Efficient Algorithms for Stochastic Decoding of LDPC Codes

A Dissertation Presented
by
Kuo-Lun Huang
to
The Department of Electrical and Computer Engineering
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Northeastern University
Boston, Massachusetts

May 2016
ABSTRACT

Efficient Algorithms for Stochastic Decoding of LDPC Codes

by

Kuo-Lun Huang

Doctor of Philosophy in Electrical and Computer Engineering
Northeastern University, May 2016
Professor Masoud Salehi, Advisor

The expanding demand for high-speed communications has resulted in development of high-throughput error-correcting techniques required by emerging communication standards. Low-Density Parity-Check (LDPC) codes are a class of linear block codes that achieve near-capacity performance and have been selected as part of many digital communication standards. Stochastic computation has been proposed as a hardware efficient approach for decoding LDPC codes. Using stochastic computation, all messages in the iterative decoding process are represented by Bernoulli sequences. Computations on these sequences are performed bit-by-bit using simple logic operations. Furthermore, serial messages used in stochastic decoders help alleviate routing congestion in hardware implementation of decoder. These factors make stochastic decoding a low complexity alternative to implement LDPC decoders. In this dissertation, we analyze the characteristics of stochastic decoding and propose reduced-latency designs for stochastic LDPC decoders to achieve improved performance on various channel models.

We statistically analyze the behavior of stochastic LDPC decoding, including randomization in the stochastic streams and convergence of transition probabilities in iterative decoding process. We also present a space and time-efficient code bit determination method for stochastic LDPC decoders. In addition, we investigate and characterize the decoding errors of stochastic LDPC decoders and as an example, study the stochastic-decoding-specific trapping sets in the (1056,528) LDPC code used in the WiMAX standard. This study helps to develop methods to lower the error floor of stochastic decoding.
We propose a reduced-latency stochastic decoding algorithm for LDPC codes. The proposed algorithm, called *Conditional Stochastic Decoding* (CSD), improves error rate performance and reduces the decoding latency by more than 30% compared with the existing stochastic decoders. We also characterize the performance of CSD in various communication schemes. For example, we show the advantages of using the proposed CSD algorithm in the Automatic Repeat reQuest (ARQ) scheme when compared with other iterative decoding algorithms.

We extend our study of stochastic decoding to non-AWGN channel models including the Binary Symmetric Channel (BSC), the Z-channel, and the Rayleigh fading channel. We introduce scaling methods to improve the performance of stochastic decoding on these channel models. On the Rayleigh fading channel, the proposed method not only reduces the computational complexity of the stochastic decoding, but also provides 3-dB improvement in performance and lowers the error floor. Simplicity of hardware implementation, low latency, and good error rate performance of the proposed schemes make them suitable for emerging communication standards.
To the memory of my beloved grandma,
and to my family.
ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisors, Prof. Masoud Salehi and Prof. Vincent C. Gaudet, for their guidance as well as their continuous encouragement and support throughout this work. I am deeply thankful to my advisor at Northeastern University, Prof. Salehi. I benefit extensively from his deep and broad knowledge in the field of communications, coding, and information theory, and I am also impressed by his open and warm personality. I would like to thank his effort to enhance the quality of this work and countless help during my study here.

I am grateful to Prof. Gaudet at University of Waterloo for helpful discussions and inspiration for this doctoral research. His expertise and insights on circuit designs and stochastic decoding always enlighten me from different perspectives. I am also appreciate his time for meeting with me in Boston.

I greatly appreciate Prof. Milica Stojanovic for taking precious time to serve on my dissertation committee. Her valuable feedback and support have greatly improved the quality of this dissertation.

I also thank Prof. Stephen W. McKnight, Prof. Gilead Tadmor, Prof. Nicol McGruer, Prof. Jose Martinez Lorenzo, and Dr. Amir Farhat for their help and support during my teaching in the labs. I would also like to thank Ms. Faith Crisley, Ms. Joan Pratt, and Ms. Sharon Heath for their wonderful service to the department of Electrical and Computer Engineering and the Communications and Digital Signal Processing (CDSP) laboratory.

I thank my friends and colleagues in the CDSP lab, Osso Vahabzadeh, Yueqian Li, Koorosh Firouzbakht, Yashar Aval, Amir Tadayon, Parastoo Qarabaqi, and Bei Yan for the helps and productive discussions. I would also like to thank Bryan Lai and his family for the support and memorable moments.

I am sincerely thankful to my father, my sister, and my family for always being supportive and encouraging throughout my life. I would like to express my deeply appreciation and gratitude to my parents and grandparents for their care and love. Without them, I would not ever have completed this dissertation.
# TABLE OF CONTENTS

Abstract ........................................ ii  
List of Figures .................................. ix  
List of Tables ................................... xiii  
List of Notations ................................. xiv  
List of Acronyms ................................ xv  

1 Introduction ................................... 1  
1.1 Motivation ................................... 1  
1.2 Objectives ................................... 3  
1.3 Dissertation Outline ......................... 4  

2 Background .................................. 6  
2.1 LDPC Codes ................................ 6  
2.1.1 Iterative Decoding Algorithms .......... 9  
2.2 Challenges in Implementing LDPC Decoders .. 14  
2.3 Stochastic Computation ..................... 16  
2.3.1 Stochastic Representation ............... 16  
2.3.2 Stochastic Arithmetic Operations ....... 19  
2.3.3 Practical Applications ................... 21  
2.4 Stochastic Decoding ......................... 23  
2.4.1 Stochastic LDPC Decoding ............... 24  
2.4.2 The Latching Problem ................... 27  
2.4.3 Previous Work on Stochastic Decoding .... 28  

3 Statistical Analysis of Stochastic LDPC Decoding 32  
3.1 Introduction ................................ 32  
3.2 Randomization in Stochastic Stream ....... 33  
3.2.1 Scaling Method for Binary Symmetric Channel 34  
3.3 Transition Probabilities in Stochastic LDPC Decoding 41  
3.3.1 Markov Chain Model for Edge Memories 41  
3.4 Determination of Code Bits in Stochastic Decoding 47  
3.5 Conclusions ................................ 51
### 4 Error Characteristics of Stochastic LDPC Decoding 53
- 4.1 Introduction .................................................. 53
- 4.2 Structures Leading to Decoding Failure ..................... 54
- 4.3 Stochastic Decoding Specific Trapping Set ................. 57
- 4.4 Analysis of Stochastic Decoding Errors .................... 58
- 4.5 Conclusions .................................................. 59

### 5 A Reduced-Latency Stochastic Decoding Algorithm for LDPC Codes 61
- 5.1 Conditional Stochastic Decoding Algorithm ............... 61
- 5.2 Analysis and Architecture of the CSD Algorithm ........... 63
- 5.3 The CSD Algorithm on the AWGN Channel ............... 65
  - 5.3.1 Error Rate Performance .............................. 66
  - 5.3.2 Decoding Latency ................................. 66
- 5.4 Hybrid Stochastic Decoding ............................... 70
  - 5.4.1 The CSD Algorithm in Hybrid Decoding .......... 70
- 5.5 Conclusions .................................................. 73

### 6 Stochastic Decoding of LDPC Codes in ARQ Scheme 74
- 6.1 Hybrid ARQ Schemes ......................................... 74
- 6.2 Comparison with Other Iterative Decoding Algorithms .... 76
- 6.3 Performance of HARQ Using Stochastic Decoding ....... 79
  - 6.3.1 Constant Transmission Energy ....................... 79
  - 6.3.2 Adjustable Transmission Energy ..................... 83
- 6.4 Performance-Energy Tradeoffs for Stochastic Decoding in ARQ 88
- 6.5 Hybrid Stochastic Decoding in HARQ .................... 91
- 6.6 Conclusions .................................................. 93

### 7 Stochastic Decoding of LDPC Codes over Rayleigh Fading Channel 94
- 7.1 Fading Channel Models ..................................... 94
- 7.2 Iterative Decoding on Rayleigh Fading Channels ....... 95
- 7.3 Scaling Scheme for Stochastic Decoding ................. 97
- 7.4 Performance of Stochastic Decoding Algorithms on Fading Channels 98
  - 7.4.1 Slow Fading ........................................ 99
  - 7.4.2 Fast Fading with Ideal CSI ......................... 100
  - 7.4.3 Fast Fading with No CSI .......................... 102
  - 7.4.4 Fast Fading with Noisy CSI ......................... 104
- 7.5 Conclusions .................................................. 108

### 8 Conclusions 109
- 8.1 Contributions ................................................ 109
- 8.2 Future Work .................................................. 110
  - 8.2.1 Non-Binary Conditional Stochastic Decoding Algorithm ... 110
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A parity-check matrix $H$ and its corresponding Tanner graph of a regular LDPC code. A length-6 cycle in the Tanner graph is shown in bold.</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Some possibilities of stochastic streams for a probability of 0.75.</td>
<td>17</td>
</tr>
<tr>
<td>2.3</td>
<td>Structure of a stochastic number generator.</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>Conversion circuit from a stochastic stream to soft information.</td>
<td>18</td>
</tr>
<tr>
<td>2.5</td>
<td>AND gate used as a stochastic multiplier with (a) independent input streams, and (b) dependent input streams.</td>
<td>20</td>
</tr>
<tr>
<td>2.6</td>
<td>Stochastic divider by a JK flip-flop.</td>
<td>20</td>
</tr>
<tr>
<td>2.7</td>
<td>Approximated stochastic adder by an OR gate.</td>
<td>21</td>
</tr>
<tr>
<td>2.8</td>
<td>Scaled stochastic adder by a multiplexer.</td>
<td>22</td>
</tr>
<tr>
<td>2.9</td>
<td>The structures of (a) a variable node and (b) a check node in stochastic decoders.</td>
<td>25</td>
</tr>
<tr>
<td>2.10</td>
<td>An example of bit transmissions at (a) variable node $v_i$ and (b) check node $c_j$ in stochastic decoding.</td>
<td>26</td>
</tr>
<tr>
<td>3.1</td>
<td>A binary symmetric channel model with the crossover probability $\varepsilon$.</td>
<td>34</td>
</tr>
<tr>
<td>3.2</td>
<td>BER performance of the (1056,528) LDPC code decoded by the stochastic decoding with and without the scaling method on the BSC.</td>
<td>36</td>
</tr>
<tr>
<td>3.3</td>
<td>Average number of DCs for the (1056,528) LDPC code decoded by the stochastic decoding with and without the scaling method on the BSC.</td>
<td>36</td>
</tr>
<tr>
<td>3.4</td>
<td>Histograms of $P(x_i = 1</td>
<td>y_i \geq 0)$ and $P(x_i = 1</td>
</tr>
<tr>
<td>3.5</td>
<td>The equivalent models: (a) the BSC with the proposed scaling method; (b) the AWGN channel with the NDS and a hard-limiter.</td>
<td>39</td>
</tr>
<tr>
<td>3.6</td>
<td>(a) A Z-channel and (b) a binary erasure channel models.</td>
<td>39</td>
</tr>
<tr>
<td>3.7</td>
<td>BER performance of the (1056,528) LDPC code decoded by the stochastic decoding with and without the scaling method on the binary Z-channel.</td>
<td>40</td>
</tr>
<tr>
<td>3.8</td>
<td>A Markov chain model for a length-$L$ EM.</td>
<td>42</td>
</tr>
<tr>
<td>3.9</td>
<td>Transition probabilities when stochastic decoding a $(d_v, d_c) = (2, 3)$ LDPC code for 20 DCs using $\gamma = 1$ (conventional), 0.8, and 0.2 values.</td>
<td>45</td>
</tr>
<tr>
<td>3.10</td>
<td>BER performance of an irregular (1056,528) LDPC code decoded by stochastic decoder having up/down counters and decoder using direct codeword determination.</td>
<td>49</td>
</tr>
<tr>
<td>3.11</td>
<td>BER performance of a regular (2048,1723) LDPC code decoded by stochastic decoder having up/down counters and decoder using direct codeword determination.</td>
<td>49</td>
</tr>
</tbody>
</table>
3.12 Latency for an irregular (1056,528) LDPC code decoded by stochastic decoder having up/down counters and decoder using direct codeword determination. .......................... 50

3.13 Latency for a regular (2048,1723) LDPC code decoded by stochastic decoder having up/down counters and decoder using direct codeword determination. .......................... 50

4.1 An example of a size-3 stopping set \(\{v_1, v_2, v_3\}\). ........................................ 55

4.2 BER performance of the (1056,528) LDPC code decoded by the SPA, the MSA, and the stochastic decoding algorithm on the AWGN channel. .......................... 56

4.3 Graphical representation of the size-10 stochastic-decoding-specific trapping set. .......................................................... 58

5.1 Implementations of (a) a parity-check status detector for a degree-\(n\) variable node \(v_i\), where \([j_1, j_2, \ldots, j_n] \in C_i\), and (b) mechanism of channel bit determination at \(v_i\) in the CSD. ........................................ 65

5.2 BER performance and decoding latency of the irregular (1056,528) LDPC code decoded by the EM-based stochastic decoder and the CSD decoder. .......................... 67

5.3 BER performance and decoding latency of the regular (2048,1723) LDPC code decoded by the EM-based stochastic decoder and the CSD decoder. .......................... 67

5.4 FER performance of the irregular (1056,528) LDPC code decoded by the EM-based stochastic decoder and the CSD decoder. ........................................ 68

5.5 FER performance of the regular (2048,1723) LDPC code decoded by the EM-based stochastic decoder and the CSD decoder. ........................................ 68

5.6 Average number of DCs for the irregular (1056,528) LDPC code decoded by the EM-based stochastic decoder and the CSD decoder. ........................................ 69

5.7 Average number of DCs for the regular (2048,1723) LDPC code decoded by the EM-based stochastic decoder and the CSD decoder. ........................................ 69

5.8 BER performance of the (1056,528) LDPC code decoded by the two-stage hybrid stochastic decoding. ........................................ 72

5.9 Average number of DCs for the (1056,528) LDPC code decoded by the two-stage hybrid stochastic decoding. ........................................ 72

6.1 FER versus BER of LDPC codes decoded by the SPA, the MSA, and the stochastic decoding algorithm. ........................................ 77

6.2 Average number of error bits in an erroneous frame of LDPC codes decoded by the SPA, the MSA, and the stochastic decoding algorithm. ........................................ 78

6.3 Overall required number of DCs for the (1056,528) LDPC code decoded by stochastic decoding in the HARQ-2 system with total 600 DCs available. .......................... 81

6.4 BER performance of the (1056,528) LDPC code decoded by the EM-based stochastic decoding and the CSD in the HARQ-2 system with total 300 DCs. .......................... 82
6.5 Average number of DCs for the (1056,528) LDPC code decoded by the EM-based stochastic decoding and the CSD in the HARQ-2 system with total 300 DCs. .......................................................... 83
6.6 BER performance of the (1056,528) LDPC code decoded by the EM-based stochastic decoding in the HARQ-2 system using adjustable transmission energy. .................................................... 85
6.7 BER performance of the (1056,528) LDPC code decoded by the CSD in the HARQ-2 system using adjustable transmission energy. ................................................................. 85
6.8 Average number of DCs for the (1056,528) LDPC code decoded by stochastic decoding in the HARQ-2 system using adjustable transmission energy. ........................................ 86
6.9 Expected transmission energy for the (1056,528) LDPC code decoded by the EM-based stochastic decoder in the HARQ-2 system using adjustable transmission energy. ......................... 87
6.10 Expected transmission energy for the (1056,528) LDPC code decoded by the CSD in the HARQ-2 system using adjustable transmission energy. ........................................ 87
6.11 Performance of the (1056,528) LDPC code decoded by the EM-based stochastic decoding in the HARQ-2 system using different \( \alpha \) values and DC distributions. .................................................. 90
6.12 Performance of the (1056,528) LDPC code decoded by the CSD in the HARQ-2 system using different \( \alpha \) values and DC distributions. .................................................. 90
6.13 BER performance of the (1056,528) LDPC code decoded by the EM-based decoder and the CSD decoder on the fast Rayleigh fading channel with ideal CSI. .............................................. 101
6.14 Average number of DCs for the (1056,528) LDPC code decoded by the HARQ-2 system using the proposed hybrid stochastic decoding. ................................................................. 91

7.1 BER performance of the (1056,528) LDPC code decoded by the SPA and stochastic decoders on the AWGN channel and the slow Rayleigh fading channel with the CSI available at the receiver. .......................................................... 99
7.2 BER performance of the (1056,528) LDPC code decoded by the SPA and the stochastic decoders on the fast Rayleigh fading channel with ideal CSI. .................................................. 101
7.3 Average number of DCs for the (1056,528) LDPC code decoded by the EM-based decoder and the CSD decoder on the fast Rayleigh fading channel with ideal CSI. .............................................. 101
7.4 BER performance of the (1056,528) LDPC code decoded by the SPA and the stochastic decoders on the fast Rayleigh fading channel with no CSI. .................................................. 103
7.5 Average number of DCs for the (1056,528) LDPC code decoded by the EM-based decoder and the CSD decoder on the fast Rayleigh fading channel with no CSI. .................................................. 103
7.6 BER performance of the (1056,528) LDPC code decoded by the SPA and the EM-based stochastic decoders on the fast Rayleigh fading channel with noisy CSI. .................................................. 106
7.7 BER performance of the (1056,528) LDPC code decoded by the SPA and the CSD decoders on the fast Rayleigh fading channel with noisy CSI. . . . 106
7.8 BER performance of the (1056,528) LDPC code decoded by the EM-based and the CSD decoders on the fast Rayleigh fading channel with noisy CSI. . 107
7.9 Average number of DCs for the (1056,528) LDPC code decoded by the EM-based and the CSD decoders on the fast Rayleigh fading channel. . . . 107
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>LDPC codes from the WiMAX standard and the 10GBASE-T standard.</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>A brief timeline for the development of stochastic computation.</td>
<td>23</td>
</tr>
<tr>
<td>2.3</td>
<td>Comparison on hardware implementations of LDPC decoders adopting different decoding algorithms.</td>
<td>31</td>
</tr>
<tr>
<td>3.1</td>
<td>The transition matrices of a length-4 EM in stochastic decoding a ((d_v, d_c) = (2, 3)) LDPC code at the (t)-th DCs.</td>
<td>44</td>
</tr>
<tr>
<td>3.2</td>
<td>Percentage of outputted stochastic bit values during 301-310 DCs at SNRs.</td>
<td>48</td>
</tr>
<tr>
<td>3.3</td>
<td>Percentage of outputted stochastic bit values during 601-610 DCs at SNRs.</td>
<td>48</td>
</tr>
<tr>
<td>4.1</td>
<td>44 multiplicities of the size-10 stochastic-decoding-specific trapping sets in the irregular (1056,528) LDPC code.</td>
<td>59</td>
</tr>
<tr>
<td>6.1</td>
<td>BER of the (1056,528) LDPC code decoded by stochastic decoding in the HARQ-2 system with total 600 DCs available.</td>
<td>80</td>
</tr>
<tr>
<td>6.2</td>
<td>Performance tradeoff for stochastic decoding in the HARQ-2 system.</td>
<td>89</td>
</tr>
</tbody>
</table>
**LIST OF NOTATIONS**

\( x \)  
A length-\( n \) codeword \([x_1 \ x_2 \cdots x_n]\).

\( m \)  
Modulated signal \([m_1 \ m_2 \cdots m_n]\).

\( y \)  
Received channel signal \([y_1 \ y_2 \cdots y_n]\).

\( C \)  
Code space of codewords.

\( H \)  
Parity-check matrix of linear block code.

\((d_v, d_c)\)  
Degrees of variable node and check node.

\( v_i \)  
The \( i \)-th variable node, for \( 1 \leq i \leq n \), of a rate-\( k/n \) code.

\( c_j \)  
The \( j \)-th check node, for \( 1 \leq j \leq (n-k) \), of a rate-\( k/n \) code.

\( C_i \)  
The index set of check nodes connected to \( v_i \).

\( V_j \)  
The index set of variable nodes connected to \( c_j \).

\( P_a \)  
Probability of the bit is equal to 1 in the stream \( a \).

\( B_{i,t} \)  
Bit received from the channel at \( v_i \) in the \( t \)-th decoding round.

\( B_{i\rightarrow j,t} \)  
Bit message from \( v_i \) to \( c_j \) at the \( t \)-th decoding round.

\( L_i \)  
Log-likelihood ratio received from the channel at \( v_i \).

\( L_{i\rightarrow j,t} \)  
Log-likelihood ratio from \( v_i \) to \( c_j \) at the \( t \)-th iteration.

\( P_i \)  
Probability \( P(x_i = 1|y_i) \) received from the channel at \( v_i \).

\( P_{i\rightarrow j,t} \)  
Probability \( P(x_i = 1|y_i) \) from \( v_i \) to \( c_j \) at the \( t \)-th decoding round.

\( P_{m,t} \)  
Probability stored in the moving average filter at the \( t \)-th decoding cycle.

\( \beta \)  
A variable used in scaling for stochastic decoding, \( \forall \beta \in \mathbb{R} : \beta > 0 \).
# LIST OF ACRONYMS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARQ</td>
<td>Automatic Repeat reQuest</td>
</tr>
<tr>
<td>AWGN</td>
<td>Additive White Gaussian Noise</td>
</tr>
<tr>
<td>BEC</td>
<td>Binary Erasure Channel</td>
</tr>
<tr>
<td>BER</td>
<td>Bit Error Rate</td>
</tr>
<tr>
<td>BFA</td>
<td>Bit-Flipping Algorithm</td>
</tr>
<tr>
<td>BPSK</td>
<td>Binary Phase-Shift Keying</td>
</tr>
<tr>
<td>BSC</td>
<td>Binary Symmetric Channel</td>
</tr>
<tr>
<td>CC</td>
<td>Chase Combining</td>
</tr>
<tr>
<td>CSD</td>
<td>Conditional Stochastic Decoding</td>
</tr>
<tr>
<td>CSI</td>
<td>Channel State Information</td>
</tr>
<tr>
<td>CSIS</td>
<td>Channel State Information Scaling</td>
</tr>
<tr>
<td>DC</td>
<td>Decoding Cycle</td>
</tr>
<tr>
<td>EM</td>
<td>Edge Memory</td>
</tr>
<tr>
<td>FEC</td>
<td>Forward Error Correction</td>
</tr>
<tr>
<td>FER</td>
<td>Frame Error Rate</td>
</tr>
<tr>
<td>HARQ</td>
<td>Hybrid Automatic Repeat reQuest</td>
</tr>
<tr>
<td>LDPC</td>
<td>Low-Density Parity-Check</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>LLR</td>
<td>Log-Likelihood Ratio</td>
</tr>
<tr>
<td>LR</td>
<td>Likelihood Ratio</td>
</tr>
<tr>
<td>MSA</td>
<td>Min-Sum Algorithm</td>
</tr>
<tr>
<td>MTFM</td>
<td>Majority-based Tracking Forecast Memory</td>
</tr>
<tr>
<td>NDS</td>
<td>Noise-Dependent Scaling</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>SPA</td>
<td>Sum-Product Algorithm</td>
</tr>
<tr>
<td>TFM</td>
<td>Tracking Forecast Memory</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

Digital communication systems are everywhere in our daily life. We all access video, audio or transmit data via wireless or wired connections on a daily basis. No one is interested in using a communication network that cannot provide reliable transmission, or a storage device that often messes up saved data. For decades, communication engineers have been looking for error correction coding techniques that can offer good protection against errors and at the same time achieve high data rates, preferably close to the channel capacity [1].

