

AMERICAN UNIVERSITY OF BEIRUT

RATE-COMPATIBLE CODING: AN INTER-FRAME  
CODING SCHEME AND ARCHITECTURE-AWARE  
CONSTRUCTION OF RAPTOR CODES

by

HADY MOUNIR ALI ZEINEDDINE

A dissertation  
submitted in partial fulfillment of the requirements  
for the degree of Doctorate of Philosophy  
to the Department of Electrical and Computer Engineering  
of the Faculty of Engineering and Architecture  
at the American University of Beirut

Beirut, Lebanon  
January 2015

AMERICAN UNIVERSITY OF BEIRUT

RATE-COMPATIBLE CODING: AN INTER-FRAME  
CODING SCHEME AND ARCHITECTURE-AWARE  
CONSTRUCTION OF RAPTOR CODES

by

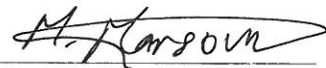
HADY MOUNIR ALI ZEINEDDINE

Approved by:

---

Mohammad Mansour, Associate Professor  
Electrical and Computer Engineering

Advisor



---

Mohamad Adnan Al-Alaoui, Professor  
Electrical and Computer Engineering

Committee Chair



---

Louay Bazzi, Associate Professor  
Electrical and Computer Engineering

Member of Committee



---

Emmanuel Boutillon, Professor  
Universite de Bretagne Sud, Lorient, France

Member of Committee



---

Rudiger Urbanke, Professor  
Ecole Polytechnique Federal de Lausanne (EPFL), Suisse

Member of Committee



Date of dissertation defense: January 5, 2015

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Ali Zeineddine Hady Mounir  
Last First Middle

Master's Thesis       Master's Project       Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, **three years after the date of submitting my thesis, dissertation, or project**, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

Hady January 27, 2015  
Signature Date

# Acknowledgements

My sincere gratitude goes for my parents for their unconditional and continuous help, support and understanding.

I am thankful for my advisor Professor Mohammad M. Mansour for his continuous support and guidance along the years of my PHD study.

I would like to thank Professors Mohamad Adnan Al-alaoui, Louay Bazzi, Ibrahim Abou-Faycal, Rouwaida Kanj, and George Turkiyyah for the motivating courses I was fortunate to take.

I would like to thank Ms. Rabab Abi Shakra and Ms. Alia Kazma from the Faculty of Engineering and Architecture for continuously helping me with the difficult navigation through paperwork, registration, and procedures.

I would like to thank Professor Louay Jalloul and Mr. Sam Alex, for their help and for the beneficial collaboration and motivating discussions we had during my internship at Broadcom Corporation.

I would like to thank my friend Mr. Ranjit Puri from Microsoft Corporation for his vital help in obtaining some hardware simulation figures in the initial phases of the work.

I would also like to thank Professor Fadi Kurdahi and Mr. Xiaoliang Chen, from the University of California, Irvine, for their vital help in the synthesis of the TDMP decoder.

Last but not least, I would like to thank my fellow graduate-student friends at AUB for their help and for making the study at AUB a reviving life experience, I particularly thank my friend Ms. Mervat Madi.

# An Abstract of the Dissertation of

Hady Mounir Ali Zeineddine for Doctorate of Philosophy  
Major: Electrical and Computer Engineering

Title: Rate-Compatible Coding: An Inter-frame Coding Scheme and  
Architecture-Aware Construction of Raptor Codes

Coping with wireless channel variability is a major challenge in channel coding. Typically, feedback-based techniques such as link adaptation and hybrid automatic-repeat-request (HARQ) are used to match the appropriate code-rate to the instantaneous channel quality, in order to achieve high-throughput communication over varying wireless channel conditions. Rate-compatible coding is a crucial element of HARQ schemes, and therefore has been subject to intense research in recent years.

In this dissertation, two related topics on rate-compatible coding are considered. First, the channel variation problem is considered in the broadcast communication scenario, in which schemes such as HARQ performs poorly. Drawing from the analogy between basic automatic-repeat-request and erasure coding, we propose to incorporate the rate-decreasing process into frame-level coding to achieve better complexity versus coding-performance tradeoff. The proposed scheme is called increment-based inter-frame coding. The corresponding coding algorithms and architectures are developed, and the coding performance of the scheme is assessed via asymptotic analysis and simulations. The results show that the proposed scheme yields higher data-rates compared to other conventional schemes such as HARQ and the state-of-the-art two-stage scheme involving both error-correcting and erasure coding. With regard to rate-compatible coding, the impact of the proposed scheme is two-fold: 1) it extends the application space of rate-compatible codes, and 2) imposes new design-requirements on these codes.

Second, Raptor codes are proposed to be deployed as rate-compatible codes at the physical layer. The motivation is that Raptor features could ensure good performance of these codes under various communication scenarios, including the proposed inter-frame coding scheme. However, these features themselves make the design of hardware-efficient decoders more challenging. A code design flow is thus proposed to combine the aspects of coding-performance and hardware-efficient decoding of short/moderate-length codes within one multi-step framework. Methods involved in each step of the flow are discussed, and the resulting decoder architecture is developed accordingly. Overall, simulation and implementation results indicate that the proposed flow can produce Raptor codes that combine good coding-performance and hardware-efficiency of decoding.

# Contents

<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>Contents</b>	<b>viii</b>
<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background and Motivation . . . . .	2
1.2 Contributions . . . . .	5
1.3 Dissertation Outline . . . . .	7
<b>2 Problem Setup</b>	<b>9</b>
<b>3 Background Material</b>	<b>12</b>
3.1 The Channel-State Variation Problem . . . . .	13
3.2 The Unicast Scenario: HARQ schemes . . . . .	16
3.3 The Broadcast Scenario: Existing Solutions and Drawbacks . . . . .	19
3.4 Rate-Compatible LDPC Code Design . . . . .	23
3.4.1 Code Puncturing . . . . .	23
3.4.2 Code Extension . . . . .	27
3.4.3 Protograph-based LDPC Code Extension: Assessment . . . . .	29
3.5 Raptor codes . . . . .	32
3.5.1 Encoding . . . . .	32
3.5.2 Decoding . . . . .	32
3.5.3 Code Properties . . . . .	34
3.6 Iterative Decoder Architectures . . . . .	36

<b>4</b>	<b>Increment-based Inter-frame Coding</b>	<b>41</b>
4.1	Algorithms . . . . .	44
4.1.1	Encoding . . . . .	44
4.1.2	Decoding . . . . .	44
4.1.3	Modified Scheme . . . . .	48
4.2	Architectures . . . . .	50
4.2.1	Inter-frame Encoder . . . . .	50
4.2.2	Inter-frame Decoder . . . . .	50
4.2.3	Discussion . . . . .	57
4.3	Channel Model . . . . .	60
4.4	Mathematical Characterization . . . . .	62
4.4.1	Optimality Question . . . . .	63
4.4.2	Bipartite Graph Model . . . . .	64
4.4.3	Asymptotic Performance Analysis . . . . .	68
4.4.4	Optimal Degree Distribution Construction . . . . .	71
4.5	Asymptotic Coding Performance . . . . .	81
4.5.1	IR-HARQ . . . . .	81
4.5.2	Two-stage Scheme . . . . .	83
4.6	Simulation Results . . . . .	88
<b>5</b>	<b>Architecture-Aware Raptor Codes</b>	<b>94</b>
5.1	Raptor Codes: Motivation and Challenges . . . . .	96
5.1.1	Intra-frame Varying Channel Condition in IIF Coding . . . . .	96
5.1.2	Raptor Coding Features . . . . .	97
5.1.3	Hardware Efficiency Challenges . . . . .	98
5.2	Code Construction Framework . . . . .	101
5.2.1	Implications on Design . . . . .	102
5.2.2	Decoding Scheduling . . . . .	103
5.3	Code Structuring . . . . .	107
5.3.1	$p$ -based Replication . . . . .	107
5.3.2	Product-group Replication . . . . .	109
5.4	LT Code Construction . . . . .	112
5.4.1	Reconfigurable CFU Design . . . . .	114
5.4.2	Structured Subcodes . . . . .	125
5.5	Precode Construction . . . . .	128
5.5.1	Row Merging Transformation . . . . .	128
5.5.2	Irregular Precodes . . . . .	129
5.6	Simulation Results . . . . .	133
5.6.1	Code Category 1: $p$ -based Replication and Row-Splitting . . . . .	133
5.6.2	Code Category 2: Product-group Replication and Subcode Structuring . . . . .	138



5.6.3	Implementation . . . . .	139
<b>6</b>	<b>Conclusion and Open Questions</b>	<b>145</b>
6.1	Conclusions . . . . .	145
6.2	Open Questions . . . . .	146
<b>A</b>	<b>Abbreviations</b>	<b>149</b>
	<b>Bibliography</b>	<b>150</b>

# List of Figures

2.1	Problem setup, transmitter . . . . .	11
3.1	FER versus frame-length . . . . .	14
3.2	HARQ mechanism . . . . .	17
3.3	Protograph-based LDPC code construction . . . . .	24
3.4	LDPC code extension using protomatrix . . . . .	28
3.5	Raptor bipartite graph . . . . .	33
3.6	Iterative decoder architecture . . . . .	36
4.1	Inter-frame encoding example . . . . .	45
4.2	Inter-frame decoding flowchart . . . . .	46
4.3	Inter-frame encoder architecture . . . . .	51
4.4	Inter-frame decoder architecture . . . . .	52
4.5	Architecture of decoding-scheduling block . . . . .	53
4.6	Bipartite graph modeling of an inter-frame code . . . . .	65
4.7	Enhancement ratio compared to IR-HARQ for infinite number of receivers, varying $\mu$ . . . . .	83
4.8	Enhancement ratio compared to IR-HARQ versus number of receivers, fixed $(\delta, \mu)$ . . . . .	84
4.9	<i>Effective frame-length</i> ratio versus intra-frame length, for different values of $\delta$ and $\mu$ . . . . .	86
4.10	Enhancement ratio compared to the two-stage scheme, versus $\mu$ . . . . .	87
4.11	Steps of construction of <i>Code1</i> and <i>Code2</i> . . . . .	90
4.12	<i>Inter-frame Decoding-failure Rate</i> versus $\sum_{\omega} \omega \cdot \delta_{\omega}$ . . . . .	91
4.13	Distributions of max. and avg. buffered-frame count and latency time, $N_F = 121$ . . . . .	92
4.14	Distributions of max. and avg. buffered-frame count and latency time, $N_F = 1210$ . . . . .	93
5.1	Serial Raptor decoder architecture . . . . .	105
5.2	$(p \cdot q)$ -permutation induced by $z^a w^b = (z^t \cdot w^u)^{-1}$ . . . . .	111

5.3	Row splitting example . . . . .	113
5.4	IVM network . . . . .	116
5.5	Reconfigurable accumulator-based CFU architecture . . . . .	118
5.6	Reconfigurable BCJR-based CFU architecture . . . . .	120
5.7	Operation and interconnect of the Min-Sum implementation . . . . .	122
5.8	Min-Sum memory reduction . . . . .	123
5.9	Row merging for $p$ -based replication, $p = 11$ . . . . .	130
5.10	Regular-to-irregular graph transformation example . . . . .	132
5.11	FER and BER performance comparison of LDPC and $p$ -based Raptor codes, rate 0.4 . . . . .	135
5.12	FER and BER performance comparison of LDPC and $p$ -based Raptor codes, rate 1/2 . . . . .	136
5.13	FER and BER performance comparison of LDPC and $p$ -based Raptor codes, rate 2/3 . . . . .	136
5.14	Decoding convergence speed comparison of LDPC and $p$ -based Raptor codes . . . . .	137
5.15	FER performance comparison of LDPC and Raptor codes (structured subcodes) . . . . .	139
5.16	Piece-wise linear approximation of $\psi(x)$ : error ratio of approximation	141
5.17	Coding performance of TDMP, for various quantization levels . . . . .	142

# List of Tables

5.1	Hardware resources required in the fixed-degree and reconfigurable parallel CFU architectures . . . . .	124
5.2	Parameters of the simulated LDPC and $p$ -based Raptor codes . . .	134
5.3	Number of edges in LDPC and $p$ -based Raptor graphs . . . . .	137
5.4	Average number of decoding iterations for LDPC and Raptor codes, at FER of $\sim 10^{-4}$ . . . . .	138
5.5	Message count for LDPC and Raptor Codes . . . . .	140
5.6	Hardware-complexity of the serial TDMP and TPMP decoders . . .	141
5.7	Power and area figures for the synthesized TDMP decoder . . . . .	144



# Chapter 1

## Introduction

With the advent of wireless communications, channel coding has gained increased importance. Channel codes are deployed to ensure reliable communication of data over the typically noisy wireless channels. A major step in channel coding is matching the code-rate to the corresponding channel-state. This channel-to-rate matching is required to achieve high communication data-rates: a code-rate higher than the appropriate rate implies a high probability of decoding failure at the receiver side, while a lower code-rate requires transmitting excessive redundancy bits, leading to power inefficiency and data-rate loss.

A major problem in matching the code-rate to the channel state is the variation of the channel-state itself, manifested as temporal and frequency variations in the signal-to-noise-plus-interference ratio. The overall impact of this variation is that the appropriate code-rate varies across the different transmitted frames. This means that the code-rate has to be adjusted, in case of fast channel variations, on frame-by-frame basis. The problem becomes more complicated in broadcast communication where the channel-state variation is now two-folded: 1) per single receiver, the channel-state varies across different frame-transmissions and 2) per single frame-transmission, the channel-state varies across different receivers.

The typical techniques developed to deal with the channel-state variation in unicast communication assume that rate-compatible channel codes are deployed. Rate-compatibility of a code means that for any two possible code-rate values  $R_h, R_l, R_h > R_l$ , the rate  $R_l$ -encoded frame is a concatenation of the rate  $R_h$ -frame and additional redundancy bits. Rate-compatible codes have to be designed to perform well under the different scenarios in which the channel-state variation problem arises.

In this dissertation, the problem of channel-to-rate matching is approached in two directions. In the first direction, it is shown that channel-to-rate matching can be performed on the receiver side, via an increment-based inter-frame coding approach. This turns out to be especially useful to enhance data rates in

broadcast communication, at the cost of a limited hardware overhead. The proposed increment-based inter-frame coding scheme deploys rate-compatible codes, therefore, extending the space of applicability of rate-compatible coding.

In the second direction, the deployment of Raptor codes as rate-compatible physical-layer channel codes is considered. For their peculiar features, Raptor codes are projected to perform well in all the scenarios arising in typical and new (like the proposed increment-based inter-frame coding) applications of rate-compatible coding. It is shown that Raptor codes can be constructed to result in hardware-efficient decoder implementations using an architecture-aware construction flow, that is developed here. Simulation results suggest the developed construction flow can produce codes that are good-performing as well.

Several papers were published based on the work presented here, basically on the first direction on architecture-aware Raptor codes [1–4]; others on the second direction of increment-based inter-frame coding are submitted and still under review [5,6].

## 1.1 Background and Motivation

*The Channel-to-Rate Matching Problem:* Channel-to-rate matching is typically achieved in unicast communication through a feedback-based scheme, as is done for example in LTE [7]. Instantaneous channel-state information (CSI) is fed back to the sender so that the appropriate code-rate is set prior to transmission. For the cases where obtaining instantaneous CSI is costly or infeasible due to fast/abrupt channel-state variations, hybrid automatic repeat request (HARQ) techniques [8,9] are used. HARQ has been adopted in several wireless communication standards (e.g. IEEE 802.16e/WiMAX [10] and 3GPP-LTE [11]). One of its most significant variations, the incremental-redundancy (IR)-HARQ [9], effectively matches the channel-state to the appropriate code-rate by successively increasing the encoding frame-length.

In broadcast communication, the channel-to-rate matching problem has to be reconsidered because feedback-based schemes do not scale well as the number of receivers grows. The underlying reason is that per single frame transmission, the channel-state varies also across different receivers. This means that for each frame, the code-rate must be matched to the worst channel-state instance among all receivers. Therefore, the average number of transmitted bits per frame is typically larger than the average number of bits required by each receiver for successful recovery of the transmitted frames.

In regard to this conclusion, the channel-to-rate matching problem can be reformulated into the more general problem of finding a coding scheme that achieves a good tradeoff between *complexity* and *date-rate*. Evaluating the complexity of

a scheme includes identifying the required hardware resources and the complexity of the underlying operations.

The state-of-the-art solution to this problem is based on a two-stage forward error-control scheme in which application-layer (APP-layer) erasure coding is combined with conventional physical-layer (PHY-layer) channel coding. The frames that fail PHY-layer decoding are discarded, whereas symbols in the successfully decoded frames are collected and forwarded to erasure decoding. Erasure decoding is then used to recover symbols erased due to channel decoding failures. This two-stage scheme however does not solve the matching problem for every frame; rather, it allows the communication to be done reliably even under some channel decoding failures, which usually occur when the code-rate is higher than the appropriate channel-matched instantaneous rate. This scheme has been included in the 3GPP multimedia broadcast/multicast services (MBMS) [12] and digital video broadcasting-handheld (DVB-H) [13] standards.

Erasure coding incurs a loss in the achieved data rates. The loss is due to the underlying erasure-channel abstraction itself: an unsuccessfully decoded frame is discarded, i.e., effectively erased. This deficiency is addressed in [14–16] by applying post-decoding processing on the frames on which decoding fails. However, these schemes incur a high complexity overhead [14] and/or impose certain assumptions on the bit-error rate in the unsuccessfully-decoded frames, thus, limiting their applicability [15, 16].

*The Rate-compatible Coding Problem:* In channel-to-rate matching procedures, it is assumed that the code-rate can be progressively decreased, that is the deployed channel code is rate-compatible. In general, for each mainstream class of error-correcting codes, a subclass of rate-compatible codes can be constructed. Two mainstream codes in communications are the Turbo [17] and LDPC [18–22] codes. These two classes of codes can achieve close-to-capacity performance while their iterative decoding algorithms allow efficient hardware implementation. Recently, polar codes [23] have been introduced and shown to be capacity-achieving, but both the finite-length performance and the hardware-efficiency of the corresponding decoders are still in need of further exploration.

A vast literature on the design of rate-compatible LDPC codes exists, e.g. [24–32]. The design methods are divided into two classes: code puncturing and code extension. Basically, the classification of a method is done according to the direction, in relation to the code rate, of the code design flow. In code puncturing, a low-rate mother code is first designed, and higher-rate codes are obtained by puncturing bits from it. Conversely, code extension essentially extends the parity-check matrix of a high-rate daughter code to obtain lower-rate codes. Both methods are shown to result in codes with asymptotically good performance. In particular, code extension was used to design near-capacity protograph-based LDPC code



sequences [31].

As is shown in [33], the coding performance of Turbo codes, in IR-HARQ, deteriorates significantly when the signal-to-noise ratio (SNR) level varies across different portions of the transmitted frame, the latter scenario called *intra-frame varying channel conditions* in this dissertation. This performance degradation is attributed to the structure of the involved code and the iterative decoding algorithm. For the same latter reasons, performance degradation under *intra-frame varying channel conditions* will also be observed in LDPC codes, particularly in the protograph-based extended LDPC codes [31] that are otherwise capacity-achieving over AWGN channels with fixed SNR values. The *intra-frame varying channel conditions* phenomenon happens in unicast communication when the channel is fast-fading. Its significance here is reconsidered because of the following: in the increment-based inter-frame coding scheme, developed for channel-to-rate matching in the first part of the work presented here, the occurrence of the *intra-frame varying channel conditions* scenario is expected and frequent.

Raptor coding [34] can be viewed, broadly, as a subclass of LDPC code extension. A Raptor code consists of a high-rate precode concatenated with a rateless LT code [35]. Raptor codes are characterized by the following important features: 1) each output bit is generated independently from other bits, 2) concatenating an increment of bits to the encoded frame does not change the number of bit-nodes in the decoding graph, and 3) the minimum distance of a Raptor code sequence can be well approximated by characterizing the weight spectrum of its precode. In [36], Raptor codes were considered for HARQ schemes and shown to have superior performance to LDPC codes at low signal-to-noise ratios (SNR). The aforementioned properties of Raptor codes suggest that these codes can be potentially deployed as PHY-layer channel codes yielding good performance in the different communication scenarios. This point will be made clear in the dissertation, when the background is thoroughly explained.

*The Architecture-aware Raptor Code Construction Problem:* Any LDPC code can be equivalently described by a Tanner bi-partite graph. LDPC decoding typically applies iterative message passing algorithms, where in each iteration, messages are exchanged along the edges connecting the two partitions of the corresponding Tanner graph. The signal flow during decoding is, therefore, determined by the Tanner graph topology corresponding to the LDPC code; the LDPC decoder architecture has to mimic, to a large extent, such graph topology. This implies that a strong relation exists between the hardware-efficiency of a LDPC decoder architecture and the underlying code structure, the latter determining the corresponding graph topology. All in all, the code structure has a major impact on the involved memory and interconnect organization, as well as on the memory access patterns, operation scheduling, and signal flow between different blocks.

It can be thus concluded, based on the aforementioned observation, that hardware-efficient LDPC decoder architectures can be obtained by appropriate structuring of the code. This design flow practice is called *architecture-aware code construction*. Typically, it consists of inserting appropriate regularity features in the structure of the code, and therefore, in the structure of the corresponding graph. Such regularity is hardware-friendly because it leads to regular memory-access patterns and signal flow, which in turn result in hardware-efficient implementations. The relation between the code structure and hardware-efficiency of decoding will be further detailed in the dissertation.

A major problem in deploying Raptor codes can be stated as follows: the peculiar features of Raptor codes, which make these codes potential candidates as good-performing codes, impose serious challenges on obtaining hardware-efficient decoder implementations. Example hardware-unfriendly features include the random LT-encoding, the variable check-degree distribution, and the two-code composition of the Raptor code. These irregularity and randomness features lead to low resource utilization, high control overhead, complex data movement patterns, in addition to stringent memory requirements, thus resulting in a highly inefficient decoder implementation. Overall, this imposes the problem of designing Raptor codes with two apparently contradicting requirements: 1) the code should be architecture-aware and 2) pertains the peculiar hardware-unfriendly features of Raptor coding.

## 1.2 Contributions

The research work presented in this dissertation addresses the problems raised in the previous section. It is composed of two parts. The first part proposes a novel increment-based inter-frame (IIF) coding approach which results in the application of channel-to-rate matching on the receiver side. Such approach is especially useful in the broadcast communication scenarios. The second part is on the development of frameworks and methods that result in architecture-aware construction of the highly irregular Raptor codes.

In the first part of the work presented in this dissertation, a novel inter-frame coding approach is developed to solve the channel-to-rate matching problem under varying channel-state conditions, in broadcast communication. The main advantage of the proposed scheme is that, compared to the state-of-the-art two-stage scheme, it can achieve significantly higher data rates, with almost the same complexity as LT erasure decoding [35]. Therefore, its hardware implementation overhead at the PHY-layer can be kept limited.

The basic feature of the proposed approach is that the appropriate code-rate is assigned to each frame on the receiver side. This is done by incorporating as-

pects of Incremental-Redundancy HARQ (IR-HARQ) [9] to coding. In particular, the coding scheme is increment-based in the sense that encoding is applied on increments corresponding to the transmitted frames, generating subframes rather than complete frames. The *inter-frame* decoding procedure performs iteratively two inter-related operations: 1) *intra-frame* decoding to recover the transmitted frames, and 2) progressive concatenation of increments to frames that are unsuccessfully-decoded prior to retrying *intra-frame* decoding on them.

The contributions in this part are stated next. First, the inter-frame encoding and decoding algorithms are proposed. Second, the corresponding inter-frame encoder and decoder architectures are developed. It can be concluded from the developed architectures that: 1) the major hardware units involved in inter-frame decoding are already available in any typical communication system, 2) the decoder can be implemented to support inter-frame code descriptions that are determined in real time, and, 3) in terms of hardware resources, the required memory size is the limiting factor in the design of large inter-frame codes. Third, the channel-state variation in relation to inter-frame decoding is modeled as a probability distribution over  $\mathbb{Z}^+$ , which is crucial in developing a generic yet simple framework for designing and analyzing inter-frame codes.

Fourth, the improvement brought by inter-frame coding to the achievable data rates is quantitatively studied using the developed channel model. This is done through a mathematical characterization of the asymptotic coding-performance of the inter-frame coding scheme, followed by comparing this performance to that of the other conventional schemes. The analysis done on the coding-performance is asymptotic in nature; it is performed using a four-step procedure that includes: 1) setting an “optimality” criterion for inter-frame coding, 2) bipartite graph modeling of the inter-frame code, 3) analyzing asymptotically the decoding progress, and 4) proving the “optimality” of inter-frame coding through constructing optimal edge degree distributions. A significant result is that *under the channel model developed in this work, iterative graph-based erasure decoding can be viewed as a special case of inter-frame decoding*. Subsequently, the performed modeling, analysis and distribution construction generalize the corresponding steps done in graph-based erasure codes.

Overall, it can be concluded that a significant enhancement in the data-rate is achieved by inter-frame coding. The extent of such enhancement is dependant of the channel-state statistical parameters. Compared to the IR-HARQ scheme having target frame-error-rate (FER) of  $10^{-3}$ , inter-frame coding increases the data rate by a factor reaching  $5\times$  when the number of receivers is infinitely large. Relative to the state-of-the-art two-stage scheme, inter-frame coding increases the data rate by a factor that can reach  $1.55\times$ . Simulation results show that significant data-rate enhancement compared to the two-stage scheme is also observed for

finite-length inter-frame codes, for example when the number of frames included in inter-frame coding is 121 or 1210.

In the second part of the work presented in this dissertation, architecture-aware Raptor code construction is considered. A three-stage code construction flow is proposed; it includes: code structuring, row-merging for precode construction, and subcode-generation for LT construction. The flow simplifies the design process by partitioning the design of the code, and thereafter decoder, into a set of disjoint, albeit related, subproblems. The construction flow generates both the LT code and precode starting from one regular matrix. On the other hand, it decouples code structuring, crucial to obtain hardware regularity and avoid short cycles in the Raptor bi-partite graph, from the inherently irregular pseudo-random LT encoding. A major consequence is that *decoding of the highly irregular Raptor code is mapped into sequential row-processing of a regular matrix*; a hardware-efficient architecture of this row-processor is subsequently developed. A major requirement for this mapping to be possible is the design of reconfigurable fixed-throughput check-function units that can process LDPC nodes as well as LT subcodes, in a sense that will be detailed later in the dissertation. In general, the design of such units can be challenging; however, three different designs of the reconfigurable check-function unit are developed, assuming a specific LT construction method. In this dissertation, each of the three stages of the construction flow is discussed, and the corresponding proposed solutions are linked to miscellaneous issues of hardware-efficiency an/or coding-performance.

The coding-performance of some sample codes, constructed according to the proposed construction flow, under additive white gaussian noise (AWGN) channels is simulated. The resulting frame-error-rate (FER) curves are comparable to that of standardized LDPC codes. This shows the potential of the proposed construction scheme to generate codes that have good coding-performance. Besides, the number of decoding iterations and memory size required in the decoding of the constructed Raptor codes are obtained, demonstrating the impact of the different proposed construction techniques on the hardware-efficiency of decoding. As a proof of concept, a serial Raptor decoder is synthesized in 65 nm, 1.2 V CMOS technology. Hardware simulations show that the decoder, decoding a rate-0.4 code instance, achieves a throughput of 36 Mb/s at SNR of 1.5 dB, dissipates an average power of 27 mW and occupies an area of 0.55 mm<sup>2</sup>.

### 1.3 Dissertation Outline

The rest of the dissertation is organized as follows. In Chapter 2, the problem setup is defined and some terminology is set.

Chapter 3 presents the background material and motivates the proposed re-

search directions. First, the channel-state variation problem is defined. Then, the conventional solutions to this problem in each of unicast and broadcast communication scenario are described. The drawbacks of these solutions in broadcast communication are thus clarified. Second, the rate-compatible coding problem is considered. The techniques to design rate-compatible LDPC codes are described and evaluated. Raptor coding is then discussed. Finally, an overview of the iterative LDPC decoder architecture is given; the main considerations in its design are identified along with the state-of-art design directions.

In Chapter 4, the increment-based inter-frame coding approach is proposed. First, the inter-frame encoding and decoding algorithms are described. Second, the corresponding architectures are presented. Third, the corresponding channel model is described. Fourth, mathematical characterization of the asymptotic coding performance of the proposed inter-frame coding scheme is done. The obtained performance is then compared to that of the state-of-the-art methods. Finally, some simulation results are presented.

Chapter 5 is on the architecture-aware construction of Raptor codes. First, the deployment of Raptor codes is motivated, partially in light of the developed inter-frame coding scheme. Next, the proposed construction framework is presented along with the resulting decoder architecture and scheduling. Each of the stages of the proposed framework is considered next, along with the architectural implications of the different design options. Finally, several aspects of the constructed Raptor codes, corresponding to their coding-performance and hardware-efficiency of decoding, are obtained via simulations.

Chapter 6 concludes the dissertation. It summarizes some main conclusions drawn from the work done, and discusses some resulting open questions.

# Chapter 2

## Problem Setup

In this chapter, the setup of the problem is described. A note on the terminology is however due prior to the description. Unless stated otherwise, symbols that are defined in this chapter have their definitions valid throughout the rest of the dissertation. Otherwise, a symbol defined elsewhere has its definition valid only within the section or chapter including it.

The communication scenario assumed in this dissertation can be summarized as follows. On the sender/transmitter side, an information sequence of  $N_F \cdot K$  bits, called here data bits, is to be sent to a number  $N_R$  of receivers. This sequence can be viewed alternatively as a sequence of  $N_F$   $K$ -bit information/data blocks. Each receiver is supposed to recover this sequence<sup>1</sup>, with a probability of failure below a certain target failure-rate. To keep the problem definition as general as possible, no further assumptions are made concerning the nature of the application involved in this communication process. If the number of Receivers  $N_R$  is equal to 1, the communication scenario is unicast; other-wise it is a broadcast communication scenario.

Figure 2.1 illustrates the problem setup, on the transmitter side, assumed throughout this work. The  $N_F \cdot K$  data bits are encoded using an APP-layer erasure encoding procedure with rate  $R_E = \frac{N_F}{N_T}$ . The occurrence of this encoding step depends on the overall communication scheme deployed to achieve the reliability of communication between the sender and receiver. In case no APP-layer erasure encoding is applied,  $R_E$  can be thought to be equivalently 1 and  $N_T = N_F$ . The resulting  $(N_T \cdot K)$ -bit sequence is partitioned into  $N_T$  blocks. Each block consists of  $K$ -bits or equivalently  $L$  symbols, where each symbol includes  $K/L$  bits. A block is forwarded to the PHY-layer channel encoder that generates a rate- $R_L$  encoded frame, where  $R_L = \frac{K}{N+D \cdot \Delta}$  and  $D, \Delta, N \in \mathbb{N}$  are parameters to be defined

---

<sup>1</sup>A relaxed requirement is that the ratio of the number of unrecovered bits to  $N_F \cdot K$  is very close to 0

next. This encoder is called “intra-frame” in this work to differentiate it from the APP-layer encoder and the inter-frame encoder described in Chapter 4. It is designed to be rate-compatible. Therefore, the rate- $R_L$  encoded frame can be viewed as a concatenation of a rate- $R_H$   $N$ -bit encoded frame, where  $R_H = \frac{K}{N}$ , and  $D$  vectors or increments of size  $\Delta$  bits each. The  $j$ th increment, is denoted by  $\Delta(f, j)$ , where  $1 \leq j \leq D$  and  $f$  is the frame index. The sequence of the  $N_T$  ( $N + D \cdot \Delta$ )-bit frames is then forwarded to a process that produces the bit stream to be transmitted over the wireless channel. One of the operations performed by this process is setting the code-rate of the frames to  $R_L \leq R \leq R_H$  by choosing the first  $K/R$  bits of each input ( $N + D \cdot \Delta$ )-bit frame for transmission. The resulting intra-frame code-size, or equivalently frame-length, is equal by definition to  $K/R$ . The inter-frame encoding process, proposed in Chapter 4 can be viewed as an example of such a process. Two notes should be made in this regard. First, the setup is made general enough to fit the proposed inter-frame coding scheme as well as the conventional solutions discussed in the following chapter. Second, it is assumed that the step of partitioning the bit sequence into blocks is done only at the APP-layer. Such assumption is made here solely for simplicity; it has no bearing on the validity of the results obtained in this work.

At the receiver side, it is assumed that the log-likelihood ratios (LLRs) of the transmitted bits are fed as input to a recovery process that restores the  $N_T$  blocks. This recovery process is the focus of this dissertation, in which two aspects of it will be discussed. In Chapter 4, the recovery process is proposed to be the inter-frame decoding process. This process, and any other conventional process, includes the PHY-layer channel-decoding (typically soft-decoding) as a major component. This channel-decoding is called here “intra-frame” to differentiate it from inter-frame decoding. The design of hardware-efficient and good-performing intra-frame Raptor decoders is the focus of Chapter 5. The following related terminology is defined. An  $N$ -LLR vector corresponding to the first  $N$ -bit portion of a transmitted frame  $f$ ,  $f \leq N_T$ , is denoted by  $\Lambda_N(f)$ , whereas a  $\Delta$ -LLR vector corresponding to  $\Delta(f, j)$  is denoted  $\Lambda_\Delta(f, j)$ .

A note should be made on the term *coding-performance* used throughout the dissertation. In the context of intra-frame coding, the performance of a code is measured in terms of its frame-error-rate (FER) versus the signal-to-noise ratio (SNR), obtained under for the considered wireless channel. In the context of inter-frame coding (Chapter 4), the term *coding-performance* is equivalent to the achievable data-rate. It is measured in terms of the resulting *effective frame-length* defined as: the average number of bits, per  $K$ -bit information/data block, that needs to be transmitted so that the probability *that the receiver fails to recover the  $(N_F \cdot K)$ -bit information sequence* is below some target value. It is therefore equal to the total number of transmitted bits divided by  $N_F$ .

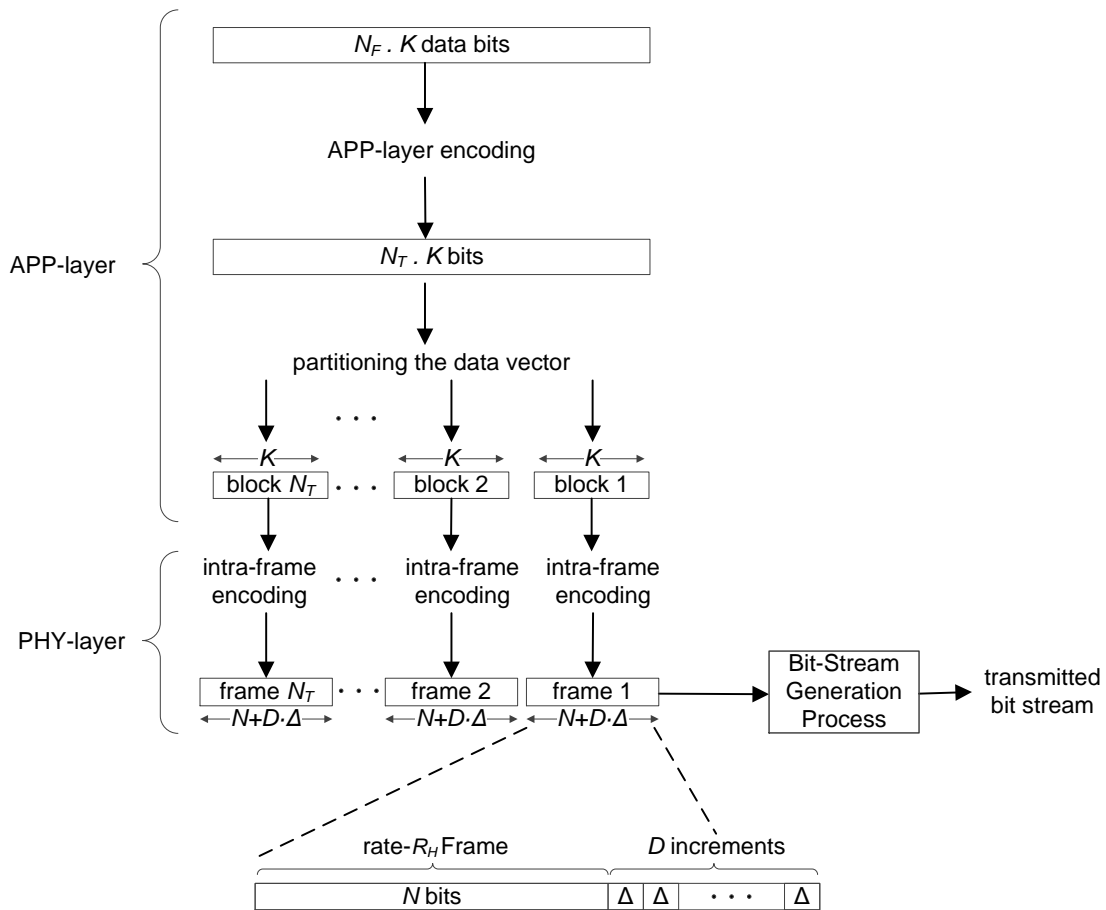


Figure 2.1: Problem setup: transmitter side.



# Chapter 3

## Background Material

The problem addressed in this dissertation is a composition of two related problems that can be briefly stated as follows:

1. Enhancing the achievable data-rates in broadcast communication, by applying channel-to-rate matching on the receiver side; this is the premise of the proposed increment-based inter-frame (IIF) coding approach. This approach extends the application of rate-compatible coding to the broadcast communication scenario.
2. The development of architecture-aware rate-compatible codes that perform well when applied in both the unicast scenario (conventional application) and broadcast scenario (proposed application).

The composition of this problem induces the division of the current chapter into two parts. In the first part, the problem of the *variation of the channel-state across the different transmitted frames* is stated. Then, the state-of-the-art approach to this problem in the unicast communication scenario is clarified by describing the hybrid automatic repeat request (HARQ) scheme. This latter scheme requires the deployment of efficient rate-compatible codes for good performance. The channel-state variation problem in the broadcast communication scenario case is then discussed. The discussion includes the particular challenges faced in the broadcast scenario, in addition to the existing solutions and their drawbacks. In the second part of this section, the design of rate-compatible codes is considered. The existing work on the design of rate-compatible LDPC codes is discussed. Of these codes, Raptor codes, being a main subject of the presented work, are explained. Then, the problems dealt with in the design of hardware-oriented codes are clarified by giving a general overview of the state-of-the-art iterative LDPC decoder architectures.

### 3.1 The Channel-State Variation Problem

A defining characteristic of the mobile wireless channel is the variation of the channel strength over time and frequency; such variation can be divided into *large-scale* and *small-scale* fading [37]. Large-scale fading is usually due to power decay with distance and shadowing while small-scale fading is typically due to constructive and destructive interference of the multiple signal paths between the transmitter and receiver. The signal quality is also affected by inter-user interference, an example of which is the intra-cell and inter-cell interference in cellular systems, which is time-varying as well. A key property of wireless communication can be then be stated as follows: *the state of the channel over which the frames are transmitted varies across different frames.*

In terms of the setup described in Chapter 2, a major impact of the *channel-state variation* problem is that *the appropriate code-rate  $R$ ,  $0 \leq R \leq 1$ , varies across different frames.* This, in turn, poses the problem of *matching the code-rate to the corresponding channel-state*, under varying channel conditions. This channel-to-rate matching is required to achieve high communication data-rate: a code-rate higher than the appropriate rate implies a high probability of decoding failure at the receiver side, while a lower code-rate requires transmitting excessive redundancy bits, leading to power inefficiency and data-rate loss.

The variation of the appropriate code-rate is quantitatively illustrated in Fig. 3.1. The corresponding plots in the figure show the frame error-rate (FER), defined as the rate of decoding-failure of a frame, versus the frame-length over a fading channel. The simulation setup can be summarized as follows: the wireless channel is modeled as a Pedestrian-B (PedB) fading channel [38], subject to white Gaussian noise with fixed variance. A  $K$ -bit block is encoded via an LTE turbo code [38] into a  $(N + D \cdot \Delta < 3K)$ -bit encoded frame  $f$  with parameters  $(K, N, \Delta) = (4096, K/0.75, 0.1N)$ . The modulation scheme used is 16-QAM. For every  $1 \leq i \leq D$ , the transmission of the  $(N + i \cdot \Delta)$ -bit portion of the frame over the wireless channel is simulated and decoding is applied on the corresponding log-likelihood ratio values. Two FER values are output  $\forall i$ : one assumes that the  $N$ -bit portion and each of the increments  $\Delta(f, j)$ ,  $1 \leq j \leq i$ , are transmitted over *decorrelated* channel instances; the other assumes the whole  $(N + i \cdot \Delta)$  portion is transmitted over a *fully correlated* channel instance. For both values, the simulation assumes an independent channel instance for every instance of the  $(N + i \cdot \Delta)$ -bit portion transmission. Overall, the simulation results show a clear example of the case where the FER changes relatively gradually rather than abruptly, as the code-rate changes.

**Channel-state Correlation:** The previously described simulations assume an independent channel instance for every single frame transmission. This block-fading channel assumption overlooks the possible correlation in the channel-state

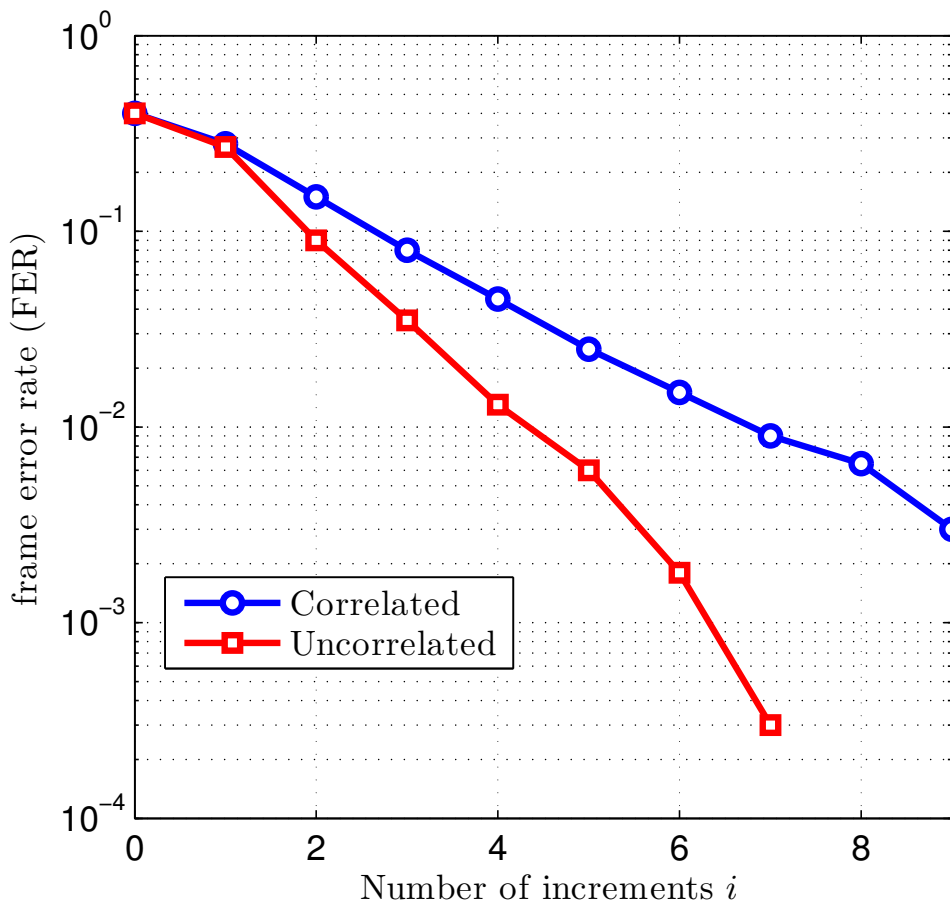


Figure 3.1: FER versus frame-length. The x-coordinates are measured in terms of the number of increments concatenated to the  $N$ -bit portion.

across consecutive frame transmissions. In general, the magnitude of this correlation is dependant on the channel characteristics, an example of which is the channel *coherence time*. It is dependant, as well, on the time elapsed between two consecutive frame transmissions which is application-dependant. As will be shown in the next section, different approaches are typically applied in unicast communication, depending on whether the channel-state changes relatively slowly (slow fading, significant correlation) or fast (fast fading, little correlation). In the broadcast communication scenario, the correlation does not affect the correctness of the inter-frame coding solution proposed in this work, since such correlation can be mitigated by increasing the inter-frame code size and using interleaving. Yet, these latter measures do have an impact on the required hardware resources and latency time, subjects that will be discussed in Chapter 4. Another correlation figure to be considered is the *intra-frame channel-state correlation* defined here

as: the magnitude of correlation between the 1) channel-state under which the transmission of the  $N$ -bit portion of a frame  $f$  occurs, and 2) the channel-state under which the transmission of the corresponding increments,  $\Delta(f, i)$ , occurs. The significance of the intra-frame channel-state correlation in the design of rate-compatible codes will be clarified later in this chapter, when discussing the HARQ scheme (3.2) and rate compatible LDPC-code extension (3.4), and in the chapter on architecture-aware Raptor code construction (5).

**Problem Setup Revisited** In light of the channel-state variation problem, the partitioning of the information sequence into  $K$ -bit blocks has to be revisited. A plausible solution to the cross-frame channel-state variation is to encode the  $N_F \cdot K$  information bits into one frame with an appropriate code-rate. The motivation is that if the frame-length is large enough, the availability of the statistical channel-state information (CSI) would be sufficient to determine the optimal code-rate, as is suggested by the basic communication theory of Shannon. Two practical reasons make this solution implausible. First, the decoder typically has to receive the whole frame before starting the decoding procedure. This would lead to a large latency between the generation of the information sequence at the sender side and recovering it at the receiver side. Second and more importantly, is that the channel decoder, usually implemented in hardware as an ASIC (Application-Specific Integrated-Circuits) solution for power and throughput reasons, has to support now much larger frame-lengths as the information block-length is multiplied by a factor of  $N_F$ , (from  $K$  to  $N_F \cdot K$ ). This is clearly impractical in virtually any hardware-based system.

## 3.2 The Unicast Scenario: HARQ schemes

In the unicast communication scenario, the channel-state variation problem is typically managed through a feedback-based scheme, as is done for example in LTE [7]. The approach used in this scheme depends on the time period over which the channel quality changes. For a slow variation in channel quality, the channel-state information (CSI) at the transmitter is updated upon feedback from the receiver, initiating what is called link adaptation: the modulation type and code-rate are changed to cope with the change in the reported channel quality. Link adaptation however does not deal with the case of decoding-failure events. In addition, for relatively fast variations in the channel quality, the channel-state information on the sender side becomes obsolete quickly and has to be updated with an impractically high frequency; thus making the link-adaptation error-prone, costly and/or even infeasible. Numerous Automatic Repeat Request techniques are developed to approach this problem [7].

Automatic repeat request (ARQ) is an acknowledge-based error-control mechanism. In its most basic form, error-detection bits are added to the data bits, typically generated using the cyclic redundancy check (CRC) codes. On the receiver side, the redundant bits are used to detect an error event occurrence in the transmitted frame. A 1-bit flag on data recovery success or failure, corresponding respectively to detection of an error event or not, is fed-back to the transmitter. In case an error is detected, the frame is retransmitted. This procedure is clearly inefficient and variations of the ARQ mechanism are developed to incorporate more efficient recovery procedures.

