Fully Parallel Implementation of Timing-Error-Tolerant LDPC Decoders

by

Xin Zuo

A thesis submitted for the degree of

Doctor of Philosophy

January, 2016

Department of Electronics and Computer Science

Faculty of Physical Science and Engineering

University of Southampton

Southampton SO17 1BJ

United Kingdom

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCE AND ENGINEERING
DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Fully Parallel Implementation of Timing-Error-Tolerant LDPC Decoders

by Xin Zuo

In this thesis, the design of fully parallel timing-error-tolerant Low-Density Parity-Check (LDPC) decoders have been investigated. LDPC decoders are employed in numerous communication systems to correct channel-induced transmission errors. The ever increasing data traffic demands require LDPC decoders that are capable of providing high processing throughput and low processing latency, using limited hardware resources and energy consumption. The fully parallel implementation of LDPC decoders is suitable, due to the high throughput and low latency that this affords. However, the task of designing reliable Very Large-Scale Integration (VLSI) systems is becoming increasingly challenging in successive generations of nanoscale fabrication technology. This may be attributed to the occurrence of timing errors, during the processing, which is caused by the increasing susceptibility to IR drop, inductive noise, crosstalk, electrostatic discharges, particle strikes, switching noise and fabrication process variations. Therefore it is necessary to consider the effects of timing errors during the design of LDPC decoders. However, the characterization of the timing error tolerance of LDPC decoders relying on measurements obtained directly from a fabricated Application-Specific Integrated Circuit (ASIC) may not be preferable, owing to the associated risk of wasting all of the invested time, effort and expense, if the ASIC is not able to facilitate the desired outcomes. A novel design flow is therefore proposed in this thesis, which allows the use of simulations at the algorithm level to investigate the decoders' error correction performance, with considerations of the occurrence of timing errors in the hardware architecture level of the design.

LDPC decoders employing the optimal Sum-Product Algorithm (SPA) have a very high implementation complexity, which requires the exchange of floating point probabilities between the parity-Check Nodes (CNs) and Variable Nodes (VNs) in their factor graph representation. In order to reduce the complexity, the Log-Sum-Product Algorithm (Log-SPA) and the Min-Sum Algorithm (MSA) may be employed in the LDPC decoder, which operate on a basis of Log-Likelihood Ratios (LLRs), rather than probabilities. These LLRs can be represented by Fixed-Point (FP) numbers, comprising

a number of bits, referred to as the bit width. It is this bit width that proportionally determines both the size of the memory required, as well as the area of the data path and hence the energy consumption imposed. We propose the use of EXtrinsic Information Transfer (EXIT) charts to select the bit widths for the Fixed-point LDPC Decoders (LDPC-FDs), in order to achieve a desirable trade-off between the implementation complexity and the error correction performance. This significantly expedites the LDPC-FD design process, relative to the conventional approach of using trial and error based Bit Error Ratio (BER) simulations. Using the proposed design flow, timing characteristics analysis may be performed on the LDPC-FD, in order to derive an error model of the causes and effects of timing errors. With the aid of the error model, the error correction performance of the LDPC-FD in the presence of timing errors may be characterized. In this way, the parametrization of the LDPC-FD may be optimized.

In Stochastic LDPC Decoders (LDPC-SDs), only a single bit is exchanged between each pair of CNs and VNs in each clock cycle. Over the course of several successive clock cycles, the individual bits that are exchanged between a particular pair of nodes collectively form a Bernoulli sequence, which may replace the LLRs conventionally used in LDPC-FDs. Owing to this, the operations of the CNs and VNs may be implemented using simple logic gates, which grants LDPC-SDs the practical opportunity for fully parallel implementation. As in LDPC-FDs, the proposed design flow may be adopted to guide the investigation of the timing error tolerance of LDPC-SDs, in order to determine their optimal parametrization.

Acknowledgements

I wish to thank my supervisors, Prof. Lajos Hanzo and Dr. Robert Maunder, for their generous and consistent help. I am deeply indebted to them for their wise guidance and stimulating encouragement through years. I could not complete this thesis without their help. I also would like to express my gratitude to all the colleagues who helped me to complete this thesis.

I also wish to express appreciation to my parents. This thesis is dedicated to my beloved grandparents.

List of Publications

- **X. Zuo, I. Perez-Andrade, R. G. Maunder, B. M. Al-Hashimi and L. Hanzo**, "Improving the Tolerance of Stochastic LDPC Decoders to Overclocking-Induced Timing Errors: A Tutorial and a Design Example", in *IEEE Access*, vol. 4, pp. 1607-1629, 2016.

- **X. Zuo, S. Zhong, K. Li, I. Perez-Andrade, R. G. Maunder, B. M. Al-Hashimi and L. Hanzo**, "High throughput timing-error-induced VLSI implementation of LDPC decoding using the base-minus-two fixed-point number representation", In preparation for *IEEE Journal of Solid-State Circuits*.

- **X. Zuo, R. G. Maunder, and L. Hanzo**, "Design of Fixed-Point Processing Based LDPC Codes Using EXIT Charts", in *IEEE Vehicular Technology Conference (VTC Fall)*, pp.1-5, 5-8 Sept. 2011.

- **Y. Huo, X. Zuo, R. G. Maunder, and L. Hanzo**, "Inter-layer FEC decoded multi-layer video streaming", in *IEEE Global Communications Conference (GLOBECOM)*, pp.2113-2118, 3-7 Dec. 2012.

- **I. Perez-Andrade, X. Zuo, R. G. Maunder, B. M. Al-Hashimi and L. Hanzo**, "Analysis of voltage- and clock-scaling-induced timing errors in stochastic LDPC decoders", in *IEEE Wireless Communications and Networking Conference (WCNC)*, pp.4293-4298, 7-10 April 2013.

- **C. Xu, X. Zuo, S. X. Ng, R. G. Maunder, and L. Hanzo**, "Reduced-Complexity Soft-Decision Multiple-Symbol Differential Sphere Detection", in *IEEE Transactions on Communications*, vol.63, no.9, pp.3275-3289, Sept. 2015.

Contents

Chapter 1

Introduction

1.1 Error-Tolerant Design of Channel Decoders

The data traffic carried by wireless communication is increasing explosively in every corner of the world nowadays, from cell phone users on the ground to the aircraft above in the sky, from the Tactile Internet [1, 2] to cloud computing, and so on. However, wireless communication channels impair the transmitted data, owing to the effects of noise, fading and interference [3]. This imposes transmission errors, which must be repaired in the receiver using channel decoders for error correction [3]. Therefore more and more powerful channel decoders are required, which are capable of processing massive amounts of data using less processing time and without using excessive hardware resources or energy consumption. The rise of 'turbo-like' channel decoders [4, 3, 5] using iterative decoding algorithms has illuminated the path towards this goal, due to their outstanding error correction capability. In particular, Low-Density Parity-Check (LDPC) decoders [6, 7] are a class of 'turbo-like' decoders that are particularly suited to very high processing throughputs, when implemented in hardware. This is because LDPC decoders offer great potential for fully parallel and distributed computation, owing to the parallel-friendly structure of the factor graph [8, 9] that defines their operation. However, there are significant challenges in converting this potential into practice, since the graph is usually too complex to be fully implemented without making some hardware specific optimizations. Therefore, the code design has to be considered jointly with the implementation. In one approach, the graph of an LDPC code may be simultaneously designed to be sufficiently random to ensure a desirable error correction performance as well as to be sufficiently structured to allow the decoder to be implemented based on a part of the graph that can be reused to implement the other parts of the graph. In this way, the implementation complexity of the decoder can be significantly reduced. Numerous

previous research efforts [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26] have investigated the partially parallel approach, which is a compromise of the fully parallel implementation. Only recently a non-binary LDPC decoder Application-Specific Integrated Circuit (ASICs) has been implemented in a fully parallel manner [27, 28], owing to the small graphs that are used in non-binary LDPC codes [29, 30]. In the future, fully parallel implementations of channel decoders will become more and more common, as fabrication technology keeps improving.

On the other hand, the task of designing completely reliable large-scale ASIC is becoming increasingly challenging in successive generations of nanoscale fabrication technology. This may be attributed to the increasing susceptibility of lower scale technology to processing errors, including IR drop[1], inductive noise, cross talk, electrostatic discharges, particle strikes, switching noise and fabrication process variations [31, 32, 33, 34, 35, 36, 37]. As fabrication technology is developing to the sub-65 nm regime, it is no longer possible to guarantee the reliability of nanoelectronic ASICs [37]. Therefore the design of future digital circuits must consider the processing errors that may occur. Motivated by this, error-tolerant electronic design has already been adopted in numerous applications [38, 39, 40, 41, 42, 33, 43, 44].

Fortunately, the error correction capability of channel decoders grants them some inherent tolerance to processing errors, as we shall demonstrate in this thesis. More specifically, we characterize the error correction performance of fully parallel LDPC decoders in the presence of timing errors. In order to characterize timing error tolerance, measurements may be taken from a fabricated ASIC implementation of an the LDPC decoder. However, the characteristics of the tape-out [2] may be unpredictable during the design stage with a risk that the tape-out will be unsuccessful. That risk increases as the complexity of the design increases. In this event, the huge amount of design time, effort and expense required for a tape-out would be wasted, if it was unsuccessful. Owing to these issues, we propose the use of simulations at the algorithm level to characterize fully parallel LDPC decoders. However, we propose to incorporate a model of the causes and effects of timing errors, which is parametrized by characteristics obtained from the later design stages. The conventional design flow and the proposed modification are illustrated in Figure 4.1. In this thesis, we will characterize the timing error tolerance of fully parallel implementation of LDPC decoders by using the proposed design flow. This simulation-only approach de-risks the expensive and time-consuming fabrication of an

[1]The resistance of the power distribution network in integrated circuit systems may cause a drop in the power supply voltage, commonly referred to as the IR drop, where I is the current flowing in the network and R is the resistance.

[2]In electronics design, tape-out or tapeout is the final result of the design for integrated circuits, which is sent for manufacture.

ASIC, allowing confidence to be gained in a timing-error-tolerant design. Furthermore, our simulations at the upper levels of the design flow are immune to random variation and are readily repeatable, using synthesis processing, SPICE [45, 46, 47] simulation or Cadence [48] simulation.

1.2 Structure and Novel Contributions of the Thesis

The outline of thesis is presented below:

- Chapter 2 briefly reviews linear binary block codes, LDPC codes and EXtrinsic Information Transfer (EXIT) charts [49, 50]. The EXIT charts of LDPC decoders using floating point operation is studied. We present define Bit Error Ratio (BER) results to characterize the error correction performance of the LDPC decoders and to validate the predictions that we made using EXIT charts. In the following chapters, EXIT charts will be used as a design tool during the implementation of LDPC decoders.

- Chapter 3 extends the EXIT charts of Chapter 2 to the characterization of the Fixed-point LDPC Decoders (LDPC-FDs) using Two's Complement (TC) Fixed-Point (FP) representation. Based on these EXIT charts, the bit width of 4 is recommended for the TC representation LDPC-FDs, in order to strike a desirable trade-off between the complexity and the error correction performance.

- Chapter 4 proposes a fully parallel implementation of LDPC-FDs, using a 4-bit Base Minus Two (BMT) representation, rather than the conventional TC representation. The causes and effects of timing errors are modelled and used for conducting simulations to characterize the timing error tolerance of the LDPC-FDs. Based on the simulations, the LDPC-FD using BMT is demonstrated to have superior timing error tolerance than these using TC, as well as 10% lower ASIC area, 10% lower power consumption and 23% lower clock period.

- Chapter 5 introduces the stochastic implementation of LDPC decoders, which only needs to process single bits in each clock cycle, rather than 4 bits as in FP implementation. The building blocks of stochastic computation and the fully parallel implementation of the LDPC-SDs are discussed in detail. The error correction performance of Stochastic LDPC Decoders (LDPC-SDs) are comprehensively

characterized as functions of the implementation parameters. Based on these simulation results, a particular parametrization of the fully parallel implementation of the LDPC-SDs is recommended.

- Chapter 6 employs the fully parallel LDPC-SD of Chapter 5 and investigates its timing error tolerance. Similarly to Chapter 4, an error model is derived and used for conduction simulations. By examining the causes and effects of each type of timing error, we have pinpointed the bottleneck of the LDPC-SD, in terms of timing error tolerance. Motivated by this, we propose a modified LDPC-SD design to significantly improve the timing error tolerance. By comparing the simulation results, this modified Edge Memory (EM) structure is demonstrated to achieve a higher tolerance to timing errors, while reducing the required implementation resources at the same time. A particular 41% reduction in clock period and 28.6% in the number of logic gates for building EMs are reported.
- Chapter 7 concludes the main findings of the thesis and presents suggestions for future research.

In summary, the novel contributions of this thesis are:

- the first comprehensive tutorial on the operation of fully parallel stochastic LDPC decoders;
- error models for the causes and efforts of timing errors in fully parallel FP and stochastic LDPC decoders;
- comprehensive characterizations of the error correction performance of fully parallel floating-point, FP and stochastic LDPC decoders as function of their parameters both in the absence and presence of timing errors;
- a technique that allows EXIT chart analysis to be used for the first time to characterize and parametrize hardware implementation of channel decoders; in order to achieve a desirable trade-off between error correction performance and complexity;
- first BMT-based implementation of LDPC-FD, which significantly improves the timing error tolerance, clock period, chip area and energy consumption, compared to TC-based implementations;
- a novel structure for the processing nodes of LDPC-FDs, which minimizes and equalizes the propagation delay of all circuit paths through the nodes, further improving the clock period;

- a novel design flow that utilizes simulation at the algorithm level to model the causes and effects of timing errors, as characterized at the implementation level of the design flow;
- a novel modification to the LDPC-SD which significantly improves its tolerance to timing errors.

Chapter 2

Low-Density Parity-Check Codes

2.1 Channel Coding in Communication Systems

Channel coding refers to a technique used in digital communications that is designed to protect the transmitted data from channel impairments, such as noise, interference and fading [51], therefore improving the performance of communication systems. The research of channel coding can be traced back to the pioneering work of Shannon's publication in 1948 [52], which suggested that there always exists a channel coding scheme that can achieve error-free communications despite the prevailing channel conditions, provided that the channel capacity is not exceeded by the channel coded throughput. Since then, the research community has endeavoured to transform Shannon's theory into practice. Commonly, this data protection is achieved by transforming the original bit sequence to some specially encoded sequence, which includes additional redundant bits. In this case, a channel code is also known as a *Forward Error Correction* (FEC) or an *error-correcting code*. During transmissions using a channel code, the channel decoder at the receiver is able to detect and correct transmission errors and therefore recover the data, based on its knowledge of the encoding process that is used in the transmitter. However, channel coding may not allow the channel decoder to correct all transmission errors, where the ratio between the number of remaining errors and the number of original data bits is referred to as Bit Error Ratio (BER), which is used to measure the error correction capability of the channel code. Figure 2.1 depicts a block diagram of a baseband digital communication system, where the channel coding is shown as a pair of encoder and decoder blocks[1].

[1]This is only an inexact sketch to show the deployment and role of channel coding in communication systems, with other necessary blocks omitted, such as radio-frequency stage, synchronization, *etc.*

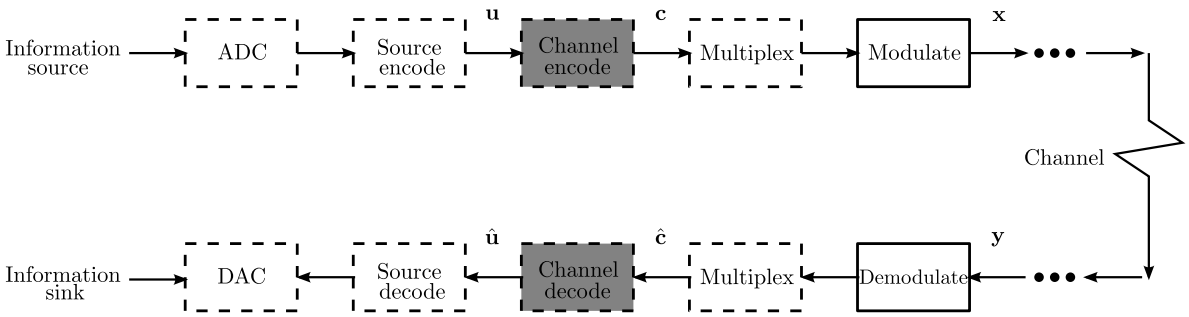


Figure 2.1: Baseband block diagram of a digital communication system. The dashed boxes indicate optional blocks, while the solid ones are essential.

Various FEC codes have been proposed and all the codes can be classified into three subcategories[2], namely *block* codes, *convolutional* codes [53] and *concatenated* codes [4, 3, 5]. A block code[3], such as a BCH code [54, 55], Reed-Solomon code [56] or an LDPC code [6, 7], expands a block of bits into a *codeword*, where the size of both the uncoded block and the encoded codeword are fixed. By contrast, a convolutional encoder may accept bit streams of arbitrary size. The decoders of both types of codes may search the set of all valid encoded codewords for the one that most closely resembles the received bit $\mathbf{y}$ using hard-decision algorithms, which only accept inputs comprising binary digits. In this case, the demodulator must make decisions as to whether each received bit is a 1 or a 0, before providing inputs to the channel decoder. However, this leads to a lack of reliability in the binary values handled by the hard-decision algorithms, limiting the error correcting performance of the block and convolutional decoders. Motivated by this, so-called soft-decision algorithms have been developed, such as the soft-decision Viterbi algorithm used for convolutional decoders [57, 58], which significantly improve the decoders' error correcting capability. In contrast to hard-decision algorithms, soft-decision algorithms process *soft* inputs and may generate soft outputs as well, which represent binary digits with multiple level of reliabilities, allowing the decoder to search for a valid codeword with higher accuracy. More specifically, soft decision expresses not only *what* the most likely value of each bit is, but also *how* likely that value is. However, in order to further improve the error correcting capability of channel codes, two or more block codes or convolutional codes may be combined to work in cooperation, such as in the turbo code [4]. Such a hybrid code may be referred to as a *concatenated* code, where each constituent block or convolutional code is referred to as a *constituent* code. The codeword of a concatenated coding scheme is generated using either parallel or serial concatenation of the constituent encoders, while the decoded codeword may be found

[2]Only linear codes are considered in depth by this thesis.

[3]Block codes also known as algebraic codes, since they operate on the basis of the algebraic properties of finite fields.

by iteratively exchanging soft decisions indicating whether the encoded bits are binary 1 or 0 between the constituent decoders. These iterative decoding algorithms schemes are broadly adopted in modern communication systems, owing to their outstanding error-correcting performance. Figure 2.2 lists the most relevant previous publications along two main timelines, namely the timeline of linear binary block codes and that of the convolutional codes. Each timeline is represented by a vertical line with the downward direction representing the chronological order, where each knot on the vertical lines represent a publication discussed above. In the remainder of this thesis, we focus on linear binary block codes and more specifically on LDPC codes, which are block codes that can benefit from the iterative decoding approach of concatenated codes.

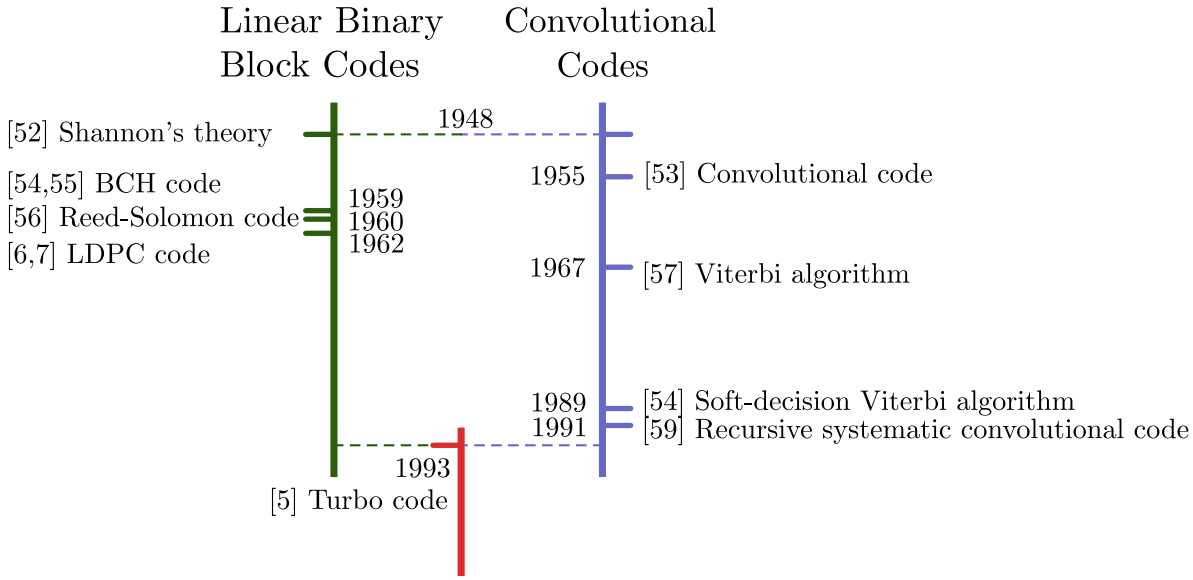


Figure 2.2: Brief timeline of linear binary block codes and convolutional codes.

2.2 Linear Binary Block Codes

As discussed in Section 2.1, in block codes, a block of information bits having a fixed length of K bits is encoded into a codeword having a fixed length of N bits, where $N > K$, since encoding inserts additional redundant bits. A block code can be denoted as a (N, K) code, with the ratio between the number of information bits and codeword bits denoted as the coding rate, $R = \frac{K}{N}$. Binary linear block codes are a class of block code, which insert linear parity sums of information bits as the redundant bits, which are hence named *parity bits*. The encoder of a binary linear block code is associated with the so-called generator matrix having the dimensions of $K \times N$, which defines the generation of all the parity bits. The generator matrix is uniquely paired with a

Parity-Check Matrix (PCM), upon which the corresponding decoder relies. The pair of matrices are discussed with examples in the following subsections.

2.2.1 Generator Matrix

Table 2.1: An example of parity bits, obtained from a set of information bits. Here, the notation $\oplus$ refers to modulo-2 addition.

information bits	parity bits
u_1, u_2, u_3	$p_4 = u_1 \oplus u_2$, $p_5 = u_1 \oplus u_2 \oplus u_3$

In a binary linear block code, the parity bits are obtained as modulo-2 additions of information bits. For example, a parity sum of the first two bits out of the three information bits, u_1, u_2 and u_3, is calculated as $u_1 \oplus u_2$, while a parity sum of the three information bits is calculated as $u_1 \oplus u_2 \oplus u_3$, notated as p_4 and p_5 in Table 2.1, respectively. Here, the notation $\oplus$ refers to the modulo-2 addition, which calculates $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$ and $1 \oplus 1 = 0$. Therefore, each pattern of the three information bits, u_1, u_2 and u_3, is uniquely transformed into a codeword of five bits, u_1, u_2, u_3, p_4 and p_5, which defines a $(5, 3)$ binary linear block code. By enumerating all the patterns of the information bits and the corresponding codewords, the parity sums can be interpreted as a Look-Up Table (LUT), referred to as the codebook, which can be used by the decoder to search for the codeword that most closely resembles the received bits or soft-decisions. Table 2.2 presents the LUT of the $(5, 3)$ code defined by the parity sums of Table 2.1.

Table 2.2: The LUT of the $(5, 3)$ code defined in Table 2.1.

message	codewords
000	00000
001	00101
010	01011
011	01110
100	10011
101	10110
110	11000
111	11101

However, as the length K of the information block increases, the corresponding codebook may become too large to implement in the encoder and decoder, since the LUT has 2^K entries, which grows exponentially with K. Therefore, the encoder may employ a

compressed version of the LUT, which is the so-called generator matrix. The sequence of the K information bits can be denoted as a vector $\mathbf{u} = [u_1, u_2, \cdots, u_K]$, while the additional sequence of parity bits can be written as $\mathbf{p} = [p_{K+1}, p_{K+2}, \cdots, p_N]$, which may be concatenated to obtain the codeword $\mathbf{c} = [\mathbf{u}, \mathbf{p}] = [u_1, u_2, \cdots, u_K, p_{K+1}, p_{K+2}, \cdots, p_N]$. Since the parity bits of each codeword are linear combinations of information bits, any codeword in the N-dimensional vector space described by the LUT can be obtained as a combination of the vectors in the K-dimensional subspace $\{\mathbf{c}\}$, which contains K linearly independent vectors $\{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_K\}$. In this way, any valid codeword may be obtained as a linear combination of all the codeword vectors in the subspace,

$$\mathbf{c} = u_1\mathbf{c}_1 + u_2\mathbf{c}_2 + \cdots + u_K\mathbf{c}_K. \tag{2.1}$$

The generator matrix can be obtained from the K base vectors $\{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_K\}$ as,

$$\mathbf{G} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_K \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,N} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,N} \\ & \vdots & & \\ c_{K,1} & c_{K,2} & \cdots & c_{K,N} \end{bmatrix}, \tag{2.2}$$

where the elements in a base vector $u_1, u_2, \cdots, u_K, p_{K+1}, p_{K+2}, \cdots, p_N$ is replaced by $c_{i,j}$, $i \in \{1, 2, \cdots, K\}, j \in \{1, 2, \cdots, N\}$, for the sake of consistency in the mathematical representation. The corresponding encoding process can be represented as the multiplication of the vector of information bits $\mathbf{u}_{1\times K}$ and the generator matrix $\mathbf{G}_{K\times N}$, according to

$$\mathbf{c}_{1\times N} = \mathbf{u}_{1\times K} \cdot \mathbf{G}_{K\times N}. \tag{2.3}$$

If all the bits in all the rows $\{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_K\}$ of the generator matrix $\mathbf{G}$ are arranged in the same order as $[u_1, u_2, \cdots, u_K, p_{K+1}, p_{K+2}, \cdots, p_N]$ shown in Table 2.2, then the left-hand part of the generator matrix can be transformed into an identity matrix through elementary row and/or column operations. In this case, the code is called a systematic code, while the generator matrix $\mathbf{G}$ is called a systematic generator matrix, having the structure

$$\mathbf{G}_{K\times N} = [\mathbf{I}_K \vdots \mathbf{P}_{K\times M}], \tag{2.4}$$

where $M = N - K$ and $\mathbf{I}_K$ is an identity matrix having the dimensions of $K \times K$. The example $(5, 3)$ code of Table 2.1 has the left-hand systematic generator matrix of

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}. \tag{2.5}$$

Accordingly, the resultant codeword will contain a duplicate of the original message in the left-hand part, according to

$$\mathbf{c}_{1\times N} = [\mathbf{u}_{1\times K} \vdots \mathbf{p}_{1\times M}], \tag{2.6}$$

2.2.2 Parity-Check Matrix

The PCM $\mathbf{H}$ of a binary linear block code is uniquely paired with its generator matrix. The PCM has the dimensions of $M \times N$ and is defined such that

$$\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}_{K\times M}, \tag{2.7}$$

where $\mathbf{0}$ represents an all-zeros matrix and T is the transpose operation. Corresponding to (2.4), $\mathbf{H}$ has a systematic form of

$$\mathbf{H}_{M\times N} = [\mathbf{P}^T_{M\times K} \vdots \mathbf{I}_M]. \tag{2.8}$$

Therefore, the systematic PCM of the previous example (5,3) LDPC code from Table 2.1 can be represented as

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{2.9}$$

The PCM can be used by a block decoder to check whether a received codeword satisfies the parity sums, which are defined at the encoder. Explicitly, a valid codeword encoded from (2.3) should have an all-zero syndrome, which can be obtained as the multiplication of the codeword and the PCM, according to

$$\mathbf{s_{1\times M}} = \hat{\mathbf{c}} \cdot \mathbf{H}^T = \hat{\mathbf{u}} \cdot \mathbf{G} \cdot \mathbf{H^T} = \mathbf{0}_{1\times M}, \tag{2.10}$$

where $\hat{\mathbf{u}}$ and $\hat{\mathbf{c}}$ are the receiver's estimations of $\mathbf{u}$ and $\mathbf{c}$, respectively, while the resultant vector $\mathbf{s}$ is called the syndrome. If an all-zero syndrome is found using (2.10), then the codeword $\hat{\mathbf{c}}$ estimated at the decoder may be considered to be a valid codeword from the set $\{\mathbf{c}\}$. By contrast, if $\mathbf{s} \neq \mathbf{0}$, then the estimated codeword $\hat{\mathbf{c}}$ definitely contains errors. In the latter case, the particular value of $\mathbf{s}$ identifies a particular error pattern in a one-to-one manner [51]. Therefore, the syndrome may be used to pinpoint and correct the error bits, relying on a particular algorithm designed for this purpose. In particular, the following section will discuss a powerful algorithm for correcting errors in the specific class of binary linear block codes known as LDPC codes. Once $\hat{\mathbf{c}}$ as been

recovered correctly, the decoded information bits can be obtained according to

$$\hat{\mathbf{u}} = \hat{\mathbf{c}} \cdot \mathbf{G}^{-1}. \tag{2.11}$$

2.3 LDPC Codes

LDPC codes [6] are a particular type of binary linear block code, whose PCM contains only a small number of '1's, giving a 'low-density'. Since LDPC codes usually have a block length N of a few hundreds to thousands of bits, a graphical representation may be used to represent the complex characteristics of the large PCM. This graph is referred to as the factor graph [8, 9] or Tanner graph [59], which is uniquely mapped to the matrix representation, namely to the pair of the generator matrix and the PCM, as introduced in Section 2.2. LDPC codes exhibit an outstanding error correction capability, when using iterative soft-decision decoding algorithms, such as the Sum-Product Algorithm (SPA) and the Min-Sum Algorithm (MSA) [8], which operate on the basis of the topology defined by the factor graph. As a result, LDPC codes have been found in numerous practical applications, such as code-division multiple-access applications [60, 18, 61], IEEE 802.16e [62, 26] and IEEE 802.11n/ac [63]. The factor graph and the iterative decoding algorithm are discussed in the following subsections.

2.3.1 Factor Graph

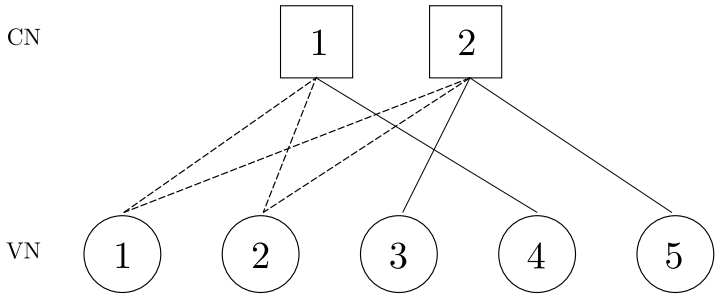


Figure 2.3: An example factor graph, corresponding to the $(5, 3)$ LDPC PCM of (2.9), although this $(5, 3)$ code is too small to be considered as a valid LDPC code.

The parametrization of an LDPC decoder is completely specified by its factor graph [59, 8, 9], which has a design depending upon that of the corresponding PCM. Figure 2.3 illustrates the factor graph of an LDPC code which uses the N encoded bits to represent K information bits, where $N > K$. Figure 2.3 depicts an example factor graph, corresponding to the $(5, 3)$ LDPC PCM of (2.9). The factor graph is comprised

of $M = N - K$ parity-Check Nodes (CNs) and N Variable Nodes (VNs). The CNs and VNs are connected by the edges, indicated as interconnections in Figure 2.3. The c^{th} CN has a degree of d_c, if it has d_c number of Input/Output (IO) ports, which are connected to d_c different VNs by edges within the factor graph, as shown in Figure 2.3. Similarly, the v^{th} VN has a degree of d_v, if it has $(d_v + 1)$ IO ports, one of which is connected to the channel and the remaining d_v ports are connected to different CNs by edges within the factor graph. The total number of edges in the factor graph is notated as N_{edge}, which is equal to the sum of the degrees of all the CNs or equivalently, of all the VNs. During LDPC decoding, the CNs and VNs work together to refine the soft messages associated with the N LDPC-encoded bits, which are then used to decide the values of N decoded bits, as shown in Figure 2.3. More specifically, the VNs and CNs iteratively exchange the soft messages along the edges of the factor graph. One decoding iteration in a LDPC decoder is defined as a period in which all the CNs and the VNs are activated once. The LDPC decoder continues processing soft messages iteration by iteration, until a fixed number of iterations is reached or until an early-stopping criterion is satisfied, such as having found an all-zero syndrome as demonstrated in (2.10) [64]. Following this, the K information bits may be extracted from the N decoded bits and output, as demonstrated in (2.11).

An LDPC code is referred to as *regular*, if all CNs have the same degree d_c and all VNs have the same degree d_v, and it is referred to as *irregular* otherwise. Note that the factor graph may comprise so-called *cycles*, which are looping paths along the edges between connected CNs and VNs, where the number of edges in a cycle is called its girth. The girth of cycles can be as small as 4, as illustrated by the dashed lines in Figure 2.3, which provides a simple example factor graph for the PCM of (2.9). The minimum girth of all the cycles in a factor graph is one of the main factors that determines the error-correcting performance of the corresponding LDPC code. This is because any unreliable soft messages loop quickly within small-girth cycles, increasing their apparent credibility quickly compared to reliable soft messages within large-girth cycles. This prevents the decoder from achieving the efficient exchange of reliable soft messages between CNs and VNs [59, 9]. Therefore, the design of LDPC codes in practical standards usually has a particular focus on eliminating the smallest girth-4 cycles in the factor graph, making the best efforts to achieve a minimum girth of 6, 8, or even higher [65, 63]. Furthermore, the node degrees d_c and d_v, code length N and coding rate $\frac{K}{N}$ also significantly affect the error-correcting performance of LDPC codes. Irregular LDPC codes usually outperform regular counterparts, since the additional freedom that is afforded when selecting the node degree distribution allows the design of the factor graph to eliminate more small-girth cycles. Similarly, since a larger factor

graph is less likely to contain small-girth cycles, LDPC codes having larger code lengths exhibit a better error-correcting performance than shorter counterparts. Furthermore, LDPC codes having a lower coding rate usually outperform codes having higher coding rates, since each uncoded bit is afforded better protection by the greater number of parity bits involved.

2.3.2 LDPC Decoder and Decoding Algorithms

Based on the factor graph, numerous soft-decision decoding algorithms have been proposed for LDPC decoding [8, 64]. Among these algorithms, the SPA [8, 64] uses probabilities to represent the soft messages exchanged between CNs and VNs. Other soft-decision LDPC decoding algorithms usually represent the soft messages using so-called Log-Likelihood Ratios (LLRs) [11, 15, 66, 67], rather than probabilities. These algorithms include the Log-SPA algorithms [8, 11, 68] and the MSA [11], which is a widely used simplification of the Log-SPA. We will discuss the schematic of the LDPC decoder, as well as the SPA, Log-SPA and MSA in the following subsections.

2.3.2.1 Decoder Schematic

Figure 2.4 provides the schematic of an LDPC encoder and the corresponding decoder. In Figure 2.4, the LDPC encoder generates the codeword $\mathbf{c}$ according to (2.3), which is then transmitted using Binary Phase Shift Keying (BPSK) modulation over an Additive White Gaussian Noise (AWGN) channel. The LDPC decoder can be viewed as a serial concatenation of two Soft-Input Soft-Output (SISO) constituent decoders, which are connected using the interleaver π, which permutes the order of the edges connecting the CNs to the VNs, as well as the de-interleaver π^{-1} for performing the opposite operation. The SISO decoder for performing the computations of all CNs may be referred as the parity-Check Node Decoder (CND), while the other SISO decoder is referred to as the Variable Node Decoder (VND), which performs the computations of the VNs.

The CND and VND may employ soft messages in the form of probabilities or LLRs, to represent the uncertainty introduced by the noisy channel. When a hard decision is used to convert a soft information sequence, Figure 2.4 applies a diacritical hat ˆ to the notation of the corresponding bit vector, in order to indicate the resultant estimation. Moreover, a sequence of soft information is indicated by applying a diacritical tilde ˜ to the notation of the corresponding bit vector. The superscripts a, e and p are used to represent *a priori*, extrinsic and *a posteriori* soft information sequences, respectively. The feedback provided to one SISO decoder from the other is referred to as the *a*

priori probabilities or LLRs, while the input from the channel is referred to as the channel probabilities or LLRs, or are also known as the *intrinsic* probabilities or LLRs. The output provided from one SISO decoder to the other is referred to as the *extrinsic* probabilities or LLRs. The combination of all soft information contained in the channel, *a priori* and extrinsic probabilities or LLRs is referred to as the *a posteriori* probabilities or LLRs, which are used to determine whether each encoded bit has a binary value of 1 or 0.

The flow diagram of Figure 2.5 shows how the decoding algorithms for LDPC codes work. During the first initialization step, all the *a priori* inputs of VNs are set to indicate that no *a priori* information is available. As shown in Figure 2.5, the iterative decoding process continues until a pre-defined number of iterations is completed or until the validity of the recovered bit sequence $\hat{\mathbf{c}}$ is confirmed by the acquisition of a zero-valued syndrome, according to (2.10). If the hard decisions $\hat{\mathbf{c}}$ satisfy (2.10), the decoding is terminated immediately and the decoder decides that a valid decoded codeword has been found. By contrast, if the pre-defined limit upon the number of iterations is reached without finding a valid codeword, then a decoding failure is declared and a tentative decoded result $\hat{\mathbf{c}}$ is output. Either way, the systematic output corresponding to the transmitted information bits $\mathbf{u}$ is obtained according to (2.11).

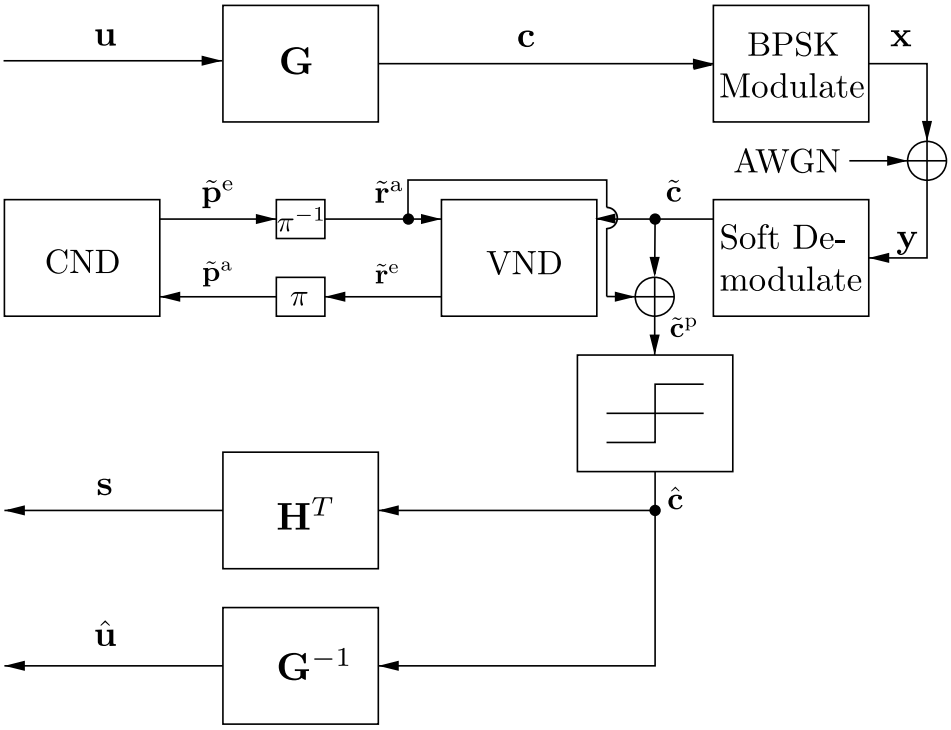


Figure 2.4: The schematic of an LDPC code, which comprises an encoder and an iterative decoder.

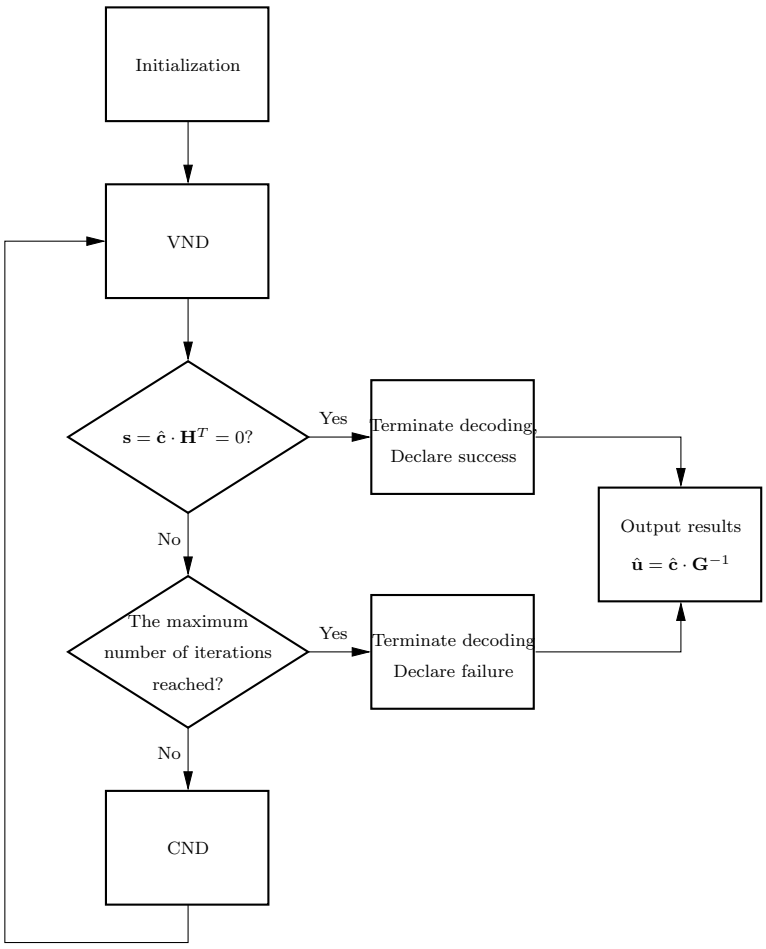


Figure 2.5: The flow diagram of iterative decoding.

2.3.2.2 Sum-Product Algorithm

As introduced previously, the SPA uses probabilities to represent the soft messages, denoted as P_A, where the subscript A corresponds to a specific port of a VN or CN that the probability is passed along. More specifically, the probability P_A quantifies the probability that the corresponding encoded bit has a binary value of 1. This notation will be used throughout all the following discussions, if not otherwise specified. In the case where BPSK modulation maps bits to symbols according to $\mathbf{x} = 1 - 2\mathbf{c}$, we have an effective throughput of $\eta = R$ bits per symbol and an energy per bit of $E_\mathrm{b} = \frac{1}{\eta}$. The AWGN channel of Figure 2.4 provides the received symbols according to $\mathbf{y} = \mathbf{x} + \mathbf{n}$, where $\mathbf{n}$ is complex Gaussian distributed with a mean of zero and a variance of $\frac{N_0}{2}$ in each of the real and imaginary parts. Here, N_0 is the noise power spectral density, giving an Signal to Noise Ratio (SNR) of $\frac{1}{N_0}$ and an SNR *per bit* of $E_\mathrm{b}/N_0 = \frac{1}{\eta N_0}$. The

BPSK demodulator provides the probabilities $\tilde{\mathbf{c}}$ according to [67, 66, 11],

$$\tilde{\mathbf{c}} = \frac{1}{1 + \exp(4\eta \cdot \frac{E_\mathrm{b}}{N_0} \cdot \mathrm{Re}(\mathbf{y}))}. \tag{2.12}$$

At the start of the iterative decoding process, the *a priori* probabilities provided to the VND are all set to 0.5. Whenever a CN or VN is activated, the probability associated with an encoded bit is output on each of its ports. The probabilities that are output on a particular port of a CN or VN are calculated as a function of the probabilities that are input to all of the *other* ports of that node, via the edges of the factor graph. The only exception to this is the probability that is output on the port of the VN connected to the channel, which is calculated as a function of the probabilities that are input to *all* ports of that VN, including the port connected to the channel. In the case of CNs and VNs, an output probability P_C that is obtained by combing two input probabilities P_A and P_B as given by

$$P_C = f_\mathrm{CN}(P_A, P_B) = P_A(1 - P_B) + P_B(1 - P_A), \tag{2.13}$$

$$P_C = f_\mathrm{VN}(P_A, P_B) = \frac{P_A P_B}{P_A P_B + (1 - P_A)(1 - P_B)}, \tag{2.14}$$

respectively [8, 11, 66]. These functions may be extended for more than two input probabilities, by recursively substituting (2.13) or (2.14) into itself. For example, three input probabilities may be combined according to

$$\begin{aligned} P_D &= f_\mathrm{CN}(P_A, P_B, P_C) \\ &= P_A(1 - f_\mathrm{CN}(P_B, P_C)) + f_\mathrm{CN}(P_B, P_C) \cdot (1 - P_A), \end{aligned} \tag{2.15}$$

$$\begin{aligned} P_D &= f_\mathrm{VN}(P_A, P_B, P_C) \\ &= \frac{P_A \cdot f_\mathrm{VN}(P_B, P_C)}{P_A \cdot f_\mathrm{VN}(P_B, P_C) + (1 - P_A)(1 - f_\mathrm{VN}(P_B, P_C))} \\ &= \frac{P_A P_B P_C}{P_A P_B P_C + (1 - P_A)(1 - P_B)(1 - P_C)}. \end{aligned} \tag{2.16}$$

Each *a posteriori* probability is obtained as the function f_VN of the corresponding channel probability and all the *a priori* probabilities provided to the corresponding VN. As shown in Figure 2.4, each a posteriori probability is converted into a decoded bit having the binary value of 1 if the probability is greater than 0.5, or to the binary value of 0 otherwise.

2.3.2.3 Log-SPA and Min-Sum Algorithm

In contrast to the SPA, Log-SPA and MSA uses LLRs instead of probabilities to represent the soft information, which are defined as $\tilde{x} = \ln \frac{1-P_A}{P_A}$ [11, 64]. The BPSK demodulator provides the LLRs $\tilde{\mathbf{c}}$ according to [67, 66, 11]

$$\tilde{\mathbf{c}} = 4\eta \cdot \frac{E_\mathrm{b}}{N_0} \cdot \mathrm{Re}(\mathbf{y})$$

At the start of the iterative decoding process, the *a priori* LLRs provided to the VND are all set to zero, which corresponds to $P_A = 0.5$.

At the end of the iterative decoding process, the LLRs are converted into hard decisions, according to

$$\hat{c}_k = \begin{cases} 0, & \tilde{c}_k^\mathrm{p} > 0 \\ 1, & \tilde{c}_k^\mathrm{p} \leq 0 \end{cases} . \tag{2.17}$$

In the Log-SPA and MSA, the functions f_CN and f_VN are modified as discussed in the following subsections. These modifications have the advantage of using the lower complexity min and sum operations, rather than the sum and product operations of the SPA. Also, LLRs have a lower dynamic range than probabilities, which makes hardware implementation simpler.

Check Node Decoding: As shown in Figure 2.4, the CND and VND are employed to convert sequences of *a priori* LLRs into sequences of extrinsic LLRs. The CND employs the boxplus operator [69, 8] to combine LLRs according to

$$\begin{aligned} \tilde{p}_3^\mathrm{e} = f_\mathrm{CN}(\tilde{p}_1^\mathrm{a}, \tilde{p}_2^\mathrm{a}) = \tilde{p}_1^\mathrm{a} \boxplus \tilde{p}_2^\mathrm{a} &= \mathrm{sign}(\tilde{p}_1^\mathrm{a})\mathrm{sign}(\tilde{p}_2^\mathrm{a}) \min(|\tilde{p}_1^\mathrm{a}|, |\tilde{p}_2^\mathrm{a}|) \\ &\quad + \log\left(1 + e^{-|\tilde{p}_1^\mathrm{a} + \tilde{p}_2^\mathrm{a}|}\right) - \log\left(1 + e^{-|\tilde{p}_1^\mathrm{a} - \tilde{p}_2^\mathrm{a}|}\right) \quad (2.18) \\ &\approx \mathrm{sign}(\tilde{p}_1^\mathrm{a})\mathrm{sign}(\tilde{p}_2^\mathrm{a}) \min(|\tilde{p}_1^\mathrm{a}|, |\tilde{p}_2^\mathrm{a}|). \quad (2.19) \end{aligned}$$

Here, we refer to an LDPC decoder that adopts the full correction terms of (2.18) as the 'Log-SPA'. By contrast, we use 'MSA' to refer to a decoder that neglects the logarithmic correction terms of (2.18), as shown in (2.19). The correction terms on the right of (2.18) grant the Log-SPA a superior error correction performance to the MSA, at the cost of having a high computational complexity [8, 70]. As a compromise, the correction terms can be approximated by several means [67, 66, 11]. In particular, LUTs can be utilized in a fixed-point decoder, in order to pre-compute the correction terms, as will be investigated in Chapters 3 and 4.

The CND converts the *a priori* LLR sequence $\tilde{\mathbf{p}}^\mathrm{a}$ into the extrinsic LLR sequence $\tilde{\mathbf{p}}^\mathrm{e}$,

as shown in Figure 2.4. This is achieved by decomposing $\tilde{\mathbf{p}}^{\mathrm{a}}$ into the sets of d_{c} LLRs that are associated with each individual CN c. Each set $\{\tilde{p}_1^{\mathrm{a}}, \tilde{p}_2^{\mathrm{a}}, \ldots, \tilde{p}_L^{\mathrm{a}}\}$ comprises $L = d_{\mathrm{c}}$ LLRs and is converted into the corresponding set of extrinsic LLRs $\{\tilde{p}_1^{\mathrm{e}}, \tilde{p}_2^{\mathrm{e}}, \ldots, \tilde{p}_L^{\mathrm{e}}\}$, which may be achieved by recursively substituting (2.18) into itself, owing to the associative property of the boxplus $\boxplus$ operator [8]. For example, three input probabilities may be combined according to

$$\begin{aligned} \tilde{p}_4^{\mathrm{e}} &= f_{\mathrm{CN}}(\tilde{p}_1^{\mathrm{e}}, \tilde{p}_2^{\mathrm{e}}, \tilde{p}_3^{\mathrm{e}}) \\ &= (\tilde{p}_1^{\mathrm{e}} \boxplus \tilde{p}_2^{\mathrm{a}}) \boxplus \tilde{p}_3^{\mathrm{a}}. \end{aligned} \tag{2.20}$$

The so-called Forward Backward Algorithm (FBA) have been employed to obtain the set of extrinsic LLRs in[11, 64], according to

$$\tilde{p}_k^{\mathrm{e}} = \begin{cases} \tilde{b}_2 & \text{if } k = 1 \\ \tilde{f}_{k-1} \boxplus \tilde{b}_{k+1} & \text{if } 1 < k < L \\ \tilde{f}_{L-1} & \text{if } k = L \end{cases}, \tag{2.21}$$

where the LLRs $\tilde{f}_k$ may be calculated using a forwards recursion, according to

$$\tilde{f}_k = \begin{cases} \tilde{p}_1^{\mathrm{a}} & \text{if } k = 1 \\ \tilde{p}_k^{\mathrm{a}} \boxplus \tilde{f}_{k-1} & \text{if } k > 1 \end{cases}, \tag{2.22}$$

and the LLRs $\tilde{b}_k$ may be calculated using a backwards recursion, according to

$$\tilde{b}_k = \begin{cases} \tilde{p}_L^{\mathrm{a}} & \text{if } k = L \\ \tilde{p}_k^{\mathrm{a}} \boxplus \tilde{b}_{k+1} & \text{if } k < L \end{cases}. \tag{2.23}$$

Finally, these sets of extrinsic LLRs are concatenated to obtain the extrinsic LLR sequence $\tilde{\mathbf{p}}^{\mathrm{e}} = [\tilde{p}_1^{\mathrm{e}}, \tilde{p}_2^{\mathrm{e}}, \ldots, \tilde{p}_L^{\mathrm{e}}]$.