Error-correcting codes are widely used to ensure reliable communications in a variety of applications [2]. The basic idea of error correction coding is to add redundancy to the information bits in the form of parity-check bits before their transmissions. The receiver then can use the added redundancy to recover the original information from the received noisy channel outputs. The original data are encoded in a well-structured way that permits error correction through a proper decoding algorithm.

Among different classes of error-correcting codes, Low-Density Parity-Check (LDPC) codes [3] are a class of linear block codes that can provide near-capacity performance [4]. Because of their outstanding performance, LDPC codes have been a major focus of communications research and have been adopted in many emerging digital communication standards, including WiMAX (IEEE 802.16e) [5], WiFi (IEEE 802.11n) [6], WPAN (IEEE 802.15.3c) [7], Digital Video Broadcast (DVB-S2) satellite communications [8], G.hn (ITU-T G.9960) wired home networking [9], and 10GBASE-T (IEEE 802.3an) Ethernet networking [10].
1.1. Motivation

LDPC codes are often decoded iteratively using some form of the belief propagation principle [11], known as the Sum-Product Algorithm (SPA) [12]. Due to the high computational complexity of the SPA, its less-complex approximation, the Min-Sum Algorithm (MSA) [13], is often considered for the LDPC decoding. Both of these decoding algorithms compute *a posteriori* probabilities and transmit these probabilities iteratively through the interconnection network in the factor graph of the code. The performance of LDPC codes and the error-correcting ability of these iterative decoding algorithms improve as the codeword length gets longer. However, when implementing an LDPC decoder, a lot of chip area is required for wiring as codeword length increases, especially in the parallel implementation of the decoder which is needed to achieve multi-Gb/s speeds required by emerging standards. When $W$-bit quantization is used to represent the probabilities, each transmission in a fully parallel implemented decoder requires $W$ wires to realize. For instance, the factor graph of the length-2048 LDPC codes adopted in the 10GBASE-T standard contains 12288 edges, which requires $12288 \times 2 \times W$ wires to transmit the $W$-bit quantized messages back and forth at the same time.

If we utilize fewer bits to quantize the transmitted messages, the performance of LDPC decoders will be degraded, particularly at the low error rate region where LDPC codes are usually supposed to operate [14]. In order to address this interconnection network problem, stochastic computation was proposed as an alternative to the SPA in 2003 [15]. Using stochastic computation, the probabilities which are transmitted during the decoding process are transformed into streams of stochastic bits in the form of Bernoulli sequences, which can be passed over a single wire. This approach greatly reduces the number of wires and alleviates the routing congestion problem in implementations of LDPC decoders. On the other hand, representing messages in a bit-wise fashion makes all the operations on these sequences to be performed bit-by-bit, which can often be implemented using simple logic arrangements [16]. Due to the simplicity and hardware efficiency of stochastic computation, this approach has been used to implement several practical LDPC decoders [17, 18, 19, 20].

However, most of studies on stochastic decoding of LDPC codes have been focused on improving their performance by adopting different circuit designs or hardware mechanisms. Research on properties of stochastic decoding are quite limited. It should also be
1.2. Objectives

In this dissertation, we investigate several properties of stochastic decoding: (a) the required randomization in stochastic streams, (b) the convergence of transition probabilities in stochastic LDPC decoder, and (c) the error characteristics of LDPC code when decoded by stochastic decoder. For (a), we study the stochastic decoding on the binary symmetric channel model. Since messages received from this channel model are binary bits instead of soft information, this type of messages will result in poor error rate performance for stochastic decoders due to lack of randomness in stochastic streams. Therefore, technique that can introduce switching activity in stochastic streams is desired.

Since stochastic decoding is nondeterministic process, an insight into the convergence of transition probabilities in stochastic LDPC decoder is helpful in estimating the decoding delay. We present a probabilistic model to help us understand the behavior of stochastic decoding. Furthermore, we investigate the error characteristics of stochastic LDPC decoding and analyze the factors leading to decoding failure in stochastic decoders. This study gives guidelines to design a code graph having lower error floor in stochastic decoding.

To increase the throughput of stochastic decoding, we develop a reduced-latency stochastic decoding algorithm for LDPC codes. The proposed algorithm is introduced to improve error rate performance and reduce the decoding latency in the meantime. In addition to the advantages in hardware implementations, exploring the possibility of stochastic decoding for different communication environments is also essential. We characterize and demonstrate the benefits of using stochastic decoding in Automatic Repeat reQuest (ARQ) schemes and furthermore present techniques to improve the decoding performance.
of stochastic decoding algorithms for channel models other than the Additive White Gaussian Noise (AWGN) channel, e.g., fading channels.

1.3 Dissertation Outline

The rest of the dissertation is organized as follows. Chapter 2 provides the background materials. We review the LDPC codes and the iterative decoding algorithms for LDPC codes, including the Bit-Flipping Algorithm (BFA), the SPA, etc. In addition, we describe some strategies and challenges of hardware implementation of LDPC decoders. Later, we show the benefits of using stochastic computation to implement LDPC decoders. Then, we describe stochastic decoding of LDPC codes and the existing methods for its performance improvement.

Chapter 3 includes the statistical analysis of stochastic LDPC decoding. We propose a scaling method for stochastic decoders to achieve improved error rate performance on the binary symmetric channel. Furthermore, we study the convergence of transition probabilities in stochastic LDPC decoding process by a Markov chain model which utilizes the memory mechanism in stochastic decoders. We also present a space and time-efficient code bit determination method for stochastic LDPC decoders in this chapter. Chapter 3 is in part based on the material in our papers [21], [22], and [23].

Chapter 4 focuses on the error characteristics of stochastic LDPC decoding. We investigate the error patterns of stochastic decoding and report stochastic-decoding-specific trapping sets for a practical LDPC code. This observation provides insight into the trapping errors in stochastic decoding and results in guidelines for designing code graphs to achieve lower error floors. Chapter 4 is in part based on the material in our paper [24].

Chapter 5 proposes a reduced-latency stochastic LDPC decoding algorithm. We analyze the algorithm and provide a hardware architecture for it. Error rate performance and decoding latency of the proposed algorithm are also characterized in this chapter. In addition, we introduce a hybrid stochastic decoding scheme using the proposed algorithm to provide improved error rates. Chapter 5 is in part based on the material in our paper [25].

Chapter 6 demonstrates the advantages of using stochastic decoding in the ARQ scheme. We show the improved decoding performance of stochastic LDPC decoding in
1.3. Dissertation Outline

the hybrid ARQ system. We also present the tradeoffs between decoding performance and transmission energy cost. Chapter 6 is in part based on the material in our paper [26].

Chapter 7 shows the application of stochastic decoding on Rayleigh fading channels. We propose a scaling method targeting fading channels for improved performance of stochastic decoders on various fading scenarios. Moreover, we characterize the performance of stochastic LDPC decoding on the fading channels. Compared to the existing LDPC decoding algorithms, we demonstrate that the proposed stochastic decoding algorithm from Chapter 5 has higher throughput on fading channels. Chapter 7 is in part based on the material in our paper [27].

Finally, Chapter 8 concludes the dissertation and provides potential venues for future work.
Chapter 2

Background

In this chapter, we review LDPC codes and the iterative decoding algorithms used to decode LDPC codes. The challenges of implementing LDPC decoders are also discussed. We also explain how stochastic decoding can be used as an alternative to standard iterative methods in order to achieve higher hardware-efficiency. Then, we describe the existing methods used to enhance the performance of stochastic LDPC decoders and provide a comparison with other iterative decoding algorithms.

2.1 LDPC Codes

LDPC codes are a class of linear block codes, which were first introduced by Gallager [3] in 1962 and then more recently rediscovered by MacKay [28]. The performance of LDPC code is close to the Shannon capacity limit [4]. The Shannon capacity limit is the upper bound on the rate of codes that can be used to transmit information reliably over a noisy channel.

The structure of a binary \((n, k)\) LDPC code is defined by a set of \((n - k)\) parity-check equations, where \(k\) is the length of an uncoded information message and \(n\) is the length of the codeword. We denote a length-\(n\) codeword \([x_1 x_2 \cdots x_n]\) by \(x\). For a rate \(R_c = k/n\) code, each codeword satisfies the constraint

\[
x H^T = 0
\]  

(2.1)

where \(H\) is an \((n - k) \times n\) matrix, called the parity-check matrix. \(H\) determines the set of parity-check equations that define the code. The key characteristic of LDPC codes is that the number of elements in each parity-check equation is small, meaning that the \(H\) matrix
2.1. LDPC Codes

A parity-check matrix $H$ and its corresponding Tanner graph of a regular LDPC code. A length-6 cycle in the Tanner graph is shown in bold.

LDPC codes can be classified into two groups: regular and irregular codes. A regular LDPC code is a linear block code whose parity-check matrix $H$ contains exactly $w_c$ non-zero elements in each column, and exactly $w_r = w_c \cdot n/(n-k)$ non-zero elements in each row, where $w_c \ll (n-k)$. If the number of non-zero elements in each column or row is not constant, the code is an irregular LDPC code.

In [29], Tanner introduced an effective graphical representation for LDPC codes: a bipartite graph now known as a Tanner graph. A Tanner graph is a graphical representation of $H$ that facilitates iterative decoding of the code. The Tanner graph consists of two distinct groups of nodes: variable nodes and check nodes. There are edges connecting the nodes in a Tanner graph, as per parity-check equations of the code determined by the $H$ matrix. Thus, the number of check nodes is equal to the number of rows in $H$. Each check node is connected by edges to the corresponding variable nodes included in the parity-check equation of the check node. The number of variable nodes is equal to the code length. The $(n-k)$ rows of $H$ specify the connections to the $(n-k)$ check nodes, and the $n$ columns of $H$ determine the connections to the $n$ variable nodes. An example of a Tanner graph is shown in Figure 2.1. An edge connects the variable node $v_i$ and the check node $c_j$ if the $i$-th codeword bit is included in the $j$-th parity-check equation, for $1 \leq i \leq n, 1 \leq j \leq (n-k)$. This, in turn, means that $H(j, i) = 1$. The weights of each column and row, $(w_c, w_r)$, are the degrees of each variable node and check node, $(d_v, d_c)$, respectively.
2.1. LDPC Codes

Table 2.1: LDPC codes from the WiMAX standard and the 10GBASE-T standard.

<table>
<thead>
<tr>
<th>$(n, k)$</th>
<th>$R_c$</th>
<th>Node degree</th>
<th>Ratio</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1056, 528)</td>
<td>0.5</td>
<td>$d_v = (2, 3, 6)$</td>
<td>${11/24, 8/24, 5/24}$</td>
<td>WiMAX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$d_c = (6, 7)$</td>
<td>${2/3, 1/3}$</td>
<td></td>
</tr>
<tr>
<td>(2048, 1723)</td>
<td>0.8413</td>
<td>$d_v = 6$</td>
<td>1</td>
<td>10GBASE-T</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$d_c = 32$</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

A cycle in a Tanner graph is a sequence of edges that starts and ends at the same node, and goes through other nodes no more than once. The girth of a Tanner graph is the minimum cycle length of the graph. The shortest possible cycle in a bipartite graph is a length-4 cycle. Short cycles degrade the performance of LDPC decoders because they affect the independence of the messages exchanged in the iterative decoding \[28, 30\]. Hence, LDPC codes with large girth are more desirable.

In this dissertation, we mainly consider two practical LDPC codes: an irregular (1056,528) code adopted by the WiMAX standard \[5\] and a regular (2048,1723) code from the 10GBASE-T standard \[10\]. The WiMAX code is a Quasi-Cyclic (QC) LDPC code \[31\], where its parity-check matrix $H$ is defined by a set of 44-by-44 circulant matrices. LPDC codes with QC structure have advantages over other types of LDPC codes in hardware implementation of encoding and decoding. Since the parity-check matrix $H$ of a QC-LDPC code is constructed by numbers of the same size circulant matrices, a QC-LDPC can be efficiently and systematically encoded with shift-registers and the QC-structure also simplifies wire routing in the decoder \[32\]. The code from the 10GBASE-T standard is a Reed-Solomon based LDPC code \[33\]. Table 2.1 summarizes the characteristics of these two practical LDPC codes. For simulations using these LDPC codes, we randomly generate information bits and encode them using the corresponding $H$ matrix, i.e., we do not rely on the transmission of the all-zero codeword. Encoded data is modulated and then sent over the channel.
2.1. LDPC Codes

2.1.1 Iterative Decoding Algorithms

LDPC decoding is usually done iteratively using a form of belief propagation principle [11], known as the Sum-Product Algorithm [12], or its less-complex approximation, the Min-Sum Algorithm [13]. These algorithms utilize the structure of the code’s factor graph to pass messages between variable and check nodes in discrete time steps. An iteration of these decoding algorithms includes the exchange of messages for each graphical edge. For initialization, each variable node directly passes the channel message to its connected check nodes. Then, check node processes these incoming messages and sends a message back to each of its neighboring variable nodes. Upon receiving the messages from the check nodes, each variable node uses these messages together with its own channel received value to determine the code bit and produces a new message that is sent back to its connecting check nodes. This process continues until a valid codeword is obtained or a preset number of iterations is exceeded.

An essential part of iterative decoding algorithms is passing the messages between the nodes in the factor graph. Different algorithms pass different types of messages. Here, we describe some most-used iterative decoding algorithms for LDPC codes.

2.1.1.1 The Bit-Flipping Algorithm

The bit-flipping algorithm is an iterative decoding algorithm using the hard information where it exchanges the 1-bit message on each edge per iteration. At first, each variable node receives a binary digit, which is the hard decision value of the channel output corresponding to the node. A 1-bit message is sent out from a variable node to its connected check nodes. Each check node sends a message back to its connected variable nodes to declare whether its parity-check is satisfied or not. The messages keep bouncing between the nodes until all check nodes are satisfied or the maximum number of iterations is reached. The BFA operates as follows:

Step 1. Initialization. Each variable node is assigned a bit equal to the hard decision of the channel output, and passes the bit to its connected check nodes.
2.1. LDPC Codes

Step 2. Check Node Update. Each check node uses all its incoming bits to calculate the even parity-check. If all parity-check equations at all check nodes are satisfied, the decoding process terminates. Otherwise, each check node sends a 1-bit message back to the connected variable nodes to indicate the condition of the parity-check.

Step 3. Variable Node Update. At each variable node, if the number of unsatisfied connected check nodes is greater than a preset threshold value, the variable node flips its current bit value. Otherwise, the bit value is retained. Then, each variable node sends its 1-bit message to the connected check nodes, and the algorithm returns to Step 2 until a preset maximum number of iterations is reached.

The optimal threshold value used in Step 3 of the BFA depends on the channel conditions. A convenient and simple way to obtain a threshold that effectively adapts to the channel quality is using the maximum number of the failed parity-check messages among all variable nodes at the iteration.

2.1.1.2 Gallager A and B Algorithms

Gallager A and B algorithms [3] are also hard-decision decoding algorithm, similar to the BFA, in which all the messages are binary. However, the node operations in Gallager algorithms are different from that in the BFA.

To describe the Gallager A algorithm, we use $B_{i \rightarrow j,t}$ to denote the bit transmitted from variable node $v_i$ to check node $c_j$ and $B_{j \rightarrow i,t}$ for the bit from check node $c_j$ back to variable node $v_i$ at the $t$-th iteration. $C_i$ is used to denote the index set of check nodes connected to $v_i$ and $V_j$ for the index set of variable nodes connected to $c_j$.

Step 1. Initialization. Variable node $v_i$ makes the hard decision of the channel output and passes the channel bit $B_i$ to its connected check nodes.

$$B_{i \rightarrow j,0} = B_i$$ (2.2)

Step 2. Check Node Update. Check node $c_j$ computes the outgoing 1-bit message using the incoming bits from all the other edges. The bit sent from check node $c_j$ back to variable node $v_i$ at the $t$-th iteration is
2.1. LDPC Codes

\[ B_{j\rightarrow i,t} = \left( \sum_{i' \in V_j \setminus i} B_{i'\rightarrow j,t} \right) \mod 2 \]  \hspace{1cm} (2.3)

**Step 3. Codeword Test.** At variable node \( v_i \), the value of the code bit is determined based on a majority decision of all the incoming bits from the connected check nodes as (2.4). The channel bit at the variable node is used in the majority decision if the degree of the variable node is even.

\[ x_i = \begin{cases} \text{Majority (} B_{j\rightarrow i,t} \text{ for } j \in C_i \text{),} & \text{if } d_v \text{ is odd,} \\ \text{Majority (} B_i, B_{j\rightarrow i,t} \text{ for } j \in C_i \text{),} & \text{if } d_v \text{ is even.} \end{cases} \]  \hspace{1cm} (2.4)

Then, if \( x^T H = 0 \), terminate the decoding process. Otherwise, go to **Step 4**.

**Step 4. Variable Node Update.** The bit sent from variable node \( v_i \) to check node \( c_j \) is determined by all the incoming bit on the other edges. At variable node \( v_i \), the outgoing bit to \( c_j \) is set to \( B_i \) unless all the extrinsic bits from all the connected check nodes except \( c_j \) disagree with \( B_i \), in which case, the bit passed to \( c_j \) is set to the flipped bit value of \( B_i \).

\[ B_{i\rightarrow j,t} = \begin{cases} B_i, & \text{if } \sum_{j' \in C_i \setminus j} \left| B_{j'\rightarrow i,t-1} = B_i^c \right| < d_v - 1, \\ B_i^c, & \text{if } \sum_{j' \in C_i \setminus j} \left| B_{j'\rightarrow i,t-1} = B_i^c \right| = d_v - 1. \end{cases} \]  \hspace{1cm} (2.5)

where \( B_i^c \) is the flipped bit value of \( B_i \). Then, the algorithm returns to **Step 2** until the maximum number of iterations is exceeded.

The Gallager B algorithm is a refinement of the Gallager A algorithm at the **Step 4**. The outgoing bit from variable node \( v_i \) to its connected check node \( c_j \) in the Gallager B algorithm is
2.1. **LDPC Codes**

\[
B_{i \rightarrow j,t} = \begin{cases} 
B_i, & \text{if } \sum_{j' \in C \setminus j} |B_{j' \rightarrow i,t-1} = B_i^c| < \eta_t, \\
B_i^c, & \text{if } \sum_{j' \in C \setminus j} |B_{j' \rightarrow i,t-1} = B_i^c| \geq \eta_t.
\end{cases}
\]  

(2.6)

where \(B_i^c\) is the flipped bit value of \(B_i\), and \(\eta_t\) is the preset threshold value which could be changed over iterations. Therefore, the Gallager A algorithm can be thought of as a special case of the Gallager B algorithm with the threshold value equals to \((d_v - 1)\).

### 2.1.1.3 The Sum-Product Algorithm

The sum-product algorithm is an iterative decoding approach using soft information. In this algorithm, the messages received from the channel and those transmitted between nodes are the *a posteriori* probabilities. When Binary Phase-Shift Keying (BPSK) modulation with energy \(E_s\) is used to transmit binary symbols over an AWGN channel which has the noise variance \(\sigma_n^2 = N_0/2\), the Likelihood Ratio (LR) of the \(i\)-th received channel symbol \(y_i\) is

\[
\frac{P_{Y|C}(y_i|x_i = 0)}{P_{Y|C}(y_i|x_i = 1)} = \exp \left( \frac{2y_i \sqrt{E_s}}{\sigma_n^2} \right)
\]  

(2.7)

By taking the logarithm of (2.7), we have the Log-Likelihood Ratio (LLR) as

\[
L_i = \frac{2y_i \sqrt{E_s}}{\sigma_n^2}
\]  

(2.8)

The benefit of using the logarithm representation of probabilities is that where probabilities need to be multiplied, only addition is needed in the logarithmic domain, thus reducing the implementation complexity. Logarithmic domain computation also enhances the computational stability. The log-domain version of the SPA operates as follows \[34\].

**Step 1. Initialization.** All the variable nodes are initialized by the received symbol \(y_i\). Variable node \(v_i\) computes and sends the LLR value based on \(y_i\) to its connected check nodes \(c_j\).
2.1. LDPC Codes

\[ L_{i \rightarrow j,0} = L_i = \frac{2y_i \sqrt{E_s}}{\sigma_n^2} \]  \hspace{1cm} (2.9)

**Step 2. Check Node Update.** Check node \( c_j \) computes the message to variable node \( v_i \) with all LLR values received from its connected nodes other than the variable node \( v_i \). The LLR from check node \( c_j \) back to variable node \( v_i \) at the \( t \)-th iteration is

\[
L_{j \rightarrow i,t} = \prod_{i' \in \mathcal{V}_j \setminus i} \text{sign}(L_{i' \rightarrow j,t}) \cdot \ln \left( \frac{1 + \prod_{i' \in \mathcal{V}_j \setminus i} \tanh(|L_{i' \rightarrow j,t}|/2)}{1 - \prod_{i' \in \mathcal{V}_j \setminus i} \tanh(|L_{i' \rightarrow j,t}|/2)} \right) \]  \hspace{1cm} (2.10)

**Step 3. Codeword Test.** A hard-decision for each code bit \( x_i \), for \( 1 \leq i \leq n \), is determined by

\[
x_i = \begin{cases} 
0, & \text{if } \left( \sum_{j \in \mathcal{C}_i} L_{j \rightarrow i,t} + L_i \right) > 0, \\
1, & \text{if } \left( \sum_{j \in \mathcal{C}_i} L_{j \rightarrow i,t} + L_i \right) \leq 0.
\end{cases} \]  \hspace{1cm} (2.11)

When either \( x \mathbf{H}^T = 0 \) is satisfied or the maximum number of iterations is reached, the algorithm terminates.