Hybrid-ARQ (HARQ) is one such variation that combines forward error correction to ARQ procedure. HARQ has been widely adopted in wireless communication standards (e.g. IEEE 802.16e/WiMAX [10] and 3GPP-LTE [11]). It involves successive feedback-based retransmissions to achieve high throughput. A rate- $R_H$  encoding frame is transmitted, and then an extra “retransmission” is invoked each time the feedback from the receiver indicates a decoding failure. HARQ schemes are mainly based on either Chase-combining [8] or incremental-redundancy (IR) [9]. Chase-combining involves repetition coding: retransmissions consist of bits that were within the initially transmitted rate- $R_H$  frame, on which the receiver applies maximal-ratio combining prior to retrying to decode the frame. In the IR-HARQ scheme illustrated in Fig.3.2, retransmissions consist of additional redundancy bits. Rate-compatible coding is, therefore, a fundamental component of IR-HARQ. In the rest of this dissertation, it is assumed that the  $i$ th retransmission corresponding to frame  $f$  is the increment  $\Delta(f, i)$ . The transmission of increment  $\Delta(f, i)$  means that the code-rate goes from  $\frac{K}{N+(i-1)\cdot\Delta}$  down to  $\frac{K}{N+i\cdot\Delta}$ . IR-HARQ can be thus viewed as matching the channel-state to the appropriate code-rate by successively increasing the encoding frame-length to its appropriate

value.

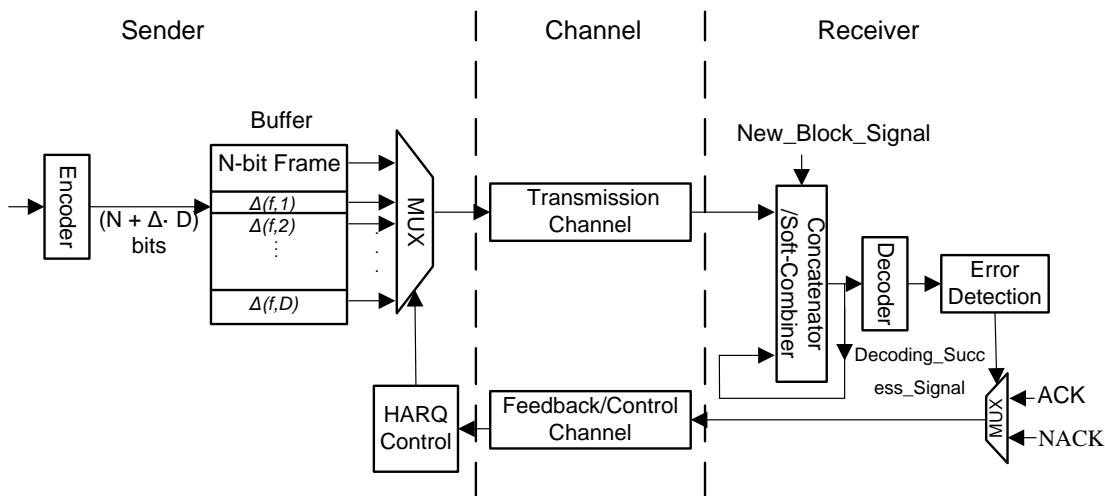


Figure 3.2: HARQ mechanism for both the transmitter and receiver.

Compared to Chase-combining, IR-HARQ achieves coding gain and, thus, should have better performance. However, it is shown in [33] that this is not always true, as will be briefly described here. The reported theoretical results in [33] were obtained using the accumulated mutual information metric conditioned on the signal-to-noise ratio (SNR) level. Besides, simulations were applied, using turbo codes, to verify the theoretical findings. Two significant results were deduced. First, the analysis shows that IR-HARQ outperforms Chase-combining, and that the corresponding gain tends to increase with the initial code rate (i.e.  $R_H$ ) but decrease with the signal-to-noise (SNR) variation between retransmissions. Second, when the 1) SNR disparity between the initial transmission and the successive retransmissions is high and 2) the initial rate- $R_H$  frame is transmitted through a channel instance with low SNR, the simulation results deviate considerably from the theoretical results. The gain achieved by IR-HARQ over Chase-combining tends to be less than that predicted by the theoretical model. In the extreme case where the systematic part of the turbo codeword is effectively erased, the Chase-combining scheme could outperform some IR-HARQ schemes. This result is reasonably attributed to the structure and iterative decoding method of the turbo code. The latter scenario of high SNR disparity between the initial transmission and the successive retransmissions could be due to the low *intra-frame channel-state correlation*, defined in Section 3.1. The scenario of low *intra-frame channel-state correlation* will be called *intra-frame varying channel condition* throughout the rest of this dissertation. Overall, the following conclusion can be made: *while IR-HARQ outperforms Chase-combining, the structure of the deployed*

*rate-compatible code and its corresponding decoding method can make it difficult to realize this performance gain in some scenarios, particularly in the case of intra-frame varying channel condition.* This point will be revisited later in this chapter, in the discussion on the design of rate compatible LDPC codes (Section 3.4).

Several enhancements were proposed to the initial HARQ method. For example, to enhance throughput, adaptive IR-HARQ was proposed and analyzed in [39, 40], where different retransmissions can have different lengths. It is noteworthy that while the IR-HARQ is typically implemented at the PHY-layer, the concept of IR-HARQ can be extended to upper-layers as in [41], where it is applied over the erasure channel.

A perfect example in which the link adaptation and HARQ techniques are used to achieve reliable and efficient data communication is LTE [7, 42]. Considering link adaptation for downlink transmissions for instance, the feedback from the receiver called channel quality information (CQI) indicates directly the appropriate combination of modulation-scheme and code-rate out of 16 possible choices. The possible modulation schemes are QPSK, 16QAM, and 64QAM, while the code-rate varies approximately from 0.93 down to 0.076. A 1/3-turbo code is deployed as the mother code which is used to realize any of the possible code-rates through puncturing and/or repetition. A circular-buffer rate-matching algorithm is used to select the bits for transmission from the rate-1/3 frame as briefly described: each of the systematic-bit partition, even-indexed parity-bit partition and odd-indexed parity-bit partition is interleaved separately, and then placed in a circular buffer via a very simple permutation. Transmission or retransmission is done by choosing a starting point, called Redundancy Version (RV), and reading the required number of bits for transmission serially with wrap around to the beginning of the buffer if the end of the buffer is reached. For efficient HARQ schemes, a number of different starting points (RVs) are chosen. The resulting HARQ is, therefore, a hybrid of chase-combining and incremental-redundancy HARQ techniques.

### 3.3 The Broadcast Scenario: Existing Solutions and Drawbacks

In broadcast communication, the channel-to-rate matching problem has to be re-considered. Feedback-based schemes, like HARQ, do not scale well as the number of receivers grows. The underlying reason is that the channel-state variation in the broadcast scenario is two-folded: 1) per single receiver, the channel-state varies across different frame-transmissions and 2) per single frame-transmission, the channel-state varies across different receivers. The impact of this two-folded variation will be clarified in this section when discussing the application of HARQ in the broadcast scenario.

In general, three conventional schemes are typically deployed, jointly or disjointly, to achieve reliable communication in the broadcast scenario: 1) PHY-layer forward error correction (FEC), 2) HARQ, and 3) erasure coding. For clarity, these schemes will be evaluated qualitatively, and the resulting arguments will be illustrated by numerical examples from Fig. 3.1 where necessary.

**1. PHY-layer FEC with no feedback:** To achieve a small FER, each  $K$ -bit information block has to be encoded using a very low code-rate. Considering the *uncorrelated channel* curve in Fig. 3.1 as an example, a frame-length of  $(N+6\cdot\Delta)$  is required to achieve a FER of  $\sim 10^{-3}$ . From the curve itself, it can be deduced that approximately 99.4% of the successfully decoded frames, could have also been successfully decoded if the frame-length was set to  $(N+5\cdot\Delta)$ , instead of  $(N+6\cdot\Delta)$ . This observation points out to the coding inefficiency of PHY-layer FEC stand-alone solution: for the vast majority of the transmitted frames, much smaller frame lengths are sufficient for successful decoding.

**2. IR-HARQ:** In the unicast scenario, HARQ is optimal in the sense that the number of retransmission trials is equal to the number of retransmissions required for successful decoding by the receiver. However, applying the HARQ technique in the considered broadcast scenario is not sufficient to result in such optimal performance, as explained next. For simplicity, assume that all the receivers have the same statistical channel-state description. Furthermore, as stated in 3.2, for each frame  $f$ , the first  $N$ -bit portion of  $f$  is initially transmitted and the  $i$ th retransmission consists of  $\Delta(f, i)$ . Per frame, the number of transmitted increments must be greater than or equal to the number of increments required by each receiver. This means the number of transmitted increments must match the receiver with the worst instantaneous channel-state condition. If the number of users is large enough, then for every frame, there exists with high probability a receiver whose corresponding channel is temporarily bad, and therefore requires a large number of transmitted increments. It can be checked that the underlying phenomenon of matching the number of retransmissions to *the worst channel-state instance*,



has its counterpart in any other feedback-based scheme. This conclusion is numerically illustrated in Chapter 4 where the data-rate values obtained by each of the IR-HARQ and the proposed increment-based inter-frame coding scheme are compared, under the channel model developed there.

**3. Erasure coding/Two-stage scheme:** The state-of-the-art scheme in broadcast communication is based on a two-stage forward error-control scheme in which APP-layer erasure coding is combined with conventional PHY-layer channel coding. The use of erasure coding is illustrated as follows: the information sequence of  $N_F \cdot L$  symbols is encoded, using a rate- $R_E$  erasure code, into  $\frac{L \cdot N_F}{R_E}$  encoding symbols. These symbols are partitioned into  $N_T = \frac{N_F}{R_E}$   $K$ -bit/ $L$ -symbol blocks, each of which is forwarded to the PHY-layer encoder to produce a rate- $R$  frame that is transmitted over the channel. At the receiver side, each of the  $N_T$  frames is intra-frame decoded; if decoding fails, the  $L$  symbols corresponding to the frame are considered erased; otherwise, the corresponding  $L$  symbols are recovered and collected. The collected symbols are then used, via erasure decoding, to recover the original  $N_F \cdot L$  information symbols.

Erasure coding is typically deployed at the APP-layer forming, together with PHY-layer intra-frame coding, a two-stage forward error-control scheme. This scheme is motivated by several factors. First, the simplicity of the erasure-channel model implies that the corresponding decoding algorithms are significantly simpler than the intra-frame decoding algorithms. Thus, erasure decoding can be efficiently implemented in software, making it flexible and adaptable to the type of application involved. Second, due to this combination of flexibility and relative simplicity, the erasure-code size can be made much longer than the PHY-layer frame-length so that the erasure code spans multiple PHY-layer frames. Therefore, erasure coding is used to withstand any type of channel-state variation that leads to a relatively high intra-frame decoding-failure rate, i.e. high FER. Third, contrary to the feedback-based retransmission schemes, coding scales well as the number of receivers goes up. It can withstand, for any receiver, an erasure rate that converges to  $1 - R_E$  as the number of transmitted frames  $N_T$  goes to infinity.

Of the codes proposed to be deployed for erasure-coding are Reed-Solomon codes [43] and LT/Raptor codes [24, 35]. Due to their linear-time encoding and decoding and their excellent coding performance, Raptor codes were included in the 3GPP multimedia broadcast/multicast services (MBMS) [12] and digital video broadcasting-handheld (DVB-H) [13] standards. RaptorQ [44] is an enhancement of Raptor codes, to support larger block sizes and achieve better coding performance [45]. Beside the encoding procedure, a key factor in achieving this coding performance is the inactivation decoding method of [46], which combines the low complexity of the belief-propagation method, originally assumed in [24], with the decoding guarantee of Gaussian elimination. While Raptor coding is typically im-

plemented in software, hardware accelerators are proposed to enhance the power and decoding throughput [47].

An optimization problem resulting from the two-stage scheme is how to determine the appropriate combination of intra-frame code-rate  $R$ , and erasure code-rate  $R_E$ , to minimize the overall redundancy involved in encoding the information sequence and, thus, optimize the *coding performance* metric. This problem is considered in [48, 49] under various assumptions on the channel model and communication scenarios. It is considered again in Chapter 4, under the channel model developed there, in order to compare the coding performance of the two-stage scheme to that of the proposed inter-frame coding.

The main drawback of the two-stage scheme is due to the underlying erasure-channel abstraction it involves: a frame of which intra-frame decoding fails is dropped and the corresponding symbols are considered erased. The erasure-channel abstraction is most appropriate for a polarized physical channel behaviour, in which the channel is either extremely noisy (deep fading) such that the received frame is nearly erased, or good enough for the frame to be successfully decoded. However, as illustrated in Fig. 3.1, most decoding failures at intra-frame code-length of  $N$  can be recovered by sending few additional increments of redundancy bits, that is by decreasing the code-rate beyond  $K/N$  and retrying intra-frame decoding. This is actually the main motivation of IR-HARQ in unicast communication. To quantitatively illustrate the data-rate loss due to the erasure-channel abstraction, the following example is considered: assume intra-frame decoding of two frames results in one decoding failure (i.e. one frame dropping) and one decoding success. Overall, in the conventional two-stage scheme, a total of  $2 \cdot N$  bits are transmitted to accumulate the  $K$  systematic bits included in the successfully-decoded frame. On the other hand, if concatenating  $i$  increments to the unsuccessfully-decoded  $N$ -bit frame is sufficient for successful intra-frame decoding, then  $(N + i \cdot \Delta) < 2 \cdot N$  bits need to be transmitted to accumulate  $K$  bits. For unicast scenarios, this is achieved through feedback-based HARQ schemes. However, feedback-based schemes do not scale well as the number of receivers grows.

The deficiency caused by erasure-channel abstraction is addressed in [14–16] by applying post-decoding processing on the frames on which intra-frame decoding fails. In [14], a “revive” stage is added which consists of applying the sum-product algorithm on the LT-code graph corresponding to these frames. The enhancement in the data-rate, however, is offset by a reported significant increase in the overall decoding complexity. Furthermore, the proposed LT-decoding is similar to the message-passing algorithm applied in the PHY-layer low-density parity-check (LDPC) decoding, and therefore should be implemented in ASIC to achieve reasonable throughput and power figures. Beside the resulting significant hardware

overhead, this limits the flexibility of LT-coding in terms of graph-connectivity and code-size. In [15,16], searching/testing procedures based on packet-combining are devised to correct the errors in the unsuccessfully-decoded frames. The main drawback of these techniques is that in order to attain good coding performance at tolerable complexity, the unsuccessfully-decoded frames are assumed to include very few errors (e.g. less than  $10^{-2}$  in [16]). Such assumption limits the applicability of these techniques. Overall, these solutions can be viewed as variations of the two-stage scheme involving different complexity versus coding performance tradeoffs.

All in all, it can be concluded that data rates that are higher than those obtained from the three described solutions are achievable. Yet, a reasonable requirement is that any solution achieving higher data-rates should involve a good tradeoff between *complexity* and *coding performance*. That is, the enhancement in the coding performance must not come along an excessive increase in the complexity of the overall scheme. Evaluating the complexity of a scheme includes identifying the required hardware resources and the complexity of the underlying operations. In Chapter 4, a novel increment-based inter-frame coding scheme is proposed and shown to involve a good tradeoff between *complexity* and *coding performance*. It should be noted here that, like the HARQ scheme, this scheme involves rate-compatible coding. However, the discussion of the requirements imposed by the proposed inter-frame coding scheme on the design of rate-compatible codes will be delayed to Chapter 5, preceded by the review of the state-of-the-art rate-compatible LDPC code design (next in this chapter) and the description of the proposed inter-frame coding scheme (Chapter 4).

## 3.4 Rate-Compatible LDPC Code Design

The rate-compatible code design problem was first considered for BCH [9] and convolutional [50] codes. This was later followed by the design of rate-compatible turbo [51–53] and LDPC codes. A main target of the research is to explore the design of hardware-efficient rate-compatible codes, which are decoded using the LDPC iterative message-passing algorithms. The Raptor coding solution is proposed to achieve this goal, and is studied accordingly. Raptor codes [24] can be viewed as a distinctive subclass of LDPC codes. Therefore, the Raptor design challenges can be fully understood within the context of the rate-compatible LDPC code-design problem, a survey on which is presented next.

The design of protograph-based rate-compatible LDPC codes [54] is, in particular, crucial due to the fact that these codes are amenable to hardware-efficient decoder architectures. A protograph is a Tanner graph with a relatively small number of nodes, connected by a small number of edges, which also allows parallel edges. A protograph is represented by a protomatrix whose entries indicate the number of edges connecting the respective variable and check nodes. The Tanner graph of the code is then obtained from the protograph using the following copy-and-permutation operation: the protograph is replicated by a factor of  $Q$ ; then, the  $Q$  edges, formed by replicating 1 edge in the protograph, are permuted. If the involved permutations, are cyclic shifts, the resulting codes are called *Quasi-Cyclic*. The protograph-based LDPC code construction is illustrated in Fig. 3.3. Being constructed by replicating a small protomatrix, the code has hardware-efficient partially-parallel decoder architectures, as discussed in 3.6. It is noteworthy that protograph-based LDPC codes are a subclass of the multi-edge type LDPC codes proposed in [55].

In general, construction of rate-compatible LDPC codes can be done through two main methods: 1) code puncturing and 2) code extension. A combination of these two methods can be used as well to achieve good coding-performance over a wide range of code-rates. From a code construction point of view, no hard line exists between these two classes of methods: code extension itself can be viewed as a special form of code puncturing and vice versa. The distinction is, however, clearer when examining the LDPC parity-check matrix, and consequently, the decoding procedure.

### 3.4.1 Code Puncturing

Code puncturing involves encoding a data block, of length  $K$ , using a low-rate ( $R_L$ ) mother code. A higher-rate ( $R_H > R_L$ ) coded-frame is then obtained by puncturing  $\frac{K}{R_L} - \frac{K}{R_H}$  bits of the rate- $R_L$  frame, where the punctured bits are not transmitted over the channel. On the decoder side, the iterative decoding

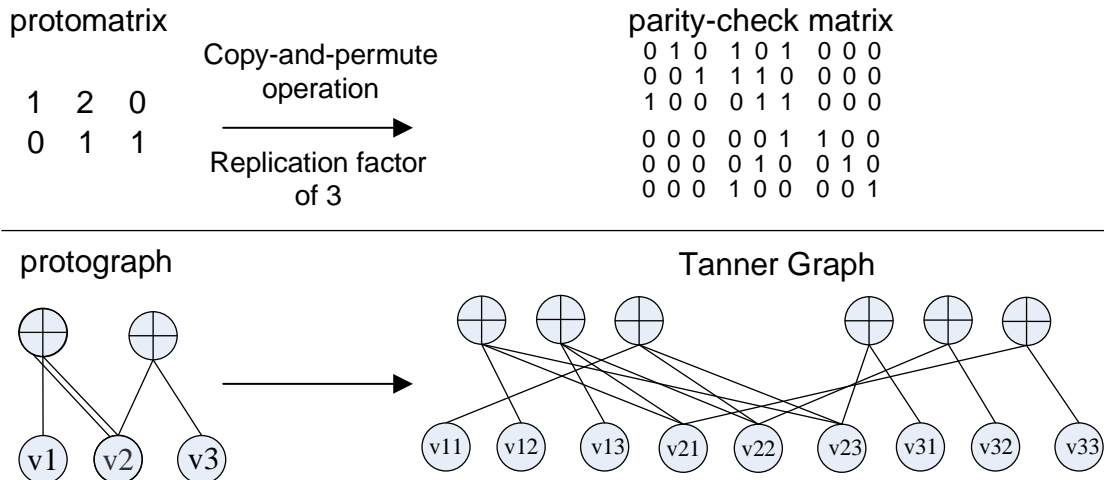


Figure 3.3: Protograph-based LDPC code construction.

algorithm is applied on the tanner graph of the mother code, with the LLRs of the punctured bits set to 0. To ensure rate-compatibility, the following condition must be satisfied:  $\forall R_H > R_{H'} > R_L$ , the punctured-bit set in the rate- $R_{H'}$  code, is a subset of the punctured-bit set of the rate- $R_H$  code.

The asymptotic performance of punctured LDPC codes has been subject to several studies (see e.g. [25–27, 56]). In [26], the average weight-distribution of the LDPC codes, and their asymptotic growth rate as well, are analyzed. Accordingly, it is proved that capacity-achieving codes under ML decoding, of any rate and for any memoryless binary-input output-symmetric (MBIOS) channel, can be constructed by puncturing some original LDPC code with small enough rate. In [25], it is shown that for any ensemble of LDPC codes of rate  $R_1$ , there exists an ensemble of punctured LDPC codes of the same rate, with parent code-rate  $R_2 < R_1$ , having the same threshold under the belief propagation algorithm. The puncturing, in [27], is done in the following way: the bits are grouped according to their corresponding variable-node degrees, and for each degree  $d$ , a fraction  $\pi_d$  of the degree- $d$  bits is punctured. The fractions  $\pi_j, j \geq 1$ , are optimized for performance of the message passing algorithm under additive white Gaussian noise (AWGN) channels. The resulting rate-compatible puncturing results in a small performance loss, that is the punctured codes are good (in terms of the  $E_b/N_0$  SNR threshold) across a range of rates when compared with the optimal codes for each rate. The obtained gap-to-capacity is in the range of 0.2 – 1 dB, with the gap increasing as the code-rate increases, that is as more bits are punctured. The phenomena of the gap-to-capacity increasing with rate is avoided in [56] by applying joint-optimization across all the considered rates.

The performance degradation with increasing rates, observed in some punc-

tured LDPC codes, can be attributed to the puncturing step itself. To clarify this point, random puncturing is considered: it is equivalent to transmitting the original rate- $R_L$  frame over a binary erasure channel. Assume the portion of punctured bits,  $1 - \frac{R_L}{R_H}$ , exceeds the maximum erasure-rate for which iterative decoding of the mother code under binary erasure channel succeeds. Then, decoding of the rate- $R_H$  frame transmitted over the wireless channel fails, regardless of the channel noise level. This threshold is called the puncturing threshold [25]. Typically, as the portion of punctured bits approach the threshold, performance deteriorates rapidly. This problem is aggravated in the case of finite-length code design, where frame-lengths are in the range of few Kbits. In an extreme case, bad puncturing can lead to the creation of a stopping set, leading to a decoding failure probability of 1, under the conventional iterative decoding and regardless of the wireless channel condition.

The design of finite-length punctured LDPC codes consists, typically, of choosing a good-performing mother-code followed by applying a heuristic to determine the appropriate puncturing pattern, that is the set of punctured bits. In [57], the mother-code was designed using the progressive edge growth algorithm with some variable degree distribution, under the constraint that the submatrix of the parity-check matrix corresponding to parity bits is lower triangular for fast encoding. A similar approach is followed in [28], where the parity-bit part of the parity-check matrix is deterministically constructed, while the design of the systematic-partition is oriented to yield a target degree distribution. In [29], a technique is developed to construct the low-rate mother-code protograph from a higher rate protograph, while optimizing the resulting AWGN and erasure decoding thresholds. The puncturing heuristics are tailored to deal with a general form of the phenomena discussed in the previous paragraph. For example, the notion of recovery tree of a punctured-bit node was introduced in [30]. For a variable/bit-node  $v$ , a recovery tree  $T(v)$  is the shortest tree, in the Tanner graph of the mother code, with the following properties: 1) the root node is  $v$ , 2) the neighboring bit-nodes of a check-node in the tree, are also in the tree, and 3) and the leaves are unpunctured bit-nodes. If the tree is of depth 2, that is  $v$  has at least 1 check-node neighbor whose all bit-node neighbors, excluding  $v$ , are unpunctured, the bit corresponding to  $v$  is recovered in 1 iteration for a channel noise level of 0. It is called one-step-recoverable (1-SR). In general, if the depth  $T(v)$  is  $2 \cdot m$ , the bit corresponding to  $v$  is recovered in  $m$  iterations for a channel noise level of 0. It is then called  $m$ -step-recoverable ( $m$ -SR). A grouping-and-sorting algorithm is then developed to determine the puncturing pattern/order of the code. Variable nodes are partitioned into groups  $G_i$ ,  $i \geq 1$ , where  $G_i$  is the set of  $i$ -step-recoverable nodes. The algorithm tries to maximize the size  $G_i$ , progressively as  $i$  increases. The bits in  $G_{i-1}$  are punctured prior to puncturing bits in  $G_i$ . Within a single  $G_i$ , the nodes

are sorted in an increasing order of the number of unpunctured nodes in the recovery trees, and punctured accordingly. A variation of this method is proposed in [58], where the puncturing order of variable nodes is determined according to a proposed cost function. The proposed cost function tries to maximize the minimum reliability among those provided from all check nodes and allocate survived check nodes (connected to only 1 punctured bit-node) evenly to all punctured variable nodes. In addition, the puncturing algorithm prevents the formation of a stopping set from the punctured variable nodes. In [59], the puncturing scheme is tailored to have the punctured bits “far” apart from each other in the Tanner graph of the mother code. Further improvement can be brought to the puncturing algorithm by considering additional criteria, an example of which is approximate cycle extrinsic message degree (ACE). The ACE measures, per cycle, the number of edges connecting the cycle to the rest of the graph; the algorithm proposed in [60] aims, in broad sense, to avoid the case that a punctured bit is involved in a high number of low-ACE cycles.

The puncturing-pattern generation, per code, involves a considerable hardware overhead. In general, the puncturing procedures do not have simple ASIC (application specific integrated circuits) implementations. Alternatively, it is not memory-efficient to store the puncturing patterns for each mother code instance. A major simplification to the problem is brought by protograph-based LDPC code construction. The puncturing can be applied on the protograph of the code rather on the whole tanner graph, and therefore the memory required to store the puncturing pattern is significantly reduced. Such block-level puncturing, has two main draw-backs: first, the number of realizable rates is limited since the frame-length changes in multiples of the replication factor of the code; and second, it becomes difficult, under some protograph-based designs, to avoid the formation of stopping sets by the punctured bits [29]. Yet, such approach yielded less than 0.3 dB gap to capacity over binary-input AWGN channels for a rate range of  $[\frac{8}{9}, \frac{8}{16}]$  in protograph-based  $E^2RC$ -like codes [56]. In addition, block-level puncturing was also applied in [61], on a rate-1/2 short-length LDPC code defined in WiMax standard. Puncturing within blocks was merely used, to realize a code-rate that is between two discrete rates obtained from block-level puncturing. With such puncturing scheme, a bit-error-rate (BER) performance degradation of less than 0.2 dB was reported.

A detailed evaluation of puncturing depends on the details of both the LDPC mother-code and the puncturing scheme. However, three observations can be made regarding 1) the code-rate range, 2) the convergence speed of decoding, and 3) the number of operations performed in one decoding iteration. First, the minimum allowable rate of the code is the mother-code rate, and rates beyond the mother-code rate can only be obtained through repetition. Second, decoding of a punctured

code combines both erasure- and error- correcting, since it involves recovering the punctured bits and correcting the erroneous unpunctured bits. This suggests that for a certain code-rate, the average number of iterations needed to decode a punctured LDPC code is higher than its counterpart in a fixed-rate LDPC code. To illustrate this conclusion, consider the puncturing scheme based on the group-and-sorting algorithm in [30]: as the needed code-rate increases, a larger portion of bits are punctured, then the maximum value of  $i$ , ( $i_{MAX}$ ) such that some  $i$ -SR nodes are punctured, increases. Under the two-phase message-passing (TPMP) algorithm, this means that at least  $i_{MAX}$  iterations are needed so that all variable nodes have nonzero posterior LLRs. This analysis is supported in [62], where decoding of a punctured rate-0.75 code required approximately 10 additional iterations to converge compared to the decoding of a dedicated/fixed rate-0.75 code. This problem is approached in [63] through a combination of turbo-decoding message-passing algorithm and efficient check node layering/reordering. However, more study is needed to understand fully the extent of improvement brought by such scheme, especially as the mother-code rate decreases. Besides, check node layering is a technique usually used to design efficient decoder architectures, and using it to increase convergence speed would, therefore, restrict the decoder-architecture design space. The third observation is that the number of operations, being computations or memory-accesses, done in one decoding iteration is higher for punctured codes than for dedicated fixed-rate codes. This can be justified by the fact that the number of performed operations in a single decoding iteration is proportional to the node and/or edge count in the tanner graph of the code. For a rate  $R_H$ , the numbers of edges and nodes in the low-rate mother-code are greater than their counterparts in the dedicated (fixed-rate) rate- $R_H$  code. For instance, in the LDPC codes defined in the WiMax standard (IEEE802.16 [10]), a dedicated rate-5/6 LDPC code has 4 edges per information bit as compared to 6.33 edges for the code formed by puncturing a rate-1/2 mother-code.

### 3.4.2 Code Extension

Code extension methods generate progressively low-rate codes starting from a high-rate daughter code, as illustrated in Fig.3.4. Considering two rates  $R_H > R_L$ , the parity matrix  $\mathbf{H}_L$  of a rate- $R_L$  code is generated from the parity matrix  $\mathbf{H}_H$  of a rate- $R_H$  code by adding an equal number,  $\frac{K}{R_L} - \frac{K}{R_H}$ , of rows and columns to  $\mathbf{H}_H$ . In rate-compatible coding, the linear constraints satisfied by the original  $\frac{K}{R_H}$ -bit frame, persist in the new lower-rate code. Therefore, the top right submatrix of  $\mathbf{H}_L$ , formed from the rows of  $\mathbf{H}_H$  and the added  $\frac{K}{R_L} - \frac{K}{R_H}$  columns, is a zero submatrix. Unlike the case of punctured codes, the tanner graph on which iterative decoding is applied grows as the code-rate goes down.



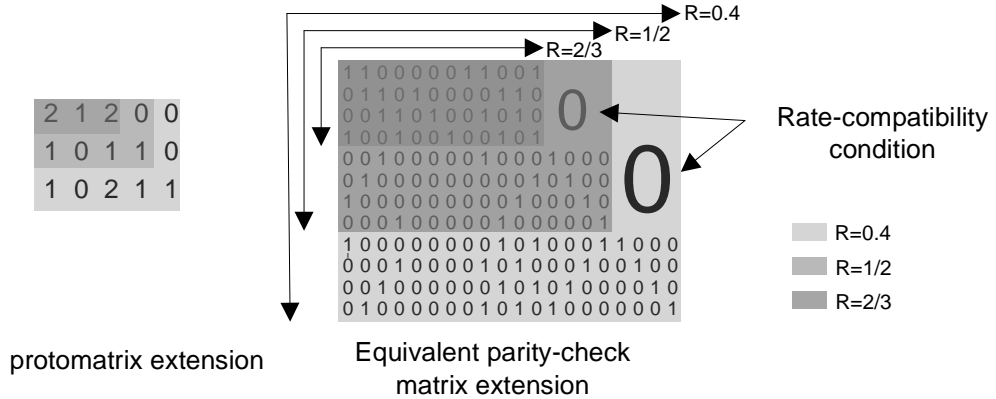


Figure 3.4: LDPC code extension, applied on the protomatrix instead of the parity-check matrix. The rate changes from  $2/3$  to  $1/2$ , and then to  $0.4$ .

One advantage of constructing rate-compatible codes using extension is that the LDPC Tanner graph can be extended indefinitely. Thus, there exists no lower bound on the minimum code-rate as the information block-length  $K$  goes to  $\infty$ . In practice, two considerations determine the minimum rate: a possible widening gap to capacity as rate decreases and the availability of sufficient hardware resources.

Several code extension techniques were proposed in literature. The construction framework in [64] includes a combination of code puncturing and extension to obtain a wide range of possible rates, but no specific extension procedure is proposed except imposing the constraint that the weights of the added columns have to be greater or equal to 3. The construction in [57] is based on three elements: 1) high-level structuring of the extended matrix, 2) some optimized variable degree distribution and 3) and progressive edge-growth techniques to avoid short cycles and approach the target distributions. In [65], code extension uses both check splitting, to vary the check-node degrees with decreasing rates, and edge growth techniques to obtain the target variable degree distributions.

Recently, a method to design near-capacity protograph-based extended codes was proposed in [31]. An example code sequence was constructed, over a rate range of  $[0.32, 0.83]$ , showing less than 0.2 dB asymptotic gap to the corresponding capacity limits. Similar techniques were used in [32, 66] to obtain coding-efficient protograph-based short-length codes. The design method progressively repeats the following: it adds one check node and one variable node to the protograph and then searches for the optimal edge-connection between these new nodes and the protograph. This is equivalent to searching for the best integer values of the entries of the row added to the protomatrix. The search space is restricted by constraints aiming to ensure low searching complexity, good coding thresholds, and linear minimum-distance growth. The search step is based on the PEXIT

method [67], an EXIT-based (extrinsic information transfer) method that can deduce the decoding threshold of a code from its protomatrix.

### 3.4.3 Protograph-based LDPC Code Extension: Assessment

Protograph-based LDPC extended codes have excellent coding-performance at different block-lengths and are amenable, via their structure, to efficient decoder implementations as discussed in 3.6. However, three observations can be made on the protograph-based LDPC code extension, pointing out to possible needed improvements.

First, code extension, in general, increases both the vertical and horizontal dimensions of the protomatrix, with decreasing code-rates. As is shown in 3.6, such variation restricts the decoder architecture design-space in terms of scheduling, interconnect, and memory organization. However, the effect of such restriction on the hardware-efficiency of decoding can be made limited for most decoder architectures.

Second, the growth of the minimum distance of short-length codes as rate decreases is, in general, difficult to control alongside the “waterfall” coding performance<sup>1</sup>. The problem is aggravated when the permutation groups involved in the replication of the protographs, are restricted to cyclic shifts where: the obtained minimum distances, in certain classes of quasi-cyclic codes, clearly deviate from the corresponding general protograph-based LDPC ensemble bounds [68]. Presence of low-weight codewords leads to relatively high error-floors as well as to failure in LDPC error-detection if the decoding converges to another codeword. Simulation results in [32] show good error-floors for the constructed code instances, however, more study is needed on the analysis and control of the minimum-distance as the rate and/or information block-length  $K$  vary.

Third, the performance of the extended LDPC codes under the communication scenarios involving *intra-frame varying-channel condition* is not clear. For sake of illustration, two of these scenarios are discussed next, assuming the codes are constructed according to the protomatrix extension shown in Fig.3.4. The initial code has a rate of  $2/3$  and  $1 \times 3$  protomatrix; upon concatenating the first increment to the frame, the rate becomes  $1/2$  and the protomatrix dimensions  $2 \times 4$ ; similarly, the rate goes down to  $2/5$  and the protomatrix dimensions change to  $3 \times 5$  upon transmitting the second increment.

1. *Out-of-order increment concatenation*: the second increment is assumed to be transmitted over the channel while the first increment is not. This behav-

---

<sup>1</sup>“Waterfall” performance corresponds to the steep part of the frame-error-rate curve plotted versus  $E_b/N_0$ .

ior, while unlikely in HARQ, can happen frequently in other communication schemes, namely the increment-based inter-frame coding proposed in Chapter 4. In the context of the *intra-frame varying channel condition* problem, the first increment is assumed to be transmitted over an AWGN channel with a noise level of standard deviation  $\sigma \rightarrow \infty$ . The code-rate is then 1/2, but decoding is applied on the whole  $3 \times 5$  protomatrix shown next, where the bit-nodes corresponding to the fourth column in the protomatrix are considered punctured.

$$\begin{bmatrix} 2 & 1 & 2 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 2 & 1 & 1 \end{bmatrix}$$

Combining the last two rows, and therefore, omitting the fourth punctured column leads to the following equivalent protomatrix, which describes the obtained rate-1/2 code:

$$\begin{bmatrix} 2 & 1 & 2 & 0 \\ 2 & 0 & 3 & 1 \end{bmatrix}$$

It can be clearly seen, that the resulting code protomatrix is determined by the order of increment-concatenation. It is not therefore the following rate-1/2 code protomatrix, assumed in the code design and which is obtained by in-order increment concatenation:

$$\begin{bmatrix} 2 & 1 & 2 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

In general, beside the resulting decoding hardware-efficiency problems, the performance of the code, effectively subject to random puncturing due to out-of-order increment concatenation, can be significantly degraded.

2. *Bad initial channel-condition:* assume the initial frame is transmitted under very bad channel conditions, while the increments are transmitted under significantly improved channel conditions. This scenario is similar to that observed in [33], and described in 3.1. This scenario can occur in unicast communication, with IR-HARQ applied, when the corresponding channel is fast-fading such that the time period separating the transmission of the initial frame and its corresponding increments is large enough for the channel to decorrelate. It can also occur when a synchronization error between the sender and receiver causes the initial rate- $R_H$  transmission to go undetected by the receiver. This scenario is relevant as well in the proposed increment-based inter-frame coding scheme.

To illustrate the problem faced in such scenario, the following extreme case is assumed: the initial rate- $R_H$  frame is transmitted over an AWGN noisy channel with standard deviation  $\sigma \rightarrow \infty$ , that is the channel converges to an erasure channel with erasure rate  $\rightarrow 1$ , then the corresponding channel changes abruptly to a noiseless channel. A number of increments is assumed to be transmitted over the noiseless channel.

Consider the protograph-based LDPC code example; the initial rate-2/3 frame is assumed erased, while the two increments are received in perfect reliability, i.e. with LLRs of  $\pm\infty$ . The decoding is applied on the code described by the whole  $3 \times 5$  protomatrix. Consider the submatrix that is formed by the rows corresponding to the increments and the columns corresponding to the initial frame, it is:

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 2 \end{bmatrix}$$

Since the number of 1-entries in each row is more than 1, the check-to-bit messages going from the check nodes corresponding to the added increments will have zero value. Therefore, all the messages exchanged in the iterative decoding algorithm will have zero value. In general, if the resulting code extension is such that each of the added rows has more than one 1-entry in the first  $\frac{K}{R_H}$  positions corresponding to the bit-nodes of the initial frame, then the following happens: these bit-nodes would form a stopping set, and the frame can never be recovered using iterative decoding regardless of how many rows are added. That is, iterative decoding cannot recover the block regardless of the number of transmitted increments.

In regard of these observations, code extension techniques require further enhancement to increase their reliability and applicability in different schemes. Raptor coding can be viewed, in a broad sense, as a subclass of code extension with distinctive properties that make it a potential solution for the aforementioned problems. It is discussed next.

## 3.5 Raptor codes

A Raptor code [24] is composed of a fixed-rate precode concatenated with a rateless LT-code [35]. The LT-code has a minimum distance that is bounded by the minimum bit-node degree and, hence, exhibits an error floor at relatively high BERs. A precode is, thus, needed to attain high code minimum-distance and avoid high error floors.

In this section, the next section, and in Chapter 5 on the architecture-aware Raptor code construction, the definitions of  $K$ ,  $N$  and rate  $R$  are changed slightly from their previous definitions in Chapter 2, as will be made clear next.

### 3.5.1 Encoding

Given a block of  $K$  data bits, the encoding process is done in two stages. First, the block is encoded using the precode into a new frame of  $K_{LT}$  bits. Next, the  $K_{LT}$ -bit codeword is LT-encoded to generate  $N = \frac{K}{R}$  output bits, for some code-rate  $R$ . Each LT output bit is generated independently as follows. A pre-designed distribution  $\mathcal{D}$  on the integers  $1, \dots, d_{\max}$  is sampled to obtain an integer  $d$ , called the output degree, then  $d$  bits of the  $K_{LT}$ -bit frame are chosen at random, and their modulo-2 sum (XOR) is transmitted. The design of the distribution  $\mathcal{D}$  is crucial for the resulting code to yield good error-correcting performance.

Similar to LDPC codes, an LT-code can be represented by a bipartite (Tanner) graph with  $K_{LT}$  variable/bit-nodes representing the input bits on one side, and  $N$  check-nodes representing the output bits on the other (see Fig. 3.5). An edge exists between check-node  $i$  and bit-node  $j$  if bit  $j$  is an input to the XOR whose output is check-node  $i$ . In this case, nodes  $i$  and  $j$  are said to be neighbors. The degree of a node  $i$ , denoted by  $\deg_i$ , is the number of edges connected to it. If all the variable(check) nodes in the graph have the same degree, the graph (code) is called *regular*, otherwise it is *irregular*. The *girth* is defined to be the minimum cycle-length in the graph.

An LT-code can be equivalently represented by an  $N \times K_{LT}$  matrix  $\mathbf{H}_{LT} = [h_{ij}]$ , where  $h_{ij} = 1$  if bit-node  $j$  is connected to check-node  $i$ , and 0 otherwise. By abuse of notation,  $\mathbf{H}_{LT}$  is called the parity-check matrix of the LT code in the rest of this document. If the precode is an LDPC code, the Raptor code can be represented by a bipartite graph with the check-node partition composed of LT and LDPC check nodes as shown in Fig. 3.5.

### 3.5.2 Decoding

A Raptor code can be decoded using Gallager's two-phase message-passing (TPMP) algorithm used for LDPC codes [18], where two types of messages are exchanged

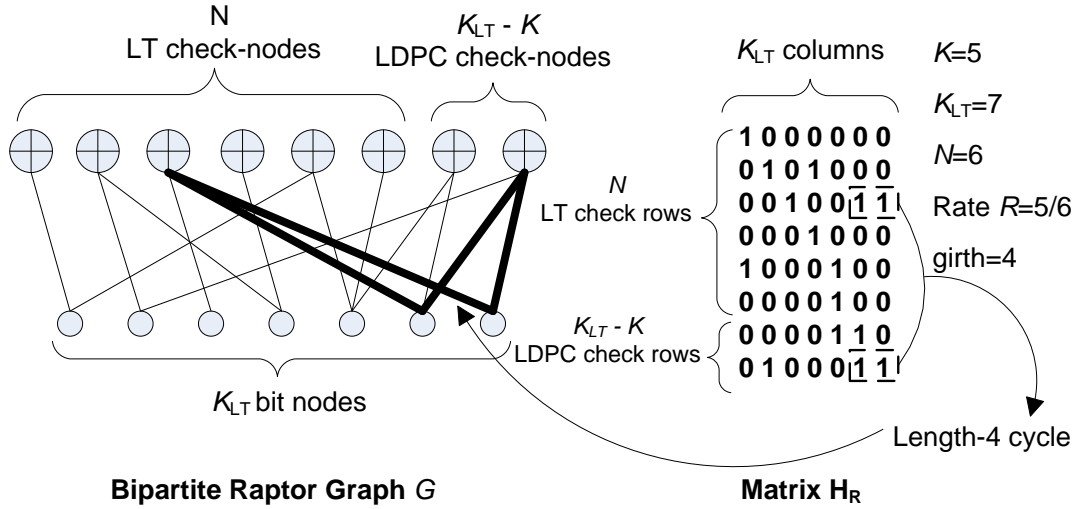


Figure 3.5: Bipartite graph  $G$  and parity-check matrix  $\mathbf{H}_R$  of a Raptor code. A length-4 cycle is highlighted on  $G$  and  $\mathbf{H}_R$ .

between bit-nodes and check-nodes in an iterative manner. By abuse of the notation defined in Chapter 2, let  $\Lambda_i$  be the intrinsic channel reliability value of the  $i$ th check-node,  $CTB_{ij}[\tau]$  the check-to-bit message from check-node  $i$  to bit-node  $j$  at iteration  $\tau$ , and  $BTC_{ji}[\tau]$  the bit-to-check message from bit-node  $j$  to check-node  $i$  at iteration  $\tau$ . We denote by  $\mathcal{C}_j$  the index set of the check-node neighbors of bit-node  $j$  (so  $\deg_j = |\mathcal{C}_j|$ ), and by  $\mathcal{B}_i$  the index set of the bit-node neighbors of check-node  $i$  (so  $\deg_i = |\mathcal{B}_i|$ ). We use the notation  $\mathbf{CTB}_j[\tau]$  to denote the  $1 \times |\mathcal{C}_j|$  vector of check messages to bit-node  $j$  at iteration  $\tau$ , and  $\mathbf{CTB}[\tau]$  to denote all  $K_{LT}$  vectors  $\mathbf{CTB}_j[\tau]$  of check messages at iteration  $\tau$ .  $\mathbf{BTC}_i[\tau]$  and  $\mathbf{BTC}[\tau]$  are similarly defined. For simplicity, we drop the subscripts  $i, j$  and iteration index  $\tau$  when the context is clear or when arbitrary nodes are considered. The decoding algorithm is described as follows:

1. **At iteration  $\tau$ ,  $0 \leq \tau \leq \tau_f$ :**

*Phase 1:* Compute for each bit-node  $j$  the message  $BTC_{ji}[\tau]$  to every check-node  $i \in \mathcal{C}_j$  according to

$$BTC_{ji}[\tau] = \sum_{\substack{i' \in \mathcal{C}_j \\ i' \neq i}} CTB_{i'j}[\tau - 1], \quad j = 1, \dots, K_{LT},$$

with initial conditions  $BTC_{ji}[0] = 0$  for  $j = 1, \dots, K_{LT}$ , and  $i \in \mathcal{C}_j$ .

*Phase 2:* Compute for each check-node  $i$ , the message  $CTB_{ij}[\tau]$  to every bit-node  $j \in \mathcal{B}_i$ :

- . If node  $i$  is an LT check-node, then

$$\begin{aligned}
|CTB_{ij}[\tau]| &= \begin{cases} |\Lambda_i|, & \deg_i = 1; \\ \psi^{-1} \left( \psi(|\Lambda_i|) + \sum_{j' \in \mathcal{B}_i, j' \neq j} \psi(|BTC_{j'i}[\tau]|) \right), & \deg_i > 1. \end{cases} \\
\text{sgn}(CTB_{ij}[\tau]) &= \begin{cases} \text{sgn}(\Lambda_i), & \deg_i = 1; \\ \text{sgn}(\Lambda_i) \cdot \prod_{j' \in \mathcal{B}_i, j' \neq j} \text{sgn}(BTC_{j'i}[\tau]), & \deg_i > 1. \end{cases} \tag{3.1}
\end{aligned}$$

. If node  $i$  is an LDPC check-node, then

$$\begin{aligned}
|CTB_{ij}[\tau]| &= \psi^{-1} \left( \sum_{\substack{j' \in \mathcal{B}_i \\ j' \neq j}} \psi(|BTC_{j'i}[\tau]|) \right); \\
\text{sgn}(CTB_{ij}[\tau]) &= \prod_{\substack{j' \in \mathcal{B}_i \\ j' \neq j}} \text{sgn}(BTC_{j'i}[\tau]). \tag{3.2}
\end{aligned}$$

where  $\psi(x) = -\log(\tanh(x/2))$ .

2. **Decision phase:** At the final iteration  $\tau_f$ , bit  $j$  is set to 0 or 1 according to the sign of  $\sum_{i \in \mathcal{C}_j} CTB_{ij}[\tau_f]$ .

Decoding is terminated when either the maximum number of decoding iterations is reached or when the decoded bits satisfy all the check-constraints of the LDPC precode.

### 3.5.3 Code Properties

Raptor coding, being a rate-compatible encoding procedure, shares common features with code puncturing and extension. A Raptor code can be, alternatively, described by a LDPC-like  $(K_{LT} - K + N) \times (K_{LT} + N)$  parity-check matrix:

$$\begin{bmatrix} \mathbf{H}_P & \mathbf{0}_{(K_{LT}-K) \times N} \\ \mathbf{H}_{LT} & \mathbf{I}_{N \times N} \end{bmatrix}$$

where  $\mathbf{H}_P$  is the precode parity-check matrix,  $\mathbf{H}_{LT}$  the LT parity-check matrix,  $\mathbf{I}_{N \times N}$  the  $N$ -dimensional identity matrix and  $\mathbf{0}_{(K_{LT}-K) \times N}$  the  $(K_{LT} - K) \times N$  zero matrix. The first  $K_{LT}$  columns correspond to the  $K_{LT}$  LT input-bits while the next  $N$  columns correspond to the  $N$  LT output bits. Therefore, a Raptor code of a certain rate, can be viewed as an LDPC code having its first  $K_{LT}$  bits punctured. On the other hand, Raptor coding shares the following key feature with LDPC code extension: rate-compatible change of the code-rate is achieved by augmenting the

bipartite graph of the code, or equivalently extending its parity-matrix. For this reason, Raptor coding can be viewed as a subclass of code extension.