Variable Node Decoding: Similarly, the VND combines the *a priori* LLR sequence $\tilde{\mathbf{r}}^{\mathrm{a}}$ and the channel LLR sequence $\tilde{\mathbf{c}}$ in order to obtain the extrinsic LLR sequences $\tilde{\mathbf{r}}^{\mathrm{e}}$ and the *a posteriori* LLR sequence $\tilde{\mathbf{c}}^{\mathrm{p}}$, as shown in Figure 2.4. This is achieved by decomposing $\tilde{\mathbf{r}}^{\mathrm{a}}$ into the sets of d_{v} LLRs that are associated with each individual VN v. Each set is appended with the LLR $\tilde{c}_{\mathrm{v}}$ from $\tilde{\mathbf{c}}$ that corresponds to the v^{th} variable node, in order to obtain $\{\tilde{x}_1^{\mathrm{a}}, \tilde{x}_2^{\mathrm{a}}, \ldots, \tilde{x}_L^{\mathrm{a}}\} = \{\tilde{r}_1^{\mathrm{a}}, \tilde{r}_2^{\mathrm{a}}, \ldots, \tilde{r}_{d_{\mathrm{v}}}^{\mathrm{a}}, \tilde{c}_{\mathrm{v}}\}$, where $L = d_{\mathrm{v}} + 1$. These are then converted into the set of extrinsic LLRs $\{\tilde{x}_1^{\mathrm{e}}, \tilde{x}_2^{\mathrm{e}}, \ldots, \tilde{x}_{d_{\mathrm{v}}}^{\mathrm{e}}, \tilde{x}_L^{\mathrm{e}}\}$ by recursively substituting $\tilde{x}_3^{\mathrm{e}} = f_{\mathrm{VN}}(\tilde{x}_1^{\mathrm{a}}, \tilde{x}_2^{\mathrm{a}}) = \tilde{x}_1^{\mathrm{a}} + \tilde{x}_2^{\mathrm{a}}$ into itself. For example, three input probabilities

may be combined according to

$$\begin{aligned} \tilde{x}_4^{\mathrm{e}} &= f_{\mathrm{CN}}(\tilde{x}_1^{\mathrm{e}}, \tilde{x}_2^{\mathrm{e}}, \tilde{x}_3^{\mathrm{e}}) \\ &= (\tilde{x}_1^{\mathrm{a}} + \tilde{x}_2^{\mathrm{a}}) + \tilde{x}_3^{\mathrm{a}}. \end{aligned} \tag{2.24}$$

Similarly to the CND decoding, these may be converted using FBA of (2.21), (2.22) and (2.23), but employing the plus operator + instead of the boxplus operator $\boxplus$, as shown in (2.25) (2.26) and (2.27). Namely the function f_{VN} simply sums two LLRs as $\tilde{x}_3^{\mathrm{e}} = f_{\mathrm{VN}}(\tilde{x}_1^{\mathrm{a}}, \tilde{x}_2^{\mathrm{a}}) = \tilde{x}_1^{\mathrm{a}} + \tilde{x}_2^{\mathrm{a}}$. These sets of extrinsic LLRs are then decomposed in order to obtain LLRs for the sequences $\tilde{\mathbf{r}}^{\mathrm{e}}$ and $\tilde{\mathbf{c}}^{\mathrm{p}}$. More Specifically, the extrinsic sequence of LLRs that are fedback to the CND is obtained as $\tilde{\mathbf{r}}^{\mathrm{e}} = \{\tilde{x}_1^{\mathrm{e}}, \tilde{x}_2^{\mathrm{e}}, \ldots, \tilde{x}_{d_{\mathrm{v}}}^{\mathrm{e}}\}$. Meanwhile, the *a posteriori* LLR $\tilde{\mathbf{c}}^{\mathrm{p}}$sequence are obtained by adding the corresponding *a priori* and extrinsic LLR sequences $\tilde{\mathbf{x}}^{\mathrm{a}}$ and $\tilde{\mathbf{x}}^{e}$, according to $\tilde{c}^{\mathrm{p}} = \tilde{x}_L^{\mathrm{a}} + \tilde{x}_L^{\mathrm{e}} = \tilde{c}_{\mathrm{v}} + \tilde{x}_L^{\mathrm{e}}$. Note that unlike the CND, the operation of the VND is identical in the Log-SPA and the MSA.

$$\tilde{x}_k^{\mathrm{e}} = \begin{cases} \tilde{b}_2 & \text{if } k = 1 \\ \tilde{f}_{k-1} + \tilde{b}_{k+1} & \text{if } 1 < k < L \\ \tilde{f}_{L-1} & \text{if } k = L \end{cases}, \tag{2.25}$$

$$\tilde{f}_k = \begin{cases} \tilde{x}_1^{\mathrm{a}} & \text{if } k = 1 \\ \tilde{x}_k^{\mathrm{a}} + \tilde{f}_{k-1} & \text{if } k > 1 \end{cases}, \tag{2.26}$$

$$\tilde{b}_k = \begin{cases} \tilde{x}_L^{\mathrm{a}} & \text{if } k = L \\ \tilde{x}_k^{\mathrm{a}} + \tilde{b}_{k+1} & \text{if } k < L \end{cases}. \tag{2.27}$$

2.4 EXIT Charts

As discussed in Section 2.3, the error-correcting performance of LDPC codes is affected by many factors, including node degree distribution, code length, coding rate, channel SNR and the number of iterations performed. However, it is not easy to jointly optimize all these factors using a trial and error approach. Motivated by this, previous efforts have proposed techniques for analysing the error-correcting performance of LDPC codes, such as density evolution [71]. However, EXIT charts were proposed in [49] as an alternative method for analysing the convergence behaviour of iterative decoders, which offers more convenience and accuracy, since EXIT charts analysis does not require the iterative decoding process to be simulated. This approach investigates the characteristics of each constituent SISO decoder separately, before considering their

combination, in order to predict the iterative decoding convergence performance. In this section, we will detail the EXIT charts of LDPC codes. In Section 2.4.1, we will begin by describing the mutual information, which is used to quantify the quality of the iteratively exchanged soft information. Following this, the application of EXIT charts to LDPC decoders will be exemplified in Section 2.4.2.

2.4.1 Mutual Information

The so-called 'Mutual Information' (MI) quantifies how well a sequence of probabilities or LLRs represents the corresponding sequence of bits. More explicitly, MI quantifies the reliability of the information conveyed by the sequence of probabilities or LLRs. MI is used when plotting an EXIT function, which characterizes the quality of the extrinsic information produced by a SISO component decoder, as a function of the quality of the *a priori* information it is provided with. More explicitly, the generation of artificial *a priori* messages having a particular *a priori* MI is performed at the input at the SISO decoder, while the *extrinsic* MI is quantified at the output of the component decoder. In this way, the EXIT function describes how the *extrinsic* MI is dependent on the *a priori* MI, as well as on the MI of the channel information, if appropriate.

In [49, 50], it was shown that a sequence of artificial *a priori* LLRs can be generated as independent random variables having an identical Gaussian distribution. More specifically, an *a priori* LLR $\tilde{x}$ pertaining to the bit x can be obtained according to

$$\tilde{x} = \frac{\sigma_{\mathrm{A}}^2}{2} \cdot (1 - 2x) + n_{\mathrm{A}}, \tag{2.28}$$

where n_{A} is a realization of Gaussian random variable having the variance σ_{A}^2 and a mean of zero. The corresponding conditional probability density function is

$$p_{\mathrm{A}}(\tilde{x}|x) = \frac{e^{-\frac{(\tilde{x}-(\sigma_{\mathrm{A}}^2/2)\cdot(1-2x))^2}{2\sigma_{\mathrm{A}}^2}}}{\sqrt{2\pi}\sigma_{\mathrm{A}}}. \tag{2.29}$$

The *a priori* MI I_{A} [49, 50], $0 \leq I_{\mathrm{A}} \leq 1$, can be calculated by

$$I_{\mathrm{A}} = 0.5 \cdot \sum_{x=0,1} \int_{-\infty}^{+\infty} p_{\mathrm{A}}(\tilde{x}|x) \cdot \log_2 \frac{2 \cdot p_{\mathrm{A}}(\tilde{x}|x)}{p_{\mathrm{E}}(\tilde{x}|x=1) + p_{\mathrm{A}}(\tilde{x}|x=0)} d\tilde{x}. \tag{2.30}$$

Here, the MIs have values in the range of $[0, 1]$, where '0' indicates that the LLRs contain no information about the corresponding bits, while '1' indicates perfect MI. When substituting (2.29) into (2.30), I_{A} becomes a function of σ_{A}, which does not have

a closed form, but which can be represented by

$$I_{\mathrm{A}} = J(\sigma_{\mathrm{A}}). \tag{2.31}$$

This function is reversible and the result can represented by

$$\sigma_{\mathrm{A}} = J^{-1}(I_{\mathrm{A}}). \tag{2.32}$$

The relationship can be used to identify the σ_{A} required to model the desired I_{A}, which may be then be used to generate artificial *a priori* LLRs according to (2.28).

The extrinsic MI I_{E}, $0 \leq I_{\mathrm{E}} \leq 1$, can be quantified by estimating the distributions of the extrinsic LLRs $p_{\mathrm{E}}(\tilde{x}|x)$, where $x \in \{0, 1\}$. The extrinsic MI I_{E} may then be obtained according to

$$I_{\mathrm{E}} = 0.5 \cdot \sum_{x=0,1} \int_{-\infty}^{+\infty} p_{\mathrm{E}}(\tilde{x}|x) \cdot \log_2 \frac{2 \cdot p_{\mathrm{E}}(\tilde{x}|x)}{p_E(\tilde{x}|x=1) + p_{\mathrm{E}}(\tilde{x}|x=0)} d\tilde{x}. \tag{2.33}$$

Note that (2.33) allows the measurement of mutual information, provided that knowledge of the transmitted bits x is available. This measuring method therefore is referred to as the histogram based method. The histogram method provides more precise measurements of MI, when the bit and LLR sequences are long, owing to the more precise estimation of the histogram. In [50], it is suggested that a sequence length of 10^4 bits is sufficient for achieving accurate measurements using the histogram method. As an alternative to the histogram method, the averaging method [49, 50] does not require knowledge of the corresponding bits, but it assumes that the LLRs are obtained using optimal decoders and hence convey an appropriate level of confidence in the soft decisions. The averaging method calculates the MI of a sequence comprising N LLRs $[\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_N]$ according to

$$I_{\mathrm{E}} = 1 - \frac{1}{N} \sum_{k=1}^{N} \sum_{x=0,1} p_{\mathrm{E}}(x_k|\tilde{x_k}) \cdot \log_2 \frac{1}{p(x_k|\tilde{x_k})}, \tag{2.34}$$

where $p(0|\tilde{x}) = \frac{e^{\tilde{x}}}{1+e^{\tilde{x}}}$ and $p(1|\tilde{x}) = \frac{1}{1+e^{\tilde{x}}}$.

In summary, the EXIT function of a constituent decoder can be obtained by considering a range of values for $I_{\mathrm{A}} \in [0, 1]$. For each value of I_{A}, we generate the sequence of *a priori* LLRs according to (2.28) and (2.32). This is input into the decoder, in order to obtain the corresponding sequence of extrinsic LLRs. Following this, the extrinsic MI of the output can be quantified as I_{E} according to (2.33) and (2.34). In this way,

the EXIT characteristics of the decoder can be described by I_{E} as a function of I_{A}, or a function of I_{E} and channel SNR.

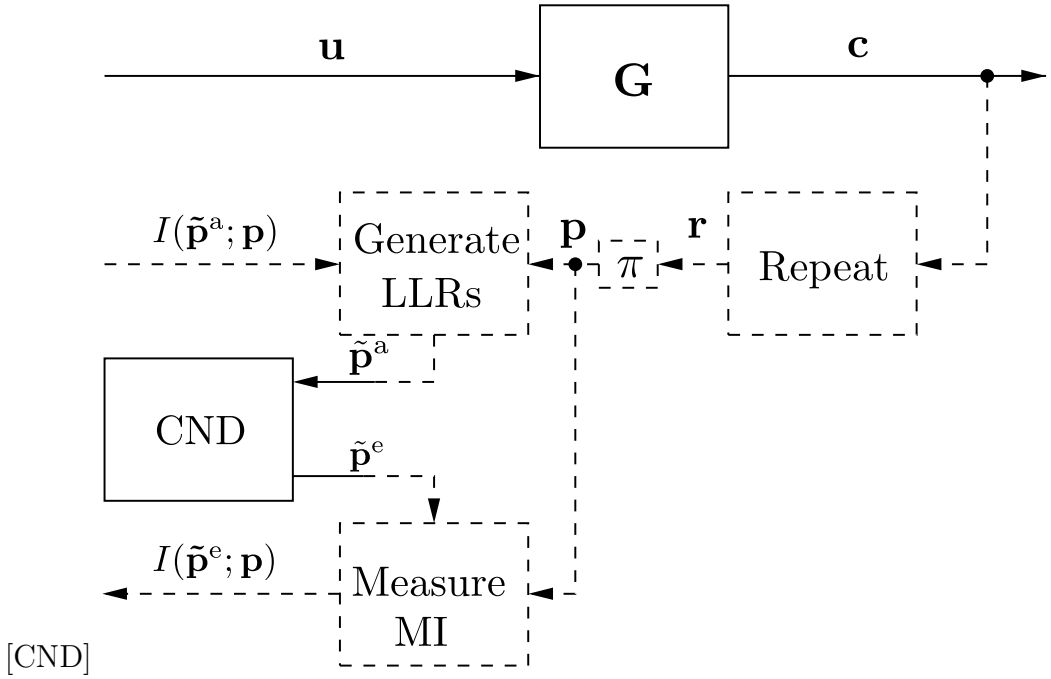
2.4.2 EXIT Chart Analysis of LDPC Codes

As previously discussed in Section 2.4.1, EXIT chart analysis has the advantages of characterizing the error correction capability of an LDPC decoder without the requirement for performing iterative decoding and focusing on a single SNR in contrast to BER analysis. Furthermore, some EXIT chart investigations are independent of both the LDPC block length and the number of decoding iterations performed. As a result, EXIT charts are usually a fast and convenient way to investigate the characteristics of an LDPC decoder's error correction capability. In the following discussions, we will give a detailed description about how to obtain the EXIT functions for an LDPC code, using the schematic of Figure 2.6, which are adapted from that of Figure 2.4.

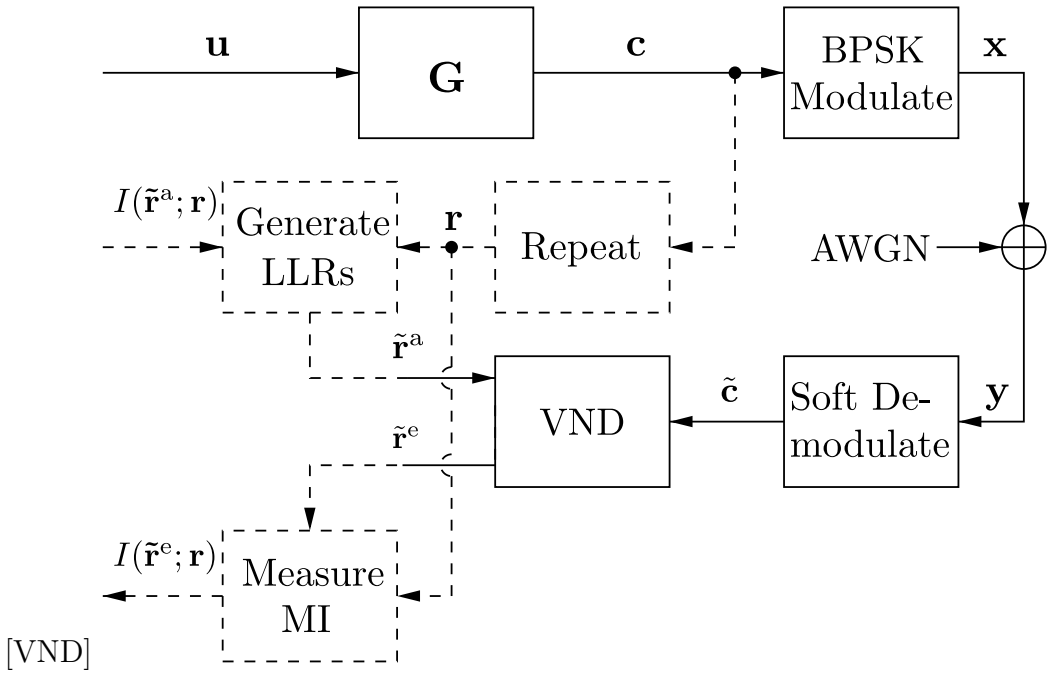
In Figure 2.6, the reliability of the extrinsic LLR sequences $\tilde{\mathbf{r}}^{\mathrm{e}}$ and $\tilde{\mathbf{p}}^{\mathrm{e}}$ may be quantified by their MIs $I(\tilde{\mathbf{r}}^{\mathrm{e}};\mathbf{r})$ and $I(\tilde{\mathbf{p}}^{\mathrm{e}};\mathbf{p})$, respectively. The iterative exchange of increasingly more reliable extrinsic information between the CND and VND can be characterized using EXIT charts. More specifically, as the dashed boxes and paths of Figure 2.6 show, the v^{th} bit of the encoded sequence $\mathbf{c} = [c_1, c_2, \cdots, c_N]$ is repeated d_{v} times to form the longer N_{edge}-bit sequence $\mathbf{r}$, where we have $r_{(\mathrm{v}-1)\cdot d_{\mathrm{v}}+1} = r_{(\mathrm{v}-1)\cdot d_{\mathrm{v}}+2} = \cdots = r_{\mathrm{v}\cdot d_{\mathrm{v}}} = c_{\mathrm{v}}$ for the v^{th} VN in a regular LDPC code[4]. After the permutation of the interleaver π, the bit sequence $\mathbf{p}$ is obtained from the bit sequence $\mathbf{r}$, as described in Section 2.3.2. The CND EXIT chart simulation of Figure 2.4.2 generates Gaussian-distributed *a priori* LLR sequences $\tilde{\mathbf{p}}^{\mathrm{a}}$ having MIs $I(\tilde{\mathbf{p}}^{\mathrm{a}};\mathbf{p})$ across the range of $[0, 1]$. Following the operation of the CND, the MI $I(\tilde{\mathbf{p}}^{\mathrm{e}};\mathbf{p})$ of the extrinsic LLR sequence $\tilde{\mathbf{p}}^{\mathrm{e}}$ is measured using the MI histogram or averaging method, as described in Section 2.4.1. Note however that as shown in Figure (2.7(c)), it is conventional to plot these CND EXIT functions on inverted axes, namely with $I(\tilde{\mathbf{p}}^{\mathrm{a}};\mathbf{p})$ as a function of $I(\tilde{\mathbf{p}}^{\mathrm{e}};\mathbf{p})$. Also note that the CND EXIT function is independent of the channel SNR. Similarly, the simulation of Figure 2.4.2 is used to plot the EXIT function of the VND. Since the VND is fed directly from the channel output, the VND EXIT function is associated with the particular channel SNR per bit E_{b}/N_0 [49, 50]. This is illustrated in the EXIT functions of Figures 2.7(a) and 2.7(b), which are combined for the different AWGN SNRs of 0 dB and 3 dB.

As described in [49, 72], a CND and a VND EXIT function may be plotted in the

[4]The extension to an irregular LDPC code is trivial.



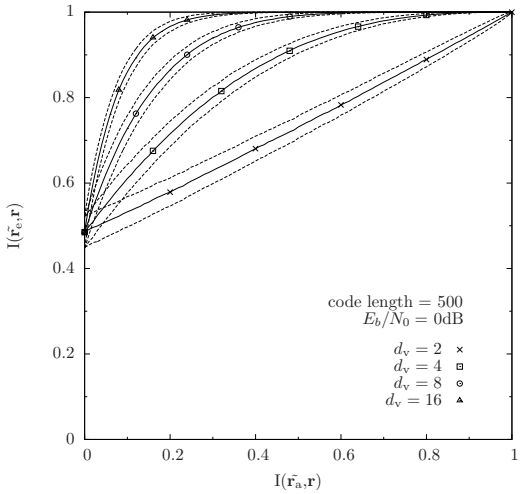
(a)



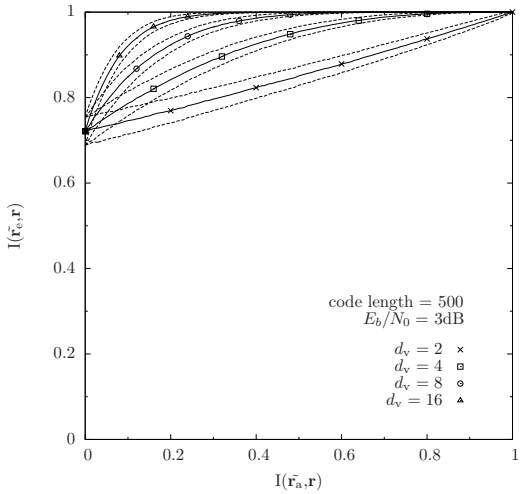
(b)

Figure 2.6: Adapted schematics used to depict the generations of EXIT functions for the (a) the CND and (b) the VND.

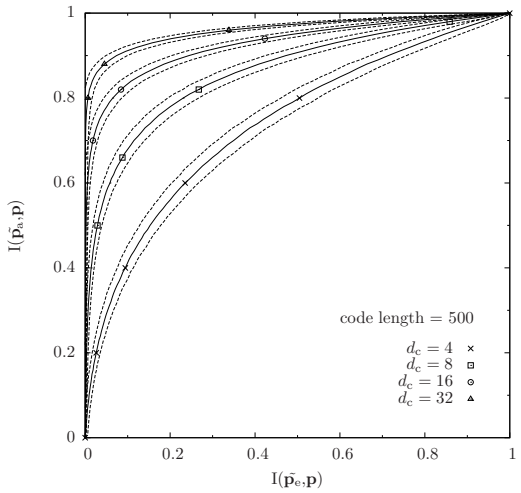
same figure to obtain an EXIT chart as shown in Figure 2.8. When the channel E_{b}/N_0 is sufficiently high, the VND EXIT function will be above the CND EXIT function,



(a) VND (0 dB)



(b) VND (3 dB)



(c) CND

Figure 2.7: EXIT functions and bands of a regular LDPC VND and CND having lengths of 500 and degree $d_v \in \{2, 4, 8, 16\}$, $d_c \in \{4, 8, 16, 32\}$, when the channel E_b/N_0 equals to 0 and 3 dB.

and they will not intersect before reaching the $(1, 1)$ point of perfect convergence to a vanishingly low BER. The emergence of the resultant so-called open EXIT tunnel implies that iterative decoding convergence towards a low BER may indeed be achieved.

The iterative exchange of increasingly more reliable extrinsic information between the CND and VND can be characterised using EXIT charts. These employ the MI $I(\tilde{\mathbf{x}}; \mathbf{x})$ to quantify the reliability of the information in the LLR sequence $\tilde{\mathbf{x}}$ about the contents of the bit sequence $\mathbf{x}$. Here, the MIs have values in the range of $[0, 1]$, where '0' indicates that the LLRs contain no information about the corresponding bits, while '1'

indicates perfect MI. More specifically, an EXIT function plots the MI of the extrinsic LLRs $\tilde{\mathbf{x}}^{\mathrm{e}}$ that are output by a decoder as a function of the MI of the corresponding input *a priori* LLRs.

2.5 Simulation Results

In this section, we discuss the EXIT charts of regular binary LDPC codes. Firstly, Section 2.5.1 shows the EXIT functions of the CND and the VND, which are obtained according to Figure 2.6. Note that the CND and the VND EXIT functions are illustrated with dashed bands in Figures 2.7, 2.8, 2.9 and 2.10, which demonstrate the standard deviations of the MIs obtained through simulations. These EXIT bands will be discussed in detail in Section 2.5.2. Subsequently, the capability of EXIT charts to predict the convergence behaviour of the LDPC decoder is demonstrated, by combing so-called 'snapshot' decoding trajectories with the EXIT functions. Section 2.5.3 compares the performance of the Log-SPA and MSA algorithms. Finally, a BER versus E_{b}/N_0 plot is used to validate of the predication based on the EXIT charts. All the parameters used in the simulations are listed in Table 2.3.

Table 2.3: Parameters used for EXIT chart simulations of LDPC codes.

code type	regular LDPC
code length N	$500, 5000$
coding rate R	0.5
node degrees	CND $d_{\mathrm{c}} = 4, 6, 8, 16, 32$ VND $d_{\mathrm{v}} = 2, 3, 4, 8, 16$
decoding algorithm	Log-SPA, MSA
modulation	BPSK
channel conditions	AWGN $E_{\mathrm{b}}/N_0 = 3\,dB$
mutual information measurement	histogram method
number of trajectories	10
maximum number of iterations	20

2.5.1 Effect of Node Degree

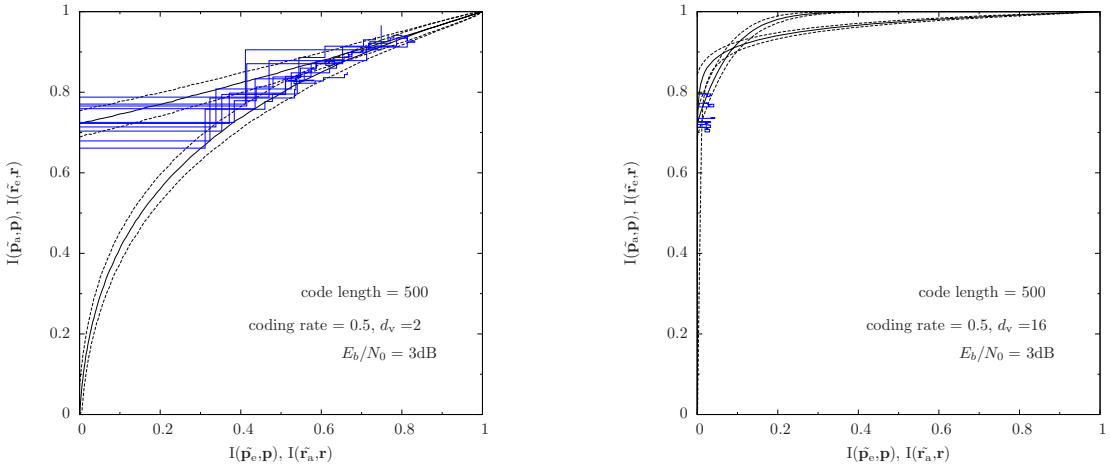
Figure 2.7(c) shows CND EXIT functions for a $R = \frac{1}{2}$ rate LDPC code having the block length of N =500 bits, having different degrees $d_{\mathrm{c}} = 4, 8, 16, 32$, where the corresponding VND EXIT functions are shown in Figures 2.7(a) and 2.7(b) for degree of $d_{\mathrm{v}} = 2, 4, 8, 16$ and for channel E_{b}/N_0 values of 0 dB and 3 dB, respectively. Based on these plots, it

can be concluded that all degrees result in increasing EXIT functions, but that lower degrees cause the EXIT function to ascend more slowly, for both the CND and the VND.

Figure 2.8(a) plots the VND EXIT functions for $d_v = 2$ and $E_b/N_0 = 3$ dB in the same plot as the CND EXIT function for $d_c = 4$. However, since the tunnel between the CND and VND EXIT functions is closed in the high MI area before the $(1, 1)$ point, the LDPC decoder will be unable to converge to a low BER, no matter how many decoding iterations are performed. Furthermore, Figure 2.8(a) shows 10 decoding trajectories, which represents the actual decoding behaviour of the LDPC decoder operating on 10 particular encoded blocks. Note that all of these trajectories stop at the intersection point of the CND and VND functions, resulting in the decoding failure for all the blocks. Similarly, Figure 2.8(b) illustrates that the decoding failure that is indicated by the early termination of 10 trajectories for the codes with $d_c = 32$ and $d_v = 16$. By comparison, Figure 2.9(a) shows 10 decoding trajectories of the codes with $d_c = 6$ and $d_v = 3$ successfully navigating through the open tunnels and reaching the $(1, 1)$ point. Note however that the VND having degrees of $d_v = 2$ exhibit a poor iterative decoding convergence [6], as will be revealed in the BER simulation of Section 2.5.4. Note that the simulations of Figure 2.8 can be repeated for different E_b/N_0 values such as $E_b/N_0 \in \{0, 6\}$ dB. In this way, the convergence behaviour and BER performance of an LDPC code can be precisely predicted by the EXIT charts. Note that when employing higher node degrees, the E_b/N_0 values required to achieve an open EXIT chart tunnel increases. Therefore, it can be predicted that when the node degrees are increased, the BER performance of regular LDPC codes degrades.

2.5.2 Effect of Code Length

The comparison between Figures 2.9(a) and 2.9(b) shows the impact of the block length N on the EXIT functions of the CND and VND, when the channel E_b/N_0 is 3 dB. The EXIT functions of the CND and of the VND of both codes are nearly identical, with the exception that the EXIT bands of the code having length of 5000 bits are much slimmer than the code having the length of 500 bits. As described in Section 5.3, the bands quantify the standard deviation in the MI measurements that are made for each block, where wider bands result in more variation between the trajectories of different blocks. However, one specific trajectory may still fall outside of the band, since around 68% of all the possible measured MI values are within one standard deviation from the mean MI. Moreover, each trajectory of the longer code is matched well with the corresponding EXIT curves exhibiting only a small amount of variation from block to



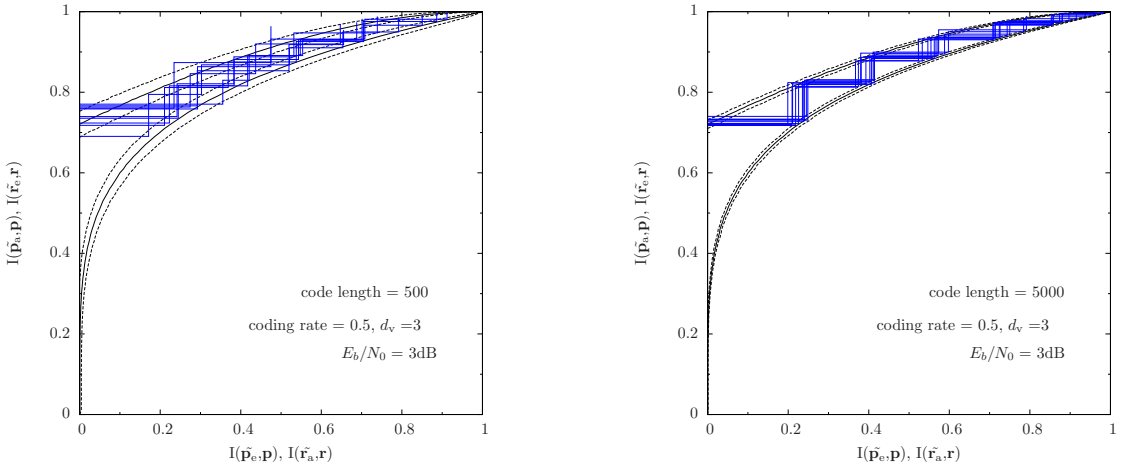
(a) (b)

Figure 2.8: EXIT bands and trajectories for regular half-rate LDPC codes having block length of 500 bits and degree of (a) $d_v = 2$ and (b) $d_v = 16$, when the channel E_b/N_0 equals 3 dB.

block. By contrast, the 10 trajectories differ widely along the tunnel for the shorter code. In the case where the tunnel is narrow, some trajectories may not navigate through a tunnel having wide bands, resulting in a gradually reducing BER as the channel E_b/N_0 is increased. By contrast, if the bands are narrow, then either the tunnel is open and all trajectories will navigate through to the $(1, 1)$ point, or the tunnel is closed and none of the trajectories will reach the $(1, 1)$ point. Owing to this, the BER reduces quickly as E_b/N_0 is increased, in the case of long blocks.

2.5.3 Effect of MSA and Log-SPA

Since the only difference between the MSA and the Log-SPA is in the operation of the CND, the VND has EXIT functions identical to those of Figures 2.7(a) and 2.7(b) for both algorithms. However, Figure 2.10 compares the CND EXIT functions of the MSA and Log-SPA algorithms, which are obtained using (2.18) and (2.19), respectively. Note that the EXIT functions corresponding to the sub-optimal MSA are slightly higher than those corresponding to the optimal Log-SPA for all LDPC codes having different degrees. This implies that a slightly higher channel SNR is required to create an open EXIT tunnel when the MSA is employed. These results demonstrate that the performance degradation imposed by sub-optimal decoding algorithms can be investigated by considering the narrowing of EXIT tunnels.



(a) (b)

Figure 2.9: EXIT bands and trajectories for regular half-rate LDPC codes having block lengths of (a) 500 and (b) 5000 bits and degrees $d_c = 6$ and $d_v = 3$, when the channel E_b/N_0 is 3 dB.

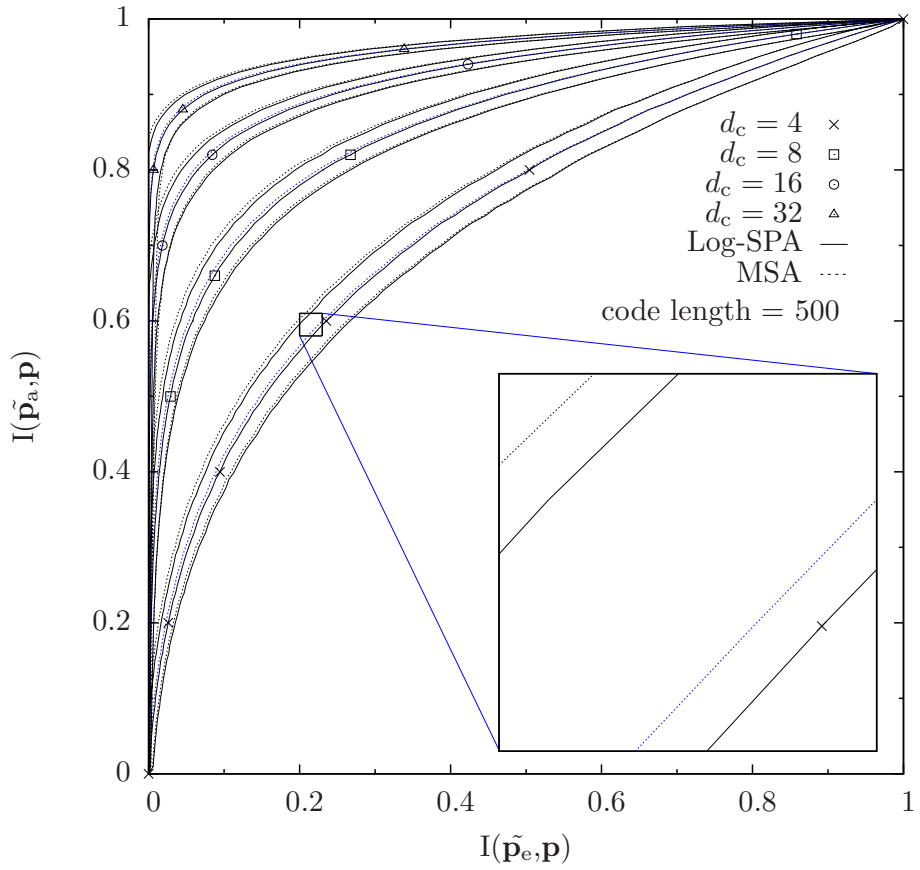


Figure 2.10: The EXIT function of LDPC CNDs having block length of 500 bits and various CN degrees of $d_c \in \{4, 8, 16, 32\}$.

2.5.4 BER Results

To demonstrate the conclusions and predictions obtained from the EXIT charts discussed in the previous subsections, a plot of BER versus E_b/N_0 is shown in Figure 2.11. This figure compares the error correcting performance of the Log-SPA and MSA, when employed by operating on various half-rate regular LDPC codes having different CN degrees $d_c \in \{4, 6, 8, 16, 32\}$ and VN degrees, $d_v \in \{2, 3, 4, 8, 16\}$. All other parameters used are the same as listed in Table 2.3.

In Figure 2.11, the $N = 5000$-bit regular LDPC code having a CN and VN degree of $d_c = 6$ and $d_v = 3$ outperforms all the $N = 500$-bit codes, when employing either the Log-SPA or MSA, matching the conclusion drawn based on the comparison between Figures 2.9(a) and 2.9(b). Furthermore, as discussed in Section 2.5.1, the $N = 500$-bit regular LDPC code having a VN degree of $d_v = 2$ exhibits a significantly slower convergence than all the other codes having degrees of $d_v \in \{3, 4, 8, 16\}$, which all exhibit the steep BER 'waterfall', but in different E_b/N_0 ranges. Explicitly, for the codes having degrees of $d_v \in \{3, 4, 8, 16\}$, the BER curves exhibit steep BER waterfalls at higher E_b/N_0 as predicted in Section 2.5.1. Additionally, the sub-optimal MSA algorithm can be seen to impose up to 1 dB of performance degradation on all the codes having various degrees and lengths, compared to the optimal Log-SPA algorithm as discussed in Section 2.5.3.

2.6 Conclusions

In this chapter, we have briefly reviewed the basics of channel codes, with a particular focus on block codes. After describing the matrix representation of block codes, we have discussed the factor graph of LDPC codes, as well as the decoding algorithms that rely on the factor graph. Among the numerous decoding algorithms that have been proposed for LDPC decoding, we have focused on the SPA, Log-SPA and MSA, which operate on soft information in the form of probabilities and LLRs, respectively. Each constituent decoder in the LDPC decoder outputs extrinsic soft information, which is fedback to the other constituent decoder, where it is used as *a priori* soft information. The two constituent decoders exchange *a priori* and extrinsic soft information iteratively, with the aim of converging upon a valid codeword. The reliability of the soft information may be quantified by the MI between the sequence of probabilities or LLRs and the corresponding sequence of bits. The relation between the extrinsic MI associated with the extrinsic probabilities or LLRs, and the *a priori* MI associated with the *a priori*

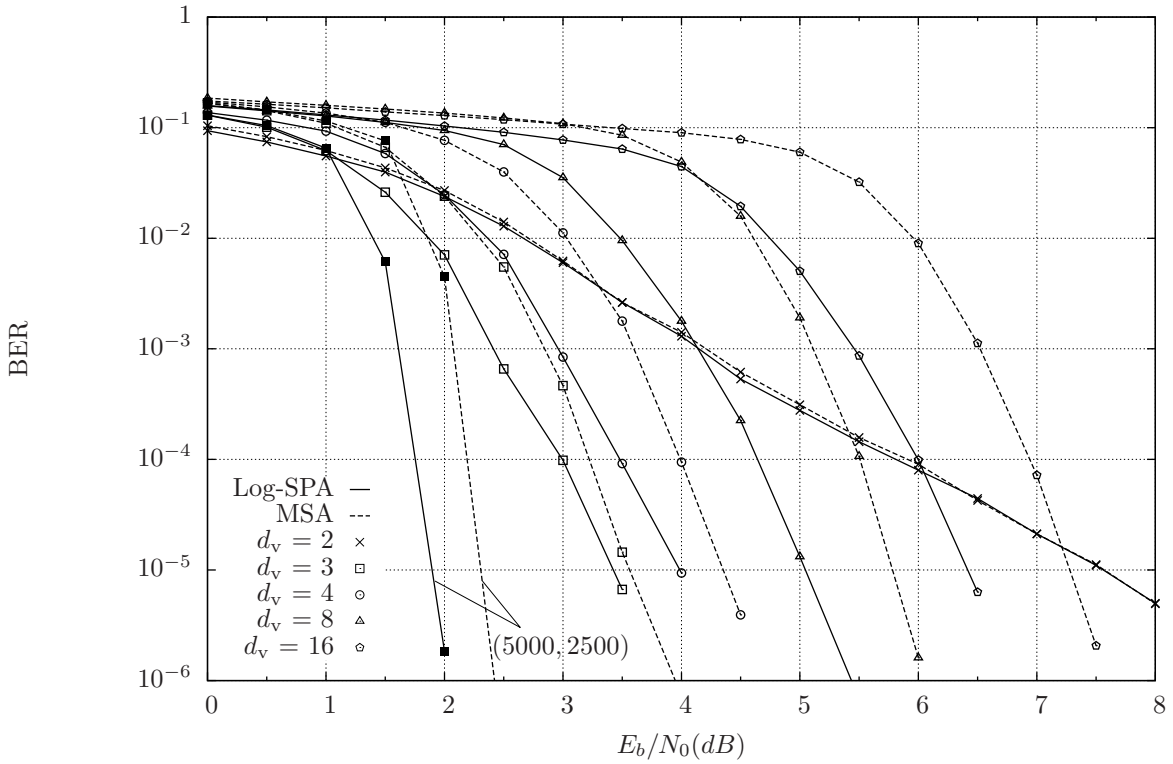


Figure 2.11: BER performance of regular LDPC codes having different degrees and code lengths. The left-most curve represents the (5000,2500) LDPC code, which has half coding rate and block length of 5000 bits, with $d_c = 6$ and $d_v = 3$. All the right-hand curves are half-rate LDPC codes having the block length of 500 bits, but various degrees.

probabilities or LLRs, can be obtained and visualized in EXIT charts. Simulations may be conducted to obtain EXIT charts for particular LDPC codes, which are utilized to characterize the error-correcting performance of the LDPC codes. We presented BER results to validate the predictions that we made using EXIT charts. In the forthcoming chapters, EXIT charts will be used as a design tool during the implementation of LDPC decoders.

Chapter 3

Minimum Bit Widths for Fixed-Point LDPC Decoder

3.1 Introduction

As discussed in Chapter 2, Low-Density Parity-Check (LDPC) codes [6] exhibit an outstanding error correction capability [73], when using the SPA. However, owing to the high dynamic range of its variables, the optimal Sum-Product Algorithm (SPA) requires a floating point implementation, which is associated with high complexity. Therefore, in order to mitigate this problem, the SPA can be transformed into the logarithmic domain, to yield the 'Log-SPA', which replaces the sum and product operations of the SPA with the lower-complexity 'boxplus' [69] and sum operations. Furthermore, by approximating the boxplus operator of the Log-SPA with the minimum-finding operation, the Min-Sum Algorithm (MSA) [67, 66, 11] achieves a further complexity reduction at the cost of marginally degrading the LDPC performance by a few tenths of a dB [67, 66, 64, 11, 15, 16, 12]. Both the Log-SPA and MSA operate on the basis of LLRs, which have a low dynamic range and can be represented by Fixed-Point (FP) numbers, having a low bit width. During typical LDPC iterative decoding processes, the LLRs typically only adopt values in the range -10 to 10, since this corresponds to bit probabilities in the range 0.00005 to 0.99995. Owing to this low dynamic range, the Log-Likelihood Ratios (LLRs) processed in practical implementation of LDPC decoders are typically represented by FP numbers. This is because FP numbers can efficiently represent values having a low dynamic range using a relatively small number of bits, which is referred to as the bit width. In a hardware implementation, it is this bit width that proportionally determines both the size of the memory required, as well as the area of the data path and hence the energy consumption imposed. It is therefore desirable to use a low bit

width. However, if the bit width is too low, then the dynamic range and/or the fidelity of the FP number representation will constrain the error correction capability of the LDPC decoder.

Table 3.1: Summary of previously proposed schemes.

Reference	LDPC code[1]	Decoding algorithm	Scheme[2]
[11]	R and IR	Log-SPA	U-Q 7 bits
		offset-MSA [11]	U-Q 4-6 bits
[10]	not mentioned	SPA	U-Q 12 bits
		SPA[3]	U-Q 6 bits
[14]	R	offset-MSA	FP(2,3,2/3)[4]
[15]	R and IR	offset-MSA	U-Q 4-6 bits
[13]	IR	Log-SPA	FP(4,4,4)
[12]	IR	offset-MSA	NU-Q 5 bits
[16]	IR	offset-MSA	U-Q 6 bits

In previous works, the authors of [10, 14, 11, 15, 13, 12, 16, 74] attempted to determine the minimum bit width that does not impose a significant performance degradation upon Fixed-point LDPC Decoders (LDPC-FDs). However, owing to the different algorithms and techniques that they considered, these efforts reached differing conclusions, as summarized in Table 3.1. The efforts of [10, 11, 12, 13, 14, 15, 16] [17, 18, 19, 20, 21, 22, 23, 24, 25, 26] relied upon BER simulations, which, have to consider a number of decoding iterations, as well as a range of channel SNRs, hence they are time consuming. This problem is further aggravated by the need to consider diverse combinations of various parameters, such as the LDPC code block length, the parity-Check Node (CN) and Variable Node (VN) degrees, the number of decoding iterations performed and the channel conditions. Since it is infeasible to exhaustively search all combinations of these parameters using BER simulations, there is no single authoritative study on the most desirable bit widths in LDPC decoders. Furthermore, BER simulations only reveal the magnitude of the performance degradation, without offering any insight into its causes, namely whether the accuracy of the integer or fraction part of the bit width is insufficient, for example. As a remedy, this chapter proposes the

[1]Here, 'R' represents regular LDPC codes constructed from a random unstructured Parity Check Matrix (PCM), which was designed to avoid length-4 cycles. Meanwhile 'IR' represents corresponding irregular LDPC codes.

[2]Here, 'U-Q' represents a uniform quantization of the LLRs in the Log-SPA and MSA, or of the Likelihood Ratios (LRs) of the SPA. Meanwhile, 'NU-Q' represents a non-uniform quantization. We define the notation FP(x, y, z) in Section 3.2.

[3]In fact, [10] adopts the "parity likelihood ratio algorithm", which is equivalent to the SPA.

[4]In this scheme, the fraction part of the FP representation is reduced from $W_{\mathrm{f}} = 3$ to $W_{\mathrm{f}} = 2$ when the integer part is clipped.

novel approach of using EXtrinsic Information Transfer (EXIT) charts [75] for the comprehensive investigation of the minimum bit widths for LDPC-FDs. In particular, the methods detailed in Chapter 2 for generating EXIT charts are extended to the case of LDPC-FDs. Additionally, we will demonstrate that the FP EXIT charts offer insights into the specific causes of performance degradation, which are unavailable when using BER simulations.

We commence by introducing the FP number representation and the corresponding Log-SPA operations in LDPC-FDs in Section 3.2. In Section 3.3, we detail the extension of EXIT charts to the case of LDPC-FDs, as well as the analysis based on the FP EXIT charts. Our simulation results are presented and discussed in Section 3.4. Finally, we conclude this chapter in Section 3.5.

3.2 Fixed-Point LDPC Decoders

In this section, we will discuss the schematic of the LDPC-FD and the corresponding FP operations of the Log-SPA, based on the adaption of the floating point LDPC decoder introduced in Section 2.3. The Two's Complement (TC) number representation will also be addressed, which is most widely adopted for the FP implementation of LDPC decoders in practice, owing to the simplicity of its arithmetic. In particular, the Look-Up Table (LUT) will be employed to simplify the FP operation of the Log-SPA using TC representation.

3.2.1 Fixed-Point LDPC Decoders and the Log-SPA Operations

The transceiver schematic of the LDPC-FD is similar to the LDPC decoder using floating point representation, as shown in Figure 2.4 in Chapter 2. Figure 3.1 depicts the schematic for an LDPC-FD, where the LDPC encoder is employed to transform the sequence of K information bits $\mathbf{u}$ into the N-bit encoded sequence $\mathbf{c}$, which is transmitted through an Additive White Gaussian Noise (AWGN) channel, while the LDPC decoder may be considered to be an iteratively-operated serial concatenation of two decoders that are separated by an interleaver π of length N_{edge}. More specifically, Variable Node Decoder (VND) comprises N VNs with the same degree[5] of d_{v}, while the parity-Check Node Decoder (CND) comprises M CNs with degree d_{c}, where we have $Md_{\text{c}} = Nd_{\text{v}} = N_{\text{edge}}$. Note that the soft demodulator provides the decoder with LLR

[5]The extension of our proposed methods to irregular LDPC codes is trivial.

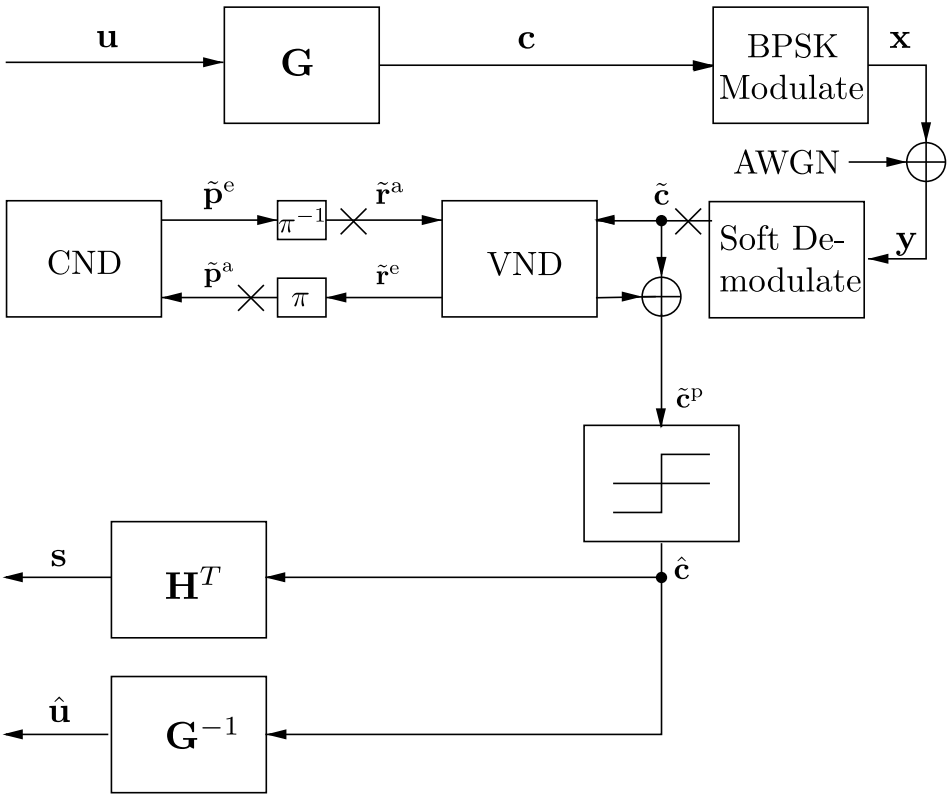


Figure 3.1: The schematic of an LDPC-FD, which comprises an encoder and an iterative decoder. The mark $\times$ indicates where clipping is performed.

sequences represented by FP numbers. In each decoding iteration, the VND uses the Forward-Backward Algorithm (FBA) [11, 15] to convert the *a priori* LLR sequences $\tilde{\mathbf{c}}^{\mathrm{a}}$ and $\tilde{\mathbf{r}}^{\mathrm{a}}$ into the extrinsic LLR sequence $\tilde{\mathbf{r}}^{\mathrm{e}}$. Meanwhile, the CND uses the FBA to convert $\tilde{\mathbf{p}}^{\mathrm{a}}$ into $\tilde{\mathbf{p}}^{\mathrm{e}}$. While the VND FBA generates extrinsic LLRs by summing *a priori* LLRs, the CND FBA combines LLRs using the boxplus operator [69], according to (2.18). However, the correction terms in (2.18) involve the logarithmic and exponential operations, the implementation of which will be detailed in Section (3.2.3). Note that while the CND employs the boxplus operator of (2.18) in a Log-SPA LDPC decoder, the CND of an MSA LDPC decoder approximates this using the minimum-finding operator of (2.19).

3.2.2 Two's Complement Representation and Arithmetic

TC representation having a bit width W consists of a fraction part having a bit width W_{f} and an integer part having a bit width W_{i} , which includes a sign bit as the Most Significant Bit (MSB) [76], where $W = W_{\mathrm{i}} + W_{\mathrm{f}}$. For a fractional FP number, there is an imaginary point set between the integer and fraction parts, like the decimal point in decimal systems. Here, it is the integer bit width W_{i} that dictates the dynamic-range

of the FP representation, while its fine-resolution is dictated by the fraction bit width $W_{\rm f}$.

The W-bit binary number $(b_{W-1}b_{W-2}\ldots b_1b_0)_{\rm b}$ represents the decimal number having the value of $(x)_{\rm d} = -b_{W-1}2^{W-1} + \Sigma_{i=0}^{W-2} b_i 2^i$, where the bit b_{W-1} indicates the sign of the number. The sign bit indicates a positive number by 0 and a negative number by 1. For a positive number, the other bits on the right of the sign bit are used to represent the magnitude of the absolute value; in the case of a negative number, its magnitude is represented by the complement of the absolute value plus 1. For example, using $W_{\rm i} = 4$ and $W_{\rm f} = 2$, 4.50 is represented by 0100.10, while the FP binary number 1010.11 represents the decimal number -6.25. Table 3.2 also shows a complete set of 3-bit binary representations. The left column of Table 3.2 gives the natural binary representations of numbers, compared with TC numbers in the right column. Note that it is the bit width of the integer part, which determines the maximum dynamic range that can be represented for large values, and the bit width of the fraction part that dictates the achievable precision.

Table 3.2: An example of two's complement representation.

Binary number ($W_{\rm i}.W_{\rm f}$)	decimal values
1.00	-1
1.01	-0.75
1.10	-0.5
1.11	-0.25
0.00	0
0.01	$+0.25$
0.10	$+0.5$
0.11	$+0.75$

The further advantage of TC is that its arithmetic operates in a natural manner, which mimics that of decimal numbers. In particular, TC subtraction satisfies

$$A - B = A + (-B). \tag{3.1}$$

For example,

$$0.25 - 0.75 = 0.25 + (-0.75) = 0.01 + 1.01 = 1.10 = -0.5.$$

This example reveals that TC performs a similar operation for both addition and subtraction, without requiring any separate operation based on the sign bits to decide whether to perform separate addition or subtraction operations.