**Step 4. Variable Node Update.** The LLR from variable node \( v_i \) to check node \( c_j \) is

\[
L_{i \rightarrow j,t} = \sum_{j' \in \mathcal{C}_i \setminus j} L_{j' \rightarrow i,t-1} + L_i \]  \hspace{1cm} (2.12)

Then, returns to **Step 2** for the next iteration.

Since the computational complexity in the SPA is relatively high, a simplified algorithm with lower computational cost is desired. The MSA is a less-complex approximation of the SPA and is described in the next section.
2.2. Challenges in Implementing LDPC Decoders

2.1.1.4 The Min-Sum Algorithm

The min-sum algorithm simplifies the update equation at the Step 2 of the SPA. Because the smallest $L_{i \rightarrow j}$ dominates the log term of (2.10), it can be approximated by

\[
\ln \left( \frac{1 + \prod_{i' \in V_j \setminus i} \tanh(|L_{i' \rightarrow j,t}|/2)}{1 - \prod_{i' \in V_j \setminus i} \tanh(|L_{i' \rightarrow j,t}|/2)} \right) \approx \min_{i' \in V_j \setminus i} |L_{i' \rightarrow j,t}| \tag{2.13}
\]

Thus, the LLR sent from check node $c_j$ to the connected variable node $v_i$ in the MSA becomes

\[
L_{j \rightarrow i,t} = \prod_{i' \in V_j \setminus i} \text{sign}(L_{i' \rightarrow j,t}) \cdot \min_{i' \in V_j \setminus i} |L_{i' \rightarrow j,t}| \tag{2.14}
\]

The computational complexity at the check nodes in the MSA is lower; however, it suffers from approximately 0.5 to 1-dB decoding loss compared to the SPA [35].

2.2 Challenges in Implementing LDPC Decoders

Implementation of LDPC decoders is usually based on one of the iterative decoding algorithms mentioned in Section 2.1.1. Since the messages in these iterative decoding algorithms are transmitted according to the code structure, the factor graph of the code needs to be constructed in hardware implementation of the decoder. In general, partially and fully parallel architectures are two main strategies to implement LDPC decoders. The partially parallel approaches only fabricate a portion of the factor graph, and employ memory and hardware resource sharing to manage the transmission between different portions of the factor graph. The benefits of the partially parallel approach are less area consumption and the flexibility to support different code lengths or code rates in applications such as WiMAX (IEEE 802.16e) [5], WiFi (IEEE 802.11n) [6], and G.hn (ITU-T G.9960) [9]. However, the partially parallel approach has a lower throughput compared to the fully parallel approach. The fully parallel approach implements the entire factor graph of the code and updates the messages transmitted between the nodes concurrently to achieve high-throughput. Hence,
2.2. Challenges in Implementing LDPC Decoders

the error-correcting decoder implemented by the fully parallel approach is mostly considered for the applications with high throughput requirements such as 10GBASE-T (IEEE 802.3an) [10].

However, the major challenge in implementations of LDPC decoders is the complexity of the interconnection network between nodes. The problem is worse for the decoders with the fully parallel design. This routine congestion problem gets even worse as the codeword length \( n \) increases. However, practical LDPC codes tend to have longer codeword length due to the improved performance of the code and error-correcting ability of iterative decoding algorithms. One extreme example is the LDPC code adopted in the DVB-S2 standard which has a codeword length of 64800 [36]. In addition, the messages passed iteratively in the soft-decision decoding algorithms, e.g., the SPA or the MSA, are \textit{a posteriori} probabilities. In order to transmit these type of messages, the number of wires will increase for finer precision. For example, the LDPC decoder introduced in [37], which has a fully parallel multi-bit architecture to transmit messages, requires a large number of wires in the hardware implementation.

A technique to reduce the number of wires required for transmitting messages between nodes is to perform the decoding process in a block-parallel bit-serial fashion. Only one bit is sent over a single wire at a time, and it accumulates the bits for later arithmetic operations. For instance, the bit-serial decoders using the MSA to decode the (256,128) LDPC code in [38] and the (660,480) LDPC code in [39]. This method greatly reduces routing congestion, but it consumes more chip area for the memories to store the multi-bit messages.

Another approach for LDPC decoders to exchange messages in a bit-serial fashion is using the differential decoding with the binary message passing algorithm [40]. Less hardware complexity and low computational requirements make this approach attractive for high-throughput applications [41]. However, the performance of this differential decoding method is confined by the quantization level of the channel message [42].

Therefore, a technique that can alleviate the routing congestion in iterative decoder with fully parallel architecture to achieve high throughput and provide improved performance with less area-consumption is desired in implementing LDPC decoders.
2.3 Stochastic Computation

Stochastic computation is an approach to design the low-precision digital circuits which provides hardware and energy efficiency. Stochastic computation represents continuous values by streams of discrete elements and processes the information in a serial-wise fashion. One major benefit of using stochastic computation instead of the conventional binary computing is the ability of performing complex computations in a bit-wise fashion without transforming the multi-bit streams back to soft information. Moreover, due to the direct bit operations, stochastic computations are implemented by simple logic arrangements that can operate at high clock rate to achieve high throughput.

2.3.1 Stochastic Representation

Stochastic computation can be traced back to 1956, Von Neumann [43] proposed a construction in which each variable can be represented by a bundle of physical wires carrying binary values. The ratio of 1 to 0 represents the value sent over these wires. In the mid-1960s, there were further works on stochastic computation. Gaines [44] and Poppelbaum et al. [45] gave a clear definition of stochastic computation and also implemented the stochastic operations of multiplication and summation using standard logic elements. In 1976, Poppelbaum described statistical processing with the concept of stochastic computation [46] in which the encoded information is represented by the probability of appearance of 1s in the stream. Some related works such as [47] and [48] are focused on generation of stochastic streams.

In stochastic computation, all variables are represented by streams of random elements. In the binary case, each bit of the stream is equal to 1 with the value of the information to be encoded, and most of the time, stochastic streams are used to represent probability values. The probability is represented by the appearance of 1s in the stream. For instance, a bit stream containing 75-percent 1s and 25-percent 0s indicates a probability $P = 0.75$, which means that the probability of observing a 1 at an arbitrary position in the stream is $P$. Figure 2.2 shows some possible streams for a probability of 0.75. Since each bit in the stream is generated randomly and independently, The relationship between
2.3. Stochastic Computation

\[
P = 0.75 \quad \Leftrightarrow \quad \begin{array}{c}
11010111 \\
10011111 \\
11101110 \\
01111110
\end{array}
\]

Figure 2.2: Some possibilities of stochastic streams for a probability of 0.75.

\[
\begin{array}{c}
\text{Random number } R \\
\downarrow \\
\text{Probability } P
\end{array} \quad \begin{array}{c}
P > R \\
\downarrow \\
\text{Stochastic stream}
\end{array}
\]

Figure 2.3: Structure of a stochastic number generator.

a probability value and stochastic stream is not unique; neither the length of the stream nor the position of 1s is fixed. A probability value can be represented by different stream, even with different length. For example, \((1, 1, 0, 1), (0, 1, 1, 1), \text{ and } (1, 0, 0, 1, 1, 1, 1, 1)\) are all possible representations of a probability \(P = 0.75\).

The circuit that converts probabilities to stochastic streams is called \textit{stochastic number generator}. The conventional stochastic number generator includes a random number generator and a comparator. Figure 2.3 shows the circuit of conversion from the probability \(P\) to stochastic bits. Stochastic bit is generated by comparing the value of probability \(P\) to a uniformly distributed random number \(R\), which varies each clock cycle. If the probability \(P\) is greater than random number \(R\), the stochastic number generator outputs a 1, and a 0 otherwise. The scheme of stochastic number generator implies that the conversion is a non-deterministic process. There are \(2^l\) possible length-\(l\) stochastic streams generated for any given probability. However, some stochastic streams have higher appearance than others after a long period of observation. For instance, \((0, 1, 0, 0)\) is more likely to be observed than \((1, 1, 1, 1)\) when the given probability \(P\) is 0.25.

Conversion from a stochastic stream back to the coded probability is much simpler, the probability \(P\) can be extracted by counting the number of 1s in the stream and then dividing the count by the length of the stream. Figure 2.4 shows this conversion by employing a counter. The precision level of this conversion is highly correlated to the length of stochastic stream. A length-\(l\) stochastic stream can represent numbers in the set
2.3. Stochastic Computation

![Conversion circuit from a stochastic stream to soft information.](image)

Figure 2.4: Conversion circuit from a stochastic stream to soft information.

Thus, only a small subset of the real numbers in \([0, 1]\) can be expressed exactly in stochastic computation. The highest resolution of a length-\(l\) stochastic stream is \(\frac{1}{l}\). At the example of presenting the probability of \(\frac{1}{1024}\), which only requires 10 bits in binary representation, stochastic representation needs 1024 bits to express exactly the probability. Generally, longer stream provides better precision of the encoded information. However, not all its applications requires the exact probability values presented by stochastic streams. Since each bit in the stochastic stream is independently generated corresponding to the encoded probability \(P\), the length of stochastic stream is not really crucial in the conversion between \(P\) and stochastic stream \([49]\). In some applications, such as iterative decoding, the flow of change in the statistic of the stochastic stream is more essential than the precise value presented by the entire frame of bits \([16, 50]\).

In addition, stochastic representation provides relatively higher error-tolerant ability than the binary number representation. Because each bit in the stochastic stream has the same weight in representing of the encoded information, flipping a single bit value in a stochastic stream will result in a small change in the value it denotes. This characteristic makes stochastic computation robust against noise. For instance, both of the binary number of 0.011 and the stochastic stream \((0, 0, 1, 0, 1, 0, 1)\) represent a value \((3/8)\). If a bit flips in the binary number from 0.011 to 0.001, its value changes from \((3/8)\) to \((1/8)\). A bit flips from 0.011 to 0.111 gives a greater change on the result from \((3/8)\) to \((7/8)\), since the most significant bit in a binary number has been flipped. However, flipping a bit in the stochastic stream \((0, 0, 1, 0, 1, 0, 1, 0)\) will change the value from \((3/8)\) to \((2/8)\) or \((4/8)\), which are the representable numbers closest to the correct value.
2.3.2 Stochastic Arithmetic Operations

By using stochastic representation, arithmetic operations such as multiplication, division, and addition can be implemented by simple logic gates. It needs to be noted that a proper arithmetic computation is only guaranteed when the input stochastic streams are independent. If this requirement cannot be satisfied, stochastic computation can fail dramatically [47]. Any two length-$l$ stochastic streams $a = (a_1 a_2 \cdots a_l)$ and $b = (b_1 b_2 \cdots b_l)$ are defined as independent if and only if

$$\sum_{i=1}^{l} a_i \cdot b_i = \frac{\sum_{j=1}^{l} a_j \times \sum_{k=1}^{l} b_k}{l}$$  \hspace{1cm} (2.15)

In order to ensure proper arithmetic operations, all the input stochastic streams used for the stochastic computations discussed in this section are assumed to be independent to each other. Many works have been done on generating stochastic streams that are sufficiently random and independent [51]. The details of these methods are described in [47], [52], and [48].

2.3.2.1 Multiplication

An AND gate is used to realize stochastic multiplication. Since any multi-input multiplication can be constructed by 2-input 1-output AND gates in a tree structure, we use a 2-input 1-output AND gate here to demonstrate the stochastic multiplier. Consider the two inputs of the AND gate are the binary streams $a$ and $b$ with the probabilities $P_a$ and $P_b$, which represent the appearance of 1 in the streams, respectively. After doing logical AND operation bit-wisely, the ratio of the appearance of 1 in the output stochastic stream of the AND gate is $P_c = P_a \times P_b$.

Figure 2.5 shows two examples of stochastic multipliers using AND gates. The two input streams to the AND gates present the probabilities $(4/8)$ and $(6/8)$ exactly. In Figure 2.5(a), the output probability of the AND gate is $(4/8) \times (6/8) = (3/8)$. The multiplier works properly, since the input streams are independent. For the case of Figure 2.5(b), the output probability is $(2/8) \neq (3/8)$ with the same input probabilities. The multiplier
2.3. Stochastic Computation

2.3.2.2 Division

The division in stochastic computation is realized by a JK flip-flop. Figure 2.6 shows the structure of stochastic divider. When the two input probabilities to the JK flip-flop are $P_a$ and $P_b$, the probability of observing 1 in the output stream of the JK flip-flop is

$$P_c = \frac{P_a}{P_a + P_b}$$

Figure 2.5: AND gate used as a stochastic multiplier with (a) independent input streams, and (b) dependent input streams.

![Figure 2.5: AND gate used as a stochastic multiplier](image)

Figure 2.6: Stochastic divider by a JK flip-flop.

![Figure 2.6: Stochastic divider by a JK flip-flop](image)

While this example illustrates the required independence between stochastic streams for proper stochastic computation.

It does not work correctly because the input stochastic streams do not satisfy the requirement of being independent. The two input streams in Figure 2.5(b) are dependent, since all the corresponding positions of 0s in $a$ are 1s in $b$ and the positions of 0s in $b$ are 1s in $a$. This example illustrates the required independence between stochastic streams for proper stochastic computation.
2.3. Stochastic Computation

\[ P_a = 4/8, \ a = (01011001) \]
\[ P_b = 6/8, \ b = (11101011) \]
\[ P_c = P_a + P_b - P_a P_b = 7/8 \]

Figure 2.7: Approximated stochastic adder by an OR gate.

\[ P_c = \frac{P_a}{P_a + P_b} \]  \hspace{1cm} (2.16)

This output probability of stochastic divider is an approximation of \( P_a / P_b \), when \( P_a \ll P_b \).

2.3.2.3 Addition

Stochastic addition is operated by the OR gate. As shown in Figure 2.7, the output bit stream of the OR gate represents \( P_c = P_a + P_b - P_a P_b \) for the two input streams with probabilities \( P_a \) and \( P_b \). The OR gate works as an approximate adder, which provides more accurate result while \( (P_a P_b) \) is small.

The operation of stochastic addition is not as straightforward as stochastic multiplication or division, since we cannot add two probability values directly; it could result in a value greater than one, which cannot be represented by stochastic stream. Therefore, the scaled addition was introduced in [53] to ensure the range of \([0, 1]\) for the output. The proposed scaled addition is performed as \( P_c = P_a \times P_s + P_b \times (1 - P_s) \), where \( P_a \) and \( P_b \) are the input probabilities and \( P_s \) is the scale probability. This type of scaled addition can be implemented by a multiplexer as shown in Figure 2.8.

2.3.3 Practical Applications

Stochastic computation has been recognized as a useful technique for low-complexity and reliable implementations due to its hardware simplicity and fault tolerance. Stochastic computing has been used in neural networks [54, 53, 55], control circuits [56, 57], reliability calculations [52, 58, 59], spectrum transforms [60], and energy-efficient circuit designs [61, 62].
Stochastic computation has also been used for approximate computing [63, 61]. Many complex functions such as exponentiation or the sine function can be implemented using stochastic computation [64, 65]. For example, the nonlinear sigmoid function in the control system of wind turbines can be realized by stochastic computing with low computational complexity [55]. In addition, stochastic computation is useful to synthesize the computation of polynomial arithmetics. Qian and Riedel [66] introduced a method that converts polynomial functions into a particular form, Bernstein polynomials more precisely, and then implements these polynomials by adders and multiplexers, which can be easily implemented by stochastic computation. This synthesis of polynomial arithmetic is now known as stochastic logic [66]. Later, Qian et al. [67] proved that when the coefficients of the Bernstein polynomial are in the interval of $[0, 1]$, the polynomial arithmetic can be realized by stochastic logic.

Furthermore, due to the error-tolerant ability of stochastic representation, stochastic computation is extended to the designs in different applications, such as an intersymbol interference compensation for communication systems [68], an error detector for nanoscale circuits [69], and a fault-tolerance implementation on imaging processing [70].

More recently, stochastic computation is employed to implement iterative decoders for LDPC codes [15, 16, 18] and a growing list of other error-correcting codes [71, 72]. Table 2.2 summarizes a brief historical development of stochastic computation.

Figure 2.8: Scaled stochastic adder by a multiplexer.
2.4. Stochastic Decoding

Stochastic computation was proposed as a hardware-efficient alternative to implement iterative LDPC decoders [15, 16]. Unlike the first fully parallel multi-bit LDPC decoder introduced in [37], which requires a large number of wires to transmit messages between nodes, the inherently serial messages in stochastic decoding only require a single wire. The bit-wise message greatly helps to alleviate the routine congestion in implementations of iterative decoders, especially for the practical LDPC codes which usually have complex interconnection network in the factor graph. In addition, the message in stochastic decoding is not considered by the entire frame of bit stream; the activity in the statistic of the stochastic stream is more crucial than the actual value represented by the entire stream.
2.4. Stochastic Decoding

in the decoding process. Therefore, stochastic decoder does not need memory to accumulate the bit-wisely transmitted messages prior to the node operations.

Because of its efficiency in hardware implementation, stochastic computation has been used to decode a (16,8) LDPC code [15], a (1024,512) LDPC code [17], the (1056,528) LDPC code in the WiMAX standard [18], the (2048,1723) LDPC code in the 10GBASE-T standard [19], the (672,588) LDPC code used in the WPAN standard [73], non-binary LDPC codes [74, 20, 75], and many other error-correcting codes [72, 76, 71, 77, 78].

2.4.1 Stochastic LDPC Decoding

In stochastic decoding, in each clock cycle only one bit is computed and passed through each edge. Each decoding round of stochastic decoding is called a Decoding Cycle (DC) [16], which includes the exchange of one bit on each edge of the Tanner graph. One DC does not directly correspond to one iteration in the SPA or the MSA. However, the node operations in a stochastic decoder are similar to those in the probability-domain version of the sum-product decoding, but are processed in a bit-serial fashion.

Here, we use $P_{i \rightarrow j,t}$ to denote a posteriori probability $P(x_i = 1 | y_i)$ from variable node $v_i$ to check node $c_j$ and $P_{j \rightarrow i,t}$ to denote the probability $P(x_i = 1 | y_i)$ from check node $c_j$ back to variable node $v_i$ at the $t$-th DC. The probability-domain operation of variable node $v_i$ in the SPA (2.12) can be rewritten as

$$P_{i \rightarrow j,t} = \frac{P_i \prod_{j' \in C_i \setminus j} P_{j' \rightarrow i,t-1}}{P_i \prod_{j' \in C_i \setminus j} P_{j' \rightarrow i,t-1} + (1 - P_i) \prod_{j' \in C_i \setminus j} (1 - P_{j' \rightarrow i,t-1})} \quad (2.17)$$

where $P_i$ is the channel receiving probability at $v_i$.

Similarly, at the check node, we have

$$P_{j \rightarrow i,t} = \frac{1 - \prod_{i' \in V_j \setminus i} (1 - 2P_{i' \rightarrow j,t})}{2} \quad (2.18)$$
2.4. Stochastic Decoding

Figure 2.9: The structures of (a) a variable node and (b) a check node in stochastic decoders.

In the hardware implementation of stochastic decoders, higher-degree nodes can be constructed by sub-graphs containing only degree-3 nodes [15] [18]. Hence, we use a 2-input 1-output node to demonstrate the node operations in stochastic decoders.

2.4.1.1 Stochastic Variable Node Operation

If we simplify (2.17) for a degree-3 variable node with two input probabilities \( P_a = P(B_a = 1) \) and \( P_b = P(B_b = 1) \), we have the output probability \( P_c \)

\[
P_c = \frac{P_a P_b}{P_a P_b + (1 - P_a)(1 - P_b)}
\]

(2.19)

The stochastic computation of (2.19) can be realized by an AND gate, a NOR gate, and a JK flip-flop as shown in Figure 2.9(a). This structure can also be simplified by using a binary C-element [79].

2.4.1.2 Stochastic Check Node Operation

At the check node, when the two input probabilities are \( P_a \) and \( P_b \), the output probability \( P_c \) of the degree-3 node is

\[
P_c = P_a (1 - P_b) + (1 - P_a) P_b
\]

(2.20)
2.4. Stochastic Decoding

Figure 2.10: An example of bit transmissions at (a) variable node \( v_i \) and (b) check node \( c_j \) in stochastic decoding.

The structure for the check node operation (2.20) is shown in Figure 2.9(b), which only requires an XOR gate to realize the even parity-check.

2.4.1.3 Summary of Stochastic LDPC Decoding

With the knowledge of node operations in stochastic LDPC decoder, we summarize the stochastic decoding process as follows.

**Step 1. Initialization.** Each variable node employs stochastic number generator to transform the received channel probability to a stochastic bit and sends the 1-bit message to its connected check nodes.

**Step 2. Check Node Update.** Check node computes the 1-bit message by Figure 2.9(b) using the incoming bits from all the other edges, and then sends it back to the connected variable node.

**Step 3. Codeword Test.** After receiving the bits from the neighboring check nodes, each variable node determines the code bit by Figure 2.9(a) with all the received bits. The decoding process is terminated if a valid codeword is decoded or a predetermined number of decoding cycles is achieved.
2.4. Stochastic Decoding

Step 4. Variable Node Update. The 1-bit message from the variable node to its connected check node through the edge is computed by Figure 2.9(a) using the next channel bit and the extrinsic bits from all the other edges.

The algorithm returns to Step 2 until the maximum number of DCs is exceeded.

An example of bit transmissions at \( v_i \) and \( c_j \) in stochastic decoding is illustrated in Figure 2.10, where \( \{c_\alpha, c_\beta\} \in C_i \) and \( \{v_\gamma, v_\delta\} \in V_j \).

2.4.2 The Latching Problem

As mentioned in Section 2.3.2, the input streams of stochastic computation should be independent of each others to ensure proper operation. This inaccuracy issue worsens as the number of inputs increases or has multiple layers of logic components. A possible solution for the problem is applying the mechanism that regenerates new independent bits corresponding to the desired probability for each computation. Especially in the stochastic LDPC decoding, an observed difficulty is the sensitivity of randomness in the stochastic streams [80]. An inherent defect resulting in lack of randomness in stochastic streams is due to the JK flip-flop in the Figure 2.9(a) structure which forces variable node in the stochastic decoder to output the same value as the previous output, when the two input bits are not equal. This problem is known as latching [16]. When latching occurs, the variable node which outputs the previous value is said to be in the hold state [18].

