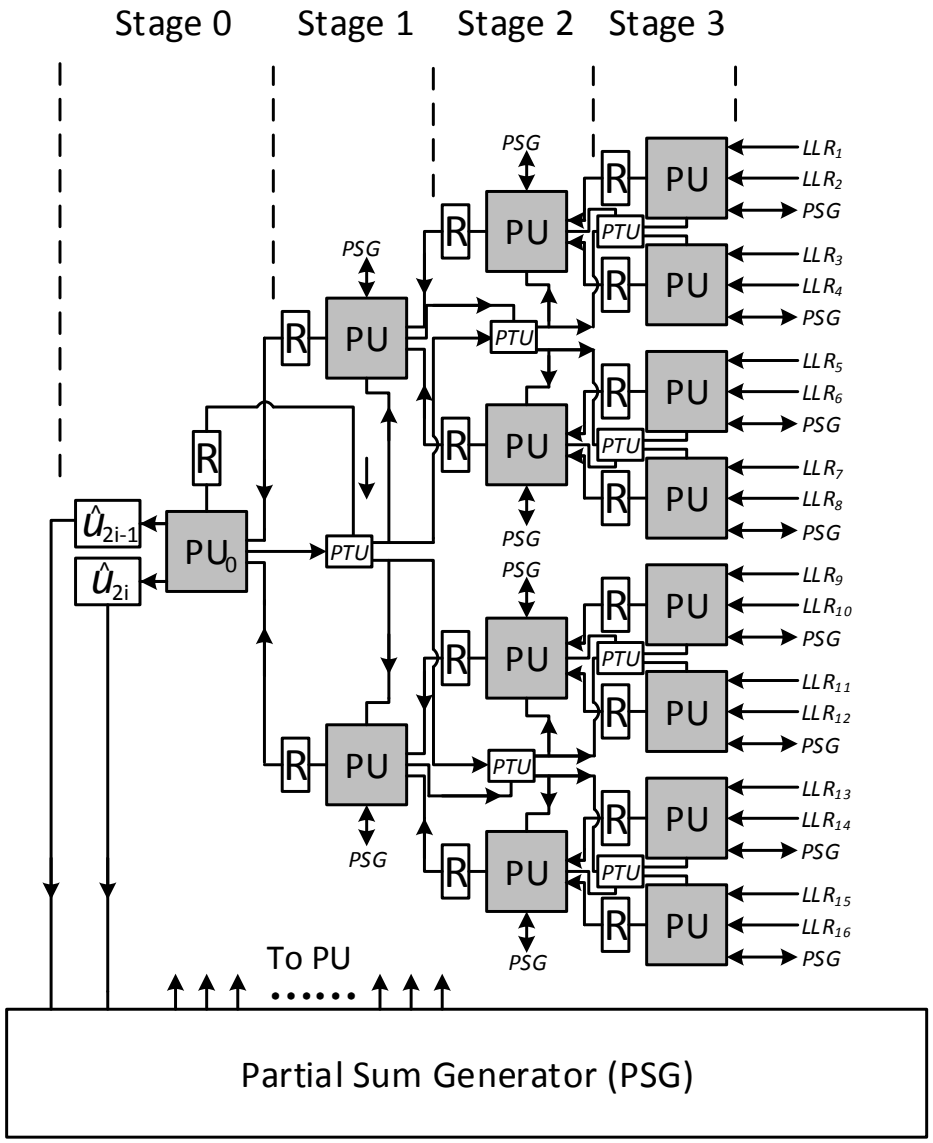


Figure 3.2: Overview of proposed system when code length = 16

Eq. (2.12) as the same clock cycle as Eq. (2.11) is calculated, the latency can be reduced to n-1. Furthermore, if this subtree is belong to fast constituent polar codes, the latency can be further reduced.

For $\mathcal{N}^0$, the feedback are all set to 0, and for $\mathcal{N}^1$, the feedback is determined by

hard decision of input LLRs. Initially, both the two computations need 1 clock cycle to generate partial sums after they are activated. Such scheme is presented in my conference paper [39]. However, by carefully investigating the principle of the hardware and algorithm, the latency for $\mathcal{N}^0$ can be further reduced. Since we already know 0s are the feedback from $\mathcal{N}^0$ constituent codes, we can skip the LLR update step and assign this value directly. The reason why we still hold one clock cycle in [36] is that one clock cycle to feed feedback into PSG is needed. If we feed the feedback one clock cycle in advance, we can avoid the timing bubble. But there is still one corner case for such scheme. That is when the $\mathcal{N}^0$ is right child, we can not do the feed operation in advance since all the partial should be feed into PSG in order. For this special case, we still need to hold one clock cycle. However, according to the principle of polar code encoding and the definition of $\mathcal{N}^0$, the right child $\mathcal{N}^0$ situation is extremely rare. Thus, for most of the time, there is no clock cycle consumption for $\mathcal{N}^0$.

For $\mathcal{N}^{SPC}$, according to Eq. (2.15), Eq. (2.16), and Eq. (2.17), we can tell that there are 3 operations needed. Finding the minimum LLR can be done by a comparator tree, which is naturally existed in SC decoder with tree architecture since every PU has a comparator for Eq. (2.11). For n LLRs, finding the smallest one use $\log_2 n$ clock cycles. Meanwhile, we can obtain the parity bit when the minimum LLR is found, which is explained in the next subsection. After that, one more clock cycle is need for signal parity check which is done by an xor gate. Thus, totally, decoding a length n $\mathcal{N}^{SPC}$ constituent polar codes need $\log_2 n + 1$ clock cycles.

For $\mathcal{N}^{REP}$, according to Eq. (2.18), 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 Eq. (2.12). For a length n $\mathcal{N}^{REP}$ constituent polar code, it needs $\log_2 n$ clock cycles to decode.

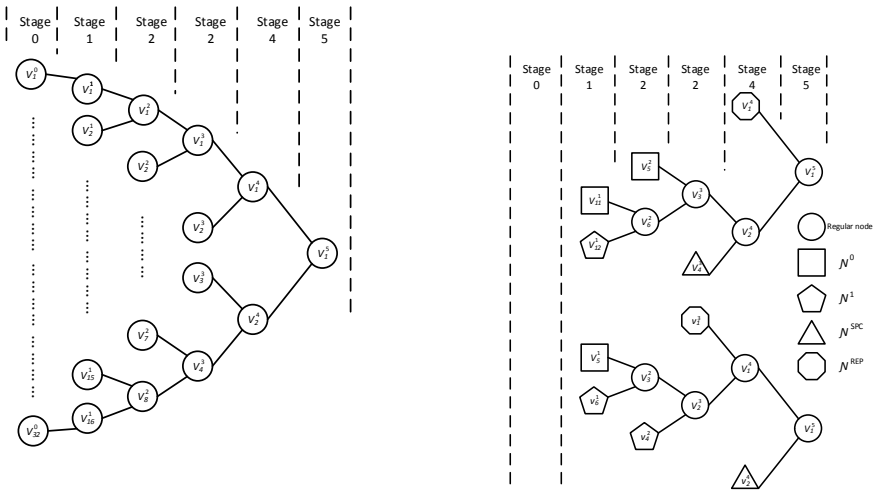
Table 3.1 gives the summary of decoding latency for each constituent polar code.

Table 3.1: Summary of decoding latency for each constituent polar code

Type	$\mathcal{N}^0$	$\mathcal{N}^1$	$\mathcal{N}^{SPC}$	$\mathcal{N}^{REP}$
Latency(clock cycle)	0	1	$\log_2 n + 1$	$\log_2 n$

$\mathcal{N}^0$ and $\mathcal{N}^1$ have time complexity $O(1)$ and $\mathcal{N}^{SPC}$ and $\mathcal{N}^{REP}$ have time complexity $O(\log_2 n)$. Compared with commonly discussed SC architecture in [28], [29] and [30], which all have linear time complexity $O(n)$, we can benefit significantly from proposed scheduling scheme in term of latency, especially with very large n. Fig. 3.3 shows the conventional tree presentation of 32-bit polar code and the simplified tree presentation of 32-bit polar code at rate 0.3125 and 0.6875. We can tell that both the number of nodes in these two simplified trees are significantly reduced. In the proposed design, we also adopt the pre-computation technique which allow us activate the left child node and right child node simultaneously. Table 3.2 lists the decoding scheduling schemes of 32-bit polar code using pre-computation SC decoder and proposed fast SC decoder at rate 0.3125 and 0.6875. Pre-computation SC decoder needs 31 clock cycles (CCs) to finish decoding, while fast SC decoder both only need 12 CCs at two different rates, which is much less than that of pre-computation SC decoder. Also note that the latency of two rates just happen to be the same in this example.

The main challenge for constituent code based decoder is that the architecture subject to the rate of codes. This is due to the reason that polar codes with different rates do not have the uniform distribution of constituent polar codes. Proposed design overcomes this obstacle by exploring the similarity between the decoding architecture of fast constituent and regular polar codes. The specific designed PU allows the tree architecture to deal with both fast constituent and regular polar codes, which means the entire decoding processing can run smoothly no matter what the distributions of constituent codes are. This architecture is independent and does not relay on the distribution of constituent codes. This



(a) (b)

Figure 3.3: (a) Conventional tree presentation of 32 bits polar code, and (b) Simplified tree presentation of 32 bits polar code at rate 0.3125 and 0.6875.

Table 3.2: Decoding schedule of pre-computation SC and TCSC for length 32 polar code

CC	1	2	3	4	5	6	7	8	9	10	11	12	$\cdots$	30	31
Pre-computation SC	v_1^4 & v_2^4	v_1^3 & v_2^3	v_1^2 & v_2^2	v_1^1 & v_2^1	v_1^0 & v_2^0	v_3^0 & v_4^0	v_3^1 & v_4^1	v_5^0 & v_6^0	v_7^0 & v_8^0	v_3^2 & v_4^2	v_5^1 & v_6^1	v_9^0 & v_{10}^0	$\cdots$	v_{29}^0 & v_{30}^0	v_{31}^0 & v_{32}^0
TCSC at rate 0.3125	v_1^4 & v_2^4	adder tree				v_3^3 & v_4^3	v_5^2 & v_6^2	v_{11}^1 & v_{12}^1	comparetor tree			parity check			
TCSC at rate 0.6285	v_1^4 & v_2^4	v_1^3 & v_2^3	adder tree			v_3^2 & v_4^2	v_5^1 & v_6^1	comparetor tree				parity check			

property provides the flexibility for multiple rates. To switch from one rate to another rate, only the control signals for given PUs need to be modified.

3.3 Processing Unit

All the outputs of each PU need to be saved to the registers. However, since the each PU is not activated every clock cycle. There should be a policy to enable and disable the

registers update. This can be done via a simple clock gating scheme. Thus such scheme also benefits the entire architecture in term of low power. Fig. 3.4 shows an example of it.

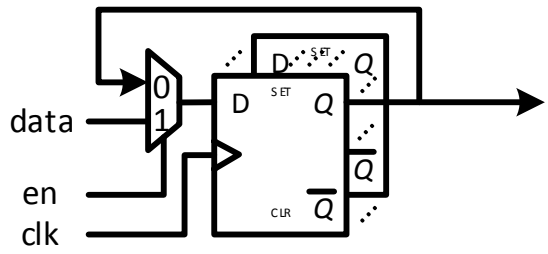


Figure 3.4: Design details of memory controller

Fig. 3.5 shows design details of PU. A single PU can perform f and g functions in Eq. (2.11) and Eq. (2.12), respectively. Also a PU tree can help to find the minimum values or do accumulation for multiple inputs. In Fig. 3.5, S stands for signed magnitude number and C stands for 2's complement number. Unlike the PU design in [30], 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 g function path. By moving number system convert modules to the f function path, which means using 2's complement as initial data form, the critical path is still along with g function path, but with significant reduction. 2). Compared with four number system convert modules are used in [30], only three are used if use 2's complement number. This is more hardware efficient. The benefits of this modification can be seen in section 3.6.

For each PU, two LLRs are fed simultaneously. Since we use the pre-computation technique, f and g functions are calculated at the same time. According to Eq. (2.12), there are only two types of possible results for g function, sum or difference. Its final

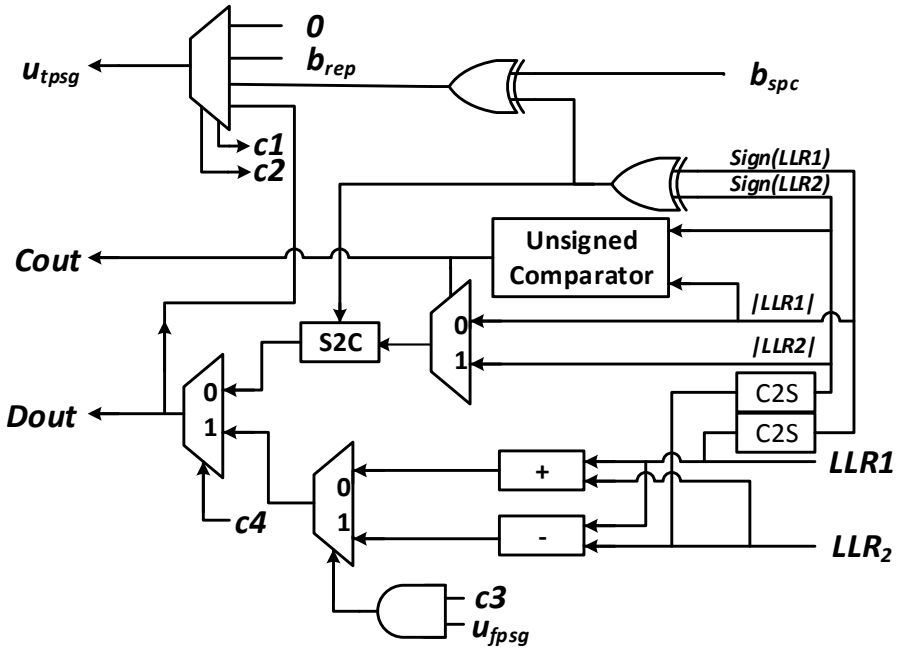


Figure 3.5: Design details of PU

result depends on the corresponding partial sum. The adder is also used for the calculation of $\mathcal{N}^{REP}$. When f function is performed, according to Eq. (2.11), both 2 inputs are divided into two parts: sign bit and unsigned number. Each part is processed separately first, and then 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 comparison indicator is output for the parity check bit routing, and the result of comparison Cout should be hold in register for PTU. u_{tpsg} is the partial sum calculated by the current stage for constituent codes scenario. All the outputs are determined by the control signals (c1,c2,c3,c4) plus corresponding partial sum b_{fpsg}.

Since every PU does exclusive-or operation to the sign bit of two inputs, according to Eq. (2.15), the sign bit of the final value in stage 0 should be equal to the parity. Eq. (2.17) can be performed using an xor gate. The PU that contains the minimum LLR

receives the parity check bit and the others receive 0s. 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 b_{parity} and the other is the comparison indicator of each stage $Cout^m$. The parity check bit is transmitted via output1 (O1) or output2 (O2) bases on the values of SS. Table. 3.3 shows the truth table of PTU. We can obtain the logic expression of O1 and O2 as: $O1 = b_{parity}\ and\ \overline{Cout^m}$, $O2 = b_{parity}\ and\ Cout^m$. This can be done by two and gates and one inverter.

Table 3.3: Truth table of PTU

b_{parity}	$Cout^m$	O1	O2
0	0	0	0
0	1	0	0
1	0	1	0
1	0	0	1

The PU in stage 0, as denote PU_0 in Fig. 3.2, has a simpler architecture. Fig. 3.6 shows the design details of PU_0. Since $\mathcal{N}^{SPC}$ cannot exist in stage 0, the top part in Fig. 3.5 which is relative to single parity check can be removed. Since pre-computing is used, two bits (u_{2i},u_{2i+1}) are obtained at the same time. This also means that the PU_0 needs to generate the corresponding partial sums for those two bit. Otherwise, it would not synchronous with the decoding timing of PSG which can degrade the performance. For g function and $\mathcal{N}^{REP}$, the output of f function can be feed back to it immediately, and the sign bit of the result of adding is the partial sum for $\mathcal{N}^{REP}$.

Compared with my paper in [39], there are three main changes. The first one is that we simplify the multiplexer logic between f function and accumulation for $\mathcal{N}^{REP}$. Instead of using two MUXs, only one MUX and one and gate are used. This results in better

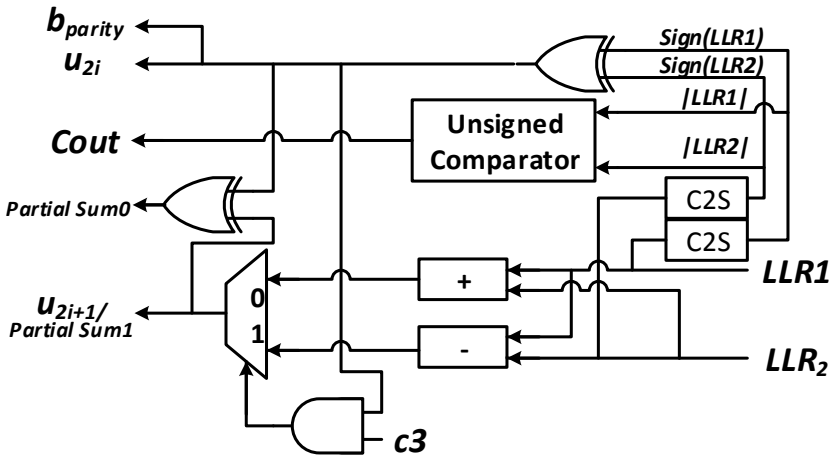


Figure 3.6: Design details of PU_0

timing and silicon area performance. The second one is that we remove the two memory parts to hold the results of addition and subtraction for pre-computing. Those two are unnecessary since the inputs of each stage do not change until it finish both left and right child computation. Logical wise, there is no need to hold those result since the outputs keeps the same until the inputs change. Timing wise, those part are not on the critical path thus no need to buffer them. The third change is that clarify the outputs about partial sum of each PU.

3.4 Shift-Register Constituent-Code Based Partial Sum Generator

Partial sum generator (PSG) is essential to both the constituent code based and the conventional SC decoder. Its importance can be evaluated from both function and timing performance. In terms of function, without a correct partial sum calculation, the entire codewords cannot be decoded correctly. In terms of timing performance, the PSG must generate the output in the same cycle when the bits are estimated. Otherwise the decoder need one more cycle to do the partial sum calculation. Such overhead becomes significant especially for long code length scenario. Additionally, such timing requirement also makes

the PSG on the decoding path of the decoder, which decides the maximum operation frequency of entire system. Thus, a good PSG design is very necessary for the system.

PSG design for constituent code based SC decoder is more complicated than that of conventional one. First, a routing system is needed for PSG input. Since the partial sum can be generated from PUs in any stages, and the number of partial sums from each is different, a multiplexer network is necessary to make sure the correct partials sums are feed in to the PSG. The details of multiplexer network are described in section 3.4.2. Second, the computation unit needs more multiplexing logic. The computation of partial sum is actually based on the encoding architecture. For PSG of consituend code based SC decoder, the computation unit needs to consider the inputs location for partial sums form different stage. A MUX is placed at each node of the computation unit. Fig. 3.7 shows an example of directly mapped constituent codes based PSG for 16-bit polar code (DM-PSG). From that, we can tell that the critical path after multiplexer network is $(\log_n -1)(\triangle Xor+ \triangle MUX)$ where $\triangle$ means the delay of that component. This is not what we expected since it increases with code length. Thus, a PSG with better timing performance needs to be investigated.