Note that it is necessary to let subtraction *overflow*, providing when a carry is generated from the leftmost bit. In this way, the carry will be abandoned, providing the correct result. For instance,

$$0.75 - 0.75 = 0.75 + (-0.75) = 0.11 + 1.01 = (1)0.00 = 0.$$

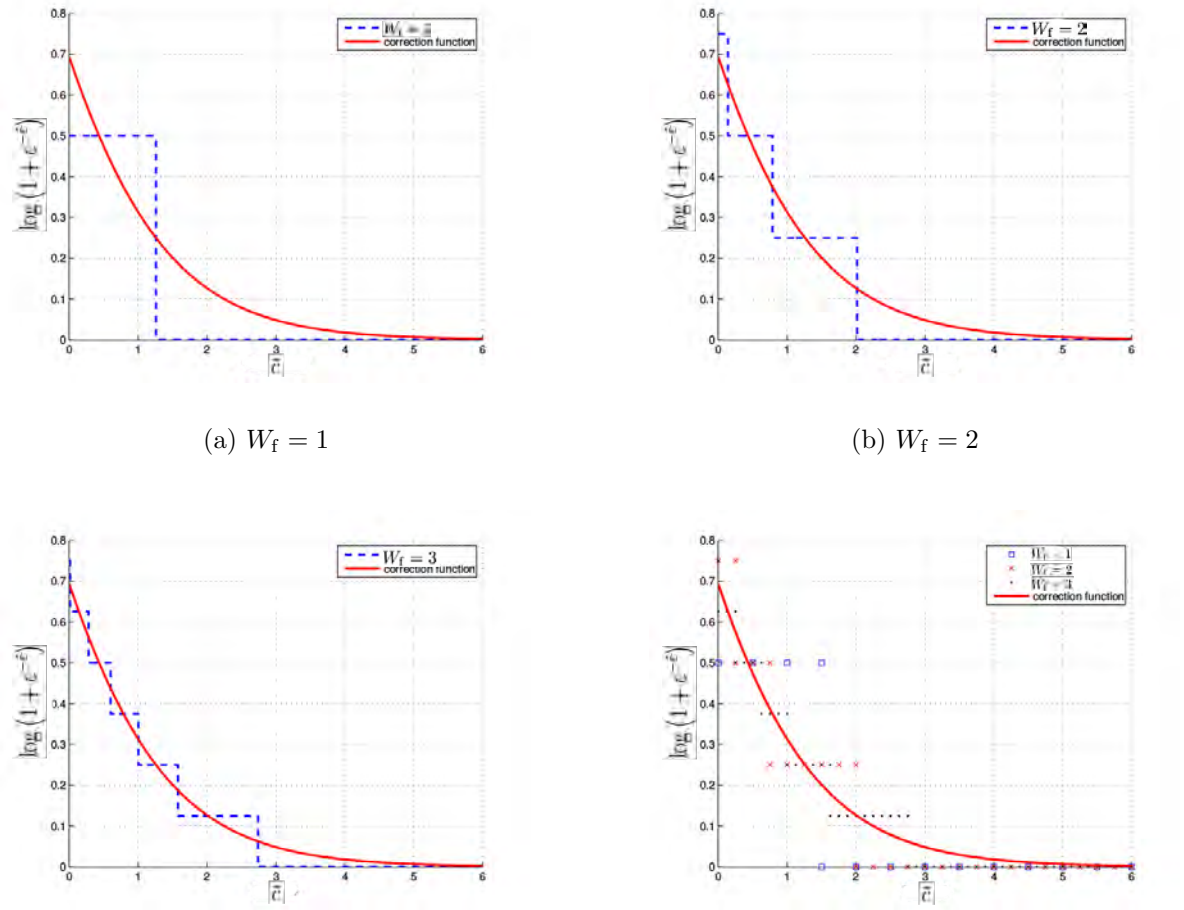
Saturation may be employed in TC arithmetic in order to avoid the potential overflow that would eventually occur by repeatedly incrementing a TC number. As we shall show in Section 3.4, clipping the integer bit width W_i of the *a priori* LLRs $\tilde{\mathbf{r}}^\mathrm{a}$, $\tilde{\mathbf{p}}^\mathrm{a}$ and $\tilde{\mathbf{c}}$ to a reduced value x may in fact improve the performance of the LDPC-FD, despite appearing to be counter-intuitive. In Figures 3.1 and 3.3, these clipping operations are indicated using $\times$ symbols. For example, when the $W_\mathrm{i} = 4$ integer bits of the TC number 1010.11 are clipped to $W_\mathrm{c} = 3$ bits, the result becomes 1100.00, which corresponds to -4 in decimal, which is the lowest number that can be represented by $W_\mathrm{c} = 3$ integer bits and $W_\mathrm{f} = 2$ fraction bits. Similarly, 0100.10 may be clipped to 0011.11, which corresponds to 3.75 in decimal, which is the highest number that can represented by $W_\mathrm{c} = 3$ integer bits and $W_\mathrm{f} = 2$ fraction bits. For the remainder of the thesis, we use the notation $\mathrm{FP}(W_\mathrm{c}, W_\mathrm{i}, W_\mathrm{f})$ to define a particular TC representation.

3.2.3 Look-Up Table

In a Log-SPA, CN relying on the boxplus operator $\boxplus$ of (2.18), the correction term $\log\left(1 + e^{-\tilde{c}}\right)$ can be computed accurately in a floating point LDPC decoder. By contrast, however, LDPC-FDs typically rely on a Look-Up Table (LUT) for the computation[6] of this correction term. Since $\log(1 + e^{-\tilde{c}}) < 1$ for $\tilde{c} > 0$, it is the number of bits W_f in the fraction part of the two's complement representation that dictates the design of the LUT. More specifically, for $W_\mathrm{f} \in \{1, 2, 3\}$, Figure 3.2 illustrates the multiples of 2^{-W_f} that most closely approximate $\log(1 + e^{-\tilde{c}})$ for various values of $\tilde{c}$, which are also selected from multiples of 2^{-2}. Here, the number of entries in the LUT is determined by the bit width W_f of the fraction part used in the FP representation of LLRs. For example, when $W_\mathrm{f} = 2$, we have

$$\log\left(1 + e^{-\tilde{c}}\right) \approx \begin{cases} 0.75 & \text{if } \tilde{c} \in [0, 0.15) \\ 0.5 & \text{if } \tilde{c} \in [0.15, 0.9) \\ 0.25 & \text{if } \tilde{c} \in [0.9, 2) \\ 0 & \text{if } \tilde{c} \in [2, +\infty) \end{cases}. \tag{3.2}$$

[6]The correction terms can also be implemented by piece-wise linear approximation, a single constant value and other methods [67, 66, 11].



(a) $W_\mathrm{f} = 1$

(b) $W_\mathrm{f} = 2$

(c) $W_\mathrm{f} = 3$

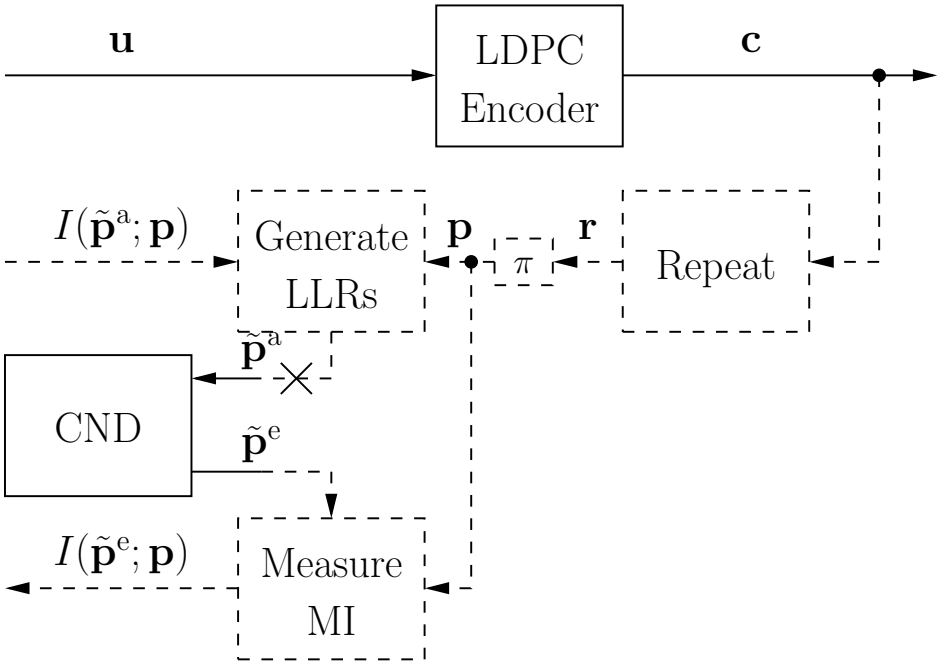
(d) $W_\mathrm{f} = 1, 2, 3$

Figure 3.2: Correction function $\log\left(1 + e^{-\tilde{c}}\right)$ and its approximation by LUT for the case of (a) fraction bit width $W_\mathrm{f} = 1$, (b) fraction bit width $W_\mathrm{f} = 2$ and (c) fraction bit width $W_\mathrm{f} = 3$.

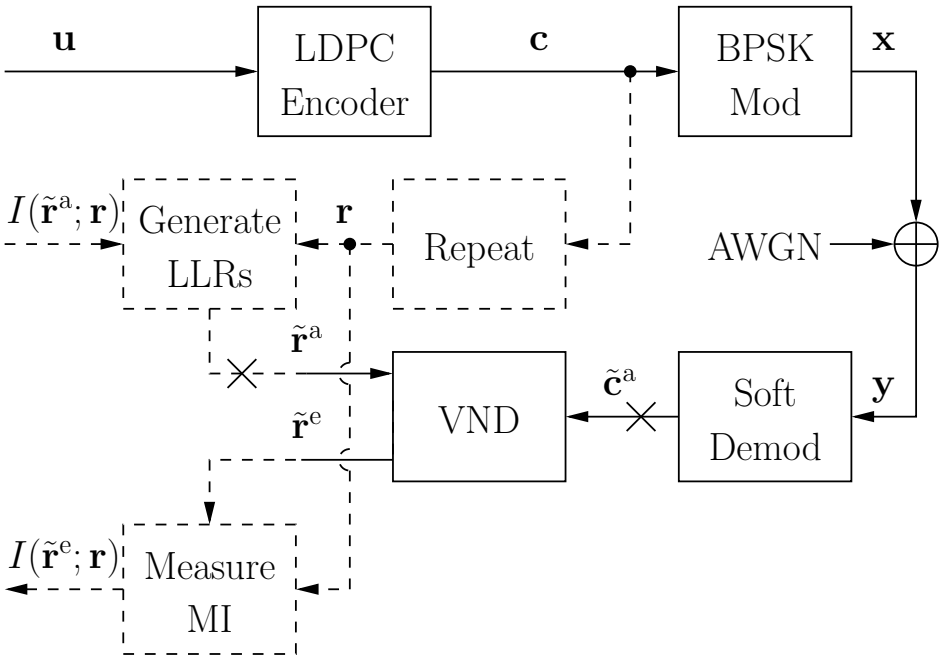
3.3 Fixed-Point EXIT Chart Analysis

In this section, we will address the adaption of the generation of the EXIT charts to the LDPC-FD, based on Section 2.4.

The iterative exchange of increasingly more reliable extrinsic information between the CND and VND can be characterised using EXIT charts. These employ the MI $I(\tilde{\mathbf{x}}; \mathbf{x})$ to quantify the reliability of the information in the LLR sequence $\tilde{\mathbf{x}}$ about the contents of the bit sequence $\mathbf{x}$. Here, the MIs have values in the range of $[0, 1]$, where '0' indicates that the LLRs contain no information about the corresponding bits, while '1' indicates perfect MI. More specifically, an EXIT function plots the Mutual Information (MI) of the extrinsic LLRs $\tilde{\mathbf{x}}^\mathrm{e}$ that are output by a decoder as a function



(a) CND



(b) VND

Figure 3.3: Adapted schematics used to depict the generations of EXIT function for the (a) CND, (b) VND. The mark × indicates where clipping is performed.

of the MI of the corresponding input *a priori* LLRs.

Figure 3.3(b) shows how the schematic of Figure 3.1 may be adapted to plot the VND EXIT function, which is exemplified in Figure 3.4(b). More specifically, as the dashed boxes and paths of Figure 3.3(b) show, each bit of the encoded sequence $\mathbf{c} = [c_1, c_2, \cdots, c_N]$ is repeated d_v times to form the longer N_edge-bit sequence $\mathbf{r}$, where we have $r_{(\mathrm{v}-1)\cdot d_\mathrm{v}+1} = r_{(\mathrm{v}-1)\cdot d_\mathrm{v}+2} = \cdots = r_{\mathrm{v}\cdot d_\mathrm{v}} = c_\mathrm{v}$ for the v^{th} VN[7]. The EXIT chart simulation of Figure 3.3(b) generates Gaussian-distributed *a priori* LLR sequences $\tilde{\mathbf{r}}^\mathrm{a}$ having MIs $I(\tilde{\mathbf{r}}^\mathrm{a}; \mathbf{r})$ across the range of $[0, 1]$. Following the operation of the VND, the MI $I(\tilde{\mathbf{r}}^\mathrm{e}; \mathbf{r})$ of the extrinsic LLR sequence $\tilde{\mathbf{r}}^\mathrm{e}$ is measured using the MI histogram based method of (2.33), the histogram based method of evaluating the MI can accurately quantify the degradation imposed by sub-optimal decoding algorithms, which produce LLRs that do not satisfy the consistency condition [49, 50]. Figure 3.4(b) plots $I(\tilde{\mathbf{r}}^\mathrm{e}; \mathbf{r})$ as a function of $I(\tilde{\mathbf{r}}^\mathrm{a}; \mathbf{r})$, when using the floating point Log-SPA or MSA for VN degrees in the set $d_\mathrm{v} \in \{2, 4, 8, 16\}$. Note that the Log-SPA and MSA have identical VND's EXIT functions, since the distinction between these algorithms affects only the CND operation. Furthermore, since the simulation based investigations outlined in Figure 3.3(b) consider the AWGN channel, the results of Figure 3.4(b) are specific to an AWGN channel E_b/N_0 of 3 dB. Higher or lower E_b/N_0 values would move the EXIT functions upwards or downwards, respectively.

Similarly, the investigations detailed in Figure 3.3(a) are used to plot the EXIT function of the CND, as exemplified in Figure 3.4(a). Here, the dashed-box interleaver π permutes the bit sequence $\mathbf{r}$, in order to obtain the N_edge-bit sequence $\mathbf{p}$. Figure 3.4(a) plots $I(\tilde{\mathbf{p}}^\mathrm{e}; \mathbf{p})$ as a function of $I(\tilde{\mathbf{p}}^\mathrm{a}; \mathbf{p})$, when using the floating point Log-SPA or MSA for CN degrees in the set $d_\mathrm{c} \in \{4, 8, 16, 32\}$. Observe that the Log-SPA and MSA have different EXIT functions for the CND, since the former operates on the basis of (2.18), while the latter employs (2.19). More specifically, the MSA CND EXIT function can be seen to achieve extrinsic MIs that are almost 0.005 lower than these of the Log-SPA. Note however that it is conventional to plot these EXIT functions on inverted axes [49, 50]. Also note that the CND EXIT function is independent of the channel's E_b/N_0 value.

As described in [49, 50], the VND's and CND's EXIT function may be plotted in the same figure to obtain an EXIT chart. When the channel E_b/N_0 is sufficiently high, the VND's EXIT function will be above the CND's EXIT function, and they will not intersect before reaching the $(1, 1)$ point of perfect convergence to the minimum attainable BER. The emergence of the resultant so-called open EXIT tunnel implies that

[7]The extension to an irregular LDPC code is trivial.

iterative decoding convergence towards a minimum BER may indeed be achieved. Observe in Figure 3.4(a) that the EXIT function values corresponding to the sub-optimal MSA are slightly higher than those corresponding to the optimal Log-SPA. This implies that a slightly higher channel E_b/N_0 is required to create an open EXIT tunnel, when the MSA is employed. These results demonstrate that the performance degradation imposed by sub-optimal decoding algorithms can be investigated by considering the quantization-induced narrowing of EXIT tunnels. In the next section, we use this technique to identify desirable bit widths for FP implementations of LDPC codes. Note that the vertical bars in Figures 3.4, 3.5 and 3.6 indicate the spread of the EXIT bands [77, 78, 79], when employing an interleaver length of N_{edge}. These bands allow the EXIT chart to accurately characterize the iterative decoding process, even when very short interleaver lengths are employed, in which case the spread of the bands is increased, but the EXIT functions remain unchanged.

3.4 Simulation Results

In this section, we determine the minimum required bit widths for FP implementations of LDPC codes in a systematic manner. We begin by using EXIT chart simulations to determine a desirable value for the fraction bit width W_f in isolation. Next, we consider the clipped integer bit width W_c in isolation, before jointly considering W_f, W_c and the integer bit width W_i. During our investigations, we shall use the Log-SPA as our benchmarker. We shall consider the degradations imposed by limited bit widths to be acceptable, provided that they are less than the difference between the CND's EXIT functions attained using the MSA and Log-SPA, namely a maximum MI degradation of 0.005. Note that in each case, we found that the spread of the EXIT bands was unaffected by the bit width and that they accurately predicted the path of the iterative decoding trajectories, even when employing short interleaver lengths. All the parameters used in FP EXIT chart simulations are listed in Table 3.3.

Table 3.3: Schemes having different bit width used for FP EXIT chart simulations of LDPC codes.

integer bit width W_i	$1, 2, 3, 4, 5, 6$ bit
fraction bit width W_f	$0, 1, 2, 3, 4, 5$ bit
clipped integer bit width W_c	$3, 4, 5,$ bit
infinite value ∞	32 bit

3.4.1 Fraction Bit Width

Figure 3.4 demonstrates the effect of employing fraction bit widths in the set $W_{\mathrm{f}} \in \{0, 1, 2, 3, 4, 5\}$ upon the CND's and VND's EXIT functions. Here, 32 bit has been selected for the integer bit widths W_{c} and W_{i}, which we assume to be an effectively infinite value, as implied by the notation $\mathrm{FP}(\infty, \infty, W_{\mathrm{f}})$. This allows us to consider the effect of the fraction bit width in isolation.

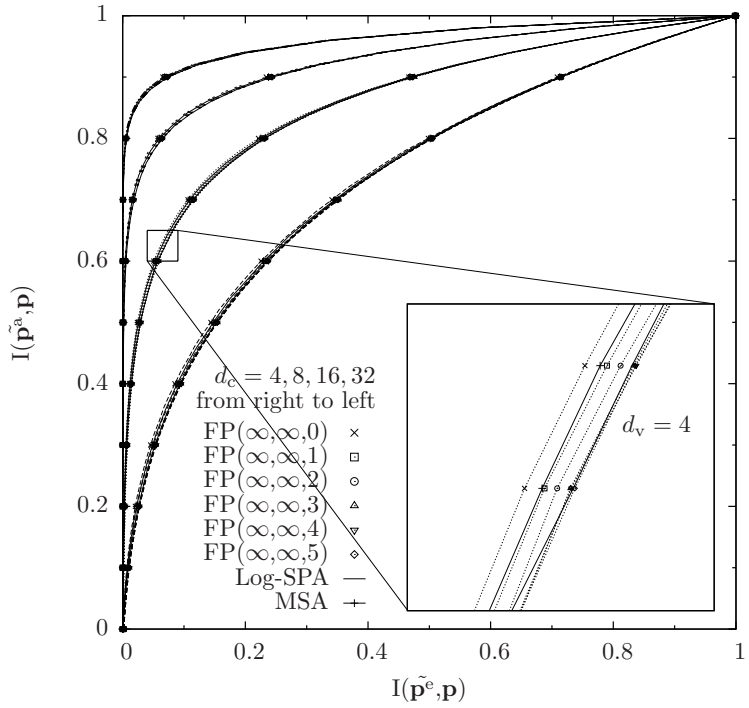
Figure 3.4 shows that a higher degradation is imposed by lower fraction bit widths W_{f}, for both the CND and VND, which is not unexpected owing to the reduced precision that this implies. This degradation is manifested as a slight EXIT tunnel narrowing across the entire range of MIs. The results show that the degradation imposed only slightly varies with the node degrees d_{c} and d_{v}.

The CND results of Figure 3.4(a) show that the degradation imposed by a fraction bit width of $W_{\mathrm{f}} \geq 2$ is less than that imposed by the MSA. Likewise, only $W_{\mathrm{f}} \geq 1$ imposes only an acceptable degradation for the VND. In order to simplify the LDPC decoder's architecture, it is desirable to adopt the same fraction bit width W_{f} in both the CND and VND. For this reason, we recommend a fraction bit width of $W_{\mathrm{f}} = 2$ for striking a desirable trade-off between the complexity imposed and the performance attained.

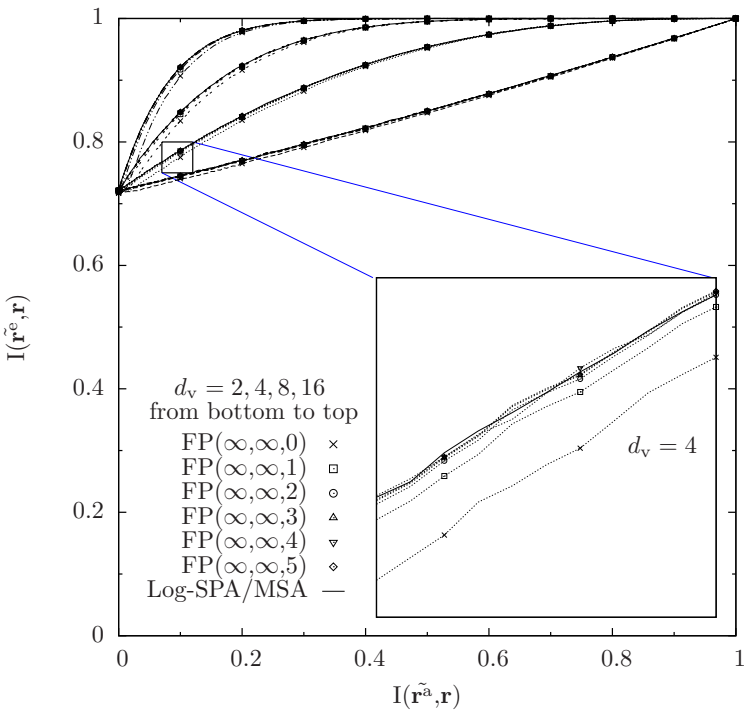
3.4.2 Clipping Range

The effect of employing clipping bit widths in the set $W_{\mathrm{c}} \in \{1, 2, 3, 4, 5, 6\}$ upon the CND's and VND's EXIT functions is considered in Figure 3.5. Here, the notation $\mathrm{FP}(W_{\mathrm{c}}, \infty, \infty)$ is employed to show that effectively infinite values have been selected for the integer and fraction bit widths W_{i} and W_{f}, respectively. In this way, the effect of the clipping bit width is considered in isolation.

Figure 3.5 shows that for both the CND and VND, a higher degradation is imposed by lower clipping bit widths W_{c}, which may indeed be expected owing to the reduced dynamic range that this implies. As shown in Figure 3.5(a), the degradation imposed on the CND EXIT functions is similar to that caused by employing a limited fraction bit width W_{f}, but the effect is more significant. In the case of the VND, the degradation causes a droop in the EXIT functions for low *a priori* MIs of $I(\tilde{\mathbf{r}}^{\mathrm{a}}; \mathbf{r}) < 0.1$, when $W_{\mathrm{c}} \in \{1, 2\}$. Briefly, this may be explained by the fact that $W_{\mathrm{c}} \in \{1, 2\}$ causes most *a priori* LLRs to become saturated, effectively transforming the VND into a hard decision decoder. While Figure 3.5(b) shows that the shape of the droop is affected by

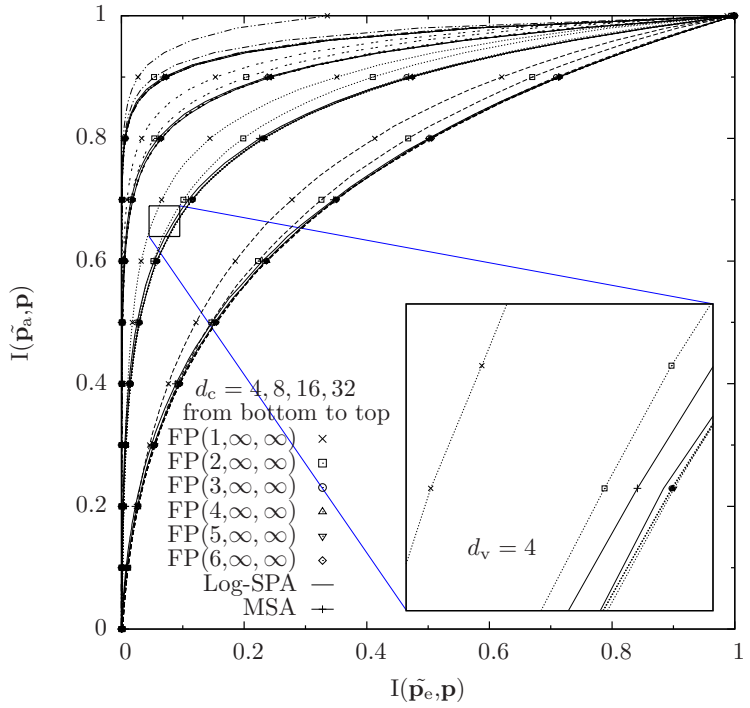


(a) CND

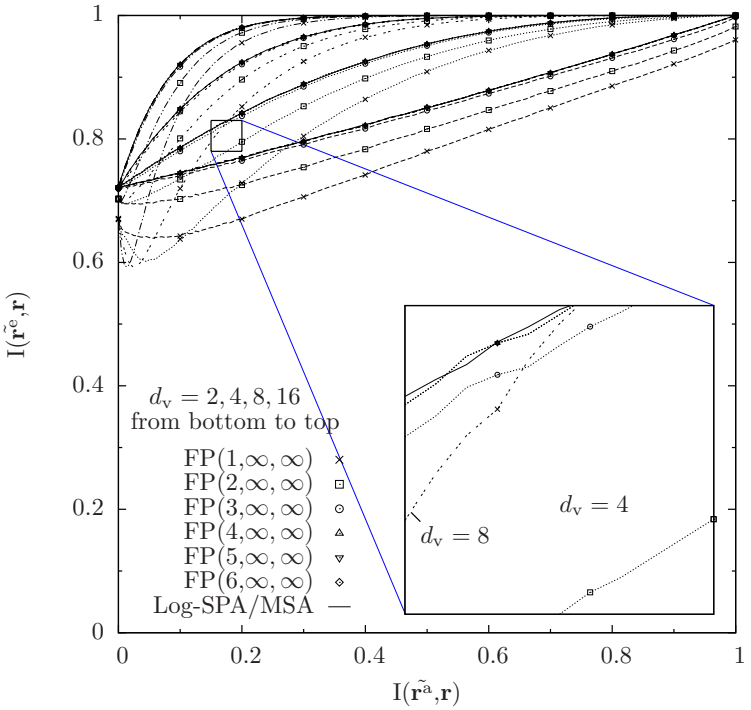


(b) VND

Figure 3.4: The EXIT functions for TC implementations of LDPC codes employing various fraction bit widths z, as well as various CN and VN degrees, for communication over an AWGN channel having an E_b/N_0 of 3 dB.



(a) CND



(b) VND

Figure 3.5: The EXIT functions for FP implementations of LDPC codes employing various clipped integer bit widths x, as well as CN and VN degrees, for communication over an AWGN channel having an E_b/N_0 of 3 dB.

the variable node degree d_v, but the magnitude of the droop is largely unaffected.

Both the CND's and VND's results of Figure 3.5 show that the degradation imposed by a clipping bit width of $W_c \geq 3$ is less than that imposed by the MSA. For this reason, we suggest that a desirable trade-off between complexity and performance can be struck by using a clipping bit width of $W_c = 3$.

3.4.3 Integer Bit Width

Let us now combine the conclusions of Sections 3.4.1 and 3.4.2 with the consideration of the integer bit width W_i. More specifically, we adopt the values of $W_c = 3$ and $W_f = 2$, while considering integer bit widths from the set $W_i \in \{3, 4, 5\}$.

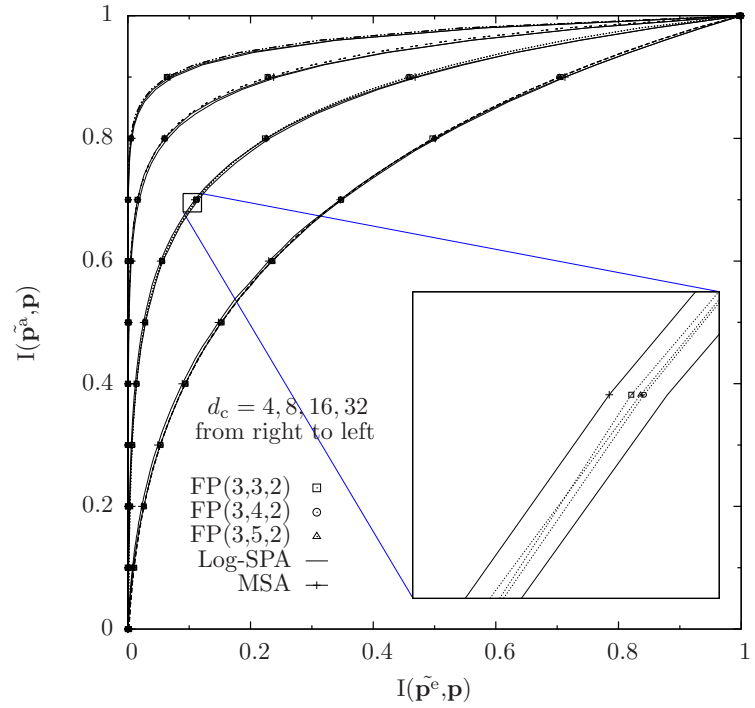
For both the CND and VND, Figure 3.6 shows that a more severe degradation is imposed by lower integer bit widths W_i, which is not unexpected owing to the reduced dynamic range that this implies. As in the case of limited fraction bit widths, this degradation is manifested as a slight tunnel narrowing across the entire range of MIs. However unlike for limited fraction bit widths, Figure 3.6(b) shows that the degradation is more pronounced for higher variable node degrees d_v.

Regardless of the node degrees, it can be seen for both the CND and VND that an integer bit width of $W_i = 4$ is sufficient to avoid a degradation that is significantly higher than that imposed by the MSA. For this reason, we conclude that the FP$(3, 4, 2)$ representation facilitates LDPC implementations that strike the most desirable trade-off between the complexity imposed and the performance attained.

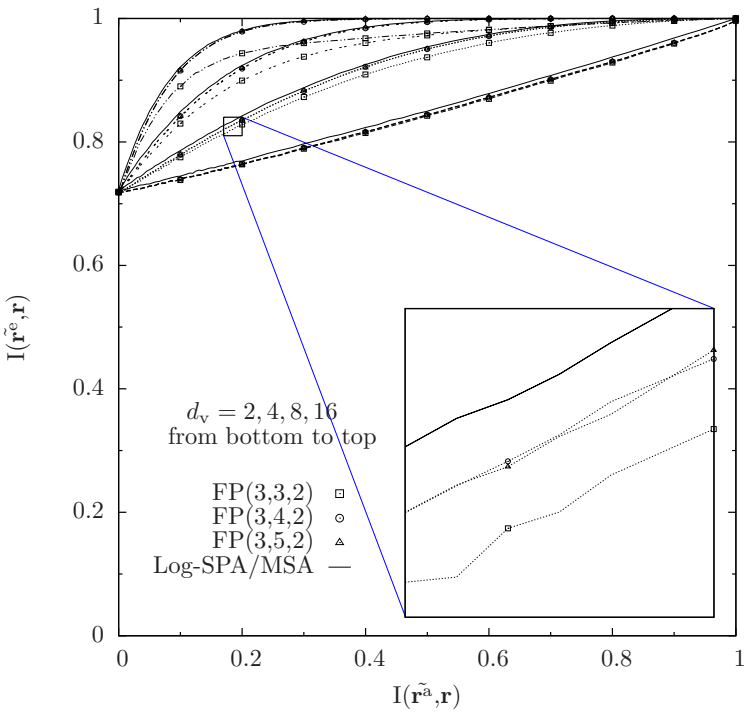
3.4.4 BER Results

In analogy with Chapter 2, this section presents BER plots to validate the conclusions drawn based on the EXIT charts of Sections 3.4.1, 3.4.2 and 3.4.3. We first demonstrate the effect of the fraction bit width W_f that was charaterized by the EXIT charts of Section 3.4.1 by using the BER plot of Figure 3.7(a), which does not apply clipping to the integer part. Following that, the effect of the integer bit width W_i is validated by the BER plot of Figure 3.7(b). Two benchmarkers are also considered, namely the optimal floating point Log-SPA and the sub-optimal floating point MSA, where the maximum number of decoding iterations considered is 20. All of the parameters used in these simulations are listed in Tables 2.3 and 3.3.

Figure 3.7(a) shows the effect of the fraction bit width on BER performance of the LDPC-FD, when the fraction bit width is increasing from $W_f = 0$ to 7, and the

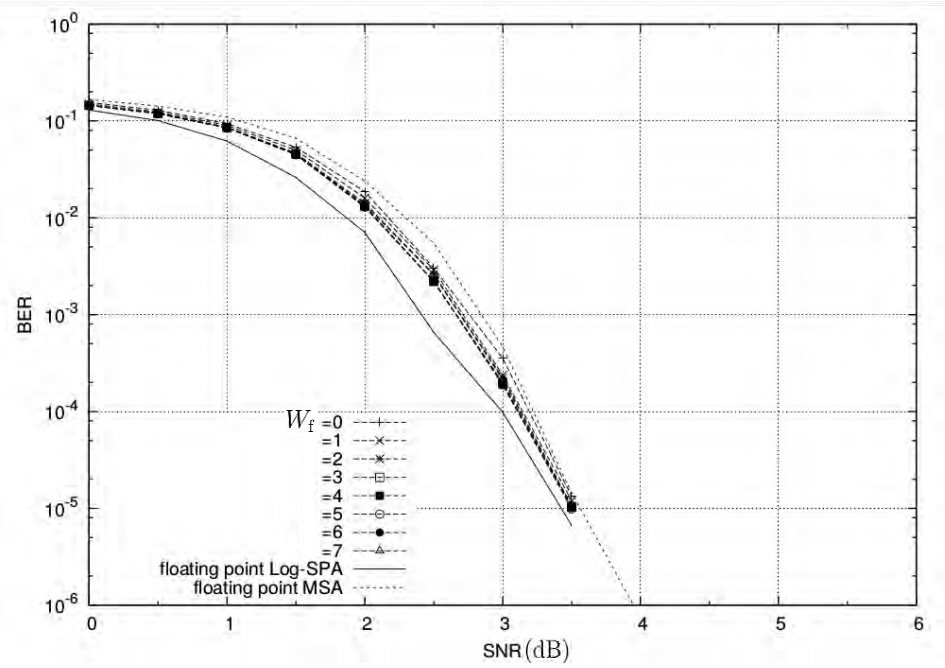


(a) CND

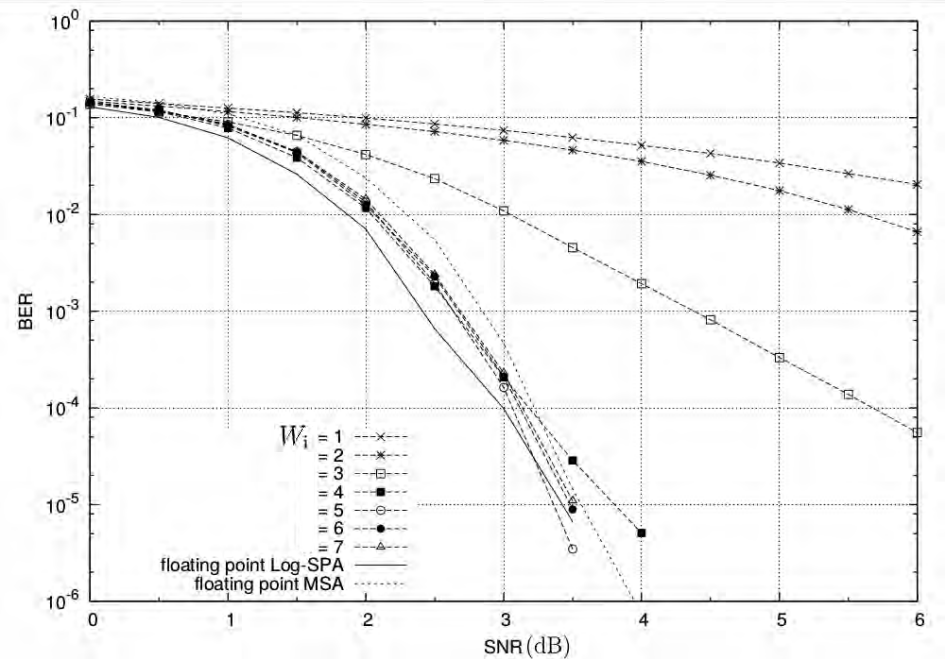


(b) VND

Figure 3.6: The EXIT functions for FP implementations of LDPC codes employing various integer bit widths y, as well as various CN and VN degrees, for communication over an AWGN channel having an E_b/N_0 of 3 dB.



(a) Fraction bit width $W_f \in \{0, 1, 2, 3, 4, 5, 6, 7\}$, when $W_c = W_i = 32$.



(b) Integer bit width $W_c = W_i \in \{1, 2, 3, 4, 5, 6, 7\}$, when $W_f = 32$.

Figure 3.7: BER plots of the LDPC-FD using different FP schemes.

integer bit width is set as $W_i = 32$ bits. As maybe observed in Figure 3.7(a), the BER performance is barely improved, when incrementing of the fraction bit width W_f.

Moreover, all of the FP schemes outperform the sub-optimal MSA benchmarker, but exhibit a gap of around 0.1 dB from the optimal Log-SPA benchmarker. This is in agreement with the observation that the EXIT functions of CND and VND using the same FP schemes exhibit an insignificant gap compared to both of the benchmarker, as shown in Figure 3.4,

Figure 3.7(b) shows the effect of the integer bit width on BER performance of the LDPC-FD, when the integer bit width is increased from $W_{\mathrm{i}} = 1$ to 7 without any clipping, while the fraction bit width is set to $W_{\mathrm{f}} = 32$ bits. It may be observed that the FP schemes exhibit slow iterative decoding convergence when $W_{\mathrm{i}} \leq 3$, while schemes having $W_{\mathrm{i}} > 3$ all approach the two benchmarkers' performance. Explicitly, the FP$(4, 4, 32)$ scheme only exhibits a gap of around 0.2 dB compared to the sub-optimal MSA benchmarker at the BER of 10^{-5}. By contrast, there is no significant performance degradation between FP schemes having $W_{\mathrm{i}} \geq 5$ and the optimal Log-SPA benchmarker. This agrees with the conclusions that $W_{\mathrm{i}} = 4$ is the lowest integer bit width that gives acceptable BER performance, as was observed from the EXIT chart analysis based on Figures 3.5 and 3.6. Therefore, by comparing Figures 3.7(a) and 3.4, as well as Figures 3.7(b) and 3.5, we have demonstrated that the BER performance of LDPC-FDs using various FP schemes can be predicted by the EXIT chart analysis discussed in Sections 3.4.1, 3.4.2 and 3.4.3.

3.5 Conclusions

In this chapter, we have proposed an EXIT chart based method to investigate desirable parameters of FP implementations of LDPC codes, in order to strike an attractive trade-off between their complexity and performance. Our technique may also be employed to investigate this trade-off in other reduced-complexity versions of LDPC codes. Since this technique does not require the simulation of iterative decoding at different number of iterations or at multiple E_{b}/N_0 values, it is significantly less time consuming than the conventional BER-based method. For example, when employing a half-rate length-500 regular LDPC code having $d_{\mathrm{v}} = 3$ and FP$(3, 4, 2)$, a total of 6×10^8 CND operations and 8×10^8 VND operations are required, to obtain the CND's and VND's EXIT functions, respectively. Here, we consider 2000 frames for each of 100 *a priori* MI values in the range of $[0, 1]$. However, a BER simulation for the same LDPC code requires approximately 10^{11} CND operations and 10^{11} VND operations to consider 10 different E_{b}/N_0 values in the range of $[0, 4.5]$ dB. Here the decoding iterations are continued only until convergence is achieved. Therefore, the resultant factor of 100 complexity

reduction allows comprehensive investigations into the effects of multiple parameters to be conducted. This is particularly beneficial, since it is typically necessary to investigate the most desirable bit widths whenever a new LDPC code is designed. Furthermore, EXIT chart analysis can offer insights into the causes of the performance degradation that is imposed by reduced-complexity versions of LDPC codes. For example, the analysis of Section 3.4.2 revealed that a droop in the VND's EXIT function implies that the *a priori* LLRs are excessively clipped.

In Section 3.4, we proposed a systematic approach for determining the desirable bit widths for FP implementations of LDPC codes. Our results indicate that as a rule of thumb an overall bit width of $W = W_\mathrm{i} + W_\mathrm{f} = 6$, namely the FP scheme (3,4,2), may offer an attractive trade-off between an LDPC code's complexity and performance. As summarized in Table 3.1, this published result [75] agrees with the conclusions of [80, 10, 11, 12, 13, 14, 15, 16], where the slight differences observed may be attributed to the different algorithms and quantization schemes considered.

Furthermore, as was also demonstrated by the comprehensive EXIT chart investigation, which was first conducted in this thesis, the fraction bit width W_f may be set as 0 in order to further reduce the implementation complexity of the LDPC-FD, while maintaining an acceptable BER performance, since the corresponding degradation imposed is relatively small. Moreover, clipping the integer bit width from $W_\mathrm{i} = 4$ to $W_\mathrm{c} = 3$ bits barely improves the BER performance of the LDPC-FD, and imposes an overhead for performing the clipping. Therefore, an overall bit width of $W = W_\mathrm{i} = 4$ is the minimum required for an LDPC-FD to achieve an insignificant performance degradation. Motivated by this, we will propose 4-bit FP schemes for the fully parallel implementation of LDPC-FDs in the next chapter.

Chapter 4

Timing-Error-Tolerant Fixed-Point LDPC Decoder Using Base Minus Two

4.1 Introduction

During data transmission via noisy communication channels, errors may be introduced into the data recovered at the receiver. As discussed previously in Chapters 2 and 3, channel decoders may be adopted at the receiver to mitigate these channel-induced errors, relying on the redundancy built into the messages by the channel encoder at the transmitter. Owing to their outstanding error correction capability, Low-Density Parity-Check (LDPC) codes can be found in many communication standards, such as IEEE 802.11 (WiFi) [63] and IEEE 802.16e (WiMAX) [81, 65]. Next-generation communication standards are expected to offer communication throughputs on the order of Gbit/s and communication latencies on the order of microseconds [82, 83]. This requires that the channel decoders provide the corresponding processing throughput and latencies, in order to avoid imposing any bottlenecks. Based on the discussions of Chapters 2 and 3, LDPC decoders have been demonstrated to have the potential to meet these requirements, since they can be processed with a very high degree of parallelism. More specifically, LDPC decoders can operate on the basis of the iterative exchange of Fixed-Point (FP) numbers between distributed and parallel processing nodes. In this way, LDPC decoders implemented as fully parallel ASICs can offer high error correction capability, high processing throughput and low latency [84].

In order to further increase the processing throughput of ASIC LDPC decoders, the

clock period of the hardware can be scaled down, which is called overclocking, so that each LDPC decoding operation can be completed using less time. However, overclocking has the potential side effect of introducing *timing errors*, which occur whenever a signal has not had enough time to propagate through the circuit before it is clocked into a memory, before the end of the reduced clock period. On the other hand, the task of designing completely reliable Very Large-Scale Integration (VLSI) systems is becoming increasingly challenging in successive generations of nanoscale fabrication technology, due to the increasing susceptibility to IR drop, inductive noise, crosstalk, electrostatic discharges, particle strikes, switching noise and fabrication process variations [31, 32, 33, 34, 35, 36, 37]. Previous efforts [40, 85, 86, 87, 88, 41, 42, 44] have been dedicated to designing general purpose circuits, as well as ASIC LDPC decoder, that are capable of tolerating a limited amount of timing errors. This has been achieved by employing additional circuitries to detect and correct errors, which implies a cost in area and energy consumption. However, in LDPC decoders, it may be possible to achieve the same degree of timing error tolerance by exploiting the inherent LDPC error-correcting capability, without the need for the extra circuitries. Motivated by this, this chapter will investigate the inherent timing error tolerance of the LDPC-FD of Chapter 3.

Two's Complement (TC) representation used to represent the Log-Likelihood Ratios (LLRs) in the Fixed-point LDPC Decoders (LDPC-FDs) is required to have a bit width of at least 4, as concluded in Chapter 3. Since the significance of the 4 bits in a TC number are different, they may have different degrees of tolerance to timing errors. Explicitly, the TC numbers may be not sensitive to timing errors affecting the bits having less significance, since the represented decimal TC numbers are not changed dramatically by these errors. However, when timing errors affect the Most Significant Bits (MSBs) of the fixed-point probabilities, the error correction performance is significantly degraded [40, 41, 42]. In this chapter, we propose the use of Base Minus Two (BMT) representation rather than TC in a fully parallel implementation of the LDPC-FD, since we will demonstrate that this can reduce the required chip area and increase the processing throughput. Furthermore, conventional methods for constructing the high-degree LDPC nodes adopt the most straightforward implementations. These either use a serial concatenation of processing elements to achieve a moderate chip area at the cost of large processing latency, or use a parallel concatenation to achieve a low latency and also a low chip area[1]. In this chapter, we propose a novel method to construct the high-degree nodes which achieves an even lower latency than the parallel structure, but with a comparable area to the serial structure. Furthermore, the pro-

[1]This parallel concatenation usually involves complex reverse operation of that employed in LDPC decoding algorithms, resulting in an unfair comparison to some degree, which will be detailed in Section 4.3.2.

posed LDPC decoder is scheduled to simultaneously decode two independent encoded frames. This is facilitated by double D-type Flip-Flop (D-FF) chains [89, 90, 35, 91], which also protect the decoder from the catastrophic propagation of metastability. By employing these three innovations, our novel fully parallel LDPC-FD architecture is shown to be more resilient to timing errors, as well as capable of achieving a higher throughput and lower latency than conventional architectures.

The chapter is organised as follows: Section 4.2 outlines the motivation and flow for designing and characterising the proposed decoder. Section 4.3 discusses the fully parallel implementation of the LDPC-FP. In particular, motivated by the conventional implementation, we propose the novel employment of the BMT number representation for LDPC decoding, as well as a novel decoder architecture that is designed to provided tolerance to processing errors. We propose the error model in Section 4.4, for conducting BER simulations in the presence of timing errors. The complete comparison between different implementations is presented and discussed in terms of timing characteristics, area, power consumption and BER simulation results, in Section 4.5. Finally, the chapter is concluded in Section 4.6.

4.2 Design Flow

In Section 4.1, we reviewed the challenges encountered in the design of previous timing error-tolerant LDPC decoders, which we will use to motivate our proposed LDPC-FD and its design flow. In this subsection, we first address the conventional design flow for LDPC decoders and discuss the modifications that are motivated for the design of timing-error-tolerant LDPC decoders. A typical design flow for ASIC implementations of LDPC decoders is illustrated in Figure 4.1. The process starts from the specifications and proceeds step-by-step until the bottom level is reached, where the ASIC is fabricated. Here, the specification defines the design goals of the LDPC decoder, which informs the selection of algorithmic parameters, such as block length, coding rate, and LDPC factor graph topology [84, 92]. Following this, an algorithm-level simulation of the iterative LDPC decoding process is employed to validate the algorithmic parameters and to identify the number of clock cycles required to achieve the desired BER. The simulation may be extended to also consider architectural parameters, such as fixed-point bit width, in order to characterize the effect on the BER. Following this, the algorithm can be converted into a Register Transfer Level (RTL) description, which precisely describes the behaviour of the decoder and allows synthesis. By linking a particular library of cells, the RTL behaviour description can be replaced by logic gates and

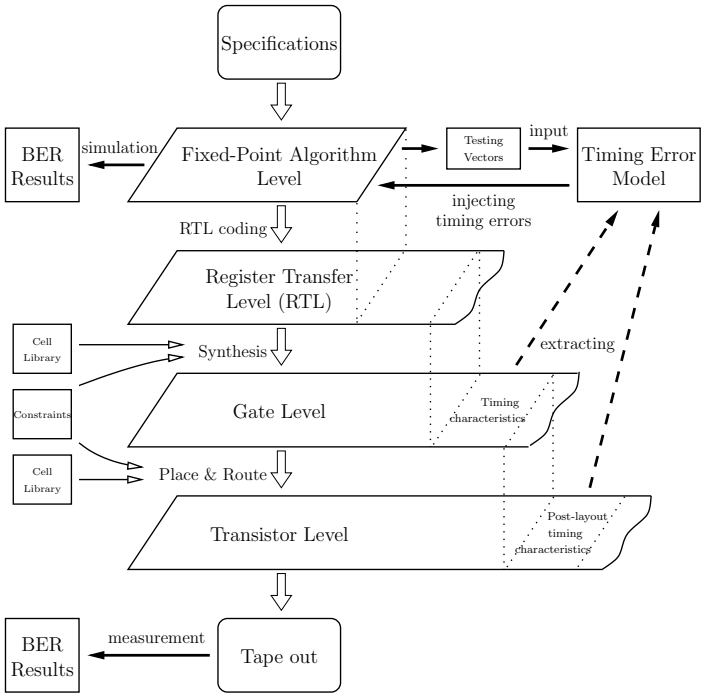


Figure 4.1: The proposed design flow for LDPC decoder ASICs.

interconnections, facilitating gate level simulation. The gates can be further replaced by transistors during placing and routing, with corresponding libraries. This facilitates transistor level simulation, which validates the operation and identifies the ASIC area, energy consumption and clock frequency. The above-mentioned design steps may be iterated several times, before finally taping out the design. As the design flow proceeds through each subsequent level, the design is enriched with more realistic simulation results, but also becomes more complex, time consuming and expensive. In the remainder of this thesis, the Cadence simulation is considered running on the transistor level with proper design tools provided, and the SPICE simulation is assumed to run on the gate level, unless otherwise specified.

Furthermore, the complexity of each level is significantly further increased when investigating the circuit behaviour under the influence of timing errors, which transform the synchronous digital operation of the circuit into asynchronous analogue operation. Consequently, it may become impossible to comprehensively simulate the timing error tolerance of an ASIC LDPC decoder at the lower design levels. As a result, the characteristics of the tape-out become unpredictable with a risk that the tape-out is unsuccessful. This risk of wasting all of the invested time, effort and expense, makes it undesirable to investigate timing error tolerance based only on measurements taken from a fabricated ASIC. Furthermore, the effect of the parameters chosen for the ASIC cannot be investigated, since these are typically hard-coded into a tape-out. Addi-

tionally, ASICs are not capable of offering insights into the internal operations of the design, in the way that simulations can. Furthermore, these experimental results may be influenced by numerous unpredictable factors, such as temperature variations, electromagnetic radiation and process variations, which may obfuscate the results. Finally, these measurements cannot be readily reproduced by other researchers, unless they have access to the fabricated chip. Owing to these issues, research is motivated in not only timing-error-tolerant LDPC implementations, but also in the design process for these implementations [33, 39, 38, 43]. This is particularly motivated now that fabrication technology has moved into the sub-65nm regime, where the reliability of nanoelectronic systems cannot be guaranteed, due to the increased effect of the IR drop, inductive noise, crosstalk, electrostatic discharges, particle strikes, switching noise and fabrication process variations [31, 32, 33, 34, 35, 36, 37].

Motivated by this, we propose a novel process for designing and characterizing our fully-parallel LDPC-FD, which has improved inherent tolerance to timing errors, compared to conventional designs. The arrows pointing upwards on the right-hand side of Figure 4.1 illustrates the proposed modification to the design flow, in contrast to the traditional design flow shown on the left-hand side. More specifically, we use simulations at the algorithm level to investigate the decoders' BER performance, but we incorporate a model of the causes and effects of timing errors, which is parametrized by characteristics obtained from the lower design levels of Figure 4.1. Although the most realistic experimental results can only be obtained by taking measurements from a fabricated ASIC, this implies a huge amount of design time, effort and financial investment, which may not deliver the desired performance as described above. Therefore, our simulation-only approach de-risks the expensive and time-consuming fabrication of an ASIC, allowing confidence to be gained in a timing-error-tolerant design. This will pave the way for our future work, which will fabricate a timing-error-tolerant LDPC decoder ASIC. Furthermore, our simulations at the upper levels of the design flow are immune to random variation and are readily repeatable, using synthesis processing, SPICE simulation and Cadence simulation. Additionally, our approach allows the characterization of the effect of each design parameter on the circuit performance to be characterised, enabling the optimization of parameter values. We will demonstrate the proposed design flow throughout the remainder of this chapter.

In this chapter, we propose a timing error model to predict the occurrence, location and effect of timing errors, according to timing characteristics of individual nodes extracted from lower levels of the design flow, which allows the algorithm level simulation of the decoding operation in fully-parallel manner at low BERs. With the aid of the developed model, it is possible to efficiently and accurately characterise the error-

correction performance of the proposed LDPC decoder, when operating in the presence of timing errors, as discussed in the following sections. Note that, transistor level simulations are utilized to characterise and verify the timing error model for each type of node.

4.3 Fully Parallel Implementation of Fixed-Point LDPC Decoder

In this section, we will discuss the fully parallel implementation of the LDPC-FD using a 4-bit FP representation and the Min-Sum Algorithm (MSA), for the sake of complexity reduction, the error correction performance of which has been discussed in Chapter 3. The corresponding MSA operations of the CNs and VNs have been explained in (2.20) and (2.24) of Section 2.3.2.3. This section commences in Section 4.3.1 by breaking down the fully parallel implementation of the LDPC-FD into the implementation of individual CNs and VNs in the LDPC code's factor graph. Based on Figure 4.2, we employ D-type Flip Flops (D-FFs) in the LDPC-FD to separate and schedule the CN and VN operations, so that the LDPC-FD is capable of simultaneously decoding two independent codewords, as well as reducing the likelihood of catastrophic metastability propagation. The improved placement of D-FFs is proposed to reduced the number of D-FFs required, without compromising the capability of simultaneous decoding. Following this, Section 4.3.2 discusses the structure of the large-degree CNs and VNs, used to implement (2.20) and (2.24). The conventional structures are presented, and contrasted to a novel structure that we propose for constructing large-degree nodes, having more desirable tolerance to timing errors. The use of BMT is proposed in Section 4.3.3, in contrast to the conventionally adopted TC.