The latching problem becomes worse when stochastic decoding operates on graphs with cycles, where a group of nodes may lock into the hold state and maintain the output values despite the new incoming information [81] [82]. For example, if the bits received from an edge are all zeros for consecutive DCs, then the variable node will keep outputting zero regardless of any activity in the other receiving messages. It has been observed that the latching problem can be worse at high Signal-to-Noise Ratio (SNR) region in which the received channel probabilities approach zero (or one). In this case, the bits in the stochastic streams are mostly 0 (or 1), i.e., switching activity in streams for proper stochastic decoding occurs rarely [16]. Moreover, the latching problem prolongs the convergence of stochastic decoding, which results in more power consumption and greater decoding delay [83].
2.4. Stochastic Decoding

2.4.3 Previous Work on Stochastic Decoding

One method to resolve the latching problem is using a special structure called *supernodes* to replace the variable nodes in stochastic decoders [81]. The supernode proposed in [81] first tabulates the incoming stochastic streams to estimate their encoded probabilities, and then does the conventional SPA node operation by using these probabilities. Before sending the messages to the neighboring check nodes, the outgoing messages are transformed back to stochastic streams. Although, this kind of half-stochastic decoding structure alleviates the latching problem, it also increases the area-consumption in hardware implementations. Later, another type of supernode was introduced in [50], in which instead of replacing the variable node in stochastic decoders, this supernode works as the middleman between stochastic variable node and check node. It estimates the probability transmitted on the edge and regenerates new stochastic bits for later node operations. However, due to the performance loss, the supernode structure only works for short codes, e.g., a specially-constructed tail-biting (16,8) LDPC code [50].

For decoding longer practical LDPC codes, a moving average filter method, namely Edge Memories (EMs) [16], was introduced to decode the (1056,528) LDPC code used in the WiMAX standard. EMs are shift registers assigned to each outgoing edges of variable nodes. When input bits of a variable node are in agreement, i.e., the variable node is not at the hold state, the EM is updated with the incoming bit. On the contrary, the EM is not updated when the variable node is in the hold state. Thus, EMs contain only the bits which are produced when the variable node is not in a hold state. A stochastic bit is randomly picked from the EM and passed through the edge as the outgoing bit while the variable node is in the hold state. With this updating scheme, the mechanism of EMs helps to improve the efficiency of variable node operation in stochastic decoding by providing bits corresponding to the stored probabilities.

A similar mechanism, Tracking Forecast Memory (TFM) [84], was introduced later for providing similar performance but less area-consumption. The TFM utilizes the *successive relaxation* [85] to estimate the probability of the outgoing stochastic bit being 1 from the variable node. The stored probability in the TFM at the \( t \)-th DC, \( P_{m,t} \), is recursively updated by using
2.4. Stochastic Decoding

\[ P_{m,t} = (1 - \delta) P_{m,t-1} + \delta B_t \]  

(2.21)

where \( 0 < \delta < 1 \) is the relaxation coefficient, and \( B_t \) is the updated bit at the \( t \)-th DC. The TFM also only updates the stored probability when the variable node is not in the hold state. In [84], Tehrani et al. show that the TFMs require 8 or 9-bit memories for a degree-6 node to have similar performance as the EM with the size of 64. More recently, an area efficient design based on the TFM, called Majority-based Tracking Forecast Memory (MTFM) [86], was proposed for the implementation of a stochastic decoder that decodes the (2048,1723) LDPC code from the 10GBASE-T standard. In the MTFM-based stochastic decoder, only one storage mechanism is needed for each variable node, instead of each edge. All these moving average filter designs help to alleviate the latching problem and improve the performance of stochastic decoder by enhancing the efficiency of stochastic node operations.

In addition to the moving average filter designs, many approaches have also been presented to make stochastic decoding suitable for long LDPC codes. For example, an algorithm called the relaxed half-stochastic decoding [87, 88], which combines the component of variable node from the SPA and the low-complexity design of check node from stochastic decoder, was presented to decode a (2048,1723) LDPC code. Furthermore, an alternative approach called the delayed stochastic decoding [19] was introduced to track the probability values of the outgoing stochastic streams from variable nodes without using the area-consuming memories. The check node in the delayed stochastic decoder performs the parity-check operation only when reliable messages are available from all its connected variable nodes and delays the check node operation in other cases. However, compared to the stochastic decoders with the moving average filter design, the delayed stochastic decoder suffers from an approximately 0.2-dB performance loss [19]. Therefore, the stochastic decoders in this dissertation all use the moving average filter, especially the EM-based stochastic decoders.

Aside from the circuit designs for improving error rate performance, some works focus on increasing the throughput of stochastic decoders. For instant, the techniques that use asynchronous clock design were introduced to obtain higher throughput [89, 90]. From
circuit design aspect, the throughput of stochastic decoder with synchronous node updating scheme is limited by the delay of transmission on the longest wire [91]. An asynchronous design, in which each node begins its next node operation immediately after completing the previous one, shows the increment on throughput of stochastic decoders [92, 93].

Based on the values obtained from [19] and [86], a comparison of the hardware implementations of LDPC decoders using different decoding algorithms is presented in Table 2.3, which illustrates the low area-consumption of stochastic decoding. Furthermore, stochastic decoder can operate at high clock rate, which results in high throughput. It is because of the simplicity of stochastic computation that one DC in stochastic LDPC decoding is equal to one clock cycle. For example, the throughput of the MTFM-based stochastic decoder with early termination can be up to 61 Gb/s. The throughput of the fully parallel implemented SPA decoder [94] is only 5.3 Gb/s, while decoding the same LDPC code.
Table 2.3: Comparison on hardware implementations of LDPC decoders adopting different decoding algorithms.

<table>
<thead>
<tr>
<th>Code</th>
<th>Regular (2048,1723) LDPC code from 10GBASE-T standard</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[94]</td>
</tr>
<tr>
<td>Decoding algorithm</td>
<td>SPA</td>
</tr>
<tr>
<td>Iterations or DCs</td>
<td>16</td>
</tr>
<tr>
<td>Implementation strategy</td>
<td>Fully parallel</td>
</tr>
<tr>
<td>CMOS technology</td>
<td>90 nm</td>
</tr>
<tr>
<td>Area</td>
<td>$14.5 , mm^2$</td>
</tr>
<tr>
<td>Clock</td>
<td>207 MHz</td>
</tr>
<tr>
<td>Clocks per decoding round</td>
<td>5</td>
</tr>
<tr>
<td>Decoding latency</td>
<td>386 ns</td>
</tr>
<tr>
<td>Throughput</td>
<td>5.3 Gb/s</td>
</tr>
</tbody>
</table>
Chapter 3

Statistical Analysis of Stochastic LDPC Decoding

In stochastic decoding, all transmitted messages are sent in a bit-wise fashion representing the corresponding probabilities. This means that the messages that can be analyzed with probabilistic model are the received messages from the channel, the probabilities stored in the moving average filter mechanism, e.g., EM or TFM, and the decision process of code bits at the end of each DC. In this chapter, we investigate the stochastic LDPC decoding from these three aspects and based on the study give suggestions to improve the performance of stochastic decoders.

3.1 Introduction

In the first part of this chapter, we address the required randomization in stochastic decoding algorithm on the binary output channels. We propose a scaling method to introduce randomness in the stochastic streams and thus improve the error rate performance of stochastic decoding on the binary channel models. After applying the proposed method, the scaled channel messages become independent of the channel error probability that simplifies the generation of stochastic streams.

Later, we utilize EMs to discuss the transition probabilities between nodes in stochastic LDPC decoders. We introduce a Markov chain model for EMs to investigate the behavior of the decoding process and study its state transitions over DCs. The proposed model can be used to determine the speed of convergence of transition probabilities and the required number of DCs for stochastic decoding. In addition, by using the proposed Markov
3.2 Randomization in Stochastic Stream

In the stochastic stream model, we demonstrate that when utilizing both the extrinsic messages from check nodes and the channel message to determine the next channel bit, the stochastic decoding process converges faster than merely using channel messages.

We also present an efficient code bit determination scheme for stochastic decoders. We use the last stochastic bit from the variable node directly as the final code bit. This approach not only gives comparable error rate performance, but also saves chip area in hardware implementation by removing redundant mechanisms.

3.2 Randomization in Stochastic Stream

Since stochastic decoders are very sensitive to the switching activity in the stochastic streams, as stated in Section 2.4.2, the latching problem degrades their performance. The latching problem occurs when the set of input bits to a variable node are unequal for consecutive DCs. Under this condition, the output bits get stuck in the same value. Note that when the channel received LLR values tend to infinity (or minus infinity), the probability $P_i$ used to generate stochastic bits approaches zero (or one). Consequently, majority of bits in the corresponding stochastic stream will be 0s (or 1s).

Noise-Dependent Scaling (NDS) has been proposed to reduce the latching problem for the AWGN channel model [16]. The main purpose of NDS is to increase the randomness in the stochastic streams by scaling the channel received LLRs. With NDS, received LLRs are scaled by a factor proportional to the noise power, thus NDS ensures a similar level of switching activity for different SNR values. Furthermore, the scaled messages are independent of channel noise and thus the decoder does not need to estimate the noise level of channel. Scaling LLR values moves the probability $P_i$ toward 0.5, which introduces more switching activity for the corresponding stochastic stream.

For a BPSK-modulated message over an AWGN channel, the scaled LLR $L'_i$ for the $i$-th received symbol $y_i$ is

$$L'_i = (2\beta\sigma_n^2) L_i = 4\beta y_i \sqrt{E_s}$$  \hspace{1cm} (3.1)
3.2. Randomization in Stochastic Stream

where \( L_i = \frac{(2y_i\sqrt{E_s})}{\sigma_n^2} \) is the \( i \)-th received LLR from an AWGN channel, and \( E_s \) is the energy of the BPSK signal. The scaling factor on LLR is \((2\beta\sigma_n^2)\), in which \( \beta \) is empirically chosen based on the Bit Error Rate (BER) performance of the stochastic decoder [16]. For example, \( \beta \) is chosen as 0.5 for the irregular (1056,528) LDPC code from the WiMAX standard, and \( \beta = 1.33 \) for the regular (2048,1723) LDPC in the 10GBASE-T standard.

3.2.1 Scaling Method for Binary Symmetric Channel

The channel received messages from a Binary Symmetric Channel (BSC) are either 0 or 1. If we directly use these binary values to generate stochastic bits, the bits in the stream will be all-zero or all-one without any switching activity. Therefore, it is more essential to introduce some randomness on the channel messages before entering the stochastic decoder. In this section, we introduce a scaling method for BSC, which is a common channel model used for communication with hard decision at the output. A BSC model with crossover probability of \( \varepsilon \) is shown in Figure 3.1.

Let \( \hat{y}_i \) be the received binary message at the \( i \)-th variable node from the BSC with crossover probability \( \varepsilon \). The corresponding LLR is

\[
L_i = (-1)^{\hat{y}_i} \ln \left( \frac{1 - \varepsilon}{\varepsilon} \right)
\] (3.2)

Therefore, the a posteriori probability \( P_i = P(x_i = 1|\hat{y}_i) \) for the BSC model is

\[
P_i = \varepsilon (1 - \hat{y}_i) + (1 - \varepsilon) \hat{y}_i
\] (3.3)
3.2. Randomization in Stochastic Stream

It has been observed that the performance of stochastic LDPC decoders gets worse at high SNR values if no scaling is used [16]. It is because that at high SNRs, the received probability $P_i$ are very close zero (or one) which is highly possible to lead to the latching problem due to lack of switching in the stochastic stream. Therefore, the direct conversion of $P_i$ in (3.3) to the corresponding stochastic stream will lead to high error rate at high SNR region.

For the BSC, we introduce a scaling method, which the scaled probability $P'_i$ becomes a constant and independent of the BSC crossover probability. The probability used to generate stochastic streams in the proposed scaling method is

$$P'_i = \hat{\beta} (1 - \hat{y}_i) + \left(1 - \hat{\beta}\right) \hat{y}_i$$

(3.4)

where $P'_i$ is the scaled a posteriori probability $P(x_i = 1|\hat{y}_i)$ and $0 < \hat{\beta} \leq 0.5$ due to the symmetry in $P'_i$.

After employing the proposed scheme, the probability used in the conversion from the channel message to stochastic streams is independent of the crossover probability $\varepsilon$ of the BSC and similar to NDS for the AWGN channel, the decoder does not need to estimate the channel error probability. Stochastic streams are directly generated by using stochastic number generator, in which comparing the uniformly distributed random number $R$ with the constant value $\hat{\beta}$ or $(1 - \hat{\beta})$, instead of $\varepsilon$ or $(1 - \varepsilon)$.

3.2.1.1 Results for the BSC Model

Based on the BER performance, we select $\hat{\beta}$ as 0.12 for the (1056,528) LDPC code from the WIMAX standard. Note that the $\hat{\beta}$ value is independent of $\varepsilon$ of the channel and different code may have different $\hat{\beta}$ value to achieve its least error rates. Figure 3.2 shows the BER curves of the (1056,528) LDPC code decoded by stochastic decoding with and without using the proposed scaling method on the BSC. The maximum number of DCs used for both stochastic decoders is selected to be 700. Compared to the case where the crossover probabilities $\varepsilon$ are used to generate the stochastic streams, the scaling method gives approximately 1-dB improvement at high SNR values.
3.2. Randomization in Stochastic Stream

Figure 3.2: BER performance of the (1056,528) LDPC code decoded by the stochastic decoding with and without the scaling method on the BSC.

Figure 3.3: Average number of DCs for the (1056,528) LDPC code decoded by the stochastic decoding with and without the scaling method on the BSC.
3.2. Randomization in Stochastic Stream

From Figure 3.2, we also observe that the scaling method not only gives better bit error rate performance, but also lowers the error floor of stochastic decoding on the BSC. In addition, the latency of stochastic decoding using the proposed scaling scheme has an 1.5-dB improvement at high SNRs, i.e., approximately 1/3 of the average DCs for the case without using any scaling scheme, as shown in Figure 3.3.

3.2.1.2 Comparison with the AWGN Channel

Although, the proposed scaling method is for the probability messages received from the BSC, this scheme has the same goal, i.e., introducing more switching activities in the stochastic streams, similar to NDS for the AWGN channel. Here, we present a comparison between these scaling schemes proposed for different channel models. For the AWGN channel model, the scaling factor in the NDS is chosen as \((2\beta\sigma_n^2) = N_0/2\) with \(\beta = 1/2\) for the (1056,528) LDPC code from the WiMAX standard, as recommended in [16]. With \(E_s = 1\), the unscaled LLR from the AWGN channel is \((4y_i/N_0)\) and the NDS-scaled LLR is \((2y_i)\). Therefore, the probabilities \(P_i\) and \(P'_i\) in the NDS case are

\[
P_i = \frac{1}{1 + \exp\left(\frac{4y_i}{N_0}\right)}
\]

and,

\[
P'_i = \frac{1}{1 + \exp(2y_i)}
\]

where the received variable \(y_i\) is a Gaussian distributed random variable with the variance \(N_0/2\). Figure 3.4 shows the probability distributions of the scaled probabilities \(P'_i\) with \(y_i \geq 0\) and \(y_i < 0\) at different SNR values. It illustrates that the noise-dependent scaled \(P'_i\) moves toward 0.5. When SNR value increases, the scaled probability \(P'_i = P(x_i = 1|y_i)\) for \(y_i \geq 0\) in the case of using NDS converges to the value 0.12, which is consistently as the constant value \(\hat{\beta}\) in our propose method.

Note that the scaled probabilities \(P'_i\) from both of the AWGN channel with NDS and the BSC with the proposed scaling method are independent of the channel error probability. The BSC with the proposed scaling model can be equally represented by the AWGN
3.2. Randomization in Stochastic Stream

Figure 3.4: Histograms of $P(x_i = 1 | y_i \geq 0)$ and $P(x_i = 1 | y_i < 0)$ for BPSK transmission over the AWGN channel with NDS.

channel with the NDS and a hard-limiter. Figure 3.5 shows these equivalent models, and $y_i$ and $\hat{y}_i$ represent the received message from the AWGN and the BSC channels, respectively. $y_i'$ denotes the scaled continue symbol and the scaled binary symbol is presented by $\hat{y}_i'$. In Figure 3.5(a), the hard-limit transfer function $f_1$ is

$$\hat{y}_i = f_1(y_i) = 1 - U(y_i)$$

(3.7)

where $U(y_i)$ is an unit step function of $y_i$. For the AWGN channel with NDS, the channel received symbol $y_i$ after the NDS becomes $y_i'$. The hard-limit transfer function in the equivalent model Figure 3.5(b) is

$$\hat{y}_i' = f_2(y_i') = \text{sgn}(y_i')$$

(3.8)
3.2. Randomization in Stochastic Stream

Figure 3.5: The equivalent models: (a) the BSC with the proposed scaling method; (b) the AWGN channel with the NDS and a hard-limiter.

\[ x_i \quad \begin{array}{c} 0 \quad 1 \quad \varepsilon \quad 1 - \varepsilon \\ y_i \quad 0 \quad 0 \quad \varepsilon \quad e \quad \varepsilon \quad 1 - \varepsilon \quad 1 \quad 0 \quad 1 \end{array} \]

(a) (b)

Figure 3.6: (a) A Z-channel and (b) a binary erasure channel models.

where \( \text{sgn}(y'_i) \) is a sign function of \( y'_i \): when \( y'_i \geq 0 \), \( \text{sgn}(y'_i) = 1 \); otherwise, \( \text{sgn}(y'_i) = -1 \).

Thus, the probabilities \( P(x_i = 1|y'_i) \) used to generate stochastic streams for both models in Figure 3.5 are the same, \( \hat{\beta} \) or \( (1 - \hat{\beta}) \), since (3.6) equals to \( \hat{\beta} \) when \( y_i = 1 \) and \( (1 - \hat{\beta}) \) when \( y_i = -1 \).

3.2.1.3 Other Binary Output Channel Models

The proposed scaling method in this section is not limited to the BSC. We can apply the scaling method to the Z-channel. As shown in Figure 3.6(a), the Z-channel has only one crossover error from 1 to 0 with the probability \( \varepsilon \) and crossover error from 0 to 1 never
3.2. Randomization in Stochastic Stream

occurs. Without scaling, the probability from the Z-channel used to generate the stochastic bits at variable node $v_i$ is

$$P_i = \left( \frac{\varepsilon}{1 + \varepsilon} \right) (1 - \hat{y}_i) + \hat{y}_i \quad (3.9)$$

After applying the proposed scaling method, the scaled probability becomes

$$P'_i = \hat{\beta} (1 - \hat{y}_i) + \hat{y}_i \quad (3.10)$$

The scaling method lets $P'_i$ be independent of the crossover error probability $\varepsilon$.

The BER performance of stochastic decoding with and without using the scaling method on the Z-channel is shown in Figure 3.7. With the constant scaling factor $\hat{\beta} = 0.12$
for the (1056,528) LDPC code, the scaling method provides improvement in the BER and greatly lowers the error floor of stochastic decoding.

It needs to be mentioned that not all the binary output channels require the scaling method to introduce the switching activity in the stochastic streams. The Binary Erasure Channel (BEC) model is an example that does not need the scaling approach. In the BEC, some bits get erased with the probability \( \varepsilon \), as shown in Figure 3.6(b). There is no error when receiving a 0 or 1 from the BEC. The a posteriori probability \( P(x_i = 1|\hat{y}_i) \) is 0.5 when an erasure message is received, i.e., the only erroneous case. It means that the corresponding stochastic streams for the only erroneous case of the BEC are generated by the probability of 0.5. Therefore, the required randomness for proper stochastic decoding inherently exists in the stochastic streams from the BEC model.

### 3.3 Transition Probabilities in Stochastic LDPC Decoding

In this section, we present a Markov chain model to analyze the transition probabilities in stochastic LDPC decoding. This Markov chain model helps us understand the behavior of stochastic decoding. We utilize EM to construct the model that lets us investigate the messages transmitted between nodes using a probabilistic perspective.

#### 3.3.1 Markov Chain Model for Edge Memories

We define the state of the EM based on the number of 1s stored in the EM. State 0 denotes the case where the bits in the EM are all-zeros. State 1 means there is only one 1 in the shift register, and State \( L \) stands for the all-ones state. Hence, there will be \( (L + 1) \) states for the length-\( L \) EM. Under the infinite codeword length and cycle-free assumption, we conclude that all EMs are identical and independent in the stochastic decoder. Figure 3.8 shows a Markov chain model for a length-\( L \) EM.

Since only one bit is updated in the EM when the variable node is not in the hold state at the DC, we see that except State 0 and State \( L \), all the other states have three transitional options: staying at the same state, or moving to one of its neighboring states. When the update bit of the EM is 1 and the shifted out bit from it is 0, it will transit from State 1 to
3.3. Transition Probabilities in Stochastic LDPC Decoding

State \((l + 1)\). On the contrary, if the update bit is 0 and the shifted out bit is 1, it will transit to State \((l - 1)\). If the variable node is locked in the hold state, i.e., not updating the EM, or the update bit and the discarded bit of the EM are the same, the EM will stay in the same state.

The transition probabilities of the Markov chain model is evolved as follows. The initial probability for each state of the EM is initialized by the received channel probability \(P_i\). Later, the probabilities evolve according to the extrinsic probabilities from the connected check node \(P_{j \rightarrow i, t}\) and the channel probability \(P_i\). At the \(t\)-th DC, the probability of each state of the Markov chain model is

\[
P_t(S) = \begin{bmatrix} P_t(S_0) \\ P_t(S_1) \\ \vdots \\ P_t(S_L) \end{bmatrix}
\]

(3.11)

where

\[
P_t(S_l) = P_{l-1}(S_{l-1}) \left[ \left( \frac{L - (l - 1)}{L} \right) P_i (P_{j \rightarrow i, t})^{d_v - 1} \right] + \\
\left[ 1 + \left( \frac{l - L}{L} \right) P_i (P_{j \rightarrow i, t})^{d_v - 1} + \left( \frac{1 - P_i}{L} \right) (1 - P_i) (1 - P_{j \rightarrow i, t})^{d_v - 1} \right] + \\
\left( \frac{l + 1}{L} \right) (1 - P_i) (1 - P_{j \rightarrow i, t})^{d_v - 1}
\]

(3.12)

for \(0 \leq l \leq L\) and \(t > 0\). Here \(P_i\) is the probability obtained from the channel at the \(i\)-th variable node, and \(P_{j \rightarrow i, t}\) denotes the extrinsic probability from check node \(c_j\) to variable node \(v_i\) at the \(t\)-th DC.

![Figure 3.8: A Markov chain model for a length-\(L\) EM.](image)
3.3. Transition Probabilities in Stochastic LDPC Decoding

The first term on the right hand side of (3.12) is the transition probability from State \((l - 1)\) to State \(l\), the second term represents the probability of staying in the same state, and the last term is the probability of moving from State \((l + 1)\) to State \(l\).