Among all the aforementioned PSG design, shift-register-based PSG (SR-PSG) [42, 43] gives us a good inspiration. For length n polar code decoder, it consists of n registers and some other simple combination logic. Along with the estimation of each $\hat{u}_i$, the registers perform shift calculation and the partial sums can be obtained from their corresponding register. Its architecture is illustrated in Fig. 3.8. This architecture is built according to the following rule:

$$\begin{cases} R_0 \Leftarrow \hat{u}_i \cdot c_{i,0} \\ R_k \Leftarrow R_{k-1} \oplus (\hat{u}_i \cdot c_{i,k}),\ if\ k \geqslant 0 \end{cases} \tag{3.1}$$

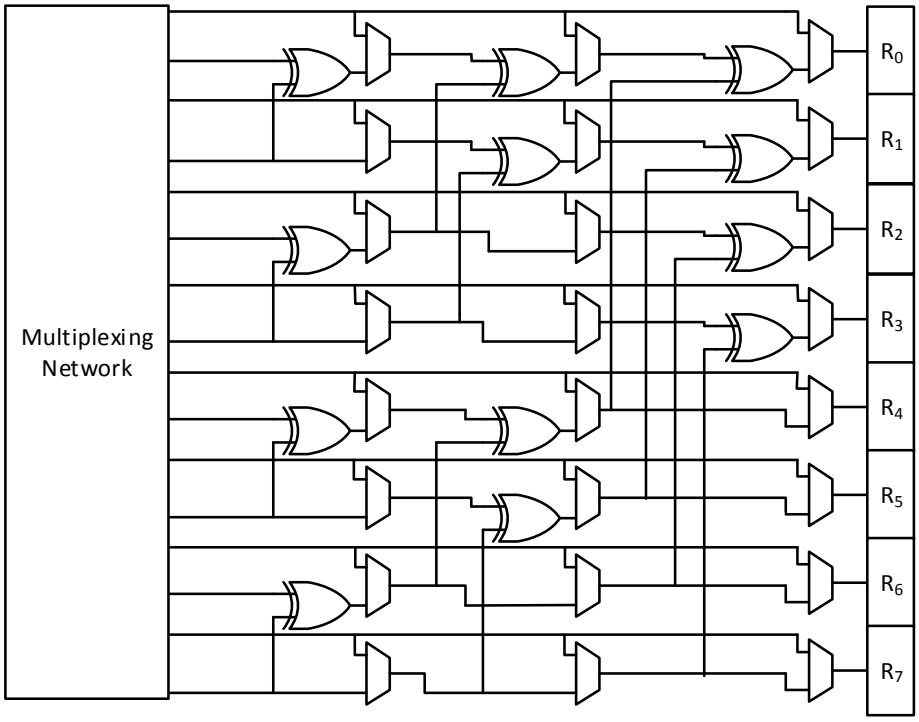


Figure 3.7: The conventional architecture of constituent based PSG

where $\cdot$ and $\oplus$ stand for and and exclusive-or operation, respectively. In Fig. 3.8, R_k means the *k*th register, $\hat{u}_i$ means the *i*th estimated bit. $\beta_{i,j}$ means the *j*th partial sum in stage *i*. $c_{i,k}$ means the *i*th row and *k*th column in the generate matrix *G*. The matrix generation unit is able to generate $c_{i,k}$ with very simple logic. The SC decoder consists of many basic computation parts called processing unit (PU). Each partial sum needs to be feed into corresponding PU. The shift register based architecture can guarantee that all partial sum required by a PU are all generated in the same register, which can avoid any extra routing logic in the circuit.

Such architecture is able to receive the estimated bit and update the corresponding partial sum by every valid cycle, which keeps highly consistent with SC decoding processing. However, this architecture is not suitable for constituent codes based SC decoder since

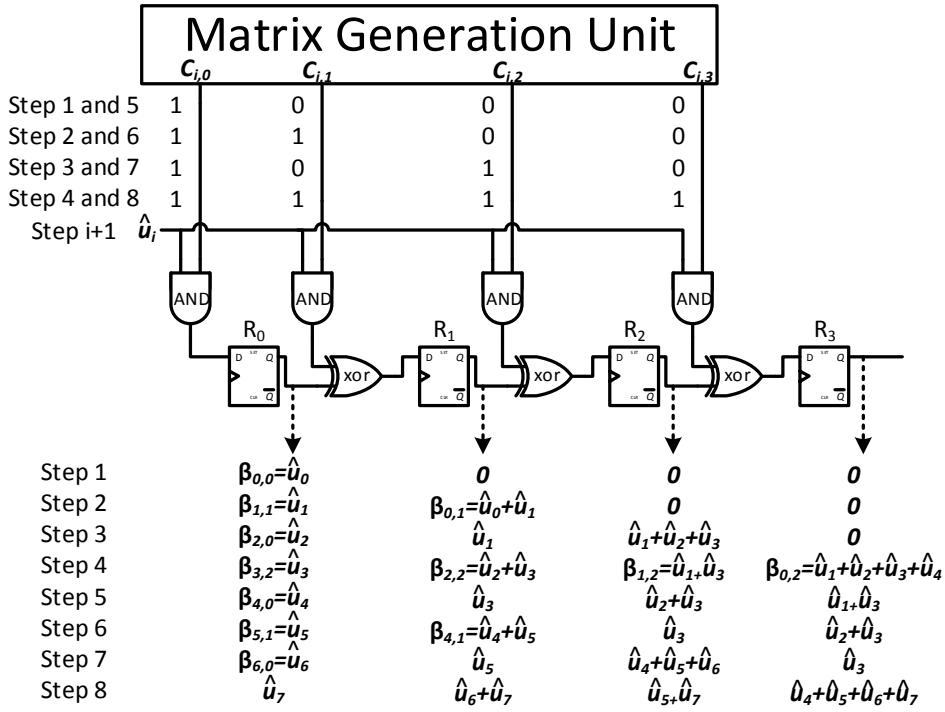
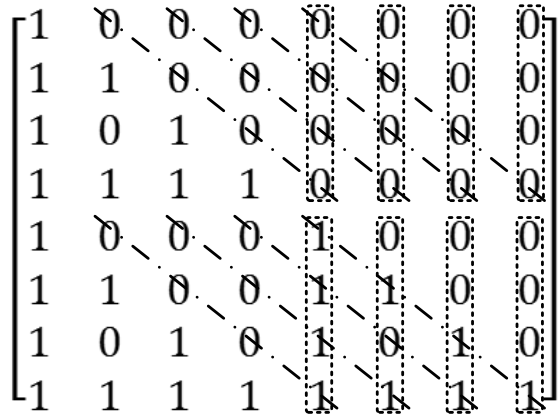


Figure 3.8: The architecture of SR-PSG

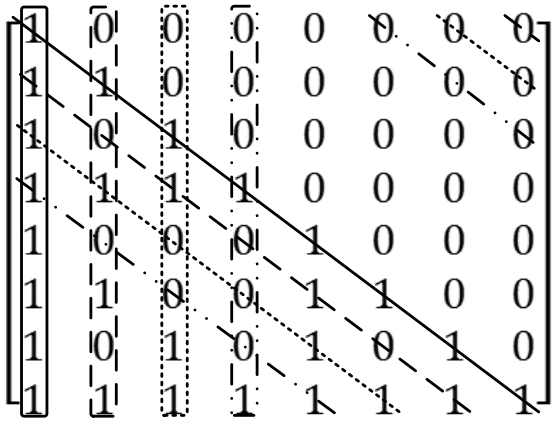
some partial sums are obtained directly instead of calculating from estimated bits. Thus, a PSG for constituent codes based SC decoder should have the capability to generate the new partial sums from either the directly got intermediate partial sums or the estimated bits, and to maintain the coherence of them.

3.4.1 Mathematical Derivation

For a length n constituent code, its corresponding estimated bits and partial sums are denoted as $\hat{u}_{i-n+1}^{c} \ \ldots \ \hat{u_i}^{c}$ and $\beta_0^c \ldots \beta_{n-1}^c$, respectively. All the $\boldsymbol{\beta^c}$ are obtained at the same time. For those bits who are not belongs to any constituent codes, we still have to calculate their corresponding partial sums accroding to Eq. (3.1). Thus, if we still want to keep consistent between directly got intermediate partial sums and the one-by-one-estimated bits, we need to derive the mathematical presentation with $\boldsymbol{\beta^c}$ from Eq. (3.1).



(a)



(b)

Figure 3.9: (a) Elements shift in generation matrix, and (b) Diagonal cycle-shift in generation matrix

For $k \geqslant n$ and $k \in [a \cdot n, (a+1) \cdot n - 1], a = 1, 2, \ldots$, according to Eq. (3.1), we

have

$$\begin{aligned} R_k &= R_{k-1} \oplus (\hat{u}_i^c \cdot c_{i,k}) \\ &= R_{k-2} \oplus (\hat{u}_{i-1}^c \cdot c_{i-1,k-1}) \oplus (\hat{u}_i^c \cdot c_{i,k}) \\ &\cdots \\ &= R_{k-n} \oplus \\ &\left(\begin{bmatrix} \hat{u}_{i-n+1}^c, & \cdots, & \hat{u}_i^c \end{bmatrix} \begin{bmatrix} c_{i-n+1,k-n+1} \\ \vdots \\ c_{i,k} \end{bmatrix} \right). \end{aligned} \tag{3.2}$$

As we know, $c_{i,k}$ is the element of generation matrix *G* which is the Kronecker power of $\boldsymbol{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. Combine this property with our observation on the matrix, we conduct the following rule which is also noted in Fig. 3.9a.

$$\begin{bmatrix} c_{i-n+1,k-n+1} \\ \vdots \\ c_{i,k} \end{bmatrix} = \begin{bmatrix} c_{i-n+1,(a+1)\cdot n-(k \bmod n)-1} \\ \vdots \\ c_{i,(a+1)\cdot n-(k \bmod n)-1} \end{bmatrix}. \tag{3.3}$$

According to the definition of generation matrix and concept of constituent code, when $c_{i,k} = 0$, the right part of Eq. (3.3) is equal to a all $\boldsymbol{zero}$ vector, and when $c_{i,k} = 1$ the right part of Eq. (3.3) is equal to the $(n - (k \bmod n) - 1)$th column in the generation matrix for length n polar code. According to the definition of partial sum and Eq. (2.14), we get

$$[\hat{u}_{i-n+1}^c, \cdots, \hat{u}_i^c] \cdot [c_{i-n+1,p(k)}, \cdots, c_{i,p(k)}]^T = \beta_{p(k)} \tag{3.4}$$

where $p(k) = (n - (k \bmod n) - 1)$.

Now we apply above observation back to Eq (3.2). We define the vector $\boldsymbol{R_a} = [R_{a\cdot n}, \cdots, R_{a\cdot n+n-1}]$ and $\boldsymbol{c_{i,a}} = [c_{i,a\cdot n}, \cdots, c_{i,a\cdot n+n-1}]$ for $k \in [a \cdot n, (a+1) \cdot n -$

$1], a = 1, 2, \ldots$. We also define the vectors $\hat{\boldsymbol{u}}^{\boldsymbol{c}} = [\hat{u}^c_{i-n+1}, \cdots, \hat{u}^c_i]$ and $\hat{\boldsymbol{\beta}}^{\boldsymbol{c}} = [\beta^c_{n-1}, \ldots, \beta^c_0]$. Then, we have

$$
\begin{aligned}
\boldsymbol{R_a} &= [R_{a\cdot n}, \cdots, R_{a\cdot n+n-1}] \\
&= [R_{(a-1)\cdot n}, \cdots, R_{a\cdot n-1}] \oplus \\
&\left([\hat{u}^c_{i-n+1}, \cdots, \hat{u}^c_i] \begin{bmatrix} c_{i-n+1,a\cdot n-n+1} & \cdots & c_{i-n+1,a\cdot n} \\ \vdots & \ddots & \vdots \\ c_{i,a\cdot n} & \cdots & c_{i,a\cdot n+n-1} \end{bmatrix} \right) \\
&= [R_{(a-1)\cdot n}, \cdots, R_{a\cdot n-1}] \oplus \\
&\left(\hat{\boldsymbol{u}}^{\boldsymbol{c}} \begin{bmatrix} c_{i-n+1,p(a\cdot n)} & \cdots & c_{i-n+1,p(a\cdot n+n-1)} \\ \vdots & \ddots & \vdots \\ c_{i,p(a\cdot n)} & \cdots & c_{i,p(a\cdot n+n-1)} \end{bmatrix} \right) \\
&= \begin{cases} \mathbf{0}, if\ \boldsymbol{c_{i,a}} = \mathbf{0} \\ \boldsymbol{R_{a-1}} \oplus \cdot\boldsymbol{\beta^c}, if\ \boldsymbol{c_{i,a}} = \mathbf{1} \end{cases}
\end{aligned}
\tag{3.5}
$$

For the consistent with Eq. (3.1), we rewrite Eq. (3.5) as follow:

$$
\boldsymbol{R_a} = \boldsymbol{R_{a-1}} \oplus (\boldsymbol{\beta^c} \& \boldsymbol{c_{i,a}}) \tag{3.6}
$$

where $\&$ stands for the bit-wise and operation.

For $0 \leqslant k < n$, similar to Eq. (3.2), we have

$$
R_k = [\hat{u}^c_{i-k}, \cdots, \hat{u}^c_i] \cdot [c_{i-k,0}, \cdots, c_{i,k}]^T \tag{3.7}
$$

According to the definition of *G* and constituent codes, we can conduct that for any length

n constituent codes, the first n columns of its corresponding rows in G should also be a generation matrix G_n for length n polar code. As described in Fig. 3.9b, the diagonal cycle shift is same as each correspond column, and consider the G_n is a lower triangular matrix, we get

$$\begin{aligned}
&[c_{i-n+1,k+1}, \cdots, c_{i-k-1,n-1}, c_{i-k,0}, \cdots, c_{i,k}]^T \\
&= [c_{i-n+1,n-1-k}, \cdots, c_{i,n-1-k}]^T \\
&= [0, \cdots, 0, c_{i-k,n-1-k}, \cdots, c_{i,n-1-k}]^T
\end{aligned} \tag{3.8}$$

Thus, Eq. (3.7) can be rewritten as:

$$\begin{aligned}
R_k &= [\hat{u}^c_{i-k+1}, \cdots, \hat{u}^c_i] \cdot [c_{i-k,0}, \cdots, c_{i,k}]^T \\
&= [\hat{u}^c_{i-n+1}, \cdots, \hat{u}^c_i] \cdot [0, \cdots, 0, c_{i-k,0}, \cdots, c_{i,k}]^T \\
&= [\hat{u}^c_{i-n+1}, \cdots, \hat{u}^c_i] \cdot [c_{i-n+1,n-1-k}, \cdots, c_{i,n-1-k}]^T \\
&= \beta^c_{n-k-1}
\end{aligned} \tag{3.9}$$

Thus, combine Eq. (3.9) and Eq. (3.6), we derive the mathematical presentation for partial sum of constituent based polar decoder as follow:

$$\boldsymbol{R_a} = \begin{cases} \boldsymbol{\beta^c},\ if\ a\ =\ 0 \\ \boldsymbol{R_{a-1}} \oplus (\boldsymbol{\beta^c} \& \boldsymbol{c_{i,a}}), if\ a \geqslant\ 1. \end{cases} \tag{3.10}$$

3.4.2 Proposed Architecture

According to Eq (3.10), the shift-register constituent-code based partial sum generator (SR-CB-PSG) is proposed as in Fig. 3.10. Compared with Fig. 3.8, there are three differences. The first difference is the input. For SR-PSG, only current estimated bit is sent into, which means the input is only from the PU from stage 0. However, for SR-CB-PSG, the inputs are from PUs of any stage, depends on the length of constituent code. Thus,

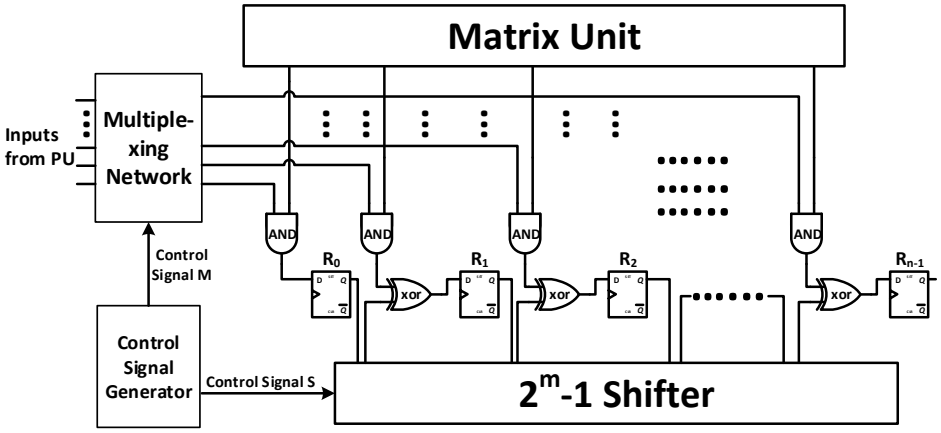


Figure 3.10: Overall architecture of SR-CB-PSG

a multiplexing networking is needed to route all the inputs values to the right registers. The second difference is about the shift function. According to Eq (3.10), instead of just shifting by one bit, the shifter should have the capability to shift n-bit where n is the length of constituent code. According to the definition of constituent code, n should be the any power of 2. Thus, A specific design (2^m-1)-bit shifter is proposed. The control signals for both the muxing networking and shifter are from the Control Signal Generator (CSG) with simple logic. The last deference is matrix generation unit. For each constituent code, its corresponding $c_{i,j}$ should be the *i*th row of the generation matrix, where *i* is the index of the last bit in the constituent code. Due to the irregularity of the constituent code, it's unnecessary to build an online generator for that. Thus, a pre-calculated ROM is placed. It's a trade-off between design complexity and hardware resource. It can be replaced by a re-configurable memory device like RAM for flexibility.

Fig. 3.11 shows an example of partial sum routing for 8 bit constituent code based polar code. We can see each register has specific corresponding PU from each stage. The essential of multiplexing networking is to route the partial sums to the right register. Noticeably, there are two output form stage 0 since pre-computing is used. For length n

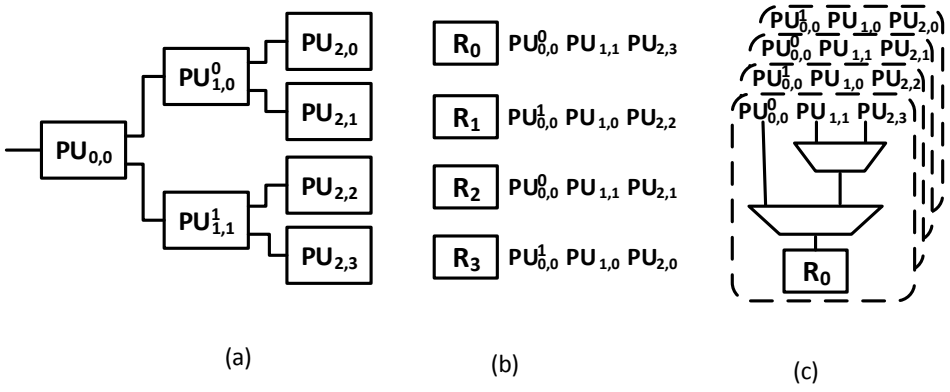


Figure 3.11: (a) PU tree of SC decoder, (b) PUs and their corresponding register, and (c) Architecture of multiplexing network

polar code, there are $\log_2 n$ stages in the decoder and n/2 registers in the SR-CB-PSG. If the multiplexing networking is built from the basic 2-bit MUX, each register is assigned an identical MUXs networking made by $(\log_2 n\ -\ 1)$ MUXs. All the networkings share the same control signal. According to its architecture, the control signals are the direct binary mapping of its stage index. Totally, $n/2 \cdot (\log_2 n\ -\ 1)$ MUXs are needed. Since the multiplexer networking needs to wait each PU finish computing to get the valid inputs, it is on the critical path of the decoder. Thus, it causes additional $\lceil \log(\log n) \rceil \cdot \triangle(MUX)$ delay, where $\triangle(MUX)$ is the delay for a single MUX.