4.3.1 Fully Parallel Scheduling

As discussed in Chapter 2, the parametrization of an LDPC decoder is completely specified by its factor graph [59, 8, 9]. Figure 4.2 illustrates a fraction of the factor graph of a $(1056, 528)$ WiMAX LDPC code [65]. This factor graph comprises CNs having degrees of $d_{\mathrm{c}} = 6, 7$, VNs having degrees of $d_{\mathrm{v}} = 2, 3, 6$, and the edges interconnecting them, each of which is analogous to a pair of wires propagating soft information bidirectionally. Although a number of algorithms have been proposed for LDPC decoding [6, 11, 66, 67, 69, 8, 71, 72, 93, 94], the MSA is adopted here for the sake of achieving low complexity [11, 64]. This algorithm uses an LLR to represent the soft in-

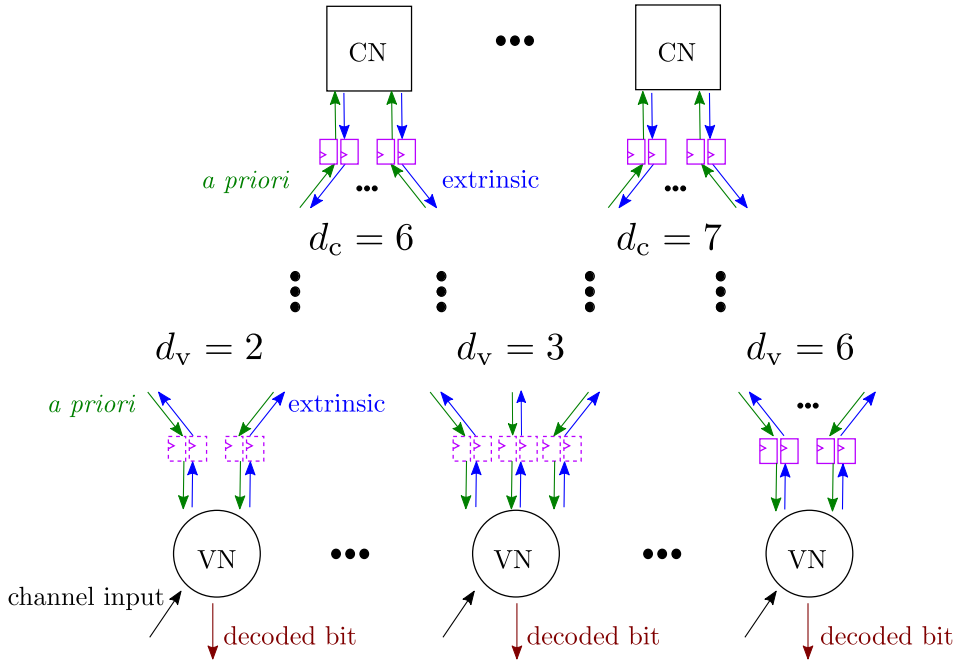


Figure 4.2: A fraction of the factor graph of a $(1056, 528)$ WiMAX LDPC code and the illustration of the decoding scheduling.

formation associated with each particular encoded bit x, expressed as $\tilde{x} = \ln \frac{P(x=1)}{P(x=0)}$. Fully-parallel MSA LDPC decoders employ an iterative decoding process in which each iteration comprises two steps. In the first step, all VNs are activated simultaneously. Each VN generates a LLR for each of the connected CNs by summing the LLRs provided by the *other* CNs with the LLR provided by the wireless channel, as indicated in (2.24). The resultant LLRs are passed along the edges of the factor graph to the CNs, which are activated simultaneously in the second step of the iteration. Each CN generates an LLR for each of the connected VNs by combining the LLRs provided by the *other* connected VNs, as indicated in (2.20). Rather than using summation to combine LLRs like in the VNs, the CNs use the boxplus $\boxplus$ operation [8] to combine LLRs. In the MSA, the boxplus operator $\boxplus$ is approximated for two LLRs $\tilde{a}$ and $\tilde{b}$ as $\tilde{c} = \tilde{a} \boxplus \tilde{b} \approx \text{sign}(\tilde{a}) \cdot \text{sign}(\tilde{b}) \cdot \min(|\tilde{a}|, |\tilde{b}|)$, as given in (2.19). The structure used to implement the combination of LLRs will be addressed in the following subsection.

4.3.1.1 Decoding Scheduling and Double D-FFs

As shown in Figure 4.2, D-FFs may be placed on the input and output ports of every CN and VN for storing the FP numbers processed. In this way, it takes 4 clock cycles to complete 1 decoding iteration, in which all CNs and VNs are activated once. This double D-FFs placement also allows the simultaneous decoding of two independent

Table 4.1: The scheduling for simultaneously decoding two independent codewords, A and B, when D-FFs are employed at the input and output ports of every CN and VN.

VNs	$d_v = 2$	$d_v = 3$	$d_v = 6$
Odd clock cycle	A	A	A
Even clock cycle	B	B	B

Table 4.2: The scheduling for simultaneously decoding two independent codewords, A and B, when D-FFs are employed at the input and output ports of CNs and VNs having a degree of $d_v = 6$ only.

VNs	$d_v = 2$	$d_v = 3$	$d_v = 6$
Odd clock cycle	A	A	B
Even clock cycle	B	B	A

codewords, provided that the received LLRs related to one of the two codewords are fed to the VNs in clock cycles having odd indices, while the received LLRs related to the other of the two are fed to the VNs in clock cycles having even indices, which is depicted in Table 4.1. Note that the use of double D-FFs implies an increased energy consumption and chip area, as reported in [95, 96, 97]. In this way, the two codewords can be decoded using the same total number of clock cycles as is required for their successive decoding without the use of double D-FFs.

Note however that the low-degree VNs have much lower critical path delays than those of the high-degree VNs and CNs. Owing to this, we propose to employ D-FFs only on the input and output ports of VNs having a degree of $d_v = 6$, as well as of CNs having degrees of $d_c = 6$ and 7. By contrast, we propose the use of no D-FFs on the input and output ports of VNs having degrees of $d_v = 2$ and 3, as shown by the dashed D-FFs in Figure 4.2. Accordingly, the scheduling may be modified to avoid the overlapped processing of received LLRs related to the two codewords. In this way, the VNs having a degree of $d_v = 6$ are fed with LLRs related to one codeword, while received LLRs related to the other codeword are fed into the other VNs, during the same clock cycle. In the next clock cycle, the received LLRs at all VNs are alternated. Table 4.2 shows this modified scheduling.

4.3.1.2 Metastability

In order to ensure that a D-FF is able to reliably clock in a sample from a binary signal, the signal is required to remain constant at '1' or '0' for a certain amount of time both before and after the sampling moment at the clock edge. The amounts of time

are referred to as the setup time and hold time, respectively. If these required timing conditions are violated, then the output of D-FF may become metastable [90, 89, 35, 91], adopting a value that is neither a valid '1' nor '0'. In this case, the D-FF requires an unpredictable amount of time to recover the output signal from the metastable state back to a valid logic value, which may be selected at random. This imposes additional delay on the next circuit path, which can cause further timing violations and metastability. In this way, metastability may propagate through the other connecting components and spread to other D-FFs. If metastability is not resolved before the arrival of the upcoming clock edge, it can take over the entire circuit and prevent it working altogether [90, 89, 35, 91]. Although the occurrence of metastability is rare in any single D-FF, it is common in circuits having many D-FFs and it is only necessary for metastability to occur at one D-FF for it to spread to the whole circuit. Therefore, metastability has to be considered during digital circuit design, especially when aggressive overclocking is applied as in this chapter.

The Mean Time Between Failures (MTBF) is a metric commonly used to characterise the likelihood of occurrence of metastability [90, 89, 35, 91]. However, a chain of double D-FFs can be used to protect a circuit from the spread of metastability. This is because the path delay between the cascaded D-FFs is very small, allowing the metastability the greatest opportunity to resolve before it reaches the second D-FF.

4.3.2 Structure of High-Degree Nodes

Nearly all CNs and around half of the VNs in the family of WiMAX LDPC codes have a degree of 6 or higher [65]. Similar node degree distributions can be found in other practical standards, such as IEEE 802.11n/ac[63]. The implementation of the high-degree CNs and VNs dictates not only the area and energy consumption of the decoder, but also the critical path propagation delay. Owing to this, the implementation of the high-degree nodes determines the tolerance of the a fully-parallel LDPC-FD to timing errors.

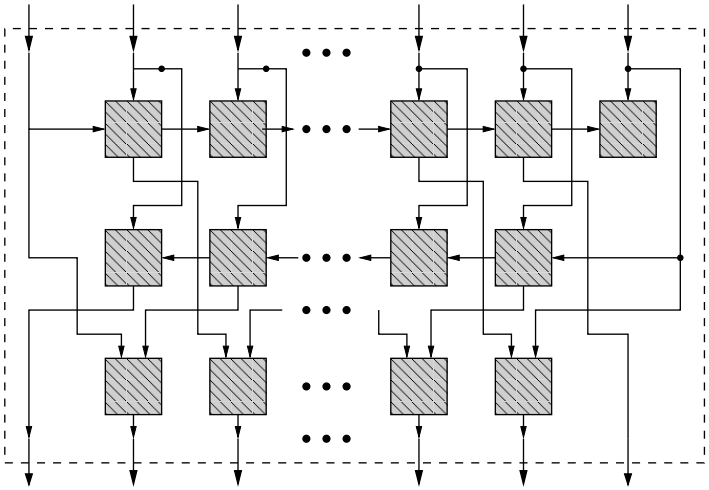
Computer-aided VLSI design can generate high-degree CN and VN circuits from either the corresponding behaviour description or the truth table. However, as the number of inputs to the nodes increases, the description and truth table representing the logic operations within the CNs and VNs becomes exponentially complex. For this reason, generating the circuits of the high-degree nodes via synthesis processing is not preferred. Alternatively, the associative property of the addition and boxplus operator allows nodes having large degrees to be constructed on the basis of 2-input 1-output

sub-nodes, as discussed in Section 2.3.2. For example, in the (1056,528) WiMAX LDPC code, the VNs having degrees of $d_v = 2, 3$ and 6 may be constructed on the basis of 2-input 1-output adders, which we refer to as sub-VNs. Likewise, the CNs having degrees of $d_c = 6, 7$ may be constructed on the basis of 2-input 1-output boxplus operators, namely sub-CNs. The circuit of a sub-node may be designed by computer-aided synthesis, due to its low complexity.

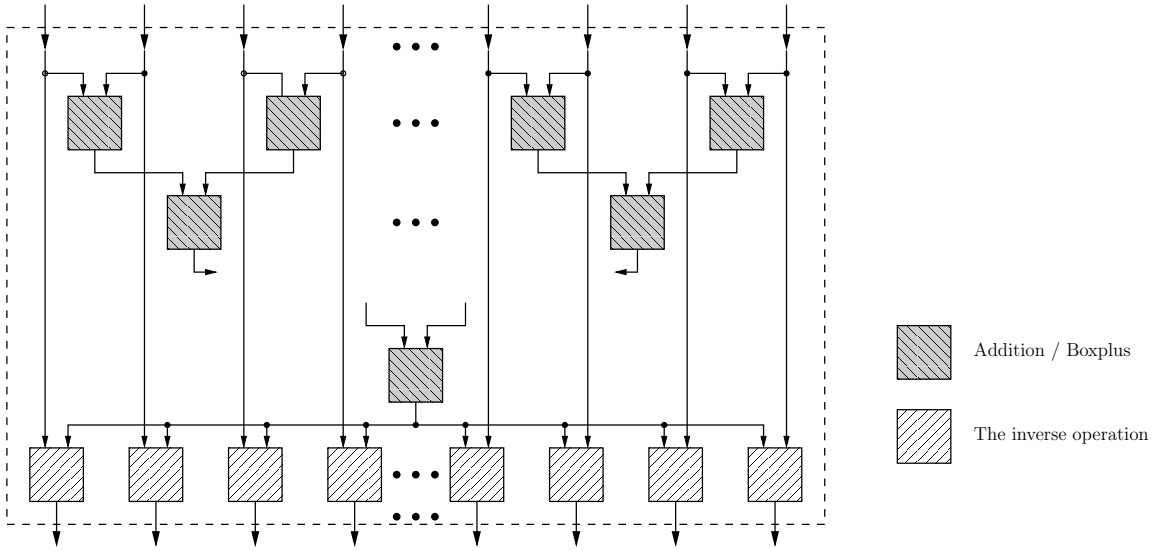
4.3.2.1 Conventional Structure

A number of methods for efficiently constructing high-degree nodes on the basis of 2-input 1-output sub-nodes have been proposed in [11, 67, 17, 16, 10, 92, 15, 98, 99, 64, 13, 12, 26, 100, 14, 101, 20, 24, 102]. The serial structure of Figure 4.3(a) is also known as *forward-backward* structure [11, 66, 67, 64], which implements the FBA of (2.21), (2.22) to (2.27). This uses sub-nodes to accumulatively combine the inputs in both directions, in order to obtain intermediate results, which are then combined to generate the outputs. However, this inherently serial processing results in a large propagation delay for the circuit's critical path. In a large node having L inputs, the forward-backward structure requires a total of $3L-6$ sub-nodes and has a critical path delay of $L-2$ times the critical path delay of an individual sub-node.

By contrast, the parallel structure of Figure 4.3(b) in [13, 14] combines all the inputs using a logarithmically-reducing number of sub-nodes to successively pair them, until the single combination of all inputs is obtained. Following this, an extra subtraction or reverse operation of (2.18) is performed, to remove each input from the combination of all inputs, in order to obtain each final output. However, the reverse operation of (2.18) has a high complexity and cannot be implemented easily [13, 14]. For a node having L inputs, the parallel structure requires at least $L-1$ sub-nodes plus L reverse sub-nodes, and achieves a critical path delay of at least $\lceil \log_2 L \rceil$ times the critical path delay of the sub-node plus the critical path delay of the reverse sub-node. Therefore, the parallel structure has a lower hardware resource requirement and a lower latency than the serial structure, assuming the reverse sub-nodes have a similar implementation complexity is similar to the original sub-nodes. Due to this advantage, the parallel structure of Figure 4.3(b) is widely adopted in the implementations of LDPC decoders [11, 67, 17, 16, 10, 92, 15, 98, 99, 64, 13, 12, 26, 100, 14, 101, 20, 24, 102]. Note that the shaded blocks in Figure 4.3 represent sub-nodes.



(a) The forward-backward structure.



(b) An instance of parallel structures.

Figure 4.3: The conventional structures for constructing nodes having high degrees based on 2-input 1-output sub-nodes.

4.3.2.2 Proposed Structure

The methods discussed in Section 4.3.2.1 for constructing high-degree nodes are not desirable for exploiting and enhancing the circuit's tolerance to timing errors, due to the large associated critical path delays. We propose a novel structure for high-degree nodes to exploit and enhance the timing error tolerance of the fully parallel LDPC decoder. Our proposed design achieves a minimum and equal propagation delay for the paths from every node input to every output. These propagation delays are lower than those of the previously mentioned forward-backward and parallel structures. Furthermore, the proposed structure employs the symmetrical connections, with the aim of reusing common pairs of inputs and common combining nodes as many times as possible. In this way, the number of consumed sub-nodes is reduced, without compromising the propagation delay at each output. Figures 4.4(b) and 4.4(d) present the schematics for the proposed high-degree nodes having 6 and 7 inputs. Figures 4.4(a) and 4.4(c) provide the schematics of nodes having 3 and 4 inputs, although their area and critical path delay are similar to those of the conventional structures. Note that the shaded blocks in Figure 4.4 represent sub-nodes.

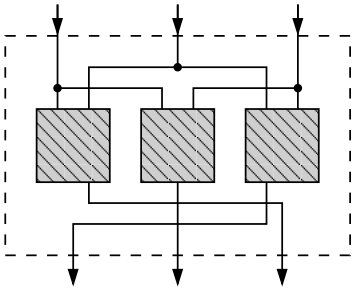
In Section 4.5, it will be demonstrated that in the fully parallel LDPC decoder using the proposed node structure of Figure 4.4, the longest critical path delay is reduced by up to 38.8%, at the cost of increase of around 12% and 4% in estimated area and power consumption, respectively. Detailed results and discussions will be provided in Section 4.5.

4.3.3 Fixed-Point Numbers and Anti-Overflow Techniques

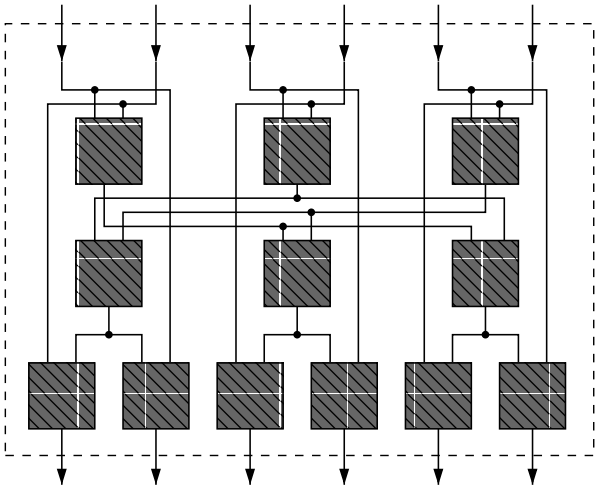
Anti-overflow techniques for TC and BMT FP numbers are discussed in Sections 4.3.3.1 and 4.3.3.2, respectively.

4.3.3.1 Two's Complement Number Representation

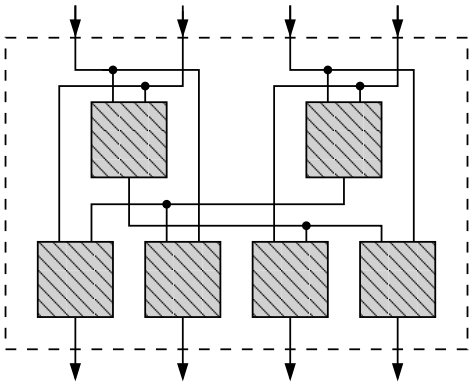
As discussed in Section 3.2.2, TC has been widely adopted in the implementation of LDPC decoders. However, overflow may occur to TC numbers having a bit width of W, when an arithmetic operation performed on FP numbers produces a result that exceeds the supported dynamic range, which may cause a positive number to become negative or *vice versa*. In an LDPC decoder implementation, ignoring this problem may increase the number of iterations required for the decoding process to converge, or may result in decoding failure [16, 10]. In order to mitigate the overflow problem, the overflowed TC



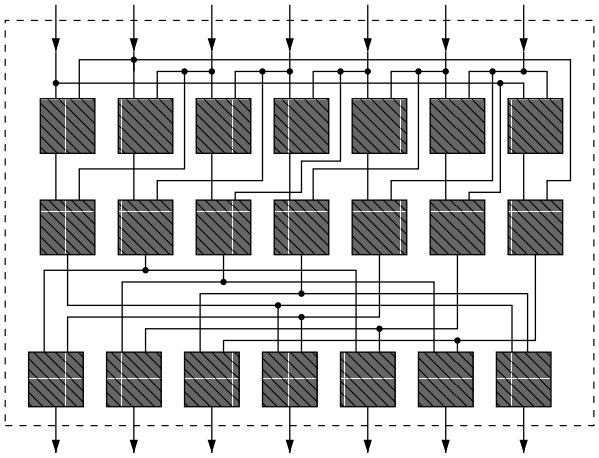
(a) The structure of nodes having 3 inputs.



(b) The structure of nodes having 6 inputs.



(c) The structure of nodes having 4 inputs.



(d) The structure of nodes having 7 inputs.

Figure 4.4: The proposed structure of nodes having $3, 4, 6$ and 7 inputs.

numbers may be saturated to the maximum or the minimum value supported, which has been demonstrated to be necessary for fixed-point LDPC decoders having limited bit width W [16, 10]. The FP EXIT analysis of Section 3.4 has revealed that a bit width W of 4 is sufficient for LDPC-FDs to achieve low computational complexity, while suffering only a slightly degraded BER performance. Since 4-bit TC numbers support decimal values in the range from -8 to 7, overflow may occur in both CNs and VNs of the MSA LDPC decoder employing 4-bit LLRs. For example, when adding $(6)_{\mathrm{d}}$, $(0110)_{\mathrm{b}}$, and $(5)_{\mathrm{d}}$, $(0101)_{\mathrm{b}}$, in a sub-VN, the desired decimal result $(11)_{\mathrm{d}}$ overflows to an incorrect value of $(-5)_{\mathrm{d}}$, $(1011)_{\mathrm{b}}$. By incorporating saturation into the operation of sub-VNs, the incorrect value of $(-5)_{\mathrm{d}}$ can be saturated to $(7)_{\mathrm{d}}$, $(0111)_{\mathrm{b}}$, in order to avoid the incorrect sign bit. Overflow occurs in a sub-CN when a boxplus operation is performed on a pair of inputs both equal to $(-8)_{\mathrm{d}}$, $(1000)_{\mathrm{b}}$, which has the desired value of $(8)_{\mathrm{d}}$, but which results in the overflowed value of $(0)_{\mathrm{d}}$, $(0000)_{\mathrm{b}}$. In order to avoid an incorrect sign bit, this overflowed value can be saturated to $(7)_{\mathrm{d}}$, $(0111)_{\mathrm{b}}$.

4.3.3.2 Base Minus Two Number Representation

Despite the widespread adoption of TC in digital circuit design, it has some disadvantages with respect to tolerance to timing errors. For example, in a full adder within a VN using TC, the sign bit is decided after the other bits, due to the ripple carry mechanism. Owing to this, the TC sign bit is most vulnerable to timing errors. However, LDPC decoders decide the decoding results based on the sign of the LLRs, which indicates that erroneous sign bits may significantly degrade the decoder's error-correcting performance. Motivated by this, we conducted a comprehensive investigation of several other number representations, in order to search for an alternative that is suited to timing error tolerance. Apart from two's complement number representation, this investigation have covered a range of standard number representations, including sign-and-magnitude, one's complement, offset binary and base minus two, and also a massive amount of random permutations of those existing ones. Through the exhaustive searching, we found that the BMT FP number representation is capable of achieving a superior timing error tolerance than that of TC. The BMT number representation is similar to TC, except that the weight of bits in BMT are powers of -2, rather than 2 as in TC. In the BMT number representation, a binary number having a bit width of W, $(b_{W-1}b_{W-2}\ldots b_1b_0)_{\mathrm{b}}$, is equal to the decimal number having the value of $\Sigma_{i=0}^{W-1}b_i(-2)^i$. Note that no specific bit indicates the sign of the number as in the TC number representation. In agreement with the discussion of Section 4.3.3, we recommend a BMT bit width of 4 bits, which has a supported range of decimal numbers from -10 to 5.

4.3.3.3 Boolean Expressions

In the case of $W = 4$, the 2 inputs to a sub-node have 256 combinations, because each of them has 4 bits. Therefore, the logic operation in a 2-input 1-output sub-node can be expressed by a 256×4 truth table or by 4 boolean expressions corresponding to the 4 bits of the node output. Note that this truth table and thus boolean expressions can natively consider built-in saturation, as explained previously in Section 4.3.3.1. The boolean expressions of sub-CNs using TC and BMT are shown in the first column of Figure 4.5, which demonstrate that the boolean expressions of BMT are much less complex than those of TC, when the operation of (2.19) is performed within the CNs. More explicitly, the most complex expression for TC has 25 terms, some of which involve 6 variables, while the corresponding expression of BMT only contains 15 terms with no more than 5 variables involved in each.

This relatively low complexity of boolean expressions translates into a relatively low complexity circuit, when computer-aided design is applied to the corresponding truth tables. Furthermore, each logic gate in the schematic can be translated into a cell containing multiple transistors, following placing and routing. This allows additional characteristics of the sub-nodes to be characterised, namely the area and energy consumption. Figure 4.5 compares the layout views[2] of a CN having a degree of $d_c = 6$ using the BMT and one using the TC number representation, which are constructed based on sub-CNs using the structure proposed in Figure 4.4(b) of Section 4.3.2.2. The detailed comparison on the timing, area characteristics, energy consumption and error-correcting performance will be discussed in Section 4.5. Here we list several numerical comparisons to prove the advantages previously discussed. When using BMT instead of TC, the longest path delay within CNs and VNs is reduced by 23%, and both the estimated area and power consumption of the decoder are reduced by around 10%. In Section 4.5.4, our design will be shown to be significantly more resilient to timing errors, owing to the nature of the BMT number representation and owing to the reduced critical path delay. For the reasons mentioned above, we propose to use the BMT number representation in our design for a fully-parallel LDPC decoder.

4.4 Overclocking-Induced Timing Error Analysis

As described in Section 4.1, timing errors may be imposed by overclocking, whenever there is insufficient time for the correct value of a signal to propagate to the input of a

[2]This result is reproduced from the work of the colleagues, Ke Li and Shida Zhong, as will be briefly introduced in Section 4.5.5.

Boolean expressions for sub-CN using TC

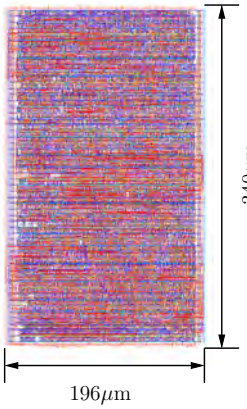
$$c_3 = \overline{a_3} \cdot a_0 \cdot b_3 + \overline{a_3} \cdot a_1 \cdot b_3 + \overline{a_3} \cdot a_2 \cdot b_3 + a_3 \cdot \overline{b_3} \cdot b_0 + a_3 \cdot \overline{b_3} \cdot b_1 + a_3 \cdot \overline{b_3} \cdot b_2$$

$$c_2 = \overline{a_3} \cdot a_2 \cdot b_2 + a_2 \cdot \overline{b_3} \cdot b_2 + \overline{a_3} \cdot \overline{a_2} \cdot a_0 \cdot b_3 + a_3 \cdot \overline{a_2} \cdot \overline{b_2} \cdot b_0 + \overline{a_3} \cdot \overline{a_2} \cdot a_1 \cdot b_3 + \overline{a_3} \cdot \overline{a_2} \cdot a_1 \cdot b_3 + a_3 \cdot \overline{a_2} \cdot b_3 \cdot \overline{b_2} + a_3 \cdot \overline{b_3} \cdot b_2 \cdot b_0 + a_2 \cdot \overline{a_1} \cdot \overline{a_0} \cdot b_3 \cdot \overline{b_2} + a_3 \cdot \overline{a_2} \cdot b_2 \cdot \overline{b_1} \cdot \overline{b_0} + a_2 \cdot \overline{a_1} \cdot \overline{a_0} \cdot b_2 \cdot \overline{b_1} \cdot \overline{b_0}$$

$$c_1 = \overline{a_3} \cdot a_1 \cdot \overline{b_3} \cdot b_1 + a_1 \cdot \overline{a_0} \cdot b_1 \cdot \overline{b_0} + a_2 \cdot a_1 \cdot \overline{b_3} \cdot b_1 + \overline{a_3} \cdot a_2 \cdot a_1 \cdot b_1 + \overline{a_3} \cdot \overline{a_2} \cdot \overline{a_1} \cdot a_0 \cdot b_3 + \overline{a_3} \cdot \overline{a_2} \cdot a_1 \cdot \overline{b_3} \cdot b_2 + \overline{a_3} \cdot a_2 \cdot \overline{b_3} \cdot \overline{b_2} \cdot b_1 + \overline{a_3} \cdot a_2 \cdot b_3 \cdot b_2 \cdot b_1 + a_2 \cdot \overline{a_1} \cdot a_0 \cdot b_3 \cdot \overline{b_2} + a_3 \cdot \overline{b_3} \cdot \overline{b_2} \cdot \overline{b_1} \cdot b_0 + a_3 \cdot \overline{a_2} \cdot b_2 \cdot \overline{b_1} \cdot b_0 + a_3 \cdot a_2 \cdot a_1 \cdot \overline{b_3} \cdot b_2 + a_1 \cdot \overline{a_0} \cdot b_3 \cdot \overline{b_2} \cdot \overline{b_1} + a_3 \cdot \overline{a_2} \cdot \overline{a_1} \cdot b_1 \cdot \overline{b_0} + \overline{a_3} \cdot \overline{a_2} \cdot a_0 \cdot \overline{a_0} \cdot b_3 + a_1 \cdot \overline{b_3} \cdot \overline{b_2} \cdot b_1 \cdot \overline{b_0} + \overline{a_2} \cdot a_1 \cdot b_2 \cdot b_1 \cdot \overline{b_0} + \overline{a_2} \cdot a_1 \cdot \overline{b_3} \cdot b_2 \cdot b_0 + a_2 \cdot a_1 \cdot \overline{a_0} \cdot \overline{b_2} \cdot b_1 + \overline{a_3} \cdot a_0 \cdot b_3 \cdot b_2 \cdot b_1 + a_3 \cdot \overline{a_2} \cdot \overline{b_1} \cdot b_3 \cdot \overline{b_2} \cdot \overline{b_1} + a_2 \cdot \overline{a_1} \cdot b_3 \cdot b_2 \cdot b_1 \cdot \overline{b_0} + a_3 \cdot \overline{a_1} \cdot b_3 \cdot b_2 \cdot \overline{b_1} \cdot b_0 + a_3 \cdot a_2 \cdot a_1 \cdot \overline{a_0} \cdot b_2 \cdot \overline{b_1} + a_3 \cdot a_2 \cdot \overline{a_1} \cdot a_0 \cdot b_3 \cdot \overline{b_1}$$

$$c_0 = a_0 \cdot b_0 + a_0 \cdot b_3 \cdot \overline{b_2} \cdot \overline{b_1} + a_3 \cdot \overline{a_2} \cdot \overline{a_1} \cdot b_0 + \overline{a_3} \cdot \overline{a_2} \cdot \overline{a_1} \cdot a_0 \cdot b_1 + a_1 \cdot \overline{b_3} \cdot \overline{b_2} \cdot \overline{b_1} \cdot b_0 + a_1 \cdot b_1 \cdot b_2 \cdot b_1 \cdot b_0 + a_3 \cdot a_2 \cdot a_1 \cdot a_0 \cdot b_1 + \overline{a_3} \cdot \overline{a_2} \cdot a_0 \cdot b_2 \cdot \overline{b_1} + a_2 \cdot \overline{a_1} \cdot \overline{b_3} \cdot \overline{b_2} \cdot b_0 + a_2 \cdot \overline{a_1} \cdot b_3 \cdot b_2 \cdot b_0 + a_3 \cdot a_2 \cdot a_0 \cdot b_2 \cdot \overline{b_1} + \overline{a_2} \cdot a_1 \cdot a_0 b_3 \cdot \overline{b_2} + a_2 \cdot \overline{a_1} \cdot a_0 \cdot b_3 \cdot \overline{b_2} + a_3 \cdot \overline{a_2} \cdot \overline{b_2} \cdot b_1 \cdot b_0 + a_3 \cdot \overline{a_2} \cdot b_2 \cdot \overline{b_1} \cdot b_0 + a_3 \cdot \overline{a_2} \cdot \overline{a_1} \cdot b_3 \cdot \overline{b_2} \cdot \overline{b_1} + \overline{a_2} \cdot a_1 \cdot a_0 \cdot \overline{b_3} \cdot b_2 \cdot b_1 + a_2 \cdot \overline{a_1} \cdot a_0 \cdot \overline{b_3} \cdot b_2 \cdot b_1 + \overline{a_3} \cdot a_2 \cdot a_1 \cdot \overline{b_2} \cdot b_1 \cdot b_0 + \overline{a_3} \cdot a_2 \cdot a_1 \cdot b_2 \cdot \overline{b_1} \cdot b_0$$

degree-6 CN using TC



Boolean expressions for sub-CN using BMT

$$c_3 = \overline{a_1} \cdot a_2 \cdot \overline{a_1} \cdot b_3 + \overline{a_3} \cdot a_2 \cdot a_0 \cdot b_3 + a_3 \cdot \overline{b_3} \cdot b_2 \cdot \overline{b_3} + a_3 \cdot \overline{b_3} \cdot b_2 \cdot b_0$$

$$c_2 = a_3 \cdot b_3 + a_2 \cdot \overline{a_1} \cdot b_3 + a_2 \cdot a_0 \cdot b_3 + a_3 \cdot b_2 \cdot \overline{b_1} + a_3 \cdot b_2 \cdot b_0 + \overline{a_2} \cdot a_1 \cdot \overline{a_0} \cdot b_3 + \overline{a_3} \cdot a_2 \cdot \overline{b_3} \cdot b_2 + a_1 \cdot \overline{b_2} \cdot b_1 \cdot \overline{b_0} + \overline{a_2} \cdot a_1 \cdot \overline{a_0} \cdot \overline{b_2} \cdot b_1 \cdot \overline{b_0}$$

$$c_1 = \overline{a_3} \cdot a_1 \cdot \overline{b_3} \cdot b_2 + \overline{a_3} \cdot a_1 \cdot \overline{a_0} \cdot b_3 + \overline{a_3} \cdot a_2 \cdot \overline{b_3} \cdot b_1 + a_3 \cdot \overline{b_3} \cdot b_1 \cdot \overline{b_0} + \overline{a_3} \cdot \overline{a_1} \cdot a_0 \cdot \overline{b_2} \cdot b_1 + \overline{a_3} \cdot \overline{a_2} \cdot \overline{a_1} \cdot a_0 \cdot b_3 + \overline{a_2} \cdot a_1 \cdot \overline{a_0} \cdot b_3 + a_3 \cdot \overline{b_3} \cdot \overline{b_2} \cdot \overline{b_1} \cdot b_0 + a_3 \cdot b_3 \cdot b_2 \cdot \overline{b_1} \cdot b_0 + a_3 \cdot a_2 \cdot \overline{a_1} \cdot a_0 \cdot b_1 + \overline{a_3} \cdot \overline{a_1} \cdot a_0 \cdot b_3 \cdot \overline{b_2} + \overline{a_3} \cdot \overline{a_1} \cdot a_0 \cdot b_3 \cdot b_1 + a_1 \cdot \overline{b_3} \cdot b_2 \cdot \overline{b_1} \cdot b_0 + a_3 \cdot \overline{a_2} \cdot \overline{b_3} \cdot \overline{b_1} \cdot b_0 + \overline{a_3} \cdot a_1 \cdot \overline{a_0} \cdot b_1 \cdot \overline{b_0}$$

$$c_0 = a_0 \cdot b_0 + a_0 \cdot b_3 \cdot \overline{b_2} + a_0 \cdot b_3 \cdot b_1 + a_3 \cdot \overline{a_2} \cdot b_0 + a_3 \cdot a_1 \cdot b_0 + \overline{a_3} \cdot \overline{a_2} \cdot a_0 \cdot b_1 + \overline{a_3} \cdot \overline{a_2} \cdot a_0 \cdot b_2 + a_1 \cdot \overline{b_3} \cdot \overline{b_2} \cdot b_0 + a_2 \cdot \overline{b_3} \cdot \overline{b_2} \cdot b_0 + a_3 \cdot \overline{a_2} \cdot b_3 \cdot b_1 + a_3 \cdot a_1 \cdot b_3 \cdot \overline{b_2} + a_3 \cdot a_1 \cdot b_3 \cdot b_1 + \overline{a_3} \cdot a_1 \cdot a_0 \cdot b_2 \cdot \overline{b_1} + a_2 \cdot \overline{a_1} \cdot \overline{b_3} \cdot b_1 \cdot b_0 + a_2 \cdot \overline{a_1} \cdot b_3 \cdot b_2 \cdot \overline{b_1} \cdot b_0 + a_3 \cdot a_2 \cdot \overline{a_1} \cdot a_0 \cdot b_2 \cdot \overline{b_1}$$

degree-6 CN using BMT

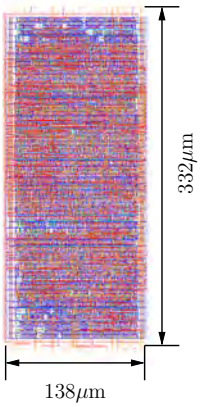


Figure 4.5: Comparison of boolean expressions for sub-CNs using TC and BMT. The layout view for CNs having a degree of $d_c = 6$ using TC and BMT is also compared, when IBM 130nm process technology is employed. Note that the saturation is employed in the operation of each sub-CN.

memory, before the arrival of the upcoming clock edge. When aggressive overclocking is employed, the clock period T_{clk} is reduced below the nominal propagation delay t of a signal, typically imposing timing errors. However, even moderate overclocking may cause timing errors, since this makes the circuit more sensitive to the late arrival of signals. The propagation delays of circuit paths may fluctuate from one clock cycle to the next, typically caused by noise imposed on the power rails [103]. This noise is caused by parasitic resistance and inductance in the power distribution network [104, 105]. These noise sources, known as IR-drop and LdI/dt noise, are related to both the mean current drawn from the power supply and to the instantaneous changes in current, respectively [105]. Changes in mean and instantaneous current are mainly determined by the switching activity levels in the circuit, which in turn depend on the combination of input vectors present during each clock cycle. Formally, a timing error occurs for a particular signal when

$$t \times \delta > T_{\text{clk}}, \tag{4.1}$$

where δ characterizes the fluctuation in the propagation delay within the current clock cycle. In Section 4.4.1.2, we describe methods to obtain the nominal propagation delay t within CNs and VNs. Following this, the fluctuation δ of signal propagation delays from one clock cycle to the next is selected from a normalized Probability Density Function (PDF). In Section 4.4.2, we propose an error model which can predict the timing errors, based on the timing analysis. Finally, this model is utilized to guide the simulations in Section 4.5.4.

4.4.1 Timing analysis

We commence in Section 4.4.1.1 by discussing the methods used to obtain the nominal path delays t within various different LDPC decoder architectures. In Section 4.4.1.2, we discuss the methods used to quantify the fluctuation δ in these propagation delays, caused by the supply noise.

4.4.1.1 Nominal Propagation Delay

Let us begin by characterizing the nominal signal propagation delays t of CNs and VNs having different degrees, when employing the number representations and nodes structures discussed in Sections 4.3.2 and 4.3.3. This is considered for two different scales of fabrication technologies, namely IBM 130 nm technology [106] and STMicroelectronics 90 nm technology [107]. Note that the delay of a path includes the output delay of the

D-FF at the beginning of the path and the total delay of the combinational logic along the path. By contrast, the hold time of the D-FF at the beginning and the setup time of the D-FF at the end of the path are not considered to be part of the path delay, since these are instead used to model the occurrence of timing errors. Unlike the delay of the paths, the setup and hold times do not vary from one clock cycle to another. The values of the nominal propagation delay t are characterized in Section 4.5.1, when IBM 130nm technology is used as an example. The comparison is also conducted and discussed in Section 4.5.1 for CNs and VNs having different degrees, when employing different number representations and nodes structures.

4.4.1.2 Fluctuation in Propagation Delay

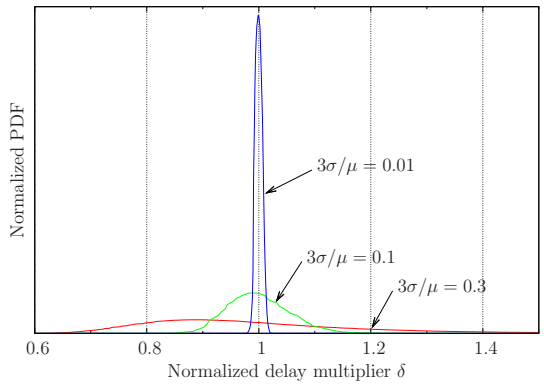


Figure 4.6: Time-normalized NAND gate delay PDF for $3\sigma/\mu = \{0.01, 0.1, 0.3\}$ when employing STMicroelectronics 90 nm technology, where $\mu = 1$ V.

In our timing error model, the value of the supply voltage V_{DD} is assumed to be randomly distributed and independent in each clock cycle, since different gates are switched in each successive clock cycle [104, 105]. We further assume that the physical design (layout of the power grid) is fully optimized, and that sufficient on-chip bypass capacitors are inserted. Owing to this, the supply voltage V_{DD} may be assumed to have a Gaussian distribution, since the fully parallel LDPC decoder contains a sufficiently high number of gates. The mean value of the Gaussian distributed supply voltage is equal to the nominal supply voltage μ, while the value of the standard deviation σ depends on the supply voltage noise, circuit layout, fabrication process and technology scale [104, 105]. Finally, we assume that during each clock cycle, each gate within the decoder is driven by the the same supply voltage V_{DD} value, that was randomly selected from the Gaussian distribution [103]. Since the propagation delay of a particular type of gate is a decreasing function of the supply voltage V_{DD}, it can be modelled as a random variable that is independently selected from a particular PDF in each successive clock cycle. Since the normalized distributions of all gates types are very similar [103],

the normalized propagation delay distribution of a NAND gate was adopted to model the propagation delay distribution of all circuit components, as recommended in [103]. Figure 4.6 shows the time-normalized NAND gate delay PDF for $3\sigma/\mu \in \{0.01, 0.1, 0.3\}$. For example, when $3\sigma/\mu = 0.1$, the propagation delay of a NAND gate extends above its nominal value by 10% or more with a probability of around 1%. Correspondingly, the probability of a NOT gate's propagation delay extending by 10% or more above its nominal value is also around 1%, when $3\sigma/\mu = 0.1$. This normalized distribution has been verified to be accurate by SPICE simulation and is capable of simplifying the analysis of the following sections.

4.4.2 Proposed error model

Based on the previous timing analysis, we model timing errors by modifying particular bits in the fixed-point number representations of the LLRs exchanged within the LDPC decoder in each clock cycle. This is implemented by comparing the simulated fluctuation propagation delay to the clock period, according to

$$\delta t < T_{\text{clk}} - T_{\text{setup}} \qquad \text{no error,} \tag{4.2}$$

$$T_{\text{clk}} - T_{\text{setup}} \leq \delta t \leq T_{\text{clk}} + T_{\text{hold}} \qquad \text{type I error,} \tag{4.3}$$

$$T_{\text{clk}} + T_{\text{hold}} < \delta t \qquad \text{type II error.} \tag{4.4}$$

Explicitly, the model categorizes the timing errors into two types. (4.2) represents the case where the propagation delay of a signal on a particular path during a particular clock cycle, δt, is small compared to the clock period T_{clk}. In this case, the signal is correctly clocked into the D-FF on time. (4.3) represents the case where δt is increased so that the signal's arrival is located in the metastability window, namely the duration from $T_{\text{clk}} - T_{\text{setup}}$ to $T_{\text{clk}} + T_{\text{hold}}$. This causes the D-FF to become potentially metastable. However, this metastability can be resolved by the double flip-flop chain, as described in Section 4.3.1.1. This provides an additional clock cycle to allow the metastability to resolve to a valid but randomly selected logic value again, causing a Type I error. To model a Type I error, during our simulations, a random bit value is clocked into the second D-FF. (4.4) represents the case where the propagation delay δt is further increased to be larger than $T_{\text{clk}} + T_{\text{hold}}$. In this case, we assume that different parts of the logic propagate asynchronously, causing glitches to appear at the input of the first D-FF throughout the metastability window. Owing to this, the resultant Type II error is modelled as a random bit being clocked into the second D-FF.

4.4.3 Validation of the error model

The proposed error model was validated using simulation in Cadence which is considered to be the most accurate tool in both academic and industry [46, 108]. This approach is exemplified by the results of Figure 4.7. More specifically, we simulated the case of for a TC-based CN having a degree of $d_c = 7$ to decode two independent codewords simultaneously, as discussed in Section 4.3. The supply voltage V_{DD} was set at the nominal value of 1.5 V, while the clock frequency was set as 800 MHz, which is an extreme condition that allows the observation of metastability. As may be observed in Figure 4.7, the first D-FF misses the switching input signal, during the first two clock cycles, then clocks incorrect values, causing two instances of Type II error. During the 9th clock cycle shown in Figure 4.7, the first D-FF enters the metastable state, due to the dramatic switching at the input before the clock edge. However, this metastability resolves to the valid logic value "1" within the clock cycle, and this incorrect value is clocked into the second D-FF at the next clock edge, cauinsg a Type I error.

4.5 Comparison of Implementations

In this section, the timing, area and power consumptions characteristics of CNs and VNs in the (1056,528) WiMAX LDPC are compared for cases of using different number representations and node structures, with the IBM 130nm process technology[106]. More explicitly, we investigate VNs have degrees of 2, 3 and 6, as well as CNs having degrees of 6 and 7, which are associated with the distributions shown in Table 4.3. Note that similar results were observed when the STMicroelectronics 90 nm process technology[107] is employed. We commence by comparing the timing characteristics in Section 4.5.1. Following this, the area characteristics and the energy consumption of each type of CN and VN compared in Sections 4.5.2 and 4.5.3, respectively. Motivated by the discussion of Section 4.2, we characterize the impact of timing errors upon the error correction capability of the proposed fully parallel LDPC decoder, as well as of several benchmarkers, in Section 4.5.4.

4.5.1 Timing Characteristics

In this subsection, we compare the timing characteristics of individual CNs and VNs using the BMT representation to those using the TC representation. The proposed structure of Figure 4.4 for constructing large-degree CNs and VNs is also compared to the forward-backward structure of Figure 4.3(a), in terms of the timing characteristics

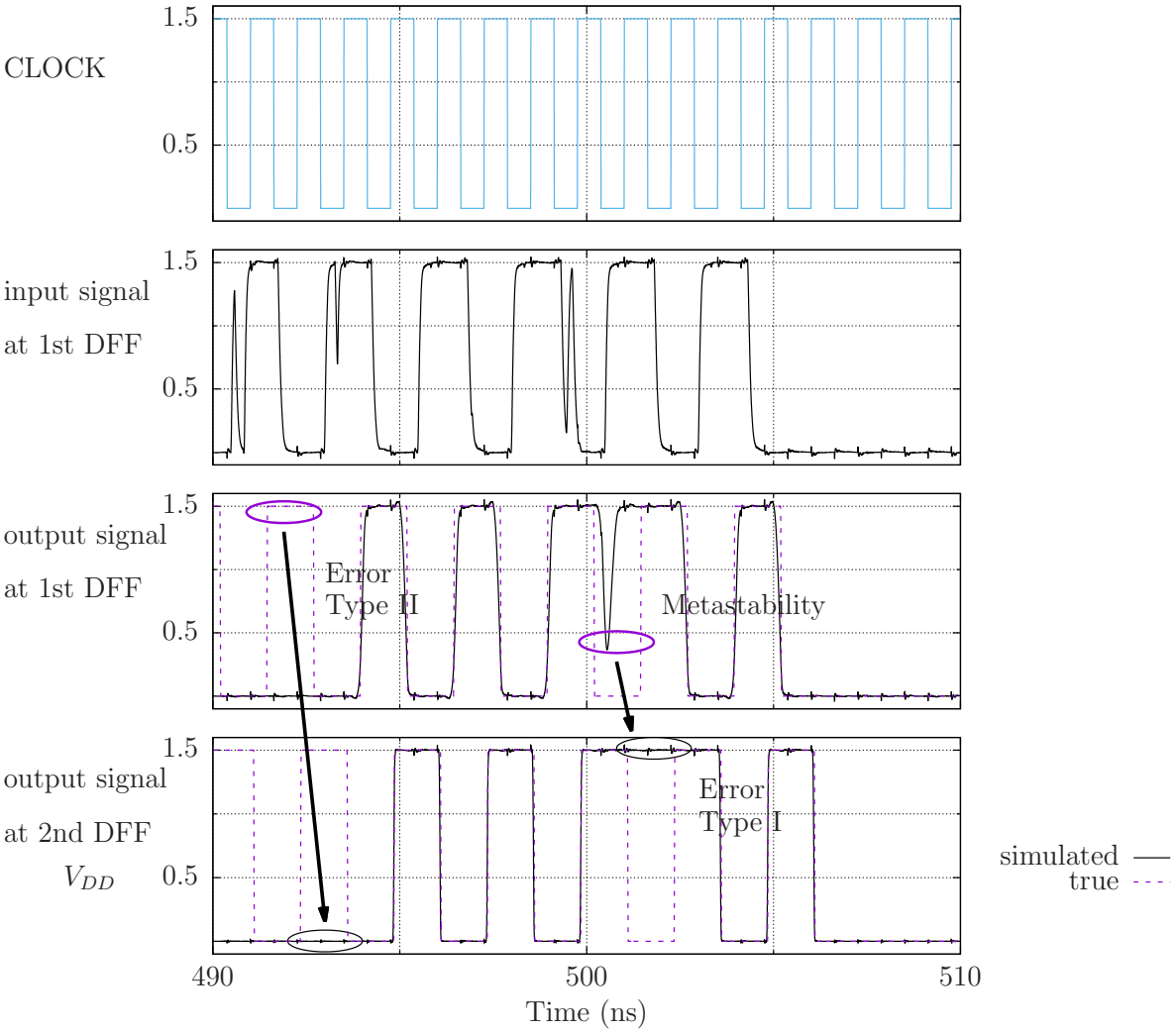


Figure 4.7: Timing diagrams demonstrating two types of timing errors, obtained from post-layout simulation in Cadence.

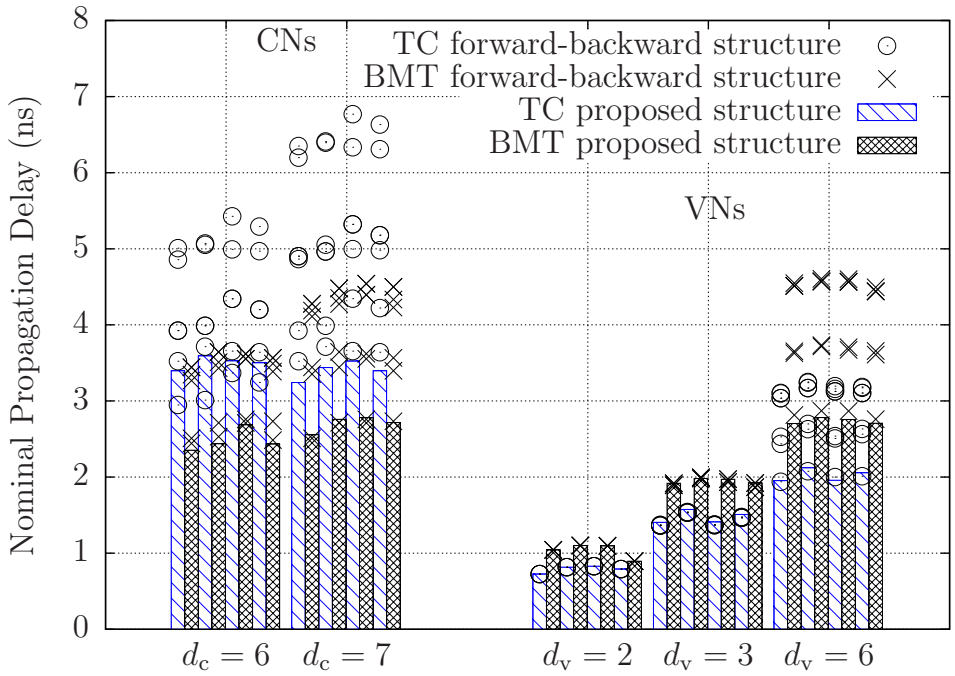


Figure 4.8: Comparison of timing characteristics for all combinations of CNs and VNs using BMT and TC, as well as the proposed node structures of Figures 4.3(a) and the forward-backward structure of Figure 4.4, for the case when IBM 130 nm technology is employed.

of CNs and VNs. We characterize the nominal propagation delay t from the synthesised schematics, according to the discussion of Section 4.4.1.1. The obtained values of the nominal propagation delay t within CNs and VNs using IBM 130nm technology [106] are plotted in Figure 4.8. Explicitly, Figure 4.8 plots the nominal propagation delay on each of 4 bits at one output port of every type of CN and VN, where the bars represent the delay for the proposed structure and the discrete marks represent the delay for the forward-backward structure. The longest critical path delay among the 4 path delays for every type of CNs and VNs is also shown in the row of 'longest critical path delay' of Table 4.3. The numbers in and outside of brackets in Table 4.3 present the results for the forward-backward and proposed node structures of Figures 4.3(a) and 4.4, respectively. Based on these results, the comparison of TC and BMT representations is discussed in Section 4.5.1.1 and the comparison of node structures is discussed in Section 4.5.1.2.

4.5.1.1 Comparison of Number Representations

Following the discussion in Section 4.3.3.2, the BMT number representation is demonstrated to achieve a lower value of the longest critical path delay within CNs and VNs having various degrees, as shown in Table 4.3. It may be observed that BMT decreases the critical path delay of all CNs and VNs to be lower than 3 ns, when using the pro-

Table 4.3: Comparison of critical path delay, estimated area and power consumption for CNs and VNs, using different combinations of BMT and TC, as well as the proposed node structures of Figures 4.4 and the forward-backward structure of Figure , for the case when IBM 130nm technology is employed. The numbers in brackets correspond to the forward-backward structure of Figure 4.3(a).

node type	FP rep.	CN		VN			Total
degree	-	6	7	2	3	6	-
distribution	-	2/3	1/3	11/24	1/3	5/24	-
# sub-nodes	TC&BMT	12 (12)	21(15)	3(3)	6(6)	21(15)	-
longest critical path delay (ns)	TC	3.60 (5.43)	3.52 (6.77)	0.83 (0.83)	1.57 (1.54)	2.12 (3.25)	-
	BMT	2.69 (3.65)	2.78 (4.54)	1.10 (1.10)	1.98 (2.00)	2.78 (4.61)	-
area(μm^2)	TC	7048.80 (7067.52)	11515.68 (8706.24)	1507.68 (1507.68)	2485.44(2485.44)	6834.24 (5418.72)	7.62 (6.82)$\times 10^6$
	BMT	5348.16 (5348.16)	8426.88 (6553.44)	1658.88 (1658.88)	2787.84 (2787.84)	7902.72 (6174.72)	6.89 (6.18)$\times 10^6$
power(μW)	TC	19.35 (19.34)	26.49 (22.95)	6.78 (6.79)	9.89 (9.89)	21.09 (19.34)	22.87 (21.86)$\times 10^3$
	BMT	15.67(15.75)	20.76 (18.53)	6.85 (6.85)	10.00 (10.00)	21.23 (19.20)	20.68 (19.87)$\times 10^3$

posed node structures of Figure 4.4. More explicitly, the longest critical path delay for CNs using BMT instead of TC, are decreased by up to 25%. By contrast, the longest critical path delay for VNs having degrees of d_v =2, 3 and 6 are increased by 33%, 26% and 10%, respectively, when using BMT instead of TC. However, the increased critical path delays are still lower than those of the CNs. Owing to this, the critical path delay for the LDPC decoder is decreased by 23% overall. When the BMT representation is employed in the forward-backward node structures of Figures 4.3(a), similar reduction in the longest critical path delay within CNs and similar increases in the longest critical path delay within VNs are observed, compared to those using TC.