We can use (3.11) to compute the probability of the outgoing bit \(B_{i\to j,t}\), which is sent from variable node \(v_i\) to its connected check node \(c_j\) at the \(t\)-th DC, is equal to 1. The probability \(P_{i\to j,t} = P(B_{i\to j,t} = 1)\) is

\[
P_{i\to j,t} = P_t^T(S)P(B_{i\to j,t} = 1|S) \tag{3.13}
\]

We also have

\[
P(B_{i\to j,t} = 1|S) = \begin{bmatrix}
P(B_{i\to j,t} = 1|S_0) \\
P(B_{i\to j,t} = 1|S_1) \\
\vdots \\
P(B_{i\to j,t} = 1|S_L)
\end{bmatrix} \tag{3.14}
\]

where

\[
P(B_{i\to j,t} = 1|S_l) = \frac{l}{L} \left[ 1 - P_i \left( P_{j\to i,t-1} \right)^{d_v-1} - (1 - P_i) \left( 1 - P_{j\to i,t-1} \right)^{d_v-1} \right] + P_i \left( P_{j\to i,t-1} \right)^{d_v-1} \tag{3.15}
\]

for \(0 \leq l \leq L\) and \(t > 0\). The first term in (3.15) is the probability that the bit \(B_{i\to j,t}\) is 1 when all the incoming bits at \(v_i\) are not in agreement and the bit randomly picked from the EM is 1. The second term is the probability that \(B_{i\to j,t} = 1\) when the incoming bits from all the other edges of \(v_i\) are 1s, i.e., \(v_i\) is not in a hold state.

The extrinsic probability from check node \(c_j\) to variable nodes \(v_i\) is computed as

\[
P_{j\to i,t} = \frac{1 - \left( 1 - 2P_{i\to j,t} \right)^{d_c} - 1}{2} \tag{3.16}
\]

which is the probability that the node operation at check node is even parity-check.

Numerical results for the Markov chain model in stochastic decoding of a \((d_v, d_c) = (2, 3)\) LDPC code are shown in Figure 3.9, which includes the evolution of the probabilities \(P_{i\to j,t}\) and \(P_{j\to i,t}\) for 20 DCs. In these simulations, the all-zero codeword is transmitted over
3.3. Transition Probabilities in Stochastic LDPC Decoding

Table 3.1: The transition matrices of a length-4 EM in stochastic decoding a $(d_v, d_c) = (2, 3)$ LDPC code at the $t$-th DCs.

<table>
<thead>
<tr>
<th>$t$</th>
<th>$S_{0,t-1}$</th>
<th>$S_{1,t-1}$</th>
<th>$S_{2,t-1}$</th>
<th>$S_{3,t-1}$</th>
<th>$S_{4,t-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t = 1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{0,t-1}$</td>
<td>0.9750</td>
<td>0.0250</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$S_{1,t-1}$</td>
<td>0.1740</td>
<td>0.8073</td>
<td>0.0187</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$S_{2,t-1}$</td>
<td>0</td>
<td>0.3479</td>
<td>0.6396</td>
<td>0.0125</td>
<td>0</td>
</tr>
<tr>
<td>$S_{3,t-1}$</td>
<td>0</td>
<td>0</td>
<td>0.5219</td>
<td>0.4719</td>
<td>0.0062</td>
</tr>
<tr>
<td>$S_{4,t-1}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.6958</td>
<td>0.3042</td>
</tr>
</tbody>
</table>

| $t = 10$ |  |  |  |  |  |
| $S_{0,t-1}$ | 0.9987 | 0.0013 | 0 | 0 | 0 |
| $S_{1,t-1}$ | 0.2178 | 0.7812 | 0.0010 | 0 | 0 |
| $S_{2,t-1}$ | 0 | 0.4357 | 0.5637 | 0.0006 | 0 |
| $S_{3,t-1}$ | 0 | 0 | 0.6535 | 0.3462 | 0.0003 |
| $S_{4,t-1}$ | 0 | 0 | 0 | 0.8714 | 0.1286 |

| $t = 20$ |  |  |  |  |  |
| $S_{0,t-1}$ | 0.9999 | 0.0001 | 0 | 0 | 0 |
| $S_{1,t-1}$ | 0.2200 | 0.7799 | 0.0001 | 0 | 0 |
| $S_{2,t-1}$ | 0 | 0.4400 | 0.5600 | 0 | 0 |
| $S_{3,t-1}$ | 0 | 0 | 0.6600 | 0.3400 | 0 |
| $S_{4,t-1}$ | 0 | 0 | 0 | 0.8800 | 0.1200 |

| $t = 30$ |  |  |  |  |  |
| $S_{0,t-1}$ | 1 | 0 | 0 | 0 | 0 |
| $S_{1,t-1}$ | 0.2202 | 0.7798 | 0 | 0 | 0 |
| $S_{2,t-1}$ | 0 | 0.4404 | 0.5596 | 0 | 0 |
| $S_{3,t-1}$ | 0 | 0 | 0.6605 | 0.3395 | 0 |
| $S_{4,t-1}$ | 0 | 0 | 0 | 0.8807 | 0.1193 |

the channel. We have selected the length of EMs to be 4 and simulated at SNR of 3 dB. The state transition matrices at various DCs are shown in Table 3.1.

After approximately 30 decoding cycles, the state transition matrix approaches the steady state. In the steady state, the transition probability from State 0 to the same state is equal to 1 indicating the EM approaching to the all-zero state, which is in agreement with
3.3. Transition Probabilities in Stochastic LDPC Decoding

Figure 3.9: Transition probabilities when stochastic decoding a \((d_v, d_c) = (2, 3)\) LDPC code for 20 DCs using \(\gamma = 1\) (conventional), 0.8, and 0.2 values.

the transmission of the all-zero codeword.

At the example of Figure 3.9, when the probability \(P_{i \rightarrow j,t}\) converges to zero, it indicates that the outgoing bit from \(v_i\) to its connected check node has very high probability to be 0 and the state of EM is stationary in State 0. Therefore, the code bit obtained from variable node \(v_i\) can be easily determined as a 0, which is agree with the transmitted code bit. The convergence of the probabilities \(P_{i \rightarrow j,t}\) and \(P_{j \rightarrow i,t}\) indicates the reliability of the outputted code bits at the \(t\)-th DC. Therefore, knowing the evolution of the probabilities \(P_{i \rightarrow j,t}\) and \(P_{j \rightarrow i,t}\) is essential to determine the number of DCs for stochastic decoder. The proposed Markov chain model can be used to predict how many DCs is required to decode the code and further estimate the decoding latency of stochastic decoding.
3.3. Transition Probabilities in Stochastic LDPC Decoding

3.3.1.1 Utilizing the Extrinsic Information to Update Channel Message

For stochastic decoding, it has been shown that efficient use of channel message to initialize the decoder helps to improve its convergence [95]. However, we can notice that the channel probability $P_i$ is a constant during the whole decoding process. If the decoder receives a biased channel message at low SNRs, the performance of decoder will be greatly affected by the biased message. To avoid this degradation, extrinsic information, i.e., messages received from the connected check nodes, is used to determine the channel incoming message.

For instance, we can let the channel incoming message be a convex combination of the original $P_i$ and the extrinsic information $P_{j\rightarrow i,t}$. If we use this scheme to update channel incoming message, the probability $P_i$ used in the Markov chain model at the $t$-th DC is replaced by

$$
\gamma P_i + (1 - \gamma) P_{j\rightarrow i,t-1} \tag{3.17}
$$

where $0 < \gamma \leq 1$.

Using (3.17) instead of the constant $P_i$ helps to increase the rate of convergence of the transition probabilities in the stochastic decoder. As shown in Figure 3.9, we select $\gamma = 0.8$ and 0.2 to decode the all-zero codeword. We observe that these $\gamma$ values let both probabilities $P_{t\rightarrow j,t}$ and $P_{j\rightarrow i,t}$ converge to zero faster than those using the constant $P_i$ value, i.e., when $\gamma = 1$, in the decoding process. Compared to the $\gamma = 1$ case, when $\gamma = 0.8$, these probabilities take 75% of the DCs to converge to zero. For the smaller $\gamma$ value, i.e., $\gamma = 0.2$, $P_{t\rightarrow j,t}$ and $P_{j\rightarrow i,t}$ only take 55% of DCs to converge to zero, but their values are relatively far from zero before the convergence. This means that utilizing the extrinsic information from check nodes to augment the channel message helps these transition probabilities converge faster in stochastic decoding. Figure 3.9 also shows that allocating more weight on the extrinsic information $P_{j\rightarrow i,t-1}$ in (3.17) will results in less decoding delay; however, $P_{i\rightarrow j,t}$ and $P_{j\rightarrow i,t}$ may be relatively off before the convergence.
3.4 Determination of Code Bits in Stochastic Decoding

The code bits in stochastic decoding are determined at the end of each DC, when each variable node outputs a bit to indicate the probability of the corresponding code bit being 1, and the decoder checks to see if the codeword satisfies the constraint $x H^T = 0$. Since it is unnecessary to accumulate all the output bits from the first DC to calculate the probability of the code bit is equal to 1, a decision mechanism is needed to determine the code bit based on the last few stochastic bits coming from each variable node. N-modular redundancy is a commonly employed fault correcting technique where the computation is replicated $N$ times and the final output is based on the majority decision of those $N$ results. For stochastic decoders, similar methods proposed in the literature including a saturating up/down counter [18] and a majority logic design [96].

In [18], the mechanism of using up/down counter in a fully parallel stochastic LDPC decoder to determine the code bits was introduced. A 4-bit saturating up/down counter that counts between -7 and 7 is used. The counter is increased by one when the stochastic bit outputted from the variable node is 1 and is decreased by one when a 0 is received. Then, the code bit of each counter is determined by the sign bit of the counter. The decoded bit is 1 when counter is positive and 0 otherwise.

Another design to make the decision on code bits is using the majority mechanism introduced in [96]. This redundancy structure makes a majority decision upon the last few bits in the output stochastic streams. For example, when more than two of the last five bits are 1s, the code bit is decided as 1 and 0 otherwise. This method helps to improve the performance of decoder by increasing its ability to against transient errors. However, this error suppression technique consumes chip area for memories and also delays the outputs, thus increasing the latency.

We extend these results to show that the latest output bit from the $i$-th variable node can be directly determined as the $i$-th code bit in the codeword. Since each stochastic bit is equal to 1 with its encoded probability and the exact value represented by the entire stochastic stream is not really critical in stochastic decoding, the redundancy mechanism can be neglected while the stochastic decoding process is convergent. Removing the de-
3.4. Determination of Code Bits in Stochastic Decoding

Table 3.2: Percentage of outputted stochastic bit values during 301-310 DCs at SNRs.

<table>
<thead>
<tr>
<th>Case</th>
<th>SNR=1.5 dB</th>
<th>SNR=2.75 dB</th>
<th>SNR=3.25 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>All 0s</td>
<td>88.04</td>
<td>99.38</td>
<td>99.70</td>
</tr>
<tr>
<td>Not Steady</td>
<td>8.43</td>
<td>0.33</td>
<td>0.08</td>
</tr>
<tr>
<td>All 1s</td>
<td>3.53</td>
<td>0.29</td>
<td>0.22</td>
</tr>
</tbody>
</table>

Table 3.3: Percentage of outputted stochastic bit values during 601-610 DCs at SNRs.

<table>
<thead>
<tr>
<th>Case</th>
<th>SNR=1.5 dB</th>
<th>SNR=2.75 dB</th>
<th>SNR=3.25 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>All 0s</td>
<td>87.27</td>
<td>99.49</td>
<td>99.72</td>
</tr>
<tr>
<td>Not Steady</td>
<td>8.87</td>
<td>0.10</td>
<td>0.03</td>
</tr>
<tr>
<td>All 1s</td>
<td>3.86</td>
<td>0.41</td>
<td>0.25</td>
</tr>
</tbody>
</table>

decision mechanism and determining code bits based on the latest outputted stochastic bit at each variable node provides a more efficient design of stochastic decoders.

This hardware-efficient technique, which directly uses the last stochastic bit as the code bit in binary codewords, provides similar error rate performance as those stochastic decoders with redundant mechanisms. We record the statistics over 10 DCs at two different decoding stages: 301-310 and 601-610 DCs. During these 10 DCs, the stochastic output bits from all variable nodes are classified into three categories: all 0s, all 1s, or not steady. The not steady case means that the outputted stochastic bits alternately change value between 0 and 1 during the recorded 10 DCs. Tables 3.2 and 3.3 show the percentages of these three cases over 10 DCs when the all-zero codeword of the (1056,528) LDPC code decoded by stochastic decoding at different SNR values.

We observe that relatively high percentages of the outputted stochastic bits are all 0s over the 10 DCs at both decoding stages at high SNRs. The case of alternate bits in the recorded 10 DCs are quite rare at all SNR values. This indicates that most of the output bits are converged at these decoding stages. Thus, we can directly use the last output bit from \( v_i \), which carries the probability \( P(x_i = 1) \), as the \( i \)-th code bit.

It needs to be mentioned that the redundancy decision mechanisms, such as up/down
3.4. Determination of Code Bits in Stochastic Decoding

Figure 3.10: BER performance of an irregular (1056,528) LDPC code decoded by stochastic decoder having up/down counters and decoder using direct codeword determination.

Figure 3.11: BER performance of a regular (2048,1723) LDPC code decoded by stochastic decoder having up/down counters and decoder using direct codeword determination.
3.4. Determination of Code Bits in Stochastic Decoding

Figure 3.12: Latency for an irregular (1056,528) LDPC code decoded by stochastic decoder having up/down counters and decoder using direct codeword determination.

Figure 3.13: Latency for a regular (2048,1723) LDPC code decoded by stochastic decoder having up/down counters and decoder using direct codeword determination.
counters, are effective at low SNRs where the output bits are less reliable. In order to ensure there is no great impact of removing the redundancy decision mechanisms on the performance of stochastic LDPC decoders, we compare the BER performance of the proposed scheme with the stochastic decoder having up/down counters. Figure 3.10 shows BER performance of an irregular (1056,528) LDPC code decoded by stochastic decoders with and without counters on an AWGN channel. The maximum numbers of DCs used for both stochastic decoding schemes are set to 150, 300, and 700. The $\pm 7$ saturating up/down counters are employed in the stochastic decoder as the redundancy decision mechanism. Figure 3.10 shows that the stochastic decoder having the simple hardware implementation, which directly uses the latest output bits as code bits, gives comparable BER performance as the one using up/down counters. When decoding other LDPC codes with different structure, e.g., the regular (2048,1723) LDPC code from the 10GBASE-T standard shown in Figure 3.11 the proposed scheme still shows no performance loss over SNRs.

We also look into the decoding latency of this efficient codeword determination scheme. Since stochastic decoding process is terminated and begins to load the next code block when reaching the maximum number of DCs or all the check nodes are satisfied, the decoding latency is determined by the average number of DCs at a particular SNR for stochastic decoder to obtain a legitimate codeword. We compare the latency of these decision mechanisms in Figures 3.12 and 3.13. These figures indicates that our efficient code bits detecting implementation does not require extra decoding latency to get a valid codeword.

The results on the BER and decoding latency shown in this section conclude that our proposed scheme reduces the complexity of hardware implementation without having any loss on decoding performance compared with the methods based on redundancy mechanism designs, e.g., the $\pm 7$ saturating up/down counters.

### 3.5 Conclusions

In this chapter, we provide a statistical study of stochastic LDPC decoding from different perspectives. First, we show the required randomization in stochastic streams and present a scaling method for stochastic decoder to properly decode the messages received...
from the binary channel models. The scaled channel message becomes independent of the channel error probability, which means that the decoder does not need to estimate the channel error probability. On the BER performance, the scaling method can provide more than 1.5-dB performance improvement over the one without scaling.

Next, we investigate the transition probabilities in stochastic decoding by using a Markov chain model for EM. This model can be employed to study the convergence conditions and the behavior of stochastic decoding process. By using the proposed Markov chain model, we also show that utilizing the extrinsic information from check nodes to help in determining the channel messages results in faster convergence in stochastic decoding.

In addition, we propose a hardware-efficient codeword determination scheme for stochastic LDPC decoders. Without using any redundancy mechanism, the latest bit from the variable node is directly used as the code bit. This scheme gives comparable BER performance and results in no throughput loss from the stochastic decoders having redundancy mechanism designs.
Chapter 4

Error Characteristics of Stochastic LDPC Decoding

In this chapter, we investigate the error characteristics of stochastic decoding and analyze the factors leading to decoding failure in stochastic LDPC decoders. In addition, we report stochastic-decoding-specific trapping sets for the (1056,528) LDPC code used in the WiMAX standard. This observation gives insight into stochastic decoding errors and provides knowledge of code structure designs that lower the error floor in stochastic decoding.

4.1 Introduction

Stochastic decoding like other iterative decoding algorithms tends to suffer the error floor phenomenon. An error floor is an abrupt flattening in the error rate performance at high SNRs. The error floor is attributed to certain sub-graphs in the factor graph of the LDPC code. Therefore, code structures must be identified to improve the error floor behavior. Depending on the decoding algorithm and the channel model, these sub-graphs have different names. For the BEC, the problematic structures are known as stopping sets [97]. For the BSC and the AWGN channel, the structures that cause error floors are called trapping sets [98], near-codewords [99] or pseudo-codewords [100]. In the dissertation, we call them trapping sets for simplicity.

It is well known that the performance of a code depends on the structure of its factor graph, including short cycles and trapping sets. A judicious choice of a code graph realization is crucial to the performance of iterative decoders. Hence, knowing the trapping sets
can help us design a code graph that selectively avoids these structures to lower the error floor. Many attempts have been made to alleviate the error floor problem by designing LDPC codes which exclude the structures that lead to decoding failure [101, 102, 103]. In addition to assist code designs, knowledge of trapping sets helps to improve performance in iterative decoding [104, 105, 106]. For example, after identifying the trapping errors, the post-processing algorithm in [105] performs the process of modifying the variable nodes inside the trapping set and running further iterations to eliminate the trapping errors. Other proposed approaches, like adding linearly dependent rows to augment the $H$ matrix in order to resolve decoding failure [107, 108], also need to identify the trapping sets in advance.

It should be noted that even for the same code, different decoding algorithms could have different error patterns. Although many studies have been carried out to lower the error floor of traditional iterative decoding schemes, for stochastic decoding such studies have not been done. Understanding error patterns is the first and crucial step to lower the error floor in stochastic decoding.

4.2 Structures Leading to Decoding Failure

Short cycles within the factor graph create sub-graphs which are harmful to iterative decoding. These subsets of variable nodes and their neighboring check nodes form a structure that results in an error floor at high SNR values. At the initial stage of iterative decoding, variable nodes send messages with low reliability to the neighboring check nodes; this is due to the presence of erroneous channel received messages. The messages sent from these nodes propagate to other variable nodes in the sub-graph. After certain decoding rounds (iterations, or decoding cycles, depending on the iterative decoding algorithm used), the external variable nodes start adjusting the incorrect values toward the correct values. However, the variable nodes inside the sub-graph may be already significantly biased toward the incorrect values because such variable nodes have low-reliability channel message and there are very few neighboring check nodes capable of detecting the errors within the sub-graph. The unreliable messages remain uncorrected until the end of the decoding process in this sub-graph. This structure results in decoding failure for iterative decoders.
4.2. Structures Leading to Decoding Failure

A. Stopping Sets

Stopping set [97] is the notion used to describe the structures leading to decoding failure of iterative decoding for LDPC codes over the BEC. A stopping set is closely defined as a minimum subset of variable nodes in which all the neighboring check nodes are connected to the subset of variable nodes at least twice [97]. Stopping sets ensure existence of cycles in the sub-graph. The space of stopping sets is closed under union. In other words, if sub-graphs $S_1$ and $S_2$ are both stopping sets, so is $(S_1 \cup S_2)$. Figure 4.1 shows an example of a size-three stopping set, in which contains three variable nodes.

Stopping sets are well defined, i.e., we can locate all stopping sets by exhaustive search in the factor graph of the code. Although stopping sets are very useful for determining the performance over the BEC, stopping sets cannot be used directly to determine LDPC performance for other channels, such as the BSC or the AWGN channels. It is because the errors in non-erasure channels are more subtle [109].

B. Trapping Sets

Richardson in [98] introduced the notion of trapping sets to describe the code structures leading to decoding failure for general channels. Assuming that the all-zero codeword is transmitted over a channel and the given received sequence is $y$, let $C^t_i(y) \in \{0, 1\}$ denote the $i$-th output code bit at the $t$-th round of decoding. The $i$-th bit is eventually correct if there exists a $T$ such that for all $t \geq T$, $C^t_i(y) = 0$ for all $i$. The sub-graph contains variable nodes that eventually outputs incorrect bit values is denoted as $T(y)$, and is called a trapping set. If there are $\alpha$ variable nodes in the trapping set with exactly $\beta$ check nodes connected to the subset of variable nodes with odd-degree connection, the trapping set is denoted by $T(\alpha, \beta)$. While all the check nodes in the trapping set are of degree-1 or
4.2. Structures Leading to Decoding Failure

Figure 4.2: BER performance of the (1056,528) LDPC code decoded by the SPA, the MSA, and the stochastic decoding algorithm on the AWGN channel.

degree-2 connections, the trapping set is known as an elementary set [110].

Small trapping sets are more harmful than large trapping sets, because they occur more frequently, especially when $\beta \ll \alpha$. In addition, the structure of trapping sets also affects the performance of iterative decoders. Dolecek et al. in [111] have demonstrated that absorbing sets, which are defined as subsets of trapping sets with more check nodes with even-degree than the check nodes having odd-degree connection to the variable nodes in the sub-graph, dominate the error floor performance and have shown that most absorbing sets appear to be elementary sets.

Although stochastic decoding has advantages in terms of hardware implementation, it suffers from the errors that are not presented in other iterative decoding algorithms, such as the SPA and the MSA. The BER curves of an LDPC code decoded by the SPA, the MSA,
and stochastic decoding algorithm are shown in Figure 4.2. Comparing with the sum-product decoding, stochastic decoding has a performance loss of approximately 0.5 dB and a higher error floor which is due to the existence of erroneous outputs that are specific to stochastic decoding. Furthermore, we observe that the errors in these iterative decoding methods are characterized as a failure to converge to a legitimate codeword, and thus the errors are detectable. These detectable errors are the most common decoding failure in belief propagation decoding of LDPC codes [106].