For (2^m-1) shifter, we proposed a barrel-shiftr-based architecture for that. For length n polar code, $m \leqslant \log_2 n/2$. The shifter performs logic right shift. For $k < n_c$, where *k* is the index of the register and n_c is the length of the current constituent code, 0s are added to the left. For $k \geqslant n_c$, we do shift. Those behaviors satisfy the first and second in Eq (3.10).

Fig. 3.12 shows an example of (2^m-1) shifter for 16-bit polar code decoder. All the MUXs in the same row can shall the same control signal. Those signals is generated by a k to 2^k decoder, where $k = \lceil \log_2(\log_2 n) \rceil$ for length n polar code. For length n polar code, there are $(n/2 - 1) \cdot (\log_2 n\ -\ 1)$ MUXs are needed for the shifter. Since the shifter

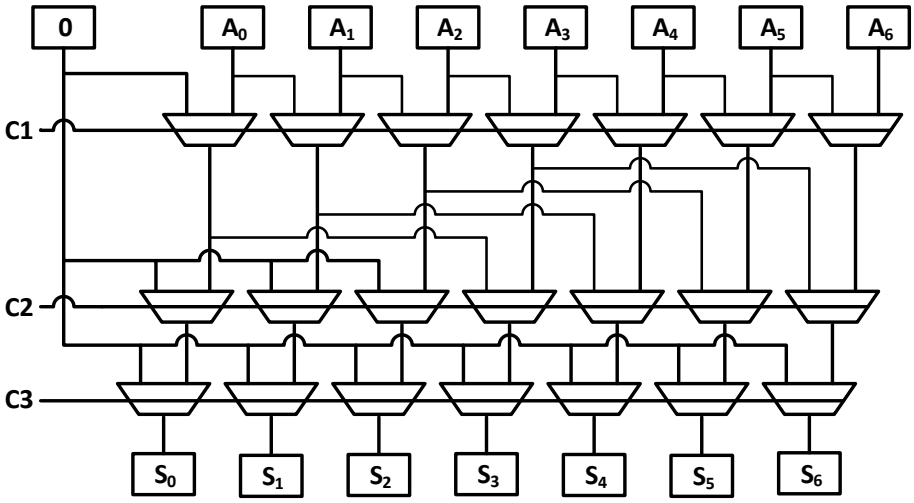


Figure 3.12: An example of (2^m-1) shifter for 16-bit polar code decoder

can start shift data without waiting PU to finish computing, it is not on the critical path. Thus, it should not deteriorate the timing performance of the decoder at all.

3.5 Fix Point Analysis

Fig. 3.13 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. Since no multiplication or division used, which means the length of fraction does not change, channel outputs and inner LLRs use the same fraction precision. As the result of the trade-off between hardware efficiency and decoding performance, we choose (4,5,0) quantization scheme in our design.

3.6 Implementation Results and Relevant Discussion

To the best of our knowledge, the proposed design is the first PSG design especially design for constituent codes based SC decoder. Thus, there is no reference design we can

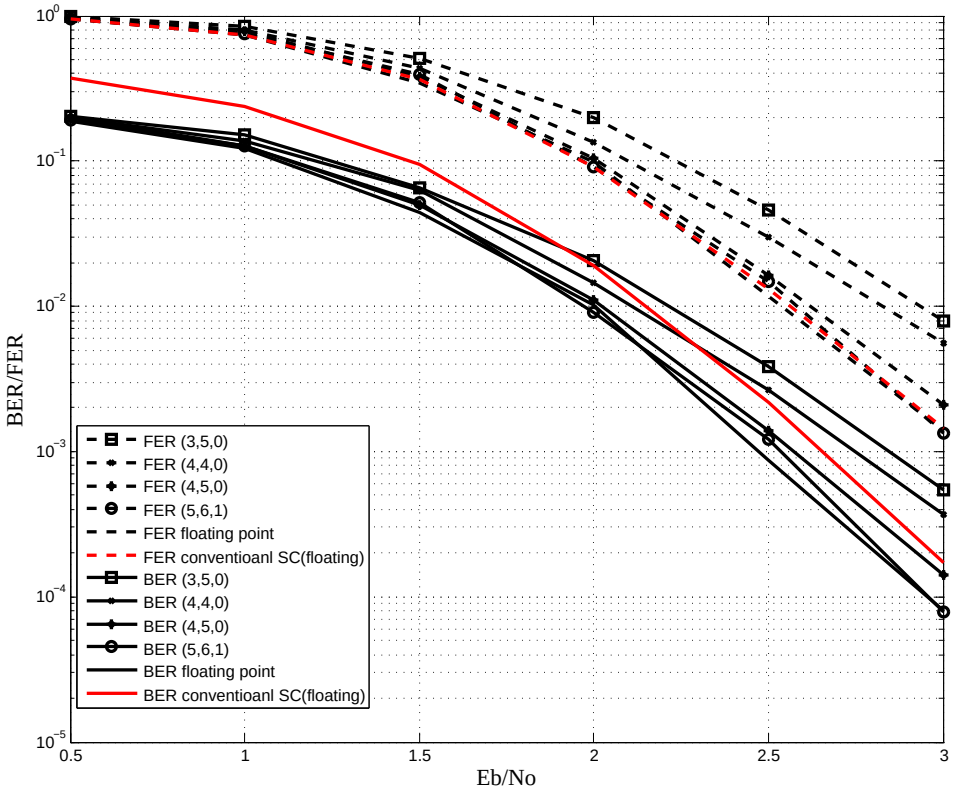


Figure 3.13: Effect of quantization on the BER/FER performance of (1024,512) code

directly compare with. In this section, we list all the result we have and presents some relevant discussions.

Table 3.4: Critical path comparison

	Critical Path
SR-PSG[42]	$\triangle(AND) + \triangle(XOR)$
DM-PSG	$\lceil \log_2(\log_2 n) \rceil \cdot \triangle(MUX) + \log_2 n(\triangle(MUX) + \triangle(XOR))$
SR-CB-PSG	$\lceil \log_2(\log_2 n) \rceil \cdot \triangle(MUX) + \triangle(AND) + \triangle(XOR)$

Table 3.4 shows the critical path comparison between proposed PSG and the PSG in [42]. We can tell the delay overhead comes from the muxing network. Ideally, the

Table 3.5: Decoder latency comparison for length=1024 polar code

	code rate				
	0.2	0.35	0.5	0.65	0.8
conventional [30]	767				
constituent code based	263	298	266	200	160
reduction(%)	65.7	61.1	65.3	73.9	79.1

maximum frequency of constituent codes based SC decoder is lower than that of conventional SC decoder. However, after taking the latency reduction into account, as shown in Table 3.5, constituent codes based SC decoder is able to achieve much higher throughput. The conventional SC decoder is referred from [30] which is the lowest latency conventional SC decoder to the best of out knowledge.

Table 3.6 shows the resource consumption estimation of proposed SR-CB-PSG for length n polar code decoder and the comparison with other two conventional PSG. The most resource consumption part is the MUX since it used in both multiplexer networking and shifter. The estimation for the ROM size is based on the average calculation since the decoding latency changes along with the code rate.

Table 3.6: Resource comparison

	SR-CB-PSG	DM-PSG	[42]	[29]
DFF	$n/2$	$n/2$	n	$(n^2-4)/12$
MUX	$(n-1)\cdot(\log_2 n-1)$	$n\cdot(\log_2 n-1)$	-	$n-2$
XOR	$n/2-1$	$n/4(\log_2 n-1)$	$n-2$	$n/2-1$
AND	$n/2$	-	$n/2$	-
ROM(bit)	$n^2/10$(average)	-	-	-

The proposed design can be targeted on either ASIC or FPGA. We synthesized both with Nangate FreePDK 45nm process and on Xilinx Kintex-7 FPGA KC705 Evaluation

board. Table 3.7 shows the hardware resource of SR-CB-PSG for 1024 code length polar code decoder on both of them.

Table 3.7: Hardware resource of SR-CB-PSG for 1024 code length polar code decoder

	XC7K325T-2FFG900C FPGA		nangate 45nm
Hardware Resource	slice LUTs	slice REGs	area
	$1569(< 1\%)$	$512(< 1\%)$	$16333\mu\text{m}^2$

Noticeably, the architecture we discussed in this paper is based on the consideration for the worst case, which is that the maximum length of constituent codes could be n/2. However, for practical application, the maximum length of constituent is fix for certain code rate and usually cannot approach n/2. For those case, the logic of both the multiplexer networking and shifter could be even simpler, which will result in a better timing and silicon area performance.

Table 3.8: Hardware comparison of different (n,k) SC decoder with q-bit quantization for inner LLRs using tree architecture

Hardware Type	[29]	[27]	[30]	Proposed Design
# of PU	$n-1$	$n-1$	$n-1$	$n-1$
# of PTU	0	0	0	$2/n-1$
# of 1 bit REG	$\approx 3qn$	$\approx qn$	$\approx 3qn$	$\approx (q+1)n$
HC	1.3	1	1.3	1.11
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	$6.02 \sim 20.05$

Table. 3.8 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 [27], 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 proposed design achieves the highest throughput per unit of hardware complexity. The exact latency depends on the code rate. Fig. 3.14 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.

Additionally, we implemented the proposed design with Verilog for the polar code with length=1024 and synthesized it using Nangate FreePDK 45nm process with Synopsys Design Complier. We calculated the throughput for (1024,870) and (1024,512) polar codes. Table 3.9 shows the synthesis result for (1024,870) and (1024,512) polar codes. Notice that the maximum frequency is higher than that reported in [30] which use the same process as our design. Our design in theory should have a lower maximum frequency since we have one more Mux delay for regular and fast constituent polar codes. This performance improving is attributable to the modification we have done to PU as described in section 3.3.

3.7 Construction Scheme with Constituent Codes Optimization

All the aforementioned works stress on the decoding sides. However, in this section, we explore the potential of constituent codes from an opposite angle. Since constituent code is an approach to reduce decoding latency, we stress on the construction scheme to make the codeword are more constituent-code-friendly. By adjusting the traditional con-

Table 3.9: 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

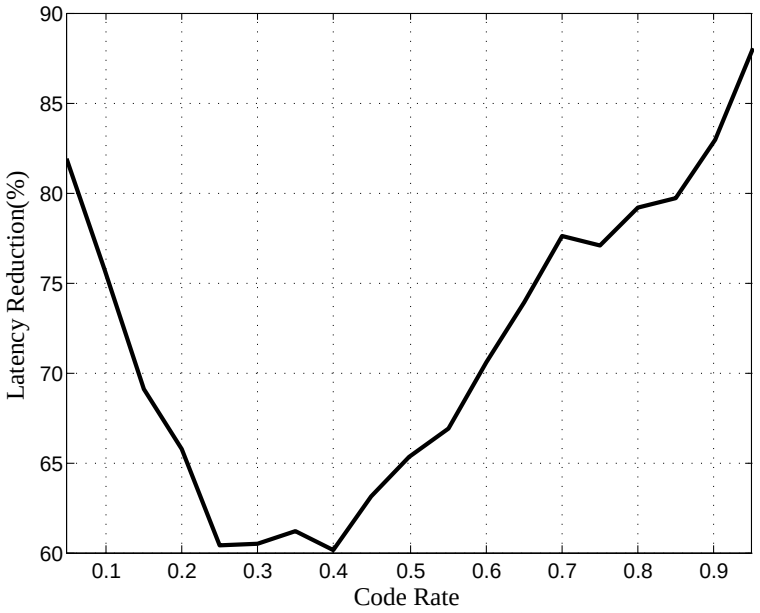


Figure 3.14: Latency reduction vs. code rate

struction approach, more expected types of constituent codes are made for decoding. For (n,k) polar code, instead of directly setting the k best and (n-k) worst bits to the information bits and frozen bits, respectively, we thoughtfully swap the locations of some information and frozen bits according to the quality of their equivalent channels. The corresponding algorithm to optimize the constituent codes division is also described. We conducted the simulation of 1024 and 2048 length polar codes with multiple rates and analyzed the decoding latency for various length codes. The numerical results shows that the proposed

construction scheme typically achieves 4-20% latency reduction with negligible loss in decoding performance with carefully selected optimization threshold. Some relevant discussions are also presented.

3.7.1 Constituent Codes Optimization

As described before, constituent codes are decoded faster than conventional polar codes. Thus, in order to reduce the decoding latency, we expect more constituents codes, especially constituent codes with large length. According to the definition of constituent codes, the initial distribution of constituent codes is determined by the location of information and frozen bits. If we can change the locations, we are able to manually produce expected constituent codes. However, the location of frozen and information bits are very sensitive, and random changes may cause negative influence on the coding performance. Thus, a thoughtful construction scheme to produce more expected constituent code is on demand. The division of information and frozen bits is decided by the qualities of the equivalent channels which are corresponding to each bit. Based on the channel model, the equivalent channel qualities can be calculated accordingly [7] [9] [54] [55]. This gives us a hint that if we swap some information and frozen bits those with similar channel qualities, this might only incur a very slight performance loss. The numerical simulation results in the following section prove this idea. In this work, binary erasure channel (BEC) is used as our channel model, and thus Bhattacharyya parameter is used as the metric for equivalent channel quality. This method can be extended to any other kind of channel model.

Now, we have the idea about how to change the division of constituent codes. Next, we need to consider what kind of changes are desired. For any length n polar code, it can be regarded as a combination of the following four types of sub-codewords.

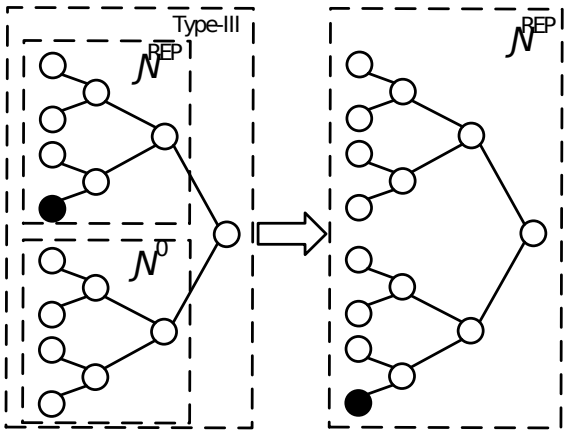
- Type-I: All the bits are frozen bits. This is also $\mathcal{N}^0$ constituent code.
- Type-II:All the bits are information bits. This is also $\mathcal{N}^1$ constituent code.

- Type-III: Only one bit is information bit, the rest are frozen bits. This can be regarded as the combination of one $\mathcal{N}^{REP}$ and multiple $\mathcal{N}^{0}$ constituent codes.

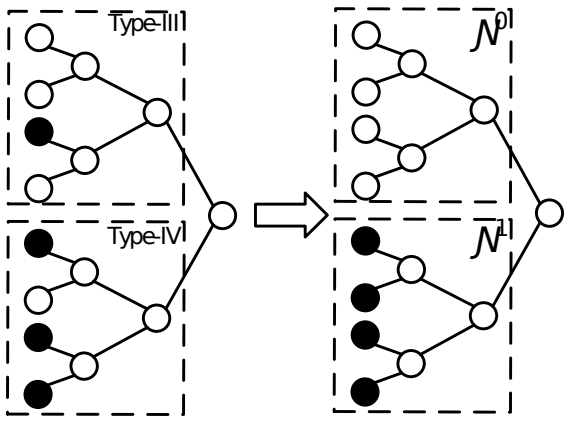
- Type-IV: Only one bit is frozen bit, the rest are information bits. This can be regarded as the combination of one $\mathcal{N}^{SPC}$ and multiple $\mathcal{N}^{1}$ constituent codes.

According to T. Che's results [39], there is only one clock cycle needed for decoding $\mathcal{N}^{0}$ and $\mathcal{N}^{1}$ node. Thus, it is unnecessary to optimize type-I and type-II codes since they are already fully optimized. Our target should be focused on the type-III and type-IV. These two types are similar, they all only have one node with different type with others. There are two situations we need to deal with. The first situation is the optimization for single type-III or type-IV sub-tree. We can swap the different node with the first or last one to make it a $\mathcal{N}^{SPC}$ or $\mathcal{N}^{REP}$ nodes for type-IV or type-III sub-tree, respectively. The second situation is that the optimization for a combination of type-III and type-IV sub-trees. For this case, we can swap the different node between the two types to make them became one type-I and one type-II sub-trees. For the first situation, suppose we have one Type-III node at stage m+1. It consist of one $\mathcal{N}^{REP}$ and one $\mathcal{N}^{0}$ node at stage m. This is shown in Fig. 3.15a. Totally, it needs $1 + \log_2 2^m = m + 1$ to finish decoding. If we move the place of the information bit to make it a $\mathcal{N}^{REP}$ constituent code, the new latency for decoding should be $\log_2 2^{m+1} = m + 1$. There is no change if we do this modification. This is also similar to type-IV situation. For the second situation, suppose we have one type-III and one type-IV nodes at stage m as shown in Fig. 3.15b, the totally latency for decoding should be 2m+1. If we swap the information bit in type-III and frozen bit in type-IV, we get one $\mathcal{N}^{0}$ and one $\mathcal{N}^{1}$ nodes. The total should be reduced to 2. This makes a huge difference. Thus, our target should be the swap operation between type-III and type-IV nodes.

Based on above discussion, the constituent code division optimization algorithm is



(a)



(b)

Figure 3.15: Examples of constituent codes division optimization

proposed in algorithm (1). After we get the information and frozen bits positions using the traditional way, we apply this algorithm to adjust the location of some information and frozen bits to make more desired constituent code. This is an enhancement to the traditional construction method.

3.7.2 Simulation Results

Fig. 3.16 shows the simulation results of proposed construction scheme. They are the ber vs Eb/N0 performance for 1024 and 2048 length at different rate with multiple

Algorithm 1 constituent code oriented polar code construction

Input: The set of Bhattacharyya parameter, $\boldsymbol{\epsilon_n}$; the set of bit property (frozen or information), $\boldsymbol{L}$; the threshold for bit swapping,T_h.

Initialize: $index = 1$.

get the sub-codeword-type look up table T from $\boldsymbol{L}$; this table gives the sub-codeword type information and its index
if $T[index]$ is type-III sub-codeword **then**
    get the index i of the information bit in $T[index]$, search all the next type-IV sub-codewords and find the one whose frozen bit's Bhattacharyya parameter has the minimum difference with $\boldsymbol{\epsilon_n}[i]$. Recode the index f of that.
    if $|\boldsymbol{\epsilon_n}[i] - \boldsymbol{\epsilon_n}[f]| < T_h$ **then**
        swap the property of $\boldsymbol{L}[i]$ and $\boldsymbol{L}[f]$, update T.
    end if
end if
if $T[index]$ is type-IV sub-codeword **then**
    get the index f of the frozen bit in $T[index]$, search all the next type-III sub-codewords and find the one whose information bit's Bhattacharyya parameter is has the minimum difference with $\boldsymbol{\epsilon_n}[f]$. Recode the index i of that.
    if $|\boldsymbol{\epsilon_n}[i] - \boldsymbol{\epsilon_n}[f]| < T_h$ **then**
        swap the property of $\boldsymbol{L}[i]$ and $\boldsymbol{L}[f]$, update T.
    end if
end if
increase $index$ by 1, repeat to 2, until to the end of T

optimization thresholds. The polar codes is constructed based on the BEC channel and with target erasure rate 0.3. Table 3.10 shows the decoding latency for each threshold and the comparison with constituent codes decoding without any optimization and other state-of-the-art decoders. We even calculated the latencies for the codes with very long length such as 16384.