4.5.1.2 Comparison of high-degree Node Structures

The numbers outside of brackets in Table 4.3 represent the numerical results of CNs and VNs using the proposed high-degree node structure of Figure 4.4. Meanwhile, those in brackets represent the numerical results for CNs and VNs using the forward-backward structure of Figure 4.3(a). It may be observed that the longest critical path delays for all CNs and VNs are significantly lower in the proposed structure, regardless of whether BMT or TC is used. The longest critical path delay is typically reduced by up to 38.8% in the case of the CNs having a degree of $d_c = 7$, when BMT and the proposed structure is employed. Moreover, Figure 4.8 provides detailed timing characteristics for each of the 4 bits on each output port of all types of CNs and VNs. When the proposed structure is used, all output ports employ equivalent combinations of inputs. Owing to this, the difference between delays of 4 bits at different output ports is negligible as discussed in Section 4.3.2. For this reason, Figure 4.8 quantifies only the delays associated with a single output port of a particular node, using a cluster of 4 bars, each of which corresponds to a different bit in the operand. The different shade patterns of the bars indicate whether the results correspond the use of BMT or TC in that node. By contrast, when the forward-backward structure of Figure 4.3(a) is used, the various output ports are associated with different methods of combining the inputs. Owing to this, Figure 4.8 quantifies the delays of all output ports of each node using groups of markers, where each group comprises 4 markers representing the delays associated with the 4 bits of the operand at each output. Similarly, different markers are used to indicate whether the results correspond to the use of BMT or TC in that node. It may be observed that all markers are located above the bars, indicating that the proposed structure of Figure 4.4 reduces the propagations delays to a minimum value at every output port, regardless of which number representation is used.

Due to the various implementations of the inverse operation that may be used in

the parallel node structure of Figure 4.3(b), it is difficult to precisely quantify the timing results, as well as chip area and power consumption. However, as discussed in Sections 4.3.2, the proposed structure of Figure 4.4 is guaranteed to achieve a lower critical path delay at the cost of a slightly increased chip area, based on the similarities between these structures and the additional cost of the complex inverse operation.

4.5.2 Area

The area for CNs and VNs using BMT and TC are obtained from the synthesis processing on the schematics of Figures 4.3(a) and 4.4, when IBM 130nm technology is employed. First, we compare the area for different CNs and VNs using BMT and TC, shown as numbers outside brackets in Table 4.3, when the proposed structure of Figure 4.4 is employed. It may be observed that BMT reduces the total area consumption by up to 27% in CNs having a degree of $d_c = 7$, at the cost of increasing it by up to 16% in VNs having a degree of $d_v = 7$. However, the area of a VN having a degree of $d_v = 6$ is still less than that for a CN having a degree of $d_c = 6$. This was also exemplified by the comparison of chip layouts of CNs having a degree of $d_c = 7$ shown in the right column of Figure 4.5, which exhibits a 31% reduction after placing and routing, when using BMT instead of TC. More explicitly, the CNs having a degree of $d_c = 7$ using BMT require around 34.6% fewer transistors, compared to the number required when using TC. Following the principle discussed in Section 4.2, the area of the full decoder is estimated by summing those of every CN and VN, as shown in the right-most column in Table 4.3, instead of by synthesising the full decoder's schematic. Note that the estimated full decoder's area excludes the consideration of interconnecting wires and other components. In this case, the total area of the decoder using BMT instead of TC is reduced by 10%, when the proposed node structure is employed. When the forward-backward structure of Figure 4.3(a) is employed, similar reduction of area in CNs and increase in VNs are observed when using BMT instead of TC, shown as numbers in the brackets of Table 4.3.

Table 4.3 also demonstrates that the proposed node structure consumes larger area only in CNs having a degree of $d_c = 7$ and VNs having a degree of $d_v = 6$, compared to the forward-backward structure, when the same number representation is employed. More explicitly, the area is increased by 29% and 28% for CNs having a degree of $d_c = 7$ and VNs having a degree of $d_v = 6$, respectively, when both use BMT. This comparison can also be presented from the total number of sub-nodes consumed by CNs having a degree of $d_c = 7$ and VNs having a degree of $d_v = 6$, as shown in Table 4.3. However, no significant differences are found in CNs and VNs having other degrees. Note that

the CNs having a degree of $d_c = 7$ using the proposed structure and BMT achieve similar areas to those using the forward-backward structure and TC, which verifies the discussions in Section 4.3.2.2. The area of the parallel structure of Figure 4.3(b) is excluded from the comparison, due to the unfair comparison caused by the inverse operation, as discussed previously in Section 4.3.2.

4.5.3 Power Consumption

Similarly, the energy consumption results for CNs and VNs using BMT and TC number representations from the synthesis processing on the schematics of Figures 4.3(a) and 4.4 are compared in Table 4.3, when IBM 130nm technology is employed. As a result of the area reduction of the proposed node structure of Figure 4.4, the energy consumption for CNs having degrees of $d_c = 6$ and 7 using BMT instead of TC is decreased by around 19% and 22%, respectively. However, the energy consumption of the VNs having a degree of $d_v = 6$ using BMT is increased by less than 1%, despite the 16% area increase observed previously. This indicates that the inputs of VNs having a degree of $d_v = 6$ using BMT may have less switching activity than those using TC. No significant differences in energy consumption were observed between the VNs having degrees of $d_v = 2$ and 3, when using BMT instead of TC. The estimated power consumption for the decoder using BMT instead of TC is reduced by around 10%, which is obtained by summing the power consumption of each CN and VN. When the forward-backward node structure of Figure 4.3(a) is employed, similar results are observed for BMT and TC.

Another comparison between the proposed and forward-backward node structures may be conducted by comparing the numbers inside and outside the brackets in Table 4.3. Only a slight 12% increase is observed for CNs having a degree of $d_c = 7$ using the proposed structure of Figure 4.4 instead of the forward-backward structure of Figure 4.3(a), when BMT is employed.

4.5.4 BER Results and Discussions

In this section, BER results were obtained using the Monte-Carlo simulation at the FP algorithm level, where timing errors are injected according to the proposed model of Section 4.4.2. The same (1056,528) LDPC code is considered here as an example, and BPSK is employed for transmission over an Additive White Gaussian Noise (AWGN) channel. All the BER results are simulated by sending at least 10^6 codewords encoded from random information bits over a noisy AWGN channel, then decoding the received

codewords iteratively using the MSA. This continues until early termination is triggered by the occurrence of an all-zero syndrome, or until the maximum affordable number of clock cycles is reached. The decoder decodes two independent codewords simultaneously as discussed in Section 4.3.1. The maximum number of clock cycles is limited to 100, which is large enough to achieve the iterative decoding convergence at the low E_b/N_0 values. However, since the early termination cannot be anticipated in advance, the amount of time that is reserved for decoding each codeword is given by $100T_{clk}$. This corresponds to a processing throughput of $R_b = \frac{2K}{100T_{clk}}$ information bits per second. Note that the decoders adopt saturation, as described in Section 4.3.3.

In Section 4.5.4.1, we list the LDPC decoder parametrizations that are considered in the simulations. The BER performances of these decoders in the absence of timing errors are compared and discussed in Section 4.5.4.2, providing the theoretical bound on the BER performance that can be achieved when timing errors are considered. The corresponding BER results in the presence of overclocking-induced timing errors are demonstrated in Section 4.5.4.3, which considers the timing characteristics of Section 4.4.

4.5.4.1 Proposed Scheme and Benchmarkers

Six types of fully parallel fixed-point LDPC decoder architectures will be considered in the following sections, as follows.

- Decoder A: The proposed scheme, which employs BMT number representation with double D-FFs *only* on the inputs and outputs of VNs having a degree of 6, as well as CNs.
- Decoder B: A benchmarker, which employs the BMT number representation with D-FFs on the inputs and outputs of *every* VN and CN.
- Decoder C: A benchmarker, which employs the BMT number representation with D-FFs *only* on the outputs of *every* VN and CN.
- Decoder D: A benchmarker, which employs the TC number representation with D-FFs *only* on the inputs and outputs of VNs having a degree of 6, as well as CNs,
- Decoder E: A benchmarker, which employs the TC number representation with D-FFs on the inputs and outputs of *every* VN and CN.

Table 4.4: Simulation parameters.

μ	1.5 V
$3\sigma/\mu$	$\{0.01, 0.3\}$
T_{clk}	$\{1, 2, 3, 4\}$ ns
bit width	4 bits
number representation	TC, BMT
high-degree node structures	proposed in Figure 4.4
max num. of T_{clk} per codeword	100
anti-overflow	saturation

- Decoder F: A benchmarker, which employs the TC number representation with D-FFs *only* on the outputs of *every* VN and CN.

Note that all decoders adopt the proposed node structures illustrated in Figure 4.4, with a bit width of $W = 4$. Each of the six decoders is capable of decoding two independent codewords simultaneously, while Decoders B and E are also capable of decoding four codewords at a time, since they employ D-FFs on both ends of every interconnecting wire. However, since Decoders C and F employ only a single D-FF on each interconnecting wire, they do not have the protection against the catastrophic propagation of metastability, as the other 4 decoders have. A limit of 100 clock cycles is employed, in order to make fair comparison with the proposed schemes. All of the parameters used in our simulations are shown in Table 4.4.

4.5.4.2 Simulation and Analysis in The Absence of Timing Errors

The error-correcting performance in the absence of timing error is investigated as follows, when the clock frequency is set sufficiently high to avoid possibility of timing errors. Figure 4.9 plots the BER performance of the six decoders listed in Section 4.5.4.1, when the maximum number of clock cycles per codeword is limited to 100. It may be observed that in each pairing of Decoders A and D, B and E, as well as C and F, the schemes achieve very similar error-correcting performance to each other, when employing the same maximum number of clock cycles. This demonstrates that the BMT and TC number representations having a bit width of $W = 4$ give very similar error correction performance, in the absence of timing errors. However, Decoders A and D achieve iterative decoding convergence at lower E_{b}/N_0 values than Decoders B and E, because the former schemes exchange more LLRs within fewer clock cycles, owing to their employment of fewer D-FFs. Decoder C and F require the highest E_{b}/N_0 values to achieve iterative decoding convergence, compared to the other two pairs.

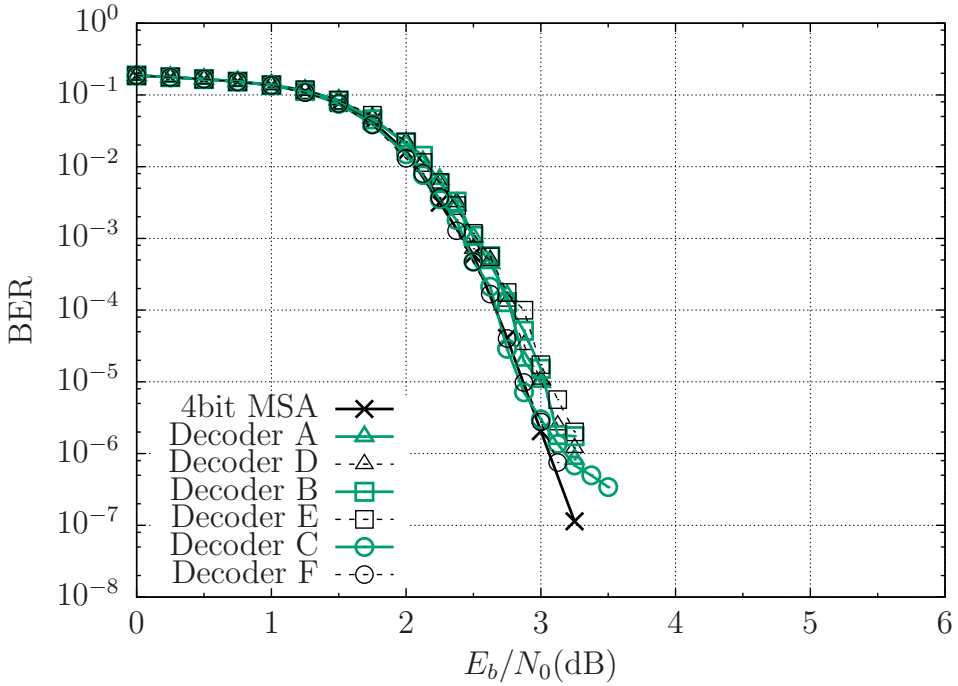
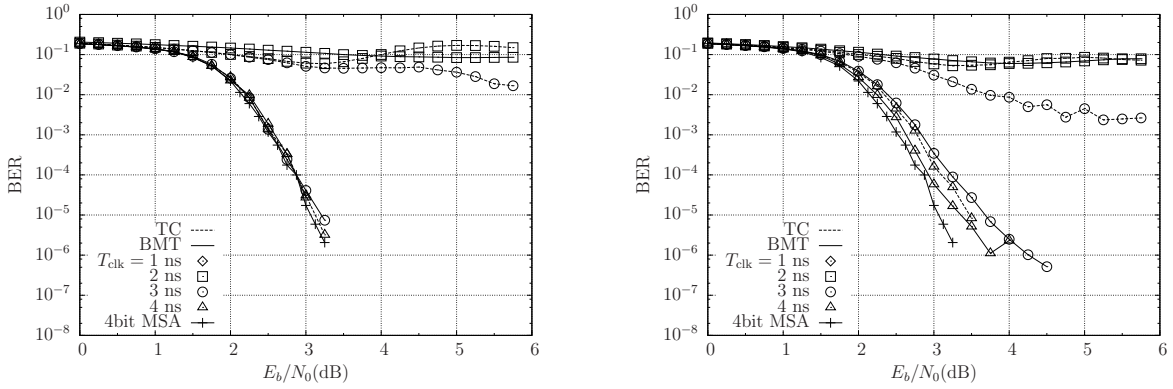


Figure 4.9: BER plots of the proposed architecture compared with several benchmarkers in the absence of timing errors.

4.5.4.3 Simulation and Analysis in The Presence of Timing Errors

In this section, the timing error tolerance of Decoders A, B, and E is investigated in the presence of timing errors caused by aggressive overclocking. Note that Decoder C and F are not considered for this case, since they are vulnerable to catastrophic metastability propagation, due to the single D-FF employed between each pair of CN and VN. The consideration of Decoder D is redundant, since it employs the combination of both inferior schemes in terms of timing error tolerance, namely TC and D-FFs *only* on the inputs and outputs of VNs having a degree of 6, as well as CNs, as will be demonstrated in the following subsections. This analysis relies on the error model and timing analysis described in Section 4.4. In order to investigate the timing errors tolerance comprehensively, the scaled clock period T_{clk} is selected from the set of $\{1, 2, 3, 4\}$ ns, while the supply voltage standard deviation is selected such that $3\sigma/\mu$ is equal to $\{0.01, 0.3\}$ [103], when μ is 1.5 V, as shown in Table 4.4. More explicitly, compared to the 2.78 ns critical path delay of Decoder A, $T_{\text{clk}} = 1$ and 2 ns represent aggressive overclocking, while $T_{\text{clk}} = 3$ and 4 ns represent moderate overclocking. As explained in Section 4.4.1.2, $3\sigma/\mu = 0.01$ represents a very small variation in the value of δ, which models the scenario where the path delay varies little between clock cycles, while $3\sigma/\mu = 0.3$ represent the scenario where the path delay may vary dramatically from one clock cycle to another.



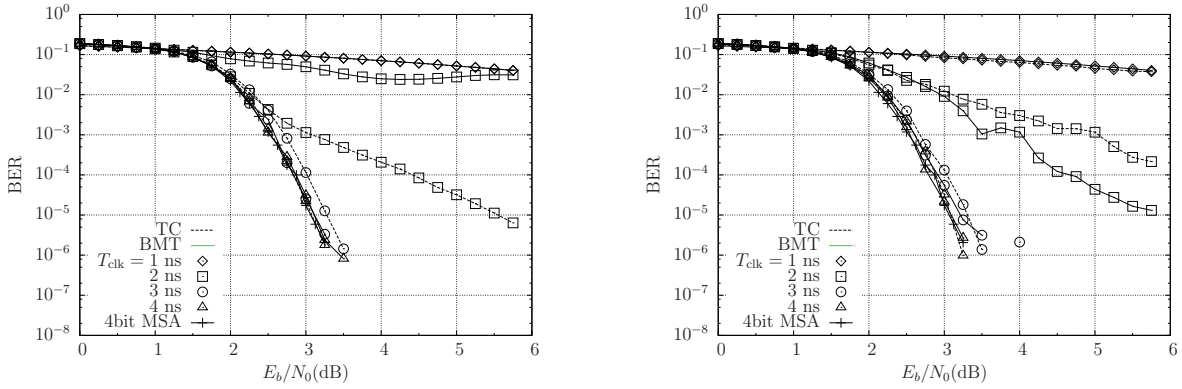
(a) $3\sigma/\mu = 0.01$ (b) $3\sigma/\mu = 0.3$

Figure 4.10: BER plots of the proposed architecture compared with 3 benchmarkers with the presence of timing errors, when $3\sigma/\mu$ equals to (a) 0.01 and (b) 0.3, for the case when IBM 130 nm technology is employed.

Decoder B and E Let us begin by comparing the BER performance of Decoders B and E for different degrees of overclocking, as shown in Figures 4.10(a) and 4.10(b). Figure 4.10(a) characterizes the scenario when $3\sigma/\mu = 0.01$. It may be observed that Decoder B achieves iterative decoding convergence to the same BER as the sub-optimal MSA benchmarker, when a clock period T_{clk} of 3 ns is employed. By contrast, Decoder E requires $T_{\text{clk}} = 4$ ns in order to achieve this BER performance. Indeed, this BER performance gap between Decoders B and E at a BER of 10^{-5} is around 2.5 dB, when $T_{\text{clk}} = 3$ ns. However, in the case of aggressive overclocking where the clock period is set to 1 and 2 ns, both decoders fail to converge at a low BER.

In the case of $3\sigma/\mu = 0.3$, where the path delays vary significantly from one clock cycle to the next, the BER performance of Decoders B and E are degraded significantly compared to the corresponding schemes in the absence of timing errors, as shown in Fig 4.10(b). More explicitly, Decoder B achieves a BER of 10^{-5} within 0.6 dB of the performance achieved in the absence of timing errors, when the clock period T_{clk} is 3 ns. This gap closes to around 0.2 dB, when T_{clk} is increased to 4 ns. By contrast, Decoder E only approaches a BER of 10^{-5} within 0.3 dB of the performance achieved in the absence of timing errors, when T_{clk} is 4 ns.

Based on Figures 4.10(a) and 4.10(b), it may be observed that Decoder B is capable of achieving the same BER performance of Decoder D, when operating with a lower T_{clk}. In other words, the fully parallel LDPC decoder using BMT has a greater tolerance to timing errors than the corresponding scheme using the TC number representation. This allows T_{clk} to be reduced by around 1 ns, which corresponds to an increase of



(a) $3\sigma/\mu = 0.01$ (b) $3\sigma/\mu = 0.3$

Figure 4.11: BER plots of the proposed architecture compared with 3 benchmarkers in the presence of timing errors, when $3\sigma/\mu$ equals to (a) 0.01 and (b) 0.3, for the case when STMicroelectronic 90 nm technology is employed.

around 33.3% in the processing throughput from 1.3 Gbit/s to 1.8 Gbit/s.

The same timing analysis and error model can be applied to fabrication technologies of other scales. Figures 4.11(a) and 4.11(b) provide the corresponding BER results, when the STMicroelectronic 90nm technology is used instead of the 130nm IBM technology. Similar BER performances can be observed for Decoders B and E, offering the similar conclusion that BMT is preferable to TC with respect to timing error tolerance.

Decoder A and B Since BMT was identified as being preferable to TC in our previous discussions, we compare the BER performance of the BMT-based Decoders A and B, when $3\sigma/\mu = 0.3$ and $T_{\text{clk}} = 3$ ns, as shown in Fig 4.12. It may be observed that the BER performance of Decoder A only exhibits a negligible degradation of less than 0.1 dB compared to Decoder B at a BER of 10^{-5}. Furthermore, both decoders closely approach the performance of the corresponding schemes in the absence of timing errors. Here, the slight discrepancy may be explained by the dramatically varying path delays when $3\sigma/\mu = 0.3$, which increases the occurrence of timing errors. However, the number of D-FFs used in Decoder A is 9328, which is 30% fewer than that 13376 D-FFs used in Decoder B. Therefore, we recommend the architecture of Decoder A, which is a fully parallel LDPC-FD using the BMT number representation, our novel node structure and D-FFs placed only on the inputs and outputs of high-degree nodes.

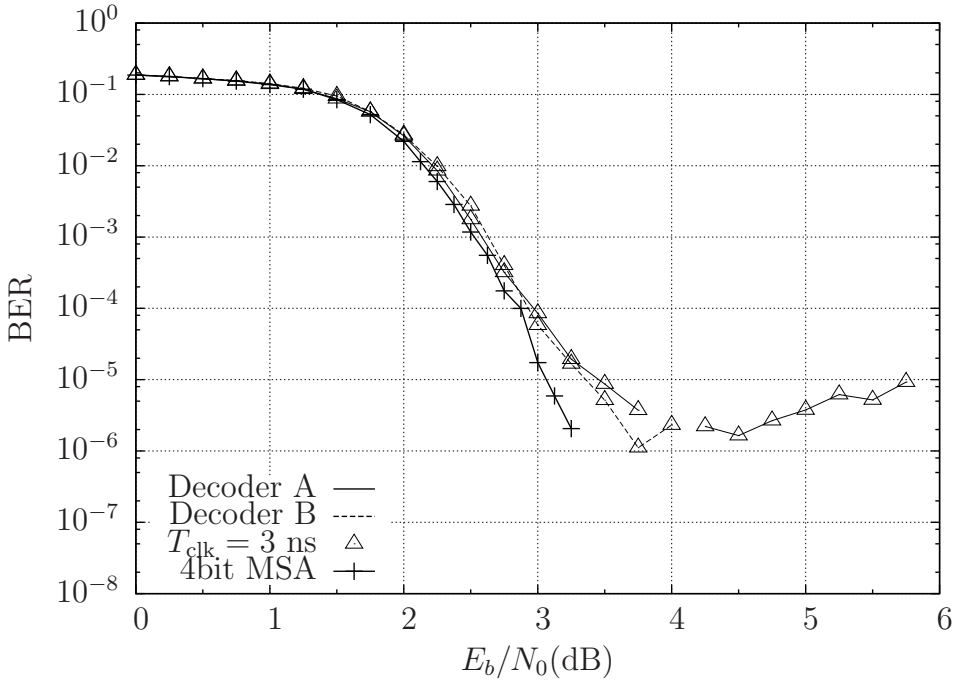


Figure 4.12: BER plots of different D-FF schemes in the presence of timing errors, when $3\sigma/\mu$ is 0.3 and $T_{\text{clk}} = 3$ ns, for the case where IBM 130 nm technology is employed.

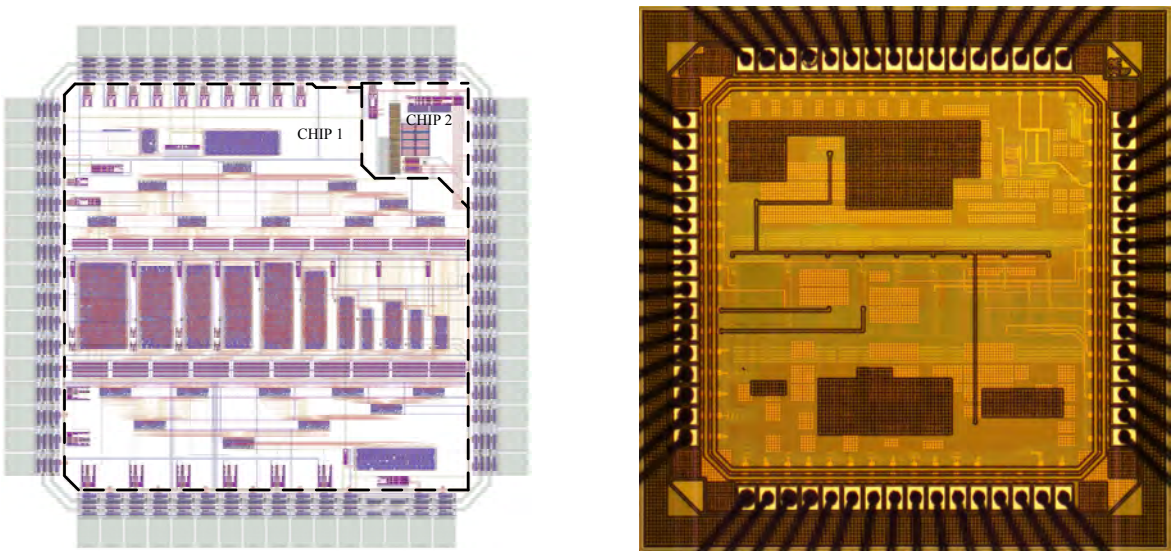
4.5.5 Tape-out

In this subsection, we present the results of further work [109][3] undertaken by our colleagues, Shida Zhong, Ke Li and Isaac Perez-Andrade, who have successfully fabricated an IBM 130 nm ASIC for LDPC decoding using the BMT representation and the proposed node structure of Figure 4.3(a). Figure 4.13(a) shows the layout view of the decoder design, and Figure 4.13(b) is the micro photography corresponding to the fabricated chip. In [109], we have demonstrated that the measurements obtained from the ASIC agree with the conclusions made based on the BER simulations, which validates our design flow proposed in Section 4.2.

4.6 Conclusions

In this chapter, we have proposed a novel design flow for timing-error-tolerant LDPC decoders. Using this design flow, we have accomplished the implementation and verification of an LDPC-FD having improved tolerance to timing errors. We have proposed a novel fully parallel LDPC-FD, having enhanced inherent tolerance to overclocking-induced timing errors. The decoder adopts 4-bit BMT FP numbers rather than TC to

[3]This work is been prepared for submission to IEEE Journal of Solid-State Circuits [109].



(a) Layout (b) Micro photography

Figure 4.13: The layout of the fabricated ASIC for the LDPC decoding using the BMT representation.

represent LLRs, a novel structure for constructing the high-degree nodes, and places D-FFs on the inputs and outputs of only high-degree nodes. These innovations increases the processing throughput by allowing the simultaneous decoding of two independent codewords, as well as protecting the decoder from the catastrophic propagation of timing-error-induced metastability. In order to demonstrate the improved timing error tolerance of the proposed architecture, we have proposed a timing error model for LDPC-FDs, which allows the simulation of timing errors with efficiency and accuracy. With the aid of this model, we investigated the BER performance of the LDPC decoder and several benchmarkers, in the presence of timing errors caused by over-clocking. Based on these BER results, we conclude that the proposed BMT number representation can increase the processing throughput by up to 33.3%, reduce the area by around 10% and maintain a similar level of power consumption, compared to the conventional TC number representation. Furthermore, the novel structure proposed for high-degree nodes and the placement of D-FFs placed at their inputs and outputs was also demonstrated to enhance the tolerance of the decoder to timing errors. Furthermore, we demonstrated that placing D-FFs at the inputs and outputs of high-degree nodes imposes only a negligible degradation in error-correcting performance, but uses 30% fewer D-FFs. An ASIC has been fabricated by our colleagues and the resultant measurements have validated our design flow proposed in Section 4.2.

Chapter 5

Stochastic LDPC Decoder

5.1 Introduction

As discussed in Chapters 3 and 4, Fixed-point LDPC Decoders (LDPC-FDs) usually adopt the Log-Sum-Product Algorithm (Log-SPA), the Min-Sum Algorithm (MSA) or on other log-domain algorithms over the probability-domain Sum-Product Algorithm (SPA). This may be attributed to the high implementational complexity involved with the SPA operations performed on probabilities. The use of reduced-complexity log-domain algorithms allows implementations of LDPC-FDs to use the Fixed-Point (FP) number representation for the soft information exchanged between and processed by the parity-Check Nodes (CNs) and Variable Nodes (VNs). Each FP number comprises several bits, each of which requires a wire to connect between a CN and a VN. This may cause interconnection congestion, due to the large number of wires required to connect the CNs and VNs. This is one of the main factors limiting the fully-parallel implementation of LDPC-FDs having a medium or large block lengths of hundreds or thousands of bits [110, 111, 112, 97, 102]. In particular, the area of wires may occupy up to 50% of the total area of an LDPC decoder Application-Specific Integrated Circuit (ASICs) [110]. Therefore, the partial parallel implementation of LDPC-FDs is more common [100, 99, 98, 113, 101, 14, 16, 10, 13, 26] than fully parallel implementation in practice. In order to reduce the number of wires required to exchange soft information between the CNs and VNs, the so-called bit-serial LDPC-FDs were proposed in [97, 112]. Meanwhile, the bit-flipping decoding algorithm was proposed and developed in [6, 114], while a compression technique was proposed in [115] to compress the probabilities or LLRs.

Despite the above-mentioned challenges, fully parallel implementations of LDPC-FDs have been reported in [27, 116, 117, 118, 28]. In [27, 28], a fully parallel LDPC-FD

for non-binary LDPC codes was fabricated as an ASIC, for the case of a small factor graph. Layout results have been reported in [116, 117, 118] for fully parallel LDPC-FDs, where various techniques for reducing routing congestion are employed.

In [119, 120, 121, 122, 123], the stochastic implementation of LDPC decoders is proposed, which uses so-called Bernoulli sequences to convey the soft information exchanged between CNs and VNs, rather than using FP numbers. In Stochastic LDPC Decoders (LDPC-SDs), only a single bit is exchanged between each pair of CNs and VNs per clock cycle. Over the course of several successive clock cycles, the individual bits that are exchanged between a particular pair of nodes collectively form a Bernoulli sequence [119, 124]. Here, the exchanged probability is represented by the particular fraction of bits in this Bernoulli sequence that have a binary value of 1. In this way, the number of interconnection wires is significantly reduced. Furthermore, the operations performed within the stochastic CNs and VNs may be implemented using simple logic gates, rather than complicated FP arithmetic circuits. These advantages reduce the chip are required, granting LDPC-SDs the potential for fully parallel implementation.

In this chapter, we will discuss the fully parallel implementation of LDPC-SDs in detail. Figure 5.1 lists the most relevant previous publications along the timeline of the stochastic implementation of LDPC decoders. The timeline is represented by a vertical line with the downward direction representing the chronological order, where each knot on the vertical lines represent a publication discussed above. The novel contributions of this chapter area a novel method for obtaining the decoding decision from the LDPC-SDs, as well as a comprehensive characterization of the performance of LDPC-SDs. In this chapter, we commence by explaining basic stochastic computation in LDPC-SDs in Section 5.2.1. Following this, we will discuss the implementation of LDPC-SDs in Sections 5.2.2 and 5.2.3. In Section 5.3, we conduct comprehensive simulations to investigate the Bit Error Ratio (BER) performance of the LDPC-SDs. Finally, this chapter will be concluded in Section 5.4.

5.2 Fully Parallel Implementation of Stochastic LDPC Decoders

Since the LDPC-SDs use Bernoulli sequences to represent probabilities, we first recall the SPA introduced in Chapter 2 and the operations performed on probabilities as expressed in (2.13), (2.14), (2.15) and (2.16). These equations are shown as below

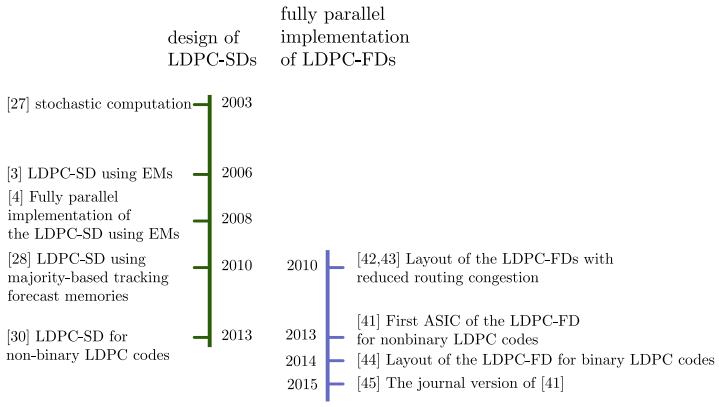


Figure 5.1: Timelines of relevant publications.

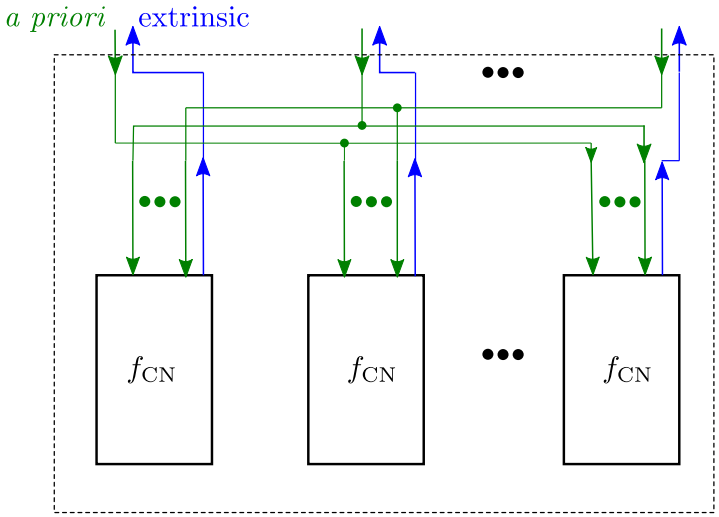


Figure 5.2: Structure of a CN.

for the sake of convenience. In Section 5.2.1, we will discuss the conversion of those operations on probabilities to basic stochastic computations in LDPC-SDs [125].

In the case of CNs having a degree of $d_{\mathrm{c}} = 3$ and VNs having a degree of $d_{\mathrm{v}} = 2$, an output probability P_C that is obtained by combining two input probabilities P_A and P_B is given by

$$P_C = f_{\mathrm{CN}}(P_A, P_B) = P_A(1 - P_B) + P_B(1 - P_A),$$

$$P_C = f_{\mathrm{VN}}(P_A, P_B) = \frac{P_A P_B}{P_A P_B + (1 - P_A)(1 - P_B)},$$

respectively [8]. These functions may be extended for more than two input probabilities, by recursively substituting (2.13) and (2.14) into itself. For example, these input probabilities may be combined according to

$$\begin{aligned} P_D &= f_{\mathrm{CN}}(P_A, P_B, P_C) \\ &= P_A(1 - f_{\mathrm{CN}}(P_B, P_C)) + f_{\mathrm{CN}}(P_B, P_C) \cdot (1 - P_A), \end{aligned}$$

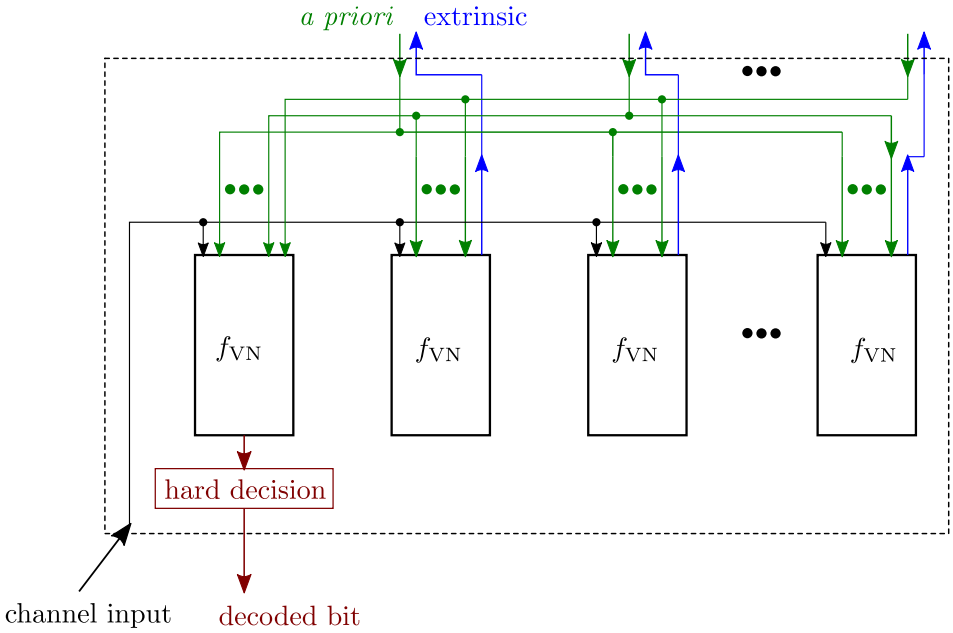


Figure 5.3: Structure of a VN.

$$\begin{aligned} P_D &= f_{\mathrm{VN}}(P_A, P_B, P_C) \\ &= \frac{P_A \cdot f_{\mathrm{VN}}(P_B, P_C)}{P_A \cdot f_{\mathrm{VN}}(P_B, P_C) + (1-P_A)(1-f_{\mathrm{VN}}(P_B, P_C))} \\ &= \frac{P_A P_B P_C}{P_A P_B P_C + (1-P_A)(1-P_B)(1-P_C)}. \end{aligned}$$

The structure of CNs and VNs in fully-parallel LDPC-SDs were discussed in Chapter 4, illustrated by Figures 5.2 and 5.3. In Sections 5.2.2 and 5.2.3, we will discuss the implementation of CNs and VNs in LDPC-SDs, respectively, based on the basic stochastic computation and the structure in LDPC-SDs. In particular, Section 5.2.3 will detail the differences of each component in stochastic VNs compared to those in LDPC-FDs.

5.2.1 Basic Stochastic Computation

In LDPC-SDs, the exchanged probability is represented by the particular fraction of bits in this Bernoulli sequence that have a binary value of 1. For example, the probability of 0.7 may be expressed by a stochastic bit stream 1011110101 .... However, the same probability of 0.7 may be represented by other streams having the same fraction of 1s in a different order, such as 0110110111 .... Accordingly, the stochastic implementation of (2.13) and (2.14), as well as of the CNs and VNs of Figure 5.2 and 5.3, are explained in the following subsections.

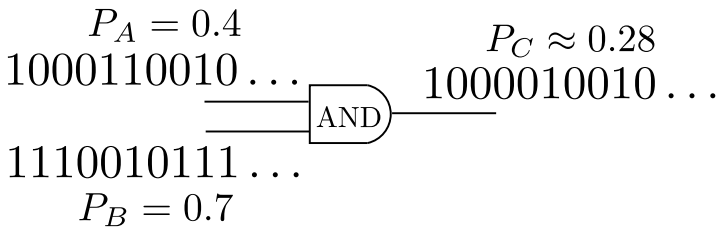
In stochastic computation, different mathematical functions of probabilities can be

implemented using different logic gates. In particular, the intersection of two probabilities is given by their product $P_C = P_{A\cap B} = P_A \cdot P_B$, which can be implemented by using an AND logic gate, to combine the corresponding Bernoulli sequences, as exemplified in Figure 5.4(a). Similarly, the difference between the union and intersection of two independent probabilities is given by $P_C = P_{A\oplus B} = P_{A\cup B} - P_{A\cap B} = P_A(1-P_B)+P_B(1-P_A)$, which can be implemented by using an eXclusive-OR (XOR) gate, as exemplified in Figure 5.4(b). The complementary probability $P_C = P_{\bar{A}} = 1 - P_A$ can be obtained using a NOT gate to invert the Bernoulli sequence, as exemplified in Figure 5.4(c). Finally, the normalized division $P_C = \frac{P_A}{P_A+P_B}$ of two probabilities can be implemented using a JK-type Flip-Flop (JK-FF), which operates on the two corresponding Bernoulli sequences, as shown in Figure 5.4(d). Note that while the gates of Figures 5.4(a), 5.4(b) and 5.4(c) produce outputs that depend only on the current inputs, the JK-FF of Figure 5.4(d) has memory, which implies that the output depends on the current inputs and the state of the memory that has been established by the previous inputs. More explicitly, the output Q of the JK-FF is updated to the value of the input J, if it disagrees with the value of K. If J and K are both equal to 0, then the state of the output Q is preserved. If J and K are both equal to 1, then the state of the output Q is toggled, as shown in the example of Bernoulli sequences of Figure 5.4(d). Note that when the Bernoulli sequences are short, the output will represent a probability that only approximates the correct result, as exemplified in Figure 5.4. However, the approximations become increasingly accurate, as the length of the Bernoulli sequences are extended.

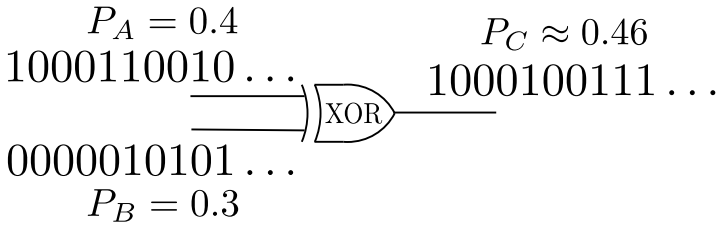
5.2.2 Stochastic Implementation of CNs

Figures 5.2 and 5.3 depict the structures of an individual CN and VN, respectively. As shown in Figure 5.2, the c^{th} CN uses d_c number of computation units to combine the d_c input probabilities, which are referred to as *a priori* probabilities. More specifically, f_{CN} is used to compute the so-called *extrinsic* probabilities. As shown in Figure 5.3, the v^{th} VN is provided with a probability from the channel, as well as d_v *a priori* probabilities from the connected CNs. The VN uses the d_v computation units shown in Figure 5.3 to compute f_{VN} and output an *extrinsic* probability back to each of the connected CNs via the corresponding edge. Furthermore, the VN uses an additional computation unit to calculate a probability for the channel port, which is then converted into a decoded bit having the binary value of 1, if the probability is greater than 0.5, or to the binary value of 0 otherwise.

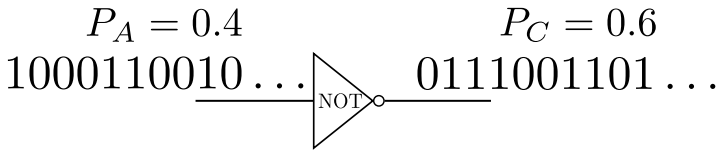
CNs in LDPC-SDs can be implemented by substituting the computation units in Figure 5.2 by the stochastic implementation of the function f_{CN}. More specifically, in



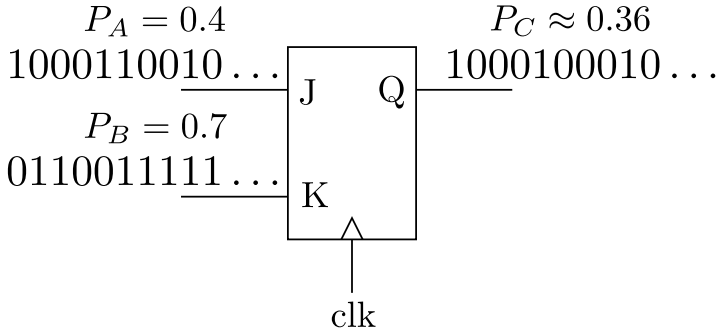
(a) Stochastic implementation for approximating $P_C = P_{A \cap B} = P_A \cdot P_B$.



(b) Stochastic implementation for approximating $P_C = P_{A \oplus B} = P_A(1 - P_B) + P_B(1 - P_A)$.



(c) Stochastic implementation for approximating $P_C = P_{\bar{A}} = 1 - P_A$.



(d) Stochastic implementation for approximating $P_C = \frac{P_A}{P_A + P_B}$.

Figure 5.4: Stochastic implementations of the computation used in LDPC decoding, as well as example Bernoulli sequences and the corresponding output Bernoulli sequences. Note that the probabilities P_C are the correct results.

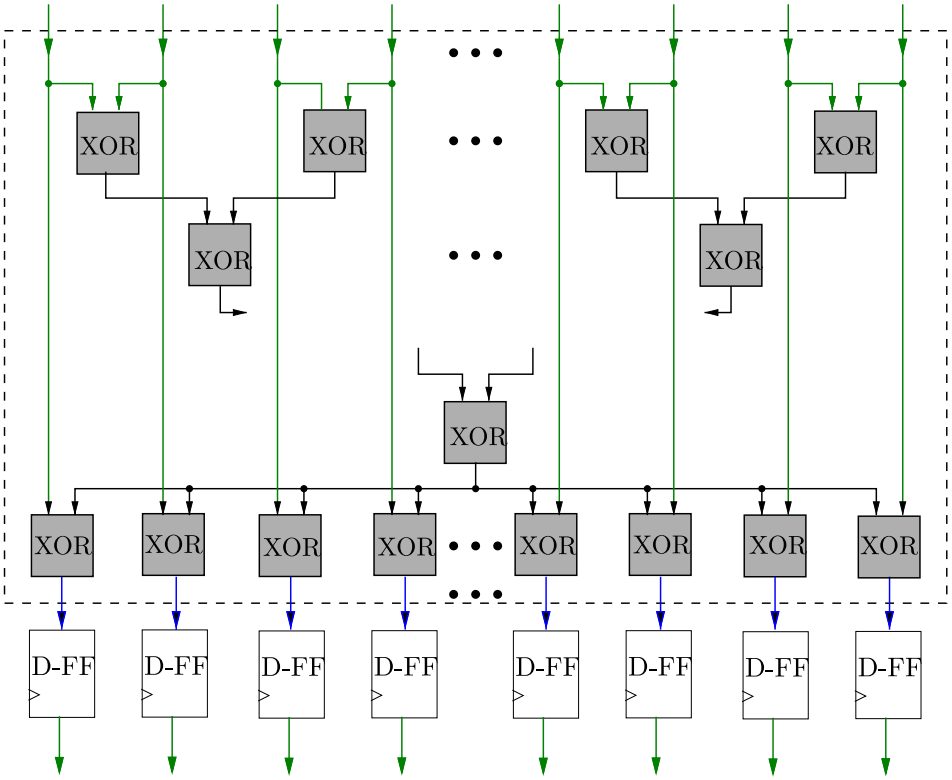


Figure 5.5: Structure of a stochastic CN.

CNs having a degree of $d_c = 3$, the stochastic implementation of (2.13) is obtained using an XOR gate, as shown in Figure 5.4(b). Note that in LDPC-FDs, the corresponding function f_{CN} is implemented as either a sum of probabilities represented by FP numbers in the Log-SPA, or a min of LLRs represented by FP numbers in the MSA. Furthermore, the stochastic implementation of f_{CN} having more than two inputs in CNs having a degree of $d_c > 3$ can be implemented by recursively combining XOR gates, in analogy with (2.15). In analogy with Figure 5.2, a stochastic CN having a degree of d_c perform parity check operations on all the input bits provided by the connected VNs, by using d_c stochastic computation units, each of which combines its $d_c - 1$ inputs using a network of $d_c - 2$ XOR gates. However, as discussed in Chapter (4), the hardware complexity of this structure can be readily reduced by using the arrangement of Figure 5.5 [64]. In this high-degree CN, an intermediate parity check result is obtained using a tree structure composing $d_c - 1$ XOR gates, in order to find the XOR of *all* the input *a priori* bits. This parity check result is then XORed with each of the d_c input bits, in order to obtain the extrinsic bit for each output port. This improved structure therefore requires a total of $2d_c - 1$ XOR gates, which is lower than the $d_c(d_c - 2)$ XOR gates required by the structure of Figure 5.2 for $d_c > 3$. Note that D-type Flip-Flop (D-FF) are employed at every input port of CNs, for storing the *a priori* bits sent from the interconnected VNs in the previous clock cycle. Likewise, at the output ports of CNs,

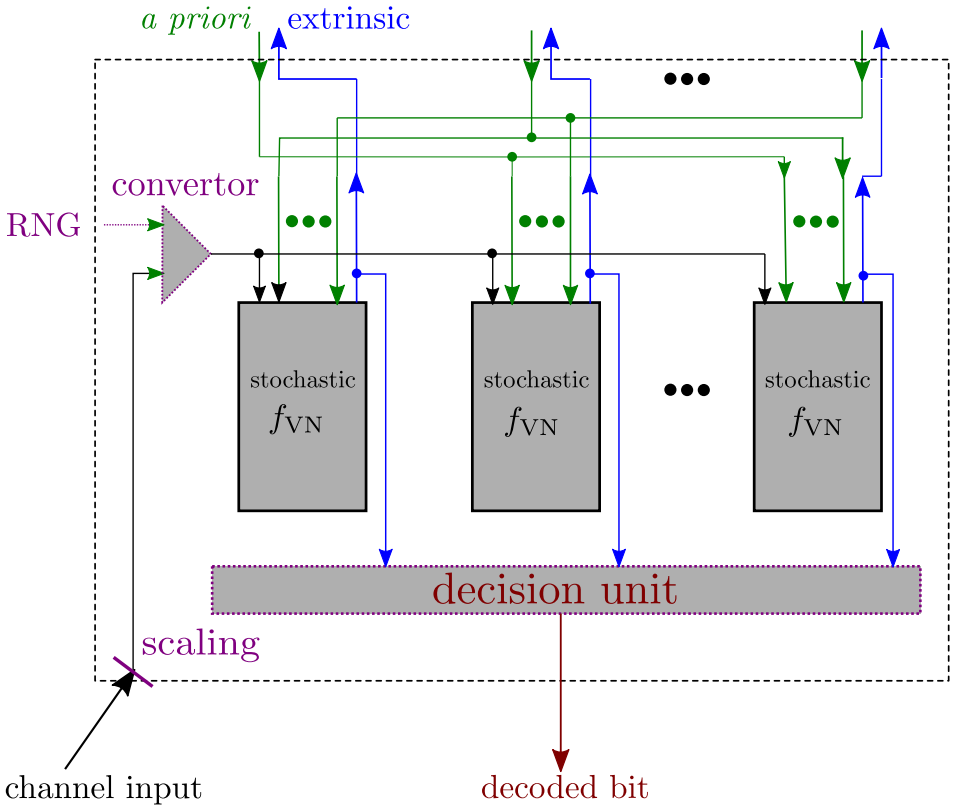


Figure 5.6: Structure of a stochastic VN.

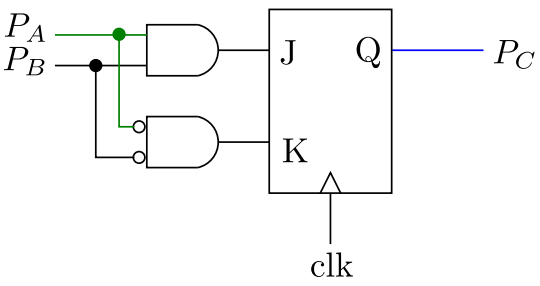
the D-FFs are employed to store the extrinsic bits in the current clock cycle. In this way, two consecutive clock cycles form one decoding iteration in LDPC-SDs.

5.2.3 Stochastic Implementation of VNs

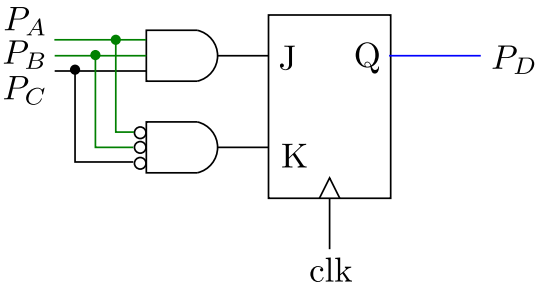
Figure 5.6 illustrates the stochastic implementation of the VN of Figure 5.3, where the shaded blocks represent the constituent components, namely the convertor used for providing the computation units with stochastic bits that represent the channel input, the computation units used for the stochastic implementation of the function f_{VN} for converting the *a priori* stochastic bits into and the decision unit used for obtaining the decoded bit. In the following discussions, we will address the implementation of the each shaded blocks, addressing the differences with the corresponding components of VNs in conventional LDPC-FDs.

5.2.3.1 Channel Input Convertor

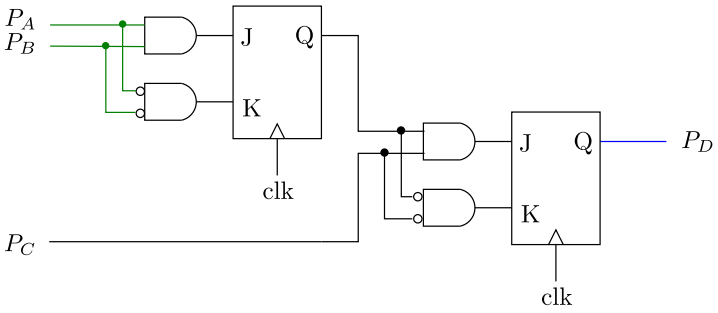
As shown in Figure 5.6, the input from channel must be converted from a bit probability to a Bernoulli sequence, which supplies a single bit to each computation block, in each



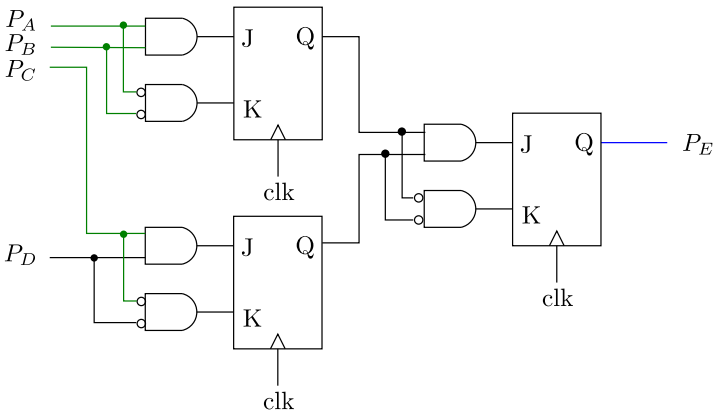
(a) Stochastic implementation of f_{VN} of (2.14) for VNs having a degree of $d_{\mathrm{v}} = 2$.