4.3 Stochastic Decoding Specific Trapping Set

We study the (1056,528) LDPC code from the WiMAX standard in our simulations, where we transmit the codeword over an AWGN channel using BPSK modulation and then decode the received signals by both SPA and stochastic decoding algorithms. The maximum number of iterations for the SPA is set to 32, and the maximum number of DCs used for the stochastic decoding is selected to be 700. From our simulation results, we determine the stochastic-decoding-specific trapping sets at SNR of 3.5 dB, which is within the error floor region of this code. By the stochastic-decoding-specific trapping set, we mean that the same noise sequence does not result in an erroneous output for the SPA. The variable nodes in these trapping sets are listed in Table 4.1. The size of these trapping sets is 10. Because the code under study is a quasi-cyclic LDPC code and its parity-check matrix \( H \) is defined by a set of 44-by-44 circulant matrices, the code structures will be the same if we shift the variable nodes inside the permutation matrix range. For example, the variable nodes \{1, 157, 395, 590, 634, 674, 678, 964, 1008, 1038\} form a trapping set and so does these variable nodes with a shift.

There are 3 degree-3 and 7 degree-2 variable nodes inside the trapping set. There are three cycles in this subset of nodes. The lengths of these cycles are 12, 14, and 18. Figure 4.3 illustrates the sub-graph of this size-10 trapping set. We use \( \bullet \) to represent the degree-3 variable node and \( \bigcirc \) to represent the degree-2 variable node. Check nodes neighboring this subset of variable nodes with odd-degree connection are represented by \( \blacksquare \), and \( \square \) for those with even-degree connection. Figure 4.3 also shows that this trapping set is an elementary
4.4 Analysis of Stochastic Decoding Errors

Since stochastic decoding is a nondeterministic process, in some trials, it succeeds in decoding the same noise sequence that result in decoding failure in other trials. In other words, decoding outcome will not be the same even with the same noise sequence in different trials. Based on our numerical results, approximately 90% of the trials give the same errors when stochastic decoding the noise sequence which results in decoding failure. No error is observed in the rest 10% of trials. In cases where a decoding error occurs, excepting those stochastic-decoding-specific trapping errors, the stochastic decoding algorithm typically generates the same error locations generated by the other decoding algorithms, plus few extra errors. The stochastic-decoding-specific trapping errors are quite rare. We also observe that the variable nodes with lower degrees are prone to have erroneous outputs in stochastic decoding. For example, it is more probable to observe an error at the degree-2 variable nodes in this length-1056 LDPC code.

The stochastic-decoding-specific trapping sets are not unique to the AWGN channel, they also occur in other channel models like the BSC. Although, the BSC can be treated
4.5 Conclusions

Table 4.1: 44 multiplicities of the size-10 stochastic-decoding-specific trapping sets in the irregular (1056,528) LDPC code.

<table>
<thead>
<tr>
<th>Location of the variable nodes, $v_i, i \in [1, 1056]$, inside the trapping set</th>
<th>Bitwise shift value ($\delta$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>${1, 157, 395, 590, 634, 674, 678, 964, 1008, 1038}$</td>
<td>$\delta = 0, 1$</td>
</tr>
<tr>
<td>${3, 159, 353, 592, 636, 676, 680, 966, 1010, 1040}$</td>
<td>$\delta = 0, 1, 2$</td>
</tr>
<tr>
<td>${6, 162, 356, 595, 639, 679, 683, 925, 969, 1043}$</td>
<td>$\delta = 0, 1, \ldots, 13$</td>
</tr>
<tr>
<td>${20, 176, 370, 609, 653, 693, 697, 939, 983, 1013}$</td>
<td>$\delta = 0$</td>
</tr>
<tr>
<td>${21, 133, 371, 610, 654, 694, 698, 940, 984, 1014}$</td>
<td>$\delta = 0, 1, \ldots, 6$</td>
</tr>
<tr>
<td>${28, 140, 378, 573, 617, 701, 661, 947, 991, 1021}$</td>
<td>$\delta = 0, 1, 2, 3$</td>
</tr>
<tr>
<td>${32, 144, 382, 577, 621, 661, 665, 951, 995, 1025}$</td>
<td>$\delta = 0, 1, \ldots, 12$</td>
</tr>
</tbody>
</table>

as the AWGN channel with a hard-limiter for stochastic decoders [21], the appearance of the same stochastic-decoding-specific trapping sets implies that the trapping errors in stochastic decoding are imputed to the noise pattern than the signal-to-noise ratio.

The stochastic-decoding-specific trapping sets can be attributed to factors, such as noise sequences from the channel, randomness in stochastic decoding process, and structure of the code. The structure of the code, e.g., the existence of short cycles in the Tanner graph or the sub-graph containing stochastic-decoding-specific trapping set, is a factor that we can manipulate to prevent the trapping errors and lower the error floor in stochastic decoding.

4.5 Conclusions

We studied the error characteristics of stochastic decoding and the factors leading to decoding failure in stochastic LDPC decoders. In addition, we determined stochastic-decoding-specific trapping sets, which do not occur in other iterative decoding algorithms even when the same noise sequence occurs. The stochastic-decoding-specific trapping set found in the (1056,528) LDPC code from the WiMAX standard is an elementary absorbing
4.5. Conclusions

\( T(10, 1) \) set.

We found that the randomness in stochastic decoding, even with the same noise sequence, can be utilized to improve the BER performance of stochastic decoder. An approach to achieve lower error rate is stochastic decoding the same noise sequence repeatedly and determining the final codeword based on the outputs of each trials. For example, we can use multiple stochastic decoders to decode the same noise sequence parallelly and select the output which has the least error as the final codeword.

In the aspect of code design, selectively avoiding the structures leading to decoding failure helps lower the error floor. Therefore, obtaining the knowledge of stochastic-decoding-specific trapping sets gives guidelines to design a code graph having lower error floor in stochastic decoding. Methods such as breaking the cycles in the trapping sets or using different code graph realizations are possible solutions to the error floor of stochastic decoding.
Chapter 5

A Reduced-Latency Stochastic Decoding Algorithm for LDPC Codes

In conventional stochastic decoding, transformation from channel message to stochastic bits is done by a stochastic number generator using channel received probabilities. We propose a reduced-latency approach that uses some of reliable messages from check nodes to augment channel inputs when calculating variable node outputs. This proposed algorithm improves the error rate performance and reduces the decoding latency. In this chapter, we characterize the performance of the proposed algorithm on the AWGN channel and present a hybrid decoding scheme which uses the proposed stochastic decoding algorithm to further improve its performance.

5.1 Conditional Stochastic Decoding Algorithm

One of the factors that affect the performance of stochastic decoding is the bit patterns of stochastic streams, where streams even with the same encoded probability but different representations could result in different error rate performance. An efficient method that can generate stochastic bits adaptively is crucial to the performance of stochastic decoders. We propose an algorithm, called Conditional Stochastic Decoding (CSD), to generates stochastic bits adaptively during the decoding process. In the CSD, the stochastic bits coming from the channel are not only based on the channel message but also the messages from the connected check nodes.

Except for the stochastic bits from the channel, the decoding process in the CSD algorithm remains similar to the conventional stochastic decoding algorithm. In the CSD,
5.1. **Conditional Stochastic Decoding Algorithm**

**Algorithm 1** Conditional Stochastic Decoding

**Initialization**

1: For all $i$, variable node $v_i$ transforms $P_i$ to stochastic bit $B_{i,0}$

2: $v_i$ sends $B_{i,0}$ to its connected check nodes

**Iterations**

3: Check node computes 1-bit message using (2.20) and sends it back to variable node

4: Variable nodes generate code bits, and then perform the codeword test: If $xH^T = 0$, terminate the decoding process. Otherwise, go to 5

5: Each variable node $v_i$ determines the next channel bit

   If (all connected check nodes are satisfied)

   $B_{i,t} =$ majority of all extrinsic bits; in case of a tie, a stochastic bit based on $P_i$ is generated

   Else

   Generate $B_{i,t}$ corresponding to $P_i$ (conventional stochastic decoding)

6: Variable node computes 1-bit message using (2.19) and sends it to the connected check node

7: Repeat 3 through 6 until reaching the maximum number of DCs

when all connected check nodes of variable node $v_i$ are satisfied, the next channel incoming bit at $v_i$ is set to the value of the majority of bits received from its connected check nodes. When there is a tie between the numbers of 1s and 0s from the satisfied check nodes, or when at least one of its connected check nodes is not satisfied, the next channel incoming bit at $v_i$ will be generated based on channel probability $p_i$ by employing the conventional stochastic number generator. Therefore, when all the extrinsic bits are from satisfied check nodes, the variable node $v_i$ will use these relatively reliable bits, instead of a randomly generated bit from a stochastic number generator, to represent the channel message at the
next DC.

The CSD is summarized in Algorithm 1, where the channel incoming bit at variable node \( v_i \) at the \( t \)-th DC is denoted as \( B_{i,t} \), for \( 1 \leq i \leq n \).

Note that variable nodes with higher degree tend to have more reliable bit values because these nodes are involved in more parity-check equations. However, these nodes require more incoming bits to be in agreement, leading to less frequent updates. Using extrinsic bits from satisfied check nodes to override the stochastic bit from the channel can help alleviate this problem.

5.2 Analysis and Architecture of the CSD Algorithm

Since the incoming and outgoing bits of a satisfied check node on the same edge of the Tanner graph are equal and the bit values are flipped for an unsatisfied node, the extrinsic bit from a satisfied check node will be equal to the one sent out from \( v_i \) through the same edge. This outgoing bit from variable node \( v_i \) has two possible sources: the channel bit when there is agreement among the bits received at \( v_i \), or the bit randomly selected from memory, e.g., EM or TFM, if \( v_i \) is in a hold state. Therefore, the probability that the extrinsic bit from a satisfied check node is equal to 1 is a convex combination of the channel probability and the probabilities stored in memories. This indicates that the bit generating approach in the CSD utilizes both the channel probability \( P_i \) and the probabilities stored in the memories at \( v_i \) to ensure progress during the decoding process.

Furthermore, the stochastic bit generating approach in the CSD lowers the weight of \( P_i \) that could diminish the influence of highly biased channel received values. These factors let the CSD have more reliable channel incoming bits than merely using \( P_i \) and thus results in improved decoding performance.

A. Efficient Architecture of the CSD

Here we provide an efficient hardware design for the operation of the CSD. Since the message indicating the parity-check status of the check node is needed in the CSD, an efficient realization to pass this information is essential for maintaining the hardware-simplicity of the stochastic decoder. An intuitive way to pass the status of check node is
adding an extra wire parallel to the existing one. This approach is the simplest but costs lots of area and results in more complicated interconnection network between nodes. If we pass the parity-check status serially with the extrinsic message from the check node back to the variable node on a single wire, no extra wire is required in the implementation; however, the decoding latency will be double because it takes two clock cycles to complete one DC. On the other hand, if we limit the available number of clock cycles for stochastic decoder using this serial-passing approach, e.g., cutting the number of clock cycles in half, its error rate performance will be worse due to the lack of available DCs.

By using the fact that the bit received from a satisfied check node will be equal to the one sent out from \( v_i \) through the same edge of the Tanner graph, we compare these incoming and outgoing bits to recognize the parity-check status of the check node. If the extrinsic bit \( B_{j \rightarrow s_i, t} \) from check node \( c_j \) is the same as the bit \( B_{i \rightarrow j, t} \), we know that \( c_j \) is a satisfied check node. The comparison between the bits \( B_{i \rightarrow j, t} \) and \( B_{j \rightarrow s_i, t} \) can be realized simply by an XOR gate.

Besides, the mechanism for storing bit information already exists on each edge of variable node, such as EMs or TFMs. These memories are employed to alleviate the latching problem, and the size of these memories corresponds to the degree of variable node. The higher degree variable node requires longer memory size to provide the decorrelation in stochastic streams. For example, the size of EM is practically chosen as 64 for degree-6 nodes and only 32 for degree-2 nodes [18, 84]. Therefore, adding 1-bit memory to store the outgoing bit for later comparison at variable node will not significantly impact the hardware complexity.

In practice, we may not even require the extra 1-bit memory to store the bit for the comparison, since the information of the outgoing bit from variable node already exists in the EM or TFM. When the variable node is in the hold state, the outgoing bit is the bit randomly picked from the memory at the edge and known at the variable node. If the variable node is not in the hold state, i.e., there is an agreement among the received bits at the variable node, the latest updated bit in the edge-mounted memory is the outgoing bit. Hence, the bits stored in the EM or TFM can be used to detect the parity-check status of the connected check node without extra memory space and wiring, which can cause clock propagation delay and routing congestion.
5.3. The CSD Algorithm on the AWGN Channel

To demonstrate the decoding performance of the CSD, we have studied two practical LDPC codes: an irregular (1056,528) code from the WiMAX standard and a regular (2048,1723) code from the 10GBASE-T standard. The degrees of variable and check nodes in the irregular (1056,528) code are \( d_v = \{2, 3, 6\} \) and \( d_c = \{6, 7\} \), respectively. The node degrees of the regular (2048,1723) LDPC code are \( d_v = 6 \) and \( d_c = 32 \).

The codewords are transmitted over an AWGN channel using BPSK modulation. We compare the performance of the CSD with the EM-based stochastic decoder [18]. The lengths of the edge memories in these stochastic decoders are set to 32, 48, and 64 for
5.3. The CSD Algorithm on the AWGN Channel

degree-2, 3, and 6 variable nodes, respectively. The latency of these decoders are computed by the average DCs needed to get a legitimate codeword at a particular SNR.

5.3.1 Error Rate Performance

In comparing error rate performance, we let both the EM-based and the CSD decoders have a maximum latency of 200 DCs. The BER performance and the decoding latency of the (1056,528) LDPC code are shown in Figure 5.2, which indicates that the CSD algorithm can achieve lower BERs with less latency. For example, at SNR=3.75 dB, the CSD reduces the BER by a factor of 2 while reducing the latency by 35% of the EM-based stochastic decoder. The performance of the (2048,1723) LDPC code is shown in Figure 5.3. The improvement of the CSD is relatively smaller in decoding codes with higher degree nodes. This is expected since for higher degree nodes, the probability of all connected check nodes being satisfied is lower, which results in decreased impact of the CSD approach.

The CSD algorithm also gives lower Frame Error Rate (FER) performance, as shown in Figure 5.4 for the (1056,528) LDPC code. The CSD provides more than 0.5-dB improvement at the maximum DC of 150. The FER of the CSD decoder with 150 maximum DCs can be even lower than that of the EM-based stochastic decoder with 200 DCs. As stated before, for codes with high degree nodes, e.g., the (2048,1723) LDPC code shown in Figure 5.5, the CSD shows less improvement in performance.

5.3.2 Decoding Latency

A major advantage of the CSD is its low decoding latency. The expected number of DCs of the CSD could be less than 75% of the EM-based stochastic decoder for various maximum numbers of DCs. For the (1056,528) LDPC code shown in Figure 5.6, the latency of the CSD converges to a lower value, only 65% of the EM-based stochastic decoder, as the SNR increases. It should be noted that compared to the EM-based stochastic decoder, the CSD, even with larger maximum number of DCs, can have reduced decoding latency. For instance, when the SNR value is greater than 2.5 dB, the CSD with more than 200 maximum DCs has less decoding delay than the EM-based stochastic decoder with 150
5.3. The CSD Algorithm on the AWGN Channel

Figure 5.2: BER performance and decoding latency of the irregular (1056,528) LDPC code decoded by the EM-based stochastic decoder and the CSD decoder.

Figure 5.3: BER performance and decoding latency of the regular (2048,1723) LDPC code decoded by the EM-based stochastic decoder and the CSD decoder.
5.3. The CSD Algorithm on the AWGN Channel

Figure 5.4: FER performance of the irregular (1056,528) LDPC code decoded by the EM-based stochastic decoder and the CSD decoder.

Figure 5.5: FER performance of the regular (2048,1723) LDPC code decoded by the EM-based stochastic decoder and the CSD decoder.
5.3. The CSD Algorithm on the AWGN Channel

Figure 5.6: Average number of DCs for the irregular (1056,528) LDPC code decoded by the EM-based stochastic decoder and the CSD decoder.

Figure 5.7: Average number of DCs for the regular (2048,1723) LDPC code decoded by the EM-based stochastic decoder and the CSD decoder.
maximum DCs. The latency of decoding the (2048, 1723) LDPC code is shown in Figure 5.7, where the CSD also requires fewer DCs and converges to a value that is 15% smaller than the EM-based case.

Simulation results of the CSD are consistent with what we found in Section 3.3.1.1, by utilizing the extrinsic information from the connected check nodes to augment the channel input at variable nodes, where it was shown that this approach results in faster convergence in the stochastic decoding process.

5.4 Hybrid Stochastic Decoding

A two-stage hybrid decoding scheme is used to improve the error rate performance or increase the throughput of the decoder [101]. In this section, we present a hybrid decoding scheme combining the proposed CSD algorithm with the EM-based stochastic decoding to improve the performance of the decoder. By changing DCs, we study the optimal tradeoff between error rates and decoding latency. Note that all the computational operations in the entire hybrid stochastic decoding scheme are still in a bit-wise fashion, the complexity in hardware implementation of the hybrid stochastic decoder will not increase significantly.

5.4.1 The CSD Algorithm in Hybrid Decoding

The proposed hybrid decoder is a cascade of the CSD and the EM-based stochastic decoding algorithms. In this two-stage approach, the received message is decoded by the CSD in Stage-I for having reduced decoding latency. When the CSD fails to decode the received message, the EM-based stochastic decoding is employed to decode the same received message in Stage-II.

As long as the error rate of the CSD in Stage-I is sufficiently low, the extra decoding rounds needed in Stage-II will have little impact on the overall decoding delay. Therefore, when we distribute the DCs properly between two stages, the combination of the CSD followed by the EM-based stochastic decoding will gain error rate improvement with just a little extra delay.
We simulate the hybrid stochastic decoding scheme with maximum total 500 DCs to decode the (1056,528) LDPC code. We have 3 different distributions of these 500 DCs between two stages: 100 DCs for Stage-I and 400 DCs for Stage-II, 200 DCs for Stage-I and 300 DCs for Stage-II, and 300 DCs and 200 DCs for each stage, respectively. Based on our simulation results, no undetectable error is observed at the end of Stage-I while decoding the length-1056 LDPC code. Therefore, the error rates of the hybrid stochastic decoding scheme are computed by the erroneous outputs of the second stage.

5.4.1.1 Decoding Performance of Hybrid Stochastic Decoding

The BER performance of the two-stage hybrid stochastic decoding scheme is shown in Figure 5.8. We note that compared with the EM-based stochastic decoder, the direct CSD shows approximately 0.5-dB loss when the maximum number of DCs is 500. It is because the error-correcting ability of the CSD does not increase as much as that of the EM-based decoder while a large number of DCs is available. However, as shown in Figure 5.8, this weakness of the CSD can be resolved by employing the proposed two-stage hybrid decoding. Figure 5.8 also illustrates that BERs of the CSD followed by the EM-based decoder converge to the error rate of an EM-based decoder whose BER is the lowest with 500 DCs.

Decoding latency, i.e., the average number of DCs, for the hybrid stochastic decoding is affected by the distribution of DCs. As shown in Figure 5.9, if more DCs are available for the second stage decoding, i.e., if fewer DCs are allocated to Stage-I, the total latency will increase because most of the time decoding in Stage-II is required, especially at low SNRs. On the other hand, allocating more DCs to Stage-I makes the decoding latency converge to that of the direct CSD, where the delay is the least. A proper way to distribute the total available DCs for the two-stage scheme is to allocate a slightly larger than the expected number of DCs to Stage-I decoding.

From Figure 5.8 and 5.9, we see that when the available number of DCs is large, the two-stage hybrid stochastic decoding provides BER performance as low as the conventional stochastic decoder and while keeping the latency as low as the CSD. This hybrid stochastic decoding scheme resolves the BER loss problem for the CSD while a large number of DCs
5.4. Hybrid Stochastic Decoding

Figure 5.8: BER performance of the (1056,528) LDPC code decoded by the two-stage hybrid stochastic decoding.

Figure 5.9: Average number of DCs for the (1056,528) LDPC code decoded by the two-stage hybrid stochastic decoding.
5.5 Conclusions

In this chapter, a conditional stochastic decoding algorithm, which makes use of reliable messages from check nodes to calculate channel stochastic bits during stochastic LDPC decoding, is presented. This approach not only diminishes the influence of biased channel received values, but also results in improved error rate performance and more than 30% latency reduction in stochastic LDPC decoding. The performance improvement of the CSD is more obvious while using codes with more low-degree variable nodes, which are usually prone to producing erroneous outputs. The improved performance and hardware-efficient design make the CSD algorithm a better choice than other stochastic decoders for high-throughput communications systems.

In addition, we present a hybrid stochastic decoding scheme, in which the CSD is followed by the EM-based stochastic decoding. The proposed hybrid scheme can provide improved error rates and remain the low decoding latency for the case that a large number of DCs is available. We also provide a tradeoff between error rate performance and latency for the hybrid stochastic decoding scheme by distributing DCs between two decoding stages.
Chapter 6

Stochastic Decoding of LDPC Codes in ARQ Scheme

In this chapter, we investigate the performance of stochastic decoders in an ARQ scheme; an error-control method used to enhance communications reliability. We show that, besides efficient hardware implementation, stochastic decoding provides improved error rate performance and requires fewer retransmissions than other iterative decoding algorithms when used in Hybrid Automatic Repeat reQuest (HARQ) systems. We analyze the advantages of using stochastic decoding and present tradeoffs between decoding performance and transmission energy cost for HARQ systems using stochastic decoding.

6.1 Hybrid ARQ Schemes

HARQ coding has become an integral part of emerging standards, such as Long-Term Evolution (LTE) [112] and WiMAX (IEEE 802.16e) [5], to enhance service reliability [113]. HARQ is a combination of Forward Error Correction (FEC) and ARQ error-control method. The error rates and latency of an HARQ system is determined by the performance of the decoding algorithm. In this chapter, we show that stochastic decoding can provide lower error rate performance with fewer retransmissions than other iterative decoding algorithms in an HARQ system. We analyze and compare the error rate performance of the SPA, the MSA, and the stochastic decoding algorithm to show that stochastic decoding is the most suitable choice to be used in ARQ schemes.

We consider LDPC codes as the FEC code in the HARQ system and adopt the type-I ARQ scheme [114], where a codeword is transmitted initially and if the receiver fails to de-
code the codeword, a retransmission is requested by sending a Negative-Acknowledgement (NACK) message back to the transmitter. Upon receiving a NACK, the transmitter sends the same codeword again. After an Acknowledgement (ACK) message is received, or the maximum number of retransmissions is exceeded, the next codeword is transmitted. This *stop-and-wait* method is the simplest ARQ scheme; however, it is less efficient than more complicated schemes due to the time-out delay between each transmission. If the application requires higher channel throughput, other types of ARQ, such as *go-back-N* and *selective repeat* are more suitable. Since the performance comparison in this chapter focuses on the decoding algorithms in the ARQ scheme, the type-I stop-and-wait method is relatively easy to analyze.