According to Fig. 3.16 and Table 3.10, we note that the proposed construction scheme generally is able to achieve at least around 20% latency reduction with an negligible gain loss. For a certain code length, we can find that the acceptable threshold is increasing along with the code size. This is due to the nature of channel polarization. There are more frozen and information bits mixed in the front and middle part of codeword for lower rate codes, which gives more flexibility during the optimization. However, this does not indicate that this coding scheme is not working well on high rate. The interesting part is that the performance on high rate is as good as the low rate according to Fig. 3.16. It's possibly attributable to the following two causes. This first is that the coding performance of high rate itself is much worse than that of low rate. This cause the difference after optimization is not so obvious. The second one is that the simulation we run is still not long enough. This is per coding theory that suggests the longer polar code will generally yield higher polarization. For a certain code rate, we can find the acceptable threshold is decreasing along with the code length. Since the longer codes are more polarized, the difference of each equivalent channel is becoming smaller and smaller as the length is increasing. This works fine with low and medium length but not so obvious with high length. This also can be explained by the two reason presented before.

We compared the latency of proposed design with the decoder in [30] which is the fastest non-constituent-codes-based polar code decoder to the best of our knowledge. We can see even constituent codes decoder without any optimization is much faster than that. Our proposed construction scheme is capable of achieving 20% or more latency. This is

very important especially for very long code length.

It is noticed that another approach that changing the frozen set to reduce the constituent codes related latency is reported in [56]. Although the fundamental idea of our work and that in [56] are similar, these two target two different decoder architectures and proposed two different constituent code oriented polar code construction methods. Without prior access to [56], we independently propose our work in this dissertation.

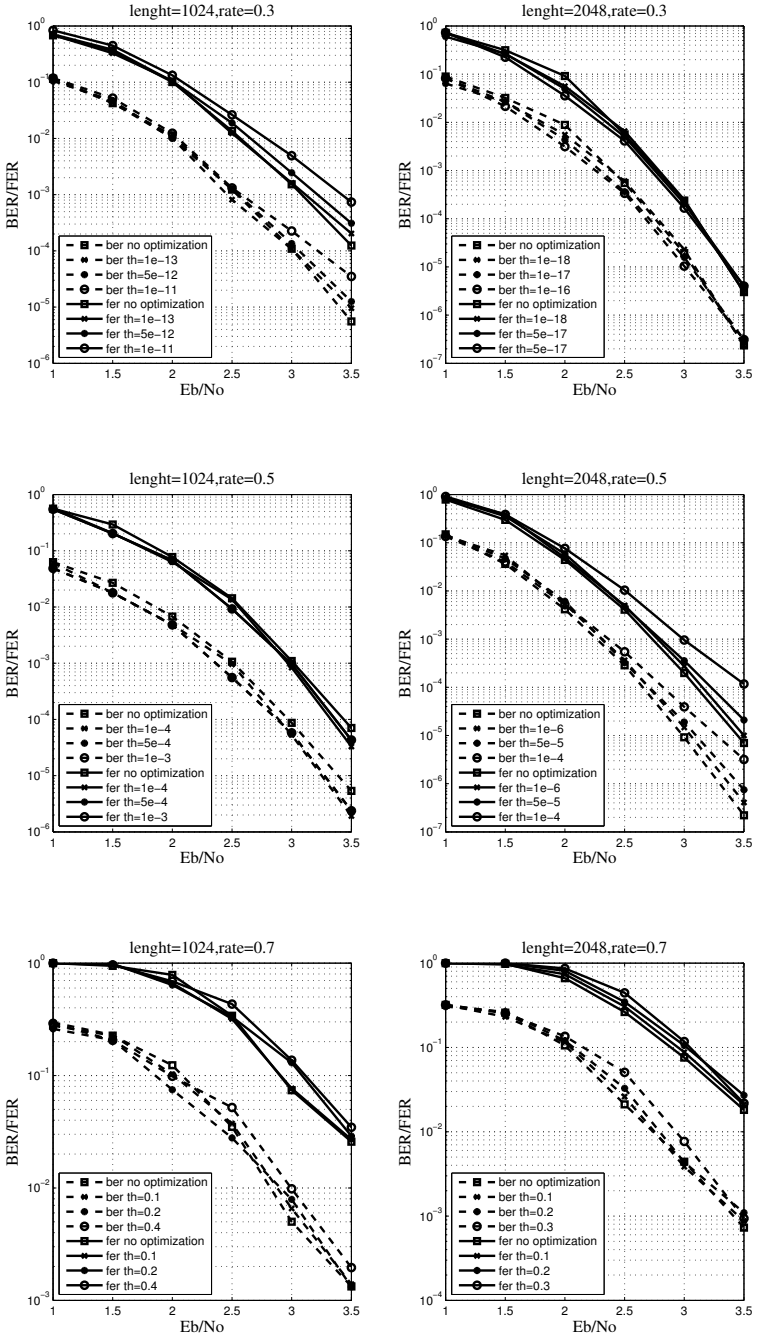


Figure 3.16: The ber vs E_b/N_0 performance for proposed construction scheme

Table 3.10: Latency reduction

decoder	length	rate	threshold	latency	reduction(%)
proposed	1024	0.3	no optimization	303	
			1e-13	288	4.9
			1e-12	260	14
			1e-11	234	22.7
		0.5	no optimization	266	-
			1e-4	255	4.1
			5e-4	218	18
			1e-3	197	21.8
		0.7	no optimization	172	-
			0.1	165	4
			0.2	137	20
			0.4	126	23.6
	2048	0.3	no optimization	576	-
			1e-18	549	4.6
			1e-17	519	9.9
			1e-16	493	14.4
		0.5	no optimization	493	-
			1e-6	487	1.2
			1e-5	436	10.5
			1e-4	323	33.6
		0.7	no optimization	297	-
			0.1	269	9.4
			0.2	248	16.5
			0.3	228	23.2
	16384	0.3	no optimization	3992	-
			1e-50	3661	8.3
			1e-45	3242	18.8
			1e-40	2721	31.8
		0.5	no optimization	3327	-
			1e-13	3187	4.2
			1e-12	2898	12.9
			1e-11	2465	25.9
		0.7	no optimization	1350	-
			0.1	1260	6.6
			0.2	1165	13.7
			0.4	898	33.4
[30]	1024	-		767	-
	2048			1535	
	16384			12287	

4. OVERLAPPED LIST SUCCESSIVE CANCELLATION APPROACH [1]

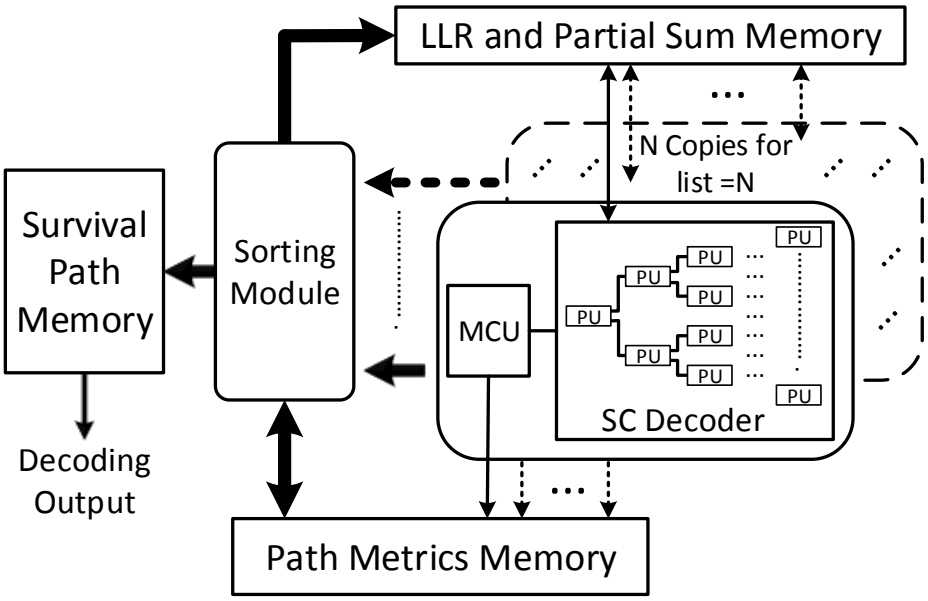


Figure 4.1: The conventional architecture of LSC decoder

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

[1]Reprinted with permission from "Overlapped list successive cancellation approach for hardware efficient polar code decoder" by Tiben Che, Jingwei Xu and Gwan Choi, 2016. *Proceedings of the 2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, Page 2463-2466, ©2016 IEEE.

the last information bit is decoded.

The main architecture of aforementioned LSC decoders are similar, as described in Fig. 4.1. 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 duplications of SC decoder yield a significant hardware redundancy of LSC decoder design. In our proposed design, we are trying to avoid such unnecessary redundancy.

In this section, we present our path-overlapping approach and discuss how performance optimization is carried out. Fig. 4.2 shows the architecture of proposed approach and the examples of the modified architecture of SC decoders associated with the list sizes two and four. Since the duplications of SC decoder involves the most hardware area consumption which is referred as hardware complexity, we removed all the copies and kept only one SC decoder. However, this modification of architecture does not mean that we just simply change parallel computing to a single-threaded lazy serial approach that computes one path at a time. Instead, every path is computed simultaneously in the decoding threads by judiciously utilizing the decoder hardware as follows: The processing timing of each path is overlapped with others in the pipeline arrangement. The architecture of SC decoder is modified to support this new paradigm. Since modifications are made only on architecture and scheduling plan, no decoding performance gain loss or change is incurred. The sorting module, MCU, and related memory components are compatible with other LSC decoders, and the partial sum generator is scheduled a similar way to be compatible with the path-overlapping SC decoder. Thus we do not discuss that in this paper. In the next subsections, the details of the scheme and the specific SC decoder are discussed.

4.1 Path-Overlapping Scheme and Relevant Analysis

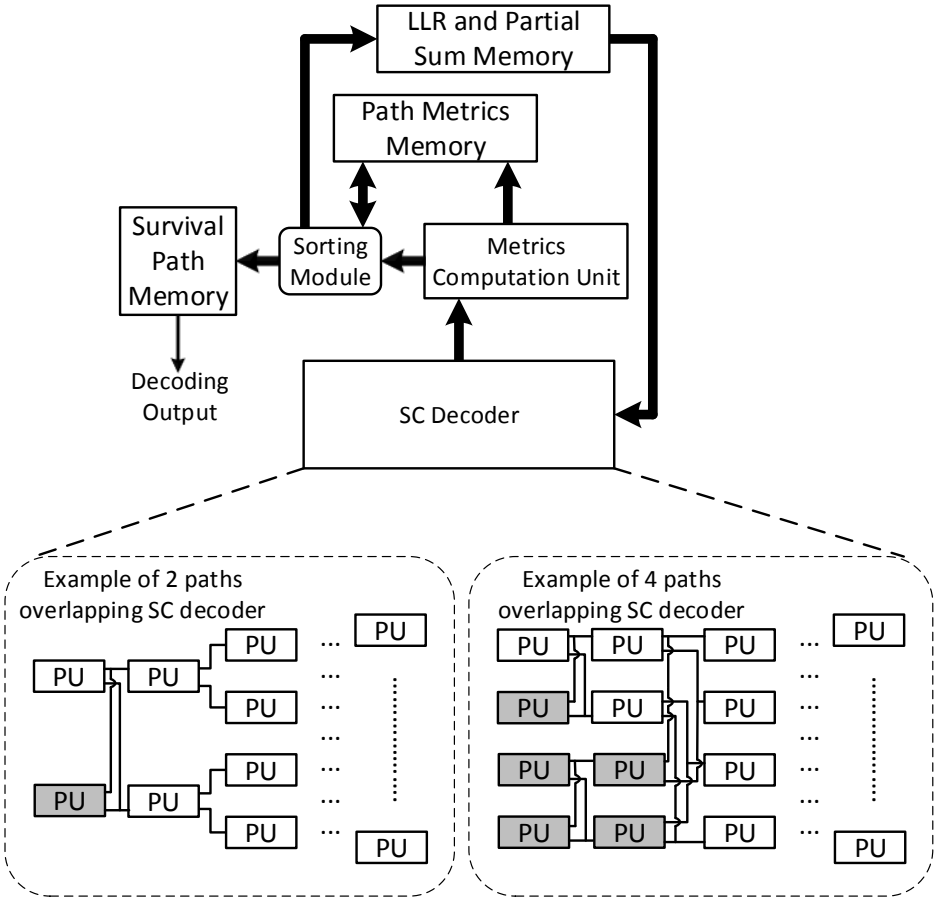
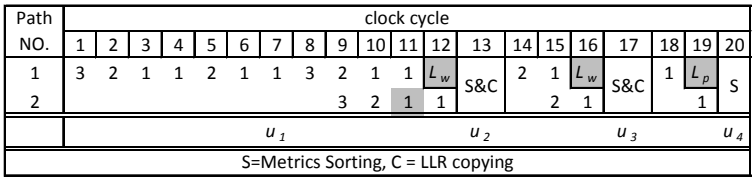
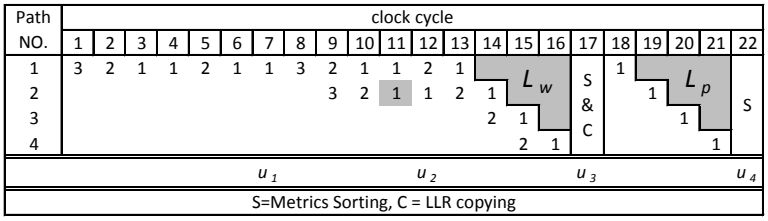


Figure 4.2: The architecture of proposed design

Simultaneous processing approach for SC decoder is already presented [29]. It is used for multiple frames in order to increase the throughput. The SC decoder with tree architecture consists of multiple processing units (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 activating multiple decoding stages in one clock cycle by feeding



(a)



(b)

Figure 4.3: Decoding schedule of the path-overlapping scheme for (8,4) polar code with (a) list size = 2 and (b) list size = 4

in several frames in pipeline. This means that each frame comes into the decoder with one clock cycle delay.

Stemming from the above idea, we realize that the duplications of SC decoder in conventional LSC decoder is unnecessary. 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 sense with only one clock cycle delay. However, the decoding scheme is not exactly the same as multiple frames overlapping SC decoder. Fig. 4.3a and Fig. 4.3b show the decoding schedule of two and four path-overlapping scheme, respectively. The numbers mean current activated stages, and the duplicated stage is marked with gray. The $S\&C$ means merits sorting and coping. According to [29], if a SC decoder is with l path-overlapping scheme, where $l \leq (2^i - 1)$, it can be constructed by duplicating $(2^{i-1} - 1)$ stages, where the index starts from the information bits side with respect to the tree architecture. The duplication plan is also shown in Fig. 4.2. Noticeably in Fig. 4.3b there is only one duplication of stage one,

which is not the same as what presented in Fig. 4.2. This is because the number of copies in Fig 4.2 are the minimum requirement for all the case. The actual requirement is decided by the code length and rate. Fig. 4.3b is just a certain case only one stage duplication is needed for four path-overlapping scheme.

Such architecture significantly reduces hardware complexity. Another advantage of proposed approach is that it can reduce the critical path length of decoder. Typically, the critical path lies in the sorting block. For 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 combination logic for the same list size. This means, by applying proposed approach, LSC decoder can run at a much higher frequency.

Although proposed approach can achieve a higher frequency compared with that of the conventional LSC decoder, there are few additional clock cycles introduced. These consist of two parts. The first part is the path pipeline latency L_p. Since all the paths are fed into decoder with one clock cycle delay, for the LSC 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:

$$L_m = L_w + L_p = (k - \log_2 l) \cdot (l-1). \tag{4.1}$$

This design approach can be applied to any current existing LSC decoders. It significantly reduce the hardware complexity by eliminating redundant instances, and it incurs only a few additional clock cycles to achieve the improvement. Henceforth, it is difficult to evaluate such design approach merely in term of the usage of hardware resource or the latency. We introduce the hardware efficiency (HE) metric which is noted as *e* to measure the performance of proposed approach. The *e* is defined as: *e* = *Throughput/Area*.

From Eq. (4.1), we can tell 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 proposed approach, 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.

4.2 Latency Reduction

4.2.1 Latency Reduction via Multi-Decision List SC Decoding

The first part of Eq. (4.1) 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 [31] [57] [39] [58]. 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 [31] [57]. 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 [58] belong to this type. It simplifies the SC decoding by finding certain pattern in the codewords. Such subcodes with certain

pattern also refer to 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.

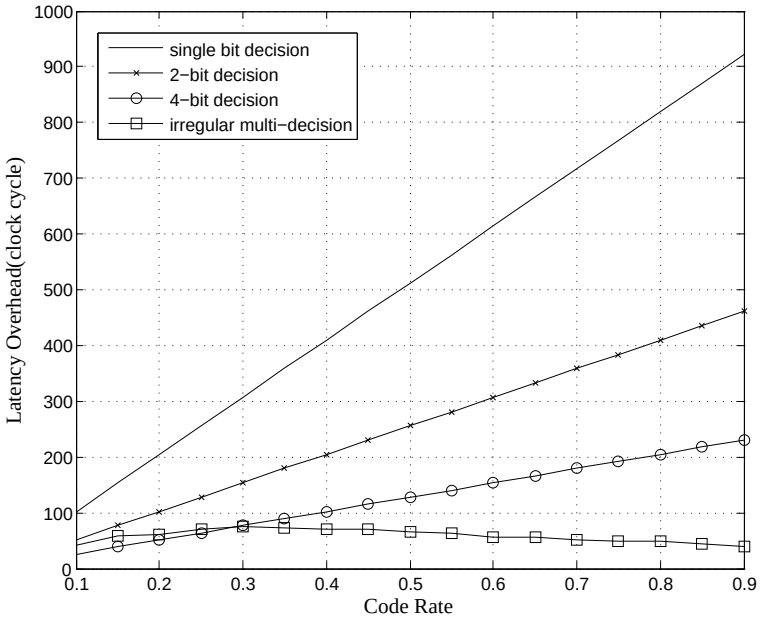


Figure 4.4: Latency overhead for different scheme

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. Fig. 4.4 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.

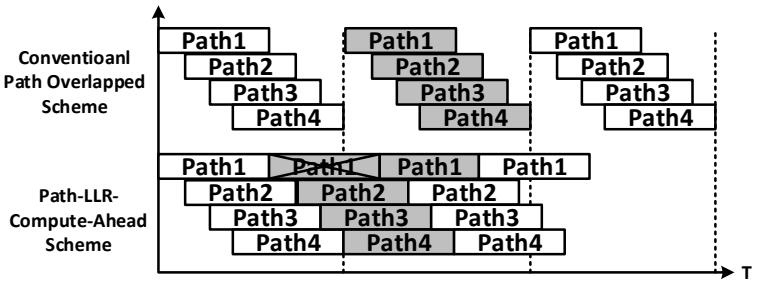


Figure 4.5: Decoding schedule of path-LLR-compute-ahead scheme

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

Besides reducing the number of estimations, the other approach to decrease latency overhead is by avoiding the pipeline stalling. This can be done via path-LLR-compute-ahead scheme (PLCAS). Fig. 4.5 shows this decoding schedule. A single bar means the decoding process between estimations of two successive information bits. When pipeline stalling occurs in one path, instead of waiting, current path can do a pre-estimating 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.