Raptor codes were initially designed to operate over erasure-channels. The performance of the designed codes over AWGN channels was shown, through simulations, to be good [69]. The application of Raptor codes over binary-input symmetric channels was analyzed in [34], and good-performing degree distributions were designed and tested. As part of the current research work on raptor codes with relatively low-rate (e.g. 5/6) precodes, the following results are obtained: for any rate  $R \in \{\frac{1}{4}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}\}$ , a LT degree distribution can be designed such that the Raptor decoding threshold is less than 0.2 dB away from capacity, for AWGN channels.

A main result proved in [34], is that, except for the case of binary erasure channel, no universally-optimal LT degree distribution exists for Raptor codes at different-rates. This means that for close-to-capacity Raptor code design, the following condition must be satisfied: to decrease the LT code-rate,  $R_{LT} = \frac{K_{LT}}{N}$ , by  $\Delta R$ , the additional  $\frac{K_{LT}}{(R_{LT}+\Delta R)} - \frac{K_{LT}}{R_{LT}}$  LT output bits are generated according to a degree distribution that is a function of the initial rate  $R_{LT}$ . This condition implies a modification in the original Raptor encoding procedure, diminishing further the distinction between Raptor and LDPC extended codes.

Raptor-like protograph-based LDPC codes were constructed in [32, 66], by extending the protomatrix of the daughter code  $\mathbf{B}_P$  as follows:

$$\begin{bmatrix} \mathbf{B}_P & \mathbf{0}_{(K_{LT}-K) \times N} \\ \mathbf{B}_{\text{extension}} & \mathbf{I}_{N \times N} \end{bmatrix}$$

The similarity to Raptor coding is achieved by imposing the condition that the  $N \times N$  bottom right submatrix is an identity matrix rather than a lower-triangular submatrix. In light of this,  $\mathbf{B}_P$  can be viewed as the precode parity-check matrix and  $\mathbf{B}_{\text{extension}}$  the LT parity-check matrix. However, the code has two properties differentiating it from conventional Raptor codes: 1) most of the LT-input bits are transmitted first, i.e. the first  $K_{LT}$  bits are almost unpunctured and 2) the LT output bits are generated (deterministically) on the protograph level. Therefore, while the codes show good coding performance over AWGN channels, they retain two problems of the LDPC extended codes: 1) it is not clear how the growth of the minimum distance with the decreasing code rate is controlled/designed for short-length codes, and 2) the performance under the some *intra-frame varying channel condition* scenarios can seriously degrade, an example of which is the *bad initial channel-condition* scenario already discussed in 3.4.3 .



### 3.6 Iterative Decoder Architectures

A main requirement in the design of LDPC codes is their architecture-awareness, that is the existence of corresponding hardware-efficient decoder implementations. This requirement persists in Raptor code design as well. The link between the code properties and decoder efficiency is clarified, next, through a general overview of the LDPC iterative decoder architectures and the main optimizations involved to enhance their hardware-efficiency.

In general, an iterative decoder architecture, illustrated in Fig. 3.6, is composed of three main components: 1) a check-node processor including check-function units (CFUs) to compute the check-to-bit messages and a check-node memory to store these messages, 2) a bit-node processor including bit-function units (BFUs) to compute the bit-to-check messages and a bit-node memory to store these messages, and 3) an interconnect network to communicate the messages between the check-node and bit-node processors. Fully-parallel decoder architectures are not efficient in the sense that they involve both a large number of processors and register files and an extremely complex interconnect that has to mimic the full topology of the corresponding Tanner graph; besides, it is has to be operate on tanner graphs of different sizes due to varying information-block lengths and/or rates. Memory is needed by the check-node and bit-node processors in serial and partially-parallel architectures in which a portion of the messages is processed per clock cycle. This need to memory stems from the edge-interleaving between the bit-node and check-node partitions of the tanner graph. For example, check-to-bit messages computed by the check-node processor in the same clock cycle will be processed by the bit-node processor in different clock cycles, thus requiring memory to store them.

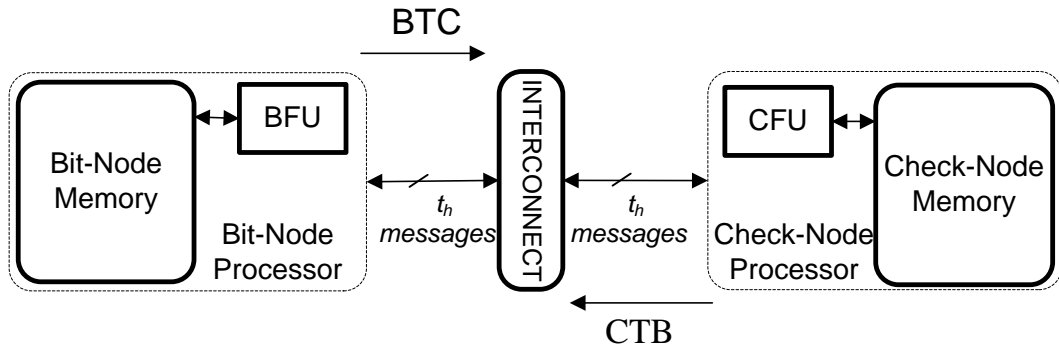


Figure 3.6: An iterative decoder architecture overview. The interconnect network communicates  $t_h$  messages per clock cycle in either direction.

A vast literature exists on the design and implementation of efficient decoder architectures, e.g. [61, 70–76]. The research in this field has evolved towards including key optimizations and design guidelines. For sake of a clear overview, these

optimizations will be discussed in light of the hardware-efficiency challenges they aim to resolve:

1. **Efficient Memory Organization and Interconnect:** An interconnect network is required to exchange the messages between the check-node processor and the bit-node processor. In partially-parallel decoders, the memory plays a role in the exchange mechanism by reading and writing a number of messages,  $t_h$ , per clock cycle. For random or unstructured codes, two complications arise: first, there exists no regular memory-access pattern and, therefore, a conventional  $t_h$ -port memory must be designed, and, second, generating the memory addresses per clock cycle may require complex operations. These complications can be resolved through protograph-based LDPC coding. Memory address generation is simplified due to the fact that the code is described using the much smaller protomatrix along with the corresponding permutations. Besides, efficient memory organization is made possible by having the memory partitioned into several banks, each accessing a small number of messages, much less than  $t_h$ , per clock cycle (see e.g. [77, 78]). Of these protograph-based LDPC codes, the most architecture-aware is a quasi-cyclic code with  $(M_b \cdot Q) \times (N_b \cdot Q)$  parity-check matrix that is composed of  $Q \times Q$  submatrices; each of these submatrices is either an all-zero  $Q \times Q$  matrix or a cyclic-shift of the  $Q \times Q$  identity matrix. The code can be fully described by a  $M_b \times N_b$  integer base matrix, the entries of which are the cyclic-shift offsets, or  $-1$  if the corresponding  $Q \times Q$  submatrix in the parity-check matrix is all-zero. For their simple description and good performance at short to moderate block-lengths, quasi-cyclic codes have been adopted in many systems such as WiMAX (IEEE80.16e) and WLAN (IEEE802.11n). The organization of memory and interconnect follows the scheduling of operations done per one decoding iteration, that is the order in which the edges or nodes of the graph are processed. For sake of illustration, two possible styles of decoding scheduling of a quasi-cyclic code, described by  $M_b \times N_b$  base matrix that is replicated by a factor of  $Q$ , are briefly discussed next. In the first style (e.g. [72]), the check-node processor processes  $M_b$  nodes per cycle. Each processed node belongs to a different block row, where a block row is the expansion by  $Q$  of a row in the base matrix. The check nodes are processed in  $Q$  cycles. In the case of TPMP implementation, the bit-node processor processes  $N_b$  nodes per cycle, where each node belongs to a different block column. The bit-nodes are processed in  $Q$  cycles. The bit-node memory is partitioned into  $N_b$ -banks each storing messages of the bit-nodes corresponding to a single column of the base matrix. Similarly, the check-node memory is partitioned into  $M_b$ -banks each storing messages of the check-nodes corresponding to a single row of the base matrix. The

interconnect between the  $N_b$  bit-function units and  $M_b$  check-function units is hardwired or contains limited reconfigurability as the right exchange is ensured through memory access. In the second style (e.g. [73]),  $Q$  messages are exchanged per cycle, corresponding to the edges in a single non-zero sub-matrix of the parity-check matrix. Second, in processing a check-node  $c$  with degree  $deg_c$ , the check node processor receives the bit-to-check messages in  $deg_c$  subsequent clock cycles, and therefore, the corresponding check-function units process these messages serially. As a result of the scheduling style, a possible memory organization is to then to partition the bit(check)-node memory into  $Q$  banks each connected to a bit(check)-function unit. In this case, interconnect is needed to permute the  $Q$  messages according to the corresponding offset in the base matrix. Unlike the first style, the second style allows the decoder to operate efficiently on codes with different dimensions of the base matrix ( $M_b$  and  $N_b$ ). Besides, it is more suitable for the turbo-decoding message-passing algorithm described in the third point on convergence speed.

2. **Memory-Size Requirement/ Operation count and complexity:** Two messages are exchanged along one tanner-graph edge in a single decoding iteration. Therefore, in partially-parallel decoder architectures, a total number of messages that is nearly equal to double the number of edges has to be stored in the bit-node processor (bit-to-check messages) and check-node processor (check-to-bit messages) for later use. Several optimizations are made to reduce this very-high memory size requirement. The memory requirement is first cut be nearly half by making use of the following observation: the bit-to-check message sent from bit-node  $v$  to check-node  $c$ , in iteration  $\tau$ , can be retrieved as such:

$$BTC_{vc}[\tau] = (\Lambda_v + \sum_{c' \in \mathcal{C}_v} CTB_{c'v}[\tau - 1]) - CTB_{cv}[\tau - 1].$$

where  $\Lambda_v$  is the intrinsic LLR value of bit-node  $v$ . Therefore, instead of storing both check-to-bit and bit-to-check messages, it is enough to store the check-to-bit messages and the posterior LLR value of each bit-node  $v$ ,  $\Lambda_v + \sum_{c' \in \mathcal{C}_v} CTB_{c'v}[\tau - 1]$ . The memory-size requirement is further reduced by using the Min-Sum approximation [79] in the check-to-bit message computation as following:

$$CTB_{cv}[\tau] = \left( \prod_{\substack{v' \in B_c \\ v' \neq v}} \text{sgn}(BTC_{v'c}[\tau]) \right) \times \min_{\substack{v' \in B_c \\ v' \neq v}} (|BTC_{v'c}[\tau]|).$$

The min-sum approximation causes a coding-performance degradation; therefore variations of the algorithm with different complexity-performance trade-offs were developed accordingly, such as the the min-sum-plus-correction-factor [80], the normalized min-sum, and the offset-based min-sum [81]. Using the min-sum approximation, the check-to-bit messages resulting from processing check-node  $c$  with degree  $deg_c$  can be retrieved from the following values: 1)  $deg_c$  bits showing the signs of each of  $CTB_{cv}[\tau]$ ,  $v \in B_c$ , 2) the minimum value of  $BTC_{vc}[\tau]$  and the second minimum-value of  $BTC_{vc}[\tau]$ , and 3) the index/position of the minimum value. Thus, it is sufficient for the memory to store these values per one check node  $c$ , instead of storing the  $deg_c$  check-to-bit messages. The min-sum algorithm reduces the memory requirement significantly. It breaks the dependence of the required memory-size on the edge-count, and, along with the aforementioned bit-node memory-saving simplification, makes this requirement dependant solely on the number of nodes in the Tanner graph. Beside memory savings, the min-sum approximation leads to a great reduction in the complexity of check-node processing. Check node processing, applying the exact equations, involves the operations of evaluating the  $\psi(\cdot)$  function,  $\psi(x) = -\log(\tanh(x/2))$ , and adding. In particular, the  $\psi(\cdot)$  function is too complex to be implemented in combinational logic and is either subject to simplifying approximations or implemented as a look-up table, i.e. as a read-only-memory (ROM). The min-sum approximation, on the other hand, involves simple comparison, which clearly leads to a major reduction in the complexity and count of the involved operations. The min-sum approximation is now widely used to design hardware-efficient decoder implementations.

3. **Decoding Convergence Speed:** The decoding convergence speed is measured as the average number of iterations needed by the decoding algorithm to converge to the correct codeword. A higher convergence speed implies higher decoding throughput or, equivalently, lower energy overhead. In regard of this aim, the turbo-decoding message-passing algorithm (TDMP) [82, 83] was proposed to replace the conventional two-phase message-passing (TPMP) algorithm in LDPC decoding. TDMP was originally developed for quasi-cyclic LDPC codes, where the code structural properties make it much simpler to implement; however, its concept can be generalized to any LDPC code. The concept can be illustrated through an example. As a prelude, the following terminology is defined: two check node  $c$  and  $c'$  are called neighbors if they are check-node neighbors to the same bit node, call it  $v$  in this example. Processing of check-node  $c$  in iteration  $i$  produces new check-to-bit messages that are used then to update the bit-to-check messages of the bit-node neighbours of  $c$ . It is in iteration  $i + 1$  that the updated bit-to-

check message of  $v$  will be used in processing check node  $c'$ . Therefore, the extrinsic information obtained from processing node  $c$  is used in processing its neighbor  $c'$  in the subsequent iteration  $i + 1$ , thus slowing the decoding. In TDMP, the results of processing node  $c$  at iteration  $i$ , are then directly forwarded to update the extrinsic posterior values of the bit-node neighbors of  $c$ . Then, if node  $c'$  is processed later in the iteration, its processing will be using the results of node- $c$  processing. This is similar to iterative decoding of turbo codes and thus the name turbo-decoding message-passing algorithm. TDMP can be shown to nearly cut the average number of decoding iterations to half, doubling the convergence speed.

*TDMP timing condition:* In terms of hardware, a timing-condition must be satisfied so that the actual decoding algorithm performance follows its simulated one, assuming floating point implementations. Following the terminology of the example, the condition is that processing each of check nodes  $c$  and  $c'$  must be spread enough in time, that is the posterior LLR value of  $v$  has to be updated using the check-to-bit message from  $c$ , before being forwarded to  $c'$ . In quasi-cyclic codes, this condition can be satisfied, fully or partially, through shuffling the base matrix to control the spacing between non-negative entries in the columns, reordering the processing of submatrices in a block row [73], reordering the rows in a block row [61, 72], and avoiding deep pipelining.

For its speed advantage, turbo-decoding message-passing has become the algorithm of choice in virtually all recently proposed decoder implementations.

The rate-compatibility of a code has a limited impact on the design of the decoder-architecture. This is clear for the punctured LDPC codes where decoding is applied on a single mother-code graph for all rates. In addition, with a decoding scheduling of the second style, the resulting decoder architecture can handle efficiently the variation in the dimensions of the base matrix caused by code extension. However, as pointed out previously in the section, some aspects of the design space can be restricted such as shuffling the rows of the base matrix and the memory organization.

The case of Raptor decoding is different in the following sense: the distinct features of the Raptor codes, which can collectively be used to resolve the problems raised in 3.4.3, lead themselves to new challenges, unknown in the LDPC case, in the design of hardware-efficient Raptor decoders. This issue will be discussed in Chapter 5.

# Chapter 4

## Increment-based Inter-frame Coding

This chapter includes the first part of the research work presented in this dissertation, namely the increment-based inter-frame (IIF) coding scheme. The proposed scheme is intended to be deployed in the broadcast communication scenario, and is motivated by the observation described next. The feedback-based retransmission schemes, such as HARQ, do not scale well as the number of receivers grows as described in Section 3.2. Therefore, these schemes are replaced, in broadcast communication, by a second stage of coding in order to recover from PHY-level intra-frame decoding failures. The introduced stage of coding consists of erasure coding: it involves an erasure channel abstraction, leading thus to a loss in the achievable data-rates. A natural question arises from this observation: *Can a scheme be developed such that, like erasure coding, it scales well as the number of receivers grows, while avoiding the erasure-channel abstraction?*

The proposed increment based inter-frame coding is one such scheme. The scheme incorporates aspects of IR-HARQ to coding. The basic feature of the proposed approach is that *the assignment of the appropriate code-rate to a frame is done on the receiver side*. In particular, the coding scheme is increment-based in the sense that inter-frame encoding is applied on the increments corresponding to the transmitted frames. Inter-frame encoding therefore generates subframes rather than complete frames. Inter-frame decoding proceeds iteratively to recover all the unsuccessfully decoded frames by progressively decreasing their code-rates and retrying channel decoding on them. The *inter-frame* decoding procedure performs iteratively two inter-related operations: 1) *intra-frame* decoding to recover the transmitted frames, and 2) progressive concatenation of increments to frames that are unsuccessfully-decoded prior to retrying *intra-frame* decoding on them.

The main premise of the proposed scheme is that it involves an efficient complexity-performance tradeoff that has two features. The first is the data-rate enhancement

achieved over the state-of-the-art two-stage scheme, and the second is the resemblance of the inter-frame decoding procedure to that of LT erasure decoding [35] stated as: beside intra-frame decoding, inter-frame decoding involves nearly the same scheduling and operations of the iterative LT erasure decoding. The data-rate enhancement is achieved because, unlike the case in the two-stage scheme, an unsuccessfully-decoded frame is not discarded; rather, its code-rate is decreased and intra-frame decoding is reapplied on it. The resemblance of increment-based inter-frame decoding scheduling to that in iterative erasure-decoding is due to the following property: only frames that are successfully-decoded are involved in the process of decreasing the code-rate, or equivalently incrementing the LLR-vector, of the unsuccessfully-decoded frames. It is noteworthy that such property is not confined to inter-frame coding; it is used for example in [84] to design efficient channel codes for the HARQ schemes.

It is, furthermore, shown that the asymptotic coding performance of the proposed inter-frame coding scheme is equal to that of IR-HARQ in unicast communication, under the channel model developed here. However, the proposed inter-frame coding scheme scales much better than IR-HARQ for broadcast communication, i.e. as the number of receivers grows up.

The chapter is divided as shown next. First, the proposed inter-frame encoding and decoding algorithms are presented and analyzed. Second, architectures of the inter-frame encoder and decoder along with the corresponding decoding scheduling are developed. It can be concluded from the developed architectures that: 1) the major hardware units involved in inter-frame decoding are already available in any typical communication system, 2) the decoder can be implemented to support inter-frame code descriptions that are determined in real time, and, 3) in terms of hardware resources, the required memory size is the limiting factor in the design of large inter-frame codes. Third, the channel variation in relation to inter-frame decoding is modeled as a probability distribution over  $\mathbb{Z}^+$ , which is crucial in developing a generic yet simple framework for designing and analyzing inter-frame codes. Fourth, the asymptotic performance of inter-frame coding is analyzed and optimized. A coding-optimality criterion is first defined. Then, the optimality of inter-frame coding, under certain assumptions on the channel model, is proven using a multi-step procedure involving modeling, analysis, and optimal degree-distribution construction. It is then shown that inter-frame coding increases the data rate by a factor that is dependant on the corresponding channel parameters and that can reach  $\sim 1.55\times$ , compared to the two-stage scheme. Compared to the IR-HARQ scheme having target frame-error-rate (FER) of  $10^{-3}$ , inter-frame coding increases the data rate by a factor reaching  $5\times$  when the number of receivers is infinitely large. Fifth, simulations performed using the developed channel model highlight a potential tradeoff between the required memory size and coding per-

formance in the design of small/moderate-size inter-frame codes. They also show significant data-rate enhancements compared to the two-stage scheme, when the number of frames included in inter-frame coding is 121 or 1210.



## 4.1 Algorithms

The inter-frame encoding and decoding procedures are described next.

### 4.1.1 Encoding

Inter-frame encoding generates  $K_S$   $\Delta$ -bit subframes starting from an initial set of  $N_T = N_F$  information blocks. Each block is intra-frame encoded into a  $(N + D \cdot \Delta)$ -bit frame. Generation of the  $K_S$  subframes, illustrated in Fig. 4.1, is done according to a  $K_S \times N_F$  “inter-frame” code generator matrix  $\mathbf{H} = [h(i, j)]$ , where  $h(i, j)$  denotes the entry in row  $i$  and column  $j$  in  $\mathbf{H}$ . Each column of  $\mathbf{H}$  has a maximum of  $D$  nonzero distinct positive integer entries that are less than  $D + 1$ . Subframe  $s$ ,  $s \leq K_S$ , is formed using row  $s$  of  $\mathbf{H}$ , as follows:

*Initialize subframe  $s$  to a  $\Delta$ -bit zero vector. For each nonzero entry  $h(s, f) = a$  of  $\mathbf{H}$ , update subframe  $s$  to the output vector formed by bit-wise XORing of subframe  $s$  with the increment  $\Delta(f, a)$ .*

The rate- $R_H$   $N$ -bit portion of each frame  $f \leq N_F$  and the  $K_S$   $\Delta$ -bit subframes are then transmitted over the channel, for a total of  $(N_F \cdot N + K_S \cdot \Delta)$  bits. The details of the transmission scheduling will not be considered in this dissertation. Frame  $f$  and subframe  $s$  are said to be neighbors if  $h(s, f) \neq 0$ . The degree of frame  $f$  is defined as the number of non-zero entries in column  $f$  of  $\mathbf{H}$ . The degree of subframe  $s$  is defined as the number of non-zero entries in row  $s$  of  $\mathbf{H}$ .

### 4.1.2 Decoding

**Input:** The input to inter-frame decoding consists of  $\mathbf{H}$  and the intrinsic LLR values of the  $N_F$   $N$ -bit frames  $\Lambda_N(f)$ , for  $f \leq N_F$ , and LLRs of the  $K_S$   $\Delta$ -bit subframes  $\Lambda_S(s)$ , for  $s \leq K_S$ .

**Output:** The output includes an  $N_F$ -bit flag vector indicating decoding success (recovery), or failure of the  $N_F$  frames. For each successfully-decoded frame, the corresponding  $K$  information bits are recovered.

The decoding procedure, illustrated in the flowchart in Fig. 4.2, applies iteratively two main steps. First, when intra-frame decoding of a frame  $f$  succeeds, the corresponding increments  $\Delta(f, i)$ ,  $i = 1, \dots, d$ , are recovered. For every subframe  $s$  such that  $h(f, s) = a \neq 0$  (i.e., subframe  $s$  is equal to  $\Delta(f, a) \oplus \Psi$  where  $\Psi$  is a XOR of other increments),  $\Lambda_S(s)$  is updated so that it corresponds to  $\Psi$ . Effectively, this is equivalent to removing  $\Delta(f, a)$  from the list of inputs to the XOR operation forming the transmitted subframe  $s$ . Second, if the number of inputs to  $s$  is effectively reduced to one increment  $\Delta(f', b)$  for some  $b \leq D$ ,  $f' \leq N_F$ , then  $\Lambda_S(s)$  corresponds now to  $\Delta(f', b)$ .  $\Lambda_S(s)$  is then concatenated to the LLR vector

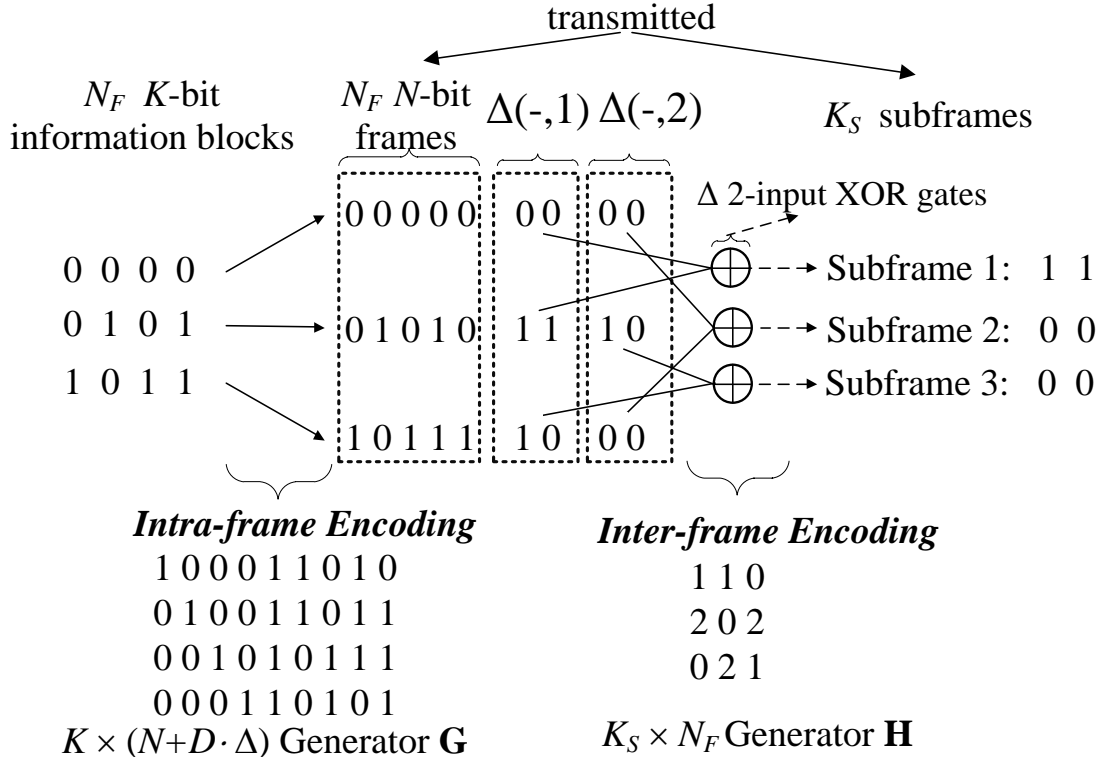


Figure 4.1: Inter-frame encoding example: Generating  $K_S = 3$  subframes from  $N_F = 3$  frames, with parameters  $(K, N, \Delta, D) = (4, 5, 2, 2)$ .

of frame  $f'$ , decreasing its code-rate. If decoding of frame  $f'$  has failed prior to this concatenation, frame  $f'$  is rescheduled for intra-frame decoding.

The decoding procedure is detailed next. For simplicity of exposition, it is reasonably assumed that each row of **H** has at least two non-zero entries.

**Procedure:** Three disjoint index sets  $\mathcal{R}$ ,  $\mathcal{U}$ ,  $\mathcal{P}$  are formed, where  $\mathcal{R}$  includes the indices of the successfully-decoded frames,  $\mathcal{U}$  the indices of the unsuccessfully-decoded frames, and  $\mathcal{P}$  the indices of the frames on which decoding will be applied/reapplied. Both  $\mathcal{U}$  and  $\mathcal{R}$  are initialized to  $\emptyset$  and  $\mathcal{P}$  to  $\{1, \dots, N_F\}$ . The inter-frame decoding procedure is:

If  $\mathcal{P} = \emptyset$ , quit decoding, else, pick randomly an element  $f$  from  $\mathcal{P}$ . Apply intra-frame decoding on the vector of LLR values corresponding to frame  $f$ . If intra-frame decoding is unsuccessful, move  $f$  to  $\mathcal{U}$  and go back to the beginning of the step. If it is successful, proceed as follows:

1.  $f$  is moved from  $\mathcal{P}$  to  $\mathcal{R}$ .
2. The  $K$  information bits corresponding to frame  $f$  are recovered and used through intra-frame encoding to obtain the increments  $\Delta(f, k), k = 1, \dots, D$ ,

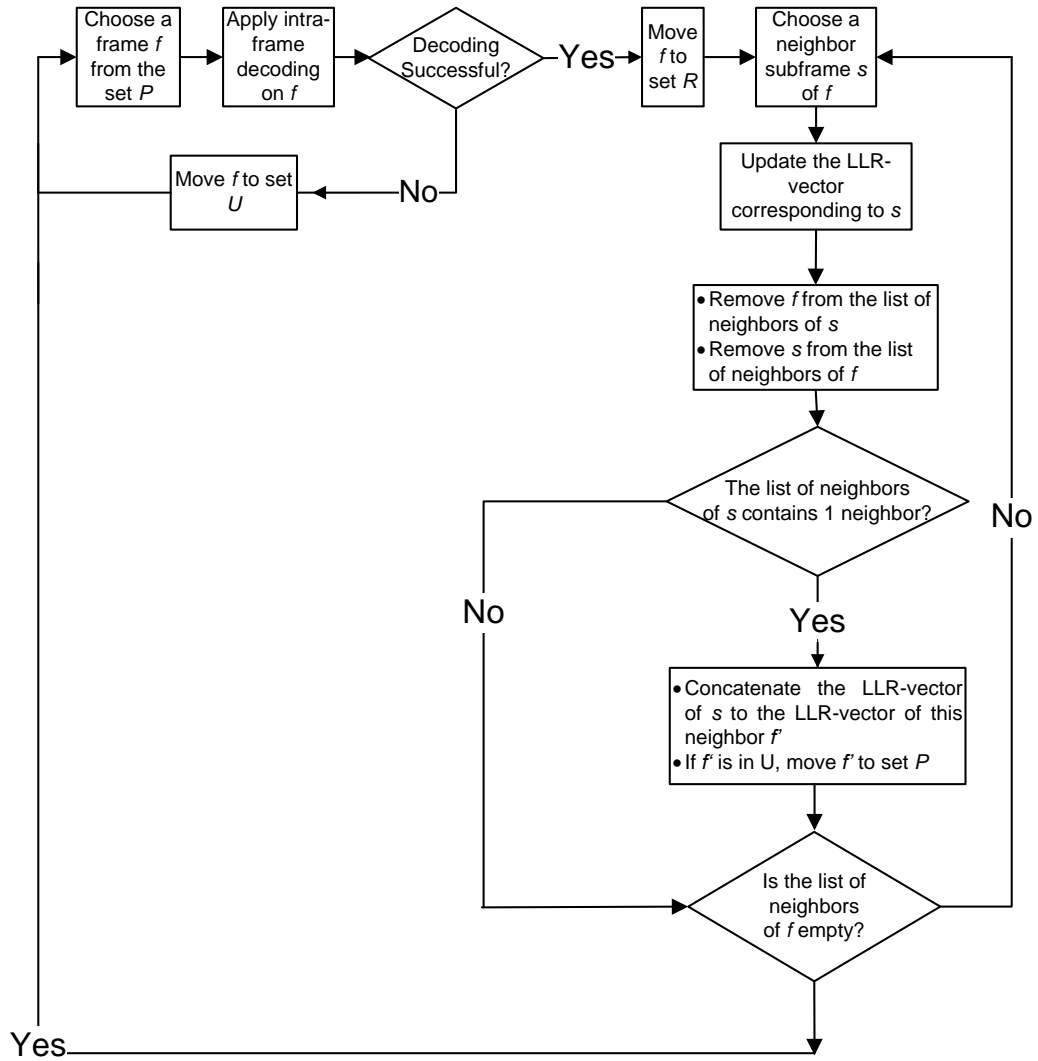


Figure 4.2: A flowchart showing the basic steps of inter-frame decoding.

corresponding to frame  $f$ .

3. For every  $s$  where  $h(s, f) = a \neq 0$ :
  - a. The LLR vector corresponding to subframe  $s$ ,  $\Lambda_S(s)$ , is updated:  $\Lambda_S(s) \leftarrow \Lambda_S(s) \times (-1)^{\Delta(f,a)}$ , where the  $m$ th entry of  $(-1)^{\Delta(f,a)}$  is  $-1$  if the  $m$ th bit of  $\Delta(f, a)$  is 1, and 1 otherwise, and  $\times$  is entry-wise multiplication.
  - b. Set  $h(s, f)$  to 0.
  - c. If the number of nonzero entries of row  $s$  of  $\mathbf{H}$  reduces to 1, where  $h(s, f') = b > 0$ , then: 1) if  $f' \in \mathcal{U}$ ,  $f'$  is moved to  $\mathcal{P}$ , and 2)  $\Lambda_S(s)$ , now equal to  $\Lambda_{\Delta}(f', b)$ , is concatenated to the LLR vector of  $f'$ . By abuse of notation,  $\Delta(f', b)$ , or equivalently subframe  $s$ , is said to be concatenated to frame  $f'$ .

Step 3 is invoked only when the corresponding frame  $f$  is successfully decoded. This has two implications. First, the LLRs corresponding to the decoded frame  $f$  are now set to  $\pm\infty$ , with the sign depending on the hard-decision of the intra-frame decoding process of the frame; therefore the LLR update in step 3a is simplified into changing signs of the LLR-vector  $\Lambda_S(s)$ , equivalent to performing  $\Delta$  2-input XOR operations. Second, step 3a will only be invoked a maximum number of times that is equal to the number of nonzero entries of  $\mathbf{H}$ , which is  $D \cdot N_F$ . Both implications have their exact counterparts in the iterative erasure decoding algorithm where the decoding procedure can be viewed as a series of successive edge-processing steps. In LT decoding [35] for example, one edge in the LT bipartite graph, or equivalently one non-zero entry of the generator matrix, is processed at most once in the whole decoding procedure; besides, this edge processing consists of a number of 2-input XOR operations.

From an algorithmic complexity perspective, the decoding procedure involves the following operations per frame: the XOR-operation (step 3a) is performed  $D \cdot \Delta$  times, the intra-frame encoding procedure in step 2 is performed 1 time, and the intra-frame decoding procedure is performed an average of  $X > 1$  times. Typically, the intra-frame decoding procedure is much more computationally intensive compared to intra-frame encoding and the  $D \cdot \Delta$  XOR operations. The average number of intra-frame decoding attempts per frame, denoted here by  $X$ , is dependent on 1) the intra-frame decoding failure-rate and its change with the concatenation of the increments LLRs, and 2) the criteria with which a frame is selected from  $\mathcal{P}$  for intra-frame decoding. The first factor is a function of the channel and the deployed intra-frame code. The second factor raises the problem of sorting the frames scheduled for intra-frame decoding according to the probability of intra-frame decoding success, given the LLRs corresponding to each frame. Therefore, this problem necessitates developing a method to estimate the probability of decoding success

of a frame without performing the computationally-intensive decoding itself. This problem, however, is outside the scope of this work, and selecting a frame from  $\mathcal{P}$  in the described decoding procedure is done randomly.

Note that even in erasure coding, the average number of intra-frame decoding processes divided over  $N_F$  can be greater than 1. The reason is that while each frame is decoded at most once, the number of transmitted frames  $N_T$  rises due to erasure encoding to  $N_T = \frac{N_F}{R_E} > N_F$ ,  $R_E$  being the erasure code-rate.

Other variations of the proposed coding scheme can be developed, all centered around the idea of progressively decreasing the code-rate of the unsuccessfully-decoded frames and retrying decoding. One such variation is discussed briefly next for illustration.

### 4.1.3 Modified Scheme

**Encoding:** The distinguishing feature of this scheme is that the subframes themselves are also subject to intra-frame encoding. The resulting modification in the inter-frame encoding scheme involves adding a step to the previously described procedure. Assume for simplicity that the fractions  $\frac{K}{\Delta} \triangleq M$  and  $\frac{K_S}{M} \triangleq N_{RED}$  are both integers. The additional step consists of partitioning the  $K_S$  subframes into  $N_{RED}$  sets, each including  $M$  subframes. For every set, the corresponding  $M$  subframes are concatenated to form a  $K$ -bit vector which, in turn, is intra-frame encoded into a  $N$ -bit encoded frame. This encoding is done such that the  $K$  systematic-bits are included in the  $N$ -bit encoding frame. By abuse of notation, the resulting  $N_{RED}$  frames and the original  $N_F$  frames are referred to as *redundant* and *systematic* frames, respectively. The  $N_{RED}$  redundant and  $N_F$  systematic frames are transmitted over the channel.

**Decoding:** The decoding procedure in the previous subsection is modified to make use of the subframe intra-frame encoding step, as follows: first, when a redundant frame is successfully-decoded, the LLRs of the subframes included in it take the values of  $\pm\infty$ . Second, when all the systematic frames involved in forming a subframe are successfully-decoded, this subframe can be recovered as well, and if the redundant frame including it is unsuccessfully-decoded, it can be rescheduled for decoding due to such recovery. Prior to starting decoding, a copy of the generator matrix  $\mathbf{H}$  is created, call it  $\mathbf{H}^{(0)}$ . The  $N_{RED}$  redundant frames are also scheduled for intra-frame decoding and, therefore, their indices  $N_F + 1, \dots, N_F + N_{RED}$  are initially included in the set  $\mathcal{P}$ . The decoding schedule is then modified as follows:

- Case 1: The index  $f$  selected from  $\mathcal{P}$  corresponds to a *redundant* frame, i.e.  $f > N_F$ . If the intra-frame decoding of the chosen frame succeeds,  $f$  is moved to  $\mathcal{R}$  and a 2-step procedure is followed:

1. The LLRs of the corresponding  $M$   $\Delta$ -bit subframes are set to  $+\infty$  or  $-\infty$  depending on the output of the decoding.
2. For each systematic frame  $f' \notin \mathcal{R}$ , i.e. not yet successfully-decoded, and subframe  $s$  included in the redundant frame  $f$  such that  $h(s, f') = a$  is the only non-zero entry in row  $s$  of  $\mathbf{H}$ , the following is done: 1) frame  $f'$  is scheduled for decoding, i.e.  $f'$  is moved to  $\mathcal{P}$ , and 2) the LLR-vector of increment  $\Delta(f', a)$ ,  $\Lambda_{\Delta}(f', a)$ , is set to the updated vector  $\Lambda_S(s)$ .

Otherwise, in case of decoding failure, the LLRs of the  $M$  subframes included in frame  $f$  are set to their intrinsic (channel) values and  $f$  is moved to the set  $\mathcal{U}$ .

- Case 2: The index  $f$  selected from  $\mathcal{P}$  corresponds to a *systematic* frame. The steps (1) through (3) of the previously described decoding procedure are followed, then a new step is added such that:
  - 3.c' If the number of nonzero entries of row  $s$  of  $\mathbf{H}$  reduces to 0, this means all the systematic frames involved in forming the subframe  $s$  are successfully decoded. If the redundant frame including  $s$  is not yet successfully decoded (i.e., its index is not in  $\mathcal{R}$ ) then: 1) subframe  $s$  is recovered using the corresponding systematic frames and  $\mathbf{H}^{(0)}$ , 2)  $\forall 1 \leq i \leq \Delta$ , the  $i$ th value of  $\Lambda_S(s)$  is set to  $-\infty$  if the  $i$ th bit of subframe  $s$  is 1 and to  $+\infty$  otherwise, and 3) the redundant frame including subframe  $s$  is scheduled for intra-frame decoding (i.e. moved to  $\mathcal{P}$ ).

## 4.2 Architectures

In this section, the hardware complexity of the inter-frame coding scheme is investigated by identifying the hardware resources and operations required to apply the scheme at the PHY-layer. Toward this, the major blocks and operations of the inter-frame encoder are briefly identified. Then, a high-level architecture of the inter-frame decoder is developed and the corresponding decoding procedure is mapped into a schedule of operations performed by the different blocks of the architecture. Main conclusions on the overhead of inter-frame coding are then drawn from the developed architecture.

### 4.2.1 Inter-frame Encoder

The inter-frame encoder architecture, shown in Fig. 4.3, is composed of six major blocks: 1) intra-frame encoder, 2) frame memory, 3) subframe memory 4) XOR-logic block, 5) encoding scheduler, and 6) transmission scheduler. The encoding process is performed using these units as described next. The  $N_F$  information blocks are input, in order of their increasing indices, to the intra-frame encoder. For each frame, this encoder outputs the corresponding  $N$ -bit frame and  $D$   $\Delta$ -bit increments.

Assume information block  $f$  is encoded. The output  $N$ -bit frame is buffered in the frame memory until it is transmitted. For every nonzero entry  $h(s, f) = a$  in the  $f$ th column of  $\mathbf{H}$ , the following is done: if  $h(s, f') = 0 \forall f' < f$ , a space is allocated for subframe  $s$  in the subframe memory and the stored vector is set to  $\Delta(f, a)$ ; else if  $\exists f' < f$  such that  $h(s, f') \neq 0$ , then a space has been already allocated to subframe  $s$ , and the corresponding stored vector is updated by XORing it with increment  $\Delta(f, a)$  using the XOR-logic block. If  $h(s, f') = 0 \forall f' > f$ , then this means that the generation of subframe  $s$  is complete and the subframe will be stored in memory until it is transmitted. The encoding scheduler controls and schedules all these operations, including memory allocation, read, write, and update operations. The transmission scheduler schedules the transmission of the formed frames and subframes. Its operation is dependant on the details of the involved communication scheme and, therefore, will not be considered further here.

### 4.2.2 Inter-frame Decoder

The inter-frame decoder, shown in Fig. 4.4, is composed of five major blocks: 1) intra-frame encoder and decoder, 2) LLR-memory, 3) concatenator, 4) XOR-logic block, and 5) decoding-scheduling block. It is assumed here that a successfully-decoded frame is forwarded directly to the upper layers, and therefore no PHY-

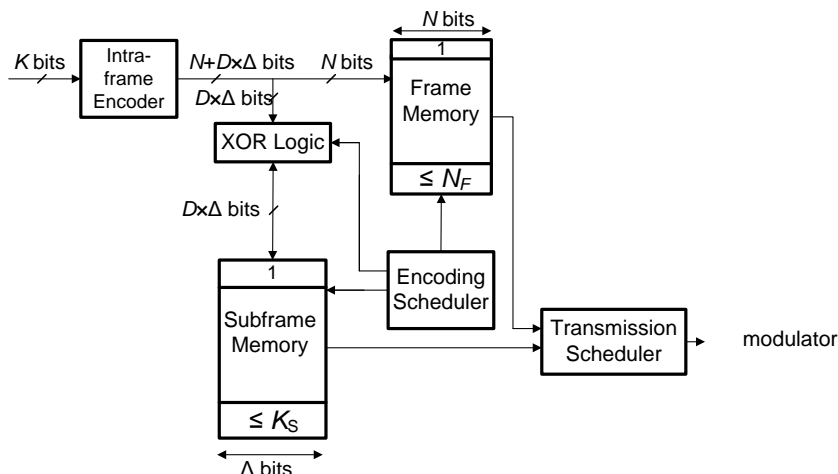


Figure 4.3: A high-level architecture of the inter-frame (a) encoder, and (b) decoder.

layer memory is needed to store the recovered information blocks. The intra-frame decoder is not dedicated solely to inter-frame coding, as it is used to apply channel decoding in all the communication scenarios in which the communication system is involved. The same can be said of the LLR-memory; for example, memory is required to store the LLR values corresponding to unsuccessfully-decoded frames if the system applies the IR-HARQ scheme in unicast communication. The concatenator is needed for IR-HARQ as well, to concatenate the LLRs of the original transmission to those of the successive retransmissions. In comparison, the XOR-logic and the decoding-scheduling blocks are dedicated solely to the inter-frame decoding procedure. Therefore, their hardware-overhead consists a part of the hardware overhead of inter-frame coding. The required number of XOR gates in the XOR-logic block is a function of the target throughput of this block, and is system dependant. An architecture of the decoder scheduling block is, in turn, developed to qualitatively evaluate its hardware-overhead.

The decoding-scheduling block, illustrated in Fig. 4.5, consists of six major units as described next:

1. **Frame managing memory** composed of 4 basic  $N_F$ -row banks, denoted here by FMR, FMD, FMP and FMA. The  $f$ th row of FMR,  $FMR(f)$ , is set to 1 if frame  $f$  has been received and to 0 otherwise.  $FMD(f)$  is set to 1 if frame  $f$  has been successfully-decoded and to 0 otherwise.  $FMP(f)$  stores the number of subframes which LLRs are to be concatenated to  $\Lambda_N(i)$  during intra-frame decoding of frame  $f$ .  $FMA(f)$  stores the memory address of frame  $f$  in the priority list described next. Each of  $FMR(f)$ ,  $FMD(f)$ ,  $FMP(f)$  and  $FMA(f)$



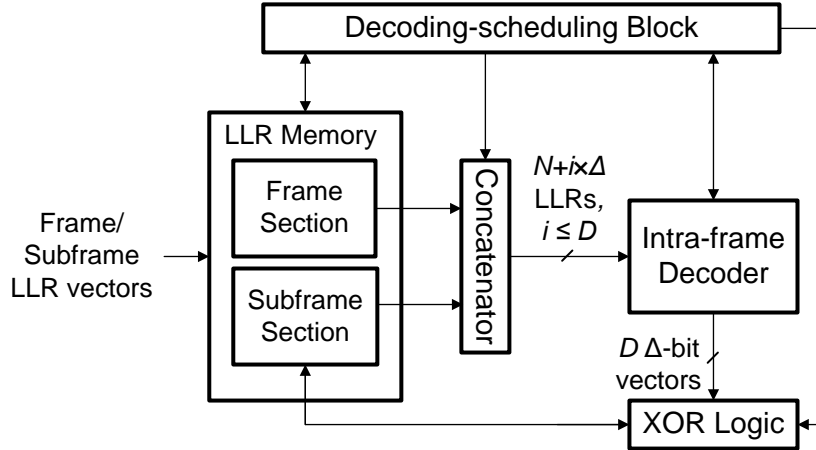
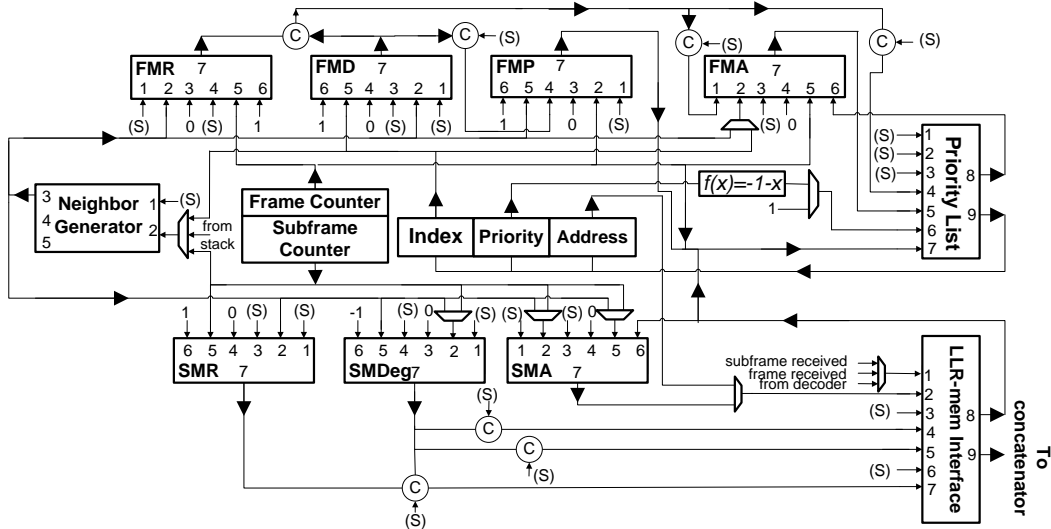


Figure 4.4: A high-level architecture of the inter-frame (a) encoder, and (b) decoder.

is initialized to 0. Each memory bank supports three basic operations: 1) reading a row, 2) writing a row, and 3) incrementing/decrementing the value of a row by 1.