We use HARQ-$N$ to denote an HARQ system with $N$ maximum transmissions for a single codeword, where the maximum number of retransmissions is $(N - 1)$. The transmitter sends out the next codeword when it receives an ACK message or after receiving $N$ NACK messages from the receiver. We assume that a noise-free feedback channel is available.

We employ iterative decoding to decode received messages, and the error rates of the ARQ scheme are computed based on the output of the final decoding attempt. In our simulations, we have observed that all errors are due to failure to converge to a valid codeword, rather than converging to a wrong but legitimate codeword. These detectable errors are the most common decoding failure in belief propagation decoding of LDPC codes [106]. Therefore, the BER of an HARQ-$N$ system is computed as

$$
\frac{P_{b,N}}{N - 1} \prod_{n=1}^{N-1} P_{f,n}
$$

(6.1)

where $P_{b,N}$ is the BER of the $N$-th decoding attempt, and $P_{f,n}$ is the FER of the $n$-th decoding attempt. The overall FER of an HARQ-$N$ system is

$$
\prod_{n=1}^{N} P_{f,n}
$$

(6.2)

Moreover, for the HARQ system of $N$ maximum transmissions for each codeword, the average number of transmissions per codeword is given by
6.2. Comparison with Other Iterative Decoding Algorithms

\[ 1 + \sum_{n=1}^{N-1} \left( \prod_{k=1}^{n} P_{f,k} \right) \] (6.3)

From (6.1), (6.2), and (6.3), we have that lower value of \( P_{f,n} \), \( 1 \leq n \leq N \), not only results in better error rate performance, but also reduces the required transmissions in the HARQ system and further saves retransmitting energy.

Since the receiver is capable of buffering previously received coded packets and the hardware complexity of adding them together is low, we employ the Chase Combining (CC) \[ [115] \] method in the HARQ system. Multiple received packets are combined according to the maximum ratio combining principle to improve the quality of the involved packets through gains in the SNR. Unless a packet is detected correctly in the first transmission, the retransmitted packets, after reception, are combined with previously received packets before being sent to the decoder. When the energy of each transmission is constant, using the Chase combining, the \( E_b/N_0 \) value at the \( n \)-th retransmission, for \( 1 \leq n < N \), will be \((n + 1)\) times the \( E_b/N_0 \) in a single transmission case.

6.2 Comparison with Other Iterative Decoding Algorithms

Equation (6.1) indicates that among the FEC decoders that have similar BER performance at a particular SNR, the one having lower FER will result in better error rate performance in the ARQ scheme. In order to understand which iterative LDPC decoding algorithm has the lowest FER at a given BER, we compare the error rate performance of the SPA, the MSA, and the stochastic decoding algorithms. To let these iterative decoding algorithms have similar BERs over SNRs, we need to carefully select the number of decoding rounds for each algorithm. We set the maximum numbers of iterations for the SPA and the MSA to 8 and 9, respectively, and the maximum number of DCs used for the stochastic decoding be 300. It should be noted that it is impossible to have exactly the same BERs for a wide range of SNR values in all iterative decoding algorithms studied here. Using these selected values, the three decoding algorithms yield similar BERs. Furthermore, the selected values of decoding rounds are quite reasonable in practice. We can improve the
error rate performance of these iterative decoding algorithms by allowing more iterations or decoding cycles.

Two different types of LDPC codes, the irregular (1056,528) code from the WiMAX standard and a regular (816,408) LDPC code generated by MacKay [116], are considered in our simulations to demonstrate the results. Figure 6.1 illustrates the FER values of these two LDPC codes decoded by iterative decoding algorithms over BERs. Compared with the SPA and the MSA, stochastic decoding gives the lowest FER for these BER values. Therefore, stochastic decoding is the best choice to be used in an HARQ scheme because it provides the lowest FER for both LDPC codes under study. This implies that stochastic decoding has the largest number of error bits in each erroneous frame than the other two iterative decoding algorithms, as shown in Figure 6.2. In cases where a decoding error oc-
6.2. Comparison with Other Iterative Decoding Algorithms

curs, the stochastic decoding algorithm typically generates the same error locations generated by the other decoding algorithm, plus some extra errors. Furthermore, since stochastic decoding is a nondeterministic process, in some cases, it succeeds in decoding certain noise sequences that cannot be decoded by other iterative decoding algorithms. These two facts result in lower FERs for stochastic decoding and more error bits per frame, when an error frame occurs. Regardless of these facts, as mentioned before, all the three algorithms have very close BER values. These are the reasons why stochastic decoding has relatively lower FER, and thus is a better candidate to be used in HARQ systems.

Figure 6.2: Average number of error bits in an erroneous frame of LDPC codes decoded by the SPA, the MSA, and the stochastic decoding algorithm.
6.3 Performance of HARQ Using Stochastic Decoding

For simplicity, we assume $N=2$ in our simulations of ARQ scheme. Extension to other values of $N$ is straightforward and simple. We study the performance of the HARQ-2 system under two different scenarios: each transmission having the same transmitting energy $E_T$, and energy in each transmission being adjustable.

Besides transmission energy $E_T$, distribution of DCs among the decoding attempts is another factor that affects the performance of HARQ. We can let all decoding attempts have the equal number of DCs or we can allocate different amount of DCs to each decoding attempt. We use $D_{1,max}$ and $D_{2,max}$ to denote the maximum number of DCs in each decoding attempt. The total DCs available for both decoding attempts is $(D_{1,max} + D_{2,max})$.

6.3.1 Constant Transmission Energy

First, we investigate the performance of the ARQ scheme using the constant transmitting energy for each transmission. BER performance and decoding latency in this case are given below.

A. Bit Error Rate

BER performance of stochastic decoding in the HARQ-2 systems having constant transmission energy is shown in Table 6.1. It shows that BER decreases as we allocate more DCs to the second decoding attempt. This observation is consistent with (6.1) that the BER of the HARQ-$N$ system becomes lower as the value of $P_{b,N}$ decreases. Therefore, we should allocate more DCs for the second decoding attempt in the HARQ-2 system to obtain a low $P_{b,2}$ value, which results in lower overall BER of the ARQ scheme.

B. Decoding Latency

The decoding latency of the HARQ system, in terms of the overall expected number of DCs, is computed as

$$
\sum_{n=1}^{N} \left( d_n \prod_{k \geq 1} p_{f,k} \right)
$$

(6.4)
Table 6.1: BER of the (1056,528) LDPC code decoded by stochastic decoding in the HARQ-2 system with total 600 DCs available.

<table>
<thead>
<tr>
<th>DCs distribution $(D_{1,max}/D_{2,max})$</th>
<th>$E_b/N_0 = 1.0 , dB$</th>
<th>$E_b/N_0 = 1.5 , dB$</th>
</tr>
</thead>
<tbody>
<tr>
<td>250/350</td>
<td>$1.4796 \times 10^{-7}$</td>
<td>$5.9186 \times 10^{-9}$</td>
</tr>
<tr>
<td>350/250</td>
<td>$3.5296 \times 10^{-7}$</td>
<td>$9.4697 \times 10^{-8}$</td>
</tr>
<tr>
<td>450/150</td>
<td>$9.7538 \times 10^{-6}$</td>
<td>$9.9001 \times 10^{-7}$</td>
</tr>
<tr>
<td>550/50</td>
<td>0.0045</td>
<td>7.4943 $\times 10^{-4}$</td>
</tr>
<tr>
<td>600, No-ARQ</td>
<td>0.1090</td>
<td>0.0360</td>
</tr>
</tbody>
</table>

where $d_n$ denotes the average DCs at the $n$-th decoding attempt. The value of $d_n$ depends on the SNR and the maximum number of DCs allocated at the $n$-th attempt, $D_{n,max}$. Equation (6.4) also indicates that the lower FER will result in less decoding latency of the HARQ system.

Reducing the decoding latency, i.e., minimizing the overall required number of DCs in the HARQ system, is crucial for communications requiring high-throughput. Figure 6.3 illustrates the overall required DCs of the HARQ-2 system with total 600 DCs available, where $D_{1,max} + D_{2,max} = 600$, and shows that large $D_{2,max}$ does not necessarily lead to long decoding delay. This is due to the SNR improvement gained from Chase combining, which results in considerable reduction in the required number of DCs at the second decoding attempt. Typically the number of required DCs in the second decoding attempt is notably smaller than $D_{2,max}$, resulting in low overall decoding latency in spite of large value of $D_{2,max}$. Note that allocating large value to $D_{1,max}$, e.g., the 550/50 ARQ case in Figure 6.3, may result in longer decoding delay than the no ARQ case. In fact at low SNRs, the second decoding attempt is required most of the time even with a large $D_{1,max}$ value, which leads to higher decoding latency in the HARQ-2 system.

When each transmission using the same energy, selecting

$$D_{i,max} < D_{j,max}$$  \hspace{1cm} (6.5)
6.3. Performance of HARQ Using Stochastic Decoding

Figure 6.3: Overall required number of DCs for the (1056,528) LDPC code decoded by stochastic decoding in the HARQ-2 system with total 600 DCs available.

for $1 \leq i < j \leq N$ would result in improved performance. This means that $D_{1,max}$ should be smaller than $D_{2,max}$ in the HARQ-2 system. Having less DCs for the first decoding attempt may increase $P_{f,1}$. However, ARQ schemes are usually applied at very low SNR region, where the $P_{f,1}$ value most of the time is close to 1 and thus the performance loss of having small $D_{1,max}$ is negligible.

6.3.1.1 Using CSD in ARQ Schemes with Constant $E_T$

From Equations (6.1)-(6.4) and our study, we observe that stochastic decoding is a better choice than other iterative decoding algorithms to be adopted in ARQ schemes. Compared to the EM-based stochastic decoder, CSD is an even more suitable candidate to be used in HARQ systems due to its relatively lower FER performance.
6.3. Performance of HARQ Using Stochastic Decoding

BER performance of the HARQ systems using CSD and EM-based stochastic decoding with total 300 DCs is shown in Figure 6.4. This figure shows that CSD results in lower BERs in the HARQ-2 system for both DC distributions: the equal maximum 150 DCs distribution, which denoted as 150/150 DCs, and the 200/100 DC distribution. For example, CSD can provide more than 0.4-dB BER improvement for the HARQ system using 200/100 DC distribution.

The decoding latency of the HARQ system is shown in Figure 6.5, which illustrates that CSD results in a significant reduction in decoding latency. Furthermore, the expected number of DCs converges to 70% of the HARQ system using the EM-based decoder. Besides, the HARQ system using CSD can operate at a relatively smaller SNR value to have a latency less than $D_{1,\text{max}}$, i.e., most of the transmitted codewords are correctly decoded.
6.3. Performance of HARQ Using Stochastic Decoding

![Graph showing latency vs. Eb/N0 for different decoding algorithms.](image)

Figure 6.5: Average number of DCs for the (1056,528) LDPC code decoded by the EM-based stochastic decoding and the CSD in the HARQ-2 system with total 300 DCs.

From Figures 6.4 and 6.5, we observe that the HARQ system using CSD algorithm results in lower error rates and less decoding latency. We also note that allocating more DCs to the second decoding attempt of the HARQ system using the CSD results in better performance, which is consistent with (6.5).

6.3.2 Adjustable Transmission Energy

For ARQ schemes using adjustable transmission energy, i.e., the case where each transmission can have different energy level, we try to optimally allocate the total energy $E_T$ among multiple transmissions to achieve the best decoding performance. For the HARQ-2 system, we assign $\alpha E_T$, where $0 < \alpha < 1$, to the first transmission of each
6.3. Performance of HARQ Using Stochastic Decoding

codeword and the rest \((1 - \alpha)E_T\) for the retransmission if required. We let the total number of DCs for the two decoding attempts in the HARQ-2 system to be 600, where we have 100/500, 200/400, 300/300, 400/200, and 500/100 DC distributions.

A. Bit Error Rate

Figure 6.6 shows the BER of the EM-based stochastic decoder in the HARQ-2 system using the adjustable transmitting energy setting with different distributions of 600 DCs. Since we employ the Chase combining for the latter received packets, the overall signal energy in the second transmission is \(E_T\), even though the transmitting energy of it is merely \((1 - \alpha)E_T\). Therefore, with the same DC distribution, BER is decreased as we allocate more energy to the first transmission, i.e., as \(\alpha\) tends to 1. Figure 6.6 also indicates that the 200/400 DC distribution gives the lowest BER among the simulated DC distributions for the HARQ-2 system using the EM-based stochastic decoding.

BER performance of the HARQ-2 system using the CSD is shown in Figure 6.7. Similar to the EM-based stochastic decoding, greater fraction of \(E_T\) on the first transmission results in lower BER. However, for the HARQ system using CSD, the DC distribution that leads to the least BER performance is the 300/300 DC distribution, which is different from the EM-based stochastic decoding case. Allocation of a large fraction of available DCs to either decoding attempt, e.g., 100/500 or 500/100 DC distribution, results in poor BER performance for both HARQ systems using the EM-based stochastic decoding and the CSD.

B. Decoding Latency

The decoding latency of the HARQ-2 system using adjustable transmission energy is shown in Figure 6.8. We observe that the HARQ-2 system having higher transmitting energy on the first transmission and allocating a large number of DCs in the second decoding attempt results in the least decoding delay.

Note that compared to the EM-based stochastic decoding in the HARQ-2 system, the decoding latency of the ARQ scheme using CSD is lower than 70% of the EM-based stochastic decoding case, for all simulated DC distributions.
6.3. Performance of HARQ Using Stochastic Decoding

Figure 6.6: BER performance of the (1056,528) LDPC code decoded by the EM-based stochastic decoding in the HARQ-2 system using adjustable transmission energy.

Figure 6.7: BER performance of the (1056,528) LDPC code decoded by the CSD in the HARQ-2 system using adjustable transmission energy.
6.3. Performance of HARQ Using Stochastic Decoding

Figure 6.8: Average number of DCs for the (1056,528) LDPC code decoded by stochastic decoding in the HARQ-2 system using adjustable transmission energy.

C. Transmission Energy Cost

The expected transmission energy cost of the HARQ-2 system is computed as

\[ E_T [(1 - P_{f,1}) \alpha + P_{f,1}] \]  \hspace{1cm} (6.6)

which depends on the FER of the first decoding attempt and \( \alpha \).

Figure 6.9 shows the expected transmitting energy cost for the HARQ-2 system with different value of \( \alpha \). The \( \alpha \) that results in the least energy cost depends on the distribution of DCs. For instance, for the EM-based stochastic decoding in the HARQ-2 system with 500/100 DC distribution, transmission energy is minimized when \( \alpha = 0.66 \).
6.3. Performance of HARQ Using Stochastic Decoding

Figure 6.9: Expected transmission energy for the (1056,528) LDPC code decoded by the EM-based stochastic decoder in the HARQ-2 system using adjustable transmission energy.

Figure 6.10: Expected transmission energy for the (1056,528) LDPC code decoded by the CSD in the HARQ-2 system using adjustable transmission energy.
6.4 Performance-Energy Tradeoffs for Stochastic Decoding in ARQ

When replacing the EM-based stochastic decoding by the CSD in the HARQ system, we have similar results on the expected transmission energy, as shown in Figure 6.10. When using the same DC distribution, the minimum transmission energy cost of the CSD is lower than that of the EM-based stochastic decoding due to the lower FER performance of the CSD.

6.4 Performance-Energy Tradeoffs for Stochastic Decoding in ARQ

From our numerical results, we see that the decoding performance, e.g., the error rates and latency, and the transmission energy may not be both optimally satisfied simultaneously for the HARQ system using stochastic decoding. In this section, we present tradeoffs between decoding performance and the transmission energy cost under two different scenarios: the ARQ scheme having the constant $E_T$ per transmission and the one having adjustable transmission energy.

A. The Constant Energy Scenario

When using constant $E_T$ on each transmission, we would like to allocate more DCs to the second decoding attempt. This will result in lower BER and less decoding latency, as shown in Table 6.1 and Figure 6.3. This is expected since we employ the Chase combining for the retransmitted packets to have higher SNR, which gives lower BER and reduces the required number of DCs.

However, this DC distribution may not guarantee the least energy cost, since allocating fewer DCs to the first decoding attempt will increase its FER. Higher FER in ARQ scheme means high probability of retransmission and further leads to extra energy consumption.

B. The Adjustable Energy Scenario

For the HARQ system using the adjustable transmitted energy, there are two variables we can manipulate: $\alpha$ and the distribution of DCs. Based on the results in Section 6.3.2, the ARQ scheme will have lower BER and less decoding latency when we use a large fraction
of available energy $E_T$ for the first transmission, i.e., choose $\alpha$ close to 1. In this case, the distribution of DCs is not a factor that affects the decoding latency, since the latencies of the HARQ systems using different DC distributions are almost similar when $\alpha > 0.75$, as shown in Figure 6.8. Nevertheless, the distribution of DCs can alter the BER performance. As the examples shown in Figures 6.6 and 6.7, different DC distributions result in different BERs even when $\alpha$ is fixed. A simple way to distribute the total available DCs among decoding attempts to improve BER is selecting $D_{1,max}$ slightly greater than the expected DCs at the particular SNR.

If the energy cost is the main concern of the ARQ scheme, we would like to select $\alpha$ that would result in the lowest expected energy cost for the chosen DC distribution. Note that the selected $\alpha$, which leads to the lowest energy cost, may not be the value giving the lowest BER performance or the least decoding delay. Figures 6.11 and 6.12 illustrate the relations between the expected transmission energy and decoding performance of the HARQ system using the EM-based stochastic decoding and the CSD. We use marker ■ to denote the 500/100 DC distribution, ♦ for the 300/300 DC distribution, and ● for the 100/500 DC distribution in Figures 6.11 and 6.12 in which the $\alpha$ value for each data point increases from top to bottom on the same curve. We observe that when the $\alpha$ value is decided, the ARQ scheme allocating more DCs to the first decoding attempt, e.g., $D_{1,max} > D_{2,max}$ in the HARQ-2 system, will result in lower expected transmission energy. This is because the FER of the first decoding attempt in this setting could be lower, i.e., less retransmissions.
6.4. Performance-Energy Tradeoffs for Stochastic Decoding in ARQ

Figure 6.11: Performance of the (1056,528) LDPC code decoded by the EM-based stochastic decoding in the HARQ-2 system using different $\alpha$ values and DC distributions.

Figure 6.12: Performance of the (1056,528) LDPC code decoded by the CSD in the HARQ-2 system using different $\alpha$ values and DC distributions.
6.5 Hybrid Stochastic Decoding in HARQ

In Section 5.4, we observed that the hybrid stochastic decoding, i.e., CSD followed by the EM-based stochastic decoder, can further improve the BER and lower latency. Here, we employ the proposed hybrid stochastic decoder in the HARQ system to demonstrate its performance improvement on ARQ schemes.
In order to have fewer retransmissions, i.e., reduce the transmission energy cost, we employ the CSD decoder, whose FER is the lowest compared to other stochastic decoders, in the first decoding attempt of the HARQ-2 system. Then, we use the hybrid stochastic decoding for the second decoding attempt due to its low BER performance and delay.

BER of the HARQ-2 system using the proposed hybrid stochastic decoding is shown in Figure 6.13 which demonstrates the BER improvement of the proposed hybrid decoding scheme. Compared to the HARQ-2 systems using the EM-based or the CSD decoders in both decoding attempts, the ARQ schemes adopting the hybrid stochastic decoding outperform in BER performance.

Figure 6.14 illustrates the latency of the ARQ schemes using the hybrid stochastic decoding are similar to the value of the system just using the CSD in both decoding attempts.
Compared to the ARQ scheme using the EM-based stochastic decoding, the HARQ system adopting the hybrid stochastic decoding has an approximately 30% reduction in decoding latency and achieves lower BER at the same time.

6.6 Conclusions

In this chapter, we demonstrate that the stochastic decoding algorithm is a better choice than the SPA and the MSA to be used in the ARQ schemes. We analyze the error rate improvement and decoding latency of the HARQ system under different scenarios, including systems having adjustable transmission energy and various DC distributions. We present tradeoffs between transmission energy cost and decoding performance of the HARQ system using stochastic decoding.

The reduced-latency CSD algorithm is also studied in the ARQ scheme in this chapter. We show that the CSD, due to its lower frame error rate, is an even more suitable choice than other stochastic decoders to be used in HARQ systems.
Chapter 7

Stochastic Decoding of LDPC Codes over Rayleigh Fading Channel

In previous chapters, stochastic computation has been shown to be a hardware-efficient alternative to implement iterative decoders that results in comparable performance on the AWGN and the binary symmetric channels. In this chapter, we investigate performance of stochastic decoders when used on fading channels. We also introduce a log-likelihood ratio scaling method targeted to Rayleigh fading channels that results in significantly improved error rate performance. In addition, we demonstrate that the CSD algorithm outperforms conventional stochastic decoding on Rayleigh fading channels.

7.1 Fading Channel Models

A fading channel is usually used to model the communication link in an environment where the received signal exhibits random variation in both phase and amplitude. These effects are due to the motion of the transceiver, multipath, and the scattering of the signal by interfering objects. If no line-of-sight path exists between the transmitter and the receiver, the channel is modeled as a Rayleigh fading channel because the envelope of the received signal at any time instant is a Rayleigh distributed random variable and the phase is uniform on \((0, 2\pi)\). On the contrary, if a line-of-sight path is present, the channel is modeled as a Rician fading channel.

For iterative decoding on Rayleigh fading channel, we consider the model that the \(i\)-th received channel symbol \(y_i\) at variable node \(v_i\) is
where $m_i$ is the modulated signal and $n_i$ represents the additive white Gaussian noise sample which has the noise variance $\sigma_n^2 = N_0/2$. The fading coefficient $a_i$ is a Rayleigh distributed random variable with

$$f_A(a) = 2ae^{-a^2}$$

where we have assumed that the expected power $E[A^2] = 1$ resulting in $E[A] = 0.8862$. In a more general model, $a_i$ can be assumed as a complex random variable with Rayleigh magnitude and uniform phase.

The fading coefficient $a_i$ changes with time and its instantaneous value is referred to as the Channel State Information (CSI). The CSI may be obtained at the receiver through transmission of pilot tones. The channel coherence time, $T_{coh}$, is a measure of how rapidly the CSI varies in time [117]. The larger the value of $T_{coh}$, the slower the CSI is changing with time. If the channel coherence time is smaller than the delay constraint of the channel, i.e., different variable nodes experience independent CSI values, we refer to it as a fast fading channel.