4.2.3 Latency Reduction via Adaptive LSC Decoding

In Eq. (4.1), the second part of the formulation is equal to the L_p. It is determined by the number of paths set in the pipeline. This makes the latency overhead increas linearly with respect to the list size l. If we can decrease this value, the latency overhead can be significantly reduced. Typically, L_p is fixed for a LSC with given length. However, by applying adaptive LSC algorithm [59], the L_p is allowed to change on the fly according to

current metrics of each path. The list size would decrease during the decoding processing, which in retrospect reduces the latency overhead.

In [59], basic hardware architecture is also proposed. Even though the list size would decrease along the decoding processing, the architecture proposed in [59] still needs l copies of SC decoder for its initial status. The usage of hardware resource is same as regular LSC decoder. Proposed approach can exploit the metric of adaptive LSC decoder via cutting down the unnecessary hardware complexity. With proposed approach there is no redundant hardware even when the list size decrease. Such property allows adaptive LSC decoder to benefit more in term of e. This will be shown in section. 4.3.

4.3 Performance Simulation and Analysis

Fig. 4.6 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 code, and the y-axis is the ratio of e with proposed approach over e with ordinary approach. The e with ordinary approach for a given LSC decoder has a consistent value. We apply proposed approach to four types of LSC 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 [33], [31], [58] and [39], the related synthesis results and the analysis we made in the previous sections.

In Fig. 4.6, 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

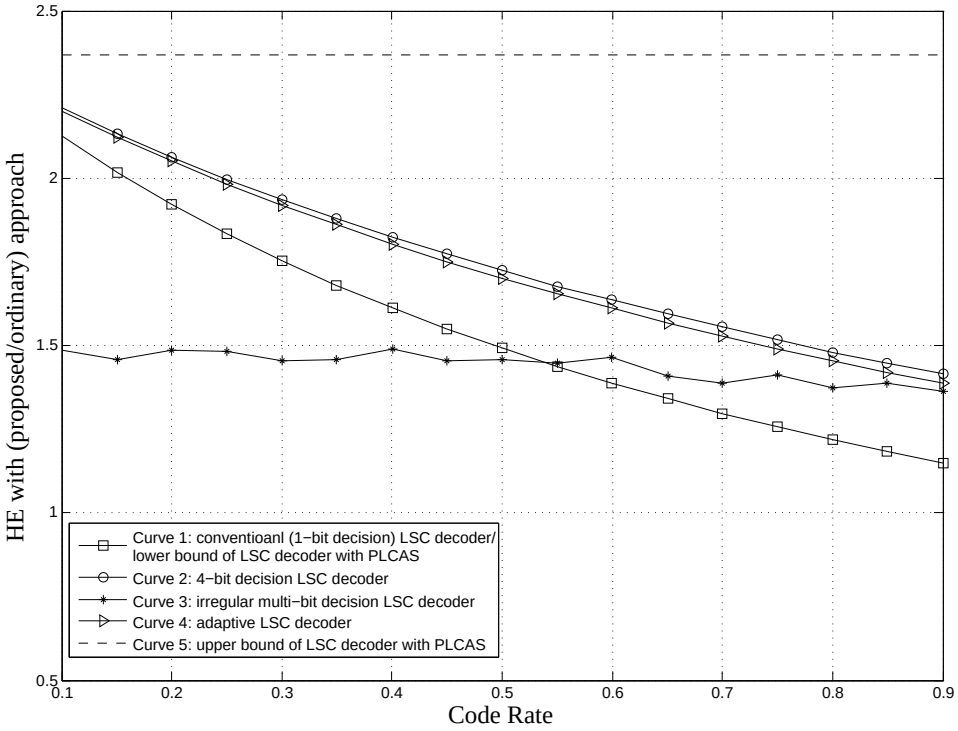


Figure 4.6: The improvement of hardware efficiency with proposed design approach

achieves more improvement of e than that of conventional (1-bit) decoder, which is due to the latency reduction as we described in section 4.2.1. 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. Such increasing benefits from the decreasing of latency overhead as we analyze in section 4.2. 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 that the inherent latency of irregular LSC decoder is already very low [39].

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 4.1, the maximum frequency of decoder with proposed approach should be measurably higher, which implies that the actual improvements of e in Fig. 4.6 should be even more pronounced in practice. Additionally, all the approaches mentioned above can argument each other. Using multiple approaches together can further increase the hardware efficiency. The presented design properties indicate that proposed approach can measurably contain the hardware complexity associated with large scale LSC decoder implementation.

5. EXPRESS JOURNEY BELIEF PROPAGATION DECODING FOR POLAR CODES [1]

In this section, a method by which the constituent codes are utilized to reduce the min-sum BP decoding complexity is presented. For those specific constituent codes, the message passing rules are innovatively simplified to reduce computational complexity for belief propagation.

Furthermore, to the best of our knowledge, all existing BP decoders schedule the computations in the same manner as mentioned in [46]. To explore an efficient BP decoder design, we derive an alternative scheduling method stemming from ideas discussed by Guo et al. at [12]. In [12], polar codes are proposed to be concatenated with parity check codes to achieve better decoding performances. We describe and compare the two different scheduling methods in this paper to show that the substitute scheduling method is significantly superior to the conventionally used one in terms of decoding efficiency.

We show that along with the alternative scheduling method, the complexity reduced min-sum BP algorithm yields almost same decoding performance of the SMS algorithm with substantially reduced amount of computations. Compared with the conventional MS BP decoding, our proposed method does reduce the computations by 91.5% as well as significantly improves the decoding performance.

5.1 Simplified Belief Propagation Decoding

In this section, we will present different types of constituent codes that exist in the factor graph of polar codes which could help reduce the complexity of BP decoding algorithm. The general idea of our algorithm is to refine the estimation of the transmitted

[1] Reprinted with permission from "XJ-BP: Express Journey Belief Propagation Decoding for Polar Codes" by Jingwei Xu, Tiben Che and Gwan Choi, 2015. *Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM)*, Page 1-6, ©2015 IEEE.

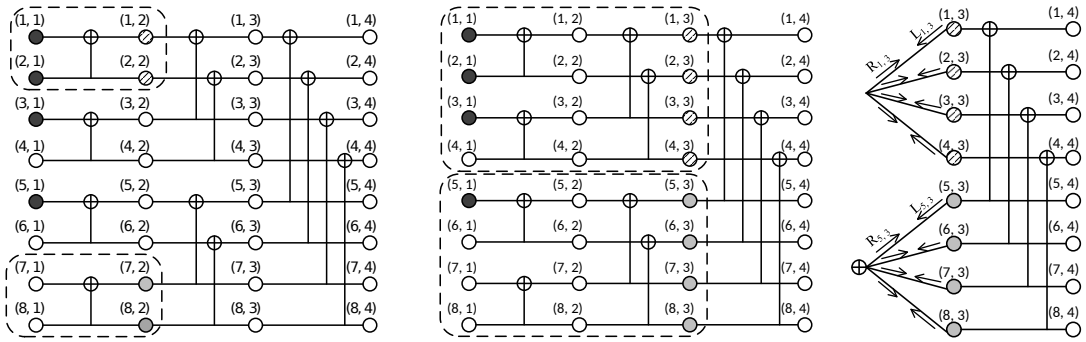


Figure 5.1: (a) An example of $\mathcal{N}^0$ codes in shadow and $\mathcal{N}^1$ codes in gray, (b) An example of $\mathcal{N}^{REP}$ codes in shadow and $\mathcal{N}^{SPC}$ codes in gray, and (c) The simplified factor graph for the example of $\mathcal{N}^{REP}$ and $\mathcal{N}^{SPC}$ codes.

codeword $\hat{\mathbf{x}}$ without traversing the whole factor graph in each iteration. The various constituent codes are studied in this section to simplify the factor graph as well as so as to reduce the decoding the complexity

5.1.1 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. Fig. 5.1a shows an example of $\mathcal{N}^0$ code, where the shadowed nodes of $\{$(1,2), (2,2)$\}$ compose a $\mathcal{N}^0$ code. Noticeably, the lengths of $\mathcal{N}^0$ codes vary significantly as the different rates and sizes of the polar codes. 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 0 in the encoding factor graph. Thus, by setting messages R of nodes $\mathcal{N}^0$ codes as ∞ before the decoding, the decoding could be performed in iterations without operating redundant processing elements left to the $\mathcal{N}^0$ codes.

5.1.2 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. We found the $\mathcal{N}^1$ codes do also help reducing the computation complexity. An existence of $\mathcal{N}^1$ code in the n=8 polar code example is given in Fig. 5.1a. 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. (2.20), it also shows that the $R_{i,j+1}$ and $R_{i+2^{j-1},j+1}$ do not get 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.

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

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 multiple times to construct the codeword. The repetition codes are referred as $\mathcal{N}^{REP}$ codes.

The example given in Fig. 2.7 does contain a $\mathcal{N}^{REP}$ code as shows in Fig. 5.1b, where the shadowed nodes {(1,3),(2,3),(3,3),(4,3)} constitute a $\mathcal{N}^{REP}$ code.

Since we already know that $\mathcal{N}^{REP}$ codes are formed by duplication, the conventional factor graph could be simplified without message passing through multiple message stages. The corresponding example of the factor graph of the $\mathcal{N}^{REP}$ code is given in the Fig. 5.1c, 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 [53] as:

$$R_{i,j} = \sum_{k \neq i} L_{k,j} \tag{5.1}$$

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 additions. Thus abundant existences of longer $\mathcal{N}^{REP}$ codes in polar codes reduce the amount of computations by the proposed simplified message passing rule.

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

The other type of constituent codes abundantly exist in polar codes is single parity check code. For those constituent codes who 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 Fig. 5.1b shows, the leaf nodes of the grayed constituent codeword $\{$(5,3),(6,3),(7,3),(8,3)$\}$ are all information bits except the first one. Similar as $\mathcal{N}^{REP}$ codes, it is unnecessary to roam thorough 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 Fig. 5.1c. The single parity check code consists of the bottom 4 nodes in the figure, where 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 could be written as:

$$R_{i,j} = \prod_{k \neq i} sgn(L_{k,j}) \cdot \min_{k \neq i} |L_{k,j}| \tag{5.2}$$

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 could be saved from the proposed algorithm.

Noticeably, the $\mathcal{N}^0$ and $\mathcal{N}^1$ codes are not usually included inside $\mathcal{N}^{SPC}$ and $\mathcal{N}^{REP}$ codes in reality. Simplifications of message passing on those four different types of constituent codes could be applied all together. The distributions of exclusive constituent codes in a (1024, 512) are shown in Table 5.1. As the table shows, there are considerable amount of constituent codes in the polar code. Especially, 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 rate of 0.5 polar codes contain relatively less number of constituent codes. With higher code rate, it is more motivated to apply the proposed methods to simplify the message passing. The details of complexity analysis will be presented in Section 5.3.2.

Table 5.1: Number of all constituent codes with different sizes in a (1024, 512) polar code with rate 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 conventional factor graph 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. (2.10). 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.

5.2 Scheduling

In this section, we present the two different ways to schedule the computations of conventional BP decoding algorithm. And the scheduling plan with the proposed BP decoding is illustrated as well. Finally, we give a method to early terminate the iterations of BP decoding.

5.2.1 Round-Trip BP Updating

As aforementioned, the computations of all existing conventional BP decoders are based on the processing element of Fig. 2.8. In the BP processing elements, the messages are computed for both directions of left-to-right and right-to-left simultaneously. Fig. 5.2a shows computations scheduled by the conventional BP decoding. As the figure shows, each iteration consists of *m* stages of computations, where $m = \log_2(N)$ 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, this scheduling method is lack of efficiency. For example, it is inefficient to update $L_{i,1}$ in time stamp 1 before having updated $L_{i,2}$ in time stamp 2.

Another way to schedule the computations is separately updating right-to-left message

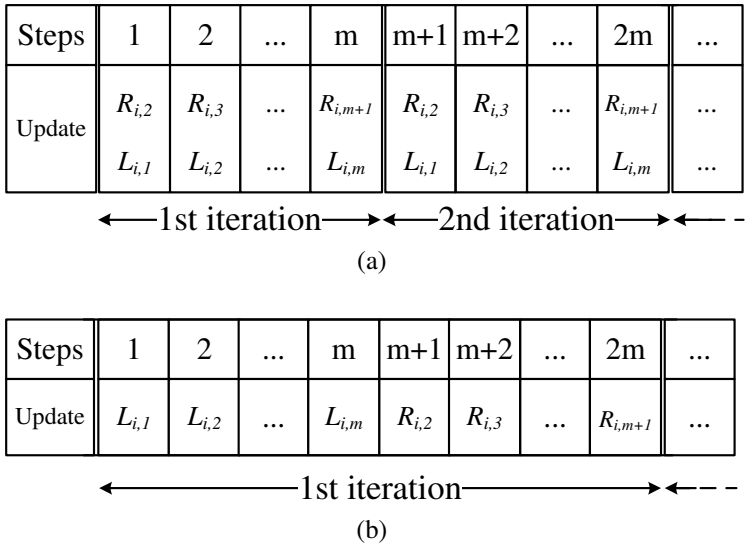


Figure 5.2: (a) Computations scheduled in the conventional BP decoders, and (b) Computations scheduled in a round-trip updating fashion.

and left-to-right messages. Fig. 5.2b shows the schedule of messages updated in this fashion. As the figure shows, the computations of each iterations are separated to two parts. In the first part, the $L_{i,j}$ messages are updated from column *m+1* to the most left nodes existing in the modified factor graph. The second is following to update the other direction message $R_{i,j}$ from left to 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 in this paper. Though each iteration of the modified scheduling contains a round-trip visiting 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, the round-trip scheduling significantly improves the efficiency in terms of number of iterations. Section 5.3.2 will discuss the number of iterations in details.

In this work, we promote the round-trip scheduling to update $R_{i,m+1}$ as mentioned above, because of the superior efficiency of it. However, different with conventional BP

decoding, for constituent $\mathcal{N}^{REP}$ and $\mathcal{N}^{SPC}$ codes, instead of Eq. (2.20), Eq. (5.1) and (5.2) are used to update messages R.

5.2.2 Early Termination

In this paper, we applied early termination to determine whether the decoding is successfully done or not.

Polar codes belong to the block codes. For block codes, H matrix could be used for codeword detection. According to the coding theory [60], 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:

$$\hat{\mathbf{x}}H = \mathbf{0} \tag{5.3}$$

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

$$\hat{\mathbf{x}}_i = \begin{cases} 0, & LLR_i^{\hat{\mathbf{x}}} > 0 \\ 1, & otherwise \end{cases} \tag{5.4}$$

5.3 Simulation and Discussion

In this section, we set up simulations to verify the proposed algorithm. Compared with the conventional BP decoding algorithm, the complexity and performance of the proposed algorithm are also analyzed and discussed in this section. As an example, (1024, 512) polar code is used to emulate the proposed decoder with max number of iterations of 60.

5.3.1 Decoding Performance

Fig. 5.3 shows the decoding performances of four decoding strategies. They are the conventional min-sum (MS) BP algorithm with conventional scheduling, the conventional

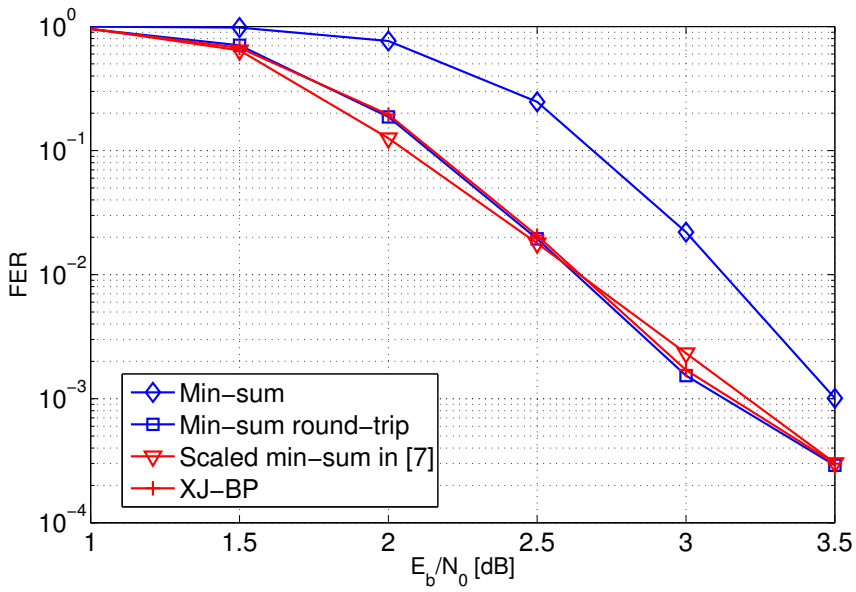


Figure 5.3: Decoding performance of the proposed BP decoding algorithm for (1024, 512) polar code with rate 0.5 and max number of iteration of 60.

MS BP algorithm with round-trip scheduling, the scaled min-sum (SMS) algorithm proposed in [48] with conventional scheduling and the proposed algorithm. As the results show, the min-sum BP decoding with the round-trip computation scheduling considerably outperforms the conventional min-sum algorithm. The performance of the min-sum BP algorithm with round-trip updating is very close to that of the scaled min-sum algorithm [48].

We also show that the proposed simplified BP algorithm yields almost same performance as the BP algorithm which roams through unpruned factor graph does. It means that the simplifications for constituent codes do not result in any degradation in decoding performance.

5.3.2 Computation Complexity Analysis

After showing the decoding performance of the proposed algorithm, here we explore complexity reduction by the proposed BP algorithm.

First of all, the average numbers of iterations of those algorithms are summarized in the Fig. 5.4. 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 in 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 computation 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 aforementioned in Section 5.2, the computations amounts of conventional scheduling and round-trip scheduling in each iteration are same. Compared with 24.5 average number of iterations, the decoding efficiency is immediately improved by 83.7% without considering the simplification on factor graph yet. Also, it is addressed that the proposed BP algorithm does not reduce the number of iterations compared with the round-trip BP scheduling without simplifying the factor graph.

Secondly, we evaluate the reduction of computations in each iteration resulting from the proposed simplified factor graph. As mentioned above, computations for nodes of $\mathcal{N}^0$ and $\mathcal{N}^1$ codes could be removed directly. While the computations of $\mathcal{N}^{REP}$ and $\mathcal{N}^{SPC}$ codes could be reduced by simplifying the factor graphs for them.