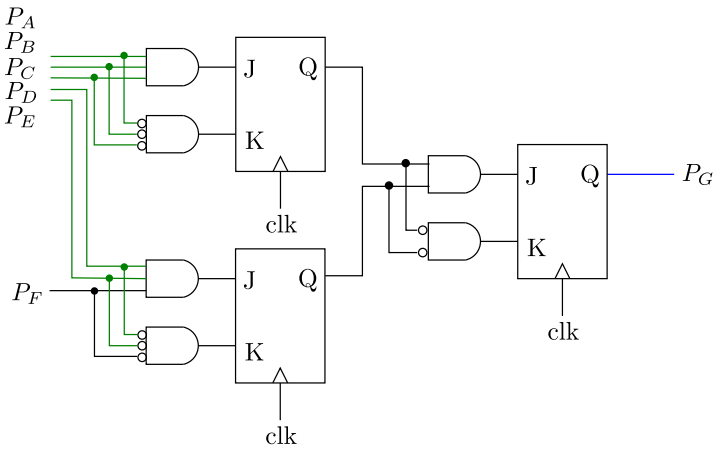
(b) Stochastic implementation of f_{VN} of (2.16) for VNs having a degree of $d_{\mathrm{v}} = 3$.



(c) An anti-latching stochastic implementation of f_{VN} of (2.16) for VNs having a degree of $d_{\mathrm{v}} = 3$.



(d) Anti-latching stochastic implementation of f_{VN} for VNs having a degree of $d_{\mathrm{v}} = 4$.



(e) Anti-latching stochastic implementation of f_{VN} for

clock cycle. This convertor can be implemented as a comparator, which outputs a stochastic bit having the value of 1, if the channel input probability exceeds a random number generated with a uniform probability distribution or outputs 0 otherwise [121]. This output is clocked into an output D-FF.

5.2.3.2 Computational Units

The implementation of the function f_{VN} can be constructed based on the stochastic arithmetic of Section 5.2.1. More specifically, (2.14) can be implemented by a JK-FF, where the J and K inputs are provided by Bernoulli sequences that represent the calculation of $P_A \cdot P_B$ and $(1 - P_A) \cdot (1 - P_B)$, respectively. As discussed previously, $P_A \cdot P_B$ can be obtained using the AND gate of Figure 5.4(a), while the second term $(1 - P_A) \cdot (1 - P_B)$ can be obtained by using an AND gate combined with two NOT gates. The complete implementation of (2.14) is shown in Figure 5.7(a), while Table 5.1 provides the corresponding truth table. Note that if the two stochastic input bits P_A and P_B have values that agree, then this value is passed to the output stochastic bit P_C, as a so-called regenerative bit. Otherwise, the value output in the previous clock cycle is preserved for the output P_C, as a so-called conservative bit. For VNs having higher degrees, the additional *a priori* stochastic bits that are input to each computation unit of Figure 5.6 can be accommodated by increasing the number of inputs provided to the AND gates. This is shown in Figure 5.7(b) for the case of VNs having a degree of $d_{\mathrm{v}} = 3$, which operate on the basis of (2.16). However, as the number of inputs to an AND gate is increased, the probability that they will all adopt the value of 1 simultaneously diminishes. As a result, the outputs of the AND gates tend to get stuck at 0, causing the output of the connected JK-FF to also become stuck.

Table 5.1: Truth table of stochastic VN.

P_A	P_B	P_C	
0	0	0	generative bits
1	1	1	
1	0	previous P_C	conservative bits
0	1	previous P_C	

This problem is exacerbated by cycles in the factor graphs of LDPC codes, which comprise looping paths along the edges between connected CNs and VNs [92, 71], as exemplified in Figure 2.3. In LDPC-SDs, these cycles can result in positive feedback loops, which can cause particular stochastic bits to get stuck at particular values over many

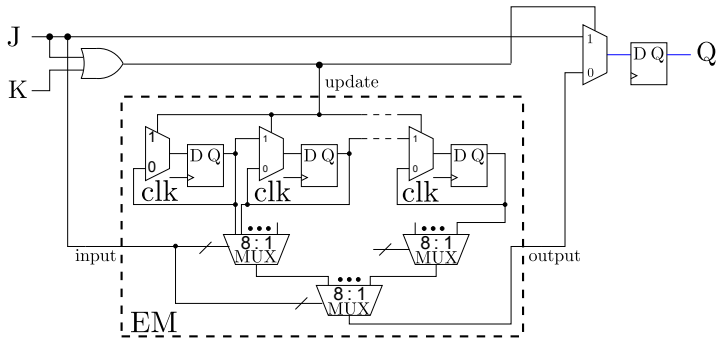


Figure 5.8: The example structure of the EM, replacing the JK-FF indicated by the dashed block.

successive clock cycles. This so-called latching problem typically prevents a LDPC-SD from converging to a valid decoding result, unless special measures are taken to mitigate the locked state of the nodes, as discussed below.

5.2.3.3 Anti-Latching Techniques

In particular, VNs having large degrees are most affected by the latching problem, since they typically form part of more cycles in the factor graph and because they pass their locked *extrinsic* bits to more CNs. Therefore, the latching problem can be addressed by redesigning the VNs having large degrees, at the cost of making them slightly more complex. In order to increase the switching activity and therefore mitigate the latching problem for stochastic VNs having high degrees, the implementations of Figures 5.7(a) and 5.7(b) can be combined recursively, as exemplified in Figure 5.7(d) for the case of a VN having a degree of $d_{\mathrm{v}} = 4$ and Figure 5.7(e) for the case of a VN having a degree of $d_{\mathrm{v}} = 6$. In order to further increase the protection of VNs having a degree of $d_{\mathrm{v}} = 3$ from the latching problem, Figure 5.7(b) can be alternatively implemented using the layered structure shown in Figure 5.7(c). This approach may be extended for VNs having higher degrees, as exemplified in Figures 5.7(d) and 5.7(e), for degrees of $d_{\mathrm{v}} = 4$ and $d_{\mathrm{v}} = 6$, respectively.

As a further step, [121, 120] proposed the replacement of the JK-FFs employed by VNs with circuits based on so-called Edge Memories (EMs), which comprise a shift register formed of several D-FFs, as shown in Figure 5.8. The circuit of Figure 5.8 behaves in analogy to a JK-FF and may therefore replace the JK-FFs of Figures 5.7. Note that the use of the AND gates in Figures 5.7 ensure that the inputs J and K of Figure 5.8 cannot both have the value of 1 simultaneously. Therefore, the circuit of Figure 5.8 does not need to implement the toggle functionality of a JK-FF. This means that a disagreement between the values of J and K can detected using the OR gate of

Figure 5.8. In this case, the MUltipleXer (MUX) of Figure 5.8 passes the value of the input J to the output D-FF, as in a JK-FF. In this case, the update signal is used to shift the contents of the EM by one position and to shift in the value of the input J. By contrast, if the inputs J and K bothy adopt the value of 0, then a random bit is selected from the contents of the EM and passed to the output D-FF. In this way, a bit value from a previous clock cycle is output, in analogy to the behavior of a JK-FF, when preserving the value from the previous clock cycles. Since the EM typically stores a mixture of both 0s and 1s, the circuit of Figure 5.8 allows a locked state to be escaped, in contrast to a JK-FF-based VN.

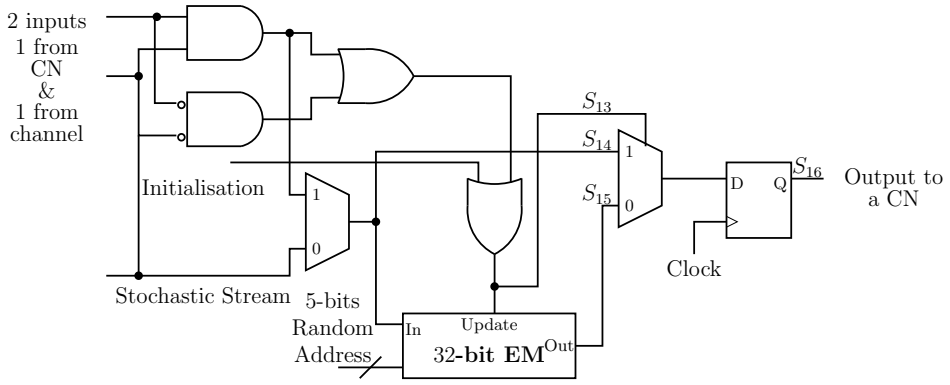


Figure 5.9: Schematic of a stochastic VN having a degree of $d_v = 2$ [121].

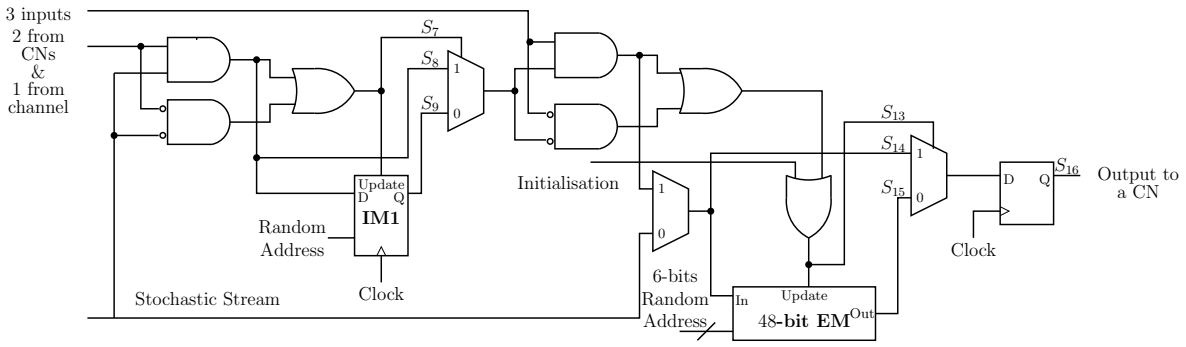


Figure 5.10: Schematic of a stochastic VN having a degree of $d_v = 3$ [121].

In VNs having degree of 2,3,4 and 6, [121, 120] recommended the use of EMs comprising 32, 48, 48 and 64 D-FFs, respectively. However, in high-degree VNs adopting the layered structures of Figure 5.7(e), the JK-FFs in the first layer may be replaced with so-called Intermediate Memories (IMs), rather than full EMs. These IMs have a simpler structure as EMs, but only store 1 or 2 bits. More specifically, [121, 120]

recommended the use of a pair of IMs comprising one or two D-FFs for VNs having degrees of 4 or 6, respectively. For VNs having a degree of 3, a single IM comprising one D-FF was recommended in [121, 120] for combining two the three inputs to each computation unit. Meanwhile, IMs are not required for VNs having a degree of 2. Only small IMs are needed, since the outputs of these IMs feed into a large EM in the second layer, which can generate the VN's final output bit and mitigate the latching problem. Furthermore, when employing IMs, the output D-FF of Figure 5.8 may be omitted, in order to reduce the number of clock cycles required for stochastic bits to propagate through the VN, as shown in Figure 5.11. Throughout the remainder of this paper, we assume the use of IMs and EMs in all LDPC-SDs discussions, unless specified otherwise.

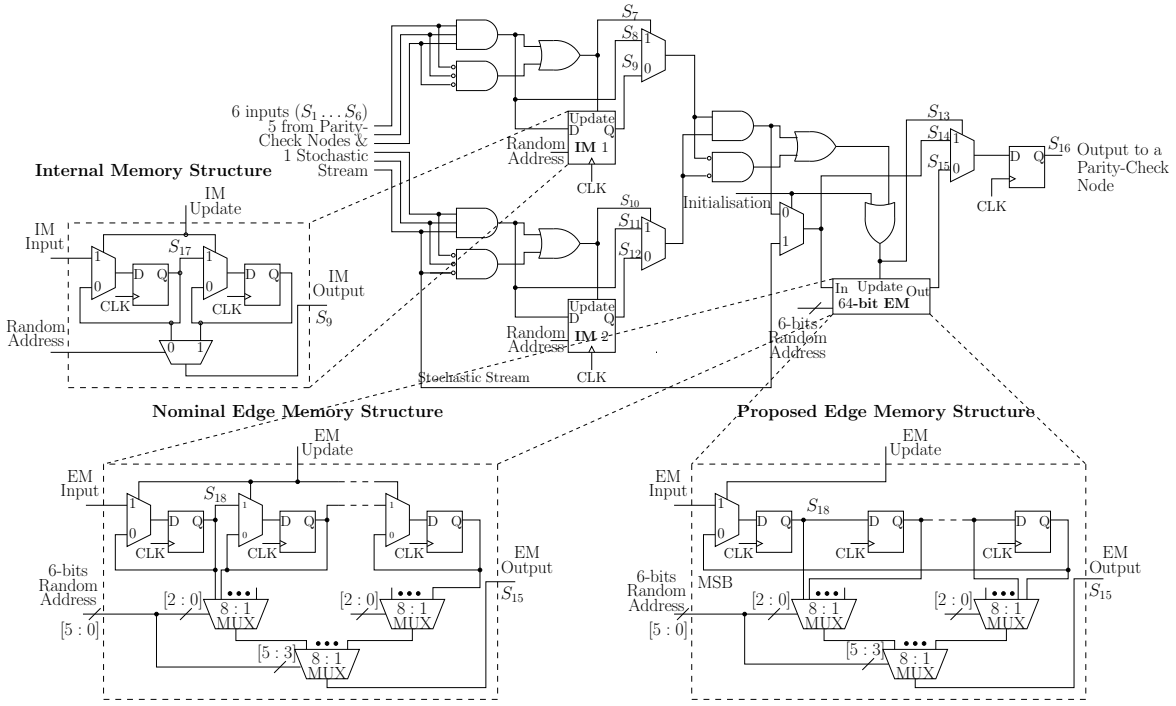


Figure 5.11: Schematic of a stochastic $d_v = 6$ VN. Corresponding schematics for $d_v = 2$ and 3 VNs can be found in [121, Figure 6].

As discussed previously, the content of EMs is essential for VNs to escape the latching state. Furthermore, EMs are heavily relied on during the beginning of the iterative decoding, since the inputs of VNs are more likely to disagree with each other. Therefore, the initial values stored in EMs before starting the decoding processing play an important role in the decoding of LDPC-SDs [121]. A solution is proposed in [121] to initialize the EM with the input probability from the channel. More explicitly, when a new codeword is received at the decoder, the initialization signal shown in Figure 5.11 is set as 1, during as many clock cycles as half of the largest EM, so that all the EMs are filled by the bits coming out of the convertors of Figure 5.6, e.g. 32 clock cycles for

the (1056,528) WiMAX LDPC code. before the iterative exchanging between CNs and VNs begins. After the initialization, with the initialization signal reset as 0, the CNs and VNs are iteratively activated, and the EM is then controlled by the update signal discussed previously, until the decoding processing is terminated. This initialization is then repeated for another received codeword. It has been demonstrated that this partial initialization allows the LDPC-SDs have the similar BER performance as the LDPC-FDs [121]. In this paper, we also use the recommended 32 clock cycles in the initialization phase.

So-called Noise-Dependent Scaling (NDS) [120] may also be used to help prevent the occurrence of the latching problem. This technique scales the bit probabilities provided by the demodulator before they are converted into stochastic bits and provided to the VNs. In this way, the degree of the switching activity within the stochastic decoder can be maintained at a sufficiently high level, that mitigates the latching problem for all SNRs. In the case where BPSK modulation is employed for transmission over an AWGN channel, the scaled bit probability is obtained as

$$P = \frac{1}{1 + \exp(\frac{\alpha N_0}{y_{max}} \cdot \frac{4\eta E_b}{N_0} \cdot \mathrm{Re}(y))}, \tag{5.1}$$

rather than (2.12), where y is the received Binary Phase Shift Keying (BPSK) symbol, N_0 is the Additive White Gaussian Noise (AWGN) power spectral density and $\frac{\alpha N_0}{y_{max}}$ is the noise-dependent scaling factor. Here, y_{max} is a predefined constant relating to the maximum value of $\mathrm{Re}(y)$ and α is a constant parameter of the scaling. By tuning the value of α, the BER performance of LDPC-SDs may be optimized [120]. According to [120], y_{max} has the value of 6 for BPSK transmission, where the optimal value of α is around 3.

5.2.3.4 Decision Unit and Termination of Decoding

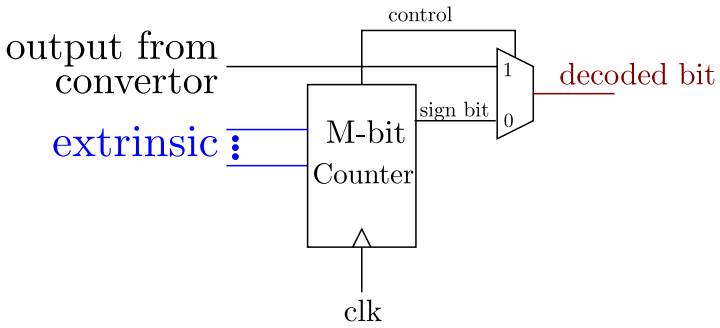


Figure 5.12: The structure of the voting decision unit, which may be used to generate the decoded bit of a VN of an LDPC-SD.

The method used by the decision unit of Figure 5.6 to generate a decision bit in each VN has a significant impact on the error correction performance of an LDPC-SD. However, neither the generation of the decision bits or their impact on the error correction performance has been addressed in previous publications on LDPC-SDs. Therefore, we propose two methods for generating these decision bits as shown in Figures 5.12 and 5.13, both of which offer a strong error-correction performance. The schematic of Figure 5.12 implements a voting mechanism based on all the extrinsic bits output from the computation units in a VN, where the majority of all the extrinsic bits decide the decoded bit. Explicitly, the counter of Figure 5.12 is reset in each clock cycle, and is then incremented by 1 for each extrinsic bit provided by a computation unit having the value of 0, or decremented by -1 otherwise. At the end of the clock cycle, if the content of the counter is non-zero, the control signal is set as 0, which allows the sign bit of the content of the counter to propagate through the MUX as the decoded bit. However, if the content of counter is evaluated as zero, namely half of the extrinsic bits are 1s and the other half are 0s, the control signal is set as 1, which sends the bit output from the convertor directly through the MUX as the decoded bit. The number of bits M used in the counter may be set as 3, which we found is sufficient for the WiMAX (1056,528) LDPC code, since the largest degree of its VNs is $d_{\mathrm{v}} = 6$. This decision unit may be referred to as the voting scheme in the following discussions.

As an alternative, we propose the use of a JK-FF, to obtain the decision bits. As illustrated in Figure 5.13, this decision unit is similar to the computation unit of Figure 5.7(b), but is obtained by replacing all *a priori* inputs, as well as the input from the channel, with the values of extrinsic outputs from the computation units of Figure 5.6. If all the extrinsic bits agree with each other, then this value is passed to the output as the decoded bit. Otherwise, the value output in the previous clock cycle is preserved for the decision bit. We refer to this decision unit as the JK-FF scheme.

The iterative stochastic LDPC decoding process can be terminated once a valid set of decoded bits is found, or when a predefined maximum number of decoding cycles is reached. The former termination condition relies on the method to validate the decoding bits, as well as a mechanism for generating a decision bit for each VN during each decoding cycle. In particular, the decoded bits may be considered to be valid, if their multiplication with the Parity-Check Matrix (PCM) produces an all-zero syndrome [121].

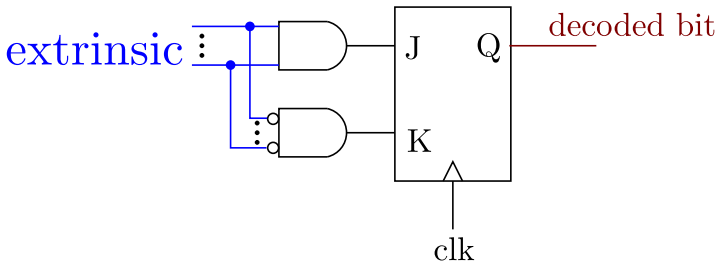


Figure 5.13: The structure of the JK-FF decision unit, which may be used to generate the decoded bit in a VN of an LDPC-SD.

Table 5.2: Simulation parameters.

Initialization method	random, all one, all zero and channel probability
Decision unit scheme	voting and JK-FF
Scaling method	with and without NDS
LDPC code	WiMAX $(1056, 528)$ LDPC [65]
EM length	$32, 48, 64$ bits for VNs having degree of $d_v = 2, 3, 6$
EM length multiplexer	$1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}$
IM length	$0, 1, 2$ bits for VNs having degree of $d_v = 2, 3, 6$
Max number of decoding cycles	up to 10000
Clock cycles per decoding cycle	1

5.3 Simulation Results and Discussion

In this section, we conduct simulations to investigate the BER performance of LDPC-SDs employing the implementation discussed in Section 5.2. Explicitly, the BER performance of the WiMAX (1056,528) LDPC code is characterised for the case of employing BPSK for transmission over an AWGN channel. In each of the following subsections, the BER performance will be characterized as a function of a different parameter, including the choice of decision unit scheme, the initialization method of EMs, the length of EMs and the number of decoding cycles. In this way, we identify recommended parameters the scheme for the full-parallel implementation of the LDPC-SD. Our simulation parameters are summarized in Table 5.2. The benchmarker is provided by the optimal floating-point LDPC decoder using the Log-SPA, which continues decoding until convergence is achieved.

5.3.1 Decision Unit Scheme

As discussed in Section 5.2.3.4, two schemes for generating the decoded bit in a VN have been proposed in this chapter, as shown in Figures 5.12 and 5.13. Their BER performance will be compared in this subsection. Apart from the decision unit, the other

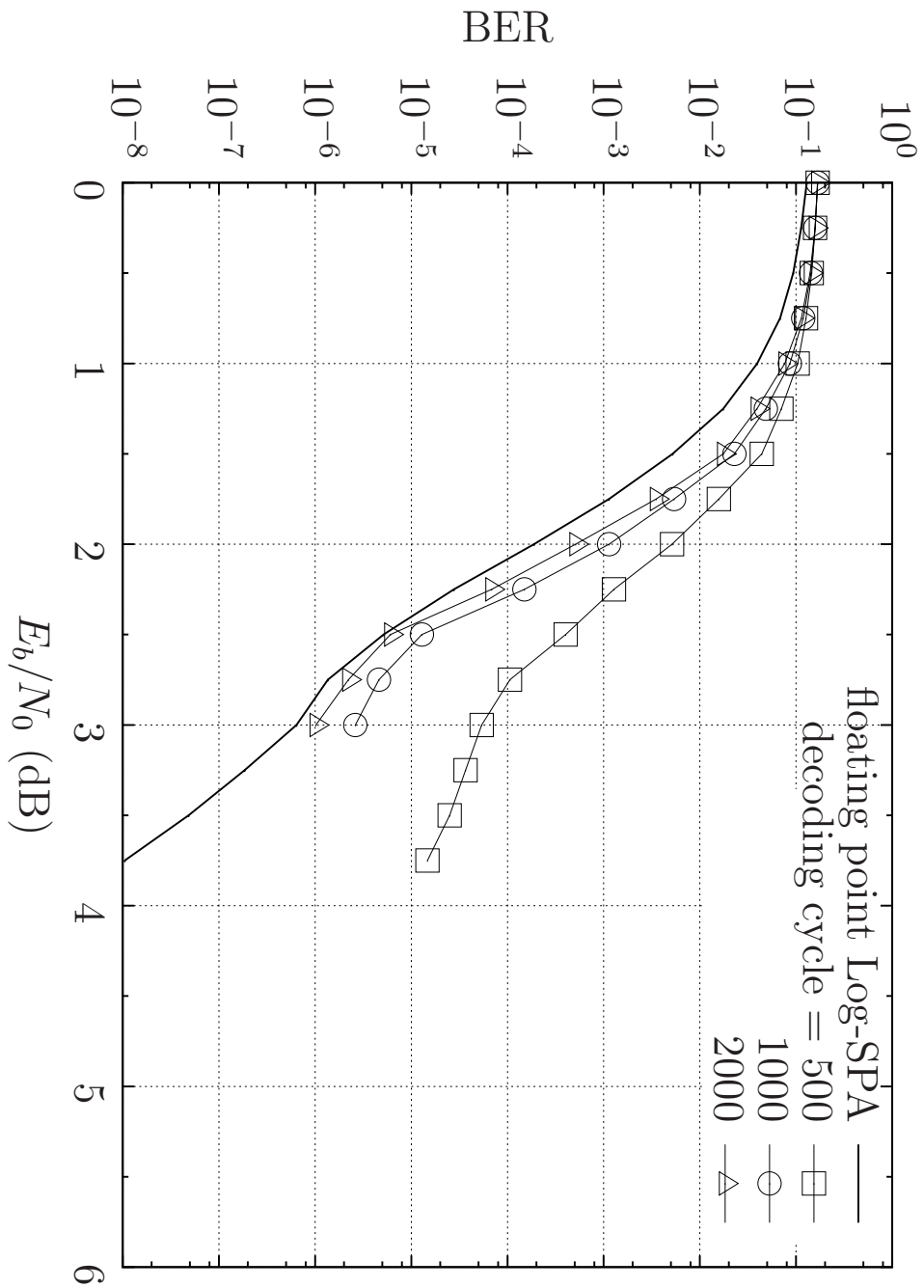
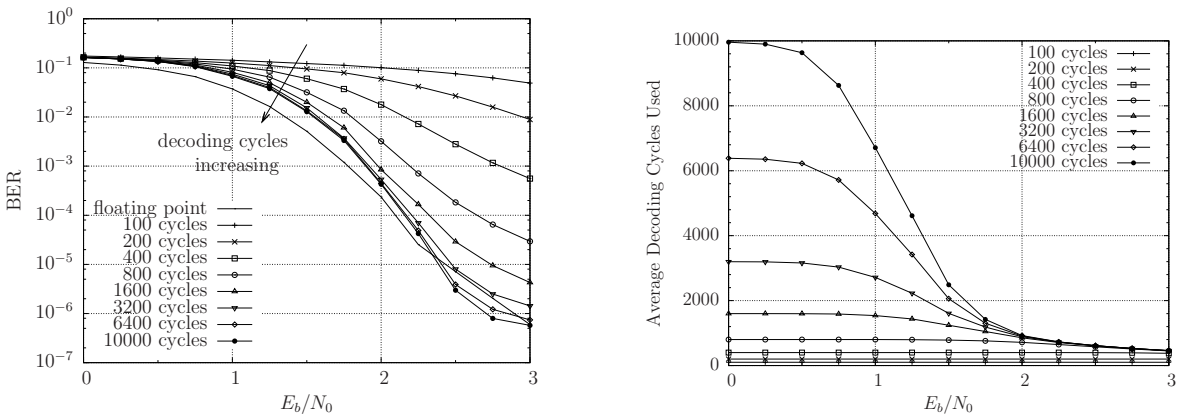


Figure 5.14: BER performance of the LDPC-SD using the voting scheme of Figure 5.12 for decision bit generation.

parameters are identical for the LDPC-SDs employing the two schemes. Explicitly, the length of the EMs are set as 32, 48 and 64 D-FFs for VNs having degrees of d_c = 2, 3 and 6, respectively. During the initialization step, the sequence of output bits from the convertor discussed in Section 5.2.3.1 are used to fill the the EMs in the corresponding VN, for 32 successive clock cycles. The plot of Figure 5.14 compares the BER performance of the voting scheme of Figures 5.12, when different number of decoding cycles are employed. As may be observed in Figure 5.14, the LDPC-SD

using the voting scheme of Figures 5.12 may achieve the same BER performance as the Log-SPA benchmarker, when the maximum number of decoding cycles is set as 2000, although it exhibits a gap of around 0.5 dB at a BER of 10^{-5}, when fewer decoding cycles are permitted. By contrast, the BER performance of the JK-FF decision scheme of Figure 5.13 is shown in Figure 5.15(a). As may be observed in Figure 5.15(a), the LDPC-SD employing the JK-FF scheme of Figure 5.13 also achieve the same BER performance as the Log-SPA benchmarker, when the maximum number of decoding cycles is 2000.

Note however that the voting scheme of Figures 5.12 has a higher implementation complexity than the JK-FF scheme of Figure 5.13, owing to the complex M-bit counter. Therefore, in the remainder of this thesis, the JK-FF scheme of Figure 5.13 is adopted for the implementation of the LDPC-SD, unless specified otherwise.



(a) BER

(b) Average number of decoding cycles performed.

Figure 5.15: (a)BER performance of the LDPC-SD and (b) the average number of decoding cycles performed, when the maximum numbers of decoding cycles allowed is $\{100, 200, 400, 800, 1600, 3200, 6400, 10000\}$.

5.3.2 Initialization of EMs

As discussed in Section 5.2.3.3, the EMs are initialized by the sequence of bits output from the convertor at the start of decoding process, which may be referred to as "channel initialization". The length of the EMs are set as 32, 48 and 64 D-FFs for VNs having degrees of $d_c = 2$, 3 and 6, respectively. In this way, the EMs can be completely initialized, provided that 64 successive clock cycles are set as the initialization depth. Alternatively, the content of the EMs may be only partially initialized, if a

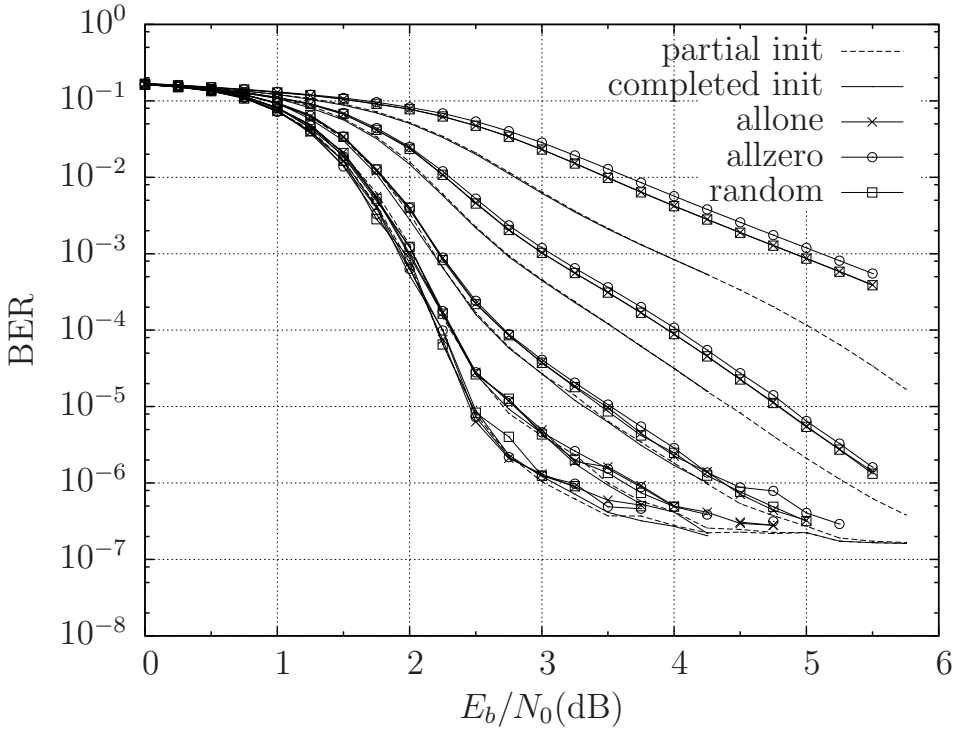


Figure 5.16: BER performance of the LDPC-SD employing different EM initialization methods and depths, for the case of performing a maximum number of decoding cycles in the set $\{200, 400, 800, 1600, 2000\}$

lower number of initialization clock cycles is chosen. In this subsection, we will compare the BER performance of the LDPC-SD employing the channel initialization method to three other methods, as shown in Figure 5.16. Explicitly, the other methods initialize the EMs using sequences of 1s, 0s and random binary values, respectively. Furthermore, the BER performance of the channel initialization methods with full depth of 64 clock cycles is also compared to that with the partial depth of 32 clock cycles in Figure 5.16. These 5 BER curves are presented in clusters, corresponding to a particular number of decoding cycles, which increases in the set of 200, 400, 800, 1600 and 2000 as the BER curves approaching the bottom left corner of Figure 5.16. As observed in Figure 5.16, the channel initialization method exhibits a faster convergence than the other three methods, although all methods achieve the same BER performance as the Log-SPA benchmarker, when the maximum number of the decoding cycles is 2000. The partial channel initialization method also exhibits an identical BER performance to the full channel initialization method in every cluster. Therefore, we recommend the channel initialization method with the depth of 32 clock cycles, in order to achieve trade-off between the error correcting performance and the implementation complexity. Additionally, this choice allows comparison between our work with previous publications, since the initialization depth of 32 clock cycles is also recommended in [121, 127].

5.3.3 Length of EMs

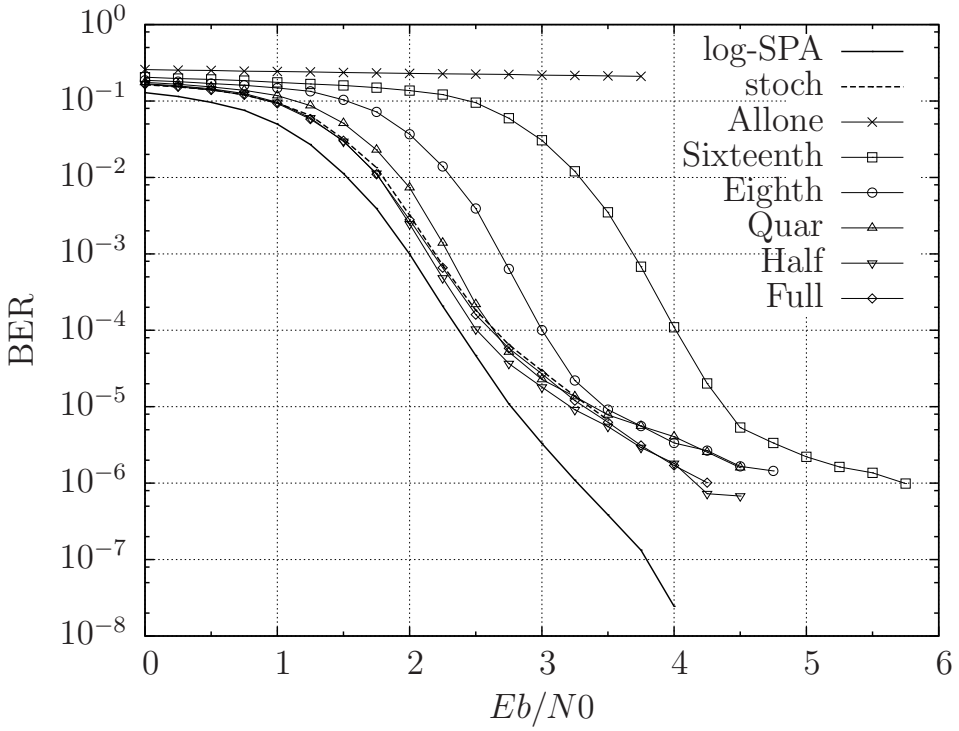


Figure 5.17: BER performance of the LDPC-SD employing EMs of various lengths.

In this subsection, we conduct simulations to investigate the effect of the length of EMs on the BER performance of the LDPC-SD, as shown in Figure 5.17. As discussed in Section 5.2.3.3, VNs having degrees of 2,3 and 6 employ EMs comprising 32, 48 and 64 D-FFs, respectively. We may refer to this configuration as the full-length EMs. Another four configuration are also considered in this subsection, which reduce the length of EMs by factors of $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ and $\frac{1}{16}$, namely resulting in EM lengths of $\{16, 24, 32\}$, $\{8, 12, 16\}$, $\{4, 6, 8\}$ and $\{2, 3, 4\}$ D-FFs, respectively, for VNs having degrees of $\{2, 3, 6\}$. We refer to these as half, quarter, eighth and sixteenth length EM schemes, respectively. Furthermore, an 'allone' configuration in which all EMs in VNs have only a single D-FF is considered as the extreme case. As observed in Figure 5.17, the extreme configuration always fails to converge to a low BER, while the two configurations having the smallest EM lengths exhibit obviously slower convergence than those having larger EM lengths. In particular, only the two configurations having the largest EM lengths, namely $\{32, 48, 64\}$ and $\{16, 24, 32\}$, can achieve the closest performance to the floating point benchmarker, although there is no distinctive difference of BER performance between them. Despite this, however, we adopt the use of the full length EM configuration for the remainder of this thesis, so that our future work may be compared to the results provided in previous publications [121, 127], which also used

full length EMs.

5.3.4 Number of Decoding Cycles

Following the conclusions of the previous subsections, Figure 5.15(a) comprehensively characterize the BER performance of the LDPC-SD when the maximum number of decoding cycles is a value in the set of $\{100, 200, 400, 800, 1600, 3200, 6400, 10000\}$. Accordingly, the average number of decoding cycles performed before early stopping is involved is characterized as a function of the channel E_b/N_0 values, in Figure 5.15(b). As shown in Figure 5.15(a), the BER performance of the LDPC-SD approaches that of the Log-SPA benchmarker when the number of decoding cycles is 1600, exhibiting a gap of around 0.25 dB. When the maximum number of decoding cycles is equal to or higher than 3200, the BER performance becomes similar to the benchmarker, and cannot be further improved significantly by using more decoding cycles. However, the average number of decoding cycles performed is almost doubled, when closing the gap of 0.25 dB, as observed in Figure 5.15(b). Therefore, a maximum number of 2000 decoding cycles is recommended, since it strikes a desirable trade-off between complexity and BER performance. This choice also allows the comparison between our work with previous publications, as stated previously.

5.3.5 Noise Dependent Scaling

As discussed in Section 5.2.3.3, NDS may help prevent the occurrence of the latching problem. In this subsection, we characterize the BER performance of an LDPC-SD without employing scaling methods, as shown in Figure 5.18. Explicitly, the BER performance is provided for the case when the maximum number of decoding cycles is a value in the set of $\{200, 400, 800, 1600, 3200, 6400, 10000\}$. This is compared to the BER performance of a floating point benchmarker that employs maximum numbers of decoding iterations from the set of $\{1, 2, 4, 8, 16, 32, 100\}$. As may beobserved in Figure 5.18, regardless of the number of decoding cycles performed, the LDPC-SD without NDS exhibits a poor error correction performance, suffering from a severe error floor. Based on the comparison between Figures 5.15(a) and 5.18, we may conclude that the NSA of the channel input probabilities is necessary for protecting the LDPC-SDs from latching. Therefore, we adopt the use of NDS in the remainder of this thesis.

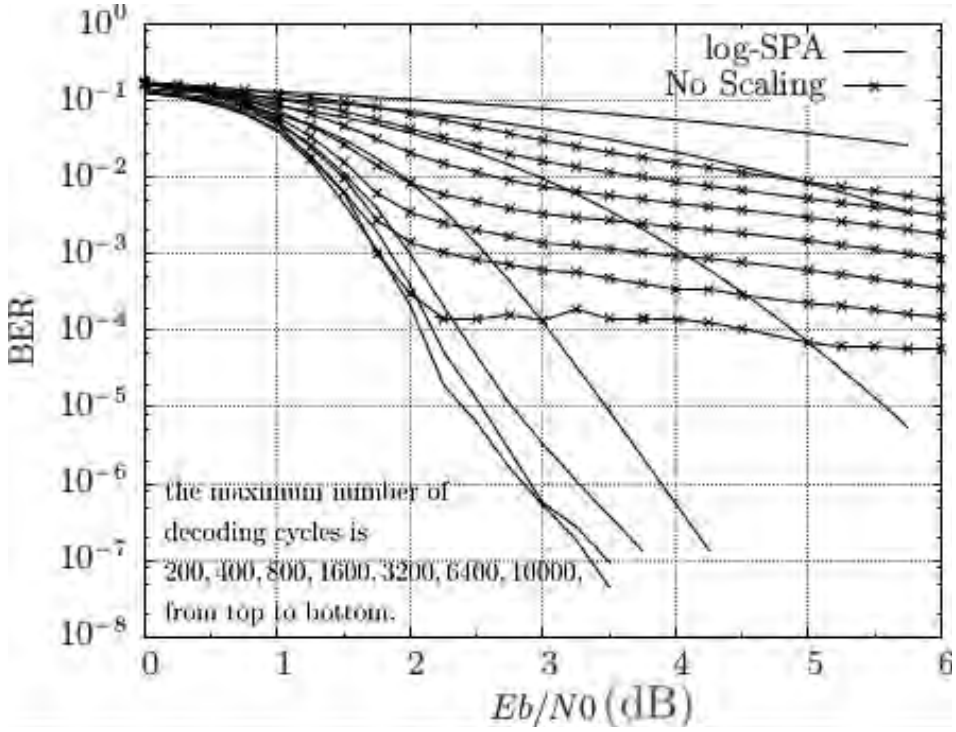


Figure 5.18: The BER performance of the LDPC-SD without NDS, when employing a maximum numbers of decoding iterations from the set of $\{200, 400, 800, 1600, 3200, 6400, 10000\}$, and is compared with a floating point Log-SPA LDPC decoder employing $\{1, 2, 4, 8, 16, 32, 100\}$ decoding iterations.

5.4 Conclusion

In this chapter, we began by identifying the challenges associated with the fully parallel implementation of LDPC-FDs. Motivated by this, we introduce the fully parallel implementation of LDPC-SDs. We discussed the details on the building blocks of stochastic computation that are required in LDPC decoding. Afterwards, we discussed the implementation of the various components of the LDPC-SD. We discussed the latching problems, as well as several techniques that may be used to protect the LDPC-SD from it. Moreover, we have proposed schematics for the decision units of the VNs in the LDPC-SD. By conducting simulations, we have characterized the BER performance of the LDPC-SD as functions of different parameters. Based on this simulation investigation, we recommend a parametrization for the fully parallel implementation of the LDPC-SD, as summarised in Table 5.3, which achieves a desirable trade-off between the implementation complexity and error correction performance. In the nex chapter, the timing error tolerance of this fully parallel implementation of the LDPC-SD will be characterized, in analogy to the investigation conducted for the LDPC-FD in Chapter 4.

Table 5.3: The proposed parametrization.

Initialization method	channel probability
Decision unit scheme	JK-FF
Scaling method	with NDS
EM length	$32, 48, 64$ bits for VNs having degree of $d_{\mathrm{v}} = 2, 3, 6$
IM length	$0, 1, 2$ bits for VNs having degree of $d_{\mathrm{v}} = 2, 3, 6$
Max number of decoding cycles	2000
Clock cycles per decoding cycle	1

Chapter 6

Timing-Error-Tolerant Stochastic LDPC Decoder

6.1 Introduction

As discussed in Chapter 4, Fixed-point LDPC Decoders (LDPC-FDs) have an inherent, but limited, tolerance to timing errors [40, 85, 86, 87, 88, 41, 42, 44]. However, when timing errors affect the Most Significant Bits (MSBs) of the fixed-point probabilities, the error correction performance of an LDPC-FD is significantly degraded [40, 41, 42]. Therefore, the designs of [41, 42] employed additional circuitry for detecting and/or correcting timing errors affecting the MSBs, although this approach is itself associated with an additional overhead. In Chapter 4, we proposed an Fixed-point LDPC Decoder (LDPC-FD) design based on Base Minus Two (BMT) numbers to represent Fixed-Point (FP) probabilities, in order to improve the tolerance to timing errors, compared to the conventional implementation of LDPC-FD using Two's Complement (TC) number representation. However, this improved tolerance to timing errors may be damaged by timing errors that affect the bits in the BMT number representation that are associated with large circuit propagation delays, which limits clock frequency that can be employed for an Application-Specific Integrated Circuit (ASIC) implementation. In Chapter 5, Stochastic LDPC Decoders (LDPC-SDs) were introduced, which use the fraction of bits having the value of 1 within so-called Bernoulli sequences [128] to represent the probabilities exchanged between parity-Check Nodes (CNs) and Variable Nodes (VNs). All bits in these Bernoulli sequences have an equal and relatively low significance, granting them an inherent tolerance to timing errors [129]. Furthermore, stochastic CN and VN circuits are significantly simpler than those of LDPC-FDs. Owing to this, the clock frequency may be further increased in LDPC-SDs, in order to

increase the processing throughput of an LDPC decoder ASIC. Motivated by this, this chapter considers the tolerance of LDPC-SDs to timing error caused by overclocking and supply voltage noise. In particular, supply voltage noise models the effects of IR drop, inductive noise, cross talk, electrostatic discharges, particle strikes, switching noise and fabrication process variations [31, 32, 33, 34, 35, 36, 37].

Several previous research efforts have investigated the tolerance of LDPC decoder ASICs to timing errors and other types of processing errors [40, 85, 86, 87, 88, 41, 42, 44]. In [40], it was demonstrated that the inherent error-correcting capability of LDPC-FDs may also grant them tolerance to timing errors, provided that they do not affect the MSBs of the fixed-point numbers used to represent bit probabilities. Analytical investigations of the error-correcting capability of LDPC decoders in the presence of processing errors were provided in [85, 86, 87, 88]. The designs of [41, 42] employed additional circuitry for detecting and/or correcting timing errors affecting the MSBs of the fixed-point numbers, although this approach is associated with an additional processing overhead, which can limit the attainable processing throughput. Recently, LDPC decoders that are implemented using stochastic processing have attracted significant research interest [121, 122, 120, 126], owing to their low hardware complexity. Compared to the traditional implementations using FP numbers, LDPC-SDs are suited to timing-error-tolerant design, since all bits in the stochastic number representation have an equal and relatively-low significance. Owing to this, a timing error affecting any single bit in a stochastic number representation has only a small effect on the represented bit probabilities, which can be readily tolerated by the inherent error correction capability of the LDPC decoder. In this chapter, we provide a comprehensive investigation into the causes and effects of timing errors on an LDPC-SD. We propose a number of enhancements to the stochastic LDPC decoder, in order to improve its tolerance to timing errors and we expand the analysis to consider timing errors and supply voltage noise, as discussed in Section 4.1. Furthermore, we propose a novel design flow for timing-error-tolerant LDPC decoders.

Figure 6.1 lists the most relevant previous publications along two main timelines, namely the timeline of timing-error-tolerant LDPC decoder design and that of the stochastic implementation of LDPC decoders. Each timeline is represented by a vertical line with the downward direction representing the chronological order, where each knot on the vertical lines represent a publication discussed above.

This chapter follows the proposed design flow of Section 4.2 and models the variation in supply voltage and hence in circuit propagation delay between clock cycles in Section 6.2. We also derive a model of the causes and effects of different types of timing

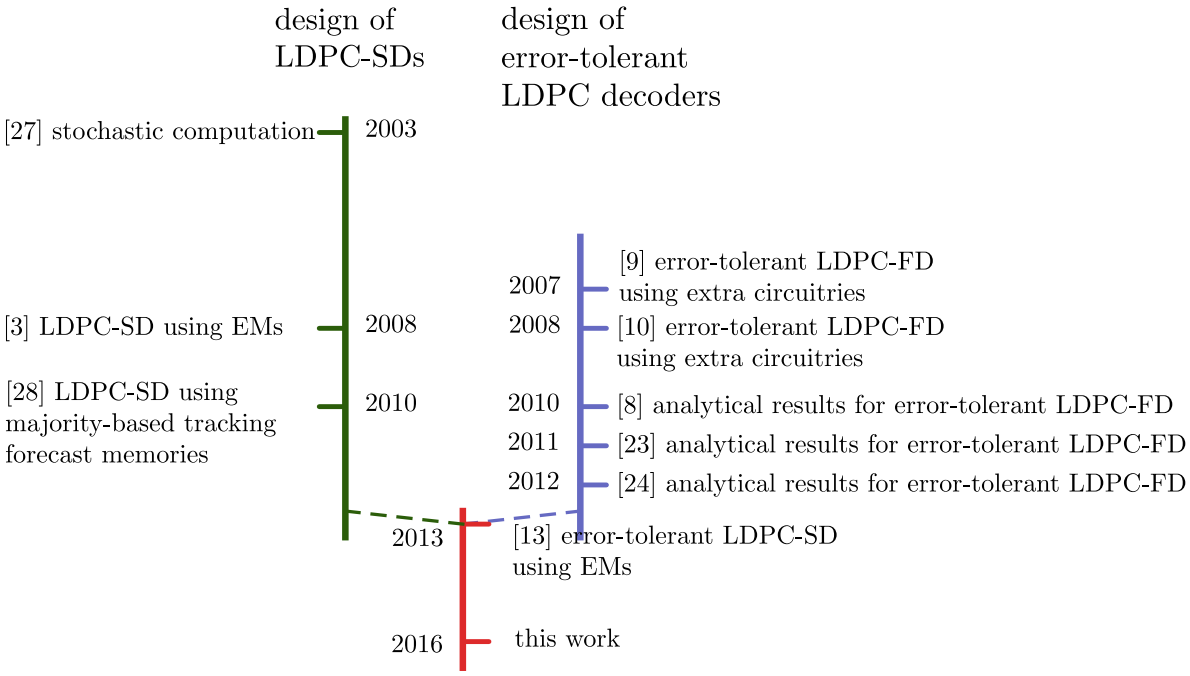


Figure 6.1: Timelines of relevant publications.

errors that are imposed upon LDPC-SDs in Section 6.2. We use this model to investigate the error-correcting performance of the LDPC-SDs in the presence of each type of timing error separately. In this way, we characterise the most detrimental types of timing errors and use this to motivate the design of a novel modified LDPC-SD, as discussed in Section 6.3. This modification redefines the functionality of the VNs, so that they can be implemented using a circuit schematic that has a significantly improved tolerance to timing errors. The error model is employed in the Bit Error Ratio (BER) analysis of Section 6.4, for characterizing the tolerance to both transmission errors and timing errors of the LDPC-SD and the modified LDPC-SD. We show that this modification does not compromise the error correction capability of the LDPC-SD in the absence of timing errors. We demonstrate that the BER performance is not degraded by applying moderate overclocking and that even for aggressive overclocking, only a 1 dB performance degradation is incurred, in the case of the conventional LDPC-SD. Furthermore, we demonstrate that our modified LDPC-SD eliminates this 1 dB performance degradation that is incurred by the LDPC-SD when employing aggressive overclocking, despite requiring no extra circuitries. This significantly improved tolerance to timing errors allows the processing throughput to be increased by up to 69.4% in practice, without compromising either the error correction capability or processing energy consumption of the LDPC-SD. Finally, we offer our conclusions in Section 6.5.

6.2 Overclocking-Induced Timing Error Analysis

As described in Section 4.1, overclocking causes timing errors, whenever there is insufficient time for a signal to propagate to the input of a memory, before its value is clocked into the memory. When aggressive overclocking is employed, the clock period T_{clk} is reduced below the nominal propagation delay t of some signals in a circuit, typically imposing timing errors. However, even moderate overclocking may cause timing errors, since this makes the circuit more sensitive to the late arrival of signals, owing to fluctuations in their propagation delay from one clock cycle to the next. These fluctuations may caused by power supply noise, which models the effects of IR drop, inductive noise, cross talk, electrostatic discharges, particle strikes, switching noise and fabrication process variations [31, 32, 33, 34, 35, 36, 37]. Formally, a timing error occurs for a particular signal when

$$t \times \delta > T_{\text{clk}}, \tag{6.1}$$

where δ characterizes the fluctuation in the propagation delay within the current clock cycle.

There are numerous methods that may be employed to quantify the nominal propagation delays t, their fluctuation δ and the occurrence of timing errors. The most realistic method for quantifying these circuit characteristics is to fabricate an ASIC and to measure them, as described in Section 4.2. However we did not opt for this method, since these experimental results may be influenced by numerous uncontrollable factors, such as temperature variations, electromagnetic radiation and processing variations, which may obfuscate the results. Furthermore, these measurements cannot be readily reproduced by other researchers, if they do not have access to the chip. By contrast, SPICE simulation and manual calculations using component datasheets are immune to random variation and are readily repeatable, as described in Section 4.2. In particular, these methods allow the isolated investigation of how the circuit characteristics are influenced by a particular parameter, such as supply voltage. Therefore, in the following sections of this chapter, the values of delays t, their fluctuation δ and the occurrence of timing errors are modelled using either the data provided in the STMicroelectronics 90 nm datasheet [107] or SPICE simulations.

In Section 6.2.1, we characterize the nominal signal propagation delays t within stochastic VNs and CNs having various degrees. Following this, the fluctuation δ of signal propagation delays from one clock cycle to the next is characterized in Section 6.2.2. Finally, Section 6.2.3 models the causes and effects of timing errors within LDPC-SDs, focussing on VNs having a degree of $d_{\text{v}} = 6$, since they are found to have

the highest susceptibility to timing errors.