2. **Subframe managing memory** composed of 3  $K_S$ -row memory banks similar to those employed in the frame managing memory, and denoted here by **SMR**, **SMDeg**, and **SMA**.  $\text{SMR}(s)$ , is set to 1 if subframe  $s$  has been received and to 0 otherwise.  $\text{SMDeg}(s)$  stores the number of neighbor frames of subframe  $s$  that are not yet successfully-decoded.  $\text{SMA}(s)$  stores the address of  $\Lambda_S(s)$  in the LLR-memory. Each of  $\text{SMR}(s)$  and  $\text{SMA}(s)$  is initialized to 0, and  $\text{SMDeg}(s)$  to the degree of subframe  $s$ .
3. **Priority list** to store the indices of the frames scheduled for intra-frame decoding (i.e., indices in the set  $\mathcal{P}$ ). Unlike the case in the decoding procedure described in Section 4.1.2, the indices are sorted according to a priority scheme, and the frame of highest priority is chosen first for intra-frame decoding. The priority list supports four basic operations: 1) *reading* the frame index with the highest priority value, 2) *inserting* a new index in the list, 3) *removing* an index from the list, and 4) *updating* the priority of an index in the list. One way to implement such a list is to impose a doubly-linked list structure on a memory bank, where each memory row corresponds to a node in the list and includes the following: a frame index  $f$ , the corresponding priority, the address of  $\Lambda_N(f)$  in the LLR-memory, a right pointer to a node with index of higher or equal priority, and a left pointer to a node



- The triplet (Index, Priority, Address) refers to registers holding these values that are read from the Priority List
- Frame Counter: counts the number of received frames so far
- Subframe Counter: counts the number of received subframes so far
- (S): logical expression involving the state of the control-unit

(C) : logical expression involving the designated inputs

Memory:	Priority List:	LLR-Memory Interface:	Neighbor Generating Unit:
1: Read_Enable	1: Read_Enable	1: Data_In	1: Start_Generation_Enable
2: Read_Address	2: Remove_Enable	2: Address_In	2: Frame/Sbframe_Index_In
3: Write_Enable	3: Insert_Enable	3: Allocate_Enable	3: Neighbor_Index_Out
4: Increment_Enable	4: Update_Enable	4: Free_Enable	4: Entry_value_of_Hg
5: Write_Address	5: Update_Step	5: Update_Enable	5: Last_Neighbor_Out_Flag
6: Data_In	6: Update_Address	6: Write_Enable	
7: Data_Out	7: Data_In	7: Read_Enable	
	8: Inserted_Address	8: Allocated_Address	
	9: Data_Out	9: LLR_Vector_Read	

Figure 4.5: An architecture of the decoding-scheduling block, showing the signal flow during the decoding process. For clarity, some of the blocks and signals are not shown here.

with index of lower or equal priority. For each priority value  $v$ , a pointer is required to store the start and end nodes, of the portion of the linked list that corresponds to indices of priority  $v$ . In this context, a pointer to a node is its address in the considered memory bank. It must be noted that for an efficient implementation of the *update* operation, a pointer to the updated node must be fed to the priority list unit.

4. **Neighbor generator** that serially produces for each input frame  $f$  the following information for every neighboring subframe: its index  $s$ , its  $h(s, f)$  value, and a flag showing whether there exists another neighbor subframe of index  $s' > s$ . The operation is serial in the sense that this information is generated for one neighbor at a time. An analogous operation is performed when the input is a subframe index. The architecture of this unit depends on the construction method of the inter-frame code generator matrices.
5. **LLR-memory** interface, between the LLR-memory and the scheduling block, that performs the following operations: 1) *allocate* a space in the LLR-memory to store the LLR vector corresponding to a frame ( $\Lambda_N(f)$ ) or subframe ( $\Lambda_S(s)$ ), 2) *free* a previously allocated space in the LLR-memory, 3) *read* an LLR vector, 4) *write* an LLR vector, and 5) *update* an LLR vector. The architecture of such interface depends on the organization of the LLR-memory.
6. **Control unit** that controls the operations of the previous units.

Inter-frame decoding can then be mapped into a schedule of operations performed by the various blocks of the inter-frame decoder. This schedule is described next to prove the validity of the developed architecture and to identify the involved operations as well. The exact scheduling depends on the details of transmission of the frames and subframes, as well as on the throughput of the different blocks such as the intra-frame decoder and LLR-memory. Instead, to keep the description clear and general, the decoding schedule assumes the inter-frame decoding time is divided into slots, where in each slot one of the three following events can take place: 1) a new frame of the  $N_F$  frames is received, 2) a new subframe of the  $K_S$  subframes is received, and 3) the intra-frame decoder is idle and therefore a frame can be forwarded to intra-frame decoding. Depending on the “type” of the slot, a series of operations is performed.

In the decoding schedule described next, the priority value of a frame  $f$  is incremented by 1 every time a received neighbor subframe  $s$  has the set of its unrecovered neighbor frames reduced to a singleton  $\{f\}$ . This means that the LLR-vector  $\Lambda_S(s)$  can now be concatenated to the LLR-vector corresponding to frame  $f$  which is to be forwarded to the intra-frame decoder. The priority value

is reset to  $-1$  every time intra-frame decoding of frame  $f$  fails. The decoding scheduling can then be described as follows:

1. If frame  $f$  is received, then: 1)  $\text{FMR}(f)$  is set to 1, 2) a space in the LLR-memory is allocated for  $\Lambda_N(f)$ , and 3) a new node, including the index  $f$  and the address of  $\Lambda_N(f)$  in the LLR-memory, is inserted in the priority list through the *insert* operation. The corresponding priority value is set to  $\text{FMP}(f)$ , and the pointer to the inserted node is written to  $\text{FMA}(f)$ .
2. If subframe  $s$  is received:
  - i)  $\text{SMR}(s)$  is set to 1.
  - ii) If  $\text{SMDeg}(s)$  is nonzero and  $\text{SMA}(s)$  is zero, a space is allocated in the LLR-memory for  $\Lambda_S(s)$  and the address of this space is written to  $\text{SMA}(s)$ . The allocated space is reset to zero vector.
  - iii) If  $\text{SMDeg}(s)$  is nonzero, the LLR vector  $\Lambda_S(s)$  is written in the LLR-memory location which is deduced from the stored value  $\text{SMA}(s)$ , such that: the absolute value of each LLR is written in its allocated position with no change, while the sign of the LLR is XORed with the value of the corresponding memory entry, analogous to step 3a of the decoding procedure in Section 4.1.2. To simplify the notation, this written vector, with updated sign, is still called  $\Lambda_S(s)$ .
  - iv) If  $\text{SMDeg}(s)$  equals 1, then the received subframe  $s$  has only one neighbor frame which is still unrecovered and a search procedure is initiated to find the frame index as explained next. Using the neighbor generator, the frame neighbors of subframe  $s$  are output serially, and it is checked whether each neighbor  $f'$  is already recovered by reading  $\text{FMD}(f')$ : if  $\text{FMD}(f') = 0$ , 1)  $\text{FMP}(f')$  is incremented by 1, and 2) if frame  $f'$  is already received, that is  $\text{FMR}(f')$  equals 1, the priority value corresponding to  $f'$  in the priority list is incremented by 1 using the *update* operation. The pointer to the node corresponding to frame  $f'$  in the priority list is read from  $\text{FMA}(f')$ .
3. If the intra-frame decoder is idle: if the highest priority value in the priority list is greater than  $-1$ , steps (i-vi) are performed, else only step (vi) is performed.
  - i) The frame index of the highest priority,  $f$ , is read from the priority list through the *read* operation.  $\Lambda_N(f)$  is then read from the LLR-memory. As mentioned in the description of the priority list, the address of  $\Lambda_N(f)$  in the LLR-memory is included in the read node.

- ii) A procedure is initiated to find the received subframes having frame  $f$  as their only unrecovered neighbor, and then concatenate their corresponding LLR vectors to that of  $f$ , as explained next. Using the neighbor generator, the subframe neighbors of  $f$  are output serially and for each subframe neighbor  $s$ , the  $\text{SMR}(s)$  and  $\text{SMDeg}(s)$  values are read: if both  $\text{SMR}(s)$  and  $\text{SMDeg}(s)$  equal 1, the LLR vector  $\Lambda_S(s)$  is read from the LLR-memory. To do so, the value  $\text{SMA}(s)$  storing the address of  $\Lambda_S(s)$  in the LLR-memory is first read.  $\Lambda_S(s)$  is then concatenated to the LLR-vector corresponding to frame  $s$ , using the concatenator block.
- iii) The LLR vector corresponding to frame  $s$  is forwarded to intra-frame decoding.
- iv) If decoding fails, the priority value in the node corresponding to frame  $s$  in the priority list is reset to  $-1$ , and the list is updated accordingly using the *update* operation.
- v) If decoding succeeds, the  $(N + D \cdot \Delta)$ -bit encoding frame  $f$  is obtained by re-applying intra-frame encoding on the recovered information block  $f$ . Then, the following procedure is done:
  - a. The node including  $f$  is removed from the priority list using the *remove* operation. The space in the LLR-memory reserved for  $\Lambda_N(f)$  is freed.
  - b. Using the neighbor generator, the subframe neighbors of the recovered frame  $f$  are output serially. For each neighbor,  $s$ , the following steps are performed. First,  $\text{SMDeg}(s)$  is decremented by 1. Second, if  $\text{SMDeg}(s) = 0$  and  $\text{SMA}(s) \neq 0$ , the space allocated to  $\Lambda_S(s)$  in the LLR-memory, and which address is stored in  $\text{SMA}(s)$ , is freed. Third, if  $\text{SMA}(s) = 0$  and  $\text{SMDeg}(s) > 0$ , this means that subframe  $s$  is not yet received and that no space in the LLR-memory is allocated to  $\Lambda_S(s)$ . Therefore, a space is allocated to  $\Lambda_S(s)$  in the LLR-memory and the address is stored in  $\text{SMA}(s)$ . The allocated space is reset to zero vector. Fourth, the sign vector of  $\Lambda_S(s)$  stored in the LLR-memory is updated by XORing it with  $\Delta(f, a)$ ,  $a = h(s, f)$ , analogous to step 3a of the decoding procedure in 4.1.2. Fifth, if  $\text{SMDeg}(s) = 1$  and  $\text{SMR}(s) = 1$ , then only one of the neighbor frames of subframe  $s$  is still unrecovered. The subframe index  $s$  is added to a stack  $\text{ST}$  for processing in the following step.
  - c. For each subframe index  $s$  in stack  $\text{ST}$ ,  $s$  is removed from  $\text{ST}$  and a search procedure is initiated to find the index of its unique unrecovered neighbor as such: using the neighbor generator, output the frame neighbors of subframe  $s$ , and for each neighbor  $f'$  read

$\text{FMD}(f')$ : if  $\text{FMD}(f') = 0$ , then frame  $f'$  is the unrecovered neighbor. Consequently, two steps are done: 1)  $\text{FMP}(f')$  is incremented by 1, and 2) if frame  $f'$  is already received, that is  $\text{FMR}(f') = 1$ , the priority value of  $f'$  in the priority list is incremented by 1 using the *update* operation.

- vi) If the number of received frames equals  $N_F$ , the number of received subframes equals  $K_S$ , and the highest priority value in the priority list is  $-1$ , terminate the inter-frame decoding procedure.

### 4.2.3 Discussion

Based on the description of the algorithms and architectures in the current and previous sections, a number of conclusions can be made. These conclusions are kept as general as possible, in accordance with the high-level description of the architecture and the generic nature of the problem defined in this dissertation.

First, the major blocks of the decoder architecture are not dedicated solely to inter-frame decoding. These blocks are the intra-frame decoder, concatenator and LLR-memory. However, these blocks should be designed to handle efficiently some phenomena that are exclusive to inter-frame decoding. An example phenomenon is the out-of-order increment concatenation, described here by considering intra-frame decoding of a frame  $f$  of length  $N + x \cdot \Delta$ . In IR-HARQ, such frame is typically formed by the initial transmission of the corresponding  $N$ -bit frame followed by the transmission of increments  $\Delta(f, a)$ ,  $a = 1, \dots, x$ . In comparison, in inter-frame decoding the frame can correspond, albeit with different probabilities, to the initially-transmitted  $N$ -bit frame and any of the possible combinations of  $x$  subframe neighbors of  $f$ . These  $x$  subframes have  $f$  as their only unrecovered neighbor.

Second, besides intra-frame encoding/decoding, the computation involved in inter-frame encoding/decoding consists of applying  $D \cdot \Delta$  XOR operations. For typically deployed channel codes such as turbo and LDPC codes, the complexity of these operations is far surpassed by the that of intra-frame decoding.

Third, memory-size is a limiting factor in the design of inter-frame codes with relatively large  $N_F$ . As discussed in 4.2.1, during inter-frame encoding, space is allocated to subframe  $s$  if it at least one of its neighbor frames is already encoded. If, in addition, at least one of its neighbor frames is not yet encoded, the generation of the subframe is not accomplished, and therefore it cannot be transmitted. The memory-size factor is more prominent in inter-frame decoding where vectors of LLR values rather than bits are stored. The LLR-memory is used to store the LLRs of the received frames that are not yet successfully-decoded and the LLRs of the subframes that have unrecovered neighbor frames. Since the LLR-memory

is not dedicated to inter-frame coding, the memory overhead of this coding is measured in terms of the increase in the required memory-size, rather than the memory-size itself, resulting from implementing the inter-frame coding scheme. It is noteworthy that a possible approach to extend the LLR-memory size beyond the available PHY-layer memory size is a cross-layer memory management/allocation between the PHY-layer and the upper layers.

Fourth, the operation of the decoding-scheduling block is serial in the sense that operations corresponding to at most one frame and one subframe are done concurrently. An implicit assumption underlies such serial mode, which is that the decoding-scheduling block does not cause a bottleneck in the throughput of inter-frame decoding; in other words, the operations performed by the units of the scheduling block<sup>1</sup> consume a relatively tiny proportion of the time span of the whole inter-frame decoding process, the latter spanning the reception of the  $N_F$  frames and  $K_S$  subframes and the application of intra-frame decoding on the frames. This can be illustrated by examining steps (1)-(3) of the decoding schedule. Each of these steps can be viewed as a sequence of operations which number, on average, is proportional to the maximum frame or subframe degree. Each operation, in turn, involves reading, comparing, and writing a limited number of memory bits. However, the serial-mode of the decoding-scheduling block has to be reconsidered if the time consumed by its operations becomes relatively significant, for example possibly when the throughput of the other blocks of the architecture is very high.

Beside its serial operation mode, the decoding-scheduling block has the related property that the organization and operation-scheduling of the block is independent of the inter-frame code generator matrix design. A significant implication on the flexibility of inter-frame coding is that the developed architecture can support inter-frame codes which size-parameters,  $N_F$  and  $K_S$ , and generator matrix  $\mathbf{H}$  can be set in real-time. This is true under three reasonable conditions: 1) the size-parameters are less than the maximum sizes supported by the units of the decoding-scheduling block, such as the frame-managing memory and subframe-managing memory, 2) the resulting inter-frame code does not cause LLR-memory overflow, and 3) the neighbor generator can be reconfigured to match any of the possible generator matrices  $\mathbf{H}$ .

Another observation on the decoding-scheduling block is that most of its units have a memory structure; this includes the frame-managing memory, the subframe-managing memory, the priority list, and possibly the neighbor generator. To reduce the hardware resources allocated exclusively to inter-frame decoding, these units can be realized using the PHY-layer memory, part of which is already used to realize the LLR-memory. Three consequences must be considered in this regard:

---

<sup>1</sup>These operations exclude all LLR-memory operations such as space allocation, reading, writing, etc.

first, the operational-throughput of the decoding-scheduling block will deteriorate, which can be overlooked under the assumption that the block does not cause a bottleneck in the throughput of inter-frame decoding. Second, the operation of the PHY-layer memory must be compatible with the memory-access patterns of the decoding-schedule described in 4.2.2, possibly complicating its design. Third, the size of the available LLR memory will be reduced by that of the memory portion dedicated to the decoding-scheduling unit, throughout inter-frame decoding.



### 4.3 Channel Model

Beside complexity, the coding performance has to be considered when evaluating the proposed inter-frame coding. The coding performance of an inter-frame code, and therefore the progress of the corresponding decoding process, is clearly dependant on a number of inter-related factors including the channel characteristics, the intra-frame code/decoder structure and the transmission scheduling. The analysis of each factor in itself is quite involved, and, besides, a multitude of possible combinations of these factors exists. Therefore, obtaining a comprehensive result on the performance of an inter-frame code is challenging.

In this dissertation, this challenge is resolved by setting some simplifying assumptions on the intra-frame code, channel, and communication scenario. Consequently, the behavior of the channel is relation to the inter-frame decoding process is abstracted into a new model. The model is general enough to encompass different communication scenarios. Yet, its simplicity allows efficient simulation and analysis of the inter-frame decoding process. The assumptions underlying the proposed model are stated in the following:

*Assumption 1:* For any integer  $m \geq 0$ , and any set of  $m$  distinct increments, the performance of the code formed by concatenating the original  $N$ -bit frame to the  $m$  increments matches that of a conventional code of rate  $\frac{K}{N+m\Delta}$ . That is, the intra-frame decoding-performance is sensitive to the number but not the indices of the concatenated increments. In practice, it can be formulated as a design requirement of the intra-frame code that the aforementioned sensitivity to the indices of the concatenated increments is made as small as possible. This design problem is not trivial given the observation in [33] on the different sensitivity of the turbo decoding-performance to the LLRs of different portions of the code.

*Assumption 2:* Given the channel statistical model, the channel-states over which different frames are transmitted are assumed independent instances. Two observations underly this assumption. First, the broadcast transmission may not be contiguous in time thus leading to uncorrelated channel-states across different transmissions. Second, for most reasonable cases, channel correlation does not affect the asymptotic performance, i.e. as  $N_F \rightarrow \infty$ , of inter-frame coding. Therefore, in practice, the correlation problem can be approached by increasing  $N_F$  and, if needed, applying interleaving on matrix  $\mathbf{H}$ . It should be noted that both measures, however, can result in increasing the number of LLR vectors that have to be buffered in the LLR-memory, aggravating thus the LLR-memory size factor.

*Assumption 3:* The number of increments required to be concatenated to a frame  $f$  for successful decoding is determined, solely, by the channel-state when the  $N$ -bit portion of  $f$  is transmitted. This assumption ignores the channel-state variations when the subframes are transmitted. It can be partially justified by two facts. First, the number of increments required to be concatenated is discrete

so it does not always follow channel-state variations. This observation is particularly relevant when the original  $N$ -bit frame is transmitted over good channel-state conditions such that, most probably, very few concatenated increments would be sufficient for successful decoding. It is illustrated in the following scenario: when the subframes, concatenated to  $f$ , are sent over some channel conditions,  $1.5 \cdot \Delta$  extra LLR values need to be concatenated to  $\Lambda_N(f)$  to achieve intra-frame decoding success; under better conditions the number is  $1.2 \cdot \Delta$ , and under worse conditions it is  $1.8 \cdot \Delta$ . All three cases will result in the number of required increments being 2. The second fact is relevant when the original  $N$ -bit frame is transmitted over bad channel conditions needing, therefore, a relatively large number of concatenated increments for successful decoding. The corresponding subframes are, most probably, transmitted over independent channel instances, and therefore the overall channel behaviour during their transmission can be well approximated from the statistical description of the channel. The number of increments required to be concatenated to  $f$  for successful decoding can then be deduced accordingly.

**Channel model:** In regard of these assumptions, a received frame  $f$  is characterized by an integer,  $\kappa(f) \geq 0$ , defined as the number of LLR increments, each  $\Delta$ -LLR wide, that should be concatenated to  $\Lambda_N(f)$  for successful decoding of frame  $f$ . Therefore, the channel behavior in inter-frame decoding can be modeled as follows: *it is characterized by a probability distribution over  $\mathbb{Z}^+$ ,  $(\delta_\omega)_{\omega \geq 0}$ , where  $\delta_\omega$  is the probability that a frame  $f$  transmitted over the channel has its  $\kappa$ -value  $\kappa(f) = \omega$ .* To deduce  $(\delta_\omega)_{\omega \geq 0}$ , the following procedure is repeated several times: First, the transmission of an  $N$ -bit frame  $f$  over the channel is simulated, then the frame is decoded. If decoding is successful, the corresponding  $\kappa$ -value  $\kappa(f)$  is set to 0. Otherwise, the following action is repeated, for  $k = 1, \dots, D$ , and stopped when decoding is successful: simulate the transmission of increment  $\Delta(f, k)$  over the channel, and then concatenate the  $(N + (k - 1) \cdot \Delta)$ -LLR vector from the previous iteration to  $\Lambda_\Delta(f, k)$ ; decoding is applied on the formed LLR-vector, and if it is successful,  $\kappa(f)$  is set to  $k$ . The distribution  $(\delta_\omega)_{\omega \geq 0}$  can be deduced accordingly.

The channel model developed simplifies the analysis and design of inter-frame codes. The success of intra-frame decoding of a frame  $f$  is equivalent to having  $\kappa(f)$  less than or equal to the number of subframe neighbors of  $f$  that have  $f$  as their only unrecovered neighbor. Consequently, the outcome of inter-frame decoding of  $N_F$  frames depends solely on the corresponding  $N_F$   $\kappa$ -values and the binary representation  $\mathbf{H}_b = [h_b(i, j)]$  of  $\mathbf{H}$ , where  $h_b(i, j)$  is set to 1 if  $h(i, j) \neq 0$  and 0 otherwise. The inter-frame code construction problem is thus that of matching the distribution(s)  $(\delta_\omega)$  to the appropriate  $\mathbf{H}_b$ .

## 4.4 Mathematical Characterization

In this section, the improvement brought by inter-frame coding to the achievable data rates is quantitatively studied. This is done through a mathematical characterization of the coding-performance of the inter-frame coding scheme, followed by comparing this performance to that of the other conventional schemes in the next section.

The *coding-performance* of a scheme is measured in terms of the resulting *effective frame-length* defined as: the average number of bits, per frame, that needs to be transmitted such that probability that the receiver fails to recover the sent information is below some target value. By its definition, the effective frame-length can be viewed as a reciprocal of the data-rate value, and thus studying one is equivalent to studying the other.

In this section, an additional assumption is made on the distribution  $(\delta_\omega)$ , which is that  $\delta_{\omega+1} = \mu \cdot \delta_\omega$ ,  $\omega \geq 1$ , for some parameter  $\mu < 1$ . This means that  $(\delta_\omega)$  is fully described by two parameters,  $\mu$  and  $\delta = \sum_{\omega>0} \delta_\omega$ , as such:  $\delta_0 = 1 - \delta$  and  $\delta_\omega = \delta \cdot (1 - \mu) \cdot \mu^{\omega-1}$  for  $\omega \geq 1$ . This geometric progression assumption is motivated by two factors. First and most importantly, it makes it possible to derive a compact and relatively simple analytic description of the outcome of inter-frame decoding in subsection 4.4.3. This, in turn, makes it easier to construct compact descriptions of optimal degree-distributions. As in other fields of mathematics, the underlying rationale is that finding a solution to this special case, characterized by the relation  $\delta_{\omega+1} = \mu \cdot \delta_\omega$ , gives an indication on the possible existence of a solution for the general problem. Second, although such assumption does not capture all the possible communication scenarios, it is certainly not unrealistic as illustrated in Fig. 3.1 which shows the frame-error-rate (FER), or equivalently intra-frame decoding-failure rate, versus the length of the transmitted frame. The plotted FER curves are nearly linear, on the logarithmic scale, specifically when the number of concatenated increments is relatively small.

The analysis done on the coding-performance is asymptotic, meaning that it assumes the number of frames,  $N_F$ , included in an inter-frame code or in any other conventional scheme to be infinitely large. The asymptotic nature of the analysis is motivated by three main considerations. First, such analysis provides the performance limit of the inter-frame coding process, as the inter-frame code-size goes to  $\infty$ . One side result is that it becomes possible to quantify the performance degradation caused by constructing inter-frame codes of relatively small sizes. Second, some complications resulting from finite-size inter-frame codes can be efficiently dealt with in the asymptotic design case. For example, channel-state correlation across consecutive frame transmission can be approached by interleaving the frames or including the consecutive frames into different inter-frame codes, both techniques being unrestricted in the asymptotic case by any upper limit on

the inter-frame code-size. Third, as will be clarified in the next paragraph, the asymptotic performance of inter-frame decoding process can be well analyzed and optimized using a small set of compact mathematical expressions and inequalities.

Analyzing the performance of inter-frame coding is done through a four-step procedure that is described next. Step (1) specifies a lower bound on the effective frame-length obtained in inter-frame coding, for a certain channel-state statistical description, and sets the goal of the following steps which is *proving whether such lower bound is achievable by inter-frame coding*. This latter question is called the question on the optimality of inter-frame coding. Answering it is preceded, in step (2), by mapping inter-frame decoding into a two-phase message-passing algorithm applied on a bi-partite graph that corresponds to the inter-frame code. This algorithm is a generalization of the decoding algorithms of the erasure codes developed in [85,86]. This means that the analytical approach, developed in [87] to analyze the asymptotic performance of these erasure codes, can be generalized to describe the performance of the inter-frame decoding process. This generalization is done in step (3), where it is shown that the outcome of the decoding process can be analyzed using a compactly-expressed mathematical function that involves the edge-degree distributions of the bi-partite graph and the channel parameters. Accordingly, optimal codes are designed in step (4) by constructing the appropriate edge degree-distributions, and the optimality of these codes is proven. As a result of this multi-step procedure, the effective-frame length of inter-frame coding, under some channel-state statistical description, is obtained. It is then compared, in Section 4.5, to the effective frame-lengths obtained in the state-of-the-art two-stage scheme and in a simple IR-HARQ scheme devised for the sake of illustration.

It should be noted that, due to the nature of the presented material in this part of the dissertation, the notation is slightly modified for the current and next section (4.5): indexing is made here by subscripting, or rarely subscripting, the index and including it in brackets.

#### 4.4.1 Optimality Question

The first step in analyzing the asymptotic performance of inter-frame coding is done in this section as follows: a bound on the asymptotic performance is set, and consequently a question is formulated on whether such bound is achieved by inter-frame coding. Finding an answer to this question, called here the question on the optimality of inter-frame coding, is then the subject of the following subsections.

The effective frame-length obtained in an inter-frame coding scheme with parameters  $(N_F, K_S, N, \Delta)$ , is equal to:

$$\frac{N_F \cdot N + K_S \cdot \Delta}{N_F} = N \cdot \left(1 + \frac{K_S}{N_F} \cdot \frac{\Delta}{N}\right).$$

Both the  $\Delta$  and  $N$  values are assumed to be predefined, in addition, they are fixed across the proposed and conventional schemes. Finding the effective frame-length attained by inter-frame coding, for a channel-characterizing distribution ( $\delta_{(\omega)}$ ), is then equivalent to finding the minimum value of  $\frac{K_S}{N_F}$  that is sufficient for the inter-frame decoding process to succeed. The latter inter-frame decoding success criterion can be defined in various ways, one of which is considered in the optimality question formulated in the end of the subsection.

The bound set on  $\frac{K_S}{N_F}$  in this section is based on the following observation: a subframe  $s$  is concatenated to at most one frame throughout inter-frame decoding. On the other hand, the minimum number of subframes that should be concatenated to the frames, for successful intra-frame decoding of the  $N_F$  received frames, is  $\sum_{f=1}^{N_F} \kappa(f)$ . Therefore, for the recovery of all the received  $N_F$  frames, the following condition must be satisfied:

$$K_S \geq \sum_{f=1}^{N_F} \kappa(f) \iff \frac{K_S}{N_F} \geq \frac{\sum_{f=1}^{N_F} \kappa(f)}{N_F}.$$

For a channel-characterizing distribution ( $\delta_{(\omega)}$ ),  $\frac{\sum_{f=1}^{N_F} \kappa(f)}{N_F} \rightarrow \sum_{\omega} (\omega \cdot \delta_{(\omega)})$  as  $N_F \rightarrow \infty$ , by the law of large numbers. Overall, the following can be concluded: to recover the  $N_F$  received frames with a probability the goes to 1 as  $N_F \rightarrow \infty$ , a necessary condition is:

$$\frac{K_S}{N_F} \geq \sum_{\omega} (\omega \cdot \delta_{(\omega)}).$$

The resulting lower bound on  $\frac{K_S}{N_F}$  motivates the following question: *Assuming a channel-characterizing distribution ( $\delta_{(\omega)}$ ), are there inter-frame codes that can be defined over increasing values of  $N_F$  such that, as  $N_F \rightarrow \infty$ :  $\frac{K_S}{N_F} \rightarrow \sum_{\omega \geq 1} (\omega \cdot \delta_{(\omega)})$ , and the probability that a randomly-chosen frame is successfully-decoded in the inter-frame decoding process converges to 1.* Such codes are said in this chapter to be optimal because the average number of subframes transmitted per frame converges to the optimal value of  $\sum_{\omega \geq 1} (\omega \cdot \delta_{(\omega)})$ . The existence of such optimal codes means that, for a distribution ( $\delta_{(\omega)}$ ), the minimum *effective frame-length* that is achieved by inter-frame coding is equal to  $N(1 + (\sum_{\omega} \omega \cdot \delta_{(\omega)}) \cdot \frac{\Delta}{N})$ . The existence of optimal codes is the subject of the rest of the section.

## 4.4.2 Bipartite Graph Model

The second step in the analysis of the performance of inter-frame coding is mapping the corresponding decoding process into a two-phase message-passing algorithm applied on a bi-partite graph. This mapping has two motivations. First, the two-phase message-passing algorithm is suitable for asymptotic analysis. Second,

the developed message-passing algorithm is a generalization of the algorithm corresponding to erasure decoding of LDPC codes and of the codes described in [85,86]. This means that the analysis and design techniques of optimal erasure codes, presented in [85–87], can be used as a starting point for the analysis and design of optimal inter-frame codes, as is seen in the next subsections.

The inter-frame code can be described using a bi-partite graph as illustrated in Fig. 4.6. The first partition  $\mathcal{V}$  includes  $N_F$  variable nodes, each corresponding to a frame, while the second partition  $\mathcal{C}$  includes  $K_S$  check nodes, each corresponding to a subframe. By abuse of notation,  $v/c$  is used to index the corresponding frame/subframe. An edge exists between variable  $v$  and check  $c$  if the corresponding frame and subframe are neighbors;  $v$  and  $c$  are neighbor nodes. The degree of a node is the number of edges connected to it. For each set of  $N_F$  received frames, variable-node  $v$  is associated to a value  $\kappa(v)$ , sampled from  $(\delta_{(\omega)})$ .

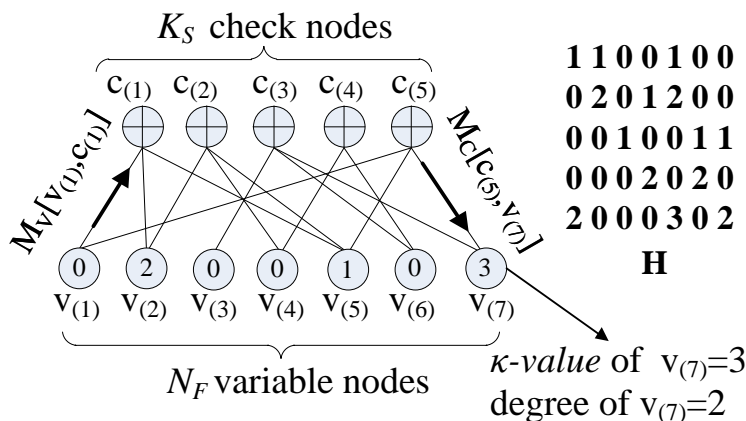


Figure 4.6: Bipartite graph corresponding to an inter-frame code. Each variable node is indexed by its  $\kappa$ -value. Note that since the  $\kappa$ -value of node  $v_{(7)}$  is greater than its degree, frame 7 cannot be recovered.

The inter-frame decoding progress, for a given set of  $N_F$   $\kappa$ -values associated with the received frames, is then mapped to a two-phase message passing algorithm applied on the bi-partite graph. This algorithm, denoted here as the decoding-on-graph procedure, is described in Algorithm 1. It does not recover the frames; rather, it mimics the progress of the actual inter-frame decoding process to conclude what frames will be successfully-decoded throughout this process. The algorithm is iterative, where two binary messages are exchanged along each edge of the bi-partite graph, per iteration. These two messages are the variable-to-check and check-to-variable messages. A check-to-variable message from check  $c$  to variable  $v$ ,  $M_C[c, v]$ , is set to 1 to indicate that the other neighbor frames of subframe  $c$  are recovered and that subframe  $c$  can be concatenated to frame  $v$ . A variable-to-check

message from  $v$  to  $c$ ,  $M_V[v, c]$ , is set to 1 if the number of 1-valued check-to-bit messages sent from all neighbors of  $v$ , excluding  $c$ , to  $v$  exceeds  $\kappa_{(v)}$ . This means that the number of subframes, excluding subframe  $c$ , that can be concatenated to  $v$  exceeds the necessary  $\kappa_{(v)}$ . On the other hand, the frame  $v$  is recovered, or equivalently  $U_{(v)}$  defined in Algorithm1 is set to 1, if the number of 1-valued check-to-bit messages sent from all neighbors of  $v$  to  $v$  exceeds  $\kappa_{(v)}$ .

---

**Algorithm 1** Decoding-on-Graph Algorithm

---

**Input:** Bipartite graph  $(\mathcal{V}, \mathcal{C})$ ; Vector  $\kappa \in \mathbb{N}^{N_F}$   
 $\forall v \in \mathcal{V}, \mathcal{N}_{(v)}$ : set of check neighbours of variable  $v$ ;  
 $\forall c \in \mathcal{C}, \mathcal{N}_{(c)}$ : set of variable neighbours of check  $c$ ;  
**Output:** Successful Decoding Vector  $U \in \{0, 1\}^{N_F}$  (1: decoding success, 0 decoding failure);  
**Procedure:**  
 $U \leftarrow (\kappa == 0)$   
**for**  $v = 1$  to  $|\mathcal{V}|$  **do**  
    **for**  $c \in \mathcal{N}_{(v)}$  **do**  
         $M_V[v, c] \leftarrow (\kappa_{(v)} == 0)$        $\triangleright$  variable-to-check message initialization  
    **end for**  
**end for**  
**for** iterations = 1 to  $MaxIter$  **do**  
    **for**  $c = 1$  to  $|\mathcal{C}|$  **do**  
        **for**  $v \in \mathcal{N}_{(c)}$  **do**  
             $M_C[c, v] \leftarrow \prod_{\substack{v' \in \mathcal{N}_{(c)} \\ v' \neq v}} M_V[v', c]$        $\triangleright$  check-to-variable  
        **end for**  
    **end for**  
    **for**  $v = 1$  to  $|\mathcal{V}|$  **do**  
        **for**  $c \in \mathcal{N}_{(v)}$  **do**  
             $M_V[v, c] \leftarrow \left( \kappa_{(v)} - \sum_{\substack{c' \in \mathcal{N}_{(v)} \\ c' \neq c}} M_C[c', v] \leq 0 \right)$        $\triangleright$  variable-to-check  
        **end for**  
    **end for**  
    **for**  $v = 1$  to  $|\mathcal{V}|$  **do**  
         $U_{(v)} \leftarrow \left( (\kappa_{(v)} - \sum_{c' \in \mathcal{N}_{(v)}} M_C[c', v]) \leq 0 \right)$        $\triangleright$  frame recovery flag  
    **end for**  
**end for**

---

The proof that this algorithm determines accurately the result of the actual inter-frame decoding process is intuitive, but the details are tedious. For sake of brevity, only the outline of the proof is sketched here.

*Proof.* (Sketch) As a prelude, the following *persisting 1-message* property can be proved: if a message, whether variable-to-check or check-to-variable, takes the value 1 in iteration  $i$ , it will pertain this value of 1 in the following iterations  $i + 1, \dots$ .

In the first part of the proof, it is shown that if a frame  $v$  is not recovered in the actual inter-frame decoding process, then  $U_{(v)}$  is always set to zero in all the iterations of the decoding-on-graph process. This can be proven by induction: assuming it is true for iterations  $1, \dots, i^*$ , it is proven to be true for iteration  $i^* + 1$  as well. Now, assume in iteration  $i^* + 1$ , there exists node  $v$  such that  $U_{(v)}$  is set to 1, while the corresponding frame  $v$  is not recovered in the actual inter-frame decoding process. Define  $\mathcal{S}_{(v)}$  to be the set of subframe neighbors of  $v$  which corresponding check-to-variable messages going to  $v$  in iteration  $i^* + 1$  have the value 1. Since  $U_{(v)}$  is set to 1,  $|\mathcal{S}_{(v)}| \geq \kappa_{(v)}$ . Define set  $\mathcal{V}' \subset \mathcal{V}$ ,  $\mathcal{V}' = \{v' \in \mathcal{V}; v' \neq v \text{ and } v' \text{ is a neighbor of } s' \in \mathcal{S}_{(v)}\}$ . For every pair  $(s', v') \in (\mathcal{S}_{(v)}, \mathcal{V}')$ , the variable-to-check message from  $v'$  to  $c'$  has value 1 in iteration  $i^*$ . This implies that  $U_{(v')}$  takes the value of 1 in iteration  $i^* < i^* + 1$  and, by the induction assumption, that every frame  $v' \in \mathcal{V}'$  is recovered. However, since frame  $v$  is not recovered, no subframe  $s \in \mathcal{S}_{(v)}$  is concatenated to any of its neighbor frames in  $\mathcal{V}'$  during inter-frame decoding. Since all the frame neighbors of the subframes in  $\mathcal{S}_{(v)}$  are recovered, all the subframes in  $\mathcal{S}_{(v)}$  can be concatenated to frame  $v$ . Besides, since  $|\mathcal{S}_{(v)}| \geq \kappa_{(v)}$ , frame  $v$  will be recovered: contradiction with the assumption.

In the second part of the proof, it is shown that if a frame  $v$  is recovered in the actual inter-frame decoding process, then there exists an iteration  $i^*$  in the two-phase algorithm, such that  $U_{(v)}$  is set to 1 for iterations  $i^*, i^* + 1, \dots$ . The proof outline is sketched here, and the details are omitted. A subgraph  $\mathcal{G}_{(v)}$  of the bipartite graph is defined recursively as follows: 1)  $\mathcal{G}_{(v)}$  contains variable node  $v$ , and 2) for each variable node  $v' \in \mathcal{G}_{(v)}$ ,  $\mathcal{G}_{(v)}$  contains the check node  $c$  and its variable node neighbors if subframe  $c$  is concatenated to frame  $v'$  in the inter-frame decoding process. Besides a direction is imposed on the edges in this subgraph, such that for each check node  $c$  and variable node  $v'$ , where  $v'$  and  $c$  are neighbors and  $\{v', c\} \subset \mathcal{G}_{(v)}$ : the edge is directed from  $c$  to  $v'$  if subframe  $c$  is concatenated to frame  $v'$  in the inter-frame decoding process, and from  $v'$  to  $c$  otherwise. Define  $L_{(v)}$  as the longest sequence of directed edges, from any variable node  $v' \in \mathcal{G}_{(v)}$  to  $v$ .

Consider the following set of messages in the decoding-on-graph process: for each check node  $c$  and variable node  $v'$ , where  $v'$  and  $c$  are neighbors and  $\{v', c\} \subset \mathcal{G}_{(v)}$ , the set includes the check-to-variable message from  $c$  to  $v'$  if the edge in  $\mathcal{G}_{(v)}$  is directed from  $c$  to  $v'$ , and the variable-to-check message from  $v'$  to  $c$  otherwise. It can be proven that these messages are all set to 1 after a number of iterations that is less than or equal to  $\frac{L_{(v)}+1}{2}$ . This can be proven using the *persisting 1-message*



property; however the proof is omitted for brevity.  $\square$

The described algorithm generalizes a two-phase message-passing algorithm that can be applied in erasure decoding of LDPC codes and of the codes defined in [85, 86]. The erasure-decoding case can be obtained by limiting the possible  $\kappa$ -values to 0 and 1, where the parameter  $\delta = \delta_{(1)}$  of  $(\delta_{(\omega)})$  represents the erasure rate while  $\mu$  is set to 0.

### 4.4.3 Asymptotic Performance Analysis

The third step of analyzing the performance of inter-frame coding is done here as follows: a compact<sup>2</sup> mathematical expression that describes the outcome of the decoding-on-graph process is derived in Theorem 1 of the subsection. The analysis performed to obtain Theorem 1 follows the analysis technique developed in [87]. Therefore, the basic features of this technique are described first, followed by the analysis of the inter-frame decoding case.

**Preliminaries:** The decoding-on-graph algorithm is analyzed when being applied on bipartite graphs with the following properties: 1)  $(N_F, K_S) \rightarrow (\infty, \infty)$ , and 2) the graphs are constructed randomly according to two probability distributions: the edge variable and check degree-distributions,  $(\lambda_{(i)})_{i \geq 1}$  and  $(\rho_{(i)})_{i \geq 1}$  respectively. The following terminology is adopted from [85], and is restated here for clarity: an edge in the bipartite graph drawn between variable node  $v$  and check node  $c$  has as variable degree the degree of  $v$ , and as check degree the degree of  $c$ . Then,  $\lambda_{(i)}(\rho_{(i)})$ ,  $i \geq 1$ , is the probability that an edge of the graph, picked randomly, has variable(check) degree  $i$ . For any two distributions of  $(\lambda_{(i)})_{i \geq 1}$  and  $(\rho_{(i)})_{i \geq 1}$ , random graphs with the respective edge degree-distributions can be constructed, and these two distributions characterize the constructed ensemble. The functions  $\lambda(x)$  and  $\rho(x)$  are defined as follows:  $\lambda(x) = \sum_i \lambda_{(i)} \cdot x^{i-1}$  and  $\rho(x) = \sum_i \rho_{(i)} \cdot x^{i-1}$ .

A basic result on the randomly constructed bipartite graph is stated in [87], and is restated in this paragraph. Consider a variable node  $v$ :  $\forall l > 0$ , define the neighbors of  $v$  within distance  $2 \cdot l$ , as the nodes that can be reached starting from  $v$  through a sequence of  $2 \cdot l$  edges or less. Consider the subgraph  $\mathcal{G}_{(v,l)}$  including  $v$  and all the neighbors of  $v$  within distance  $2 \cdot l$ : the probability that  $\mathcal{G}_{(v,l)}$  fails to be a tree goes to zero as  $N_F$  grows to infinity, for a fixed value of  $l$ . For simplicity of exposition, it is assumed here that  $\mathcal{G}_{(v,l)}$  is a tree with probability 1. This assumption, denoted here as *tree assumption*, does not impact the validity of the derived results but makes the following derivations clearer.

The decoding-on-graph process is applied on a randomly constructed bipartite graph. As in [87], it is viewed as a random discrete process and the evolution

---

<sup>2</sup>The term ‘‘compact’’, used here to describe a mathematical expression, means that the corresponding expression involves few well-defined terms.

of one of its parameters  $Q_{(i)}$  throughout the decoding process is analyzed, where  $Q_{(i)}$  is defined as the probability *that a randomly picked check-to-variable message at iteration  $i$  is 1*. Besides its significance, it can be seen that  $\{Q_{(i)}\}$  determines the sequence  $\{P_{(i)}\}$  defined as such:  $P_{(i)}$  is the probability *that a randomly picked variable-to-check message at iteration  $i$  is 1*. Let  $f(\cdot)$  be the function defined over  $[0, 1]$  such that  $f(\cdot)$  maps  $Q_{(i)} = y$  to  $Q_{(i+1)} = f(y)$ . Define  $y^* \in [0, 1]$  as follows:

$$f(y) > y \quad \forall 0 \leq y < y^* \quad \text{and} \quad f(y^*) = y^*. \quad (4.1)$$

The function  $f(\cdot)$  is clearly increasing; therefore, the convergence value of the sequence  $\{Q_{(i)}\}$ ,  $i = 1 \cdots \infty$ , is  $Q_{(\infty)} = y^*$ . Besides, it is noteworthy that the function  $f(y) - y$  indicates the speed of convergence of  $\{Q_{(i)}\}$  to  $y^*$ . Therefore, the inequality (4.1) describes the outcome of the decoding-on-graph process. Next in this subsection, this inequality will be reformulated into another compact expression relating the degree-distributions and channel parameters. This reformulated expression, obtained in Theorem 1, will be used to construct optimal degree-distributions.

**Analysis:** Let  $g(x)$  be a function of  $0 \leq x \leq 1$ , defined as the probability *that a randomly-picked variable-to-check message is 0*, given the probability *that a randomly-picked incoming check-to-variable message is 0* is  $x$ , then  $f(1 - x) = \sum_i \rho_{(i)} \cdot (1 - g(x))^{i-1} = \rho(1 - g(x))$ . The latter equality use the *tree assumption* to deduce that the variable-to-check messages incoming to a check node of degree  $i$  are independent random variables. Therefore, the probability *that the outgoing check-to-variable message is 1* is a product of  $i - 1$  identical values, each equal to  $1 - g(x)$ . Define  $x^* = 1 - y^*$ , inequality (4.1) can be reformulated to:

$$\rho(1 - g(x)) > 1 - x \quad \forall x \in ]x^*, 1]. \quad (4.2)$$

For the erasure decoding case where  $\delta_{(\omega)} = 0$  for  $\omega > 1$ ,  $g(x) = \delta_{(1)} \cdot \sum_i \lambda_{(i)} \cdot x^{i-1} = \delta_{(1)} \cdot \lambda(x)$ , and therefore  $f(1 - x) = \rho(1 - \delta \cdot \lambda(x))$ . Similar to the aforementioned reasoning, evaluation of  $g(x)$  uses the tree assumption to assume the independence of the incoming check-to-variable messages. The inequality (4.2) is thus equivalent to:

$$\rho(1 - \delta \cdot \lambda(x)) > 1 - x \quad \text{for} \quad 0 \leq x^* < x \leq 1. \quad (4.3)$$

The contribution of this subsection is the derivation of an expression, analogous to that in (4.3), for the inter-frame decoding case. This is done as follows: inequality (4.1) can be reformulated as shown in the following theorem.

**Theorem 1.** (decoding-on-graph outcome characterization) *For a distribution  $(\delta_{(\omega)})$  described by  $(\delta, \mu)$ , and degree-distributions,  $(\lambda_{(i)})_{i \geq 1}$  and  $(\rho_{(i)})_{i \geq 1}$ :*

$$\rho(1 - \delta \cdot \lambda(z)) > \frac{1 - z}{1 - \mu} \quad , \quad \forall z \in ]z^*, 1].$$

where  $z^* = (1 - y^*) + \mu \cdot y^* > \mu$ .