On the contrary, a slow fading channel means that the channel has a large coherence time, i.e., the CSI remains the same for a long period of time. In this chapter, we assume that in the case of slow fading channel model, the fading coefficient $a_i$ remains the same for the entire transmitted signal block, which is known as block fading.

### 7.2 Iterative Decoding on Rayleigh Fading Channels

Performance of LDPC codes on fading channels has been studied in [117], [118], and [119]. The iterative decoding algorithms for LDPC codes, e.g., the SPA and the stochastic decoding algorithms, are initialized by the received message, which is usually the LLR value. For a fading channel, the likelihood value of the received symbol $y_i$, given the modulated signal $m_i$ and the fading coefficient $a_i$ at the $i$-th variable node is
7.2. Iterative Decoding on Rayleigh Fading Channels

\[ P(y_i|m_i, a_i) = \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp \left( -\frac{(y_i - a_i m_i)^2}{2\sigma_n^2} \right) \]  

(7.3)

where \( \sigma_n^2 = N_0/2 \). For simplicity, we assume that the encoded code bit is modulated by BPSK with the energy \( E_s = 1 \).

A. Receiver with No CSI

When no CSI is available at the receiver, the LLR is computed as

\[ L_i = \ln \left( \frac{\int_{-\infty}^{\infty} P(y_i|m_i = +1, a_i) f_A(a_i) da_i}{\int_{-\infty}^{\infty} P(y_i|m_i = -1, a_i) f_A(a_i) da_i} \right) \]  

(7.4)

Here, \( f_A(a) = 2a \exp(-a^2) \) for the Rayleigh fading channel and therefore the LLR is

\[ L_i = \ln \left( \frac{\int_{0}^{\infty} a_i \exp \left( -\frac{(y_i - a_i)^2}{2\sigma_n^2} + a_i^2 \right) da_i}{\int_{0}^{\infty} a_i \exp \left( -\frac{(y_i + a_i)^2}{2\sigma_n^2} + a_i^2 \right) da_i} \right) \]  

(7.5)

No closed form exists for the \( L_i \) given above. Several authors have tried to approximate \( L_i \) in [118], [120], [121], and [122]. For instance, Yazdani and Ardakani [120, 123] presented a linear LLR approximation which is easy to implement but results in some performance degradation compared to using the true LLR value. More recently, approaches using the Taylor series expansion [121] or the Padé approximation [122] are proposed to approximate the true LLR value. Nonetheless, the computational burden of these approximations is still too high.

A widely used simple linear approximation [118] for the channel LLR is

\[ L_i \approx \frac{2y_i}{\sigma_n^2} E[A] \]  

(7.6)

The approximation is based on the assumption [124] that \( p(y_i|m_i, a_i) \) is Gaussian distributed at the most probable value, i.e., \( m_i E[A] \). In this chapter, we use this simple approximation for the case where no CSI is available at the receiver.
7.3 Scaling Scheme for Stochastic Decoding

B. Receiver with Ideal CSI

When the CSI of the fading channel is available at the receiver, the initial LLR value at the $i$-th variable node is

$$L_i = \ln \left( \frac{P(y_i|m_i = +1, a_i)}{P(y_i|m_i = -1, a_i)} \right)$$

(7.7)

$$= \frac{2y_i}{\sigma_n^2} a_i$$

Therefore, the LLR value in (7.7) or (7.6) is the received message from the Rayleigh fading channel. In stochastic decoding, the \textit{a posteriori} probability $P_i$ used to generate stochastic bits can be extracted from the corresponding LLR.

7.3 Scaling Scheme for Stochastic Decoding

In order to circumvent the latching problem in stochastic decoding process, scaling the channel messages to introduce switching activity in stochastic streams is suggested for the AWGN [16] and the binary symmetric channels [21]. However, the existing scaling methods may not be optimum for fading channels. In this section, we present a scaling scheme, called \textit{CSI Scaling} (CSIS), for stochastic decoding to achieve improved error rates and lower error floor on Rayleigh fading channel models.

A. Receiver with No CSI

When CSI is not available at the receiver, the approximated LLR in (7.6) is scaled as

$$L'_i \approx \left( \frac{2\beta \sigma_n^2}{E[A]} \right) L_i$$

(7.8)

$$= 4\beta y_i$$

where the constant $\beta$ in the scaling factor $(2\beta \sigma_n^2/E[A])$ is empirically chosen based on the BER performance. Since the scaled LLR in (7.8) becomes independent of the Gaussian noise and the CSI of the fading channel, it means that the \textit{a posteriori} probability $P_i$ can be computed merely by knowledge of the received symbol $y_i$, which greatly simplifies generation of stochastic bits.
7.4. Performance of Stochastic Decoding Algorithms on Fading Channels

B. Receiver with Ideal CSI

When the CSI is available at the receiver, we can use this information in the proposed scaling method. If the $a_i$ value is smaller than 1, i.e., the received message is in a deep fade, the scaled LLR in (7.8), which is independent of $a_i$, is used by stochastic decoder. When $a_i$ is greater than 1, the LLR value is multiplied by $a_i$, which enhances the reliability of the channel message at the $i$-th variable node. Therefore, the scaled LLR for the case of known the CSI is

$$L'_i = \begin{cases} 4\beta y_i a_i, & \text{if } a_i > 1, \\ 4\beta y_i, & \text{if } a_i \leq 1. \end{cases} \quad (7.9)$$

Here, our results indicate that the threshold value of 1 results in near-optimal error rate performance.

7.4 Performance of Stochastic Decoding Algorithms on Fading Channels

To demonstrate the performance of CSD and the proposed scaling scheme, we use it to decode the (1056,528) LDPC code from the WiMAX standard over a Rayleigh fading channel. We decode this code using SPA, EM-based stochastic decoding, and CSD algorithms. We let both the EM-based and the CSD decoders have a maximum latency of 300 DCs and the maximum number of iterations for the SPA is set to 8. Using these values, the three decoding algorithms yield similar BERs on the AWGN channel. Furthermore, the selected values of decoding rounds are quite reasonable in practice. We can improve the error rate performance of these algorithms by allowing more iterations or decoding cycles.

Note that we consider two different implementations for the SPA; a double-precision floating-point implementation, and a 6-bit fixed-point implementation where 1 bit is reserved for the sign, 2 bits for the integer part, and 3 bits for the fraction part of the encoded value. The double-precision SPA is nearly-ideal for implementation of the SPA but would result in very high hardware complexity. The fixed-point implementation is usually used in
7.4. Performance of Stochastic Decoding Algorithms on Fading Channels

Figure 7.1: BER performance of the (1056,528) LDPC code decoded by the SPA and stochastic decoders on the AWGN channel and the slow Rayleigh fading channel with the CSI available at the receiver.

hardware implementations and the uniform 6-bit quantization scheme has been shown to be a good tradeoff between hardware complexity and decoding performance [125].

7.4.1 Slow Fading

First, we consider the slow fading case, i.e., the case where the entire transmitted signal block is affected by a single fading coefficient. Under this assumption, the signal received from the channel can be treated as the output of an AWGN channel with a different $E_b/N_0$. Since the three iterative decoding algorithms with the selected decoding rounds have similar BER on the AWGN channel as shown in Figure 7.1, the BER performance of these iterative decoding algorithms are expected to be similar on the slow Rayleigh
fading channel model. Figure 7.1 also indicates roughly 37.5-dB loss in BER for the fading channel model when compared to the AWGN channel case.

For the AWGN channel case, we have used the conventional NDS given by Equation (3.1) in Section 3.2. For the fading channel, we may rewrite the received symbol as $\bar{y}_i = m_i + (n_i/a_i)$ and extend the NDS approach by letting the noise variance in the scaling factor be $(\sigma_n^2/a_i^2)$. Hence, when the receiver has the CSI, the NDS-scaled LLR from the slow fading channel is

$$L'_i = \left(2\beta \sigma_n^2/a_i^2\right) L_i = 4\beta y_i/a_i$$

(7.10)

where $L_i$ is the LLR in (7.7) and $\beta$ is set to 0.5 as suggested in [16]. If the CSI is not available at the receiver, the NDS-scaled LLR becomes

$$L'_i \approx 4\beta y_i/\mathbb{E}[A]$$

(7.11)

The scaling method in (7.10) or (7.11) results in similar error rate performance as the SPA on the slow fading channel. Because each $y_i$ is affected by the same $a_i$ during the entire block, each $L'_i$ has the same scaling factor, which ensures the same scaling at different variable nodes. However, in the fast fading case, NDS cannot guarantee a performance similar to the SPA. This means that the NDS for stochastic decoders on the AWGN channel may not be the best scaling approach for the fast fading channel.

7.4.2 Fast Fading with Ideal CSI

For the fast fading channel model with ideal CSI available at the receiver, we use the proposed CSIS of (7.9) instead of the extended form of the NDS. We select the constant $\beta = 0.5$ in the scaling factor of CSIS for the (1056,528) LDPC code to minimize the BER. Figure 7.2 shows that the NDS method does not help to lower the BER of stochastic decoding on the fast fading channel. On the other hand, using the CSIS significantly improves
7.4. **Performance of Stochastic Decoding Algorithms on Fading Channels**

![BER Performance Graph](image1)

**Figure 7.2:** BER performance of the (1056,528) LDPC code decoded by the SPA and the stochastic decoders on the fast Rayleigh fading channel with ideal CSI.

![Latency Graph](image2)

**Figure 7.3:** Average number of DCs for the (1056,528) LDPC code decoded by the EM-based decoder and the CSD decoder on the fast Rayleigh fading channel with ideal CSI.
the BER performance of stochastic decoders. We observe that the CSIS improves the performance by 3-dB compared to the case without scaling and moves the error floor from $10^{-4}$ to lower than $10^{-8}$.

The EM-based and the CSD stochastic decoders show similar BER performance on the fast fading channel. Both stochastic decoders give error rates comparable with the SPA using the 6-bit quantization.

The latency of stochastic decoders on the fast fading channel with ideal CSI is shown in Figure 7.3. The expected number of DCs for both the EM-based and the CSD decoders with the CSIS become almost half of those in the case without scaling. Compared to the EM-based stochastic decoder, the CSD algorithm demonstrates 15% reduction in the latency when no scaling is employed. After employing the CSIS, the CSD can have similar error rate performance as the EM-based decoder while reducing the decoding latency by 30%.

### 7.4.3 Fast Fading with No CSI

When the CSI is not available at the receiver, the scaling scheme of (7.8) is used for stochastic decoders. Figure 7.4 shows the BER performance of the SPA, the EM-based, and the CSD decoders on the fast Rayleigh fading channel without knowledge of the CSI and illustrates that the proposed CSIS helps to reduce BER and the error floor of stochastic decoding. The simplicity of (7.8) not only reduces the computational complexity in the conversion of stochastic bits, but also results in improved error rates.

In addition, BERs of both the EM-based and the CSD decoders can be lower than that of the SPA using 6-bit quantization. Clearly, when the CSI is available, the BERs of these two stochastic decoders are lower than the case when CSI is not available.

The decoding latency of the stochastic decoders on the fast fading channel with no CSI is shown in Figure 7.5, which illustrates that the CSIS results in a significant reduction in the latency. We also note that the CSD algorithm reduces the latency by 30% for having a BER performance similar to the EM-based stochastic decoding, when no CSI is available at the receiver.
7.4. Performance of Stochastic Decoding Algorithms on Fading Channels

Figure 7.4: BER performance of the (1056,528) LDPC code decoded by the SPA and the stochastic decoders on the fast Rayleigh fading channel with no CSI.

Figure 7.5: Average number of DCs for the (1056,528) LDPC code decoded by the EM-based decoder and the CSD decoder on the fast Rayleigh fading channel with no CSI.
7.4. Performance of Stochastic Decoding Algorithms on Fading Channels

7.4.4 Fast Fading with Noisy CSI

We also study the case where the CSI at the receiver is not ideal, i.e., there exists some error while estimating the CSI. Having noisy CSI at the receiver is quite common in practical implementations of communications over fading channels. We model the noisy CSI as

\[ g_i = a_i + e_i \] (7.12)

where \( a_i \) is the fading coefficient at the \( i \)-th received symbol, i.e., the ideal CSI, and \( e_i \) is a zero-mean Gaussian noise sample with distribution \( N(0, \sigma^2_e) \). Therefore, the received symbol in this case is

\[ y_i = g_im_i + n_i = a_im_i + (e_im_i + n_i) \] (7.13)

where \((e_im_i + n_i)\) is a zero-mean Gaussian distributed random variable having the variance of \((\sigma^2_e + \sigma^2_n)\).

While the receiver only has the biased CSI value \( g_i \), the BPSK-modulated LLR from the Rayleigh fading channel is computed as

\[
L_i = \ln \left( \frac{\int_{-\infty}^{\infty} P(y_i|m_i = +1, g_i) f_G(g_i) dg_i}{\int_{-\infty}^{\infty} P(y_i|m_i = -1, g_i) f_G(g_i) dg_i} \right)
\]

\[
= \ln \left( \frac{\int_{-\infty}^{\infty} \int_{0}^{\infty} P(y_i|m_i = +1, a_i, e_i) [f_A(a_i) * f_E(e_i)] da_i de_i}{\int_{-\infty}^{\infty} \int_{0}^{\infty} P(y_i|m_i = -1, a_i, e_i) [f_A(a_i) * f_E(e_i)] da_i de_i} \right)
\] (7.14)

where \( f_G(g_i) \) is the convolution of the Rayleigh distribution \( f_A(a_i) \) and the Gaussian distribution \( f_E(e_i) \), since \( g_i = a_i + e_i \) and \( a_i \) is independent of \( e_i \). There exists no closed form for the LLR in (7.14); however, we can approximate it as what we did in (7.6) since
Equations (7.13) and (7.1) are in the same form of a Rayleigh distributed random variable adds a zero-mean Gaussian distributed random variable. Therefore, we have

\[
L_i \approx \frac{2y_i}{\sigma_n^2} \mathbb{E}[G]
\]

(7.15)

where \( \mathbb{E}[G] = \mathbb{E}[A] \) because of \( g_i = a_i + e_i \) and the expected value of \( e_i \) is zero. This approximation is the same as the approximated LLR value in (7.6). Therefore, when using these approximated channel messages, the channel LLR for the SPA decoding in the case of having noisy CSI is the same as the one without knowing the CSI at the receiver. Note that we use \( g_i \) instead of \( a_i \) in the scaling method of (7.9) to introduce required switching activity in stochastic streams.

In our simulations, we select the energy of CSI estimation error \( \sigma_e^2 = 0.1 \) and 0.5, respectively. BER performance of the SPA and the EM-based stochastic decoders on the fading channel with noisy CSI at the receiver is shown in Figure 7.6. We observe that the SPA is more sensitive to the CSI value and its BER performance of using noisy CSI is worse than the case using the expected value of CSI. This is because that the noisy CSI \( g_i \) has larger variation than the \( \mathbb{E}[G] \) from the ideal \( a_i \) when the \( \sigma_e^2 \) is greater than 0.1 in the example of Figure 7.6. While having the same \( \sigma_e^2 \) value, compared to the SPA, the EM-based stochastic decoder has lower BER performance, which can even lower than the SPA using the expected CSI value. In Figure 7.7, we show that as the EM-based stochastic decoder, the CSD in the noisy CSI case outperforms the SPA decoder. It is due to the CSIS method, in which the noisy CSI is only used when \( g_i > 1 \), that diminishes the impact of noisy CSI.

In Figure 7.8, we show BER performance of the EM-based and the CSD decoders with different CSI scenarios. When the estimation error is small, e.g., \( \sigma_e^2 = 0.1 \) in Figure 7.8, both the EM-based and the CSD decoders have similar BERs as the case having the ideal CSI value. The larger value of \( \sigma_e^2 \) results in worse BER performance. For example, when \( \sigma_e^2 = 0.5 \), both the EM-based and the CSD decoders have approximately 1.5-dB BER degradation from the case of having \( \sigma_e^2 = 0.1 \).

The decoding latency of stochastic decoders for noisy CSI at the receiver is shown in Figure 7.9. The CSD has lower decoding latency while having the similar BER per-
Figure 7.6: BER performance of the (1056,528) LDPC code decoded by the SPA and the EM-based stochastic decoders on the fast Rayleigh fading channel with noisy CSI.

Figure 7.7: BER performance of the (1056,528) LDPC code decoded by the SPA and the CSD decoders on the fast Rayleigh fading channel with noisy CSI.
7.4. Performance of Stochastic Decoding Algorithms on Fading Channels

Figure 7.8: BER performance of the (1056,528) LDPC code decoded by the EM-based and the CSD decoders on the fast Rayleigh fading channel with noisy CSI.

Figure 7.9: Average number of DCs for the (1056,528) LDPC code decoded by the EM-based and the CSD decoders on the fast Rayleigh fading channel.
formance as the EM-based stochastic decoder. We also note that compared to the case of having ideal CSI, noisy CSI helps to slightly reduce the average number of DCs for stochastic decoding; however, the noisy CSI results in worse BER performance. This implies that if the received messages can be successfully decoded during the given DCs, the stochastic decoder having noisy CSI requires slightly less DCs to correctly decode it; otherwise, the decoder outputs relatively more errors due to the embedded noise on the CSI.

Based on Figures 7.6, 7.7, 7.8, and 7.9, we conclude that among the studied iterative decoding algorithms, the CSD is a better choice to decode messages from fading channels when having noisy CSI.

7.5 Conclusions

In this chapter, we characterize the performance of stochastic decoding on Rayleigh fading channels. We present a scaling scheme, CSIS, that provides the required switching activity in stochastic streams which results in improved error rate performance and lower error floor of stochastic LDPC decoding on fading channels. We also demonstrate that the proposed CSD algorithm has improved performance compared with the EM-based stochastic decoding on slow and fast fading channels when CSI is available at the receiver and when it is not available. The CSD can provide more than 30% latency reduction while giving similar BER performance. The high throughput of the CSD algorithm makes it outperform other stochastic decoders on fading channels.
Chapter 8

Conclusions

8.1 Contributions

We presented a reduced-latency stochastic LDPC decoding algorithm and proposed several performance improving designs for stochastic decoders to achieve lower error rate performance and higher throughput. We also statistically analyzed the decoding behavior of stochastic LDPC decoders, including the convergence of decoding process and its error characteristics. Furthermore, we presented the advantages of stochastic decoding over other iterative decoding algorithms in the ARQ scheme and studied the performance of stochastic decoding on fading channels. We demonstrated that stochastic decoding performs better than other iterative decoding schemes when applied to fading channels.

We presented a Markov chain model to explore the switching activity and the convergence of transition probabilities in stochastic decoding. We utilized this model to demonstrate possibility of decreasing the decoding latency of stochastic decoders. In addition, we proposed a space-efficient code bit determination design to further improve the throughput without any performance loss.

We also studied the error characteristics of stochastic LDPC decoding and reported the stochastic-specific-trapping sets. The trapping set found in the (1056,528) LDPC code from the WiMAX standard is an elementary absorbing $T(10, 1)$ set. Knowledge of these specific trapping sets helps improve performance by selectively avoiding these structures in the code graph. This observation is also helpful in lowering the error floor of stochastic decoder and makes stochastic decoding suitable for applications requiring extremely low error rates.

A reduced-latency stochastic decoding algorithm, CSD, is proposed to decode prac-
8.2. Future Work

The CSD algorithm improves the error rate performance and reduces the decoding latency of the existing stochastic decoders. For instance, when decoding the (1056,528) LDPC code, CSD can provide 0.3-dB BER improvement while reducing the latency to 65% of the EM-based stochastic decoder. Furthermore, its lower FER performance makes it a better choice to be used as the decoding algorithm in the applications requiring low FER, e.g., ARQ schemes.

We also studied the advantages of using stochastic decoding in ARQ schemes. The tradeoff between decoding performance and transmission energy cost were presented. For example, compared to the EM-based stochastic decoder in the HARQ system, the CSD can give 0.5-dB improvement in the BER and reduces the decoding latency by more than 30%.

Furthermore, we employed stochastic decoding to other channel models. In order to have efficient randomization in stochastic streams to alleviate the latching problem, we presented a constant scaling method for the binary symmetric channel. The proposed method results in low computational complexity in generation of stochastic streams and provides significant BER improvement.

In the last chapter, we characterized the performance of stochastic decoding on fading channels and introduced a scaling method, CSIS, to achieve 3-dB improvement in the BER and to significantly lower the error floor. In the study on the fading channels, we found that CSD outperforms the existing stochastic decoders. After employing our proposed CSIS, the CSD provides 30% decoding latency reduction compared with the EM-based stochastic decoder.

8.2 Future Work

Inspired by the results and intuitions gained from this work, some directions can be proposed for further research. These potential research topics are briefly discussed below.

8.2.1 Non-Binary Conditional Stochastic Decoding Algorithm

Stochastic decoding is not limited to binary error-correcting codes. Non-binary stochastic decoders can be realized with some adjustment in hardware design and implementation.
For instance, non-binary EM design [20] and non-binary version of TFM [126] have been used to implement stochastic decoders for non-binary LDPC codes. Non-binary version of the CSD algorithm has high potential to provide high throughput in stochastic decoding of non-binary error-correcting codes.

### 8.2.2 Different Channel Models or Higher Order Modulations

Previous work on stochastic decoding has been limited to the AWGN channel model. Study of stochastic decoding on other channel models is quite limited. The partial-response channel was studied in [127]. We investigated the possibility of stochastic decoding on the binary symmetric channel models in Chapter 3 and the fading channels in Chapter 7. However, other channel models such as the channel with non-linear characteristics are not studied. Besides, higher order modulations, like M-ary phase-shift keying and M-ary quadrature amplitude modulations, are feasible methods to transmit data for applications requiring high data rate, e.g., LTE [112] and High Speed Packet Access (HSPA) [128]. Investigation of the tradeoff between decoding performance and hardware complexity of stochastic decoding for other channel models and higher order modulations are other possible directions for future work.

### 8.2.3 Improved Error Floor Performance of Stochastic LDPC Decoding

Many approaches have been proposed to lower the error floor of iterative decoding. As for stochastic decoding, there are few work on this topic. In [129], an asynchronous hardware design is presented to reduce the error floor. In this dissertation, we provided several designs that result in lower error floor under stochastic decoding. We have also found that other approaches proposed for belief propagation algorithm to lower its error floor may not work with stochastic decoding. Study of approaches to effectively lower the error floor of stochastic decoding would be a valuable research work.
Bibliography


[88] F. Leduc-Primeau, S. Hemati, S. Mannor, and W. J. Gross, “Relaxed half-
stochastic belief propagation,” *IEEE Transactions on Communications*, vol. 61, 

 asynchronous data transfer scheme for interleaving in LDPC decoders,” in *IEEE 