6.2.1 Nominal Signal Propagation Delays of Stochastic LDPC Decoders

In this section, we characterize the nominal signal propagation delays t of stochastic CNs and VNs having various degrees, as discussed in Section 5.2.2 and 5.2.3, respectively. More specifically, we consider VNs having degrees of $d_{\mathrm{v}} \in \{2, 3, 6\}$ and CNs having degrees of $d_{\mathrm{c}} \in \{6, 7\}$, as employed in the (1056,528) WiMAX LDPC code [65]. Later, another $(2304, 1920)$ WiMAX LDPC code containing additional VNs having a degree of 4 and additional CNs having a degree of 20 will also be considered. More particularly, we model the nominal propagation delay t of the signal arriving at each D-type Flip-Flop (D-FF) within these VNs and CNs. These are obtained as the maximum of the nominal propagation delays of all paths that end at the particular D-FF. Here, the nominal propagation delay of a path includes the output delay of the D-FF at the beginning of the path, the total delay of the combinational logic along the path and the setup time of the D-FF at the end of the path. We initially focus on the particular case of VNs having a degree of $d_{\mathrm{v}} = 6$, before summarizing the analysis of the other nodes. For VNs having a degree of $d_{\mathrm{v}} = 6$, we consider four sets of D-FFs, which we refer to as the output D-FF, the Intermediate Memory (IM)1 D-FFs, the IM2 D-FFs and the Edge Memory (EM) D-FFs. In Figure 5.11, the output D-FF provides the signal S_{16}, while the signals S_{17} and S_{18} are provided by IM1 and EM D-FFs, respectively. Note that VNs having a degree of $d_{\mathrm{v}} = 3$ require only a single IM and hence they do not have any IM2 D-FFs [121, Figure 6], while VNs having a degree of $d_{\mathrm{v}} = 2$ do not have any IMs and therefore do not have any IM1 D-FFs either, as discussed in Section 5.2.3.

The nominal propagation delay t of the signal arriving at a particular D-FF depends on the state of the VN or CN during both the previous and current clock periods. However, it is not feasible to simulate all the possible combinations of current and previous states. For example, the VN having a degree of $d_{\mathrm{v}} = 6$ shown in Figure 5.11 has 15 inputs, including control signals. In two consecutive clock cycles, these inputs will adopt one of $2^{2\times15} \approx 10^9$ combinations of values. The amount of time required to simulate all of these combinations would be prohibitively excessive. Instead, our analysis focuses on the combinations of the particular signals that have the greatest effect on the operation and nominal propagation delay of the VNs. Unlike in the stochastic CNs, the flow of information within the VNs is controlled by MUltipleXers (MUXs), which have a significant impact on the operations of IMs, the EM and the final output. Owing to their importance, our analysis carefully considers the values of the MUX selector signals in

both the current and the previous clock cycles. More specifically, when a MUX selector signal remains constant between consecutive clock cycles, the propagation delay of the MUX output depends only on the delay of the selected signal. By contrast, if the MUX selector signal is toggled in the current clock cycle, then the propagation delay of the MUX output is given by the maximum of the MUX selector signal's delay and the selected signal's delay. In the case of VNs having a degree of $d_v = 6$, we consider three MUX selector signals, which we refer to as the IM1, the IM2 and the EM MUX selector signals, that are respectively labelled as S_7, S_{10} and S_{13} in Figure 5.11. Our analysis considers all $2^{2\times3} = 64$ combinations of these MUX selector signals in both the current and previous clock cycles, offering a significant simplification, while capturing the main operation of the VN. Similarly, the IM1, IM2 and the EM MUX selector signals are considered for $d_v = 4$ VN, while we consider only the IM1 and the EM MUX selector signals for VN having a degree of $d_v = 3$, as well as only the EM MUX selector signal for the VN having a degree of $d_v = 2$, since they have only a single IM and no IMs, respectively.

Table 6.1 provides the nominal propagation delays associated with the input of each D-FF in stochastic VNs and CNs having various degrees. In Table 6.1, 'toggle' indicates that the corresponding MUX selector signal of a VN has different values in the previous and current clock cycles. By contrast, '1' and '0' indicate that the MUX selector signal has maintained a constant value of either 1 or 0 in the previous and current clock cycles, respectively. Finally 'any' indicates that the nominal propagation path delay is not dependent on the corresponding MUX selector signal. Here, in order to simplify the analysis, we model the propagation delay of each gate using a constant value, which is obtained by considering its worst-case switching condition and the worst-case loading that is imposed by the particular set of gates that it drives. This approach is justified, since the loads imposed by the different pins of each gate vary by only about 1%. Furthermore, this worst-case approach guarantees a worst-case upper bound for the performance degradation of Sections 6.4, since timing errors are most likely to occur for signals having long propagation delays. For providing deeper insights into the propagation delays and for obtaining reproducible results, the values of Table 6.1 were calculated manually using data provided in the STMicroelectronics 90 nm datasheet [107]. Note that the nominal path delay of CNs having a degree of $d_c = 20$ is omitted from Table 6.1 for simplicity, since it is only slightly higher than that of CNs having a degree of $d_c = 7$, when implemented using the CN structure of Figure 5.5. Also note that the wire delay is neglected in our analysis of the nominal signal propagation delays of the LDPC-SD. This is justified, since 90 nm ASIC implementations of LDPC-SDs typically have dimensions that are no greater than a few millimetres [122]. Owing to

this, the expected maximum wire delays may be of the order of tens of picoseconds [130], which is negligible compared to the propagation delays within the stochastic nodes.

6.2.2 Propagation Delay Fluctuation

As described in Section 4.1, supply voltage noise can model the effects of IR drop, inductive noise, cross talk, electrostatic discharges, particle strikes, switching noise and fabrication process variations [31, 32, 33, 34, 35, 36, 37]. In this case, the supply voltage V_{DD} may be assumed to have a Gaussian distribution with a mean value that is equal to the nominal supply voltage μ and a standard deviation σ, which is related to the circuit layout, fabrication process and technology scale, in circuits containing a sufficiently high number of gates [131, 103]. Since the propagation delay of a particular type of gate is a decreasing function of the supply voltage V_{DD}, it can be modelled as a random variable that is independently selected from a particular Probability Density Function (PDF) in each successive clock cycle. We found that the time-normalized NAND gate delay PDF can accurately represent the PDFs of other types of logic gates, which is shown in Figure 4.6 for $3\sigma/\mu \in \{0.01, 0.1, 0.3\}$ [103]. For example, when $3\sigma/\mu = 0.1$, the propagation delay of a NAND gate extends above its nominal value by 10% or more with a probability of around 1%. Correspondingly, the probability of a NOT gate's propagation delay extending by 10% or more above its nominal value is also around 1%, when $3\sigma/\mu = 0.1$. Hence, we employ the time-normalized NAND gate delay PDF to model the propagation delay distribution of all circuit components, in order to simplify the analysis of the following sections, as recommended in [103].

6.2.3 Effects of timing errors in Stochastic LDPC Decoders

As shown in Table 6.1, the maximum nominal propagation delay t within stochastic VNs is 727.6 ps, rendering them more susceptible to timing errors than the CNs, which have a maximum propagation delay of 618.1 ps. In order to simplify our analysis, we assume that the normalized delay multiplier δ is never large enough and the clock period T_{clk} is never low enough to cause timing errors within the CNs. More specifically, we assume that $(620 \times \delta) > T_{\text{clk}}$ never happens. This assumption allows us to focus our attention on the specific paths within the stochastic VNs that have nominal propagation delays t of at least 620 ps, as highlighted in bold within Table 6.1. This approach is justified, since our experiments show that the inherent error tolerance of the LDPC-SD entirely breaks down, when timing errors occur in the CNs. As shown in Table 6.1, nominal propagation delays t of at least 620 ps are manifested in $d_{\text{v}} = 2, 3, 4$ and 6 VNs. In

Table 6.1: Nominal propagation delays within the VNs and CNs of the LDPC-SD of [121], when employing STMicroelectronics 90 nm technology. The nominal propagation delays of the modified LDPC-SD are provided in brackets, where they differ.

Node	Updated D-FF	MUX selector signal			Nominal propagation delay t (ps)				
		EM	IM1	IM2	Degree 2	Degree 3	Degree 4	Degree 6	Degree 7
VN	EM		toggle	any	463.6(439.0)	**687.9(650.6)**	**687.9(650.6)**	**727.6(674.6)**	–
		toggle	1	toggle		595.4 (558.1)	**687.9(650.6)**	**727.6(674.6)**	
				1			595.4(558.1)	**633.5**(580.5)	
				0				**633.5**(580.5)	
			0	toggle		542.1 (504.8)	**687.9(650.6)**	**727.6(674.6)**	
				1			595.4(558.1)	**633.5**(580.5)	
				0			542.1(504.8)	**624.5** (571.5)	
		1	toggle	any	387.9	573.6	573.6	597.6	
			1	toggle		481.1	573.6	597.6	
				1			481.1	503.5	
				0				503.5	
			0	toggle		427.8	573.6	597.6	
				1			481.1	503.5	
				0			427.8	494.5	
		0	any	any		573.6	573.6	597.6	
	Output	toggle 0 → 1	toggle	any	463.6(439.0)	**687.9(650.6)**	**687.9(650.6)**	**727.6(674.6)**	
			1	toggle		595.4 (558.1)	**687.9(650.6)**	**727.6(674.6)**	
				1			595.4(558.1)	**633.5**(580.5)	
				0				**633.5**(580.5)	
			0	toggle		542.1 (504.8)	**687.9(650.6)**	**727.6(674.6)**	
				1			595.4(558.1)	**633.5**(580.5)	
				0			542.1 (504.8)	**624.5** (571.5)	
		toggle 1 → 0	toggle	any	**656.6(653.3)**	**724.7(723.5)**	**724.7(723.5)**	**727.6(723.5)**	
			1	toggle				**727.6(723.5)**	
				1				**724.7(723.5)**	
				0				**724.7(723.5)**	
			0	toggle				**727.6(723.5)**	
				1				**724.7(723.5)**	
				0				**724.7(723.5)**	
		1	toggle	any	387.9	573.6	573.6	597.6	
			1	toggle		481.1	573.6	597.6	
				1			481.1	503.5	
				0				503.5	
			0	toggle		427.8	573.6	597.6	
				1			481.1	503.5	
				0			427.8	494.5	
		0	any	any	**656.6(653.3)**	**724.7(723.5)**	**724.7(723.5)**	**724.7(723.5)**	
	IM1	any	toggle	any	N/A	393.0	393.0	417.0	
			1			301.1	301.1	323.5	
			0					323.5	
	IM2	any	any	toggle	N/A	N/A	393.0	417.0	
				1			301.1	323.5	
				0				323.5	
CN	Output	N/A	N/A	N/A	–	–	–	511.0	618.1

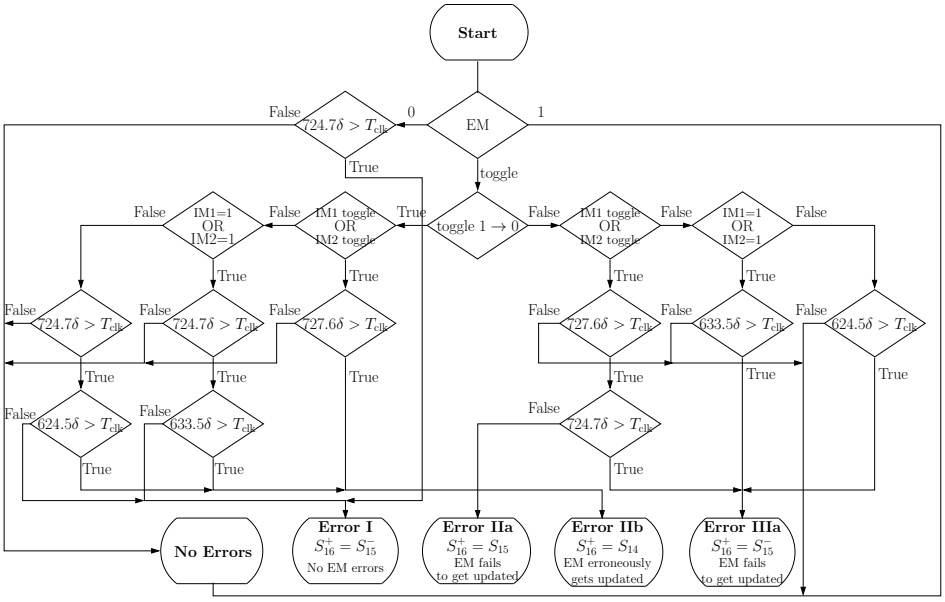
order to simplify our discussions, we exemplify our analysis by focusing on $d_v = 6$ VNs of Figure 5.11. This is because Section 6.2.1 revealed that these VNs are associated with the longest propagation delays and hence have the highest susceptibility to timing errors. Note that the corresponding analysis carried out for VNs having lower degrees is similar, but simpler. In the case of the $d_v = 6$ VN of Figure 5.11, only three types of timing errors need to be considered, as summarized by the flowchart of Figure 6.2(a). Note that corresponding flowcharts are provided for VNs having degrees of $d_v = 2$ and 3 in Figure 6.3 and 6.4, respectively. Due to the identical implementation in the first layer of Figure 5.7(c) and 5.7(d), VNs having a degree of $d_v = 4$ have the similar timing characteristic to the VNs having a degree of $d_v = 3$, as shown in Table 6.1, despite employing two IMs in parallel instead of only a single IM in the first layer of Figure 5.7(c) in parallel. Therefore, the flowchart for VNs having a degree of $d_v = 4$ can be obtained by replacing the condition 'IM1 toggle' in the flowchart of Figure 6.4(a) with 'IM1 or IM2 toggle'. These flowcharts may be derived by combining (6.1) with the MUX selector signal conditions and with the nominal propagation delays t of Table 6.1, as described for VNs having a degree of $d_v = 6$ in the following paragraphs for each type of timing error. More specifically, each value of t shown in Figure 6.2(a), *e.g.* 674.6 ps, is obtained from the appropriate entry in Table 6.1, which quantifies the timing characteristics of the corresponding path within the selected VN, as obtained during our simulations described in Section 6.2.1.

6.2.3.1 Timing Error Type I

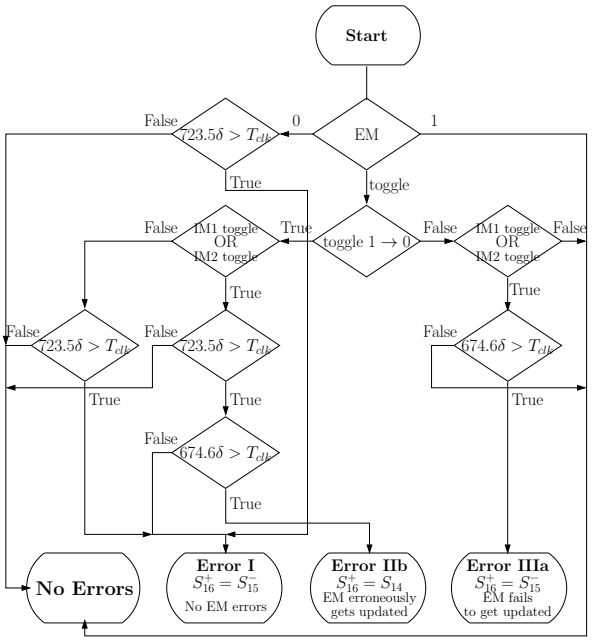
In this type of timing error, the propagation of S_{15} is not completed before the end of the clock cycle, but the EM MUX selector signal S_{13} arrives on time. However, the EM MUX will not select the late S_{15} signal if $S_{13} = 1$, preventing the occurrence of a timing error. Therefore, $S_{13} = 0$ is a condition for a Type I error to occur. Instead of clocking the correct value of the signal S_{15} into the output D-FF, its value from the previous clock cycle S_{15}^{-} is latched. Note that timing errors are not inflicted upon the EM D-FFs, when a Type I error occurs, since the late S_{15} signal is not an input to the EM.

6.2.3.2 Timing Error Type II

Type II errors occur if S_{15} arrives on time, but S_{13} is toggled and arrives late. In this case, it is the previous value of S_{13}^{-} that controls both the updating of the EM D-FFs and the selection of the signal that is clocked into the output D-FF. Type II errors can

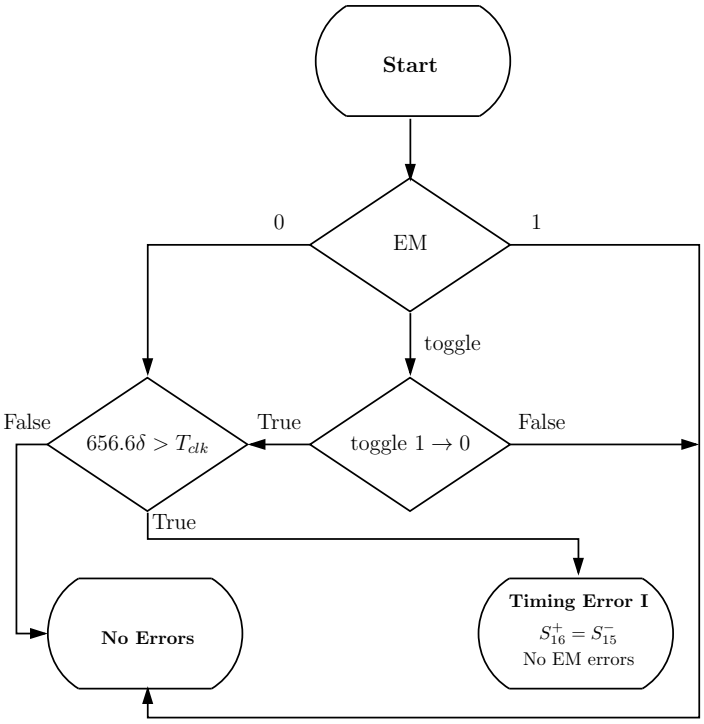


(a) Before modification.

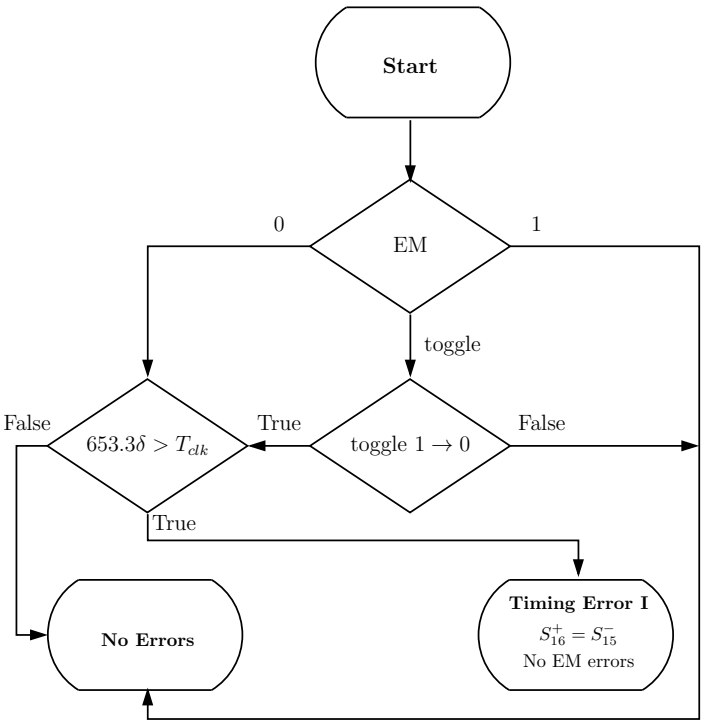


(b) After modification. Observe that a number of paths leading to type II and II timing errors have been removed, hence eliminating the type IIa errors.

Figure 6.2: Flowcharts illustrating the causes and effects of timing errors in stochastic VNs having a degree of $d_{\mathrm{v}} = 6$.

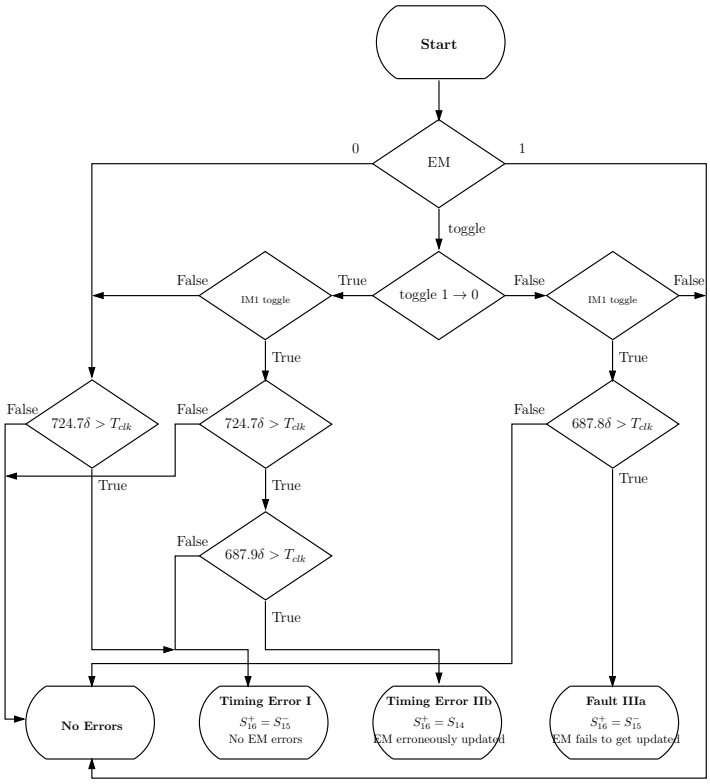


(a) Before modification.

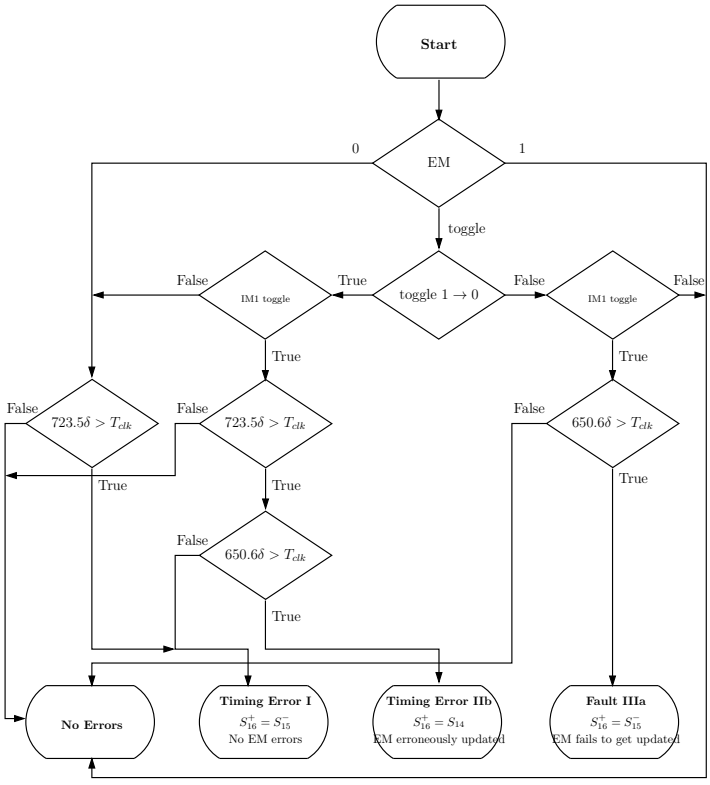


(b) After modification. Observe that the nominal propagation delay of the path has been reduced.

Figure 6.3: Flowcharts illustrating the causes and effects of timing errors in stochastic VNs having a degree of $d_{\mathrm{v}} = 2$.



(a) Before modification.



(b) After modification. Observe that the nominal propagation delays of all paths have been reduced.

Figure 6.4: Flowcharts illustrating the causes and effects of timing errors in stochastic VNs having a degree of $d_{\mathrm{v}} = 3$. Note that the flowchart for VNs having a degree of $d_{\mathrm{v}} = 4$ can be obtained by replacing the condition 'IM1 toggle' with 'IM1 or IM2

be further classified into Type IIa and IIb errors, depending on the value of S_{13}^{-}. A Type IIa error occurs, when the EM MUX selector signal is toggled according to $S_{13}^{-} = 0$ and $S_{13} = 1$. In this case, the updating of the EM will be erroneously prevented and instead of S_{14}, it will be S_{15} that is clocked into the output D-FF. By contrast, $S_{13}^{-} = 1$ and $S_{13} = 0$ is associated with a Type IIb error, which results in S_{14} being erroneously clocked into the first EM D-FF, as well as into the output D-FF.

6.2.3.3 Timing Error Type III

Finally, the Type III errors occur, when S_{13} is toggled and arrives late, as well as S_{15} arriving late. Again, it is the previous value of S_{13}^{-} that controls the updating of the EM D-FFs and the selection of the signal that is latched into the output D-FF. Similarly, Type III errors can also be further classified into Type IIIa and IIIb errors, depending on the value of S_{13}^{-}. A Type IIIa error occurs when $S_{13}^{-} = 0$ and $S_{13} = 1$, causing the updating of the EM to be erroneously prevented and the wrong signal to be clocked into the output D-FF, instead of S_{14}. However, since S_{15} is late, it will not be clocked into the output D-FF as in the Type IIa error. Instead, its value in the previous clock cycle S_{15}^{-} will be clocked. By contrast, $S_{13}^{-} = 1$ and $S_{13} = 0$ is associated with a Type IIIb error, in which case the EM MUX will not select the late S_{15} signal. Instead, the timing error will cause S_{14} to be erroneously clocked into the first EM D-FF, as well as into the output D-FF. Note that Type IIIb errors have the same effect as Type IIb errors and so are merged into the type IIb outcome of Figure 6.2(a).

6.2.4 Validation in SPICE

SPICE simulation has been shown to accurately predict practical measurements of circuit behaviour [46, 108], when operating in the presence of timing errors and other types of processing fault. However, since the SPICE simulation of the entire LDPC-SD would be impractical, our simulations considered only individual VNs and CNs in isolation. During these simulations, the overclocking was applied so that we could observe the occurrence of timing errors and validate the timing error model of Figure 6.2(a). This approach is exemplified by the results of the VNs having a degree of $d_{\mathrm{v}} = 6$ portrayed in Figure 6.5. More specifically, Figure 6.5 compares the ideal zero-delay response of the $d_{\mathrm{v}} = 6$ VN with the simulated response for the case of implementation using STMicroelectronics 90 nm technology, with $V_{\mathrm{DD}} = 1.0$ V and $T_{\mathrm{clk}} = 700$ ps. As shown in Figure 6.5, timing errors of Type IIb and I occur in clock cycles 2 and 3, respectively.

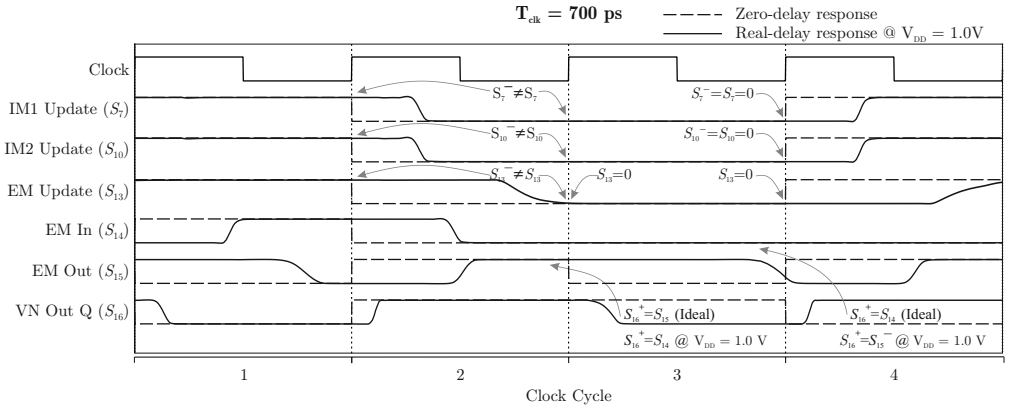


Figure 6.5: SPICE simulation demonstrating the occurrence of Type IIb and Type I timing errors in a stochastic VN of [121] having a degree of $d_v = 6$, when employing STMicroelectronics 90 nm technology, where the supply voltage V_{DD} is set as 1.0 V, the clock period is set as 700 ps and random signals are used as the input.

6.3 Modified Stochastic LDPC Decoder

As we will demonstrate later in Section 6.4, the LDPC-SD has a inherent tolerance to Type I timing errors. However, our results show that its BER performance is limited by the occurrence of Type II and III timing errors. In Section 6.2.3 we showed that these types of timing errors are caused by the late arrival of the EM MUX selector signal, which is labeled S_{13} in Figure 5.11. Motivated by this, Section 6.3.1 proposes a modification to the EM, which reduces the nominal propagation delay of S_{13} and mitigates the occurrence of Type II and III timing errors. The causes and effects of timing errors in the modified LDPC-SD are analysed in Section 6.3.2, using a similar approach to that discussed in Section 6.2. The BER performance of the modified LDPC-SD will be characterised later in Section 6.4.2 and compared with that of the LDPC-SD of [121], both in the absence and presence of timing errors.

6.3.1 Modified EM

As shown in Figure 5.11, the EM MUX selector signal S_{13} also acts as the selector signal for a large number of MUXs within the EM structure of [121]. More specifically, 32, 48, 48 and 64 MUXs are employed within the EMs of VNs having degrees of $d_v = 2, 3, 4$ and 6, respectively. These MUXs generate a large capacitive load, which is the cause of the high nominal propagation delay of the EM MUX selector signal S_{13}.

In order to reduce this capacitive load and reduce the nominal propagation delay of the EM MUX selector signal S_{13}, we propose the modified EM structure of Figure 5.11. This schematic of Figure 5.11 is repeated here for the convenience of reading. Here,

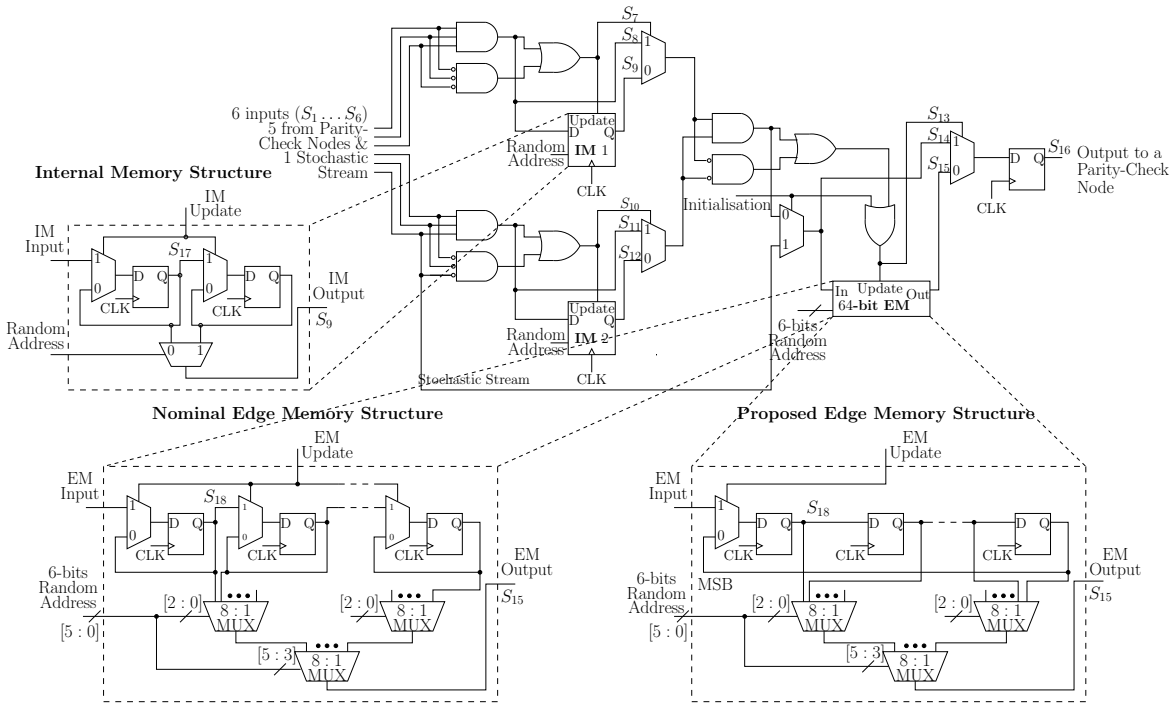


Figure 6.6: Schematic of a stochastic $d_v = 6$ VN. Corresponding schematics for $d_v = 2$ and 3 VNs can be found in [121, Figure 6].

the EM MUX selector signal S_{13} acts as the selector signal for only one MUX within the EM structure. This is achieved by configuring the EM as a ring buffer, rather than as a shift register, as shown in Figure 5.11. This is motivated by the observation that a ring buffer can fulfil the same role as a shift register in an EM, despite having a different operation. More explicitly, the role of an EM is to store recent VN decisions, so that when a decision cannot be made in future decoding cycles, one may be selected randomly from the EM. This role may be fulfilled by both a shift register and a ring buffer, since both replace older decisions with new ones, as they are clocked into the EM. Note however that in contrast to a shift register, the ring buffer does not necessary replace the oldest decision when it is provided with a new one. Despite this, we will show in Section 6.4.2, that the replacement of the shift registers with ring buffers does not significantly affect the BER performance of the LDPC-SD in the absence of timing errors. Note that this modification eliminates all but one of the 2:1 MUXs that are employed in each EM. Owing to this, the total number of gates required to implement the $k = 528$ and $n = 1056$ IEEE 802.16e (WiMAX) LDPC decoder is reduced by 28.6%, from around 5.6×10^5 gates to around 4.0×10^5 gates. Furthermore, the proposed modification allows the OR gate that supplies the signal S_{13} to be replaced with a significantly smaller OR gate having a lower drive capability, owing to the reduced capacitive load that is imposed by these 2:1 MUXs.

6.3.2 Overclocking-Induced Timing Error Analysis

The same approach described in Section 6.2 is employed here for quantifying the nominal propagation delays of every path within the modified LDPC-SD, as provided in brackets in Table 6.1. It can be seen that many of the large nominal propagation delays are reduced in the modified LDPC-SD, owing to the reduced capacitive load that it imposes upon the EM MUX selector signal S_{13}. As shown in Table 6.1, the number of highlighted entries having propagation delays exceeding that of the CN having a degree of $d_c = 7$ is reduced after the modification. Owing to this, a number of routes through the flowchart of Figure 6.2(a) leading to Type II and III timing errors can be removed, as shown in the modified flowchart of Figure 6.2(b). Consequently, the Type IIa timing errors are eliminated. Furthermore, higher values are required for the normalized delay multiplier δ to satisfy the conditions for the remaining routes towards the Type II and III timing errors. As a result, Type II and III timing errors can be expected to occur significantly less often in the modified LDPC-SD, compared to the design of [121].

6.4 Simulation Results and Discussions

In this section, we characterize the impact of timing errors upon the error correction capability of LDPC-SDs. Again, the most realistic method of characterizing this impact would be to fabricate an ASIC and to measure the occurrence of decoding errors. However, this is not preferred for the reasons discussed in Section 6.2. Instead, the BER results of Figure 6.7 and 6.8 were obtained by applying the timing error model of Section 6.2.3 within Monte-Carlo simulations written in C++. This has the advantage of allowing us to investigate the individual impact of the different types of timing errors that were introduced in Section 6.2.3, by disabling the occurrence of all other types of timing errors. This C++ simulation also allows the BER of the LDPC-SD to be characterised without incurring a significant overhead associated with simulating the occurrence of timing errors using SPICE, for example. Owing to this, a single CPU core requires about 24 hours of runtime to simulate the decoding of 10^9 message bits, which is typical of C++ BER simulations.

Our simulation parameters are summarized in Table 6.2. In addition to the STMicroelectronics 90 nm technology [107] discussed in Section 6.2.1, the Oklahoma State University FreePDK 45 nm technology [133] is also employed, for the sake of investigating timing errors in different fabrication scales. The BER performance of the LDPC-SD and the modified LDPC-SD of Section 6.3 are characterised for the case of employing

Table 6.2: Simulation parameters.

Manufacturer's datasheets	STMicroelectronics 90 nm [132] and FreePDK 45 nm [133]
$(T_{\text{clk}}, 3\sigma/\mu)$ for 90 nm	$(628.3, 0.01)$, $(718.8, 0.01)$, $(718.8, 0.1)$, $(1217.3, 0.1)$ and $(1217.3, 0.3)$
$(T_{\text{clk}}, 3\sigma/\mu)$ for 45 nm	$(760.0, 0.01)$, $(869.5, 0.1)$, $(1171.2, 0.1)$, $(1472.8, 0.1)$ and $(1472.8, 0.3)$
LDPC code	WiMAX LDPC $(1056, 528)$ $(2304, 1920)$ [65]
EM length	$32, 48, 48, 64$ bits for VNs having degree of $d_{\text{v}} = 2, 3, 4, 6$
IM length	$0, 1, 1, 2$ bits for VNs having degree of $d_{\text{v}} = 2, 3, 4, 6$
Max number of decoding cycles	2000
Clock cycles per decoding cycle	1

Binary Phase Shift Keying (BPSK) for transmission over an Additive White Gaussian Noise (AWGN) channel, in order to facilitate comparisons with the results of [121]. In the following sections, we characterise the BER performance of two WiMAX LDPC codes, for which $(1056, 528)$ and $(2304, 1920)$. The former code has medium code rate of $1/2$ and a medium length of $K = 528$. By contrast, the latter code has the longest code length of $K = 2304$ and the highest code rate of $5/6$ supported in WiMAX, offering diverse application examples. Moreover, since our design is dependent on the characteristics of individual nodes, rather than on the interconnectivity among them, the BER plots of the two codes demonstrate that our design may be applied to various types of LDPC codes, provided that a similar timing analysis to that discussed in Section 6.2.1 is completed for nodes having degrees different from those considered here. The number of memory elements used in the EMs are selected as 32, 48, 48 and 64 for VNs having degrees of $d_{\text{v}} = 2, 3, 4$ and 6, respectively, since these are the numbers employed in [121]. Likewise, for this reason, the IMs use $1, 1$ and 2 memory elements for VNs having degrees of $d_{\text{v}} = 3, 4$ and 6, respectively, while VNs having a degree of $d_{\text{v}} = 2$ contain no IMs [121]. The choices of T_{clk} and $3\sigma/\mu$ have been discussed previously in Sections 6.2 and 6.3.2. In each clock cycle, our simulations employed a different value for the normalized delay multiplier δ, which was randomly selected from the distribution of Figure 4.6 associated with the selected values of $3\sigma/\mu$. The cause and effect of timing errors on each path within each VN in LDPC-SDs are revealed by identifying the appropriate path through the flowcharts of Figure 6.2(a) and 6.2(b), by using the comparisons $t \times \delta > T_{\text{clk}}$. All of the BER results are obtained by simulating the transmission of at least 10^6 codewords encoded from random information bits, then decoding the received codewords iteratively until early termination is triggered by the occurrence of an all-zero syndrome [121], or until the maximum affordable number of decoding cycles is reached. We limit the maximum number of decoding cycles to 2000, since this value is recommended for achieving the lowest BER shown in [121, Figure 12]. However, since the early termination cannot be anticipated in advance, the amount of

time that is reserved for decoding each codeword is given by $2000T_{\text{clk}}$. This corresponds to a processing throughput of $R_{\text{b}} = k/(2000T_{\text{clk}})$ information bits per second.

In Section 6.4.1, simulations were conducted for characterizing the BER performance of the LDPC-SD in the presence of timing errors, when employing diverse values of both the clock period T_{clk}, as well as of the nominal supply voltage μ and its standard deviation σ. Following this, Section 6.4.2 investigates and compares the BER performance of the novel modification proposed in Section 6.3 using similar simulations.

6.4.1 Inherent Timing Error Tolerance

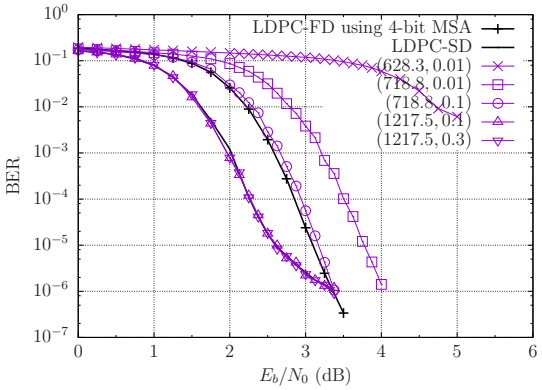
As described in Section 6.2.3, the analysis presented in the flowchart of Figure 6.2(a) assumes that $\delta < T_{\text{clk}}/620$, so that timing errors do not occur in the CNs. In order to satisfy this assumption in at least 99% of the clock cycles, the specific combinations of $(T_{\text{clk}}, 3\sigma/\mu)$ that we consider for the LDPC-SD of [121] are $(628.3, 0.01)$, $(718.8, 0.01)$, $(718.8, 0.1)$, $(1217.3, 0.1)$ and $(1217.3, 0.3)$. Note that $T_{\text{clk}} = 1217.3$ ps represents moderate overclocking, since this value exceeds 727.6 ps, which is the longest nominal propagation delay of Table 6.1. By contrast, both $T_{\text{clk}} = 628.3$ ps and 718.8 ps represent aggressive overclocking.

In order to consider the effect of different technology scales, the analysis described in the preceding sections for the STMicroelectronics 90 nm technology was also conducted for the Oklahoma State University FreePDK 45 nm technology [133]. In this case, our analysis assumes that $\delta < T_{\text{clk}}/750$, which is satisfied in 99% of the clock cycles, when using the specific combinations for $(T_{\text{clk}}, 3\sigma/\mu)$ of $(760.0, 0.01)$, $(869.5, 0.1)$, $(1171.2, 0.1)$, $(1472.8, 0.1)$ and $(1472.8, 0.3)$. Here, $T_{\text{clk}} = 1472.8$ ps represents moderate overclocking, while $T_{\text{clk}} = 1171.2$ ps, 869.5 ps and 760.0 ps represent aggressive overclocking.

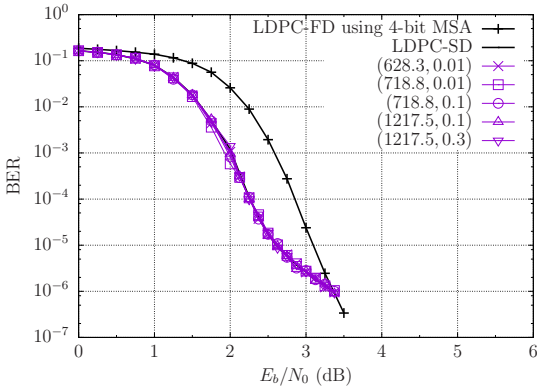
6.4.1.1 Tolerance to All Types of Timing Errors

Figure 6.7(a) demonstrates the effect of all types of timing errors on the BER performance, when the LDPC-SD is implemented using 90 nm technology, while Figure 6.8(a) provides the corresponding plot for 45 nm technology. The BER performance is also plotted for three benchmarkers, namely for the corresponding widely-used LDPC-FD using Min-Sum Algorithm (MSA) and a 4-bit FP TC number representation [75] and for the LDPC-SD in the absence of timing errors. In each case, iterative decoding is continued until convergence to a legitimate LDPC codeword is achieved.

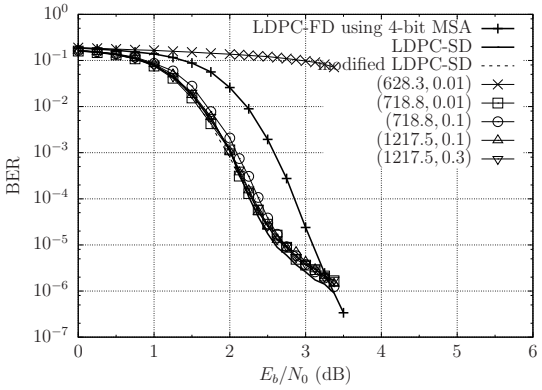
As shown in Figure 6.7(a) and 6.8(a), the BER performance of the LDPC-SD out-



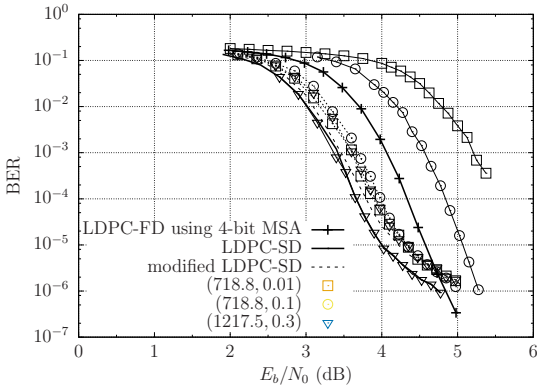
(a) LDPC-SD for decoding (1056,528) code, in the presence of Type I, II and III timing errors.



(b) LDPC-SD for decoding (1056,528) code, in the presence of only Type I timing errors.



(c) Modified LDPC-SD for decoding (1056,528) code, in the presence of Type I, II and III timing errors.



(d) LDPC-SD and modified LDPC-SD for decoding (2304,1920) code, in the presence of Type I, II and III timing errors.

Figure 6.7: BER performance of the LDPC-SD and modified LDPC-SD for decoding (1056,528) and (2304,1920) WiMAX LDPC codes, using 90 nm technology.

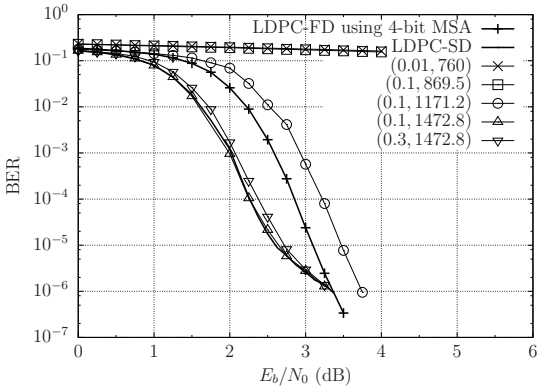
performs the 4-bit MSA benchmarker, confirming the high-performance operation of the LDPC-SD in the absence of timing errors. Furthermore, these figures show that when the moderate overclocking of $T_{\text{clk}} = 1217.3$ ps is employed for 90 nm and when $T_{\text{clk}} = 1472.8$ ps is employed for 45 nm, the resultant timing errors do not significantly degrade the performance of the LDPC-SD of [121]. It is only when employing the aggressive overclocking of $T_{\text{clk}} = 718.8$ ps for 90 nm and $T_{\text{clk}} = 1171.2$ ps for 45 nm, that the BER performance of the LDPC-SD becomes degraded by about 1 dB. When employing the aggressive overclocking of $T_{\text{clk}} = 628.3$ ps for 90 nm and $T_{\text{clk}} = 869.5$ ps or $T_{\text{clk}} = 760.0$ ps for 45 nm, the BER performance of the decoder is degraded so severely that it converges perceivably slower. Based on these observations, we consider the LDPC-SD to have a degree of inherent tolerance to timing errors, even though no additional circuitry is employed for detecting or correcting these errors.

6.4.1.2 Tolerance to Timing Error Type I

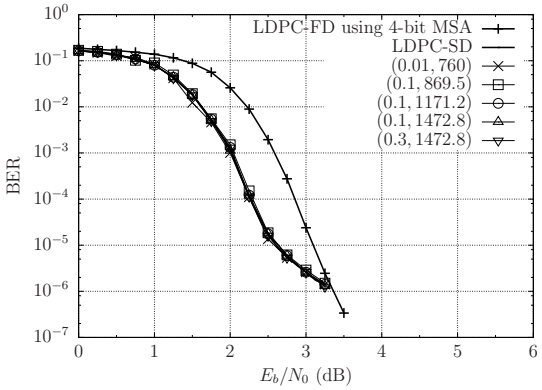
To further investigate the effects imposed by different types of timing errors on the BER performance of the LDPC-SD of [121], we repeated the simulations of Figure 6.7(a) and 6.8(a), but with the timing error types II and III turned off. This was achieved by simulating the *occurrence* of timing errors as usual, but only actually *imposing* those timing errors, if they were of Type I. As shown in Figure 6.7(b) and 6.8(b), the resultant simulations yielded BER performances that are within 0.1 dB of those achieved by the benchmarkers. This indicates that it is the occurrence of Type II and III timing errors that causes the significant degradation of the BER in Figure 6.7(b) and 6.8(b), when aggressive overclocking is employed. Furthermore, this demonstrates that the LDPC-SD has an inherent tolerance to Type I timing errors. It is these observations that motivated the modified LDPC-SD of Section 6.3, which is designed to have an increased tolerance to Type II and III timing errors.

6.4.2 Improved Timing Error Tolerance

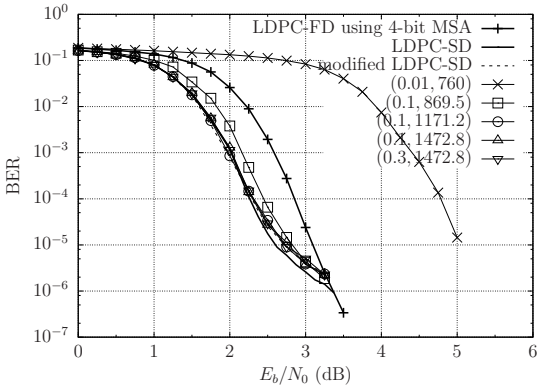
The simulations of Figure 6.7(a) and 6.8(a) were repeated, but employing the modified LDPC-SD instead of the conventional LDPC-SD of [121], in order to characterize the corresponding improvement in BER performance, shown in Figure 6.7(c) and 6.8(c). An additional benchmaker was introduced, namely the modified LDPC-SD in the absence of timing errors. In this case, Figure 6.7(c) and 6.8(c) show that the BER performance of the modified LDPC-SD structure is similar to that of the conventional LDPC-SD of [121], despite having a different EM operation, as described in Section 6.3.1. Further-



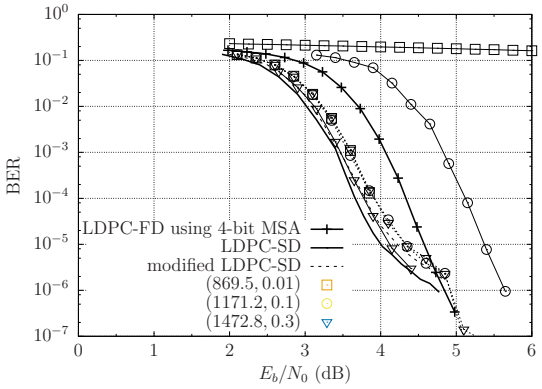
(a) LDPC-SD for decoding (1056,528) code, in the presence of Type I, II and III timing errors.



(b) LDPC-SD for decoding (1056,528) code, in the presence of only Type I timing errors.



(c) Modified LDPC-SD for decoding (1056,528) code, in the presence of Type I, II and III timing errors.



(d) LDPC-SD and modified LDPC-SD for decoding (2304,1920) code, in the presence of Type I, II and III timing errors.

Figure 6.8: BER performance of the LDPC-SD and modified LDPC-SD for decoding (1056,528) and (2304,1920) WiMAX LDPC codes, using 45 nm technology.

more, the BER results of Figure 6.7(c) and 6.8(c) reveal that the modified LDPC-SD has a higher tolerance to timing errors than the conventional LDPC-SD of [121]. As shown in Figure 6.7(c), even the aggressive overclocking of $T_{\text{clk}} = 718.8$ ps for 90 nm technology fails to impose a significant degradation on the BER performance of the modified LDPC-SD. Likewise, a significant BER degradation is avoided, when employing the aggressive overclocking of $T_{\text{clk}} = 1171.2$ ps for the 45 nm technology, as shown in Figure 6.8(c). It may be also observed from Figure 6.7(d) and 6.8(d), that our modification facilitates a similar BER improvement for the (2304,1920) code of the WiMAX LDPC family, which has a different code rate and length. In summary, it may be observed that the proposed modification eliminates the 1 dB performance degradation that is suffered by the LDPC-SD, when employing aggressive overclocking.

6.4.3 Processing Throughput

An alternative comparison can be made by observing that for 90 nm, the BER performance of the modified LDPC-SD recorded for $(T_{\text{clk}}, 3\sigma/\mu) = (718.8, 0.1)$ is similar to that of the conventional LDPC-SD of [121] for $(T_{\text{clk}}, 3\sigma/\mu) = (1217.5, 0.1)$. Therefore, the proposed modification may be deemed to offer a 41% clock period reduction. This corresponds to a 69.4% increase in processing throughput, namely from $R_{\text{b}} = 225.2$ Mbit/s using the conventional LDPC-SD to $R_{\text{b}} = 367.3$ Mbit/s for the modified design, when employing the upper limit of 2000 clock cycles to decode every encoded codeword. However, this upper limit is typically only hit at low channel SNR values. At higher channel SNR values, the stochastic decoding process is able to reach a syndrome of zero using significantly fewer clock cycles. When this happens, the decoding of the current encoded codeword can be stopped early and the decoding of the next codeword can commence without significantly degrading the BER performance. Indeed, Figure 6.9 shows that the modified decoder is capable of achieving a processing throughput as high as 3.8 Gbit/s at a channel SNR per bit of $E_{\text{b}}/N_0 = 5$ dB, which is about 60% higher than throughput reported for the conventional LDPC-SD of [121]. Note that this average number of decoding cycles is obtained based on the simulation of 10^6 codewords.

Similarly, for 45 nm, the BER performance of the modified LDPC-SD recorded for $(T_{\text{clk}}, 3\sigma/\mu) = (1171.2, 0.1)$ is similar to that of the conventional LDPC-SD of [121] for $(T_{\text{clk}}, 3\sigma/\mu) = (1472.8, 0.1)$, which corresponds to a 20% clock period reduction. This represents a 25.5% increase in processing throughput, namely from $R_{\text{b}} = 179.3$ Mbit/s using the conventional LDPC-SD of [121] to $R_{\text{b}} = 225.2$ Mbit/s for the modified design.