*Proof.* Define  $g_{(d;\omega)}(x)$  as the probability that a randomly-picked variable-to-check message is 0 given: 1) the corresponding variable-node  $v$  has degree  $d$ , 2)  $\kappa_{(v)} = \omega$  and 3) the probability that a randomly-picked incoming check-to-variable message is 0 is  $x$ . Similarly, define  $g_{(d)}(x)$  as the probability that a randomly-picked variable-to-check message is 0 given: 1) the corresponding variable-node  $v$  has degree  $d$  and 2) the probability that a randomly-picked incoming check-to-variable message is 0 is  $x$ . The function  $g(x)$  can be expressed as:

$$g(x) = \sum_{d=1}^{\infty} \lambda_{(d)} \cdot g_{(d)}(x) \quad , \quad g_{(d)}(x) = \sum_{\omega} \delta_{(\omega)} \cdot g_{(d;\omega)}(x).$$

Then,

$$g_{(d;\omega)}(x) = \sum_{j=0}^{\min(\omega,d)-1} \binom{d-1}{j} \cdot (1-x)^j \cdot x^{d-1-j}.$$

The geometric progression property of  $\delta_{(\omega)}$  is now used to express  $g_{(d)}(x)$  as follows:

$$\begin{aligned} g_{(d)}(x) &= \sum_{\omega=0}^{\infty} \delta_{(\omega)} \cdot \left( \sum_{j=0}^{\min(\omega,d)-1} \binom{d-1}{j} \cdot (1-x)^j \cdot x^{d-1-j} \right) \\ &= \sum_{j=0}^{d-1} \left( \binom{d-1}{j} \cdot (1-x)^j \cdot x^{d-1-j} \cdot \sum_{\omega=j+1}^{\infty} \delta_{(\omega)} \right) \\ &= \sum_{j=0}^{d-1} \left( \binom{d-1}{j} \cdot (1-x)^j \cdot x^{d-1-j} \cdot \delta_{(1)} \cdot \mu^j \cdot \sum_{\omega=0}^{\infty} \mu^{\omega} \right) \\ &= \left( \delta_{(1)} \cdot \sum_{\omega=0}^{\infty} \mu^{\omega} \right) \cdot \sum_{j=0}^{d-1} \binom{d-1}{j} \cdot \mu^j \cdot (1-x)^j \cdot x^{d-1-j} \\ &= \frac{\delta_{(1)}}{1-\mu} \cdot (x + \mu \cdot (1-x))^{d-1}. \end{aligned}$$

We have  $\delta = \sum_{\omega=1}^{\infty} \delta_{(\omega)}$ , then  $\delta = \delta_{(1)} + \sum_{\omega=2}^{\infty} \delta_{(\omega)} = \delta_{(1)}(1 + \sum_{\omega=1}^{\infty} \mu^{\omega}) = \frac{\delta_{(1)}}{1-\mu}$ ,  
Therefore:

$$g(x) = \sum_{d=1}^{\infty} \lambda_{(d)} g_{(d)}(x) = \delta \cdot \lambda(x + \mu \cdot (1-x)). \quad (4.4)$$

Then, inequality (4.2) can be rewritten as:

$$\rho(1 - \delta \cdot \lambda(x + \mu \cdot (1-x))) > 1 - x, \quad x^* < x \leq 1.$$

Then, by applying the following change of variable  $z = x + \mu(1 - x)$ , the expression becomes, for  $z^* = (1 - y^*) + \mu \cdot y^* \geq \mu$ :

$$\rho(1 - \delta \cdot \lambda(z)) > \frac{1 - z}{1 - \mu}, \quad z \in ]z^*, 1].$$

□

The derived analytic expression is a generalization of that corresponding to erasure decoding: by setting  $\mu$  to 0, the derived inequality reduces to  $\rho(1 - \delta \cdot \lambda(z)) > 1 - z$ ,  $1 - y^* < z \leq 1$ . This result is not unexpected since the decoding-on-graph algorithm is itself a generalized form of a LDPC erasure-decoding message-passing algorithm.

Ideally, decoding is successful if  $y^* = 1$ , that is the sequence  $\{Q_{(i)}\}$  converges to 1. This means that the sequence  $\{P_{(i)}\}$ , converges to 1 as well. This can be checked from the following inequality:  $Q_{(i)} \leq P_{(i-1)}^{m-1}$ ,  $m$  here is the minimum check-node degree<sup>3</sup>. However, such ideal scenario is impossible under the geometric progression model of  $(\delta_{(\omega)})$ . To see this, assume  $Q_{(\infty)} = y^* = 1$  or equivalently  $x^* = 0$ ,  $\rho(1 - g(0))$  equals 1 by inequality (4.2). This in turn implies that  $g(0)$ , evaluated to  $\delta \cdot \lambda(\mu)$  from (4.4), equals 0 which is clearly contradictory. Therefore, the sequence  $\{Q_{(i)}\}$  does not converge to 1, regardless of the channel parameters  $(\delta, \mu)$  and the degree distributions. A related observation is that for any variable-node  $v$  of degree  $d$ , the probability that decoding does not recover the frame corresponding to  $v$  is lower bounded by the nonzero value of  $\sum_{\omega=d+1}^{\infty} \delta_{(\omega)}$ .

In light of this result, the optimality question raised in 4.4.1 is slightly modified as follows: For any  $\epsilon > 0$ , are there codes that can be defined over increasing values of  $N_F$  such that: as  $N_F \rightarrow \infty$ , 1)  $\frac{K_S}{N_F} \rightarrow \sum_{\omega \geq 1} (\omega \cdot \delta_{(\omega)})$ , and 2) the probability that a randomly-chosen frame is successfully-decoded in the inter-frame decoding process is greater than  $1 - \epsilon$ ?

#### 4.4.4 Optimal Degree Distribution Construction

To obtain inter-frame codes that are optimal in the sense defined in subsection 4.4.1, the following methodology is devised: a sequence of degree-distribution couples parameterized by  $J = 1 \cdots \infty$ ,  $((\lambda_{(i)})_{i \geq 1}, (\rho_{(i)})_{i \geq 1})^{(J)}$ , is constructed such that:

1.  $\frac{K_S}{N_F} \rightarrow \sum_{\omega \geq 1} (\omega \cdot \delta_{(\omega)}) = \frac{\delta}{1 - \mu}$ , as  $(J, N_F) \rightarrow \infty$ .
2.  $\exists 0 < Z_0 < 1$  and  $J^* \in \mathbb{Z}^+$ , such that:

$$\rho^{(J)}(1 - \delta \cdot \lambda^{(J)}(z)) > \frac{1 - z}{1 - \mu}, \quad \forall J > J^* \quad \text{and} \quad Z_0 < z \leq 1.$$

<sup>3</sup>The special case  $m = 1$  is not considered here merely for simplicity of exposition.

This inequality is that obtained in Theorem 1 of the previous subsection.

3. The minimum variable-node degree goes to  $\infty$  as  $J \rightarrow \infty$ .

It is shown here that if these properties are satisfied, then the constructed sequence of degree-distributions results in optimal inter-frame code design. First, property (1) clearly means that  $\frac{K_S}{N_F}$  converges to the optimal value of  $\sum_{\omega \geq 1} (\omega \cdot \delta_{(\omega)})$  as  $N_F \rightarrow \infty$ . Second, properties (2) and (3) imply that for any  $\epsilon > 0$ , there exists an integer  $J_{(\epsilon)} \in \mathbb{Z}^+$ , such that the probability *that a frame  $f$  is unrecovered in the decoding of a random code constructed according to  $((\lambda_{(i)})_{i \geq 1}, (\rho_{(i)})_{i \geq 1})^{(J)}$* , converges to a value less than  $\epsilon$  as  $N_F \rightarrow \infty$  if  $J > J_{(\epsilon)}$ . This latter result can be proved by noticing that property (2) implies that,  $\forall J > J^*$ , the value  $z^*$  defined in the previous subsection is less than  $Z_0$ , or equivalently that the sequence  $\{Q_{(i)}\}$  converges to a value  $Q_{(\infty)} = y^*$  that is greater than  $Y_0 = \frac{1-Z_0}{1-\mu}$ . Let  $\tau$  be the minimum variable-node degree, i.e.  $\lambda_{(i)} = 0$  for  $i < \tau$ . Consider the probability *that a randomly-picked variable-to-check message is 0* given 1) the corresponding variable-node  $v$  has degree  $\tau$  and 2) the probability *that a randomly-picked check-to-variable message incoming to  $v$  is 0* is  $1 - y^*$ : it is denoted by  $g_{(\tau)}(1 - y^*)$  in the proof of Theorem 1 and equals  $\delta \cdot ((1 - y^*) + \mu \cdot y^*)^{\tau-1} = \delta \cdot ((1 - y^* \cdot (1 - \mu))^{\tau-1} < \delta \cdot ((1 - Y_0 \cdot (1 - \mu))^{\tau-1}$ . Property (3) means that  $\tau$  goes to  $\infty$  as  $J \rightarrow \infty$ , and therefore  $g_{(\tau)}(1 - y^*) < \delta \cdot ((1 - Y_0 \cdot (1 - \mu))^{\tau-1}$  approaches 0 as  $J$  goes to  $\infty$ . Furthermore, it can be checked that the probability *that a frame corresponding to a randomly-picked variable node  $v$  is unrecovered*, given the probability *that a randomly-picked check-to-variable message is 0* is  $x$ , is less than  $g_{(\tau)}(x)$ . Therefore, as the probability *that a randomly-picked check-to-variable message is 1* converges to  $y^* > Y_0$ , the probability *that a frame corresponding to a randomly-picked variable node is unrecovered* converges to 0 as  $(J, N_F) \rightarrow \infty$ .

A sequence of degree-distributions is constructed next. Then, it is proved that this sequence satisfies properties (1)-(3), and thus results in optimal code design. Some concluding remarks are finally made.

**Degree-distributions:** The distributions constructed here are a generalization of the optimal distributions of the erasure codes in [85]. The erasure code distributions are described briefly next. The  $\lambda$ -distribution is:

$$\lambda_{(i)} = \frac{1}{H \cdot (i - 1)}, \quad i = 2, \dots, d + 1.$$

for a chosen integer  $d$  and  $H = \sum_{i=1}^d \frac{1}{i}$ . The average variable-node degree is:

$$a_v = \left( \sum_i \frac{\lambda_{(i)}}{i} \right)^{-1} = \left( \sum_{i=2}^{d+1} \frac{1}{H \cdot i \cdot (i - 1)} \right)^{-1} = H \cdot (1 + 1/d).$$

The function  $\lambda(x) \sim -\ln(1-x)/H$  (but strictly less).

The  $\rho$ -distribution is described as follows:

$$\rho_{(i)} = \frac{e^{-\alpha} \cdot \alpha^{i-1}}{(i-1)!} \quad i = 1, \dots, \infty.$$

Thus,  $\rho(x) = e^{\alpha \cdot (x-1)}$ . The average check-node degree is:

$$a_c = \left( \sum_i \frac{\rho_{(i)}}{i} \right)^{-1} = \alpha / (1 - e^{-\alpha}).$$

Then, we have for erasure codes:

$$\begin{aligned} \rho(1 - \delta \cdot \lambda(x)) &= e^{-\alpha \cdot \delta \cdot \lambda(x)} \\ &> e^{\frac{\alpha \cdot \delta}{H} \cdot \ln(1-x)} = (1-x)^{\frac{\alpha \cdot \delta}{H}} = (1-x)^{\frac{a_c}{a_v} \cdot (1+1/d) \cdot (1-e^{-\alpha}) \cdot \delta} \\ &\geq (1-x), \quad \forall 0 \leq x \leq 1. \end{aligned}$$

The last inequality is valid when  $\frac{a_v}{a_c} \geq \delta \cdot (1 + 1/d)$ .

*Optimal IIF Degree Distributions:* The proposed inter-frame code distributions are constructed next. The  $\lambda$ -distribution is parameterized by two integers: the parameter  $J$  of the constructed distribution couple  $((\lambda_{(i)})_{i \geq 1}, (\rho_{(i)})_{i \geq 1})^{(J)}$  and another integer  $d$ . It is assumed that as  $J$  goes to  $\infty$  so does  $d$ , however, no specific relation involving both of them is imposed. In the rest of the section and for sake of notation simplicity, the super-index ( $J$ ) will be omitted from the  $\lambda$  and  $\rho$  terms. The constructed  $\lambda$ -distribution can be described as: ( $H = \sum_{i=1}^d i^{-1}$ )

$$\lambda_{(J \cdot (i-1) + 1)} = \frac{1}{H \cdot (i-1)}, \quad i = 2, \dots, d+1.$$

and  $\lambda_{(k)} = 0$ , otherwise.

The construction of the  $\rho$ -distribution involves two different distributions. Define the distribution  $(\beta_{(\alpha; i)})$ , where:

$$\beta_{(\alpha; i)} = \frac{e^{-\alpha} \cdot \alpha^{i-1}}{(i-1)!}.$$

This distribution is similar to the  $\rho$ -distribution in erasure codes. Define another distribution  $(\Omega_{(i)})$ , parameterized by the integer  $d_c$ , as such:

$$\Omega_{(i)} = \frac{1}{i \cdot H_c}, \quad i = 1, \dots, d_c.$$

where  $H_c = \sum_{i=1}^{d_c} \frac{1}{i}$ . The parameter  $d_c$  is chosen such that:

$$\sum_{i=1}^{d_c} \frac{1}{i} \leq J \cdot (1 - \mu) < \sum_{i=1}^{d_c+1} \frac{1}{i}.$$

From the definition of  $d_c$ , it can be deduced that  $\frac{J}{H_c} \geq \frac{1}{1-\mu}$  and that  $|\frac{J}{H_c} - \frac{1}{1-\mu}|$  goes to 0 as  $J \rightarrow \infty$ . The  $\rho$ -distribution is formed as follows:

$$\rho(i) = \sum_{j=1}^{d_c} \Omega_{(j)} \cdot \beta_{(j \cdot H/\delta; i)}.$$

**Proof of Optimality:** Property (3) is satisfied by the constructed degree-distributions, as the minimum variable-node degree, obtained from a distribution  $(\lambda_{(i)})^{(J)}$ , is  $J+1$  which clearly goes to  $\infty$  as  $J \rightarrow \infty$ .

Property (1) is shown to be satisfied as stated in Lemma 1.

**Lemma 1.**  $\frac{K_S}{N_F} \rightarrow \frac{\delta}{1-\mu}$  as  $(J, d) \rightarrow (\infty, \infty)$ .

*Proof.* Each of the expressions  $K_S \cdot a_c$  and  $N_F \cdot a_v$  represents the number of edges in the bi-partite graph, therefore  $K_S \cdot a_c = N_F \cdot a_v$  which implies that  $\frac{K_S}{N_F} = \frac{a_v}{a_c}$ .

The average variable-node degree is:

$$\begin{aligned} a_v &= \left( \sum_j \frac{\lambda_{(j)}}{j} \right)^{-1} \\ &= \left( \sum_{i=2}^{d+1} \frac{1}{H \cdot (i-1) \cdot (J \cdot (i-1) + 1)} \right)^{-1} \\ &= J \cdot H \left( \sum_{i=1}^d \frac{1}{i \cdot (i + 1/J)} \right)^{-1}. \end{aligned}$$

The average check-node degree is:

$$\begin{aligned} a_c &= \left( \sum_{i=1}^{\infty} \frac{\rho(i)}{i} \right)^{-1} = \left( \sum_{i=1}^{\infty} \sum_{j=1}^{d_c} \frac{\Omega_{(j)} \beta_{(j \cdot H/\delta; i)}}{i} \right)^{-1} \\ &= \left( \sum_{j=1}^{d_c} \Omega_{(j)} \cdot \sum_{i=1}^{\infty} \frac{\beta_{(j \cdot H/\delta; i)}}{i} \right)^{-1} \\ &= H_c \cdot \left( \sum_{j=1}^{d_c} \frac{1}{j} \cdot \frac{\delta}{jH} \cdot (1 - e^{-j \cdot H/\delta}) \right)^{-1} \\ &= \frac{H_c \cdot H}{\delta} \cdot \left( \sum_{j=1}^{d_c} \frac{1 - e^{-j \cdot H/\delta}}{j^2} \right)^{-1}. \end{aligned}$$

The rate  $\frac{K_S}{N_F} = \frac{a_v}{a_c}$  is then equal to:

$$\frac{a_v}{a_c} = \frac{J \cdot \delta}{H_c} \cdot \left( \sum_{i=1}^d \frac{1}{i \cdot (i + 1/J)} \right)^{-1} \left( \sum_{j=1}^{d_c} \frac{1 - e^{-j \cdot H/\delta}}{j^2} \right).$$

As  $(J, d) \rightarrow \infty$ ,  $\frac{J}{H_c} \rightarrow \frac{1}{1-\mu}$  by the definition of  $d_c$ . In addition, as  $d \rightarrow \infty$ ,  $H = \sum_{i=1}^d \frac{1}{i} \rightarrow \infty$ . Therefore, both  $\left( \sum_{i=1}^d \frac{1}{i \cdot (i+1/J)} \right)$  and  $\left( \sum_{j=1}^{d_c} \frac{1 - e^{-j \cdot H/\delta}}{j^2} \right)$  converge to the same finite value  $\left( \sum_{i=1}^{\infty} \frac{1}{i^2} \right)$ . Overall,  $\frac{a_v}{a_c} \rightarrow \frac{\delta}{1-\mu}$ .  $\square$

The proof that property (2) is satisfied is more complicated; yet, it motivates the specific construction of distributions  $(\lambda_{(i)})$  and  $(\rho_{(i)})$ . Prior to explaining this, the expressions of the  $\lambda(x)$  and  $\rho(x)$  functions are obtained as follows:

$$\begin{aligned} \lambda(x) &= \sum_{i=2}^{d+1} \frac{x^{J \cdot (i-1)}}{H \cdot (i-1)} \\ &= \frac{1}{H} \cdot \sum_{i=1}^d \frac{(x^J)^i}{i} \sim (<) - \frac{1}{H} \cdot \ln(1 - x^J). \end{aligned}$$

where the finite sum  $\sum_{i=1}^d \frac{(x^J)^i}{i}$  is approximated to the following power series  $\sum_{i=1}^{\infty} \frac{(x^J)^i}{i} = -\ln(1 - x^J)$ .

$$\begin{aligned} \rho(x) &= \sum_{i=1}^{\infty} \rho_{(i)} \cdot x^{i-1} \\ &= \sum_{i=1}^{\infty} x^{i-1} \cdot \left( \sum_{j=1}^{d_c} \Omega_{(j)} \cdot \beta_{(j \cdot H/\delta; i)} \right) \\ &= \sum_{j=1}^{d_c} \Omega_{(j)} \cdot \left( \sum_{i=1}^{\infty} \beta_{(j \cdot H/\delta; i)} \cdot x^{i-1} \right) \\ &= \sum_{j=1}^{d_c} \frac{1}{j \cdot H_c} e^{\frac{j \cdot H}{\delta} \cdot (x-1)}. \end{aligned}$$

The construction of the degree-distributions,  $(\lambda_{(i)})$  and  $(\rho_{(i)})$ , can now be motivated



by observing the following:

$$\begin{aligned}
\rho(1 - \delta \cdot \lambda(x)) &> \rho\left(1 + \frac{\delta}{H} \cdot \ln(1 - x^J)\right) \\
&= \sum_{j=1}^{d_c} \frac{1}{j \cdot H_c} e^{j \cdot \ln(1 - x^J)} \\
&= \frac{1}{H_c} \cdot \sum_{j=1}^{d_c} \frac{(1 - x^J)^j}{j} \\
&\sim -\frac{\ln(x^J)}{H_c} = -\frac{J}{H_c} \cdot \ln(x) \\
&\geq \frac{J}{H_c} \cdot (1 - x) \\
&> \frac{1 - x}{1 - \mu} \quad \forall 0 \leq x < 1.
\end{aligned}$$

This result, based on the approximation  $\sum_{j=1}^{d_c} \frac{(1-x^J)^j}{j} \sim -\ln(x^J)$ , is clearly not accurate. For example, it can be noticed that while  $\rho(1 - \delta \cdot \lambda(x)) \leq 1 \forall x \in [0, 1]$ ,  $-\ln(x) \rightarrow \infty$  as  $x \rightarrow 0$ . However, such result hints that the relation  $\rho^{(J)}(1 - \delta \cdot \lambda^{(J)}(z)) > \frac{1-z}{1-\mu}$  is valid for a sufficiently wide range of  $[0, 1]$  and for  $J$  large enough, and hence that property (2) is satisfied. This is rigorously proven next, using the following sequence of lemmas.

**Lemma 2.** *If  $(1 - x^J)^{d_c} < (1 - x) \forall 0 < X_l < x < 1$ , then  $\frac{1}{H_c} \cdot \sum_{i=1}^{d_c} \frac{(1-x^J)^i}{i} > \frac{1-x}{1-\mu} \forall X_l < x < 1$ .*

**Lemma 3.**  *$\forall \eta > 0, \exists J_0$  such that  $e^{1-\mu-\eta} < d_c^{\frac{1}{J}} < e^{1-\mu+\eta}, \forall J > J_0$ .*

**Lemma 4.** *For each value  $J \in \mathbb{Z}^+$ , there exists a single value  $X_c \in [0, 1]$  such that  $(1 - x^J)^{d_c} > 1 - x$  if  $x < X_c$ , and  $(1 - x^J)^{d_c} < 1 - x$  otherwise. In addition,  $X_c$  converges to  $e^{-(1-\mu)}$  as  $J \rightarrow \infty$ .*

*Proof.* (Lemma 2)

Define  $T(x) = \frac{1}{H_c} \cdot \sum_{i=1}^{d_c} \frac{(1-x^J)^i}{i}$ , and  $S(x) = \frac{1-x}{1-\mu}$ .

$$\begin{aligned} -\frac{\partial S}{\partial x} &= \frac{1}{1-\mu} \\ -\frac{\partial T}{\partial x} &= \frac{J}{H_c} \cdot x^{J-1} \sum_{i=1}^{d_c} (1-x^J)^{i-1} \\ &= \frac{J}{H_c} \cdot x^{J-1} \cdot \sum_{i=0}^{d_c-1} (1-x^J)^i \\ &= \frac{J}{H_c} \cdot x^{J-1} \cdot \frac{1-(1-x^J)^{d_c}}{1-(1-x^J)} = \frac{J}{H_c} \cdot \frac{1-(1-x^J)^{d_c}}{x}. \end{aligned}$$

If  $(1-x^J)^{d_c} < (1-x)$ , then  $\frac{1-(1-x^J)^{d_c}}{x} > 1$ , and since by definition  $\frac{J}{H_c} \geq \frac{1}{1-\mu}$ , then  $-\frac{\partial T}{\partial x} > -\frac{\partial S}{\partial x}$ . Since 1)  $-\frac{\partial T}{\partial x} > -\frac{\partial S}{\partial x} \forall X_l < x \leq 1$  and 2)  $T(1) = S(1) = 0$ , then  $T(x) > S(x) \forall X_l < x < 1$ .  $\square$

*Proof. (Lemma 3)*

Define  $\theta = \frac{\eta}{2 \cdot (1-\mu) - \eta}$ ,  $\epsilon = \frac{\eta}{2 \cdot (1-\mu)}$ , and  $\nu = \frac{\eta}{2 - \frac{\eta}{1-\mu}}$ . Each of  $\theta$ ,  $\epsilon$ , and  $\nu$  goes to 0 as  $\eta \rightarrow 0$ .

$H_c = \sum_{j=1}^{d_c} \frac{1}{j}$ , so  $\frac{H_c}{\ln(d_c)} \rightarrow 1$  as  $d_c \rightarrow \infty$  or, equivalently,  $J \rightarrow \infty$ . Therefore,  $\exists J_1$  such that  $\forall J > J_1$ :

$$\begin{aligned} \ln(d_c) &< H_c < (1+\theta) \cdot \ln(d_c) \\ &\iff d_c < e^{H_c} < d_c^{(1+\theta)} \\ &\iff e^{H_c \cdot (1-\epsilon)} = e^{\frac{H_c}{1+\theta}} < d_c < e^{H_c} \\ &\iff e^{\frac{H_c}{J} \cdot (1-\epsilon)} < d_c^{\frac{1}{J}} < e^{\frac{H_c}{J}}. \end{aligned}$$

By definition of  $d_c$ ,  $\frac{J}{H_c} \rightarrow \frac{1}{1-\mu}$  and  $\frac{J}{H_c} \geq \frac{1}{1-\mu}$ . Therefore,  $\exists J_2$  such that  $\forall J > J_2$ :

$$(1-\mu) - \nu < \frac{H_c}{J} \leq (1-\mu).$$

Then,  $\forall J > J_0 = \max(J_1, J_2)$ :

$$\begin{aligned} e^{((1-\mu)-\nu) \cdot (1-\epsilon)} &< d_c^{\frac{1}{J}} < e^{(1-\mu)} \\ \implies e^{(1-\mu)-\nu(1-\epsilon)-\epsilon(1-\mu)} &< d_c^{\frac{1}{J}} < e^{(1-\mu)}. \end{aligned}$$

where  $\nu \cdot (1-\epsilon) = \epsilon \cdot (1-\mu) = \eta/2$ , therefore:

$$e^{1-\mu-\eta} < d_c^{\frac{1}{J}} < e^{1-\mu+\eta} \quad \forall J > J_0.$$

$\square$

*Proof.* (Lemma 4)

Define  $P(x) = (1 - x^J)^{d_c}$  and  $Q(x) = 1 - x$ .

$$\frac{\partial P}{\partial x} = -d_c \cdot J \cdot x^{J-1} \cdot (1 - x^J)^{d_c-1}.$$

$$\begin{aligned} \frac{\partial^2 P}{\partial x^2} &= -d_c \cdot J \cdot (J-1) \cdot x^{J-2} \cdot (1 - x^J)^{d_c-1} \\ &\quad + d_c \cdot (d_c - 1) \cdot J^2 \cdot x^{2(J-1)} \cdot (1 - x^J)^{d_c-2} \\ &= -d_c \cdot J \cdot x^{J-2} \cdot (1 - x^J)^{d_c-2} \left( (J-1) \cdot (1 - x^J) - (d_c - 1) \cdot J \cdot x^J \right) \\ &= -d_c \cdot J \cdot x^{J-2} \cdot (1 - x^J)^{d_c-2} \left( (J-1) + (1 - d_c J) \cdot x^J \right). \end{aligned}$$

Some properties of  $P(x)$  and  $Q(x)$  are: 1)  $\frac{\partial^2 P}{\partial x^2}$  changes its sign once in the range  $[0, 1]$ , 2)  $P(0) = Q(0) = 1$  and  $P(1) = Q(1) = 0$ , and 3)  $P'(0) = P'(1) = 0 > Q'(0) = Q'(1) = -1$ . It can be deduced from these properties that there exists one value  $X_c$  such that  $P(x) > Q(x)$  if  $x < X_c$  and  $P(x) \leq Q(x)$  otherwise.

The convergence value of  $X_c$  as  $J \rightarrow \infty$ , is now considered. An arbitrarily small value of  $\eta$  is chosen, and  $J_0$  is defined according to Lemma 3.

$$\begin{aligned} 1 < -\frac{\ln(P(x))}{d_c \cdot x^J} &= -d_c \cdot \frac{\ln(1 - x^J)}{d_c x^J} \\ &= 1 + \frac{x^J}{2} + \frac{x^{2J}}{3} + \dots \\ &< 1 + x^J + x^{2J} + \dots = \frac{1}{1 - x^J}. \end{aligned}$$

Consider the value  $x_1 = e^{-(1-\mu)-2\eta}$ . Since  $\frac{1}{1-x^J} \rightarrow 1$  as  $J \rightarrow \infty$ , then  $\forall \chi > 0, \exists J_3$  such that:  $d_c \cdot x^J < -\ln(P(x)) < (1 + \chi) \cdot d_c \cdot x^J \forall J > J_3$ . Fix the value of  $\chi$ .

$$\begin{aligned} -\ln(P(x_1)) &< (1 + \chi) \cdot d_c \cdot x_1^J = (1 + \chi) \cdot (d_c^{\frac{1}{J}} \cdot x_1)^J \\ &< (1 + \chi) \cdot (e^{1-\mu+\eta} \cdot e^{-(1-\mu)-2\eta})^J \\ &= (1 + \chi)e^{-\eta \cdot J} \quad \forall J > \max(J_0, J_3). \end{aligned}$$

Since  $(1 + \chi)e^{-\eta \cdot J} \rightarrow 0$  as  $J \rightarrow \infty$ ,  $\exists J_4$  such that  $(1 + \chi)e^{-\eta \cdot J} < -\ln(Q(x_1)) \forall J > J_4$ . Therefore:

$$-\ln(P(x_1)) < -\ln(Q(x_1)) \quad \forall J > \max(J_0, J_3, J_4).$$

Now, consider the value  $x_2 = e^{-(1-\mu)+2\eta}$ .

$$\begin{aligned} -\ln(P(x_2)) &> (d_c^{\frac{1}{J}} \cdot x_2)^J \\ &> (e^{(1-\mu)-\eta} \cdot e^{-(1-\mu)+2\eta})^J \\ &= e^{\eta \cdot J} \quad \forall J > J_0. \end{aligned}$$

Since  $e^{\eta J} \rightarrow \infty$  as  $J \rightarrow \infty$ ,  $\exists J_5$  such that  $e^{\eta J} > -\ln(Q(x_2)) \forall J > J_5$ . Therefore:

$$-\ln(P(x_2)) > e^{\eta J} > -\ln(Q(x_2)) \quad \forall J > \max(J_0, J_5).$$

Overall, define  $J_6 = \max(J_0, J_3, J_4, J_5)$ :

$$P(x_1) > Q(x_1) \quad \text{and} \quad P(x_2) < Q(x_2) \quad \forall J > J_6.$$

Therefore:

$$e^{-(1-\mu)-2\eta} = x_1 < X_c < x_2 = e^{-(1-\mu)+2\eta} \quad \forall J > J_6.$$

Since  $\eta$  can be chosen arbitrarily small, then  $X_c$  converges to  $e^{-(1-\mu)} > \mu$  as  $J \rightarrow \infty$ .  $\square$

From Lemma 4, it can be concluded that for an arbitrarily small  $\gamma$ :  $\exists J^*$  such that:

$$X_c < e^{-(1-\mu)+\gamma} \quad \forall J > J^*.$$

Therefore:

$$(1 - x^J)^{d_c} < 1 - x \quad \forall x > e^{-(1-\mu)+\gamma} \quad \text{and} \quad J > J^*.$$

This means, by Lemma 2, that:

$$\rho(1 - \delta \cdot \lambda(x)) = \frac{1}{H_c} \cdot \sum_{i=1}^{d_c} \frac{(1 - x^J)^i}{i} > \frac{1 - x}{1 - \mu} \quad \forall x > e^{-(1-\mu)+\gamma}, \quad J > J^*.$$

Property (2) is thus satisfied.

A stronger result can be derived on the function  $\rho(1 - \delta \cdot \lambda(x)) = T(x)$ . The result is briefly described next. It can be observed from the proof of Lemma 4 that when  $x < e^{-(1-\mu)}$ ,  $-\ln P(x) \rightarrow 0$  or equivalently  $P(x) \rightarrow 1$  as  $J \rightarrow \infty$ . On the other hand, when  $x > e^{-(1-\mu)}$ ,  $-\ln P(x) \rightarrow \infty$  or equivalently  $P(x) \rightarrow 0$  as  $J \rightarrow \infty$ . As seen in the proof of Lemma 2:

$$-\frac{H_c}{J} \cdot \frac{\partial T}{\partial x} = \frac{1 - (1 - x^J)^{d_c}}{x} = \frac{1 - P(x)}{x}.$$

As  $J \rightarrow \infty$ ,  $\frac{1-P(x)}{x} \rightarrow \frac{1}{x} = \frac{\partial \ln(x)}{\partial x}$  if  $x > e^{-(1-\mu)}$ , and  $\frac{1-P(x)}{x} \rightarrow 0$  otherwise. Besides, as  $J \rightarrow \infty$ ,  $\frac{J}{H_c} \rightarrow \frac{1}{1-\mu}$ . It can be thus concluded that the function  $\rho(1 - \delta \cdot \lambda(x)) = T(x)$  converges to the piecewise function  $T_{(\infty)}(x)$ , as  $J \rightarrow \infty$ , where:

$$T_{(\infty)}(x) = \begin{cases} 1, & x < e^{-(1-\mu)} \\ -\frac{\ln(x)}{1-\mu}, & x \geq e^{-(1-\mu)} \end{cases}$$

**Remarks:** Overall, it is proved in the section that optimal codes exist, for a channel-characterizing distribution  $(\delta_{(\omega)})$  described by a  $(\delta, \mu)$  pair.

Two related questions can then be raised. These questions form topics of further research which are beyond the scope of the dissertation. The first question is whether inter-frame coding is optimal under any arbitrary channel-characterizing distribution  $(\delta_{(\omega)})$ , that is when  $(\delta_{(\omega)})$  does not form a geometric progression.

The second question is motivated, in broadcast communication, by the fact that the different channels corresponding to different receivers may be described by different  $(\delta, \mu)$  pairs. Consequently, it can be formulated as the problem of finding, for any two channels described by  $(\delta^{(1)}, \mu^{(1)})$  and  $(\delta^{(2)}, \mu^{(2)})$  respectively, the inter-frame code with the minimum  $\frac{K_S}{N_F}$  that can be deployed for successful inter-frame decoding under both channels, as  $N_F \rightarrow \infty$ . A lower bound on the achievable  $\frac{K_S}{N_F}$  is  $\max(\frac{\delta^{(1)}}{1-\mu^{(1)}}, \frac{\delta^{(2)}}{1-\mu^{(2)}})$ . The optimal degree distributions, constructed in this section, have  $\delta$  and  $\mu$  as parameters in their definition, and therefore may not be appropriate for the current case of multiple distributions  $(\delta_{(\omega)})$ . In this regard, the following simple result can be stated: if  $\delta^{(1)} \geq \delta^{(2)}$  and  $\mu^{(1)} > \mu^{(2)}$ , then the optimal degree-distribution designed for  $(\delta^{(1)}, \mu^{(1)})$  resulting in  $\frac{K_S}{N_F} = \frac{\delta^{(1)}}{1-\mu^{(1)}}$ , results also in successful inter-frame decoding for  $(\delta^{(2)}, \mu^{(2)})$ . This result can be proved by observing that if  $\delta^{(1)} \geq \delta^{(2)}$  and  $\mu^{(1)} \geq \mu^{(2)}$ , then:

$$\begin{aligned} \rho(1 - \delta^{(1)} \cdot \lambda(z)) &> \frac{1 - z}{1 - \mu^{(1)}} \\ \Rightarrow \rho(1 - \delta^{(2)} \cdot \lambda(z)) &> \frac{1 - z}{1 - \mu^{(2)}}. \end{aligned}$$

The details of the proof are omitted for brevity. The case where  $\delta^{(1)} - \delta^{(2)}$  and  $\mu^{(1)} - \mu^{(2)}$  have different signs, is more challenging and requires further investigation. This problem is denoted here as the *problem of designing universal inter-frame codes*.

## 4.5 Asymptotic Coding Performance

In this section, the performance of inter-frame coding is compared to that of two conventional solutions: IR-HARQ and the state-of-the-art two-stage scheme. This comparison is done by computing the enhancement ratio, defined as the ratio of the effective frame-length obtained in the conventional solution to that obtained in inter-frame coding. The key result of the previous section is that the optimal effective frame-length, obtained in inter-frame coding for a channel-characterizing distribution  $(\delta_{(\omega)})$ , is  $N + \frac{\delta}{1-\mu} \cdot \Delta = N(1 + \frac{\delta}{1-\mu} \cdot \frac{\Delta}{N})$ .

It is assumed throughout this section that all the receivers, in the broadcast communication, have the same channel-characterizing distribution  $(\delta_{(\omega)})$ . This *single-distribution* assumption is motivated by two considerations. First, the comparison setup has to be kept within the range of the results obtained in the previous sections. In this regard, the *single-distribution* assumption rules out the scenarios for which the optimal value of  $\frac{K_S}{N_F}$  is not derived in the chapter and, consequently, for which an accurate estimation of the effective frame-lengths cannot be made. One such scenario is the following: the channel-characterizing distributions corresponding to different receivers are described by different  $(\delta, \mu)$  couples which, however, result in close or identical mean  $\kappa$ -value, i.e.  $(\frac{\delta}{1-\mu})$ . Second, the *single-distribution* assumption can be viewed as a simplification of the *worst-distribution* assumption explained as follows: in broadcast communication where receivers have different channel-characterizing distributions, the effective frame-length should be large enough to account for the distribution with the highest value of  $\sum_{\omega} (\omega \cdot \delta_{(\omega)}) = \frac{\delta}{1-\mu}$ . The *worst-distribution* assumption is that this distribution has the highest value of each of  $\delta$  and  $\mu$ , as well. This means that the design of the inter-frame code, and therefore the value of  $\frac{K_S}{N_F}$ , is determined solely by this distribution, as seen in Section 4.4. It is, therefore, the distribution considered in evaluating the resulting effective frame-length obtained from inter-frame coding.

An underlying assumption made in this section is that the distribution  $(\delta_{(\omega)})$  does not change with the different schemes (IR-HARQ, two-stage, or inter-frame coding) used. The possibility of  $(\delta_{(\omega)})$  varying with schemes, due to the resulting change in transmission scheduling for example, is not considered here. Therefore, the reported results can be fully attributed to the peculiar coding features of the compared schemes.

### 4.5.1 IR-HARQ

Many variations of the IR-HARQ scheme exist. One simple IR-HARQ scheme is assumed here and the resulting effective frame-length is obtained. The scheme can be described as follows: for each frame  $f$ , the rate- $R_H$   $N$ -bit portion of the frame is initially transmitted. A retransmission is initiated as long as: 1) feedback from at

least one receiver reports a decoding failure and 2) the number of retransmissions corresponding to  $f$  are less than some upper bound  $n^*$ . The  $i$ th retransmission,  $1 \leq i \leq n^*$ , consists of increment  $\Delta_{(f,i)}$ . For this subsection, the following terminology is defined:  $n_{(f)}$  is the the number of transmitted increments for frame  $f$ ,  $\mathcal{R}$  is the set of receivers, and  $\kappa_{(f;i)}$ ,  $i = 1, \dots, |\mathcal{R}|$ , is the  $\kappa$ -value of frame  $f$  for the  $i$ th receiver. It can be deduced from the description of the IR-scheme that  $n_{(f)}$  is equal to  $\min(\max_{1 \leq i \leq |\mathcal{R}|} \kappa_{(f;i)}, n^*)$ , and that the frame-error-rate (FER) for each receiver is equal to  $\sum_{\omega > n^*} \delta_{(\omega)} = \delta \cdot \mu^{n^*}$ . Therefore, for a target FER of  $10^{-a}$ ,  $n^*$  is set to  $\lceil \frac{-a - \log_{10}(\delta)}{\log_{10}(\mu)} \rceil$ .

As the number of receivers,  $|\mathcal{R}|$  goes to infinity, the value of  $\max_{1 \leq i \leq |\mathcal{R}|} \kappa_{(f;i)}$  goes to  $\infty$  and, therefore,  $n_{(f)} \rightarrow n^*$ . This means that the feedback from the receivers has no impact on the number of retransmissions when  $|\mathcal{R}| \rightarrow \infty$ . Therefore, the IR-HARQ becomes equivalent to a stand-alone forward-error-correction scheme with no feedback, and with the frame-length set to  $N \cdot (1 + n^* \cdot \frac{\Delta}{N})$ . The resulting enhancement ratio is:

$$\frac{1 + n^* \cdot \frac{\Delta}{N}}{1 + \frac{\delta}{1-\mu} \cdot \frac{\Delta}{N}}$$

The enhancement ratio is plotted versus  $\mu$  in Fig. 4.7 for  $\frac{\Delta}{N} = 0.1$ ,  $\delta \in \{0.3, 0.5, 0.7\}$ , and FER values in  $\{10^{-1}, 10^{-2}, 10^{-3}\}$ . Two observations can be made accordingly. First, the sensitivity of the enhancement ratio to the variations of  $\delta$  is significantly less than its sensitivity to variations in FER and  $\mu$ . Second, the enhancement ratio values are considerably high. For example, even for a relatively low target FER of  $10^{-2}$ , the enhancement ratio exceeds 2 for  $\mu \sim 0.75$ , and reaches 3 or above as  $\mu \rightarrow 0.9$ .

The enhancement ratios plotted in Fig.4.7 assumes an infinitely large number of receivers. Next the case of finite values of  $|\mathcal{R}|$  is considered. For  $1 \leq n < n^*$ , the probability that  $n_{(f)} = n$  is equal to  $(1 - \mu^n \cdot \delta)^{|\mathcal{R}|} - (1 - \mu^{n-1} \cdot \delta)^{|\mathcal{R}|}$ . Therefore, the expected value of  $n_{(f)}$ , for some value of  $|\mathcal{R}|$  is:

$$E(n_{(f)}) = n^* - \sum_{i=0}^{n^*-1} (1 - \mu^i \cdot \delta)^{|\mathcal{R}|} = \sum_{i=0}^{n^*-1} (1 - (1 - \mu^i \cdot \delta)^{|\mathcal{R}|}).$$

It can be easily checked that  $E(n_{(f)})$  converges to a finite value as the target FER  $10^{-a}$  converges to 0, or equivalently as  $a \rightarrow \infty$ , by noting that: 1)  $E(n_{(f)})$  increases as  $a \rightarrow \infty$  since  $n^*$  increases with  $a$ , and 2)  $E(n_{(f)})$  is upper bounded since  $n_{(f)} \leq \sum_{i=1}^{|\mathcal{R}|} \kappa_{(f;i)}$  and, therefore,  $E(n_{(f)}) \leq |\mathcal{R}| \cdot \frac{\delta}{1-\mu}$ . This convergence can be qualitatively justified by the fact that if  $|\mathcal{R}| \ll 10^a$ , then the probability that  $\max_{1 \leq i \leq |\mathcal{R}|} \kappa_{(f;i)} \geq n^*$  is very small, therefore, so is the probability that  $n_{(f)}$  varies with the increase in  $n^*$ .

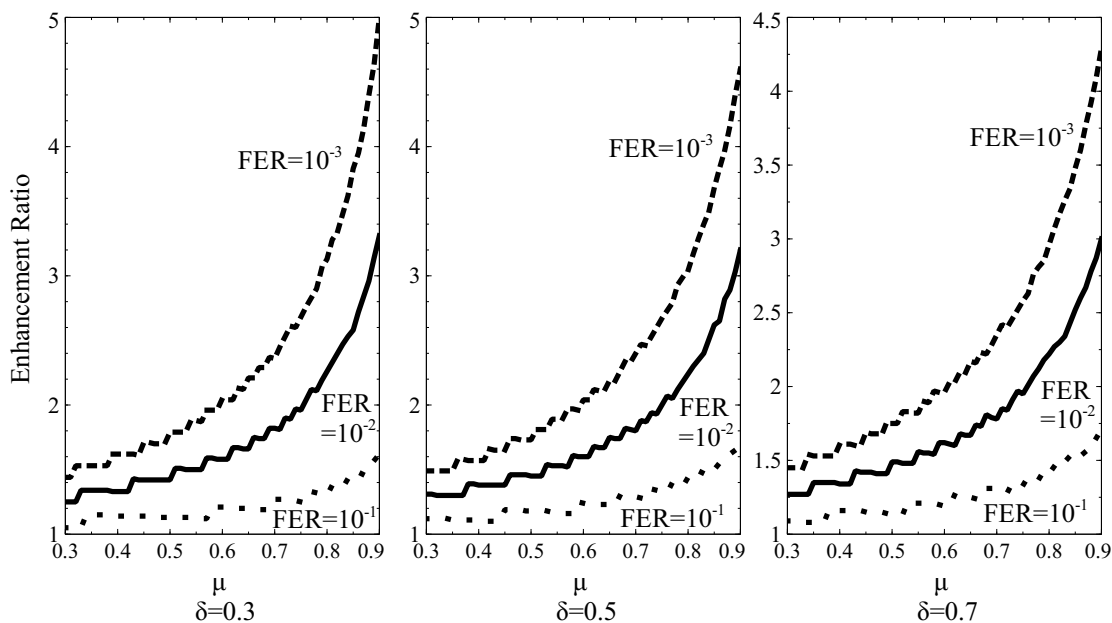


Figure 4.7: Enhancement ratio compared to IR-HARQ as  $|\mathcal{R}| \rightarrow \infty$ .

The growth of  $E(n_{(f)})$  as  $|\mathcal{R}|$  increases, for a fixed target FER  $10^{-a}$ , is illustrated in Fig. 4.8. The curves show the enhancement ratio for  $1 \leq |\mathcal{R}| \leq 100$ , target FER of  $10^{-1}$  and  $10^{-2}$ , and  $(\delta, \mu) \in \{(0.5, 0.5), (0.8, 0.6), (0.6, 0.8)\}$ . It can be deduced from these curves that the enhancement brought by inter-frame coding over the IR-scheme is significant, even when the number of receivers is small. For example, the enhancement ratios for a target FER of  $10^{-2}$  and  $|\mathcal{R}| = 10$  exceeds the corresponding enhancement ratio, for a target FER of  $10^{-1}$  and  $|\mathcal{R}| \rightarrow \infty$ . The latter values themselves are significant, standing at 1.18, 1.25, and 1.46 for  $(\delta, \mu)$  equal to  $(0.5, 0.5)$ ,  $(0.8, 0.6)$ , and  $(0.6, 0.8)$  respectively.

## 4.5.2 Two-stage Scheme

To evaluate the coding-performance of the state-of-the-art two-stage scheme, the combination of the intra-frame code-rate and the erasure code-rate that results in the lowest effective frame-length should be found.

For clarity of the following discussion, the two-stage scheme is briefly described here. On the transmitter side, the  $N_F \cdot K$  information bits are encoded into  $N_T \cdot K$  bits, using an erasure code of rate  $R_E = \frac{N_F}{N_T}$ . The resulting  $(N_T \cdot K)$ -bit sequence is then partitioned into  $N_T$   $K$ -bit blocks that are intra-frame encoded into  $N_T$  rate- $R$  frames that are transmitted over the channel. On the receiver side, intra-frame decoding is applied on each received frame: if decoding fails, the frame is dropped



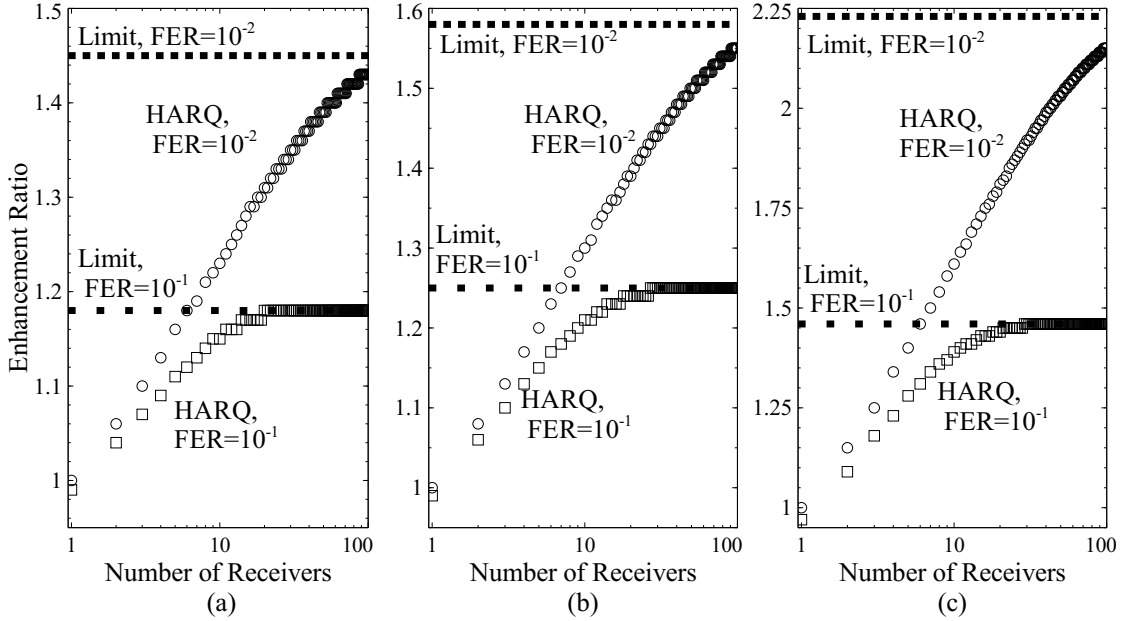


Figure 4.8: Enhancement ratio compared to IR-HARQ versus  $|\mathcal{R}|$ , for  $\text{FER} \in \{10^{-1}, 10^{-2}\}$  and  $(\delta, \mu) = (0.5, 0.5)$  (a),  $(\delta, \mu) = (0.8, 0.6)$  (b), and  $(\delta, \mu) = (0.6, 0.8)$  (c).

and considered erased; else, if decoding succeeds, the  $K$  bits corresponding to the frame are recovered and collected. The collected bits, resulting from the intra-frame decoding of the  $N_T$  frames, are then forwarded into erasure decoding. In case of erasure-decoding success, the  $N_F \cdot K$  information bits are recovered. The transmitter is assumed to know the distribution  $(\delta_{(\omega)})$ . Besides, for simplicity, no further feedback from the receiver to the transmitter is assumed.