The numbers of total operations (2-input addition or 2-input comparison) are shown in the Table. 5.2. 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 used in decoding rate=0.5 polar codes could be

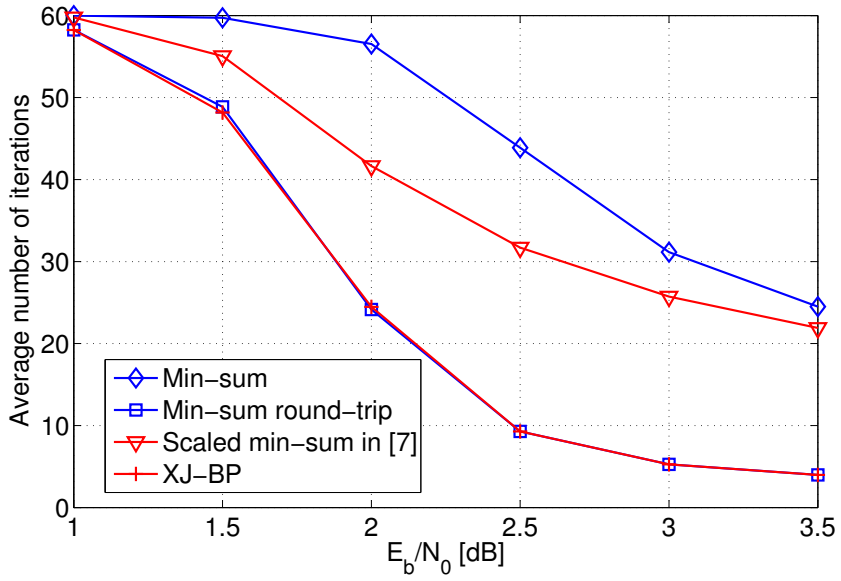


Figure 5.4: Average numbers of iterations of the proposed BP decoding algorithm for (1024, 512) polar code with rate 0.5.

Table 5.2: Number of computations of reduced-complexity 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
Proposed BP	1040	2488	5536	12160	27304
Ratios [%]	58.0%	60.9%	60.1%	59.4%	60.6%

reduced by around 40% in each iteration using the proposed simplified BP algorithm. And we found that this ratio keeps around 40% even with the codeword length size increasing to significantly long. 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. Because with more either frozen bits or information bits in the messages, the constituent codes happen more

Table 5.3: Comparison of computations at different code rates

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

frequently in the polar code. Table. 5.3 shows the number of computations for proposed algorithm decoding a polar code of length 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.

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, Fig. 5.5 shows the average number 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 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 proposed simplified factor graph so as to reduce the computations in each iteration by 40.6%. As a results, the overall complexity is reduced by 91.5% using the proposed algorithm with round-trip scheduling, compared with conventional BP decoding.

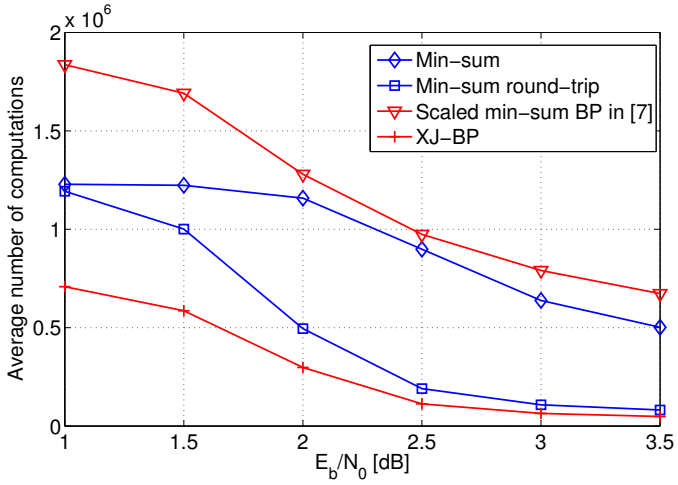


Figure 5.5: Average numbers of computations consumed to decode each codeword of by the proposed BP decoding algorithm for (1024, 512) polar code with rate 0.5.

5.3.3 Discussion

From the aspect of practical implementation, the conventional BP processing element symmetrically computes updates for messages R and L. Traditional computations for R are as same as those for L. 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.

The message updating rules are different between normal nodes and nodes of the specific constituent codes in mathematics. But the basic operations of additions and comparisons for them are very similar. Thus the proposed processing elements could be multiplexed between normal and specific constituent codes.

6. ASYNCHRONOUS CIRCUIT APPLICATION [1]

In the previous sections several novel design approaches of system or architecture level to enhance the performance of polar code decoder are proposed. In order to further strengthen the throughput and energy performance of polar code decoder, efforts from bottom circuit level have been made as well. Asynchronous circuit is a technique can yield a good power and reliability performance, which is suitable for channel coding system.

Asynchronous circuit is receiving an increased attention due to its low power consumption, high speed, less emission of electromagnetic noise, better modularity, omission of clock distribution related problems, and robustness with respect to variations in supply voltage, temperature and fabrication process parameters [51]. Asynchronous circuits that can mask 100% of the non-permanent faults is presented in [61]. A comparison between synchronous and asynchronous circuits under the scope of power-supply noise has been described in [62]. The obtained results demonstrate that the asynchronous circuit is significantly more robust than the synchronous one. A low-energy asynchronous interleaver for clockless fully parallel low-density parity-check (LDPC) decoder is presented in [63] that achieves 92% energy reduction. Asynchronous circuits are unlike the synchronous circuits that consist of registers and combinational logic. The transitions in a synchronous circuit are largely driven by a global clock which triggers all the registers at the same time. A significant design, performance, and energy considerations must be given to ensure correct register-to-register data forwarding. On the contrary, in asynchronous circuits, data forwarding from the input to the output ports are not controlled by an external signal

[1]Reprinted with permission from: (1) "Accelerated Dual-Path Asynchronous Circuit" by Tiben Che, Jingwei Xu and Gwan Choi, 2015. *IEEE Transactions on Circuits and Systems II: Express Briefs*, Page 856 - 860, ©2015 IEEE. (2) "Asynchronous Design for Precision-Scaleable Energy-Efficient LDPC Decoder" by Jingwei Xu, Tiben Che, Ehsan Rohani and Gwan Choi, 2014. *Proceedings of the 2014 48th Asilomar Conference on Signals, Systems and Computers*, Page 136-140, ©2014 IEEE.

such as clock, but instead by handshaking between receiving and forwarding units via acknowledgement signals. The processing-flow model of asynchronous circuits is shown in Fig. 6.1. The REQ signal from previous stage informs the next stage that the new data is ready. Then the combinational logic circuit commences to compute. After the combinational computing, the control module will then generate 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.

Depending on the number of phases used in this process, the handshake is classified as 2-phase or 4-phase protocol. The comparison between these is illustrated in [64]. In this paper, 4-phase protocol is taken into consideration since very efficient optimization can be done at the logic and architectural levels when using four-phase protocols. Fig. 6.2 shows the how it works [65]:(1) the sender issues data and sets REQ high. (2) the receiver receives the data and sets ACK high. (3) the sender responds by taking REQ low. From this point data may not be valid. (4) the receiver acknowledges this by taking ACK low. At this point the sender may prepare for the next communication cycle.

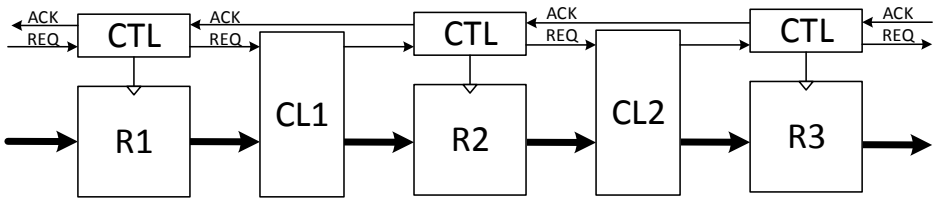


Figure 6.1: General model of asynchronous circuits

There are two ways to implement the hand shake protocol. One is the bundled data and the other is the dual rail logic. Bundled data means the data signal uses normal boolean levels to encode information, and the separate request and acknowledge wires are bundled

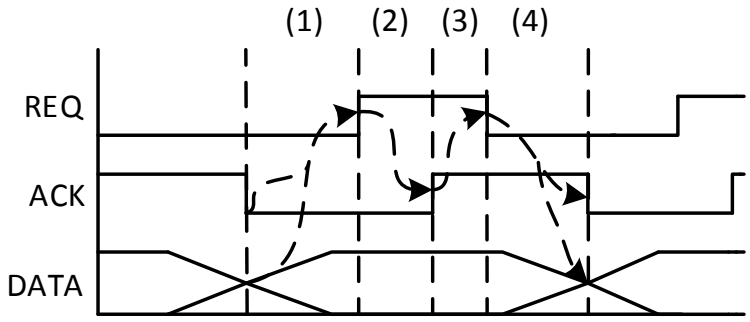


Figure 6.2: 4 phase protocol

with the data signals. Dual rail protocol encodes the request signal into the data signals, and uses two bits to present one information bit. There are three channel states of every single bit: invalid, valid with a 1 and valid with a 0, which are represented by 00, 01 and 10, respectively. All the bundled data protocols rely on delay matching, which means we need to fix the delay of request and acknowledge signals when we design it. In order to satisfy the timing of the hand shaking protocol, the delay should be longer than the critical path of the corresponding combination logic. In our design, we consider the situation where the completion time of combination logic has some fluctuation in every asynchronous cycle due to transient faults, and we can always choose the faster path to trigger the acknowledge signal to previous stage. Under such circumstance, the timing of protocol signals should not be fixed. Compared with bundled data, dual rails should be a better candidate for our design since the acknowledge signal is the data itself, which means the timing of acknowledge signal can change in real-time and respond adaptively. Thus, we adopt the 4-phase handshake protocol and dual-rail encoding schemes in our design.

In this section two asynchronous applications related to my dissertation works, dual-path asynchronous circuit and asynchronous design for precision-scaleable energy-efficient LDPC, are presented. Although both of them use LDPC decoder as a case study, they can be easily extended to polar code scenarios.

6.1 Accelerated Dual-Path Asynchronous Circuit

This section proposes a design approach that duplicates asynchronous circuit in a novel way. This approach not only exploits the merits of redundancy for fault tolerance [66] [67], but also considers the potential timing benefits rooted in the fact that there must be a faster path out of the two. By introducing an elaborate scheduling of the datapaths in the design, our approach significantly reduces overall system delay while substantiating the robustness attributable to redundancy and asynchronous circuits. In the dual-path design, the first valid signal received from the two replicated units is used to drive both subsequent paths; the valid output is then multiplexed out to both replicas of the next stage. This ensures that the combinational processing is forwarded only through the components that lie in the faster paths through the redundant network. Besides the tolerance to transient fault, the system is also designed to be immune from latch-ups through use of a power cycling scheme. The design approach is illustrated through a case study of LDPC decoder, which is often embedded in low-power error-prone mobile devices. An architecture reported in [68] is adopted and the proposed designs are instantiated. The designs are subjected to timing and fault-injection analysis to study relative merits of the proposed schemes. Compared with ordinary design, at the FR of 400/clock cycle, the proposed system reduces the delay overhead from 19.5% to 7.5%. Meanwhile the arbiter introduces only 2% area overhead in additional to the 2X duplication overhead. The results show the decoder can be significantly improved in terms of speed.

6.1.1 Dual-Path Circuit Design

Fig. 6.3 shows the general model of the dual-path asynchronous design. The system consists of duplicate asynchronous units in every asynchronous stage, and uses the same hand shake protocol as the regular asynchronous circuit. The acknowledgement signal (ACK) is generated by a specifically designed arbiter based on the performance of each

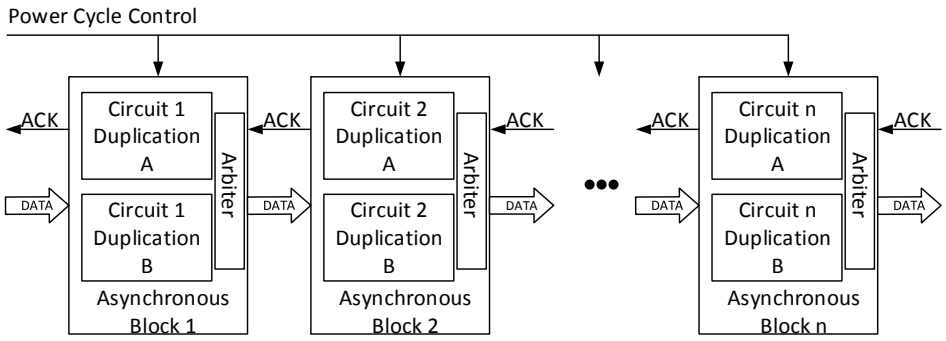


Figure 6.3: Overview of dual-path asynchronous circuit system

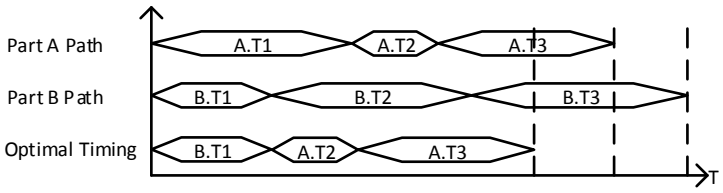


Figure 6.4: Timing of DMR

duplicate path. A power cycle control signal is also assigned to each asynchronous block.

As reported in [69], transient faults can induce output delay in QDI asynchronous circuits. Under hazardous environment, the accumulative delay of each asynchronous stage can become considerably large. Dual-path design has two candidates in every asynchronous stage, and the next stage is always stimulated by the output of the faster one of the two. This approach can dramatically decrease the delay of the whole system. An example with three asynchronous stages of this scheduling is shown in Fig. 6.4. Each of same stage in path A and path B has a variation of completion time. For every stage, the faster path is always chosen. Eventually, the delay of system is dramatically decreased since it is a combination of the faster paths from all the stages. In addition to the timing benefits, the dual-path design significantly enhances the inherent robustness that conventional asynchronous circuits provide. A carefully devised dual-path design can guarantee

that the system will always have working alternate path for continuous processing.

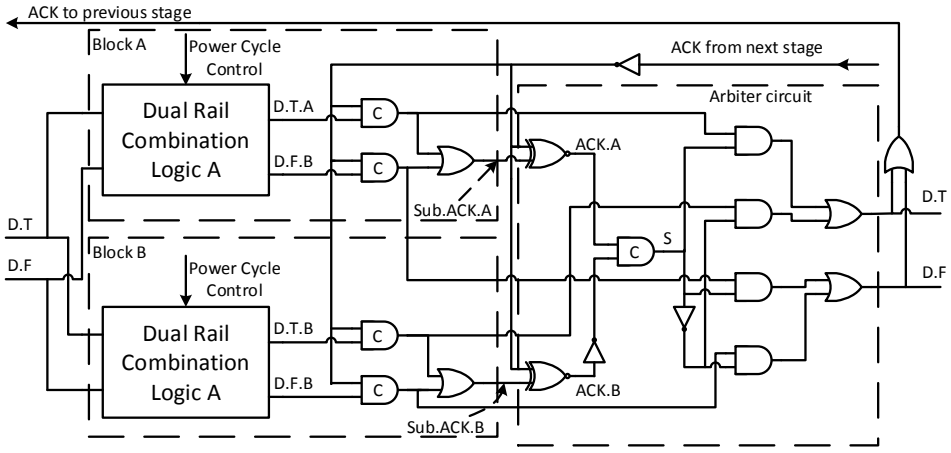


Figure 6.5: The details of one asynchronous block

Fig. 6.5 shows the details of one asynchronous block in Fig. 6.3. An example of 1-bit pipeline is given. It can be easily extended to n-bit situation. N bit pipeline can be implemented by using a number of 1 bit pipeline in parallel, and using a completion detector to synchronize acknowledge of each bit to generate the final acknowledge signal.

Fig. 6.5 shows that the asynchronous block consists of two regular asynchronous block and one arbiter unit. Block A and block B are exactly the same. All the combination logics used are dual rail, which guarantee that the circuit avoids losing its QDI property. Block A and Block B receive the input dual rail signals at the same time, then the combination logic starts working. However, an consequence of transient faults, these two might not finish at the same time. Sub.ACK.A and Sub.ACK.B are acknowledge signals generated by Block A and Block B. These two signals are the inputs of the arbiter unit. They are used to generate an arbitrating signal which contains the information of faster one of these two blocks. This arbitrating signal is fed into following multiplex network to choose the

corresponding dual rail path. The final acknowledge ACK is determined by these dual rail signals of the chosen path. In order to correctly choose the faster path, two factors need to be taken into consideration. (1) In 4 phase protocol, the detection of faster path does not only apply to the phase 2, where ACK changes from low to high, but also applies to the phase 4, where ACK changes from high to low. Thus, a way to unify the two situations is necessary since the behaviors of these two phases are in reverse. (2) Once the right path is chosen, it cannot change, no matter the other path will finish or not.

As shown in Fig. 6.2, in phase 2 and phase 4, the ACK signal from next stage ACK.N has value of 0 and 1, respectively. Thus, by using the signal with a XNOR gate we can unify the outcome of phase 2 and phase 4. ACK.A and ACK.B are two signals after unifying. Any one becomes high first means that path is faster no matter which phase they are in. Then, a C gate is used to generate the control signal S. C gate is a basic element in asynchronous circuit; its output will not change unless the two inputs have the same value. With this property, since ACK.B goes through an inverter, as long as neither of these two path are done, ACK.A and ACK.B are different. Thus, C holds the old value according to the previous choice. Once one path is done and the other is not, ACK.A and ACK.B become the same, which makes C gate output the corresponding value. After a while, even though the slower path is done, at that moment, since ACK.A and ACK.B go to different again, the output of C gate will not change. If ACK.A and ACK.B happen to go high at the same time, the C hold gate would still hold the previous value, which means it would choose previous path.

Consequently the system is always driven by the faster of the Block A and Block B in every asynchronous stage. This ensures that the overall system timing is spanned by only the faster modules in each stage. The performances of this design under application example are discussed later in Section 6.1.3.

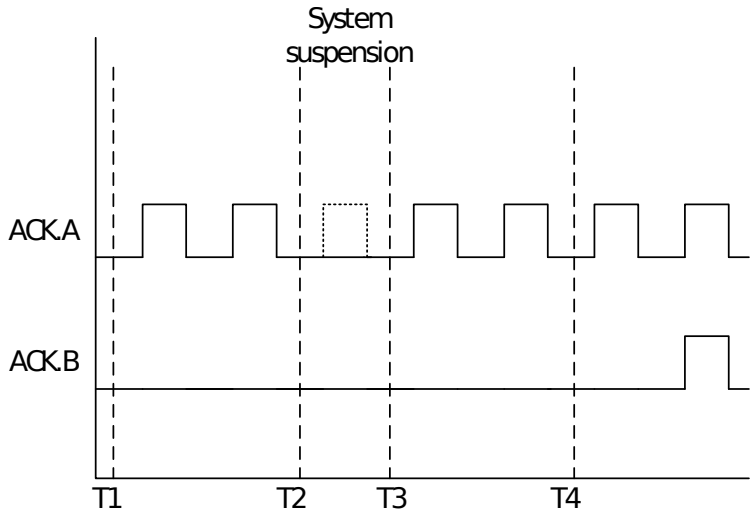


Figure 6.6: Timing of power cycle scheduling

6.1.2 Power Cycle Schedule

In our design, we only consider single event latch-up (SEL) situation in every asynchronous stage since the probability of coincidental latch-ups in the same asynchronous stage in both paths is low. Since it is dual-path design, system is assured to run smoothly regardless of whichever block exhibits latch-up in any asynchronous stage. However, this situation still can slow down the system since only one circuit path is alive in some asynchronous stages and thus there is no faster one to be chosen. To prevent the vulnerability from the latch-up faults, a specific power cycle scheme is also developed in our design. At first, a power cycling scheme based on the likelihood of having latch-ups for each path has been considered. However, its hardware overhead is too high since a significant number of registers are needed to hold statistical information about the winner of two paths in every asynchronous stage in each and every cycle. Therefore, we employee a simpler power cycle scheme alternatively power cycle after a certain number of clock cycles. If the path that triggers the power cycle incurs a latch-up, the faults will be fixed. During recycle pe-

riod there is another path can make sure the system continues to operate without faults. If one path has latch-up while recycling the other, then the whole system will not shut down but only hold for a recycling period since it is an asynchronous system, and the latched up path will be fixed in the next power cycle. The inherent property of asynchronous and dual-path design allows the system to benefit a lot from a simple power cycle scheme with negligible cost.