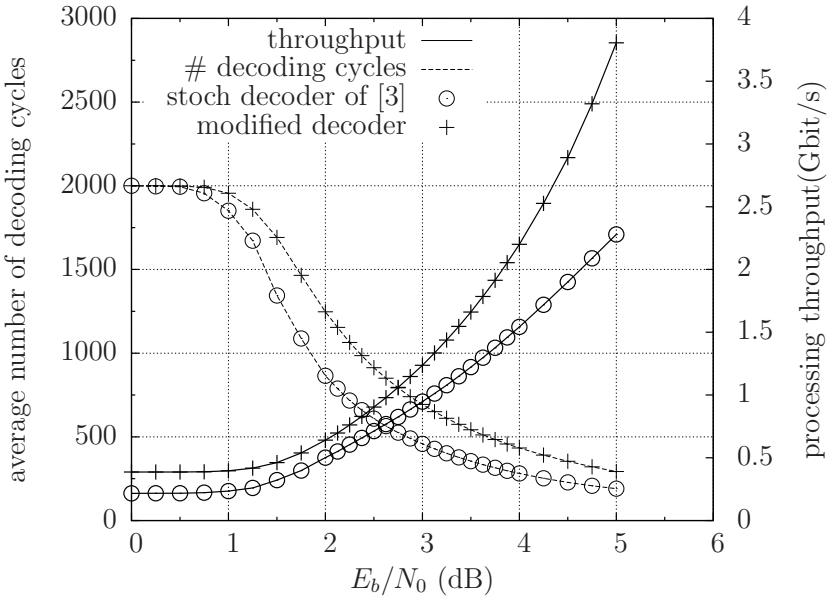


Figure 6.9: Processing throughput (information bits per second) and average number of decoding cycles used for decoding the (1056,528) code, at different E_b/N_0, using 90 nm technology, when the overclocking is adopted as $(T_{\text{clk}}, 3\sigma/\mu) = (1217.5, 0.1)$ and $(718.8, 0.1)$, for the LDPC-SD and the modified LDPC-SD, respectively.

6.4.4 Processing Energy Consumption

Table 6.3 quantifies the processing energy consumption of individual $d_{\text{v}} = 2, 3, 6$ VNs and $d_{\text{c}} = 6, 7$ CNs, within the conventional LDPC-SD of [121] and the modified LDPC-SD. These results were obtained by using SPICE simulations, when employing various T_{clk}. The STMicroelectronics 90 nm technology [107] and the FreePDK 45 nm technology [133] are designed to work at nominal power supply voltages of 1.0 V and 1.1 V, respectively. The same values of the scaled clock period T_{clk} that were selected and justified in Sections 6.2 and 6.3 were used to conduct the simulation for energy consumption estimation. The total LDPC-SD energy consumption was estimated by multiplying the individual CN and VN energy consumptions by the total number of nodes having the corresponding degree in the WiMAX LDPC (1056,528) code. As shown in Table 6.3, neither overclocking, nor the modification of Section 6.3.1 has a significant effect on the energy consumption of the LDPC-SD. Therefore, we can conclude that the attained processing throughput increases of up to 69.4% were achieved without significantly increasing the processing energy consumption.

Table 6.3: Energy consumption from SPICE simulations.

		Energy consumption per 2 clock cycles (pJ)						
		ST90 nm, $\mu = 1.0$ V			FreePDK45 nm, $\mu = 1.1$ V			
		$T_{clk} = 628.3$ ps	718.8 ps	1217.5 ps	760 ps	869.5 ps	1171.2 ps	1472.8 ps
CNs	$d_c = 6$	0.297	0.302	0.303	0.262	0.264	0.275	0.286
	$d_c = 7$	0.325	0.332	0.336	0.282	0.287	0.288	0.288
Conventional VNs	$d_v = 2$	1.47	1.48	1.51	1.52	1.55	1.58	1.59
	$d_v = 3$	3.22	3.34	3.38	3.19	3.22	3.26	3.31
	$d_v = 6$	8.25	8.36	8.58	8.36	8.39	8.45	8.65
Conventional	Total	3.83×10^3	3.90×10^3	3.97×10^3	3.84×10^3	3.88×10^3	3.92×10^3	3.99×10^3
Modified VNs	$d_v = 2$	1.53	1.54	1.54	1.43	1.44	1.44	1.44
	$d_v = 3$	3.10	3.32	3.43	3.10	3.13	3.37	3.38
	$d_v = 6$	8.64	8.88	8.90	8.08	8.10	8.38	8.40
Modified	Total	3.89×10^3	4.03×10^3	4.08×10^3	3.70×10^3	3.72×10^3	3.88×10^3	3.88×10^3

6.5 Conclusions

This chapter proposed a novel design flow for timing-error-tolerant LDPC decoders. Using this design flow, we have demonstrated that LDPC-SDs are capable of exploiting their inherent error correction capability, to correct not only transmission errors, but also timing errors. We have characterized the causes and effects of timing errors within LDPC-SDs by developing a timing error model, which we have validated using SPICE simulations. Drawing upon our findings, we proposed a modified LDPC-SD, having an improved timing error tolerance. In a particular practical scenario, we demonstrated that this modification eliminates the 1 dB performance degradation suffered by the LDPC-SD of [121], which allows the processing throughput to be increased by up to 69.4% in practice, due to the significantly improved tolerance to timing errors. This is achieved without the requirement for additional circuitry or the associated processing energy consumption. Note that since the timing analysis and the timing error model of Sections 6.2 and 6.3 are particular to VNs and CNs having specific degrees and are independent of the interconnectivity of the VNs and CNs, our methodology may be readily applied to other LDPC decoders having different block lengths and coding rates. Additionally, our methodology may be applied to other stochastic decoders that rely on EMs, such as the turbo decoder of [134]. Our future work will fabricate timing-error-tolerant channel decoder ASICs and compare practical measurements of their performance with simulations.

Chapter 7

Conclusions and Future Work

7.1 Summary and Conclusions

In this thesis, we have investigated the fully parallel implementation of Fixed-point LDPC Decoders (LDPC-FDs) and Stochastic LDPC Decoders (LDPC-SDs), as well as characterized their tolerance to timing errors using comprehensive simulations. Based on the simulation results, we have recommended parametrizations for fully parallel LDPC-FDs and LDPC-SDs. These parametrization have been demonstrated to be capable of achieving improved timing error tolerance.

In Chapter 2, the basics of channel codes were reviewed, with a particular focus on LDPC codes. The factor graph representation of LDPC codes was discussed as the basis of the decoding algorithm and the error correction performance. We investigated several iterative LDPC decoding algorithms, including the Sum-Product Algorithm (SPA), Log-Sum-Product Algorithm (Log-SPA) and Min-Sum Algorithm (MSA). The principle of EXtrinsic Information Transfer (EXIT) charts was outlined and used to analyse LDPC decoders using different decoding algorithms. Based on the EXIT charts, the error correction performance of LDPC decoders was evaluated, and the insights of the performance degradation were exhibited and concluded. The error correction performance of the LDPC decoders was also characterized through Bit Error Ratio (BER) simulations, allowing the conclusions based on EXIT charts to be validated.

In Chapter 3, the Two's Complement (TC) number representation and the Fixed-Point (FP) implementation of LDPC decoders were outlined. The EXIT chart analysis of Chapter (2) was extended to consider LDPC-FDs. The FP EXIT chart analysis revealed that the minimum bit width required for LDPC-FDs is 4 bits, having a fraction part of 0 bits. This bit width was demonstrated to allow the LDPC-FD to achieve

an attractive trade-off between their complexity and performance. The advantage of EXIT charts analysis was also characterized. In particularly, factor of 100 times complexity reduction was demonstrated for the case of performing EXIT chart analysis for a particular LDPC code, rather than the conventional method of using BER simulations.

In Chapter 4, previous publications on error-tolerant LDPC decoder design were reviewed and the challenges and the bottlenecks were identified in the conventional LDPC decoder Application-Specific Integrated Circuit (ASIC) design process. Motivated by this, a novel design flow was proposed for timing-error-tolerant LDPC decoders. The fully parallel implementation of the LDPC-FD using 4-bit TC and Base Minus Two (BMT) representations was detailed. A novel structure for high-degree nodes was also proposed for improving the timing error tolerance of the fully parallel LDPC-FD. Timing characteristics analysis was performed on the LDPC-FDs and the resultant error model was detailed. Using the proposed design flow, the error correction performance of the LDPC-FD in the presence of timing errors was characterized using BER simulations. It was demonstrated that the LDPC-FD employing BMT and the proposed node structure is more resilient to timing errors than several benchmarkers. The placement of D-type Flip-Flops (D-FFs) and the associated decoding scheduling was also addressed. By employing D-FFs on the inputs and outputs of every parity-Check Node (CN) and Variable Node (VN), the simultaneous decoding of two independent codewords is permitted for the LDPC-FD. However, the number of D-FFs used may be reduced by up to 30%, if the D-FFs are employed on only the Input/Output (IO) ports of high-degree nodes. The resultant tolerance to timing error was demonstrated to be unaffected by this area reduction, while maintaining the capability for the simultaneous decoding of two independent codewords. The improvements in other aspects of the LDPC decoders were also established. More specifically, the BMT representation can increase the processing throughput by up to 33.3%, reduce the chip area by around 10% and maintain a similar level of energy consumption, compared to the conventional TC representation. The proposed structure for high-degree nodes was demonstrated to significantly improve throughput and timing error tolerance, at the cost of slightly increasing the area and the energy consumption, compared to the conventional node structure. A fabricated ASIC was presented and in the work of our colleagues [109], the resultant measurements were shown to support our simulation results and the design flow proposed in Section 4.2.

In Chapter 5, the challenges associated with the fully parallel implementation of LDPC-FDs were identified. Motivated by this, LDPC-SDs were introduced as a solution. The fully parallel implementation of LDPC-SDs was outlined and the building blocks of stochastic computation that are required in LDPC decoding were detailed.

The latching problems was raised, since it was identified as one of the many main factors that limits the error correction performance of LDPC-SDs. Several techniques used to protect LDPC-SDs from the latching problem were also discussed. BER simulations were conducted to characterize the performance of LDPC-SDs as functions of different parameters. A parametrization for the fully parallel implementation of the LDPC-SD was recommended, which was demonstrated to achieve a desirable trade-off between the implementation complexity and error correction performance.

In Chapter 6, an error model was derived for evaluating the causes and effects of timing errors within the LDPC-SD. With the aid of this error model, the error correction performance of the LDPC-SDs in the presence of timing errors was characterized using BER simulations. Drawing upon our findings, a modification of the LDPC-SD was presented and demonstrated to offer an improved timing error tolerance. In a particular practical scenario, this modification was shown to eliminate the 1 dB performance degradation that is suffered by the conventional LDPC-SD, which increased the processing throughput by up to 69.4% in practice. This was achieved without the requirement for additional circuitry or the associated processing energy consumption. Our methodology was demonstrated to be readily applicable to other LDPC decoders having different block lengths and coding rates, as well as to other stochastic decoders that rely on Edge Memories (EMs). This is because the timing analysis and the timing error model of Sections 6.2 and 6.3 depend only on the degrees of the CNs and VNs and are independent of the interconnectivity.

7.2 Suggestions for Future Work

In analogy to Chapter 4, an LDPC-SD ASIC may be fabricated as the next step following the work of Chapter 6. Measurements may be obtained from this LDPC-SD ASIC to characterize the timing error tolerance, which can be compared with the simulation results demonstrated in Section 6.4. Based on these results, we may adjust the timing characteristics analysis and the error model of Section 6.2.

As demonstrated in Chapter 3, the thorough investigation of LDPC decoding having many design parameters can be efficiently accomplished by the use of EXIT charts. This motivated the extension of the EXIT charts of Chapter 2 to LDPC-SDs. The Bernoulli sequences employed in LDPC-SDs represent probabilities by the fraction of the bits having the value of binary 1 in the entire sequence of bits. Given a number L of successive clock cycles, the probability may be estimated based on the streams of *a priori* or extrinsic bits $\mathbf{x}$ at a CN or a VN, according to $\frac{\sum_i^L x_i}{L}$. Furthermore,

vectors of estimated probabilities can be obtained at all CNs and VNs, then used to quantify the Mutual Information (MI) between the encoded bits and the corresponding probabilities conveyed by the streams of stochastic bits. In this way, the extrinsic MI can be obtained as a function of the *a priori* MI, and visualized as EXIT charts. Note that this quantification of MIs is independent of the structures of CNs and VNs, which may allow the EXIT chart technique to be applied to any other types of LDPC-SDs.

The proposed design flow of Chapters 4 and 6 has been demonstrated to be capable of conducting simulations to characterize the timing error tolerance of LDPC decoders. In order to further improve the accuracy of the simulations, such that they approach the accuracy of practical measurements, more complicated information may be considered in the algorithm level simulation, such as the geographical topology of the logic elements and the clock tree after placing and routing. For example, in our analysis, two VNs having identical degrees are considered to have identical timing characteristics, which may not be an accurate representation of a practical tape-out. For example, when the VNs are located at opposite corners of a large-scale ASIC, they may experience significantly different clock delays and skews, even if they have identical designs and clock tree optimizing techniques. By extending the proposed design flow, we may enhance our error model and the timing analysis for individual CNs and VNs, by taking into consideration this topology information extracted from the transistor design level, using Cadence for example. However, an increase in simulation complexity is inevitable, so it may become challenging to simulate the enhanced error model and more complicated combinations of parameters using conventional C++, as accomplished in Chapters 4 and 6. However, simulators that are used for communication networks may be suited for this task, since they have built-in features for managing the network comprised by a huge number of distributed and parallel entities, as well as for managing the messages propagating through them. In particular, OMNeT++ [135, 136] is a general-purpose platform for simulating such networks. It has been widely used for simulating wired and wireless communication networks [137, 138], while in [139, 140], it has been employed to simulate network-on-chip systems. In order to tackle the massively increased simulation complexity, OMNeT++ may be adopted as the simulation platform in future work.

List of Figures

List of Tables

List of Symbols

General notation

- The bold lower-case letter, such as $\mathbf{a}$, indicates a sequence of message bits.
- The tilde notation, such as that in $\tilde{\mathbf{a}}$, represents LLRs corresponding to the bit sequence $\mathbf{a}$.
- The hat notation, such as that in $\hat{\mathbf{a}}$, represents the corresponding estimation of $\mathbf{a}$ at the receiver.
- The superscript $^{\mathrm{a}}$ represents *a priori*.
- The superscript $^{\mathrm{e}}$ represents extrinsic.
- The superscript $^{\mathrm{p}}$ represents *a posteriori*.
- The superscript ^{T} is used to indicate matrix transpose operation.
- The superscript A^{T} is used to indicate a inverse matrix.
- The subscript $_{a\times b}$ indicates the size of a matrix or vector is a-by-b.
- The superscript $^{+}$ is used to indicate the value of a signal in the forthcoming clock cycle.
- The superscript $^{-}$ is used to indicate the value of a signal in the previous clock cycle.
- the subscript $_{\mathrm{b}}$ is used to indicate a binary number, such as $(000)_{\mathrm{b}}$
- the subscript $_{\mathrm{d}}$ is used to indicate a decimal number, such as $(000)_{\mathrm{d}}$

Special symbols

u: information bit

u: parity bit

$\mathbf{u}$: vector of information bits

c: encoded bit

$\mathbf{c}$: vector of encoded bits

x: BPSK modulated symbol

$\mathbf{x}$: vector of symbols

y : received symbol from channel

$\mathbf{y}$: vector of received symbols

$\mathbf{r}$: vector of repeated bits, associated with VNs

$\mathbf{p}$: vector of permuted bits, associated with CNs

$\mathbf{s}$: syndrome vector

$\mathbf{G}$: generator matrix

$\mathbf{H}$: PCM

$\mathbf{I}$: identity matrix

P: probability

P_A: probability of a message corresponding to port A of a CN or VN

N : codeword length; number of columns in PCM; number of VNs in factor graph

K: block length of information bits;

M: number of rows in PCM; number of CNs in factor graph

N_{edge} : number of edges in factor graph

R: coding rate

η: processing throughput

d : the degree of a node

d_{c}: the degree of c^{th} CN

d_{v}: the degree of v^{th} VN

c: c^{th} CN

v: v^{th} CN

π : interleaver

π^{-1} : de-interleaver corresponding to π

N_0: noise power spectral density

E_{b}: energy per bit

σ_{A}^2 : standard deviation of n_{A}

n_{A}: Gaussian random variable

f_{CN} : operation within CN

f_{VN} : operation within VN

$\boxplus$: boxplus operator

I : mutual information

I_A : *a priori* mutual information

I_E : extrinsic mutual information

W : bit width for FP representation

W_{i} : bit width for integer part

W_{f} : bit width for fraction part

W_{c} : bit width for integer part after clipping

T_{clk} : clock period

t : nominal propagation delay

δ : a multiplier modelling the fluctuation in t

V_{DD} : supply voltage

μ : mean of V_{DD}

σ : standard deviation of V_{DD}

Bibliography

[1] G. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, pp. 64–70, Mar. 2014.

[2] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5g heterogeneous networks," *IEEE Signal Processing Magazine*, vol. 31, pp. 45–55, Nov 2014.

[3] L. Hanzo, T. H. Leiw, and B. L. Yeap, *Turbo Coding, Turbo Equalisation and Space-Time Coding*. Piscataway, NJ: IEEE Press, 2002.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *IEEE International Conference on Communications*, vol. 2, pp. 1064–1070, May 1993.

[5] C. B. Schegel and L. C. Perez, *Trellis and Turbo Coding*. Piscataway, NJ: IEEE Press, 2004.

[6] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, pp. 21 –28, Jan. 1962.

[7] R. Gallager, *Low Density Parity Check Codes*. PhD thesis, 1963.

[8] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, Feb. 2001.

[9] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, pp. 28–41, Feb. 2004.

[10] L. Ping and W. K. Leung, "Decoding low density parity check codes with finite quantization bits," *IEEE Communications Letters*, vol. 4, pp. 62–64, Feb. 2000.

[11] J. Chen, A. Dholakia, E. Eleftherious, M. P. C. Fossorier, and X. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Transactions on Communications*, vol. 53, pp. 1288–1299, Aug. 2005.

[12] D. Oh and K. Parhi, "Optimally quantized offset min-sum algorithm for flexible LDPC codes," in *Asilomar Conference Signals, Systems and Computers*, pp. 1886–1891, Oct. 2008.

[13] G. Masera, F. Quaglio, and F. Vacca, "Finite precision implementation of LDPC decoders," *IEE Proceedings-Communications*, vol. 152, pp. 1098–1102, Dec. 2005.

[14] T. Zhang, Z. wang, and K. Parhi, "On finite precision implementation of low density parity check codes decoder," in *IEEE International Symposium on Circuits and Systems, Sydney, Australia*, vol. 4, pp. 202–205, May 2001.

[15] J. Zhao, F. Zarkeshvari, and A. Banihashemi, "On implementation of min-sum algorithm and its modification for decoding low-density parity-check (LDPC) codes," *IEEE Transactions on Communications*, vol. 53, pp. 549–554, Apr. 2005.

[16] D. Oh and K. Parhi, "Min-sum decoder architecture with reduced word length for LDPC codes," *IEEE Transactions on Circuits and Systems I*, vol. 57, pp. 105–115, Jan. 2010.

[17] G. Masera, F. Quaglio, and F. Vacca, "Implementation of a flexible LDPC decoder," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54, p. 542, June 2007.

[18] V. Sorokine, F. R. Kschischang, and S. Pasupathy, "Gallager codes for CDMA applications .II. implementations, complexity, and system capacity," *IEEE Transactions on Communications*, vol. 48, p. 1818, Nov. 2000.

[19] H. Zhong and T. Zhang, "Joint code-encoder-decoder design for LDPC coding system VLSI implementation," in *International Symposium on Circuits and Systems*, vol. 2, p. II, May 2004.

[20] H. Zhong and T. Zhang, "Design of VLSI implementation-oriented LDPC codes," in *IEEE 58th Vehicular Technology Conference*, vol. 1, p. 670, Oct. 2003.

[21] M. M. Mansour and N. R. Shanbhag, "Memory-efficient turbo decoder architectures for LDPC codes," in *IEEE Workshop on Signal Processing Systems*, p. 159, Oct. 2002.

[22] M. M. Mansour and N. Shanbhag, "Architecture-aware low-density parity-check codes," in *Proceedings of the 2003 International Symposium on Circuits and Systems*, vol. 2, pp. II–57, May 20093.

[23] G. Lechner, J. Sayir, and M. Rupp, "Efficient dsp implementation of an LDPC decoder," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. iv–665, May 2004.

[24] Y. Zhu, Y. Chen, D. Hocevar, and M. Goel, "A reduced-complexity, scalable implementation of low density parity check (LDPC) decoder," in *IEEE Workshop on Signal Processing Systems Design and Implementation*, p. 83, Oct. 2006.

[25] Z. Wang, H. Suzuki, and K. K. Parhi, "VLSI implementation issues of TURBO decoder design for wireless applications," in *1999 IEEE Workshop on Signal Processing Systems*, p. 503, 1999.

[26] Y. Zhu, Y. Chen, D. Hocevar, and M. Goel, "A reduced-complexity, scalable implementation of low density parity check (LDPC) decoder," in *IEEE 8th International Conference on ASIC*, p. 501, Oct. 2009.

[27] Y. S. Park, Y. Tao, and Z. Zhang, "A 1.15Gb/s fully parallel nonbinary LDPC decoder with fine-grained dynamic clock gating," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International*, pp. 422–423, Feb 2013.

[28] Y. S. Park, Y. Tao, and Z. Zhang, "A fully parallel nonbinary LDPC decoder with fine-grained dynamic clock gating," *IEEE Journal of Solid-State Circuits*, vol. 50, pp. 464–475, Feb. 2015.

[29] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular $(2,d_c)$-LDPC codes over GF(q) using their binary images," *IEEE Transactions on Communications*, vol. 56, pp. 1626–1635, Oct. 2008.

[30] J. Kang, Q. Huang, L. Zhang, B. Zhou, and S. Lin, "Quasi-cyclic LDPC codes: an algebraic construction," *IEEE Transactions on Communications*, vol. 58, pp. 1383–1396, May 2010.

[31] R. Ahmadi and F. Najm, "Timing analysis in presence of power supply and ground voltage variations," in *International Conference on Computer Aided Design*, pp. 176–183, Nov 2003.

[32] M. Alioto, G. Palumbo, and M. Pennisi, "Understanding the effect of process variations on the delay of static and domino logic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, pp. 697–710, May 2010.

[33] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation," in *Proceedings of 36th Annual IEEE/ACM International Symposium on Microarchitecture.*, pp. 7–18, Dec 2003.

[34] M. Gupta, J. Oatley, R. Joseph, G.-Y. Wei, and D. Brooks, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *Design, Automation Test in Europe Conference Exhibition*, pp. 1–6, April 2007.

[35] S. Beer, J. Cox, R. Ginosar, T. Chaney, and D. Zar, "Variability in multistage synchronizers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, pp. 2957–2969, Dec 2015.

[36] N. Ahmed, M. Tehranipoor, and V. Jayaram, "A novel framework for faster-than-at-speed delay test considering IR-drop effects," in *IEEE/ACM International Conference on Computer-Aided Design*, pp. 198–203, Nov 2006.

[37] S. Ghosh and K. Roy, "Parameter variation tolerance and error resiliency: New design paradigm for the nanoscale era," *Proceedings of the IEEE*, vol. 98, pp. 1718–1751, Nov. 2010.

[38] B. Shim and N. Shanbhag, "Energy-efficient soft error-tolerant digital signal processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 336–348, April 2006.

[39] R. Abdallah and N. Shanbhag, "Error-resilient low-power Viterbi decoder architectures," *IEEE Transactions on Signal Processing*, vol. 57, pp. 4906–4917, May 2009.

[40] V. C. Gaudet, "Low-power LDPC decoding by exploiting the fault-tolerance of the sum-product algorithm," in *Contemporary Mathematics*, vol. 523, pp. 165–171, Jun. 2010.

[41] M. Alles, T. Brack, and N. Wehn, "A reliability-aware LDPC code decoding algorithm," in *IEEE 65th Vehicular Technology Conference (VTC), Dublin, Ireland*, pp. 1544–1548, Apr. 2007.

[42] M. May, M. Alles, and N. Wehn, "A case study in reliability-aware design: A resilient LDPC code decoder," in *Design, Automation and Test in Europe, Munich, Germany*, pp. 456 –461, Mar. 2008.

[43] R. Uppu, R. Uppu, A. Singh, and A. Chatterjee, "A high throughput multiplier design exploiting input based statistical distribution in completion delays," in *26th International Conference on VLSI Design and 12th International Conference on Embedded Systems (VLSID) , Hyatt Regency, Pune, India*, pp. 109–114, Jan. 2013.

[44] I. Perez-Andrade, X. Zuo, R. Maunder, B. Al-Hashimi, and L. Hanzo, "Analysis of voltage- and clock-scaling-induced timing errors in stochastic LDPC decoders," in *IEEE Wireless Communications and Networking Conference (WCNC), Shanghai, China*, pp. 4293–4298, Apr. 2013.

[45] L. W. Nagel and D. Pederson, "SPICE (simulation program with integrated circuit emphasis)," Tech. Rep. UCB/ERL M382, EECS Department, University of California, Berkeley, Apr 1973.

[46] Synopsys, Inc., "HSPICE: The gold standard for accurate circuit simulation," 2014. [Online]. Available: http://www.synopsys.com/Tools/Verification/AMSVerification/CircuitSimulation/HSPICE/Documents/hspice_ds.pdf.

[47] Synopsys, Inc., *HSPICE user guide: simulation and analysis. (Version B-2008.09)*, 2008.

[48] Cadence Design Systems, Inc. [Online] http://www.cadence.com/products/pages/default.aspx.

[49] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, 2001.

[50] J. Hagenauer, "The EXIT chart - introduction to extrinsic information transfer in iterative processing," in *12th European Signal Processing Conference*, pp. 1541–1548, Sep. 2004.

[51] B. Sklar, *Digital Communications Fundamentals and Applications*, ch. 6,7,8, pp. 304–510. Prentice Hall, Englewood Cliffs: Prentice-Hall International Editions, 1988.

[52] C. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, July 1948.

[53] P. Elias, "Coding for noisy channels," *IRE Convention Record*, vol. 3, no. 4, pp. 37–46, 1955.

[54] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3(1), pp. 68–79, Mar. 1960.

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

[56] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics (SIAM)*, vol. 8(2), pp. 300–304, 1960.

[57] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260–269, April 1967.

[58] J. Hagenauer and P. Hoeher, "A viterbi algorithm with soft-decision outputs and its applications," in *IEEE Global Telecommunications Conference and Exhibition 'Communications Technology for the 1990s and Beyond' (GLOBECOM)*, vol. 3, (Dallas, USA), pp. 1680–1686, Nov. 1989.

[59] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, pp. 533–547, Sep. 1981.

[60] V. Sorokine, F. R. Kschischang, and S. Pasupathy, "Gallager codes for CDMA applications .I. generalizations, constructions, and performance bounds," *IEEE Transactions on Communications*, vol. 48, p. 1660, Oct. 2000.

[61] H. Behairy and S.-C. Chang, "Parallel concatenated Gallager codes for CDMA applications," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 2, p. 1002, 2001.

[62] *IEEE Std 802.16e-2005 and IEEE Std 802.16-2004/Cor 1-2005 (Amendment and Corrigendum to IEEE Std 802.16-2004), IEEE Std 802.16e-2005*, 2006.

[63] IEEE 802.11n/ac. The IEEE 802.11 Working Group. [Online]. Available: http://www.ieee802.org/11/.

[64] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *IEEE Global Telecommunications Conference, San Antonio, USA*, vol. 2, pp. 1036–1036E, Nov. 2001.

[65] IEEE 802.16e. The IEEE 802.16 Working Group. [Online]. Available: http://www.ieee802.org/16/.

[66] E. Eleftheriou, T. Mittelholzer, and A. Dholakia, "Reduced-complexity decoding algorithm for low-density parity-check codes," *Electronics Letters*, vol. 37, pp. 102–104, Jan 2001.

[67] M. P. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity-check codes based on belief propagation," *IEEE Transactions on Communications*, vol. 47, pp. 673–680, Mar. 1999.

[68] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, 1999.

[69] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 47, pp. 429–445, Aug. 2002.

[70] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log-domain," in *IEEE International Conference on Communications*, vol. 2, p. 1009, 1995.

[71] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, pp. 599–618, Feb. 2001.

[72] N. Bonello, S. Chen, and L. Hanzo, "Low-density parity-check codes and their rateless relatives," *IEEE Communications Surveys & Tutorials*, vol. P, p. 1, May. 2010.

[73] D. J. C. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457–458, 1997.

[74] Y. Wu and B. D. Woener, "The influence of quantization and fixed point arithmetic upon the BER performance of turbo codes," in *IEEE Vehicular Technology Conference*, vol. 2, p. 1683, Jul. 1999.

[75] X. Zuo, R. Maunder, and L. Hanzo, "Design of fixed-point processing based LDPC codes using EXIT charts," in *2011 IEEE Vehicular Technology Conference (VTC Fall), San Francisco, CA, USA*, pp. 1–5, Sep. 2011.

[76] G. Montorsi and S. Benedetto, "Design of fixed-point iterative decoders for concatenated codes with interleavers," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 871–882, May 2001.

[77] J. Lee and R. Blahut, "A note on the analysis of finite length turbo decoding," in *IEEE International Symposium on Information Theory*, (Lausanne, Switzerland), p. 83, June 2002.

[78] J. Lee and R. Blahut, "Lower bound on BER of finite-length turbo codes based on exit characteristics," *IEEE ASSP MagazineCommunications Letters*, vol. 8, pp. 238–240, Apr. 2004.

[79] J. Lee and R. Blahut, "Convergence analysis and BER performance of finite-length turbo codes," *IEEE Transactions on Communications*, vol. 55, pp. 1033–1043, May 2007.

[80] E. Boutillon, C. Douilard, and G. Montorsi, "Iterative decoding of concatenated convolutional codes: Implementation issues," in *Proceedings of the IEEE*, vol. 95, pp. 1201–1227, Jun. 2007.

[81] The IEEE 802.16 Working Group [Online]. http://www.ieee802.org/16/.

[82] Z. Pi and F. Khan, "An introduction to millimeter-wave mobile broadband systems," *IEEE Communications Magazine*, vol. 49, pp. 101–107, June 2011.

[83] E. Larsson, O. Edfors, F. Tufvesson, and T. Marzetta, "Massive MIMO for next generation wireless systems," *IEEE Communications Magazine*, vol. 52, pp. 186–195, February 2014.

[84] M. Mansour and N. Shanbhag, "High-throughput LDPC decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pp. 976–996, 2003.

[85] L. Varshney, "Performance of LDPC codes under faulty iterative decoding," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4427–4444, 2011.

[86] S. Yazdi, C.-H. Huang, and L. Dolecek, "Optimal design of a Gallager B noisy decoder for irregular LDPC codes," *IEEE Communications Letters*, vol. 16, pp. 2052–2055, Dec. 2012.

[87] S. Tabatabaei Yazdi, H. Cho, and L. Dolecek, "Gallager B decoder on noisy hardware," *IEEE Transactions on Communications*, vol. 61, pp. 1660–1673, May 2013.

[88] C.-H. Huang, Y. Li, and L. Dolecek, "Gallager B LDPC decoder with transient and permanent errors," *IEEE Transactions on Communications*, vol. 62, pp. 15–28, Jan. 2014.

[89] K. Bowman, J. Tschanz, N. S. Kim, J. Lee, C. Wilkerson, S. Lu, T. Karnik, and V. De, "Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance," *IEEE Journal of Solid-State Circuits*, vol. 44, pp. 49–63, Jan 2009.

[90] A. Martin and M. Nystrom, "Asynchronous techniques for system-on-chip design," *Proceedings of the IEEE*, vol. 94, pp. 1089–1120, June 2006.

[91] R. Ginosar, "Metastability and synchronizers: A tutorial," *IEEE Design Test of Computers*, vol. 28, pp. 23–35, Sept 2011.

[92] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE transcations on Information Theory*, vol. 47, pp. 619–637, Feb. 2001.

[93] Z. Cui, Z. Wang, and X. Zhang, "Reduced-complexity column-layered decoding and implementation for LDPC codes," *IET Communications*, vol. 5, pp. 2177–2186, October 2011.

[94] S. Papaharalabos, P. Sweeney, B. Evans, P. Mathiopoulos, G. Albertazzi, A. Vanelli-Coralli, and G. Corazza, "Modified sum-product algorithms for decoding low-density parity-check codes," *IET Communications*, vol. 1, pp. 294–300, June 2007.

[95] X. Chen, Q. Huang, S. Lin, and V. Akella, "FPGA-based low-complexity high-throughput tri-mode decoder for quasi-cyclic LDPC codes," in *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pp. 600–606, Sept 2009.

[96] H. Ding, S. Yang, W. Luo, and M. Dong, "Design and implementation for high speed LDPC decoder with layered decoding," in *WRI International Conference on Communications and Mobile Computing*, vol. 1, pp. 156–160, Jan 2009.

[97] A. Darabiha, A. Carusone, and F. Kschischang, "A bit-serial approximate min-sum LDPC decoder and FPGA implementation," in *Proceedings of IEEE International Symposium on Circuits and Systems*, (Kos, Greece), pp. 149 – 152, May 2006.

[98] T. Zhang and K. K. Parhi, "Joint (3,k)-regular LDPC code and decoder/encoder design," *IEEE Transactions on Signal Processing*, vol. 52, p. 1065, April 2004.

[99] D. E. Hocevar, "LDPC code construction with flexible hardware implementation," in *IEEE International Conference on Communications*, vol. 4, p. 2708, May 2003.

[100] E. Yeo, B. Nikolic, and V. Anantharam, "Architectures and implementations of low-density parity check decoding algorithms," in *Midwest Symposium on Circuits and Systems*, vol. 3, pp. III–437, Aug. 2002.

[101] T. Zhang and K. K. Parhi, "VLSI implementation-oriented (3,k)-regular low-density parity check codes," in *IEEE Workshop on Signal Processing Systems*, p. 25, Sep. 2001.

[102] A. Darabiha, *VLSI Architecture for Multi-Gbps Low-Density Parity-Check Decoders*. PhD thesis, University of Toronto, Toronto, ON, Canada, 2008.

[103] F. Martorell, M. Pons, A. Rubio, and F. Moll, "Error probability in synchronous digital circuits due to power supply noise," in *International Conference on Design Technology of Integrated Systems (DTIS) in Nanoscale Era*, pp. 170–175, Sep. 2007.

[104] S. Pant, D. Blaauw, V. Zolotov, S. Sundareswaran, and R. Panda, "A stochastic approach to power grid analysis," in *Proceedings of the 41st Design Automation Conference*, (New York, New York, USA), pp. 171–176, ACM Press, 2004.

[105] M. Nourani and A. Radhakrishnan, "Power-supply noise in SoCs: ATPG, estimation and control," *IEEE International Conference on Test, 2005.*, pp. 507–516, 2005.

[106] IBM, "Foundry technologies 130-nm CMOS and RF CMOS," 2003.

[107] STMicroelectronics, "CORE90 GP SVT 1.00V," *User manual & Databook*, May 2006.

[108] S. Zhong, S. Khursheed, B. Al-Hashimi, and W. Zhao, "Efficient variation-aware delay fault simulation methodology for resistive open and bridge defects,"

IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 33, pp. 798–810, May 2014.

[109] X. Zuo, S. Zhong, K. Li, I. Perez-Andrade, R. Maunder, B. Al-Hashimi, and L. Hanzo, "High throughput timing-error-induced VLSI implementation of LDPC decoding using the base-minus-two fixed-point number representation," *IEEE Journal of Solid-State Circuits*. In preparation.

[110] A. Blanksby and C. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 404–412, Mar. 2002.

[111] A. Darabiha, A. Carusone, and F. Kschischang, "Block-interlaced LDPC decoders with reduced interconnect complexity," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, pp. 74–78, Jan. 2008.

[112] A. Darabiha, A. Carusone, and F. Kschischang, "A 3.3-Gbps bit-serial block-interlaced min-sum LDPC decoder in 0.13-μm CMOS," in *IEEE Custom Integrated Circuits Conference*, (San Jose, USA), pp. 459–462, Sep. 2007.

[113] H. Zhong and T. Zhang, "Block-LDPC: a practical LDPC coding system design approach," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, p. 766, April 2005.

[114] F. Guo, *Low density parity check coding*. PhD thesis, University of Southampton, UK, 2005.

[115] K. Cushon, C. Leroux, S. Hemati, S. Mannor, and W. Gross, "A min-sum iterative decoder based on pulsewidth message encoding," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, pp. 893 –897, Nov. 2010.

[116] T. Mohsenin, D. Truong, and B. Baas, "A low-complexity message-passing algorithm for reduced routing congestion in LDPC decoders," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, pp. 1048–1061, May 2010.

[117] N. Onizawa, T. Hanyu, and V. Gaudet, "Design of high-throughput fully parallel LDPC decoders based on wire partitioning," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, pp. 482–489, March 2010.

[118] C.-C. Cheng, J.-D. Yang, H.-C. Lee, C.-H. Yang, and Y.-L. Ueng, "A fully parallel LDPC decoder architecture using probabilistic min-sum algorithm for high-throughput applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, pp. 2738–2746, Sep. 2014.

[119] V. Gaudet and A. Rapley, "Iterative decoding using stochastic computation," *Electronics Letters*, vol. 39, pp. 299–301, Feb. 2003.

[120] S. Tehrani, W. Gross, and S. Mannor, "Stochastic decoding of LDPC codes," *IEEE Communications Letters*, vol. 10, pp. 716–718, Oct. 2006.

[121] S. S. Tehrani, S. Mannor, and W. J. Gross, "Fully parallel stochastic LDPC decoders," *IEEE Transactions on Signal Processing*, vol. 56, pp. 5692–5703, Nov. 2008.

[122] S. Tehrani, A. Naderi, G.-A. Kamendje, S. Hemati, S. Mannor, and W. Gross, "Majority-based tracking forecast memories for stochastic LDPC decoding," *IEEE Transactions on Signal Processing*, vol. 58, no. 9, pp. 4883–4896, 2010.

[123] A. Ciobanu, S. Hemati, and W. Gross, "Adaptive multiset stochastic decoding of non-binary LDPC codes," *IEEE Transactions on Signal Processing*, vol. 61, pp. 4100–4113, Aug. 2013.

[124] J. P. H. A. Alaghi, "Survey of stochastic computing," *ACM Transactions on Embedded Computing Systems*, 2012.

[125] X. Zuo, I. Perez-Andrade, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "Improving the tolerance of stochastic LDPC decoders to overclocking-induced timing errors: A tutorial and a design example," *IEEE Access*, vol. 4, pp. 1607–1629, 2016.

[126] S. Tehrani, S. Mannor, and W. Gross, "Survey of stochastic computation on factor graphs," in *37th International Symposium on Multiple-Valued Logic (ISMVL)*, p. 54, May 2007.

[127] W. J. Gross, V. C. Gaudet, and A. Milner, "Stochastic implementation of LDPC decoders," in *Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers*, pp. 713–717, Oct. 2005.

[128] B. Gaines, *Advances in Information Systems Science*, ch. 2. New York: Plenum, 1969.

[129] N. R. Shanbhag, R. A. Abdallah, R. Kumar, and D. L. Jones, "Stochastic computation," in *47th ACM/IEEE Design Automation Conference (DAC)*, pp. 859 –864, June 2010.

[130] X. Qi, S. Lo, Y. Luo, A. Gyure, M. Shahram, and K. Singhal, "Simulation and analysis of inductive impact on VLSI interconnects in the presence of process

variations," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 309–312, 2005.

[131] T. Enami, S. Ninomiya, and M. Hashimoto, "Statistical timing analysis considering spatially and temporally correlated dynamic power supply noise," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 4, pp. 541–553, 2009.

[132] STMicroelectronics, *User Manual & Databook*, May 2006.

[133] A variation-aware 45nm design flow for the Semiconductor Research Corporation, jointly developed by Oklahoma State University and North Carolina State University [Online]. Available: http://vlsiarch.ecen.okstate.edu/flows/OSUFreePDK45/.

[134] Q. T. Dong, M. Arzel, C. Jego, and W. J. Gross, "Stochastic decoding of turbo codes," *IEEE Transactions on Signal Processing*, vol. 58, pp. 6421–6425, Dec. 2010.

[135] A. Varga, "OMNeT++," in *Modeling and Tools for Network Simulation*, pp. 35–59, Springer, 2010.

[136] A. Varga *et al.*, "The OMNeT++ discrete event simulation system," in *Proceedings of the European simulation multiconference (ESM)*, vol. 9, p. 65, sn, 2001.

[137] J. Zuo, S. X. Ng, and L. Hanzo, "Fuzzy logic aided dynamic source routing in cross-layer operation assisted Ad Hoc networks," in *IEEE Vehicular Technology Conference Fall (VTC 2010-Fall)*, pp. 1–5, Sept. 2010.

[138] Y. He, W. Xu, and X. Lin, "A stable routing protocol for highway mobility over vehicular Ad-Hoc networks," in *IEEE Vehicular Technology Conference (VTC Spring)*, pp. 1–5, May 2015.

[139] Y. Ben-Itzhak, E. Zahavi, I. Cidon, and A. Kolodny, "NoCs simulation framework for OMNeT++," in *IEEE/ACM International Symposium on Networks on Chip (NoCS)*, pp. 265–266, May 2011.

[140] A. Mansour and J. Gotze, "An OMNeT++ based network-on-chip simulator for embedded systems," in *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 364–367, Dec 2012.

A. Alaghi, J. P. Hayes 87

Abdallah, R.A. 2, 56

Abdallah, Rami A. 108

Ahmadi, R. 2, 53, 56, 109, 111, 114

Ahmed, N. 2, 53, 56, 109, 111, 114

Akella, V. 59

Al-Hashimi, B.M. 2, 53, 70, 83, 108, 109, 120, 132

Albertazzi, G. 57

Alioto, M. 2, 53, 56, 109, 111, 114

Alles, M. 2, 53, 108, 109

Anantharam, Venkat 61, 86

Arnold, D.-M. 13, 14, 18, 19, 34, 57, 61, 92

Arzel, Matthieu 130

Austin, T. 2, 53, 56, 109, 111, 114

Baas, B.M. 86, 87

Banihashemi, A.H. 2, 14, 34, 35, 37, 45, 61

Barbarossa, S. 1

Beer, S. 2, 53, 54, 56, 59, 60, 109, 111, 114

Behairy, H. 12

Ben-Itzhak, Y. 134

Benedetto, S. 37

Berrou, C. 1, 7

Blaauw, D 66, 69

Blahut, R.E. 41

Blanksby, A.J. 86

Bonello, Nicholas 25, 57

Bose, R. C. 7

Boutillon, E. 45

Bowman, K.A. 54, 59, 60

Brack, T. 2, 53, 108, 109

Brooks, D.M. 2, 53, 56, 109, 111, 114

Carusone, A.C. 59, 86

Chaney, T. 2, 53, 54, 56, 59, 60, 109, 111, 114

Chang, Shih-Chun 12

Chatterjee, A. 2, 56

Chen, J. 2, 14, 16–19, 34, 35, 37, 39, 45, 57, 61

Chen, Sheng 25, 57

Chen, Xiaoheng 59

Chen, Yanni 2, 12, 35, 61, 86

Cheng, Chung-Chao 86, 87

Cho, Hyungmin 53, 108, 109

Cidon, I. 134

Ciobanu, A. 87

Corazza, G.E. 57

Cox, J. 2, 53, 54, 56, 59, 60, 109, 111, 114

Cui, Z. 57

Cushon, K. 86

Darabiha, A. 59, 61, 86

Das, S. 2, 53, 56, 109, 111, 114

De, V.K. 54, 59, 60

Declercq, D. 2

Dholakia, A. 2, 13, 14, 16–19, 34, 35, 37, 39, 45, 57, 61, 92

Di Lorenzo, P. 1

Ding, Hong 59

Dolecek, L. 53, 108, 109

Dolecek, Lara 53, 108, 109

Dong, Mingke 59

Dong, Quang Trung 130

Douilard, C. 45

Edfors, O. 52

Eleftheriou, E. 13, 14, 16–19, 34, 39, 57, 61, 92

Eleftherious, E. 2, 14, 16–19, 34, 35, 37, 39, 45, 57, 61

Elias, Peter 7

Enami, T. 114

Ernst, D. 2, 53, 56, 109, 111, 114

Evans, B.G. 57

Fettweis, G.P. 1

Flautner, K. 2, 53, 56, 109, 111, 114

Fossorier, M. 2

Fossorier, M. P. C. 2, 14, 16–19, 34, 35, 37, 39, 45, 57, 61

Fossorier, Marc P.C. 14, 16, 18, 19, 34, 39, 57, 61

Frey, B.J. 1, 12–14, 17, 19, 57, 58, 88

Gaines, B. 108

Gallager, R. 1, 7, 12, 27, 34, 57, 86

Gaudet, V. C. 2, 53, 94, 104, 105, 108, 109, 138

Gaudet, V.C. 86, 87, 94, 138

Ghosh, S. 2, 53, 56, 109, 111, 114

Ginosar, R. 2, 53, 54, 56, 59, 60, 109, 111, 114

Glavieux, A. 1, 7

Goel, M. 2, 12, 35, 61, 86

Gotze, J. 134

Gross, W. J. 87, 94–100, 104, 105, 109, 112, 115, 120–124, 126, 128–130, 138, 139, 141

Gross, Warren J. 130

Gross, W.J. 86, 87, 94, 96–99, 109, 113, 138

Guo, Feng 86

Gupta, M.S. 2, 53, 56, 109, 111, 114

Gyure, A. 114

Hagenauer, J. 3, 7, 19, 21–23, 25, 34, 37, 40, 41, 57

Hanyu, T. 86, 87

Hanzo, L. 2, 36, 53, 83, 108, 109, 126, 132, 134

Hanzo, Lajos 1, 7, 25, 57

Hashimoto, M. 114

He, Yang 134

Hemati, S. 86, 87, 109, 113

Hocevar, D. 2, 12, 35, 61, 86

Hocevar, D. E. 61, 86

Hocquenghem, Alexis 7

Hoeher, P. 7, 19

Howland, C.J. 86

Hu, X. 2, 14, 16–19, 34, 35, 37, 39, 45, 57, 61

Hu, Xiao-Yu 13, 14, 18, 19, 34, 57, 61, 92

Huang, Chu-Hsiang 53, 108, 109

Huang, Qin 2, 59

IBM 68, 72, 73

Imai, H. 14, 16, 18, 19, 34, 39, 57, 61

Jayaram, V. 2, 53, 56, 109, 111, 114

Jego, Christophe 130

Jones, Douglas L. 108

Joseph, R. 2, 53, 56, 109, 111, 114

Kamendje, G.-A. 87, 109, 113

Kang, Jingyu 2

Karnik, T. 54, 59, 60

Khan, F. 52

Khursheed, S. 70, 120

Kim, Nam Sung 2, 53, 54, 56, 59, 60, 109, 111, 114

Kolodny, A. 134

Kschischang, F. R. 2, 12, 35

Kschischang, F.R. 1, 12–14, 17, 19, 57–59, 86, 88

Kumar, Rakesh 108

Larsson, E. 52

Lechner, G. 2, 35

Lee, Huang-Chang 86, 87

Lee, J.C. 54, 59, 60

Lee, J.W. 41

Leiw, T. H. 1, 7

Leroux, C. 86

Leung, W. K. 2, 35, 45, 61, 63, 86

Li, Ke 83, 132

Li, Yao 53, 108, 109

Lin, Shu 2, 59

Lin, Xuehong 134

Lo, S.C. 114

Loeliger, H.-A. 1, 12–14, 17, 19, 57, 58, 88

Loeliger, Hans-Andrea 1, 12, 13, 57

Lu, S.L. 54, 59, 60

Luo, Wu 59

Luo, Yansheng 114

MacKay, D. J. C. 34

Mannor, S. 86, 87, 94–100, 104, 105, 109, 112, 113, 115, 120–124, 126, 128–130, 138,

139, 141

Mansour, A. 134

Mansour, M. M. 2, 35

Mansour, M.M. 52, 54

Martin, A.J. 54, 59, 60

Martorell, Ferran 66, 69, 80, 114

Marzetta, T. 52

Masera, G. 2, 35, 45, 61, 86

Mathiopoulos, P.T. 57

Maunder, R.G. 2, 36, 53, 83, 108, 109, 126, 132

May, M. 2, 53, 108, 109

Mihaljevic, M. 14, 16, 18, 19, 34, 39, 57, 61

Milner, A. 94, 104, 105, 138

Mittelholzer, T. 14, 16–19, 34, 39, 57, 61

Mohsenin, T. 86, 87

Moll, Francesc 66, 69, 80, 114

Montorsi, G. 37, 45

Mudge, T. 2, 53, 56, 109, 111, 114

Naderi, A. 87, 109, 113

Najm, F.N. 2, 53, 56, 109, 111, 114

Neal, R. M. 34

Ng, Soon Xin 134

Nikolic, Borivoje 61, 86

Ninomiya, S. 114

Nourani, M. 66, 68, 69

Nystrom, M. 54, 59, 60

Oatley, J.L. 2, 53, 56, 109, 111, 114

Offer, E. 19, 34, 37, 57

Oh, Daesun 2, 34, 35, 45, 61, 63, 86

Onizawa, N. 86, 87

Palumbo, G. 2, 53, 56, 109, 111, 114

Panda, R. 66, 69

Pant, S. 2, 53, 56, 66, 69, 109, 111, 114

Papaharalabos, S. 57

Papke, L. 19, 34, 37, 57

Parhi, K. K. 2, 35, 61, 86

Parhi, Keshab K. 61, 86

Parhi, K.K. 2, 34, 35, 45, 61, 63, 86

Park, Youn Sung 2, 86

Pasupathy, S. 2, 12, 35

Pennisi, M. 2, 53, 56, 109, 111, 114

Perez-Andrade, I. 2, 53, 83, 108, 109, 132

Perez, L. C. 1, 7

Pham, T. 2, 53, 56, 109, 111, 114

Pi, Zhouyue 52

Ping, Li 2, 35, 45, 61, 63, 86

Pons, Marc 66, 69, 80, 114

Poulliat, C. 2

Qi, Xiaoning 114

Quaglio, F. 2, 35, 45, 61, 86

Radhakrishnan, A. 66, 68, 69

Rao, R. 2, 53, 56, 109, 111, 114

Rapley, A.C. 87, 94, 138

Ray-Chaudhuri, D. K. 7

Reed, Irving S. 7

Richardson, T.J. 21, 54, 57, 61, 96

Robertson, P. 19

Roy, K. 2, 53, 56, 109, 111, 114

Rubio, Antonio 66, 69, 80, 114

Rupp, M. 2, 35

Sardellitti, S. 1

Sayir, J. 2, 35

Schegel, C. B. 1, 7

Shahram, M. 114

Shanbhag, N. 2, 35

Shanbhag, N. R. 2, 35

Shanbhag, Naresh R. 108

Shanbhag, N.R. 2, 52, 54, 56

Shannon, C.E. 6

Shim, Byonghyo 2, 56

Shokrollahi, M.A. 54, 61, 96

Singh, A.D. 2, 56

Singhal, Kishore 114

Sklar, Bernard 6, 11

Solomon, Gustave 7

Sorokine, V. 2, 12, 35

STMicroelectronics 68, 72, 111, 113, 123, 129

Sundareswaran, S. 66, 69

Suzuki, H. 2, 35

Sweeney, P. 57

Synopsys 70, 120

Tabatabaei Yazdi, S.M.Sadegh 53, 108, 109

Tanner, R. 12, 13, 57

Tao, Yaoyu 2, 86

Tehrani, S. Sharifi 87, 95–100, 104, 105, 109, 112, 115, 120–124, 126, 128–130, 138, 139, 141

Tehrani, S.S. 87, 94, 96–99, 109, 113, 138

Tehranipoor, M. 2, 53, 56, 109, 111, 114

ten Brink, Stephan 3, 21–23, 25, 40, 41

Thitimajshima, P. 1, 7

Truong, D.N. 86, 87

Tschanz, J.W. 54, 59, 60

Tufvesson, F. 52

Ueng, Yeong-Luh 86, 87

Uppu, R.K. 2, 56

Uppu, R.T. 2, 56

Urbanke, R.L. 21, 54, 57, 61, 96

Vacca, F. 2, 35, 45, 61, 86

Vanelli-Coralli, A. 57

Varga, András 134

Varshney, L.R. 53, 108, 109

Villebrun, E. 19

Viterbi, A.J. 7

Wang, Z. 57

Wang, Zhongfeng 2, 35

Wehn, N. 2, 53, 108, 109

Wei, Gu-Yeon 2, 53, 56, 109, 111, 114

Wilkerson, C.B. 54, 59, 60

Woener, Brian D. 35

Wu, Yufei 35

Xu, Wenjun 134

Yang, Chia-Hsiang 86, 87

Yang, Jeng-Da 86, 87

Yang, Shuai 59

Yazdi, S.M.S.T. 53, 108, 109

Yeap, B. L. 1, 7

Yeo, Engling 61, 86

Zahavi, E. 134

Zar, D.M. 2, 53, 54, 56, 59, 60, 109, 111, 114

Zarkeshvari, F. 2, 14, 34, 35, 37, 45, 61

Zhang, Li 2

Zhang, Tang 2, 35, 45, 61, 86

Zhang, Tong 2, 35, 61, 86

Zhang, X. 57

Zhang, Zhengya 2, 86

Zhao, Jianguang 2, 14, 34, 35, 37, 45, 61

Zhao, Wei 70, 120

Zhong, Hao 2, 35, 61, 86

Zhong, Shida 70, 83, 120, 132

Zhou, Bo 2

Zhu, Yuming 2, 12, 35, 61, 86

Ziesler, C. 2, 53, 56, 109, 111, 114

Zolotov, V. 66, 69

Zuo, Jing 134

Zuo, Xin 2, 36, 53, 83, 108, 109, 126, 132