For the asymptotic case, that is when  $N_F \rightarrow \infty$ , the optimal rate of the erasure-code  $R_E$  is  $1 - \Gamma$ ,  $\Gamma$  being the frame-error-rate (FER) or, equivalently, the intra-frame decoding-failure rate. If the intra-frame code-rate  $R$  is set to  $\frac{K}{N+i \cdot \Delta}$ ,  $i \in \mathbb{Z}^+$ , that is the frame-length is set to  $N+i \cdot \Delta$ , the FER is equal to  $\Gamma = \sum_{\omega=i+1}^{\infty} \delta_{(\omega)} = \delta \cdot \mu^i$ . Unlike the proposed inter-frame coding and the considered IR-HARQ schemes, the intra-frame code rate  $R$  in the two-stage scheme can be set such that the frame-length is  $N+i \cdot \Delta$ , where  $i \in \mathbb{Q}^+$ . It can be reasonably assumed that for the general case of  $i \in \mathbb{Q}^+$ , the FER is also equal to  $\delta \cdot \mu^i$ . Therefore, the effective frame-length of the two-stage scheme when  $(R, R_E) = (\frac{K}{N+i \cdot \Delta}, 1 - \delta \cdot \mu^i)$  is denoted here by  $L^{(i)}$  and is equal to:

$$L^{(i)} = \frac{N + i \cdot \Delta}{R_E} = \frac{N + i \cdot \Delta}{1 - \delta \cdot \mu^i}.$$

The effective-frame length of the two-stage scheme is then the minimum, over

$i \in \mathbb{Q}^+$ , of  $L_{(i)}$ . Two notes are made in this regard. First,  $i$  is restricted to positive values to rule out the unrealistic case of having  $R = \frac{K}{N+i\Delta} > 1$  given that the exact value  $\frac{K}{N}$  is not set here. Second, the effective frame-length of the two-stage scheme is independent of the number of receivers. This is not unexpected given the fact that the two-stage scheme is a forward-error-control scheme that is based on coding. The enhancement ratio is then equal to:

$$\min_{i \in \mathbb{Q}^+} \frac{L_{(i)}}{\left(N + \frac{\delta}{1-\mu} \cdot \Delta\right)} = \min_{i \in \mathbb{Q}^+} \frac{1 + i \cdot \frac{\Delta}{N}}{\left(1 - \delta \cdot \mu^i\right) \cdot \left(1 + \frac{\delta}{1-\mu} \cdot \frac{\Delta}{N}\right)}.$$

The following result on the enhancement ratio can be derived:

**Lemma 5.** *For the two-stage-scheme, the enhancement ratio is upper bounded by  $\frac{1}{1-e^{-1}} = 1.582$*

*Proof.*

$$\begin{aligned} & \min_{i \in \mathbb{Q}^+} \frac{1 + i \cdot \frac{\Delta}{N}}{\left(1 - \delta \cdot \mu^i\right) \cdot \left(1 + \frac{\delta}{1-\mu} \cdot \frac{\Delta}{N}\right)} \\ & \leq \left( \frac{1 + i \cdot \frac{\Delta}{N}}{\left(1 - \delta \cdot \mu^i\right) \cdot \left(1 + \frac{\delta}{1-\mu} \cdot \frac{\Delta}{N}\right)} \right)_{i=\frac{\delta}{1-\mu}} \\ & = \left(1 - \delta \cdot \mu^{\frac{\delta}{1-\mu}}\right)^{-1}. \end{aligned}$$

Consider the function  $f(x) = x \cdot \mu^{\frac{x}{1-\mu}}$  over the range  $[0, 1]$ . By simple calculus, it can be checked that for  $x^* = -\frac{1-\mu}{\ln(\mu)}$ ,  $\frac{\partial f}{\partial x} = 0$  and  $\frac{\partial^2 f}{\partial x^2} < 0$ . Therefore,  $f(x^*)$  is the maximum of  $f(x)$  for  $x \in [0, 1]$ .

$$\begin{aligned} f(x^*) &= -\frac{1-\mu}{\ln(\mu)} \mu^{-\frac{1}{\ln(\mu)}} \\ &= -\frac{1-\mu}{\ln(\mu)} \cdot e^{-1} < e^{-1}. \end{aligned}$$

The last inequality obtained from  $-\log(\mu) > 1 - \mu$  for  $\mu < 1$ . Therefore,  $\left(1 - \delta \cdot \mu^{\frac{\delta}{1-\mu}}\right)^{-1} < (1 - e^{-1})^{-1}$ .  $\square$

The curves plotted in Fig. 4.9 show  $\frac{L_{(i)}}{\left(N + \frac{\delta}{1-\mu} \cdot \Delta\right)}$  versus  $i$ . The curves correspond to all possible combinations of  $\delta \in \{0, 3, 0.5, 0.8\}$  and  $\mu \in \{0.5, 0.7, 0.85\}$ , for  $\frac{\Delta}{N} = 0.1$ . Two observations can be made accordingly. First, for all the considered  $(\delta, \mu)$  pairs, except one, we have: the minimum of  $\frac{L_{(i)}}{\left(N + \frac{\delta}{1-\mu} \cdot \Delta\right)}$  occurs for strictly positive values of  $i$ . This implies that restricting  $i$  to positive values has no impact

on the computed enhancement ratios, particularly when these ratios are relatively high. Second, the enhancement ratio, in general, grows as the channel statistical parameters “worsens”, that is as  $\delta$  and  $\mu$  increase. For example, while the enhancement ratio is  $\sim 1.2$  for  $(\delta, \mu) = (0.3, 0.5)$  and  $\sim 1.25$  for  $(\delta, \mu) = (0.5, 0.5)$ , it is  $\sim 1.35$  for  $(\delta, \mu) = (0.5, 0.7)$ ,  $\sim 1.4$  for  $(\delta, \mu) = (0.5, 0.85)$ , and  $\sim 1.5$  for  $(\delta, \mu) = (0.8, 0.85)$ .

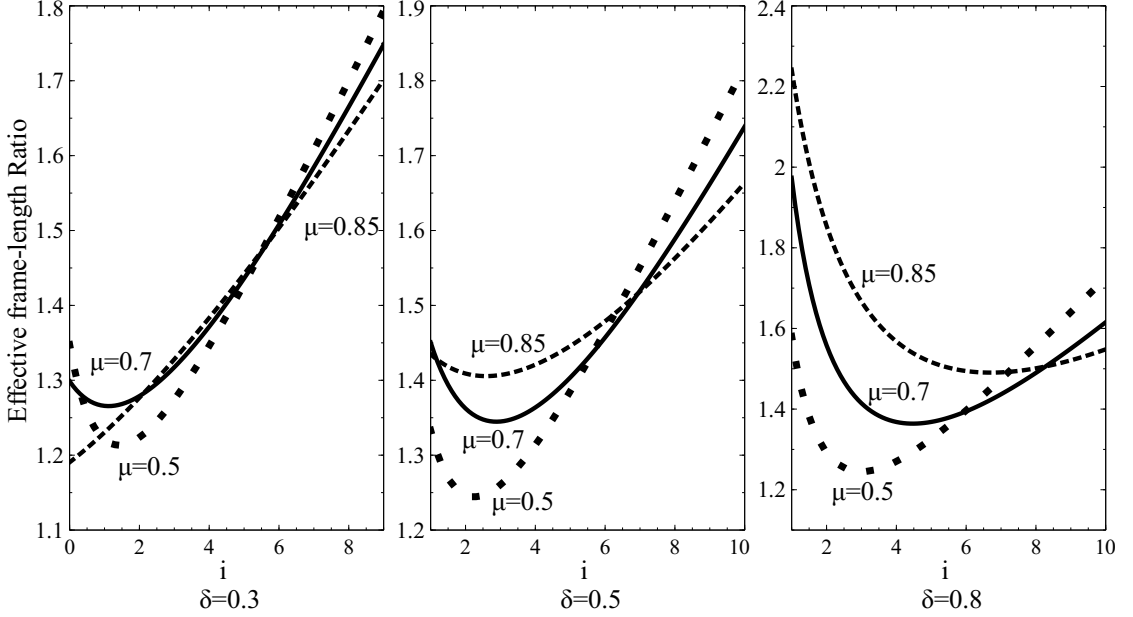


Figure 4.9: *Effective frame-length* ratio versus  $i$ , for different values of  $\delta$  and  $\mu$ .

The variation of the enhancement-ratio with the value of  $\mu$  is illustrated in Fig. 4.10, assuming  $\frac{\Delta}{N} = 0.1$ . Three curves are plotted. One curve corresponds to a fixed value of  $\delta = 0.99$ ; the other two curves assume  $\delta$  varies with  $\mu$  such that  $\delta = \mu^3$  and  $\delta = \mu^6$  respectively. The assumption that the two parameters of the distribution  $(\delta_{(\omega)})$ ,  $\delta$  and  $\mu$ , are related is justified as follows: the intra-frame decoding-failure rate, or equivalently FER, is  $\delta$  when the frame-length is  $N$ , and  $\delta \cdot \mu^{\frac{N}{\Delta}} = \delta \cdot \mu^{10}$  when the frame-length is  $2 \cdot N$ . If the same  $N$ -bit frame is transmitted twice under independent channel-state instances and each frame is independently intra-frame decoded, the probability of recovering the frame is  $1 - \delta^2$  for total number of sent bits of  $2 \cdot N$ . This probability is typically less than the probability of intra-frame decoding-success of a  $(2 \cdot N)$ -bit frame, formed by concatenating 10 subframes to the  $N$ -bit frame; therefore,  $1 - \delta^2 < 1 - \delta \cdot \mu^{10}$  implying  $\delta > \mu^{10}$ . Therefore, the assumed relations of  $\delta = \mu^3$ , and  $\delta = \mu^6$ , consist merely a simple way to satisfy the condition  $\delta > \mu^{10}$  over the full considered range of  $\mu$ . Three observations can be made from the figure. First, the derived upper bound of  $\frac{1}{1-e^{-1}}$

on the enhancement ratio is relatively tight: the maximum value of the plotted enhancement ratios is 1.55 and it occurs when  $(\delta, \mu) = (0.99, 0.94)$ . Besides, even when  $\delta$  varies with  $\mu$ , such that  $\delta = \mu^3$ , the enhancement ratio exceeds 1.5 for  $0.89 \leq \mu \leq 0.95$ . Second, the variation of the enhancement ratio with  $\mu$ , for the three curves, follows the same trend. The enhancement ratio increases as  $\mu$  increases to a value near 0.94, at which the maximum is attained, and then it decreases as  $\mu$  goes from 0.95 to 1. The rate of the increase in the plotted enhancement ratio is such that the enhancement ratio becomes relatively high as  $\mu$  exceeds 0.75 or 0.8. Third, for the same value of  $\mu$ , higher enhancement ratios are obtained for higher  $\delta$  values. Overall, the observed trend that the enhancement ratio increases with increasing  $\delta$  and  $\mu$  can be qualitatively justified as follows: for good channel parameters, that is relatively low values of  $\delta$  and  $\mu$ , the effective frame-lengths in each of the compared schemes is relatively close to  $N$ . This implies that the enhancement ratio is relatively close to 1 as well. In comparison, as  $\delta$  and  $\mu$  increase, the effective-frame length increases in both schemes. That is the value of the effective frame-length minus  $N$  increases, relative to  $N$ . Consequently, the reduction in this value achieved by inter-frame coding compared to the two-stage scheme becomes more significant in the computation of the enhancement ratio.

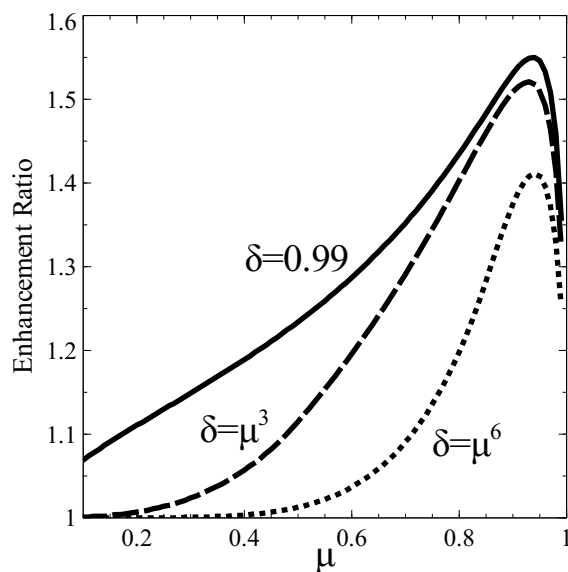


Figure 4.10: Enhancement ratio curves versus  $\mu$ , for different values of  $\delta$ .

## 4.6 Simulation Results

The simulation experiments are intended to highlight aspects of inter-frame coding that need further investigation, namely the coding performance, memory size requirement, latency time, and intra-frame decoding-attempts count. In accordance with this goal, the decoding scheduling described in 4.2 is simulated using Matlab, assuming the channel model developed in 4.3. Thus, in the simulation process,  $N_F$   $\kappa$ -values are sampled using  $(\delta_\omega)$  instead of generating LLR values, and the intra-frame decoding procedure of a frame  $f$  is replaced by checking if the number of subframe neighbours of  $f$  which have  $f$  as their only unrecovered neighbour is greater than or equal to  $\kappa(f)$ . To simplify its description,  $(\delta_\omega)$  is assumed to form a geometric progression described by two numbers  $\delta, \mu < 1$ , such that:  $\delta = \sum_{\omega=1}^{\infty} \delta_\omega$  and  $\delta_{\omega+1} = \mu \cdot \delta_\omega, \forall \omega \geq 1$ . It can be reasonably presumed that, due to their qualitative nature, the conclusions obtained from the simulations are also valid for a large class of channel-characterizing distributions  $(\delta_\omega)$  that do not follow the geometric progression model.

The performance of an inter-frame code, under distribution  $(\delta_\omega)$ , is measured in terms of its failure-rate, defined in this section as the probability that inter-frame decoding fails to recover all the received  $N_F$  frames. In accordance with this definition, the cases of very high  $\kappa$ -values are overlooked. That is, for any distribution described by some  $(\delta, \mu)$  couple,  $\delta_\omega$  is set to 0 for  $\omega > \omega_0, \omega_0 = 10$  in this section. In broadcast communication, the failure-rate under a distribution  $(\delta_\omega)$  is sufficient to describe the performance of the inter-frame code for all receivers with corresponding distributions equal to  $(\delta_\omega)$ , regardless of the number of receivers. The code performance can be presented using curves showing the failure-rate versus the mean  $\kappa$ -value  $\sum_{\omega} \omega \cdot \delta_\omega$ , where each curve corresponds to distributions with the same  $\mu$  but different  $\delta$ .

The performance of an inter-frame code relative to that of the conventional two-stage scheme, under some  $(\delta_\omega)$ , is obtained by comparing their *effective frame-lengths*, defined as the average number of transmitted bits per information block required to achieve a target failure-rate. For sake of comparison, the target failure-rate is set to the decoding failure-rate of the considered inter-frame code. The effective frame-length of the inter-frame code is  $N + \frac{K_S}{N_F} \cdot \Delta$ . In the two-stage scheme, the effective frame-length is obtained as follows: for a frame length of  $N+i \cdot \Delta$ , the rate of the erasure code,  $R_E(i, (\delta_\omega))$ , is chosen such that the probability that the number of  $\kappa$ -values between 0 and  $i$  is less than  $N_F$  is below the target failure-rate. The reason is that if the number of successfully-decoded frames is less than  $N_F$ , erasure decoding will fail to recover all the  $N_F$  information blocks. The effective frame length is then  $\min_i N \cdot \frac{1+i \cdot \Delta/N}{R_E(i, (\delta_\omega))}$ . The ratio of the effective-frame length of the two-stage scheme over that of the inter-frame code, denoted the enhancement ratio, quantifies the improvement brought by inter-frame coding.

In broadcast communication, if  $F$  is the set of distributions  $(\delta_\omega)$  corresponding to the different receivers, the effective-frame length of the two-stage scheme is  $\min_i \max_{(\delta_\omega) \in F} \frac{N+i\cdot\Delta}{R_E(i,(\delta_\omega))} \geq \min_i \frac{N+i\cdot\Delta}{R_E(i,(\delta_\omega))} \forall (\delta_\omega) \in F$ .

The other aspects considered here are the LLR-memory size, latency time, and intra-frame decoding-attempts count. An accurate estimation of these metrics requires a detailed description of the decoder architecture and transmission scheduling. Instead, the following simplifying assumptions are made here on the inter-frame decoding scheduling in 4.2.2: 1) the throughput of the intra-frame decoder is infinitely high, 2) and the subframe neighbors of a frame are received prior to the frame itself. The inter-frame decoding time span can then be viewed as consisting of  $N_F$  time slots, where in each slot one frame is received, and as many intra-frame decoding attempts as needed are made. The average of these attempts per slot is the intra-frame decoding-attempts count. This metric is strongly related to the power overhead of inter-frame decoding because, as concluded in 4.2.3, the intra-decoding process significantly surpasses the other processes involved in inter-frame decoding in terms of computational complexity. The required LLR-memory size metric is estimated through the *buffered-frame count* defined, for any time slot  $t$ , as the number of frames that are already received by the time  $t$  but not yet recovered. The underlying rationale is that for any such frame  $f$ , the LLR-memory stores, at time  $t$ ,  $\Lambda_N(f)$  in addition to the LLR-vectors corresponding to the received neighbor subframes of  $f$ . The fourth metric, the latency time of frame  $f \forall f \leq N_F$ , is defined as the time separating the receipt of  $f$  and its successful decoding. It is measured here in terms of the time slots elapsed between receiving  $f$  and recovering it. Four corresponding values can then be obtained from inter-frame decoding of a single set of  $N_F$  frames, which are the average and maximum values of each of the buffered-frame count and latency-time. The probability distribution of each of these values, under some  $(\delta_\omega)$ , can then be deduced through simulations.

Three codes with  $(N_F, K_S) = (121, 363)$  are constructed: *Code1*, *Code2*, and *Code3*. The simplified generator matrix of *Code1* is constructed as illustrated in Fig.4.11: a structured matrix that resembles the parity-check matrix of LDPC convolutional codes [88] is constructed, a row splitting is applied on this matrix according to some splitting scheme, and then a small number of randomly-constructed binary rows is concatenated atop of the formed matrix. The construction of *Code2* is similar to that of *Code1*, differing merely in the splitting scheme and in the additional step of concatenating a  $N_F \times N_F$  identity matrix atop of the matrix resulting from row-splitting. Construction of *Code3* is different from those of the two former codes in that row-splitting is applied on an  $11 \times 11$  matrix of  $11 \times 11$  permutation matrices. Two larger codes, *Code1'* and *Code3'*, are constructed using schemes that are similar to those used in constructing *Code1* and *Code3* respectively such

that  $(N_F, K_S) = (1210, 3630)$ . That is,  $\frac{K_S}{N_F}$  is kept fixed for the two values, 121 or 1210, of  $N_F$ .

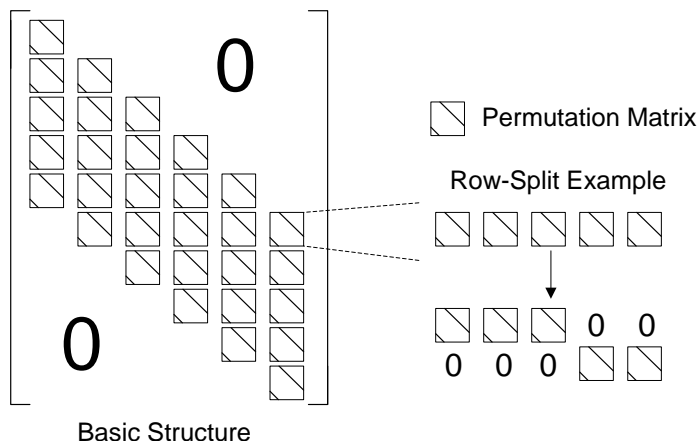


Figure 4.11: Illustration of the basic steps involved in constructing *Code1* and *Code2*. Note that the maximum number of permutation matrices per row is limited in the figure to 5 for clarity; it is 11 in the corresponding code-construction.

The coding performance of these codes under different distributions characterized by  $\mu \in \{0.6, 0.75\}$  is shown in Fig. 4.12. The relative performance of the codes is sensitive to the value of  $\mu$ : *Code1* outperforms *Code2* for  $\mu = 0.75$ , while *Code2* outperforms *Code1* for  $\mu = 0.6$ . This raises the problem of designing inter-frame codes with good performance over a wide range of  $\mu$ -values, needed in case the receivers have channels with close values of  $\sum_{\omega} \delta_{\omega}$  but different values of  $\mu$ . Besides, as seen from Fig. 4.12(b), better coding performance is obtained by increasing the code-size  $N_F$  from 121 to 1210.

It is noteworthy that the reported performance results, specifically for *Code1* (*Code1'*) and *Code2*, are highly dependant on the fact that the  $\kappa$ -values are independent and identically distributed. Different assumptions on the channel behaviour can lead to results significantly different than the previously obtained. For example, the following scenario is considered: for a range  $[f_1, f_2] \subset [1, N_F]$ , the value  $\kappa_{(f)}$  corresponding to a frame  $f_1 \leq f \leq f_2$  is sampled from a distribution characterized by  $(\delta, \mu) = (0.8, 0.85)$ , while if  $f \notin [f_1, f_2]$ ,  $\kappa_{(f)}$  is sampled from a distribution characterized by  $(\delta, \mu) = (0.4, 0.5)$ . This scenario can happen when the communication time is long enough, and consequently  $N_F$  is large enough, for the channel statistics to change. Due to its specific structure, *Code1'* performs poorly under such scenario: even when the span of  $[f_1, f_2]$ ,  $f_2 - f_1$ , is chosen such that the overall mean of the  $\kappa$ -values is  $1.45 \ll \frac{K_S}{N_F} = 3$ , inter-frame decoding fails with a probability  $\sim 1$ . In comparison, the performance of *Code3'* is much less

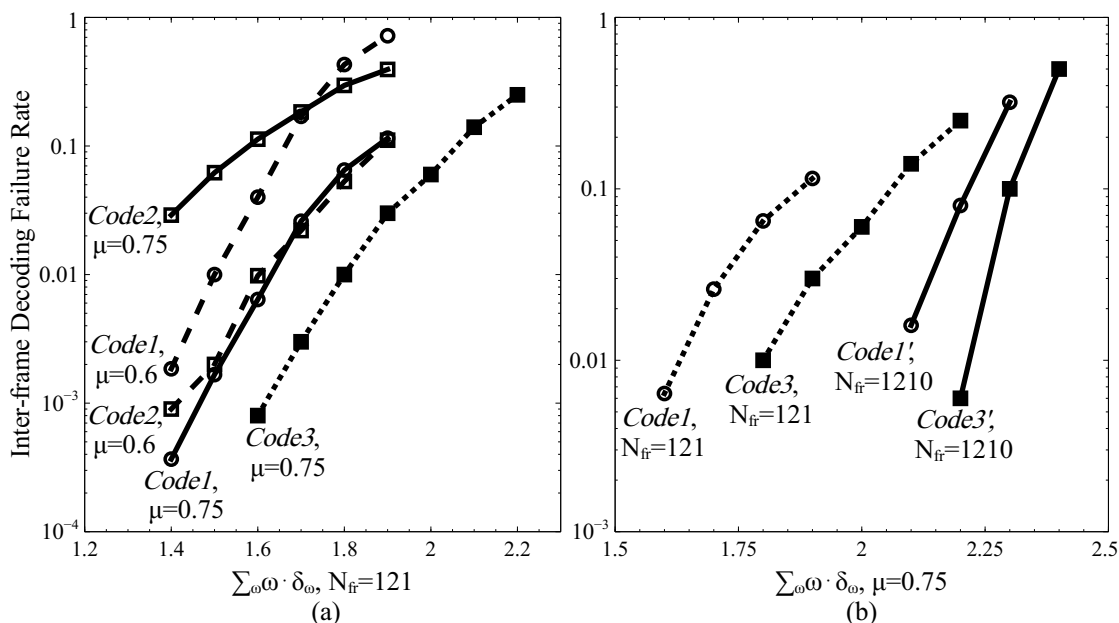


Figure 4.12: The *inter-frame decoding-failure rate* curves versus  $\sum \omega \cdot \delta_\omega$  for  $\mu \in \{0.6, 0.75\}$  and  $N_F \in \{121, 1210\}$ .

sensitive to such scenarios: for example, when the span of  $[f_1, f_2]$ ,  $f_2 - f_1$ , is chosen such that the overall mean of the  $\kappa$ -values is 2.33, inter-frame decoding fails with a probability  $\sim 0.5$ .

The coding performance of each constructed code is compared to that of the two-stage scheme. For each code, the distribution ( $\delta_\omega$ ) under which the comparison is done, is chosen using Fig.4.12 such that the corresponding inter-frame code failure-rate is equal to the target value of  $10^{-2}$ . The results can be summarized as follows, assuming  $\frac{\Delta}{N} = 0.1$ : the enhancement ratio is  $\sim 1.22$  for  $N_F = 121$  and  $\mu = 0.6$ , and  $\sim 1.3 - 1.35$  for  $\mu = 0.75$  and  $N_F \in \{121, 1210\}$ . It is noteworthy the the difference in the coding performance of the constructed codes does not result in large differences in the enhancement ratio. For example, the enhancement ratio is  $\sim 1.32$  for *Code1* and  $\sim 1.35$  for *Code3*. The following broadcast scenario is then simulated: the distributions corresponding to the different receivers have the same values of  $\sum \omega \cdot \delta_\omega = 1.5$  but different  $\mu \in \{0.6, 0.75\}$ . Assuming *Code1* is deployed, the resulting enhancement ratio ( $\sim 1.3$ ), for a target failure rate of  $10^{-2}$ , is close to that observed for the aforementioned case of  $\mu = 0.75$ . Overall, the results show that inter-frame coding enhances the data-rates; yet, they also show such enhancement varies with  $\mu$ . This verifies the asymptotic analysis done in Section 4.4.

Two possible tradeoffs involving the memory-size requirement and coding performance can be deduced from Figs. 4.12, 4.13, and 4.14, the two latter figures



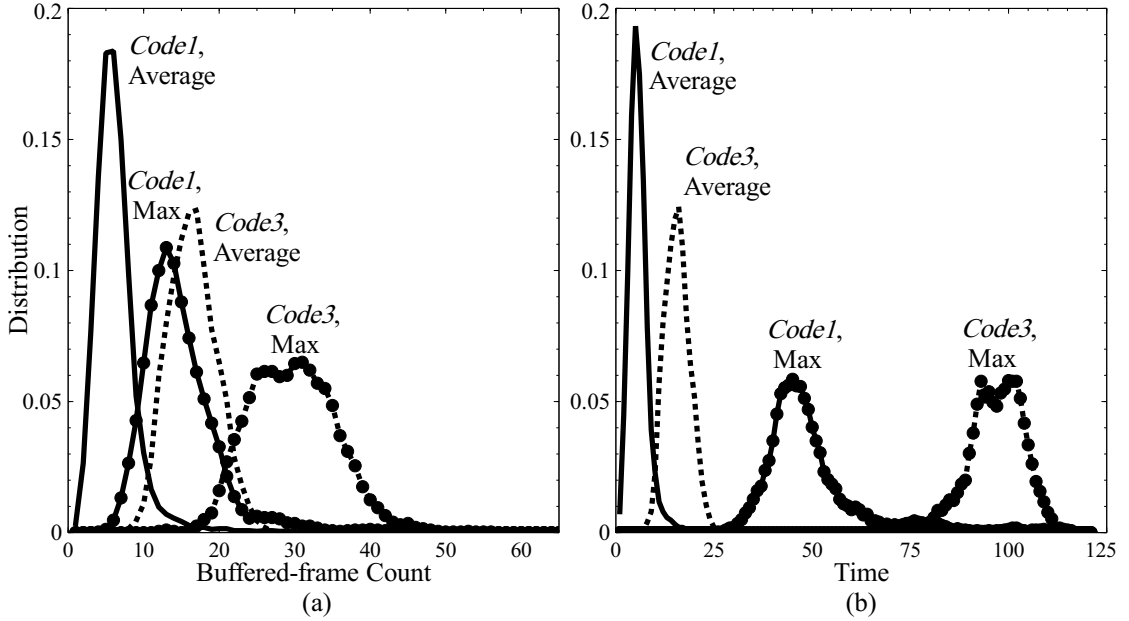


Figure 4.13: Distributions of the maximum and average buffered-frame count and latency time for *Code1* and *Code3*,  $\mu = 0.75$ .

showing the distribution of the average and maximum values of the buffered-frame count and latency time for the constructed codes. This first tradeoff considers inter-frame codes with the same size, and is exemplified as follows: in terms of coding performance, *Code3* outperforms *Code1*. However, as shown in Fig. 4.13(a), the maximum buffered-frame count is likely to be significantly less for *Code1* compared to *Code3*. If *Code1* is deployed, the case where the maximum buffered-frame count is relatively high, for example  $\geq 35$ , is very infrequent. Therefore, significant savings in memory can be obtained from deploying *Code1* instead of *Code3*. As seen from the Figs. 4.12(b) and 4.14(a), this tradeoff persists for *Code1'* and *Code3'* as well. The involved tradeoff can be attributed to the structure of *Code1* (*Code1'*): the initially transmitted frames are neighbors to relatively-low degree subframes; besides, each subframe has the indices of its frame neighbors bounded within a small range of  $[1, N_F]$ . This leads to a relatively low latency time for recovering a frame, and thus to a lower number of frames buffered at a time. The second tradeoff considers codes with different sizes. As can be deduced from Figs. 4.13(a) and 4.14(a), the improvement in the coding performance brought by increasing  $N_F$ , is accompanied by an increase in the buffered-frame count. However, while  $N_F$  increases by a factor of 10, the general increase in the buffered-frame count is by a much less factor for *Code1'*, and by slightly larger factor for *Code3'*. As seen from Figs. 4.13(b) and 4.14(b), similar tradeoffs exist between the latency time and coding performance.

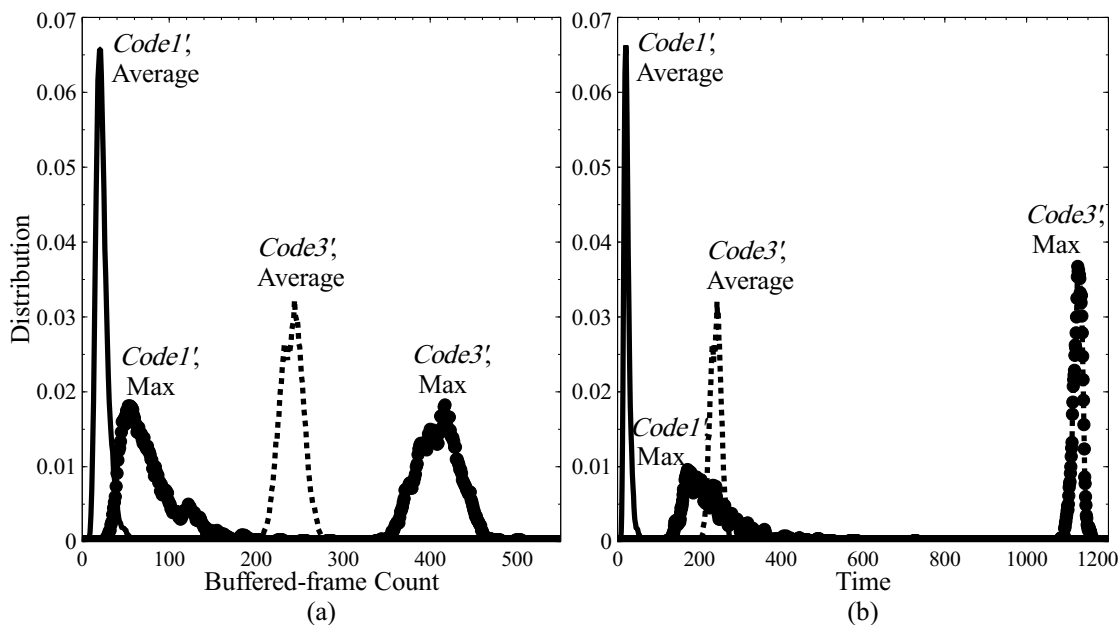


Figure 4.14: Distributions of the maximum and average buffered-frame count and latency time for *Code1* and *Code3*,  $\mu = 0.75$ .

The intra-frame decoding-attempts count is obtained for  $\sum_{\omega} \omega \cdot \delta_{\omega} = 1.4$  and  $N_F = 121$ . When  $\mu = 0.6$ , this figure is equal to 1.58 for *Code1* and 1.5 for *Code2*. When  $\mu = 0.75$ , it is equal to 1.63 for *Code1*, 1.57 for *Code2*, and 1.8 for *Code3*. An equivalent metric in the two-stage scheme is the average value of the number of frames on which intra-frame decoding is attempted before the number of successfully-decoded frames reaches  $N_F$ . It is equal to 1.18 and 1.14 when  $\mu$  is equal to 0.6 and 0.75 respectively, and can be decreased further to less than 1.1 for marginal losses in the resulting data rates. The relatively high intra-frame decoding-attempts count motivates the problem raised in subsection 4.1.2 which is estimating the success probability of intra-frame decoding of a frame without performing the decoding itself.

## Chapter 5

# Architecture-Aware Raptor Codes

This chapter includes the second part of the research work presented in this dissertation, namely the construction of architecture-aware Raptor codes. It should be noted that, although the construction procedure(s) proposed in this chapter does not specify a frame-length range, Raptor codes considered in this dissertation are assumed to have short to moderate block lengths ( $\sim$  few Kbits).

The design of rate-compatible codes is driven by the following requirement: the codes should have good coding performance in all the scenarios in which they are applied; besides, they should allow a hardware-efficient decoder implementation. A set of challenges is faced in the design of such codes. Some of these challenges gain increased significance in the newly proposed IIF coding scheme, in which rate-compatible codes are deployed. Others challenges are already faced in the design of LDPC rate-compatible codes, but their significance is aggravated in Raptor code construction due to the inherent irregularity features of Raptor codes. The significance of the Raptor-specific challenges means that the following question has to be considered first: *What are the possible advantages of deploying Raptor codes, at the PHY-layer, as the rate-compatible intra-frame codes?* Thereafter, the main question follows: *Can Raptor codes be constructed such that they have, simultaneously, good coding performance and hardware-efficient decoder architectures?*

A partial answer to the previous questions is given in this chapter. The rationale underlying this answer is described next. First, a brief qualitative analysis is done to show that Raptor codes have features that may lead to good coding-performance in the communication scenarios in which the performance of other codes deteriorate. The rest of the work is focused on answering whether Raptor codes can be constructed to pertain these hardware-unfriendly features while leading to hardware-efficient implementations. Answering this question is crucial in two aspects. First, it proves that deploying Raptor codes does not result in high

hardware overhead compared to other LDPC codes, and therefore that coding-performance is the major factor in comparing LDPC to Raptor codes. Second, it limits the design space of good-performing Raptor codes to the subclass of architecture-aware codes, that is it imposes hardware-related constraints on the construction of good-performing Raptor codes. A code construction framework is proposed in this dissertation in regard to these two aspects. The framework is hardware-oriented as it *maps the decoding of the highly irregular Raptor code into row processing of a regular matrix*. The framework simplifies the design process by partitioning the design of the code, and thereafter decoder, into a set of disjoint, albeit related, subproblems. The methods it involves in each of these subproblems are flexible enough to allow a large number of codes to be realized using the same architecture, under limited hardware reconfigurability. This means that a large space of candidate architecture-aware codes are available for performance-driven code search/design. Frame-error-rate simulations done on sample constructed codes show the potential of the proposed framework to produce good-performing codes; yet, no comprehensive analysis of the coding-performance of the resulting architecture-aware Raptor codes is presented in this dissertation.

The organization of this chapter is described next. First, the deployment of rate-compatible Raptor codes at the PHY-layer is discussed. Second, a code construction framework is proposed; its architectural implications are clarified by describing the resulting decoder architecture and scheduling. The proposed framework breaks the design problem into three subproblems: code structuring through source matrix construction, precode construction through row-merging, and LT-code construction through subcode generation. Each of these subproblems is then approached, and the proposed solutions are associated to miscellaneous issues of hardware-efficiency and/or coding-performance. Third, sample codes are constructed according to the proposed construction flow, and their performance under AWGN channels is simulated. The resulting FER curves are comparable to that of standardized LDPC codes. This shows the potential of the proposed construction scheme to generate codes that have good coding performance. Other hardware-efficiency metrics are the average number of decoding iterations and memory size required in decoding the constructed Raptor codes. Their values are obtained for the constructed sample codes, and the impact of the different proposed construction techniques on the hardware-efficiency of decoding is therefore demonstrated. As a proof of concept, a serial Raptor decoder is synthesized in 65 nm, 1.2 V CMOS technology. Hardware simulations show that the decoder, decoding a rate-0.4 code instance, achieves a throughput of 36 Mb/s at SNR of 1.5 dB, dissipates an average power of 27 mW and occupies an area of 0.55 mm<sup>2</sup>.

## 5.1 Raptor Codes: Motivation and Challenges

The first step in evaluating Raptor codes is to consider the occurrence of the *intra-frame varying channel condition* scenario in the proposed IIF coding scheme. Raptor codes are projected, through a qualitative analysis, to outperform other LDPC codes in such scenario, and, in addition, to have a minimum distance that is relatively easy to analyze and control over a wide range of code-rates. The features of Raptor coding leading to these possible advantages are identified throughout the analysis. Finally, the challenges facing hardware-efficient Raptor decoder implementation are considered. They include, but are not confined to, the aforementioned Raptor code features.

### 5.1.1 Intra-frame Varying Channel Condition in IIF Coding

First, the *out-of-order increment concatenation* scenario is considered. In IIF decoding, the progress of increment concatenation is dependent on the progress of the IIF decoding process itself: the concatenation of a subframe  $s$  to frame  $f$  occurs if, and only if, the set of unrecovered neighbor frames of  $s$  is reduced to a singleton  $\{f\}$ . Besides, the decoding process is random because the  $\kappa$ -values corresponding to the involved  $N_F$  frames are themselves random. This means that *out-of-order increment concatenation* is possible: for example, increment  $\Delta(f, i)$  can get concatenated to  $f$  during IIF decoding while  $\Delta(f, i - 1)$  does not, for some integer  $i > 1$ . It should be noted, however, that the probability of the out-of-order increment-concatenation events is a function of many factors, among which is the IIF code design. For example, consider a frame  $f$  and two integers  $j > i$ , and denote the subframes for which  $\Delta(f, i)$  and  $\Delta(f, j)$  are inputs to the XOR forming them as  $s_i$  and  $s_j$  respectively. If the degree of  $s_j$  is much higher than the degree of  $s_i$ , then there is a low probability that the set of unrecovered neighbors of  $s_j$  is reduced to a singleton  $\{f\}$  before the set of unrecovered neighbors of  $s_i$ . Consequently, it is much more probable that  $s_i$  is concatenated to  $f$  before  $s_j$ .

The second scenario considered here, namely *very bad initial channel-condition*, can happen in unicast communication, but is much more probable in the broadcast communication, under IIF coding. The reason is that, in IIF coding, transmission of a frame  $f$  and its neighboring subframes can be separated by a time span that is long enough for the channel to decorrelate. Subsequently, it is possible that a high SNR disparity exists between the transmission of the initial  $N$ -bit frame and that of its neighbor subframes.

## 5.1.2 Raptor Coding Features

The significance of the peculiar Raptor code features in the *intra-frame varying channel condition* scenario is understood here by comparing these features to that of the protograph-based extended LDPC codes explained in 3.4.2.

One feature of LT-encoding is that each generated LT output bit is defined as a XOR of a subset of the  $K_{LT}$  LT input bits. This definition is relevant in Raptor decoding in the following sense: when an increment is concatenated to the frame, the resulting change induced in the Raptor graph, on which decoding is applied, is that the check-node partition is incremented by  $\Delta$  check nodes. These latter nodes correspond to the LT output bits included in the increment, and their degrees are the degrees of the corresponding respective output bits included. This is unlike the case of LDPC codes, where out-of-order increment-concatenation can result in the decoding algorithm applied on a randomly-punctured graph (see 3.4.3). Besides, if the LT-degree distribution changes slowly with increasing code-size then the effect of out-of-order increment-concatenation, involving increments of close indices, can be minimized.

Another feature of LT-encoding is that the encoding of each LT output bit is performed independently from others. This means that each increment can include output bits of different degrees. The significance of this property is clarified in dealing with the extreme case of *bad initial channel-condition* scenario, in which the channel is extremely noisy (deep fading, effectively erasure) when the initial rate- $R_h$  frame is transmitted, and is noiseless otherwise. It should be noted that this extreme case is discussed here for the mere purpose of clarifying the distinct features of Raptor coding. In fact, as seen in Chapter 4, the premise of IIF coding is that the polarized behaviour of the channel (deep fading versus noiseless), corresponding to such extreme case, is not the main cause of intra-frame decoding failure. Iterative decoding in this case is equivalent to iterative erasure decoding. By including in each increment a portion, possibly varying across increments, of low degree bits (e.g. degrees 1-5), a frame can be recovered by collecting a sufficient, albeit non-optimal, number of increments. In comparison, in protograph-based LDPC codes, most of the systematic bits are transmitted in the initial  $N$ -bit frame and thus retransmission of degree-1 output bits involves a repetition, in case the *bad initial channel-condition* scenario does not happen. Besides, the degrees of the output bits change every  $Q$  bits,  $Q$  being the replication factor. Therefore degree-1 output bits are sent in very few increments. If none of these increments are collected/concatenated, the frame cannot be recovered. To avoid this, the replication factor can be made small so that 1 increment involves adding more than 1 check-node to the protograph. This, however, restricts both the code design and the decoder parallelism which is dependant on the minimum replication factor. The former argument can be equivalently stated as follows: in

protograph-based codes, iterative erasure decoding is applied on the protograph. In LT codes, it is applied on a randomly-constructed graph that is larger than the LDPC protograph by a factor equal to the replication factor. Since small-sized codes (i.e.  $< 100$ ) perform worse than bigger-sized code, it can be expected that less increments will be needed in the LT code to recover the frame.

*Minimum Distance Growth:* The two-code composition of a Raptor code is crucial to increase the minimum distance, since the LT code has a very low minimum distance that equals the minimum LT bit-node degree. The code minimum-distance is important because a high number of low-weight codewords leads to relatively high error-floors. In the IR-HARQ scheme, if an error-detection code is applied separately from the LDPC precode, the error-floor problem can be overlooked. However, other communication schemes may require low error-floors, and thus avoiding low-weight codewords becomes necessary, though not sufficient, to satisfy such requirement. The growth of the minimum-distance with rate decrease can be well approximated in the case of Raptor coding. Consider a  $K_{LT}$ -bit precode codeword, of weight  $d_w \ll K_{LT}$ , is LT-encoded into a  $N$ -bit frame. Define  $B_1$ ,  $|B_1| = d_w$ , to be the set of indices of the bits of value 1 in the weight- $d_w$  precode codeword and define  $d_v(i)$  to be the LT-degree of the bit-node of index  $1 \leq i \leq K_{LT}$ . The number of edges in the Raptor graph, whose bit-node neighbors have indices in  $B_1$ , is  $\sum_{i \in B_1} d_v(i)$ . If  $\sum_{i \in B_1} d_v(i) \ll N$ , which is typically the case especially for short block-lengths, the randomness of LT-encoding implies that the probability that two or more of the considered edges have the same LT check-node neighbor is small, and, therefore, the weight of the output  $N$ -bit codeword can be fairly approximated as  $\sum_{i \in B_1} d_v(i)$ . Now, assume the LT code is constructed such that the LT variable-node degree grows uniformly across all variable nodes, that is the variable-node degree-distribution is concentrated around 1 value, say  $d_v$ , that is itself a function of the LT-rate  $R_{LT}$ . Then, the weight of the output  $N$ -bit codeword will be  $\sim d_v \cdot d_w$ . This simple expression makes it relatively easy to incorporate the minimum-distance growth-control in the overall code design process, even for short-length codes. It however, assumes a LT graph with nearly equal variable-node degrees.

These largely qualitative arguments are not quantitatively studied or analyzed further in this dissertation. Instead, they motivate the question on whether the, possibly advantageous, Raptor code features can be pertained along with attaining hardware-efficient decoder implementation.

### 5.1.3 Hardware Efficiency Challenges

The Raptor decoder architecture follows the basic organization of the iterative LDPC decoder architectures discussed in 3.6. While the distinct Raptor encoding can bring possible advantages, it complicates the design of hardware-oriented good-

performing codes. The relation between the Raptor features and the hardware-related complications is summarized next.

1. **Decoding scheduling:** In most recent protograph-based LDPC decoders, decoding is mapped to processing 1, or more, edge(s) of the protograph per clock cycle. This leads to serial operation of the check-function units (CFUs), where each CFU reads 1 input-message per cycle. This scheduling cannot be readily adopted in Raptor decoding because, unlike the case of protograph-based designs, each LT output-bit is encoded independently from the others. This suggests that parallel CFUs, processing a check node in a single clock cycle, are more appropriate.
2. **Structuring versus Irregularity:** As discussed in 3.6, structuring the code leads to efficient memory-organization and interconnect. However, Raptor code structuring is made difficult by the two inherent irregularity features of Raptor coding: 1) the pseudo-random LT-encoding and 2) the two-code composition of the Raptor code.

LT-encoding generates check-nodes, or equivalently output bits, of different degrees in a pseudo-random manner. This has two implications: first, the pseudo-randomness in choosing the number and positions of the bit-node neighbors of a generated LT-check node leads to complications in the design of the bit-node memory and interconnect. Second, the check-function units have to process check nodes of different degrees. This different-degree processing can be performed by serial CFUs, but is complicated here by the fact that serial CFU processing is not straight-forward as in protograph-based LDPC codes. The latter conclusion is mentioned in the previous point on **Decoding scheduling**.

The two-code composition of a Raptor code leads to two main complications. First, the disparity in the degrees of the processed check-nodes is aggravated due to the presence of high-degree LDPC check nodes alongside relatively low-degree LT check nodes. Second, any structure imposed on the Raptor code must be compatible with the two code components: the LDPC precode and LT code.

3. **Decoding convergence speed:** In Raptor encoding, most of the LT-input bits are not included in the  $N$ -bit codeword. This means that even for zero noise, the LT decoder needs several iterations to recover the codeword. This problem is especially significant when the code-rate  $R = \frac{K}{N}$  is close to precode code-rate  $\frac{K}{K_{LT}}$ , that is the LT code rate  $\frac{K_{LT}}{N}$  approaches 1. In such cases, the number of LT check-nodes, generated pseudo-randomly, in the decoding graph may be insufficient to recover the bit values within a reasonable number of iterations, even in the absence of noise. Low decoding



convergence speed implies a high number of iterations is needed to recover a frame, and therefore high power consumption and low decoding-throughput. It is noteworthy that this problem resembles, in terms of the underlying causes and consequences, the problem of low decoding convergence speed and high-rate performance degradation in punctured LDPC codes (see 3.4.1).

Beside these complications, the Raptor code, and thereof the decoder architecture, must attain the following properties:

1. **Flexibility:** The decoder architecture must be flexible to decode Raptor codes with different LT check-degree distributions and different precode codes.
2. **Multi-length decoding capability:** The length  $K$  of the block that is input to the intra-frame encoder varies in time. Therefore, the decoder architecture should support a family of Raptor codes that are defined over equally spaced values of  $K$  or  $K_{LT}$ .
3. **Parallelism-awareness:** The constructed codes should be amenable to decoder implementations with different levels of parallelism. In this context, parallelism is related to the number of messages processed per one clock cycle.