Fig. 6.6 shows a timing example of this scheme. It shows the acknowledge signal of an asynchronous stage triggering completion acknowledgement for the both paths. Suppose at time T1 path B exhibits a latch-up. Then, no acknowledge signal will be triggered from the path B. In this case, the system can still operate without an error but its speed will now be governed by the performance of path A. At time T2, a power cycle operation is issued to path A. During this process, no acknowledge signal will generated by path A until the power cycle completes at time T3. During this time, the entire system will suspend for a very small period. At time T4, another power cycle for the path B is issued, and the entire system continues to operate relying on the path A. After path B completes its power cycle, the latch-up is now eliminated and the system continues to operate with two active duplicated paths.

6.1.3 Case Study

A development of satellite communication (SatCom) involves coordinating performance, power and the reliability challenges from transients and permanent hazards from space radiation. Channel coding techniques are usually accompanied with the satellite communication receiver to increase the fidelity. We performed a design study of a low-density parity-check (LDPC) decoder [68], one of the widely used codec hardware components, to illustrate the effectiveness of the proposed asynchronous redundant designs. LDPC decoder has the property to decode received data through iterative methods to clean

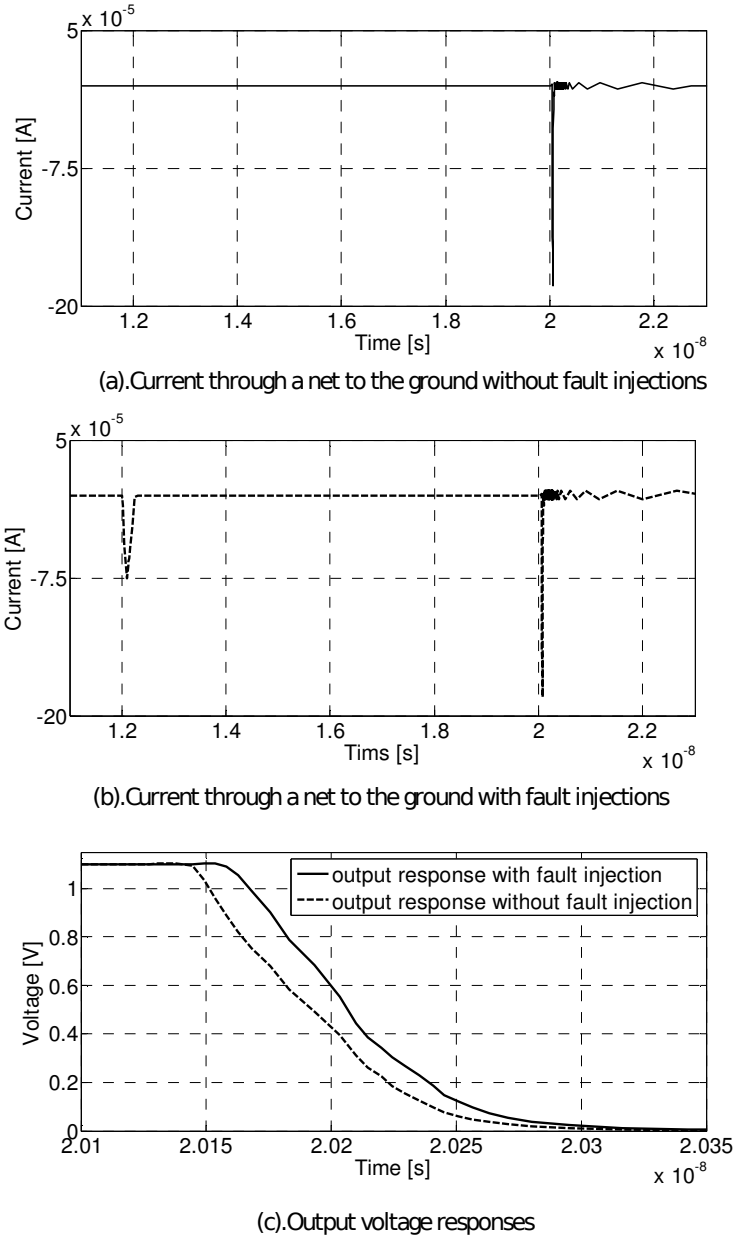


Figure 6.7: One transient fault delay example

the code words of errors. If the delay in each decoding step is reduced, more iterations can be done to achieve higher fidelity transmission. Especially for SatCom, effective decoding is crucial for the power-sensitive and error prone satellite receiver.

To evaluate the timing performance and overhead cost of the proposed methodology under transient faults, the RTL codes of basic units of LDPC decoder is first synthesized by *Synopsys Design Compiler*. Then the corresponding SPICE netlist is generated by *Synopsys Primetime* on *Nangate* 45nm CMOS technology library. The overheads of the control module are estimated by the synthesis results. An example of the delays is presented in the Fig. 6.7, where a net is chosen for transient fault injection. According to [70], transient faults, especially those generated by particle strikes, which is very common in SatCom, can be modeled by an electric charge injected to a circuit node. Thereby the transient fault is modeled as a negative impulse of current in our experiment. Fig. 6.7(a) shows the current response of a net without any transient fault. Fig. 6.7 (b) shows the current response of the same net in Fig. 6.7 with transient fault.The delay induced by transient faults is modeled as is introduced in [71], and a negative current pulse is added at around 12 ns. As a result, the output response has an extra delay caused by the transient fault as shown in Fig. 6.7 (c). By injecting the delay faults, the timing performance of the proposed system is analyzed.

To explore relationship between error injection and delay, the error model present in [71] is used in our experiments. In the reference, the radiation is modeled as a current source connected to nets. For a certain level of radiation, a quantity of electric charge will be injected to a random net during the transition time. Noticeably, the fault injection does not always necessarily result in a propagation delay, which depends on when the fault happens and the magnitude of charge. In order to obtain the effect of radiation on delays, SPICE simulation is running with additional current source injections. Fig. 6.8 shows the histogram how the injected current affects the completion timing. To ensure the randomness of these faults, each current source is added randomly to one node of the circuit at a random time slot. The experiment unit has the inherent baseline delay of 177.5 ps without any interference. As the figure shows, 200 experiments were done for the fault injection

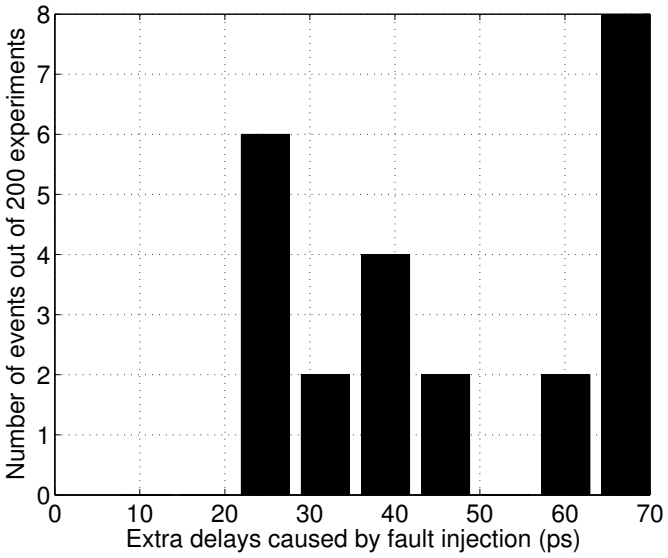


Figure 6.8: Histogram of delays caused by fault injections

over the circuits. Among that, 24 out of 200 experiments are delayed by the injected fault, and the extra delays range from 20 ps to 70 ps, which presents 11.3% to 39.5% increase respectively to the inherent delay.

To evaluate how the modeled error affects the whole system performance, fault rate (FR) is also introduced here as the number of injected faults over one clock cycle. According to the statistic information in fig. 6.8, a total delay can be estimated. Based on the introduced fault model, the timing analysis of the proposed scheme is given following.

To compare with the ordinary asynchronous circuits, the proposed design is implemented with the computation units of the LDPC decoder. All designs are tested over same FR to explore the timing performance. As Fig. 6.9 shows, the time to complete the operation is affected by the fault injection rate. In the figure, the delays for two schemes, the ordinary asynchronous circuit and the proposed dual-path asynchronous circuit, are presented. As the figure shows, a higher FR results in a larger delay, and traditional circuits has the maximum delay out of these two. Under the FR 400/clock cycle, the delay

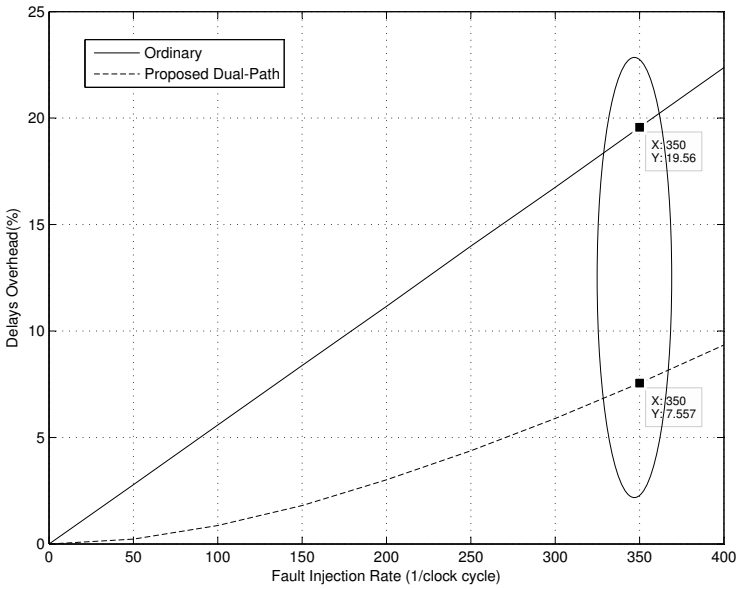


Figure 6.9: Delays under difference schemes with different FR

overhead of traditional circuit can reach 19.5%, and the proposed design can reduce the overhead to only 7.5%. We also can tell that the delay of proposed design increases slower than that of ordinary design as the increasing of FR. According to the result presented in Fig. 6.8, a transient fault can cause delay or not. For ordinary asynchronous circuit, as the increasing of FR, it gets a higher chance of being hit by a transient fault with delay. However, for proposed dual-path design, the chance is much lower since the probability of being hit by a transient fault with delay is squared own to the duplicated path. Even under a very small chance that both paths will be introduced delay, only the faster one will be chosen.

Check Node Unite (CNU) and Variable Node Unite (VNU) are two most costly and crucial combinational logic parts in a LDPC decoder. We use *Synopsys* to synthesis the circuits on *Nangate* 45nm CMOS technology library. Table 6.1 shows the area overhead of CNU,VNU, and proposed Arbiter.

As the table shows, proposed Arbiter incurs only 2.12% increase of area to original

Table 6.1: Hardware overhead (unit:μm^2)

CNU	VNU	proposed Arbiter
2369.793	1207.373	42.560

design. Since it is duplicate design, the 100% overhead of duplication cannot be avoid. However, considering the huge speed up and a higher robustness of entire system especially under harsh electromagnetic interference environments, the whole overhead is acceptable.

In this example, we show that the proposed asynchronous circuits design could be adopted into an application of LDPC decoder. With the acceptable overhead, our scheme could highly speed up the circuits in a hazard electromagnetic or radiation surrounding .

6.2 Asynchronous Design for Precision-Scaleable Energy-Efficient LDPC Decoder

Low-density parity-check codes [6] are powerful error-correcting codes that perform very close to the Shannon limit and are used in many communication standards such as IEEE 802.16e (WiMAX) [72], DVB-S2 and IEEE 802.11n (WiFi). 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 log-likelihood ratio (LLR) messages along the edges of a factor graph [73], 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.

The basic data flow chart of LDPC decoder is given in Fig. 6.10. 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 its simplified operation on

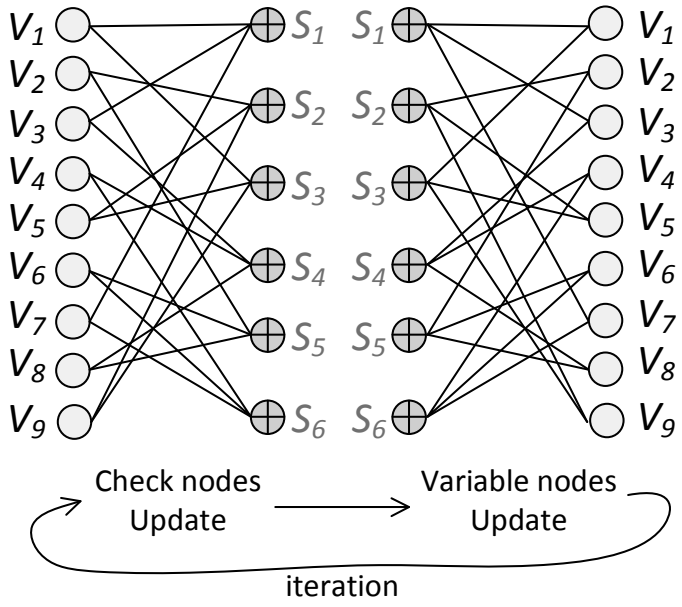


Figure 6.10: Generic LDPC decoding data flow graph

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 *i*th 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: $y_n + \sum_{m' \in M(n) \backslash m} R^{(i)}_{mn}$. $R^{(i)}_{mn}$ 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:

$$R^{(i)}_{mn} = (\prod_{n' \in N(m) \backslash m} sign(Q^i_{n'm})) \min_{n' \in N(m) \backslash m} (Q^i_{n'm}) \tag{6.1}$$

Power consumption is crucial to a decoder system. Thus, the dynamic voltage and

frequency scaling (DVFS) technique is introduced into the LDPC decoder design for the purpose of low power [74]. Although DVFS helps significantly reduce the power consumption, 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 powerful method to do low-power design [75].

In this work, a precision-scalable based LDPC decoder facilitated by the asynchronous circuits technique is proposed. First, a set of precision-adaptive calculation units are proposed to premise the flexibility of the decoder in precision. Furthermore, to get the optimal quantization scheme in terms of performance for each precision, the impacts of different fixed-point word sizes with various quantization schemes are presented. To target a given BER performance, the moderation of the necessary precision and circuits delays are evaluated by the SPICE simulations; and the corresponding power reduction is derived.

6.2.1 Proposed System

Fig. 6.11 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-scalable units which are designed in asynchronous circuits as various delays depending on the precision of the calculation in use. The control model is introduced here to determine necessary the pair of computation precision and voltage supply, for a given decoding fidelity, according to the strength of the interference. Different with the conventional synchronous precision-scalable design, event-driven asynchronous circuits allow not to dynamically scale the supply voltage along with voltage scaling. To accommodate the precision scalability, the modifications of CNU and VNU are presented as well.

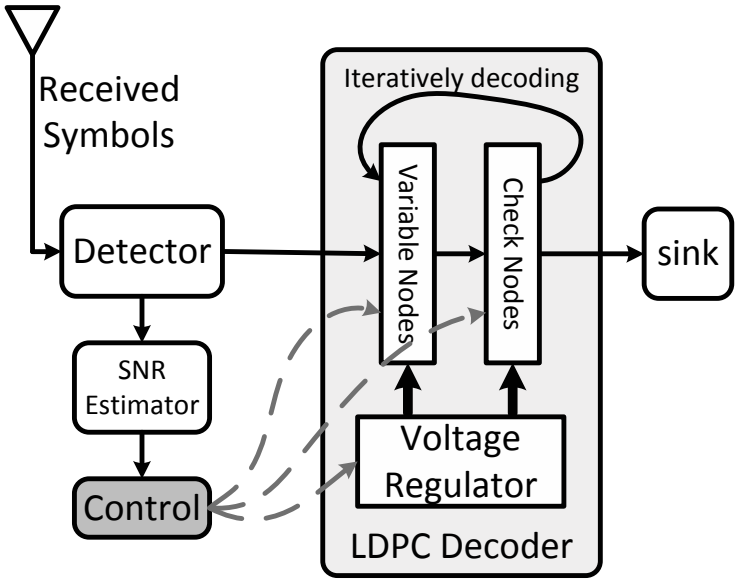


Figure 6.11: The overview system flow of proposed LDPC decoder

In this work, 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 clockless nature of the 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 section.

6.2.2 Design Details

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.

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, Fig. 6.12

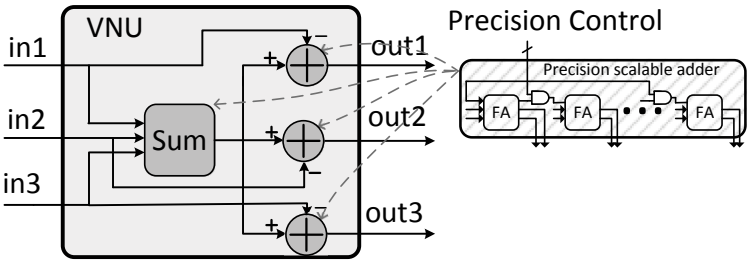


Figure 6.12: Asynchronous precision-salable VNU design.

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. 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 Fig. 6.12shows, 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.

Compared with the VNU, CNU is of greater complexity. Fig. 6.13 shows the architecture of the CNU. The critical path of the CNU is marked by the dashed line in the

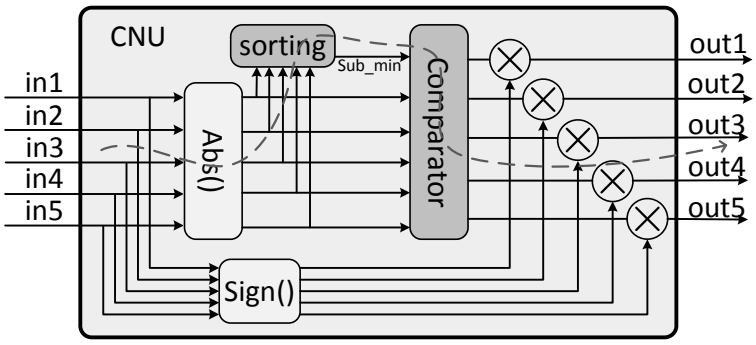


Figure 6.13: Asynchronous precision-salable CNU design

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.

To accommodate the scalability of the precision, the bit comparator units (BCUs) is introduced as the single units in the comparators. As Fig. 6.14 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

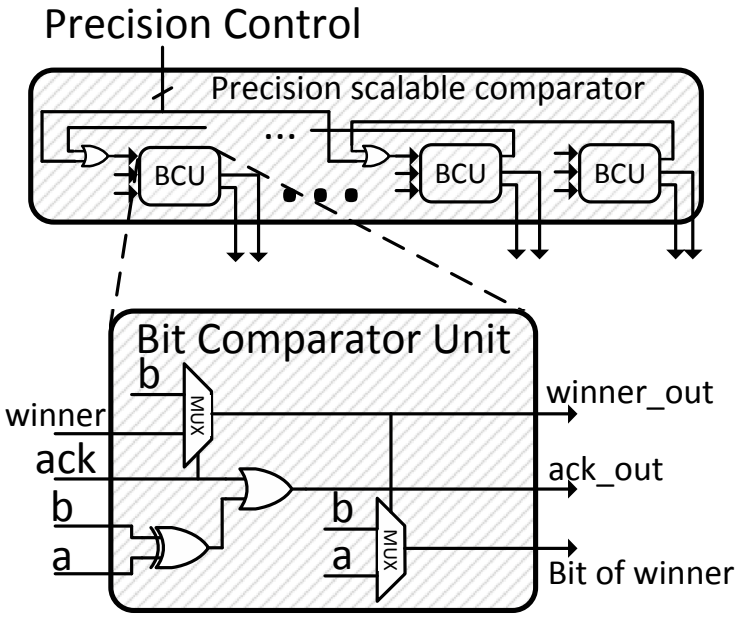


Figure 6.14: Proposed asynchronous comparator

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.