## 5.2 Code Construction Framework

In this dissertation, a multi-stage framework is proposed for the construction of architecture-aware Raptor codes. The framework is described here, and its implications subsequently discussed. The steps involved in the framework can be stated as follows:

1. **Source Matrix Construction - Code Structuring:** A  $(P \cdot Q) \times (M \cdot Q)$  source matrix  $\mathbf{H}_0 = [\mathbf{h}_{ij}]$  is constructed, by replication, starting from a  $P \times M$  base matrix  $\mathbf{B} = [b_{ij}]$ . The entries of  $\mathbf{B}$  are elements in the group of permutations operating on a set of size  $Q$ . The replication is done by simply replacing each entry  $b_{ij}$  of  $\mathbf{B}$  by the  $Q \times Q$  matrix  $\mathbf{h}_{ij}$  in  $\mathbf{H}_0$ ,  $\mathbf{h}_{ij}$  being the permutation matrix corresponding to the group-of-permutations element  $b_{ij}$ . The source matrix is regular in the following sense: all the rows have the same weight  $M$ , all the columns have the same weight  $P$ , and the entries of value 1 in  $\mathbf{H}_0$  are distributed evenly across each row(column).
2. **Precode Construction - Row Merging:** A  $Q \times (Q \cdot M)$  matrix  $\mathbf{H}_0^{(P)}$  is constructed as follows:  $\forall 1 \leq i \leq Q$ , row  $i$  of  $\mathbf{H}_0^{(P)}$  is formed by bitwise logic-ORing of the  $C$  row vectors indexed by  $i + j \cdot Q$ ,  $j = 0 \cdots C - 1$ , of  $\mathbf{H}_0$ , an operation called here row-merging.  $C$  is called the merging factor. The constructed matrix  $\mathbf{H}_0^{(P)}$  has row-weight  $C \cdot M$  and column weight  $C$ . It can be either chosen as the parity-check matrix of the LDPC precode, here of rate- $(1 - \frac{1}{M})$ , or alternatively used to construct the precode parity matrix.
3. **LT Construction - Row Encoding:** A  $((P - C) \cdot Q) \times (M \cdot Q)$  submatrix  $\mathbf{H}_0^{(L)}$  of  $\mathbf{H}_0$  is formed from the rows of indices  $i \in \{Q \cdot C + 1, Q \cdot C + 2, \dots, Q \cdot P\}$  in  $\mathbf{H}_0$ , that is from the rows not involved in the aforementioned row-merging step. The LT code is generated by applying row-encoding on  $\mathbf{H}_0^{(L)}$  as described next. For each row of  $\mathbf{H}_0^{(L)}$ , the  $M$  LT input-bits corresponding to the entries of value 1 in the row are encoded using a row-specific subcode; that is, a number  $t$  of LT output-bits are generated according to the  $M$ -input  $t$ -output encoder corresponding to the subcode. The subcodes vary across rows, and can be chosen by sampling from a distribution on the available subcodes of input-length  $M$ . Thus, the LT encoding method generates subcodes, rather than individual output bits, pseudo-randomly. It can be thought of as a form of *Generalized LT encoding* analogous to the Generalized LDPC coding proposed early by Tanner [19]. The output bits, generated by the  $(P - C) \cdot Q$  row-encoding processes, are indexed in a well-designed way. For a rate- $R$  code, the output bits of indices  $1 \cdots \frac{K}{R}$  are chosen for transmission.

The framework constraints the Raptor code design in the following ways: 1) the degree of an LT output bit is upper bounded by  $M$ , 2) the degree of an LDPC check-node is upper bounded by  $C \cdot M$ , 3) the precode rate  $R_P$  is upper bounded by  $1 - \frac{1}{M}$ , and 4) the LT input-bit frame length  $K_{LT}$  is equal to  $M \cdot Q$ .

The full implications of the construction framework can only be realized by satisfying the following hardware-related condition: *a reconfigurable check function unit (CFU) can be designed, such that decoding is applied efficiently on each formed subcode **and** on the precode resulting from  $\mathbf{H}_0^{(P)}$* . This condition does not impose merely a hardware-design problem, but rather a problem of designing code and architecture jointly; it will be revisited shortly when discussing LT row encoding (5.4).

### 5.2.1 Implications on Design

The advantage of the proposed multi-stage framework is that it achieves a combination of 1) *design simplification*, 2) *flexibility*, and 3) *hardware-efficiency of decoding*. The framework features that lead to such combination are discussed next.

The first feature is that pseudo-random LT-encoding (*step 3*) and precode design (*step 2*) are decoupled from code structuring (*step 1*). As a result of such decoupling, two goals can be achieved in the source matrix construction step: first, a structure is imposed on the Raptor code which makes it amenable for hardware-efficient decoder implementation; second, the Raptor bipartite graph is constructed to be short-cycle free (especially 4-cycle free). While structuring is done in the first step of the construction framework, its architectural implications withstand the next two precode and LT construction steps. This is due to the fact that both the LT-code and precode are generated from  $\mathbf{H}_0$ , which means that the memory-access patterns are largely determined by the structure of this matrix. For example, architecture-aware source matrix construction would allow efficient partitioning of the bit-node memory into  $M$  banks, leading to regular memory-access patterns and simple interconnect. The source matrix construction step must however be merge-aware in the sense that it should be performed such that no short cycles or very low-weight precode codewords appear due to the row-merging operation in (*step 2*).

The second feature is that the row-merging step partially decouples the maximum check-node degree in the LDPC precode  $C \cdot M$  from the number of columns in the base matrix  $\mathbf{B}$  or equivalently from the number of bit-node memory partitions  $M$ . This means that the maximum check-node degree can be changed for different block lengths,  $K$ , or different code designs by merely changing the merging factor  $C$ . It can be reasonably assumed that the decoder architecture can be developed

to handle efficiently multiple possible values of  $C$ . On the other hand, the parameter  $M$  can be determined based on hardware considerations such as bit-node memory organization, efficiency of LT subcode decoding using the reconfigurable CFUs, etc  $\dots$ .

The third feature is that the irregularity of the LT code is visible only to the check-node processor and dealt with using *reconfigurable check-node processing*. A major implication is related to the hardware efficiency of decoding and can be summarized as: *Raptor decoding is mapped into row processing of the regular source matrix  $\mathbf{H}_0$* . The LT-encoding design process is recast as the problem of finding the appropriate subcode distribution, within a hardware-restricted space of possible subcodes. With this in regard, a large number of different LT code instances can be decoded using the same hardware, allowing the LT code description to change at different stages of the code/decoder design process, or even to change with the involved communication scenarios, i.e. in real time.

## 5.2.2 Decoding Scheduling

For sake of illustration, the decoding scheduling is set such that 1 row of  $\mathbf{H}_0$  is processed per clock cycle. Higher throughput can be achieved through partially-parallel decoder architectures where a number of rows is processed per cycle. For a rate- $R$  Raptor code, let  $\mathbf{H}'_0$  be the submatrix of  $\mathbf{H}_0$  including the rows involved in the precode-related row-merging and in the generation of the  $N$  LT output bits. The  $C$  rows corresponding to one row of  $\mathbf{H}_0^{(P)}$  are grouped adjacently in  $\mathbf{H}'_0$ . The resulting decoder architecture processes  $\mathbf{H}'_0$  serially, with a throughput of 1 row per cycle; therefore, processing one row of  $\mathbf{H}_0^{(P)}$  requires  $C$  cycles. Figure 5.1 illustrates the proposed architecture of the serial decoder implementing the turbo-decoding message-passing (TDMP) algorithm. The operation of the decoder proceeds as follows.

Let  $N_r$  be the number of rows of  $\mathbf{H}'_0$ . At sub-iteration  $\delta$  of iteration  $\tau$ , for  $\delta = 1, \dots, N_r$ , the decoder performs the following steps:

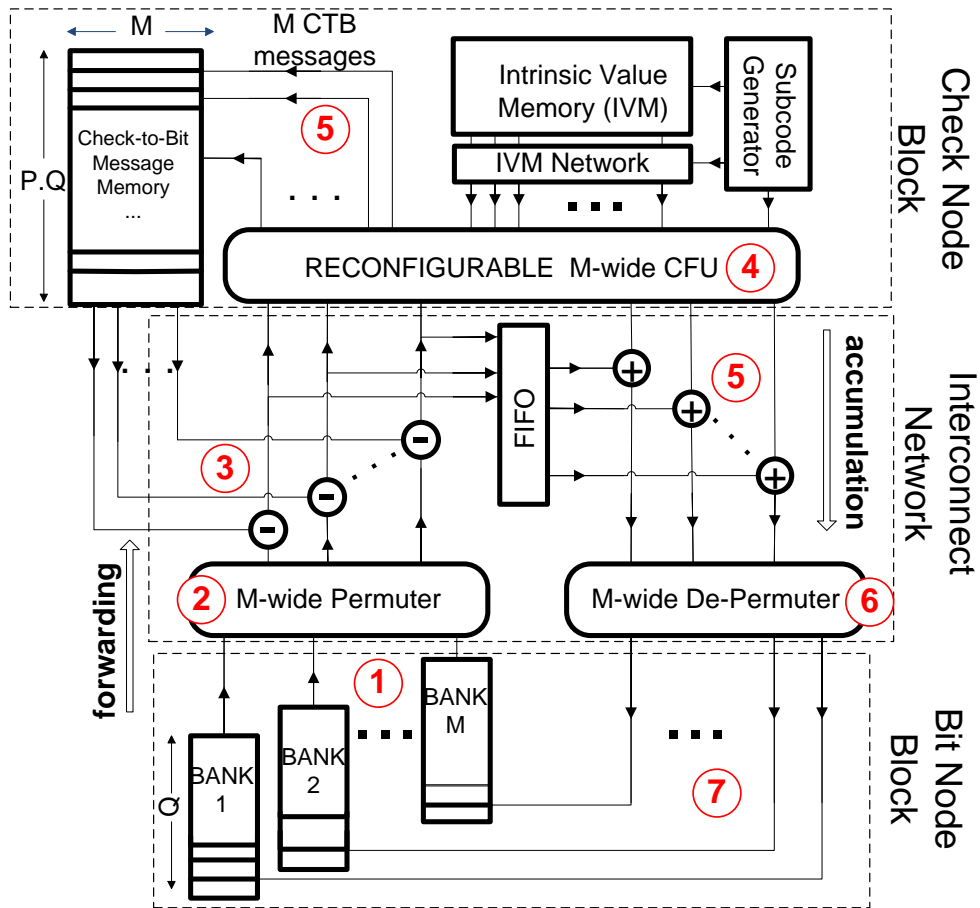
1. *Forwarding*: A  $M$ -dimensional message vector  $\mathbf{E}[\delta]$  is read from bit-node memory. Assuming  $e_i$ ,  $1 \leq i \leq M$ , is the edge corresponding to  $i$ th non-zero entry of row  $\delta$  of  $\mathbf{H}'_0$ , then  $\mathbf{E}_i[\delta]$  denotes the posterior extrinsic LLR of the bit-node connected to  $e_i$ .
2. *Permuting*:  $\mathbf{E}[\delta]$  is sent to a pseudo-random permuter  $\pi$  to generate  $\mathbf{E}'(\delta) \triangleq \pi(\mathbf{E}[\delta])$ .
3. *BTC operation*: The vector  $\mathbf{CTB}[\delta, \tau - 1]$  of dimension  $M$  is read from the check-node memory, and  $M$  new bit-to-check messages,  $\mathbf{BTC}[\delta, \tau]$ , are

computed as  $\mathbf{BTC}[\delta, \tau] = \mathbf{E}'[\delta] - \mathbf{CTB}[\delta, \tau - 1]$ , and forwarded to the CFU and a FIFO (First-In First-Out) buffer.

4. **CTB operation:**  $M$  new check-to-bit messages,  $\mathbf{CTB}[\delta, \tau]$ , are generated by performing either LT-decoding of the row-related subcode or LDPC decoding, depending on the row processed.
5. **Posterior reliability update and check-memory write-back:** The vector  $\mathbf{CTB}[\delta, \tau]$  is written back to check-node memory, and simultaneously added to the corresponding bit-to-check messages read from the FIFO buffer, to update the posterior reliability vector  $\mathbf{E}'$ :  $\mathbf{E}'[\delta] = \mathbf{BTC}[\delta, \tau] + \mathbf{CTB}[\delta, \tau]$
6. **Inverse permuting:** The vector  $\mathbf{E}'[\delta]$  is inverse permuted using the  $\pi^{-1}$ -block to generate:  
 $\mathbf{E}[\delta] \triangleq \pi^{-1}(\mathbf{E}'[\delta])$ .
7. **Accumulation:**  $\mathbf{E}[\delta]$  is written back to bit-node memory.

The bit-to-check messages are computed in a serial manner. There may be a discrepancy in the latency between LDPC- and LT- CFU processing resulting in the appearance of idle cycles when shifting between LT and LDPC decoding modes and slightly increasing the number of cycles needed to complete one iteration. The main components of the serial decoder are described below.

- . **Bit-Node Block:** The block performs the forwarding (1) and accumulation (7) steps. It is composed of  $M$  banks, where each bank holds  $Q$   $b$ -bit words and performs one read and one write operation per cycle. Address generation depends on the complexity of the permutations involved in the expansion from  $\mathbf{B}$  to  $\mathbf{H}_0$ .
- . **Interconnect Network:** It consists mainly of a permuter and inverse permuter that perform step 2 and step 6 of the scheduling respectively. It contains as well the adder and subtractor blocks. The subtractor block in the network forms the bit-to-check messages (Step 3), while the adders update the posterior extrinsic LLR values (Step 5).
- . **Check-Node Block:** It is composed of four main components.
  1. *Check-Node Memory:* It consists of  $M$  banks, each storing up to  $P.Q$  words and performing 1-read and 1-write per cycle. The memory banks are accessed sequentially.
  2. *Subcode-Index Generator:* It is composed of buffer and logic, used to generate the index of the subcode related to the processed row.



- ① Forwarding      ② Permuting      ③ BTC operation
- ④ CTB operation    ⑤ Posterior LLR Update and check-node memory writeback
- ⑥ Inverse Permuting
- ⑦ Accumulation

Figure 5.1: A serial Raptor decoder architecture.

3. *Intrinsic Values Memory (IVM)*: It stores the intrinsic channel LLRs of the LT check-nodes. Its organization is non-trivial due to the fact that different subcodes involve different output-bits, and thus the throughput of the memory varies across rows.

*Reconfigurable CFU*: It receives  $M$  bit-to-check messages, the intrinsic value messages, and the subcode index or precode information. It computes the check-to-bit-messages corresponding to one row in  $\mathbf{H}'_0$ .

The proposed decoding scheduling has three properties. First, it allows the pseudo-random permutation,  $\pi(\cdot)$ , to vary across rows. Second, the  $M$  input messages to a subcode-decoding are available simultaneously to the reconfigurable CFU which may simplify its implementation. Third, it allows using several permutation subgroups in expanding  $\mathbf{B}$  to  $\mathbf{H}_0$ , other than the cyclic-shift subgroup, without affecting memory and interconnect.

The design of the decoder blocks will be elaborated in the rest of the section, in light of the methods proposed to construct the source matrix, precode, and LT code.

*Partially-parallel decoder architectures*: To increase the decoder throughput by a factor of  $t$ ,  $t$  rows of  $\mathbf{H}_0$  can be processed simultaneously, i.e. per clock cycle. The resulting partially-parallel architecture has the same organization of the serial architecture. The required modifications are stated briefly in the following. The  $M$  banks of the bit-node memory are organized such that  $t$  messages are read from (written to) one bank every clock cycle. The check-node memory and IVM are partitioned into  $t$  blocks, each sized down by a factor  $1/t$  but retaining the access rate and pattern of its counterpart in the serial architecture. Likewise, since one CFU and one interconnect-network are required to process one row of  $\mathbf{H}_0$ , the number of CFUs and interconnect networks has to be replicated by  $t$ . One complication arises in the design of the partially-parallel architectures which can be stated as follows: it becomes increasingly difficult, as  $t$  increases, to satisfy the *TDMP timing condition* described in 3.6. The reason is that as more check nodes are processed per one cycle, the number of clock cycles needed to complete one decoding iteration decreases; therefore, it becomes more challenging to sufficiently spread the processing of check nodes that are neighbors to the same bit node, over time.

## 5.3 Code Structuring

In this work, two source matrix construction methods were devised. They will be described here in detail.

### 5.3.1 $p$ -based Replication

The  $p$ -based replication method applies two subsequent replications, each by a factor of a prime  $p$ . For simplicity of exposition, the method will be described here by defining these two replications rather than defining the base matrix  $\mathbf{B}$ . It can be described as follows:

**Input:** A prime  $p$ .

**Output:** A  $(p^3 \times p^3)$  matrix  $\mathbf{H}_0$  describing a girth-8 graph.

**Procedure:**

- . *Replication 1:* Construct a  $p \times p$  matrix  $\mathbf{L} = [l_{ij}]$  such that  $l_{ij} = i \cdot j \bmod p$ . Form the  $p^2 \times p^2$  matrix  $\mathbf{T} = [t_{ij}]$  by replacing each entry  $l_{ij}$  of  $\mathbf{L}$  by  $\mathbf{I}_{p \times p}^{l_{ij}}$ , where  $\mathbf{I}_{p \times p}^j$  is a  $p \times p$  identity matrix cyclically shifted to the right by  $j$  positions.
- . *Replication 2:* Form matrix  $H_0 = [\mathbf{h}^{(b)}_{ij}]_{\substack{1 \leq i \leq p^2 \\ 1 \leq j \leq p^2}} = [h_{ij}]_{\substack{1 \leq i \leq p^3 \\ 1 \leq j \leq p^3}}$  by replacing every scalar entry  $t_{ij}$  in  $\mathbf{T}$  by a  $p \times p$  matrix  $\mathbf{h}^{(b)}_{ij}$  such that:

$$\mathbf{h}^{(b)}_{ij} = \begin{cases} \mathbf{I}_{p \times p}^{\nu_i \cdot \theta_j \bmod p}, & \nu_i = i \bmod p, \theta_j = \lfloor j/p \rfloor & t_{ij} \neq 0; \\ \mathbf{0}_{p \times p}, & & t_{ij} = 0. \end{cases}$$

for  $i, j = 0, 1, \dots, p-1$ .

**Theorem 2.** *The bipartite graph described by  $\mathbf{H}_0$  has girth  $\geq 8$ .*

*Proof.* Let  $b$  be a node of the graph described by  $\mathbf{H}_0$ . By abuse of notation, let  $b$  also be the index of the corresponding node in matrix  $\mathbf{H}_0$ .  $\mu(b)$  is the node in the graph described by the  $p^2 \times p^2$  matrix  $\mathbf{T}$ , from which  $b$  is formed by graph replication. Node  $b$  can be described by three quantities  $0 \leq x_b, y_b, z_b < p$ , where  $x_b = \lfloor b/p^2 \rfloor$ ,  $y_b = \lfloor b \bmod p^2 / p \rfloor$ , and  $z_b = b \bmod p$ .

We first show that the graph described by  $\mathbf{H}_0$  is 4-cycle free. Assume the graph has a length-4 cycle  $b_1 c_1 b_2 c_2$ , where  $b_i$ 's are bit-nodes and  $c_i$ 's are check-nodes. Two cases exist. **Case 1:** If  $\mu(b_1) \neq \mu(b_2)$  and  $\mu(c_1) \neq \mu(c_2)$ , then a length-4 cycle  $\mu(b_1)\mu(c_1)\mu(b_2)\mu(c_2)$  exists in the graph described by  $\mathbf{T}$ , which is impossible by the construction of  $\mathbf{T}$ . **Case 2:** If  $\mu(b_1) \equiv \mu(b_2)$ , then edges  $(b_1, c_1)$  and  $(b_2, c_1)$  result from one edge  $(\mu(b_1), \mu(c_1))$  in the graph described by  $\mathbf{T}$ , which is impossible by the replication method.



We next prove that the graph described by  $\mathbf{H}_0$  is 6-cycle free. Assume the graph has a length-6 cycle  $b_1c_1b_2c_2b_3c_3$ . By construction of  $T$ , we have  $y_{b_2} - y_{b_1} = (x_{b_2} - x_{b_1}) \cdot x_{c_1} \bmod p$ ,  $y_{b_3} - y_{b_2} = (x_{b_3} - x_{b_2}) \cdot x_{c_2} \bmod p$ , and  $y_{b_1} - y_{b_3} = (x_{b_1} - x_{b_3}) \cdot x_{c_3} \bmod p$ . Therefore,  $(x_{b_2} - x_{b_1}) \cdot x_{c_1} + (x_{b_3} - x_{b_2}) \cdot x_{c_2} + (x_{b_1} - x_{b_3}) \cdot x_{c_3} = 0 \bmod p$ , or equivalently

$$x_{b_1} \cdot (x_{c_3} - x_{c_1}) + x_{b_2} \cdot (x_{c_1} - x_{c_2}) + x_{b_3} \cdot (x_{c_2} - x_{c_3}) = 0 \bmod p. \quad (5.1)$$

By the replication method, we have  $z_{b_2} - z_{b_1} = (x_{b_2} - x_{b_1}) \cdot y_{c_1} \bmod p$ ,  $z_{b_3} - z_{b_2} = (x_{b_3} - x_{b_2}) \cdot y_{c_2} \bmod p$ , and  $z_{b_1} - z_{b_3} = (x_{b_1} - x_{b_3}) \cdot y_{c_3} \bmod p$ . Therefore,  $(x_{b_2} - x_{b_1}) \cdot y_{c_1} + (x_{b_3} - x_{b_2}) \cdot y_{c_2} + (x_{b_1} - x_{b_3}) \cdot y_{c_3} = 0 \bmod p$ , or equivalently

$$x_{b_1} \cdot (y_{c_3} - y_{c_1}) + x_{b_2} \cdot (y_{c_1} - y_{c_2}) + x_{b_3} \cdot (y_{c_2} - y_{c_3}) = 0 \bmod p. \quad (5.2)$$

Similarly, by the construction of  $T$ ,  $y_{c_1} = y_{b_1} - x_{b_1} \cdot x_{c_1} = y_{b_2} - x_{b_2} \cdot x_{c_1} \bmod p$ ,  $y_{c_2} = y_{b_2} - x_{b_2} \cdot x_{c_2} = y_{b_3} - x_{b_3} \cdot x_{c_2} \bmod p$ , and  $y_{c_3} = y_{b_3} - x_{b_3} \cdot x_{c_3} = y_{b_1} - x_{b_1} \cdot x_{c_3} \bmod p$ . Substituting in (5.2) yields

$$x_{b_1}^2 \cdot (x_{c_3} - x_{c_1}) + x_{b_2}^2 \cdot (x_{c_1} - x_{c_2}) + x_{b_3}^2 \cdot (x_{c_2} - x_{c_3}) = 0 \bmod p. \quad (5.3)$$

Solving (5.1) and (5.3) gives  $x_{c_3} = x_{c_1}$  or  $x_{c_2} = x_{c_1}$  or  $x_{b_1} = x_{b_2}$ , which are impossible by construction of  $\mathbf{H}_0$ .  $\square$

The  $p$ -replication method has many desirable features. The graph described by  $\mathbf{H}_0$ , and, therefore, the LT graph has girth 8. Besides, the two-stage replication makes it possible to develop a row-merging procedure in 5.5, corresponding to the  $p$ -replication method, that results in a 4-cycle free Raptor graph. Due to the simple description of the code construction, the logic needed to generate the bit-node memory addresses is very simple.

The two-stage replication in the source matrix construction has an additional advantage: a partially-parallel architecture can be obtained by processing  $p$  rows of  $\mathbf{H}_0$  or, equivalently, 1 row of  $\mathbf{T}$  per clock cycle. The banks of the bit-node memory can be organized efficiently so that  $p$  messages are accessed per bank, every clock cycle. Each memory bank is composed into  $p \times p$  partition, where one row of the partition stores the posterior LLRs of the  $p$  bit-nodes that result from the replication of one bit-node in the graph described by  $\mathbf{T}$ . One row of the partition is accessed per cycle, and shifted  $\bmod p$  accordingly.

The  $p$ -replication method has a main drawback: for a certain decoder architecture, the range of values of the LT input frame-length,  $K_{LT}$ , supported by this architecture is relatively small and not regularly spaced. As a prelude, it should be first noted that  $K_{LT}$  can be set to  $M \cdot p^2$ ,  $M \leq p$ , for example by omitting the first  $p^3 - M \cdot p^2$  columns of the  $p^3 \times p^3$  matrix  $\mathbf{H}_0$ . For a serial decoder architecture,  $M$  is a fixed parameter and, therefore, varying the value of  $K_{LT}$  is only possible by varying

$p$ . However, the set of possible values of  $K_{LT}/M$ ,  $\{49, 121, 169, 289, 361, 529, \dots\}$ , is relatively small and does not contain regularly spaced values. Besides, the organization of the bit-node memory in partially parallel architectures has to be designed for varying  $p$  which reduces its efficiency.

### 5.3.2 Product-group Replication

One of the goals of the architecture-aware code construction is to achieve the *multi-length decoding capability*, that is to generate a family of Raptor codes that are defined over equally spaced values of  $K_{LT}$  and are supported by the same hardware decoder architecture (5.1). As already seen, the  $p$ -replication method does not achieve this goal. The approach of the product-group replication is to restrict the entries of the base matrix  $\mathbf{B}$  (see 5.2) to some well-chosen subgroup  $(G, \cdot)$  of the *group of possible permutations operating on a set of size  $Q$* . In the discussion below,  $(G, \cdot)$  will be defined, and it will be clarified how such definition helps achieve the *multi-length decoding capability*. It should be noted that unlike the  $p$ -based replication method, the product-group replication does not specify the values of the entries of  $\mathbf{B}$ , but rather the group  $(G, \cdot)$  to which they should belong. The method can then be described as:

**1:** Define a non-abelian group  $(G, \cdot)$  as the direct product of the dihedral group  $D_n$  and the order- $pq$  group  $C_{pq}$  as follows:

$$\begin{aligned} G &\triangleq D_n \times C_{pq} = \{(x^i \cdot y^j, z^g \cdot w^h) \mid x, y \in D_n, z, w \in C_{pq}\}, \\ D_n &= \langle x, y \mid x^2 = 1, y^n = 1, x \cdot y = y^{-1} \cdot x \rangle, \\ C_{pq} &= \langle z, w \mid z^p = 1, w^q = 1, w \cdot z = z^2 \cdot w \rangle. \end{aligned}$$

where  $n, p, q \in \mathbb{N}$ . The group  $D_n$  has order  $|D_n| = 2n$ ,  $C_{pq}$  has order  $|C_{pq}| = pq$ , and  $G$  has order  $|G| = 2n \times pq$ . For the above relations to generate a group, the parameters  $n, p, q$  should satisfy the following: (i)  $p$  prime and  $q = p - 1$ ; or  $p = 3$  and  $q = 4$ , and (ii)  $n > 1$ . It is noteworthy that the elements of  $G$  can index any set of size  $2npq$ .

**2:** Construct the  $P \times M$  base matrix  $\mathbf{B} = [b_{ij}]$ , such that  $b_{ij} \in G$ .

**3: (Replication)** Form matrix  $\mathbf{H}_0$  by replacing each entry  $b_{ij}$  by a  $2npq \times 2npq$  permutation submatrix; for each row  $\rho$  of index  $\eta \in G$  in the submatrix, the entry  $\eta \cdot b_{ij}$  is set to 1.

For the graph described by  $\mathbf{H}_0$  to be 4-cycle free, the following inequality must be satisfied by the entries of  $\mathbf{B}$ :

$$b_{rj} \cdot b_{sj}^{-1} \neq b_{rj'} \cdot b_{sj'}^{-1}, \forall j \neq j', \forall r, s \geq 1.$$

In this replication method,  $K_{LT} \triangleq Q \cdot M \triangleq 2npqM$ . For a single family of Raptor codes, the triplet  $(p, q, M)$  is kept fixed while  $K_{LT} = 2npqM$  varies in steps

of  $2pqM$ , by incrementing/decrementing  $n$ . The reason of this is clarified when discussing the architectural implications of the construction scheme. In general, the construction of  $G$  is motivated by several considerations.

1.  $G$  is constructed to be non-abelian, because otherwise if  $G$  is abelian and the precode has parity matrix  $\mathbf{H}_0^{(P)}$ , there will be a large number of weight- $2C$  precode codewords. These codewords will thus have low weights because, in the typical case, the row merging factor  $C$  is relatively low ( $\sim 3, 4$ ). An example weight- $2C$  codeword can be found as follows: randomly pick  $1 \leq j \neq j' \leq M, x \in G$ ; consider the  $2npq$  bits corresponding to the replication of column  $j$  of  $B$ , and, among them, set exclusively the bits  $x \cdot b_{ij}^{-1}$  to 1,  $\forall 1 \leq i \leq C$ . Similarly, consider the  $2npq$  bits corresponding to the replication of the column  $j'$  of  $B$ , and, among them, set exclusively the bits  $x \cdot b_{ij'}^{-1}$  to 1,  $\forall 1 \leq i \leq C$ . It can be checked that the resulting binary vector is a weight- $2C$  precode codeword. This means that the row merging step producing  $\mathbf{H}_0^{(P)}$  results in a poor precode performance when  $G$  is abelian. Row merging is a crucial step in the construction framework, therefore,  $G$  is constructed to be non-abelian.
2. Choosing the dihedral group as a subgroup of  $G$  which order controls  $|G|$  is motivated by the fact that the dihedral groups of  $\{D_n\}_{n>1}$  have equally spaced orders, thus leading to equally spaced values of  $K_{LT} = 2npqM$ .
3. *Architectural Implications:* The third consideration is the simplification brought by the structure of  $(G, \cdot)$  to the organization and access of the bit-node memory in the decoder. This is best, though not solely, manifested in the case of partially-parallel architectures, where  $pq$  rows of  $\mathbf{H}_0$  are processed in one clock cycle. The memory is partitioned into  $M$  banks, each storing the posterior LLRs of  $2npq$  consecutive bits nodes. For clarity, consider the access of messages corresponding to the submatrix of  $\mathbf{H}_0$  formed by replicating entry  $b_{ji} = (x^b \cdot y^{b'}, z^t \cdot w^u)$ . Each cycle, a distinct coset of  $C_{pq}$  in  $G$  is chosen, call it  $C_{pq} \cdot x, x \in G$ , and the  $pq$ -rows indexed by elements of the coset are processed. The messages to be read are then the posterior LLRs of the bit nodes indexed by elements of the coset  $C_{pq} \cdot (x \cdot b_{ji})$ . Then memory bank  $i$  can be organized into a  $2n \times pq$  partition, where each row of the bank stores the LLRs of the bit nodes indexed by a coset of  $C_{pq}$  in  $G$ . Since  $C_{pq}$  is a fixed-order subgroup of  $G$ , regardless of the order of  $G$ , the size of a memory row of bank  $i$  is then fixed for a Raptor family, while its count varies with  $n$ , a major simplification to the design of efficient memory implementation. Thereafter, memory access can be divided into two steps: address generation of the memory row and permutation of the  $pq$  messages read from the row. The product nature of  $G$  simplifies these two steps as explained next. Any

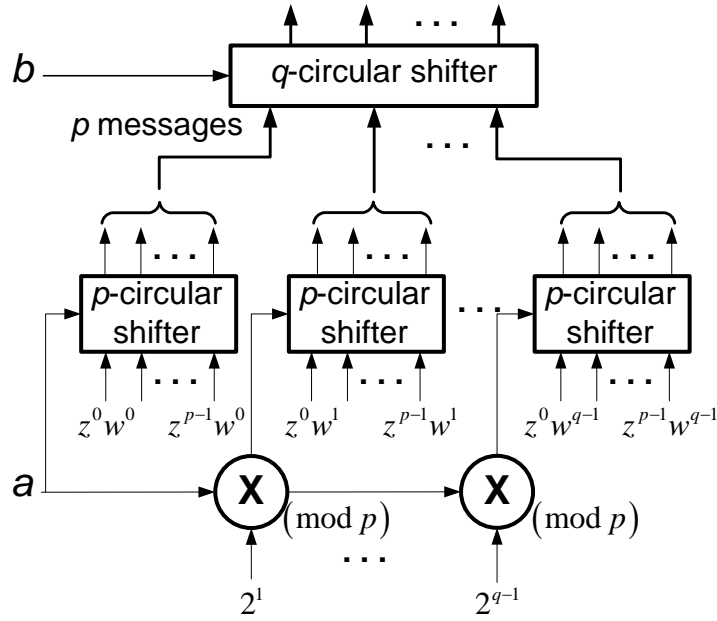


Figure 5.2: A  $pq$ -permutation induced by  $z^a w^b = (z^t \cdot w^u)^{-1}$ . The input message of a bit-node indexed by  $z^i \cdot w^j$ , is denoted as  $z^i \cdot w^j$  for simplicity.

coset of  $C_{pq}$  in  $G$  can be written as  $C_{pq} \cdot (x^a \cdot y^{a'}, z^0 \cdot w^0)$ . When the rows indexed by such coset are processed, the LLRs read are for the bit nodes indexed by the elements of coset  $C_{pq} \cdot (x^a \cdot y^{a'}, z^0 \cdot w^0) \cdot (x^b \cdot y^{b'}, z^t \cdot w^u)$ , which is a permutation by  $z^t \cdot w^u$  of the coset  $C_{pq} \cdot (x^{a+b} \cdot y^{(-1)^b \cdot a' + b'}, z^0 \cdot w^0)$ . The first step of memory access, the memory row address generation, is then reduced into two simple dihedral-related computations  $a + b \pmod{2}$  and  $(-1)^b \cdot a' + b' \pmod{n}$ . This can be done by updating the address values obtained in the previous clock cycle, thus requiring hardware comparable in complexity to a simple incrementor mod  $n$ . The second step is then done using a  $C_{pq}$ -dependent permutation network shown in Fig. 5.2. The network directs the LLRs read whose  $C_{pq}$ -subindex is  $z^{t'} \cdot w^{u'}$  to the processor of the matrix row whose subindex is  $z^{t'} \cdot w^{u'} \cdot (z^t \cdot w^u)^{-1}$ . Since  $C_{pq}$  is fixed, the same permutation network can be used for all possible values of  $K_{LT}$ , reducing substantially the overhead of the *multi-length decoding capability* on the interconnect network.

## 5.4 LT Code Construction

A simple way for constructing the subcode corresponding to a row  $\mathbf{V}$  of  $\mathbf{H}_0^{(L)}$ , is to form the generator matrix of the subcode by splitting the row  $\mathbf{V}$ , as explained next:

**Procedure:**

1. Design a LT check degree distribution  $\mathcal{D}$ , with maximum degree of  $M$ .
2. Starting from distribution  $\mathcal{D}$ , form another distribution  $\mathcal{D}'$  over the space of sets  $\mathcal{S}$ ; The space of sets  $\mathcal{S}$  is characterized by the following: each set in  $\mathcal{S}$  is a set of positive integers that sum to  $M$ .
3. For every row  $\mathbf{V}$  of  $\mathbf{H}_0^{(L)}$ :
  - . Random sampling from  $\mathcal{S}$  is done according to  $\mathcal{D}'$ . Denote the chosen set by  $T = \{T_1, \dots, T_t\}$ ,  $T \in \mathcal{S}$  and  $t = |T|$ .  $T$  describes the *split-pattern* of row  $\mathbf{V}$ .
  - . *Row-Splitting*: The row vector  $V$  is split into  $t$  rows,  $\mathbf{V}'_i$ ,  $i = 1, \dots, t$ , such that:  $\mathbf{V} = \mathbf{V}'_1 \oplus \dots \oplus \mathbf{V}'_t$ , and the Hamming weight of  $\mathbf{V}'_i$  is equal to  $T_i$ . This means that if the  $i$ th entry of  $\mathbf{V}$  is equal to 1, there exists exactly 1 vector  $\mathbf{V}'_i$ ,  $1 \leq i \leq t$ , that has its  $i$ th entry equal to 1. Within these constraints, the partitioning of the indices which have 1-entry in  $\mathbf{V}$  is made random by randomly permuting these indices prior to applying splitting. This is illustrated in Fig. 5.3. The generator matrix of the subcode corresponding to row  $\mathbf{V}$  is thus the  $t \times (M \cdot Q)$  matrix formed from the concatenation of the vectors  $\mathbf{V}'_i$ ,  $i = 1, \dots, t$ .

Step 1 is crucial in the design of the LT code. It is however, not considered in the research work presented in this dissertation.

One way to perform Step 2 is to minimize the weighted quadratic distance between  $\mathcal{D}$  and the check degree distribution resulting from  $\mathcal{D}'$ . The minimization is discussed next. Index each set in the space  $\mathcal{S}$ , and denote the set of index  $j$  as  $S_j$ ,  $j = 1, \dots, |\mathcal{S}|$ . Let  $\mathbf{A} = [a_{ij}]$  be a  $M \times |\mathcal{S}|$  matrix such that  $a_{ij}$  is the number of elements of value  $i$  in  $S_j$  divided by  $|S_j|$ . Then, the check degree distribution resulting from  $\mathcal{D}'$  is  $\mathbf{A} \cdot \mathcal{D}'$ . Finding the  $|\mathcal{S}|$ -vector  $\mathcal{D}' \triangleq [d'_i]$  can then be formulated as the following quadratic optimization problem:

$$\begin{aligned} & \underset{\mathcal{D}'}{\text{minimize}} && (\mathbf{A} \cdot \mathcal{D}' - \mathcal{D})^T \cdot \mathbf{W} \cdot (\mathbf{A} \cdot \mathcal{D}' - \mathcal{D}), \\ & \text{subject to} && 0 \leq d'_i \leq 1, i = 1, \dots, |\mathcal{S}| \quad \text{and} \quad \sum_{i=1}^{|\mathcal{S}|} d'_i = 1. \end{aligned}$$

where  $\mathbf{W} = [w_{ij}]$  is a diagonal matrix of weights. Steps 1 and 2 can involve an iterative co-design of  $\mathcal{D}$  and  $\mathcal{D}'$  to reach the best approximation, while having the quantization of distribution  $\mathcal{D}'$  entries in accordance with the number of rows of  $\mathbf{H}_0^{(L)}$ .

One possible simple way to do the sampling of Step 3 in practice is to read sequentially a circular array of split-pattern sets, denoted here the *Partition Set*. These sets are chosen to approximate the distribution  $\mathcal{D}'$ , and thus can be easily reconfigured in hardware for arbitrary  $\mathcal{D}'$ . This is illustrated in the row-splitting example in Fig. 5.3.

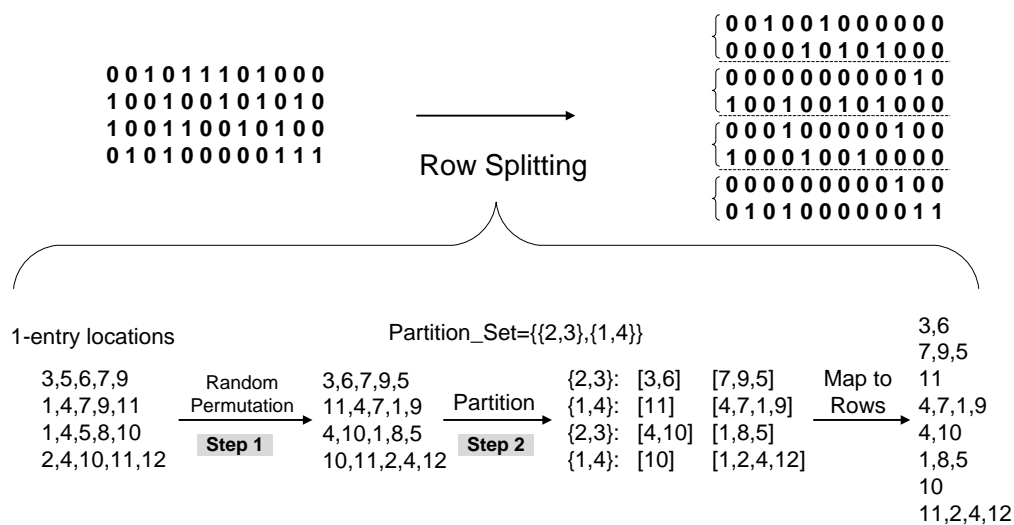


Figure 5.3: Random splitting of a row is done in two steps: 1) Random permutation of the non-zero entries, and 2) sampling from the partition set to obtain the resulting row weights. For simplicity, sampling is performed here by reading the partition set in a circular fashion.

Row-splitting is implemented, in hardware, by randomly permuting the  $M$  bit-to-check messages then applying check-node processing by reconfiguring the CFU, having a constant throughput of  $M$ , according to the split-pattern set  $T$ .

The main advantage of row-splitting is that *hardware-efficient check-function units can be designed to process the formed subcodes (step 4 of the decoding schedule 5.2.2)*. In other words, the designed CFUs can process efficiently any subcode that is formed by row-splitting, for any set  $T$  describing the split-pattern of the corresponding row. The design of these CFUs will be considered next.

### 5.4.1 Reconfigurable CFU Design

The design of the reconfigurable CFUs presented in this section is based on the architectures of their fixed-degree counterparts. Three basic algorithms for the CFU operation exist, namely 1) the conventional algorithm implementing equations (3.1) and (3.2), 2) the BCJR-based algorithm [70, 83], and the Min-Sum algorithm [79, 81]. The latter two give a reduced complexity approximation of the check-update equations, and thus, need no LUTs to compute the  $\psi$  function. The Min-Sum algorithm, in particular, reduces the check memory requirements. However this comes at the expense of varying degradation in performance for the Min-Sum update, and an increase in the complexity of computational units and latency in the CFU for the BCJR-based update.

**Input:** For efficient design, the following is input to the reconfigurable CFU. As an illustrating example, consider the split-pattern set  $T$  is  $\{3, 2, 4, 1\}$ ; thus, the generator matrix is as follows:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

1. A  $M$ -LLR vector **BTC** consisting of the bit-to-check messages corresponding to the processed row of  $\mathbf{H}_0^{(L)}$ :  $BTC_i$  denotes indicates the message of index  $i$ . The **BTC** vector is ordered such that the messages corresponding to the same LT output bit are consecutive in the vector. This ordering is made possible by the permutation step of the decoding schedule (step 2).
2. A  $M$ -bit partition-vector that fully describes the split-pattern set as follows: the first bit of the vector is 0 and equality of bits  $i$  and  $i + 1$  of the vector indicates the messages  $i$  and  $i + 1$  belong to the same LT check-node. For the example split pattern set  $\{3, 2, 4, 1\}$ , the partition vector is 0001100001 (i.e.,  $3 \leftrightarrow 000, 2 \leftrightarrow 11, 4 \leftrightarrow 0000, 1 \leftrightarrow 1$ ). The Subcode-Index Generator (5.2.2) is therefore implemented as a partition table that stores a number, say  $r$ , of partition vectors. In the simplest case, choosing the split pattern set  $T$  can be done by simply accessing the partition table as a circular buffer. That is the  $i$ th row of  $\mathbf{H}_0^{(L)}$  is split according to the partition vector of index  $i \bmod r$  stored in the partition table.
3. A  $M$ -LLR intrinsic-value vector  $\Lambda'$  that includes the intrinsic channel LLR values of the LT output-bits formed from splitting the corresponding row.  $\Lambda'$  is formed such that: if bits  $i$  and  $i - 1$  of the partition vector differ, the  $i$ th value of  $\Lambda'$  must equal the intrinsic LLR of the check-node whose message index starts with  $i$ . Consider the aforementioned example of  $T = \{3, 2, 4, 1\}$ ,

and denote by  $\Lambda_1, \Lambda_2, \Lambda_3$  and  $\Lambda_4$  the intrinsic LLR values of the first, second, third and fourth LT output-bits of the subcode respectively. The resulting intrinsic-value vector  $\Lambda'$  is then:

$$[\Lambda_1 \ \Lambda_1 \ \Lambda_1 \ \Lambda_2 \ \Lambda_2 \ \Lambda_3 \ \Lambda_3 \ \Lambda_3 \ \Lambda_3 \ \Lambda_4]$$

The organization of the Intrinsic-Value Memory (IVM) and the IVM network (see Fig. 5.1) is considered in light of the specification of  $\Lambda'$ . The IVM stores the intrinsic channel LLRs of the check-nodes. The number of LLRs read from the IVM varies per cycle depending on the split-pattern of the corresponding row of  $\mathbf{H}_0^{(L)}$ , but is at most  $M$ . The IVM block is, therefore, divided into  $M$  banks. The intrinsic LLR of the  $i$ th check-node is written at location  $\lfloor \frac{i}{M} \rfloor$  in bank  $i \bmod M$ . The IVM network, shown in Fig. 5.4, can be thought as a “decompressor” network composed of a  $M$ -wide cyclic shifter, to apply shifting  $\bmod M$ , followed by a multi-stage multiplexer network composed of  $\frac{M \cdot (M-1)}{2}$  [2:1] multiplexers to obtain the correct distribution of the intrinsic LLR values over  $M$  locations. The operation of the IVM network is illustrated assuming the aforementioned split-pattern example  $T = \{3, 2, 4, 1\}$ . For sake of illustration assume that the four generated LT-output bits have the indices 12, 13, 14, and 15 respectively. Then the intrinsic LLR values  $\Lambda_1, \Lambda_2, \Lambda_3$ , and  $\Lambda_4$  are read from blocks 2, 3, 4 and 5 respectively. The output of the IVM memory is  $M$ -wide and is as such (the  $x$  is a don't care):

$$[x \ x \ \Lambda_1 \ \Lambda_2 \ \Lambda_3 \ \Lambda_4 \ x \ x \ x \ x]$$

This  $M$ -word vector is shifted using the cyclic shifter of the IVM network to become:

$$[\Lambda_1 \ \Lambda_2 \ \Lambda_3 \ \Lambda_4 \ x \ x \ x \ x \ x \ x]$$

This vector is input to the multi-stage multiplexer network; where the control signal to the multiplexers in stage  $i$  of the network is the XOR of bits  $i$  and  $i - 1$  of the partition vector. The operation of the network is illustrated by showing the output of the successive  $M - 1$  stages of the network.

$$\begin{bmatrix} \Lambda_1 & \Lambda_1 & \Lambda_2 & \Lambda_3 & \Lambda_4 & x & x & x & x & x \\ \Lambda_1 & \Lambda_1 & \Lambda_2 & \Lambda_3 & \Lambda_4 & x & x & x & x & x \\ \Lambda_1 & \Lambda_1 & \Lambda_2 & \Lambda_2 & \Lambda_3 & \Lambda_4 & x & x & x & x \\ \Lambda_1 & \Lambda_1 & \Lambda_2 & \Lambda_2 & \Lambda_2 & \Lambda_3 & \Lambda_4 & x & x & x \\ \Lambda_1 & \Lambda_1 & \Lambda_2 & \Lambda_2 & \Lambda_2 & \Lambda_3 & \Lambda_3 & \Lambda_4 & x & x \\ \Lambda_1 & \Lambda_1 & \Lambda_2 & \Lambda_2 & \Lambda_2 & \Lambda_3 & \Lambda_3 & \Lambda_3 & \Lambda_4 & x \\ \Lambda_1 & \Lambda_1 & \Lambda_2 & \Lambda_2 & \Lambda_2 & \Lambda_3 & \Lambda_3 & \Lambda_3 & \Lambda_3 & \Lambda_4 \\ \Lambda_1 & \Lambda_1 & \Lambda_2 & \Lambda_2 & \Lambda_2 & \Lambda_3 & \Lambda_3 & \Lambda_3 & \Lambda_3 & \Lambda_4 \end{bmatrix}$$



**Output:** The output of the reconfigurable CFU is a  $M$ -LLR vector  $\mathbf{CTB}$  consisting of the check-to-bit messages corresponding to the processed row of  $\mathbf{H}_0^{(L)}$ .

The reconfigurable CFUs are also designed to process the LDPC precode check nodes when the matrix  $\mathbf{H}_0^{(P)}$  describes the precode parity-matrix. The processing of every degree- $(C \cdot M)$  check node is performed in  $C$  consecutive clock cycles. Then, the CFU outputs the check-to-bit messages corresponding to every row of  $\mathbf{H}_0$  involved in the row-merging operation that forms the corresponding check-node, in  $C$  consecutive cycles.