6.2.3 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.

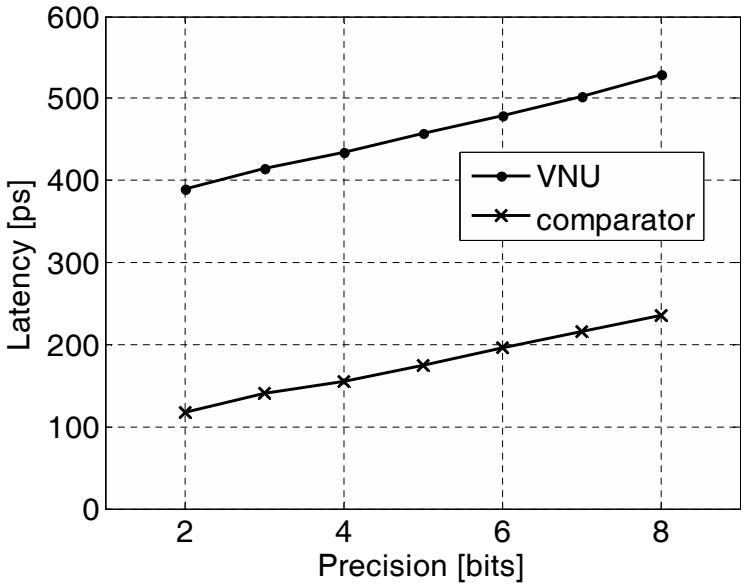


Figure 6.15: Units delays for different bits of precision

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 Fig. 6.15. 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.

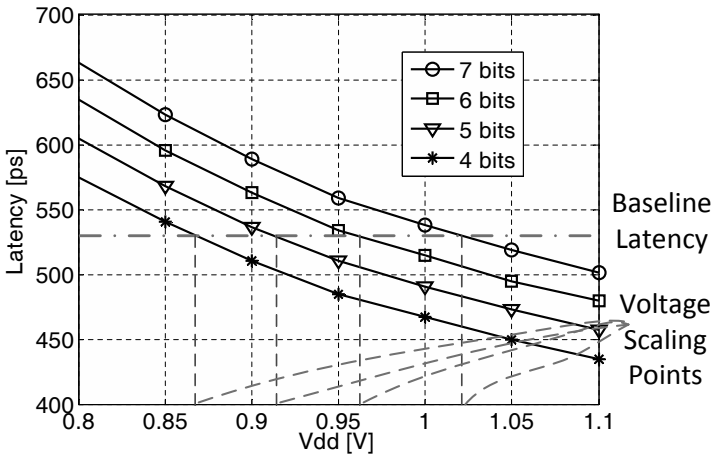


Figure 6.16: Voltage scaling to align processing latency

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. Fig. 6.16 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, $P \propto V_{dd}^2$. Fig. 6.17 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.

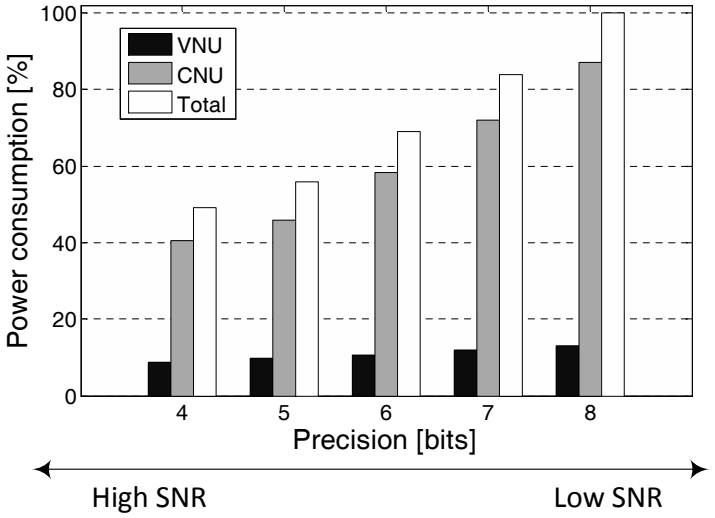


Figure 6.17: Normalized power reduction compared with fixed precision LDPC decoder

7. SUMMARY

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

TCSC decoder is proposed for high throughput demanding scenario. By absorbing the concept of constituent code into hardware design, the decoding latency is significantly reduced. Constituent code based SC decoder lacks the flexibility since it is highly code rate dependent. Our design overcomes the weakness by by exploiting the homogeneousness between the decoding processes of constituent polar codes and regular polar codes, which the presented design is compatible with any rate. The scheduling plan and the intendedly designed processing core are also described. Additionally, a specifically designed partial sum generator (PSG) which is compatible with constituent code based SC decoder is proposed as well. We derive the mathematical presentation with the partial sums set which is corresponding to each constituent code. From this, we concoct a shift-register based PSG from . Results show that, compared with the state-of-art decoder, our design can achieve at least 60% latency reduction for the codes with length n = 1024. This design is validated via ASIC design with Nangate FreePDK 45nm process. Besides, a polar code construction scheme that reduces constituent-code supplemented decoding latency is also presented. This construction schme is constituent codes oriented, which allows more our expected kinds of constituent are generated. We conducted the simulation of 1024 and 2048 length polar codes with multiple rates and analyzed the decoding latency for various length codes. The numerical results show that the proposed construction scheme generally is able to achieve at least around 20% latency deduction with an negligible gain loss with carefully selected optimization threshold.

Path-overlapped approach is proposed for a more hardware efficient LSC design. By

applying path-overlapping 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. The architecture of SC decoder is modified to support this new paradigm as well. Since modifications are made only on architecture and scheduling plan, no decoding performance gain loss or change is incurred. Three approaches, multi-decision list SC decoding, path-LLR-compute-ahead scheme and adaptive LSC decoding, to reduce the latency associated with the pipeline scheme are presented and evaluated as well. Simulation results show that with proposed design approach the hardware efficiency is increased significantly over the recently proposed LSC decoders.

XJ-BP is the result of algorithm and system level study of BP decoding, which significant reduced the computation complexity. The proposed algorithm facilitates belief propagation by utilizing the specific constituent codes that exist in the factor graph, which results in an express journey for belief information to propagate in each decoding iteration. In addition, a novel more efficient round-trip message passing scheduling method is proposed. The proposed method simplifies min-sum (MS) BP decoder by 40.6Along with the round-trip scheduling, the XJ-BP algorithm reduces the computational complexity of MS-BP decoding by 90.4%; this enables an energy-efficient hardware implementation of BP decoding in practice.

Asynchronous circuit is bottom level technique can be obtained in the decoder system to achieve a better performance. In this dissertation work, it is introduced into two applications. They are dual-path asynchronous circuit and asynchronous design for precision-scaleable energy-efficient LDPC. The first one yields a faster and more reliable system and the second one makes the entire system more energy efficient. Although both of them use LDPC decoder as a case study, they can be easily extended to polar code scenarios.

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[2] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[3] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, no. 1, pp. 68–79, 1960.

[4] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of The Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[5] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261–1271, 1996.

[6] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[7] E. Arikan, "Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

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

[9] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582, 2013.

[10] I. Tal and A. Vardy, "List decoding of polar codes," in *Proceedings of the 2011 IEEE International Symposium on Information Theory (ISIT)*, pp. 1–5, IEEE, 2011.

[11] K. Niu and K. Chen, "Crc-aided decoding of polar codes," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1668–1671, 2012.

[12] J. Guo, M. Qin, A. G. i Fabregas, and P. H. Siegel, "Enhanced belief propagation decoding of polar codes through concatenation," in *Proceedings of the 2014 IEEE International Symposium on Information Theory (ISIT)*, pp. 2987–2991, IEEE, 2014.

[13] K. Niu and K. Chen, "Stack decoding of polar codes," *Electronics Letters*, vol. 48, no. 12, pp. 695–697, 2012.

[14] K. Chen, K. Niu, and J. Lin, "Improved successive cancellation decoding of polar codes," *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3100–3107, 2013.

[15] R. Mori and T. Tanaka, "Performance of polar codes with the construction using density evolution," *IEEE Communications Letters*, vol. 13, no. 7, 2009.

[16] E. E. Gad, Y. Li, J. Kliewer, M. Langberg, A. A. Jiang, and J. Bruck, "Asymmetric error correction and flash-memory rewriting using polar codes," *IEEE Transactions on Information Theory*, vol. 62, no. 7, pp. 4024–4038, 2016.

[17] Y. Li, H. Alhussien, E. F. Haratsch, and A. A. Jiang, "A study of polar codes for mlc nand flash memories," in *Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC)*, pp. 608–612, IEEE, 2015.

[18] S.-N. Hong, D. Hui, and I. Marić, "Capacity-achieving rate-compatible polar codes," in *Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 41–45, IEEE, 2016.

[19] S. B. Korada, E. Sasoglu, and R. Urbanke, "Polar codes: characterization of exponent, bounds, and constructions," *IEEE Transactions on Information Theory*, vol. 56, no. 12, pp. 6253–6264, 2010.

[20] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *Proceedings of the 2009 IEEE International Symposium on Information Theory (ISIT)*, pp. 1488–1492, IEEE, 2009.

[21] Y. Wang and K. R. Narayanan, "Concatenations of polar codes with outer bch codes and convolutional codes," in *Proceedings of the 2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 813–819, IEEE, 2014.

[22] O. O. Koyluoglu and H. El Gamal, "Polar coding for secure transmission and key agreement," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 5, pp. 1472–1483, 2012.

[23] E. Arikan, "Systematic polar coding," *IEEE Communications Letters*, vol. 15, no. 8, pp. 860–862, 2011.

[24] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE Communications Letters*, vol. 16, no. 12, pp. 2044–2047, 2012.

[25] C. Xiong, J. Lin, and Z. Yan, "Symbol-decision successive cancellation list decoder for polar codes," *IEEE Transactions on Signal Processing*, vol. 64, no. 3, pp. 675–687, 2016.

[26] T. Che and G. Choi, "An encoding scheme with constituent codes optimization for polar code-aim to reduce the decoding latency," *arXiv preprint arXiv:1612.02545*, 2016.

[27] C. Leroux, I. Tal, A. Vardy, and W. J. Gross, "Hardware architectures for successive cancellation decoding of polar codes," in *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1665–1668, IEEE, 2011.

[28] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, "A semi-parallel successive-cancellation decoder for polar codes," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 289–299, 2013.

[29] C. Zhang and K. K. Parhi, "Low-latency sequential and overlapped architectures for successive cancellation polar decoder.," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2429–2441, 2013.

[30] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation polar decoder architectures using 2-bit decoding," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 4, pp. 1241–1254, 2014.

[31] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation list decoders for polar codes with multibit decision," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 10, pp. 2268–2280, 2015.

[32] A. Mishra, A. Raymond, L. Amaru, G. Sarkis, C. Leroux, P. Meinerzhagen, A. Burg, and W. Gross, "A successive cancellation decoder asic for a 1024-bit polar code in 180nm cmos," in *Proceedings of the 2012 IEEE Asian Solid State Circuits Conference (A-SSCC)*, pp. 205–208, IEEE, 2012.

[33] A. Balatsoukas-Stimming, A. J. Raymond, W. J. Gross, and A. Burg, "Hardware architecture for list successive cancellation decoding of polar codes," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 8, pp. 609–613, 2014.

[34] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "Llr-based successive cancellation list decoding of polar codes," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5165–5179, 2015.

[35] J. Lin, C. Xiong, and Z. Yan, "A high throughput list decoder architecture for polar codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2378–2391, 2016.

[36] T. Che, J. Xu, and G. Choi, "Overlapped list successive cancellation approach for hardware efficient polar code decoder," in *Proceedings of the 2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2463–2466, IEEE, 2016.

[37] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Communications Letters*, vol. 15, no. 12, pp. 1378–1380, 2011.

[38] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast polar decoders: algorithm and implementation," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 946–957, 2014.

[39] T. Che, J. Xu, and G. Choi, "Tc: throughput centric successive cancellation decoder hardware implementation for polar codes," in *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 991–995, IEEE, 2016.

[40] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast list decoders for polar codes," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 318–328, 2016.

[41] J. Lin and Z. Yan, "A hybrid partial sum computation unit architecture for list decoders of polar codes," in *Proceedings of the 2015 IEEE International Conference*

on Acoustics, Speech and Signal Processing (ICASSP), pp. 1076–1080, IEEE, 2015.

[42] G. Berhault, C. Leroux, C. Jego, and D. Dallet, "Partial sums generation architecture for successive cancellation decoding of polar codes," in *Proceedings of the 2013 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 407–412, IEEE, 2013.

[43] G. Berhault, C. Leroux, C. Jego, and D. Dallet, "Partial sums computation in polar codes decoding," in *Proceedings of the 2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 826–829, IEEE, 2015.

[44] Y. Fan and C.-y. Tsui, "An efficient partial-sum network architecture for semi-parallel polar codes decoder implementation," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3165–3179, 2014.

[45] T. Che and G. Choi, "An efficient partial sums generator for constituent code based successive cancellation decoding of polar codes," *arXiv preprint arXiv:1611.09452*, 2016.

[46] A. Pamuk, "An fpga implementation architecture for decoding of polar codes," in *Proceedings of the 2011 8th International Symposium on Wireless Communication Systems (ISWCS)*, pp. 437–441, IEEE, 2011.

[47] B. Yuan and K. K. Parhi, "Architecture optimizations for bp polar decoders," in *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2654–2658, IEEE, 2013.

[48] B. Yuan and K. K. Parhi, "Early stopping criteria for energy-efficient low-latency belief-propagation polar code decoders," *IEEE Transactions on Signal Processing*, vol. 62, no. 24, pp. 6496–6506, 2014.

[49] Y. Zhang, Q. Zhang, X. Pan, Z. Ye, and C. Gong, "A simplified belief propagation decoder for polar codes," in *Proceedings of the 2014 IEEE International Wireless*

Symposium (IWS), pp. 1–4, IEEE, 2014.

[50] J. Xu, T. Che, and G. Choi, "Xj-bp: express journey belief propagation decoding for polar codes," in *Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2015.

[51] J. Sparso, "Asynchronous circuit design-a tutorial," in *Chapters 8 Principles of Asynchronous Circuit Design-A Systems Perspective*, pp. 1–152, Kluwer Academic Publishers, 2006.

[52] E. Arikan, "A performance comparison of polar codes and reed-muller codes," *IEEE Communications Letters*, vol. 12, no. 6, 2008.

[53] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.

[54] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Transactions on Communications*, vol. 60, no. 11, pp. 3221–3227, 2012.

[55] H. Vangala, E. Viterbo, and Y. Hong, "A comparative study of polar code constructions for the awgn channel," *arXiv preprint arXiv:1501.02473*, 2015.

[56] P. Giard, A. Balatsoukas-Stimming, G. Sarkis, C. Thibeault, and W. J. Gross, "Fast low-complexity decoders for low-rate polar codes," *Journal of Signal Processing Systems*, pp. 1–11, 2016.

[57] C. Xiong, J. Lin, and Z. Yan, "Symbol-based successive cancellation list decoder for polar codes," in *Proceedings of the 2014 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 1–6, IEEE, 2014.

[58] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Unrolled polar decoders, part ii: fast list decoders," *arXiv preprint arXiv:1505.01466*, 2015.

[59] C. Zhang, Z. Wang, X. You, and B. Yuan, "Efficient adaptive list successive cancellation decoder for polar codes," in *Proceedings of the 2014 48th Asilomar Conference on Signals, Systems and Computers*, pp. 126–130, Nov 2014.

[60] T. K. Moon, "Error correction coding," *Mathematical Methods and Algorithms. Jhon Wiley and Son*, 2005.

[61] J. Cortadella, A. Kondratyev, L. Lavagno, and C. P. Sotiriou, "Desynchronization: Synthesis of asynchronous circuits from synchronous specifications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1904–1921, 2006.

[62] L. Cristofoli, A. Henglez, J. Benfica, L. Bolzani, F. Vargas, A. Atienza, and F. Silva, "On the comparison of synchronous versus asynchronous circuits under the scope of conducted power-supply noise," in *Proceedings of the 2010 Asia-Pacific Symposium on Electromagnetic Compatibility (APEMC)*, pp. 1047–1050, IEEE, 2010.

[63] N. Onizawa, T. Hanyu, and V. C. Gaudet, "High-throughput bit-serial ldpc decoder lsi based on multiple-valued asynchronous interleaving," *IEICE Transactions on Electronics*, vol. 92, no. 6, pp. 867–874, 2009.

[64] M. Renaudin, "Asynchronous circuits and systems: a promising design alternative," *Microelectronic Engineering*, vol. 54, no. 1, pp. 133–149, 2000.

[65] J. Spars and S. Furber, *Principles Asynchronous Circuit Design*. Springer, 2002.

[66] G. Rui, C. Wei, L. Fang, D. Kui, and W. Zhiying, "Modified triple modular redundancy structure based on asynchronous circuit technique," in *Proceedings of the 2006 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT).*, pp. 184–196, IEEE, 2006.

[67] R. Gong, W. Chen, F. Liu, K. Dai, and Z. Wang, "A new approach to single event effect tolerance based on asynchronous circuit technique," *Journal of Electronic Testing*, vol. 24, no. 1-3, pp. 57–65, 2008.

[68] K. Gunnam, G. Choi, W. Wang, and M. Yeary, "Multi-rate layered decoder architecture for block ldpc codes of the ieee 802.11 n wireless standard," in *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1645–1648, IEEE, 2007.

[69] Y. Monnet, M. Renaudin, and R. Leveugle, "Asynchronous circuits transient faults sensitivity evaluation," in *Proceedings of the 2005 42nd Design Automation Conference*, pp. 863–868, ACM, 2005.

[70] D.-I. F. W. Friesenbichler, *Effects and Mitigation of Transient Faults in Quasi Delay-Insensitive Logic*. PhD thesis, Technische Universität Wien, 2011.

[71] Q. Zhou and K. Mohanram, "Cost-effective radiation hardening technique for combinational logic," in *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pp. 100–106, IEEE Computer Society, 2004.

[72] IEEE 802.16 Working Group and others, "IEEE standard for local and metropolitan area networks, part 16: air interface for fixed broadband wireless access systems," *IEEE Std*, vol. 802, pp. 16–2004, 2004.

[73] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[74] W. Wang, E. Kim, K. K. Gunnam, and G. 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*, vol. 5, no. 3, pp. 303–312, 2009.

[75] T. Lin, K.-S. Chong, J. S. Chang, and B.-H. Gwee, “An ultra-low power asynchronous-logic in-situ self-adaptive vdd system for wireless sensor networks,” *IEEE Journal of Solid-State Circuits*, vol. 48, no. 2, pp. 573–586, 2013.