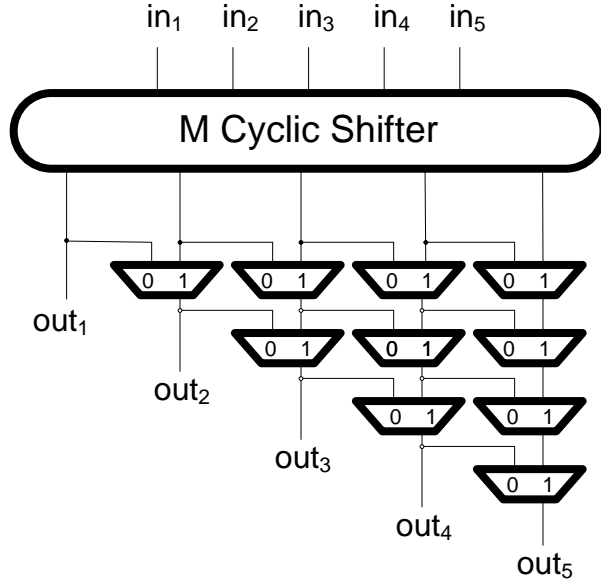


Figure 5.4: IVM Decompressor Network for  $M=5$ .

**Accumulator-Based CFU Architecture** The  $\psi$  function and its inverse in (3.1) and (3.2) can be implemented using look-up tables (LUTs). The  $\psi$  operation is also applied on the intrinsic LLR values only once and the generated values are stored in IVM. For simplicity of exposition, the signs of the  $BTC$  messages will be overlooked, and these messages will be treated as positive numbers throughout the following description. The proposed reconfigurable CFU is based on the design of a  $M$ -degree CFU that implements the following *transformed* equation:

$$CTB_j = \psi^{-1} \left( \sum_{j'=1}^M \psi(BTC_{j'}) - \psi(BTC_j) \right).$$

This equation can be implemented using a tree of  $M - 1$  adders of depth  $\lceil \log(M) \rceil$  to compute  $\sum \psi(BCT_{j'})$ , followed by  $M$  subtractors to extract the individual messages  $\psi(BTC_j)$  from the total sum. At level  $t$  of the adder tree, the  $M$  bit-to-check

messages are partitioned into  $q$  sets,  $P_{1,t}, \dots, P_{q,t}$ , such that 1) set  $P_{i,t}$ ,  $1 \leq i \leq q$ , includes messages with consecutive indices, 2) the sum of the  $\psi$  transformations of the messages in set  $P_{i,t}$  is computed in a stage  $\leq t$  (call it output value of the set), and 3) the output values of these  $q$  sets are fed to the adder sub-tree starting from stage  $t+1$  which, in turn, computes their sum. The left (right) boundary  $L_{i,t}$  ( $R_{i,t}$ ) of a set  $P_{i,t}$  is defined as the minimum (maximum) index of the messages in the set.

In the LT-decoding mode,  $M$  intermediate values,  $\sigma$ , are maintained at each level  $t$  of the adder tree. The  $i$ th value of  $\sigma$ ,  $\sigma(i)$ , stores the sum of the  $\psi$  transformations of messages that 1) belong to the same partition as message  $i$  at level  $t$ , and 2) have their corresponding LT-graph edges connected to the check node that the  $i$ th edge is connected to. The algorithm for updating  $\sigma$  is shown below, and the corresponding architecture is shown in Fig. 5.5.

---

**Algorithm 2** Accumulator-Based Reconfigurable CFU Algorithm

---

```

 $\sigma(i) \leftarrow \psi(BTC_i)$ ,  $i = 1, \dots, M$ 
 $P_{i,0} \leftarrow i$ ,  $i = 1, \dots, M$ 
for  $t = 1$  to  $\lceil \log(M) \rceil$  do
  for all  $a, b$  s.t. an adder in fixed-rate mode adds output values of  $P_{a,t-1}$  and  $P_{b,t-1}$  at stage  $t$  with  $a < b$  do
    if right boundary  $r_a$  of  $P_{a,t-1}$  and left boundary  $l_b$  of  $P_{b,t-1}$  correspond to same check-node then
       $\theta \leftarrow \sigma(r_a) + \sigma(l_b)$ 
    else
       $\theta \leftarrow \psi(\Lambda'(l_b)) + \sigma(l_b)$ 
    end if
     $\forall j \in P_{a,t-1}$ , if  $j, r_a, l_b$  share same check-node, then  $\sigma(j) \leftarrow \theta$ 
     $\forall j \in P_{b,t-1}$ , if  $j$  and  $l_b$  share same check-node, then  $\sigma(j) \leftarrow \theta$ 
  end for
end for
 $\psi(\mathbf{CTB}) \leftarrow \sigma - \psi(\mathbf{BTC})$  ▷ Vector subtraction

```

---

In LDPC decoding mode, the  $C \cdot M$  bit-to-check messages corresponding to one check-node are fed to the CFU in  $C$  consecutive cycles. An extra accumulator is used to add the  $C$  output values to output  $\sum_{\delta=1+\nu}^{C+\nu} (\sum_{i=1}^M \psi(BTC_i[\delta]))$ , where  $BTC_i[\delta]$  is the  $i$ th bit-to-check message in the  $BTC$  vector corresponding to the row of index  $\delta$  in  $\mathbf{H}'_0$ , and  $\nu + 1$  is the index of the first row in  $\mathbf{H}'_0$  corresponding to the current check-node. The last stage of subtraction in the CFU operation is delayed for an extra  $C$  cycles required to obtain the aforementioned sum. Consequently, The CFU attains a constant throughput of  $M$  messages/cycle.

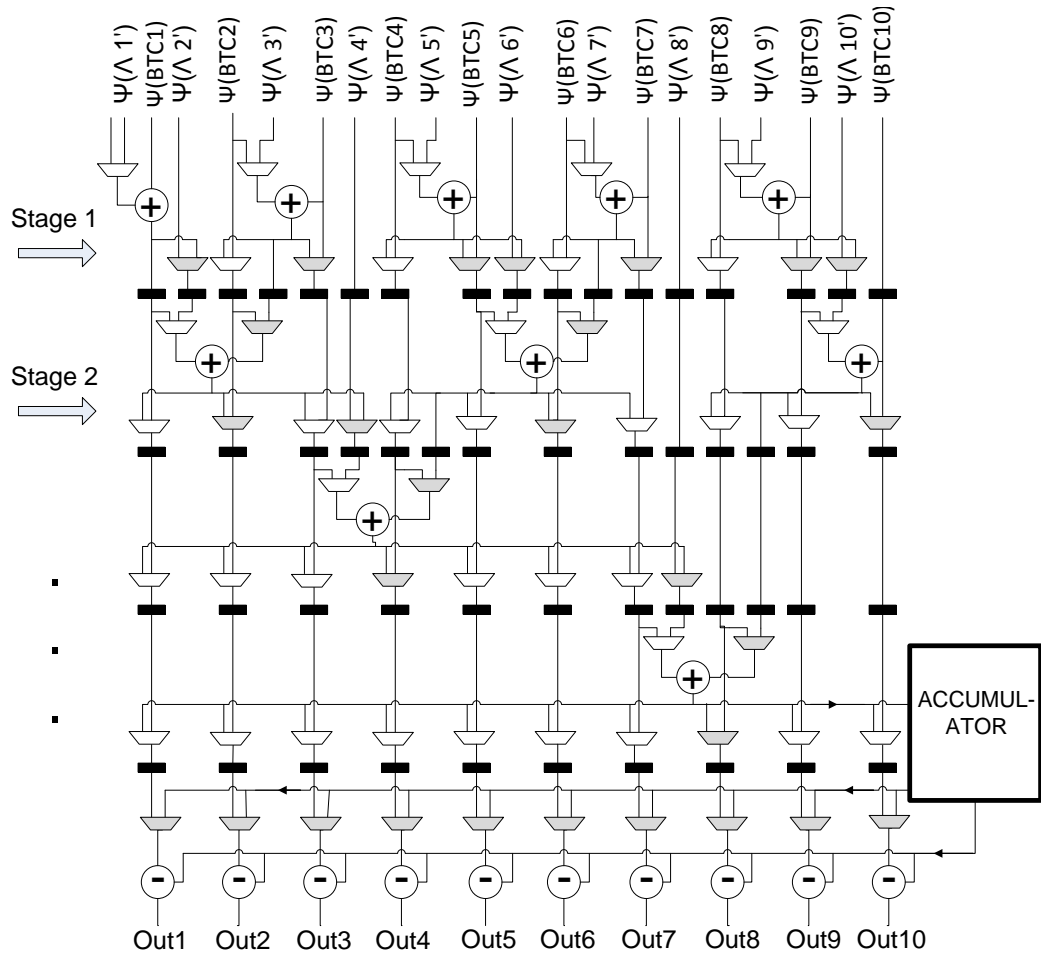


Figure 5.5: Accumulator-based CFU architecture. The grey-shaded muxes are needed for LDPC-decoding. Registers used to propagate *BTC* messages from the accumulator to the subtractor are omitted.

The registers for storing the intermediate vector  $\sigma$  during LT-processing are now re-used to store the vector  $\psi(\mathbf{BTC})$  for the  $C$  additional cycles. The extra latency of  $C$  cycles is due to the quasi-serial mode of CFU operation in LDPC decoding. This is unlike the LT-decoding latency caused by pipelining targeted to enhance the operating frequency.

**Forward-Backward BCJR-Based CFU Architecture:** In [70, 83], a SISO message-processing unit (MPU) was proposed to implement check-message processing. The check-to-bit messages are computed using a simplified form of the BCJR algorithm [89]. Let  $\otimes$  denote an operator which performs the operation  $x \otimes y = \ln(e^x + e^y) - \ln(1 + e^{x+y})$ . Then the message from a degree- $r$  check-node  $j$  to bit-node  $i$  at iteration  $\tau$ , where  $1 \leq i \leq r$ , is computed as [70]  $CTB_{ji}[\tau] = \bigotimes_{i' \neq i} BTC_{i'j}[\tau]$ . Moreover, the following simple yet fairly accurate approximation of  $\otimes$  was proposed [70]:

$$x \otimes y \approx \max(x, y) - \max(x + y, 0) + \max(5/8 - |x - y|/4, 0) - \max(5/8 - |x + y|/4, 0).$$

The unit implementing this approximation is called the Max-Quartet MPU. The resulting CFU implements the BCJR algorithm on the syndrome trellis of an  $(r, r-1)$ -SPC code, where intermediate forward and backward metrics are propagated each cycle. At stage  $1 \leq t < r$ , two metrics  $\alpha_t$  (forward) and  $\beta_t$  (backward) are computed according to the recursions  $\alpha_t \triangleq \bigotimes_{1 \leq i \leq t} BTC_i = \alpha_{t-1} \otimes BTC_t$  and  $\beta_t \triangleq \bigotimes_{0 \leq i \leq t-1} BTC_{r-i} = \beta_{t-1} \otimes BTC_{r+1-t}$ , respectively. Then starting from stage  $t > r/2$ , the check-to-bit messages  $CTB_i$  are generated as  $CTB_i = \alpha_{i-1} \otimes \beta_{r-i}$ .

A *reconfigurable* version of the SISO MPU is proposed in Fig. 5.6. LT-node processing capability is achieved by multiplexing the inputs to the Max-Quartet units implementing the  $\alpha$ - and  $\beta$ -recursions. The computation of the forward and backward metrics is done as follows:

$$\alpha_t = \begin{cases} \alpha_{t-1} \otimes BTC_t & \text{if messages } t \text{ and } t+1 \text{ correspond to the same check node;} \\ \Lambda'_{t+1}, & \text{otherwise.} \end{cases}$$

$$\beta_t = \begin{cases} \beta_{t-1} \otimes BTC_{M+1-t} & \text{if messages } M+1-t \text{ and } M-t \text{ correspond to same check node;} \\ -\infty & \text{otherwise.} \end{cases}$$

In LDPC-decoding mode, the metric  $F_\delta = \bigotimes_{i=1}^M BTC_i[\delta] = \alpha_{M/2} \otimes \beta_{M/2}$  corresponding to the row of index  $\delta$  in  $\mathbf{H}'_0$  (one of the rows merged to form the check-node  $j$ ) is computed and then forwarded to a degree- $C$  serial CFU. The CFU output  $\bigotimes_{0 \leq \delta' < c, \delta' \neq \delta} F_{\delta'}$  is then forwarded to  $M$  Max-Quartet units to compute  $\mathbf{CTB}[\delta]$ .

**Min-Sum CFU Architecture** A reduced complexity approximation of the check-to-bit message computation is given in [79] as:

$$CTB_i = \left( \prod_{i' \neq i} \text{sgn}(BTC_{i'}) \right) \times \min_{i' \neq i} |BTC_{i'}|.$$

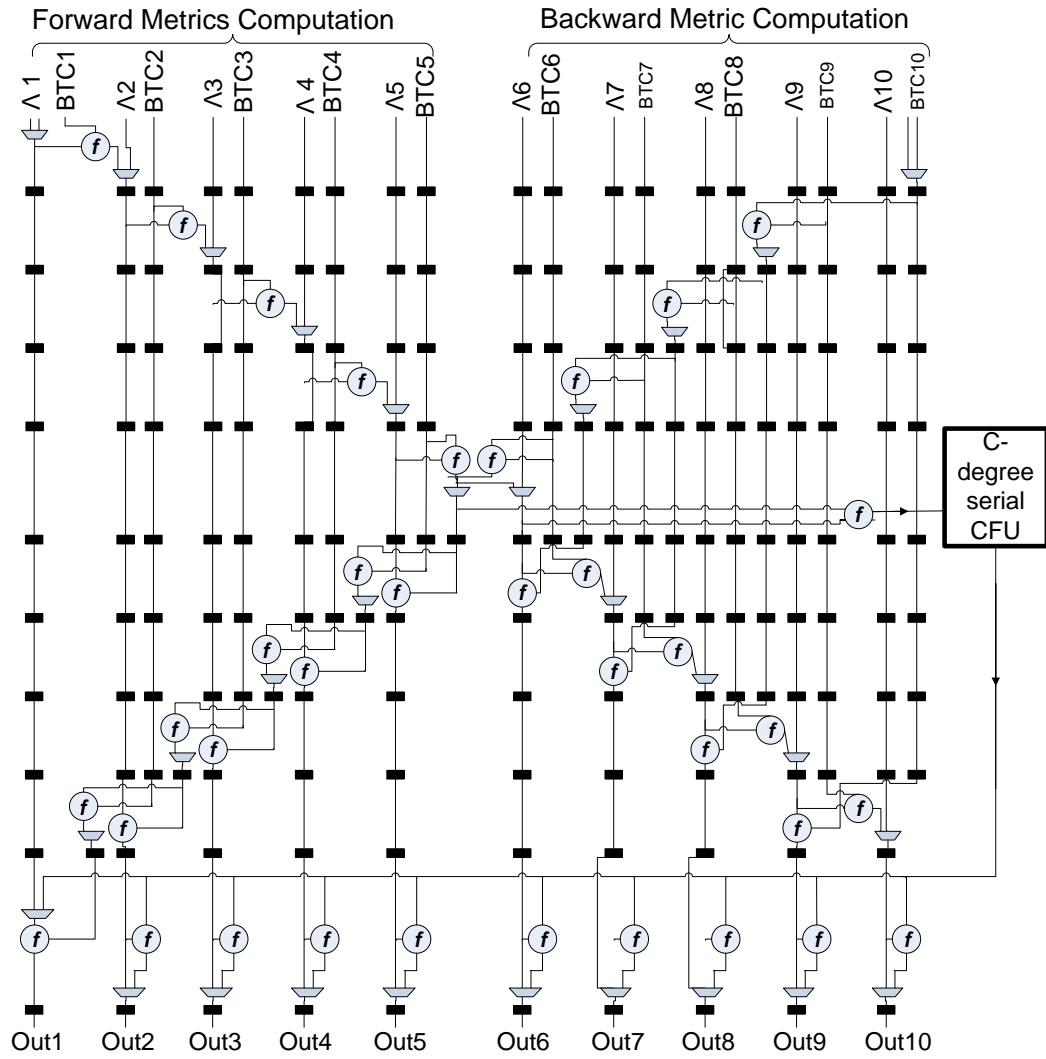


Figure 5.6: BCJR-based CFU. Blocks labeled  $f$  perform the Max-Quartet operation.

The Min-Sum approximation results in a degradation in the error correcting performance. To reduce the performance gap, a correction step, consisting of subtracting an offset or multiplying by a normalization factor [81], is applied on the resulting minimum value.

The reconfigurable CFU implementing the Min-Sum approximation is based on the constant-degree implementation given in [75]. The architecture has a tree structure similar to the accumulator-based CFU, where the addition operation is substituted by a 4-input 2-output partial-sorting function that computes the minimum and the second minimum of two sorted input pairs. The final output of the tree are the minimum-value, its index, and the second minimum value of the input values. During LT-decoding, two intermediate values instead of one are needed for every edge  $i$  per level. These are the minimum and second minimum values of the messages that 1) belong to the same partition as message  $i$  at level  $t$ , and 2) have their corresponding LT-graph edges connected to the check node that the  $i$ th edge is connected to. To reduce the number of intermediate values per edge to one, the following modifications are made. For a degree-1 LT-node edge  $i$ , the intermediate value is set to the corresponding intrinsic channel reliability value. For a degree-2 LT-node edge  $i$ , the intermediate value holds the corresponding minimum value if  $i$  is odd or the second minimum value if  $i$  is even. A maximum of  $M + 1$  extra intermediate values are needed throughout the CFU operation, used when the right boundary of a partition does not share its check-node with any other edge in the partition. The indices of the minimum values, corresponding to the row check nodes, are tracked across the CFU by keeping a  $M$ -bit minimum-index vector, which is updated per stage upon the results of the partial-sorting operations. In the final stage, the correction step is applied on the  $M$  intermediate values and the  $CTB$  messages are generated using the partition and minimum-index vectors. In LDPC-decoding mode, the minimum index-vector and two odd- and even-indexed outputs of the correction step are forwarded to an additional partial-sorter. The minimum, second minimum, and the minimum index over the  $C$  rows of every corresponding check node are thus computed with an extra latency of  $C$  cycles.

The Min-Sum algorithm changes the check-node memory requirements since up to three values are sufficient to regenerate the absolute values of the messages corresponding to one node. This results in significant savings in the LDPC memory. Three LT-memory organization schemes are considered: 1) the LT-memory retains its conventional organization, 2) the LT memory stores two values per node (the minimum and second minimum values) and a  $M$ -bit minimum-index vector per row, and 3) the LT-memory stores three values per LT-node. In the second scheme the LT-memory is decomposed into three blocks: two for storing the odd/even-indexed values output by the correction step, prior to the  $CTB$  generation, and

one for storing the minimum-index vector. The odd/even-indexed memory block is read/written at most once for every check node of the processed row, and thus has its access pattern similar to that of the IVM memory. Consequently, it has a similar organization, but contains  $\frac{M}{2}$ , instead of  $M$ , memory banks. A  $\frac{M}{2}$ -wide “compressor” is needed to store the CFU output in memory, its operation is the inverse of the previously discussed “decompressor” (see Fig. 5.4). Besides, a  $\frac{M}{2}$ -wide decompressor is needed to regenerate the corresponding  $\frac{M}{2}$  messages in the next iteration. Fig. 5.7 shows the CFU interconnect under this scheme. In the third scheme, additional interconnect and logic are needed to correctly store and extract the minimum indices. Figure 5.8 compares the reduction in the check-node memory size in the three respective schemes versus the average LT-node degree. As the average LT-node degree increases, the memory size reduction brought by the Min-Sum algorithm becomes more significant.

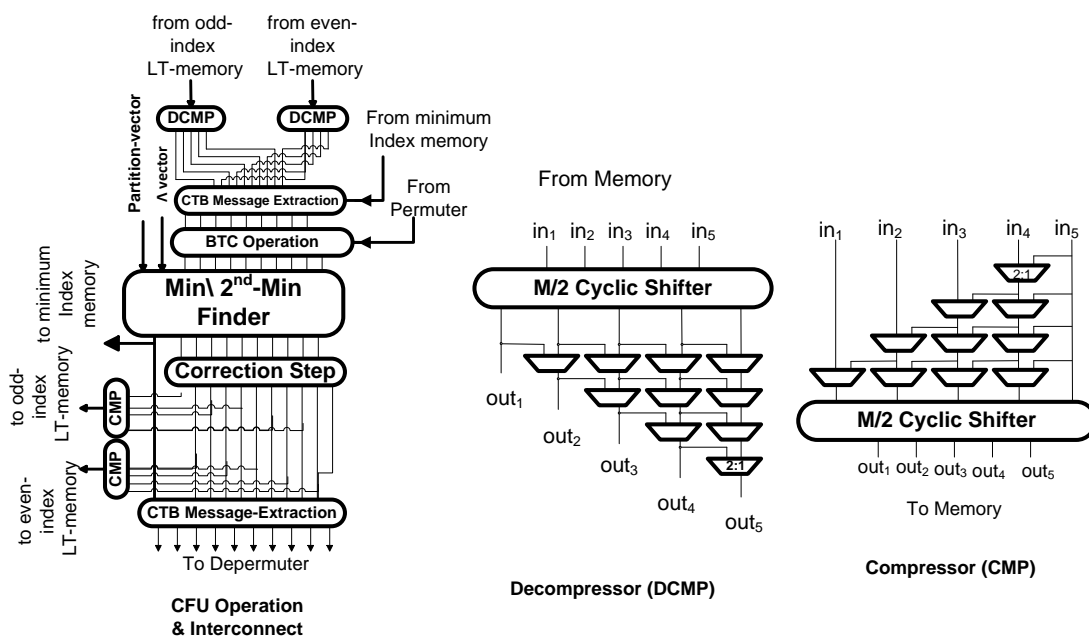


Figure 5.7: operation and interconnect of the Min-Sum implementation. Only LT-decoding mode is shown for clarity.

Table 5.1 compares the hardware complexity of the fixed-degree and reconfigurable implementations of the three algorithms. Reconfigurability in results in a replication of the number of registers. For the BCJR-based update the number of function units increases by  $\sim 4/3$ , while a large number of [2:1] muxes is needed for the other two.

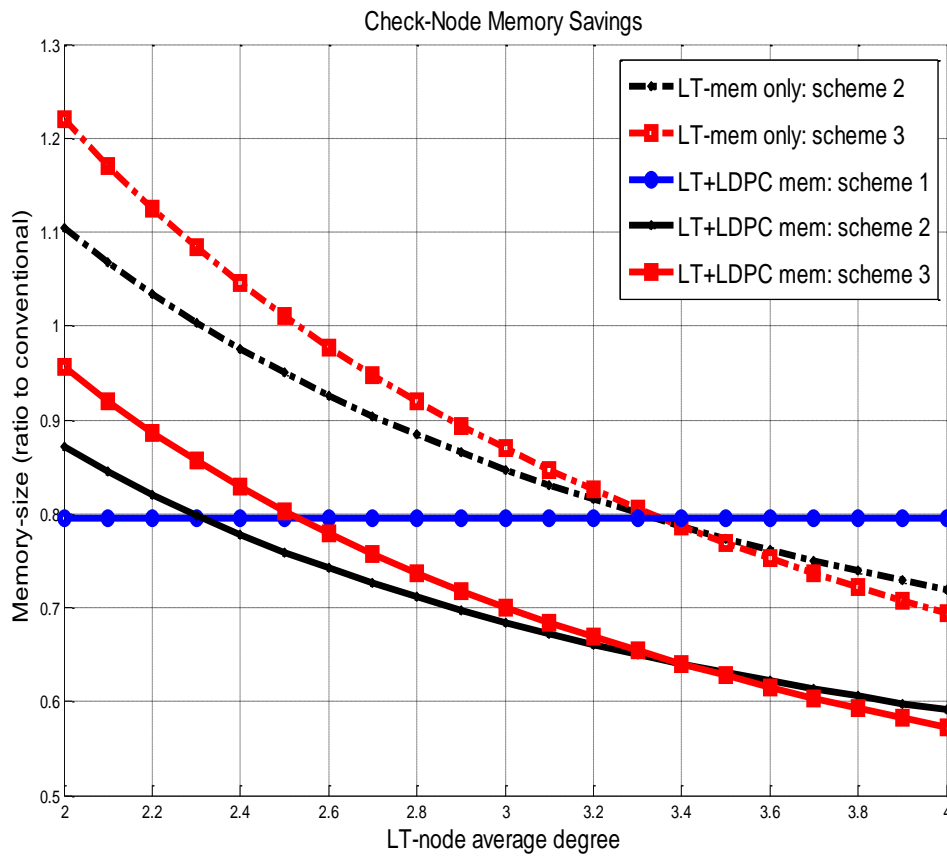


Figure 5.8: Check-node memory size of the 3 schemes in Min-Sum implementation,  $M = 10$ .



Table 5.1: Hardware resources of fixed-degree and reconfigurable architectures, of the three algorithms. The Min-Sum correction step and the sign-producing logic are not included.

Component	Accumulator-based		BCJR		Min-Sum	
	<i>Fixed</i>	<i>Reconfigurable</i>	<i>Fixed</i>	<i>Reconfigurable</i>	<i>Fixed</i>	<i>Reconfigurable</i>
LUTs	$2 \cdot M$	$2 \cdot M$	-	-	-	-
Adders	$2 \cdot M - 1$	$2 \cdot M + 1$	-	-	-	-
Max-Quartet Units	-	-	$3 \cdot M - 6$	$4 \cdot M - 4 + \lceil C/2 \rceil$	-	-
Comparators	-	-	-	-	$2 \cdot M - 3$	$2 \cdot (M + 1)$
Registers	$M \cdot \lceil \log(M) \rceil + O(M)$	$M \cdot \lceil \log(M) \rceil + O(M) \pm O(C) + M \cdot \max(\lceil \log(M) \rceil, C)$	$3 \cdot M^2/2 + O(1)$	$2 \cdot (M + 1)^2 + O(C)$	$O(M)$	$M \cdot \lceil \log(M) \rceil + O(M)$
[2:1] Muxes	-	$M \cdot \lceil \log(M) \rceil + O(M)$	-	$3 \cdot (M + 1) + O(C)$	$O(M)$	$M \cdot \lceil \log(M) \rceil + O(M)$

## 5.4.2 Structured Subcodes

When the value of the code-rate  $R = \frac{K}{N}$  is close to  $\frac{K}{K_{LT}}$ , the LT code rate  $\frac{K_{LT}}{N}$  approaches 1. Therefore, in pseudo-random LT bit generation, the number of LT check-nodes in the decoding graph may be insufficient to recover the bit values within a reasonable number of iterations, even in the absence of noise. As will be shown in 5.6, simulation results on LT-codes constructed using the row-splitting technique indicate that the highest-rate code obtained with comparable performance to LDPC had a rate  $2/3$ , while the average number of required iterations was  $3\times$  higher than in LDPC.

A possible way to approach this problem is to insert additional structure and/or constraints on the  $M$ -sized subcodes. As discussed next, the steps are targeted to 1) enhance performance at high code rates, 2) speed up convergence, and 3) reduce memory requirements.

Constraining/structuring LT encoding is, primarily targeted at constructing performance-efficient Raptor codes for rates close to the precode rate. Therefore, the choice of the subcode is now dependant on the index of the increment/frame-portion in which the encoding bits will be included. One possible subcode choice guide is that for high code-rates, LT encoding must be done such that: *in the absence of noise, every bit-node can be recovered by applying no more than a small number  $k$  of the LT-row operations described in [35]*. One way to do this is to constraint row-encoding of the first  $M$  rows of  $\mathbf{H}_0^{(L)}$  as follows: encode row  $\mathbf{V}$ , such that  $M$  check-nodes are generated from the corresponding  $M$  bits, and exclusive local decoding of the resulting subcode is sufficient to retrieve the values of these  $M$  bits, in the absence of noise. The subcode generator matrix can vary across rows. An example  $M \times M$  generator matrix of a subcode satisfying this property is: ( $M = 6$ )

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

As can be easily checked, processing the subcode once is sufficient to recover the corresponding  $M$  LT output bits in one iteration, in the absence of noise. This will also make the convergence speed of the decoding process higher. Two notes have to be made on such structuring. The first is the resemblance between the subcode structuring requirement and that in the puncturing methods discussed in 3.4.1; this is expected since both requirements are motivated by the aim to decrease

the number of decoding iterations required to recover all the bit-nodes in the absence of noise. The second note is that such structuring has a major problem: *it makes subcode generation strongly dependant on the portion of the frame in which the output bits of the subcode is included, therefore, the Raptor code may face the problem of performance degradation in the previously considered scenarios of intra-frame varying channel condition, similar to the case of conventional LDPC and Turbo codes.* An open question is on performing subcode structuring and pseudo-random row-splitting such as to achieve the desired tradeoff between the performance of the code when all the portions of the sent frame experience one SNR and the performance when different portions of the frame experience different SNR values. In the rest of this subsection the advantages and possible hardware overhead of structured subcodes are discussed.

*Memory Savings:* In message-passing algorithms where exact check-node equations are applied, the number of stored messages is equal to the graph's edge-count. The proposed row-encoding partially decouples the number of stored messages from the underlying edge count, thus reducing the required memory size as explained next. Local decoding of a subcode involves the following inputs/outputs: the inputs are  $M$  LLRs of the corresponding  $M$  bit-nodes and the intrinsic LLRs of the corresponding check-nodes; the outputs are  $M$  LLR values. The  $M$ -word output is stored in memory to be used in the subsequent iteration. However, since the column-weight in the generator matrix of a subcode can exceed 1, the number of edges in the subcode graph is  $> M$ .

*Convergence Speed:* Beside structuring, local subcode decoding increases the decoding convergence speed. This is illustrated in the following example of a LT code generator matrix ( $M = 6$ ):

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

In decoding, each check constraint produces one extrinsic LLR value for the third bit (call these values  $v_1$  and  $v_2$ , respectively). In TDMP,  $v_1$  is used in the processing of check 2. On the other hand,  $v_2$  will be used in check 1 processing in the next iteration. Conversely, having these two checks within 1 subcode implies that the processing of check-nodes 1 and 2 makes use of  $v_2$  and  $v_1$  respectively, within the same decoding iteration. This lowers the required number of decoding iterations. It should be noted here that the column-weight values in the subcode generator matrix is limited by the acyclic condition which is imposed on the subcode graph. This condition is merely motivated by the fact that the subcode local decoding and performance analysis are both complicated by the existence of cycles.

Compared to simple row-splitting, LT subcode structuring clearly increases the complexity of check-node processing in terms of routing, control, and operation

count. An accurate overhead estimation needs well-defined constraints on subcode structuring, and therefore, is outside the scope of the dissertation.

## 5.5 Precode Construction

Two points on precode construction are considered in this section. The first point is on how to apply row-merging such that the resulting Raptor graph is 4-cycle. The second point is on the generation of relatively low-rate precodes.

### 5.5.1 Row Merging Transformation

In general, the construction of a 4-cycle free Raptor graph is done by imposing appropriate constraints on the base matrix  $\mathbf{B} = [b_{ij}]$ , the violation of which implies 4-cycles exist in the resulting graph. These constraints can be stated as follows:

$$\begin{aligned}
 1 & : b_{rj} \cdot b_{sj}^{-1} \neq b_{rj'} \cdot b_{sj'}^{-1}, & \forall j \neq j', & \forall r, s \geq C + 1; \\
 2 & : b_{rj} \cdot b_{sj}^{-1} \neq b_{tj'} \cdot b_{uj'}^{-1}, & \forall 1 \leq j, j' \leq M, & \forall 1 \leq r, s, t, u \leq C, (r, s) \neq (t, u); \\
 3 & : b_{rj} \cdot b_{sj}^{-1} \neq b_{tj'} \cdot b_{sj'}^{-1}, & \forall 1 \leq j \neq j' \leq M, & \forall 1 \leq r, t \leq C, s > C;
 \end{aligned}$$

If constraint 1 is satisfied, the LT-graph is 4-cycle free; similarly, if constraint 2 is satisfied, the precode graph is 4-cycle free. Satisfying constraint 3 means that if two bit-nodes are neighbors to a check node in the LT-graph, there exists no check-node that is neighbor to these bit-nodes in the precode-graph. Overall, satisfying the three constraints imply the Raptor graph is 4-cycle free.

In the particular case where the source matrix is constructed according to the *p*-based replication method, row-merging is done by applying a specific row-merging procedure, illustrated in Fig. 5.9.

**Procedure:** Row Merging Transformation for *p*-based replication

**Step 1:** Form a set of integers  $\Gamma$  with maximum cardinality  $|\Gamma| = C$  such that:

- i.  $\forall t \in \Gamma, 0 \leq t \leq p - 1$ .
- ii.  $\forall t, x, y, z \in \Gamma$ , if  $t - x = y - z \neq 0 \pmod{p}$ , then the pair of elements  $(t, x)$  is identical to the pair  $(y, z)$ .

**Step 2:** For  $i = 1, \dots, p - 1$ :

- i. Form the  $p \times p$  matrix  $\mathbf{W}_i = [w_{jk}]$  by adding modulo 2 the  $C$  shifted identity matrices  $I_p^{i \cdot \Gamma_j \pmod{p}}$ , for  $1 \leq j \leq C$ , where  $\Gamma_j$  is the  $j$ th element of  $\Gamma$ .
- ii. Replace every 0-entry  $w_{jk}$  of  $\mathbf{W}_i$  with  $-1$ , and every 1-entry  $w_{jk}$  with the quantity  $i \cdot j \pmod{p}$ , for  $0 \leq j < p$ .

- iii. Form the  $p^2 \times p^2$  matrix  $\mathbf{F}_i = [f_{jk}]$  by replicating  $\mathbf{W}_i$  by a factor of  $p$ . Each entry  $w_{jk}$  is replaced by a  $I_p^{w_{jk}}$  if  $w_{jk} > -1$ , and by a  $p \times p$  zero matrix otherwise.

**Step 3:** Form the  $p^2 \times (p^3 - p^2)$  LDPC base-matrix  $\mathbf{H}_0^{(\mathbf{P})}$  by concatenating the  $p - 1$  matrices  $\mathbf{F}_i$ , for  $1 \leq i < p$ .

**Step 4:**  $\forall 1 \leq i \leq C$ , omit from  $\mathbf{H}_0$  the rows whose indices are  $p^2 \cdot \Gamma_i \leq i < p^2 \cdot (\Gamma_i + 1)$  to form a  $(p^2 \cdot (p - C)) \times p^3$  matrix. Then omit the first  $p^2$  columns; the resulting matrix is  $\mathbf{H}_0^{(\mathbf{L})}$ .

Step 2 of the procedure is a rigorous way to describe the row merging of  $C$   $p^2$ -row submatrices of  $\mathbf{H}_0$ , where the  $i$ th submatrix is the formed from the rows of indices between  $\Gamma_i \cdot p^2$  and  $(\Gamma_i + 1) \cdot p^2 - 1$  in  $\mathbf{H}_0$ .

**Theorem 3.** *The Raptor graph described by  $\mathbf{H}_R \triangleq [\mathbf{H}_0^{(\mathbf{L})}; \mathbf{H}_0^{(\mathbf{P})}]$ , is 4-cycle-free.*

*Proof.* The terminology used here is similar to that in the proof of Theorem 2. First, we prove the precode graph is 4-cycle free. Assume the graph described by matrix  $\mathbf{H}_0^{(\mathbf{P})}$ , has a length-4 cycle  $b_1c_1b_2c_2$ , where  $b_i$ 's are bit-nodes and  $c_i$ 's are check-nodes. Two cases exist. **Case 1:**  $x_{b_1} = x_{b_2}$  then  $\exists t, x, y, z \in \Gamma$  such that  $t - x = y - z \pmod p$  and  $(t, x) \neq (y, z)$ , which is impossible by condition (1.ii) of the row merging transformation. **Case 2:**  $x_{b_1} \neq x_{b_2}$ , then  $c_1$  is connected to  $b_1, b_2$  in the 4-cycle which implies that  $z_{b_1} - z_{b_2} = y_{c_1} \cdot (x_{b_1} - x_{b_2}) \pmod p$ . Similarly,  $c_2$  is connected to  $b_1, b_2$  in the 4-cycle, hence  $z_{b_1} - z_{b_2} = y_{c_2} \cdot (x_{b_1} - x_{b_2}) \pmod p$ . Therefore,  $y_{c_1} = y_{c_2}$ , which implies that  $c_1 \equiv c_2$  by construction.

Now, assume the graph described by matrix  $\mathbf{H}_R$ , has a length-4 cycle  $b_1c_1b_2c_2$  where  $b_i$ 's are bit-nodes and  $c_i$ 's are check-nodes. Three cases exist. **Case 1:** Both  $c_1$  and  $c_2$  are LT-check nodes. By Theorem 2, this case is impossible. **Case 2:** Both  $c_1, c_2$  are LDPC-check nodes, which is impossible by the first part of the proof. **Case 3:**  $c_1$  is an LT check-node and  $c_2$  is an LDPC check-node.  $c_1$  is connected to  $b_1, b_2$  in the 4-cycle implies that  $z_{b_1} - z_{b_2} = y_{c_1} \cdot (x_{b_1} - x_{b_2}) \pmod p$  and  $x_{b_1} \neq x_{b_2}$ . Also,  $c_2$  is connected to  $b_1, b_2$  implies that  $z_{b_1} - z_{b_2} = y_{c_2} \cdot (x_{b_1} - x_{b_2}) \pmod p$ . Therefore,  $y_{c_1} = y_{c_2}$ . Hence in the  $p^3 \times p^3$  matrix  $\mathbf{H}_0$  constructed by the  $p$ -based replication, bit-node  $b_1$ , where  $b_1 > p^2$  (i.e.  $b_1$  does not correspond to any of the first  $p^2$  columns), is connected to two check nodes having different  $x$ -coordinates but equal  $y$ -coordinates, which is impossible by the  $p$ -based replication method.  $\square$

## 5.5.2 Irregular Precodes

In the original Raptor code construction [34], the precode rate approaches 1 and therefore its rate overhead is negligible. The precode check-nodes are then assumed

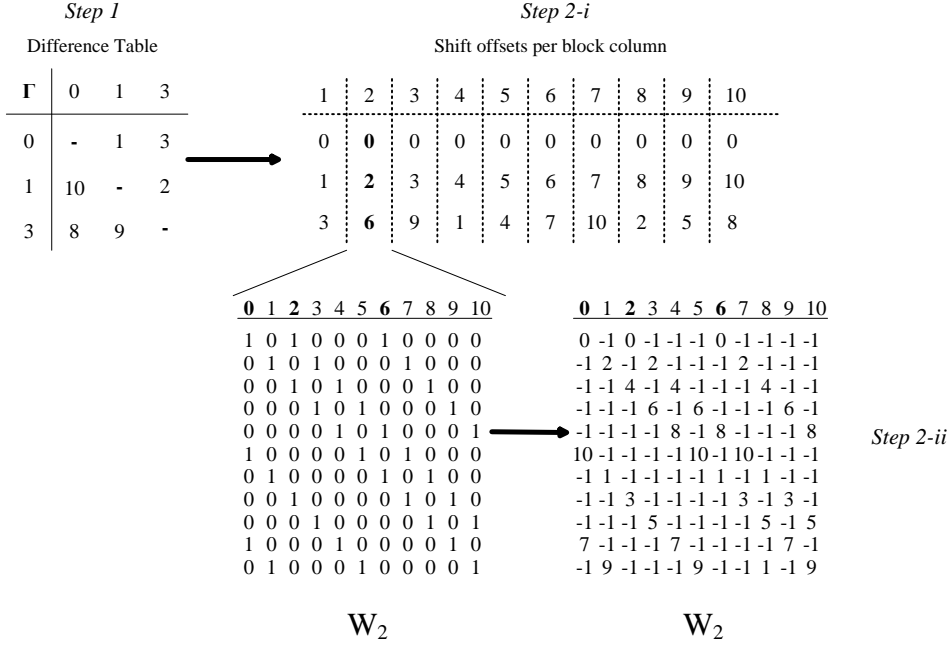


Figure 5.9: Row merging for  $p = 11$ ,  $C = 3$ .

to start decoding effectively when the mean of the exchanged messages becomes relatively high. For short block lengths (e.g.  $< 3K$ ), a high rate precode ( $\sim 1$ ) would lead to a high number of short cycles in the code graph and a relatively high error-floor [90], thus degrading performance. Relatively low-rate regular precodes, with rates  $> 0.9$ , were considered in [90,91], and the LT check-degree distribution was optimized using density evolution methods [91]. In lower rate precoding, with rates even lower than 0.9, regular LDPC precodes may not be optimal and a joint LT-LDPC degree optimization is needed. Consequently, two additional points on precode design are considered here. The first is to set a new criterion aiming at hardware efficiency beside performance, to determine the appropriate precode rate values. The second is to extend row-merging to generate irregular lower rate precodes starting from a regular  $\mathbf{H}_0^{(P)}$ , according to some input precode check-degree distribution. This extension, called splitting-after-merging, enables the construction of irregular precodes while preserving the underlying structural properties of  $\mathbf{H}_0$ , crucial for hardware-efficient row processing.

*Hardware-Constrained Rate Criterion:* The proposed criterion, heuristic in nature, seeks to maximize the minimum distance of the Raptor code under a hardware-oriented constraint. The criterion is needed to compare two possible precode rate values  $R_0$  and  $R'_0$ , and choose one of them. If the number of rate options considered is greater than 2, the comparison step can be applied iteratively, each time ruling out one option. Let  $d_0, d'_0$  be the minimum distances of

the precodes with rates  $R_0$  and  $R'_0$  respectively,  $d_v, d'_v$  the respective average LT bit-node degrees, and  $C$  the average precode bit-node degree, assumed fixed across both rates. The approach compares the minimum distances of the two resulting Raptor codes, which are approximated by  $d_0 \cdot d_v$  and  $d'_0 \cdot d'_v$  respectively. The two quantities  $d_v$  and  $d'_v$  are related using the following simplifying constraint: the number of edges in the two respective Raptor graphs must be equal, that is  $(d'_v + C)/R'_0 = (d_v + C)/R_0$ . This constraint is hardware-oriented since in virtually all decoding algorithms, the operation count and memory requirements are a strong function of the edge-count, and thus the constraint approximately fixes these hardware-related figures when comparing the rate values. Thus, the condition for choosing  $R'_0$  over  $R_0$  is having  $d'_0 \cdot d'_v \geq d_0 \cdot d_v$  which is equivalent to ( $r = \frac{R'_0}{R_0}$ ):

$$d'_0 \geq d_v \cdot (d_v \cdot r - C \cdot (1 - r))^{-1} \cdot d_0.$$

Since  $\frac{d_v}{d_v \cdot r - C \cdot (1 - r)}$  decreases with increasing  $d_v$ , it is sufficient to test the inequality condition for the highest code rate of interest, corresponding to the minimum  $d_v$  value.

*Row Splitting-after-Merging:* Following the choice of the precode rate  $K/K_{LT}$ , a corresponding precode check-degree distribution is determined, the procedure of which is out of the scope of the dissertation. Then, a  $(K_{LT} - K)$ -row parity matrix is generated from regular matrix  $\mathbf{H}_0^{(P)}$  through the row splitting-after-merging step: Split each row  $\mathbf{V}$  in  $\mathbf{H}_0^{(P)}$  into binary rows  $\mathbf{V}'_1, \mathbf{V}'_2, \dots$  similar to the splitting technique described in Section 5.4. Splitting is done according to a distribution on the space of split-patterns that is derived from the intended check-degree distribution, in a similar manner to LT row-splitting.

*Special Regular-to-Irregular Graph Transformation for  $p$ -based Replication:* In the particular case of  $p$ -based replication, irregular precodes can be constructed starting from the regular matrix  $\mathbf{H}_0^{(P)}$ ; the latter matrix formed according to the previously described row-merging transformation. The row-weights of the submatrices  $\mathbf{W}_i$ , obtained from Step (2.i) of the row merging method, can be changed via 1-entry substitutions across the matrix. This is done by picking an integer  $0 \leq d < p$  that satisfies the following property:  $\exists g \in \Gamma$  such that  $\forall x \in \Gamma \setminus \{g\}, y, z \in \Gamma \cup \{d\}$  with  $(d, x) \neq (y, z), d \neq x$  and  $y \neq z: d - x \neq y - z \pmod p$ . Then, in submatrix  $\mathbf{W}_i, 0 < i < p$ , the 1-entries corresponding to  $\mathbf{I}^{i \cdot g \pmod p}$  can be interchangeably replaced by those of  $\mathbf{I}^{i \cdot d \pmod p}$  in a column-per-column or row-per-row fashion. The result is an irregular 4-cycle-free LDPC graph. This procedure is illustrated in Fig. 5.10. By applying identical row shuffling in  $\mathbf{W}_i$  and in the  $p^2 \times p^2$  submatrices of matrix  $\mathbf{H}_0^{(L)}$  that share the same bit-nodes with  $\mathbf{W}_i$ , a 4-cycle-free Raptor graph with irregular LDPC graph is obtained. However, the resulting LT graph is not 6-cycle-free anymore.

From a hardware perspective, this regular-to-irregular transformation is simpler



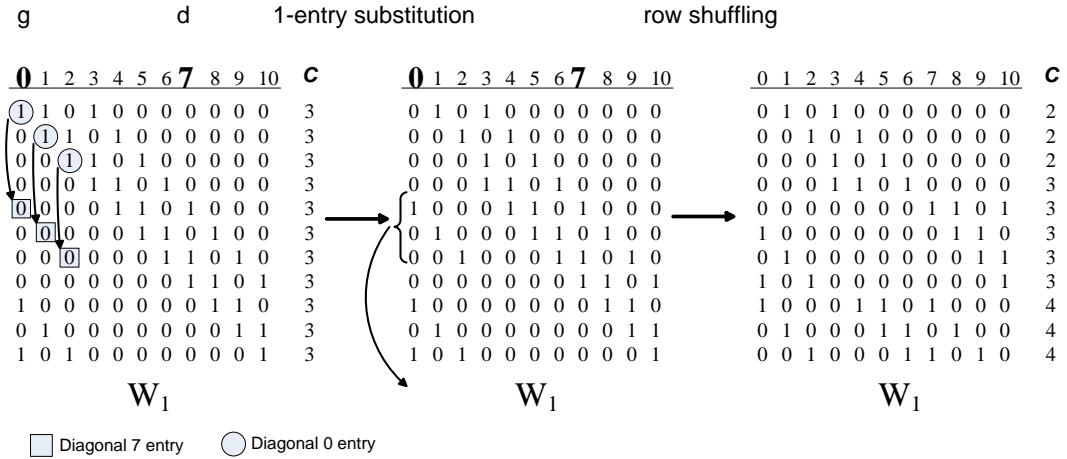


Figure 5.10: Regular-to-irregular graph transformation for  $\Gamma = \{0, 1, 3\}$ ,  $d = 7$  and  $g = 0$ . The non-zero entries corresponding to  $\mathbf{I}^0$  in the first three rows are replaced by non-zero entries corresponding to  $\mathbf{I}^7$ .  $C$  here denotes the vector of the corresponding row-weights. Note that the column weight remains unchanged.

than the splitting-after-merging technique, because the CFU will be processing 1 check node in a variable number of clock cycles. In the row-splitting-after-merging technique, the CFU will be processing the check nodes resulting from the splitting of the corresponding row in  $\mathbf{H}_0^{(p)}$ , over  $C$  clock cycles. However the row-splitting-after-merging technique can generate much more irregular precodes, and is thus better from the point of view of coding-performance.

## 5.6 Simulation Results

In this section, simulations are used to explore and/or verify different aspects of the proposed Raptor code construction. These aspects include: 1) the FER/BER performance of the constructed codes over AWGN channel, 2) the decoding convergence speed, and the 3) the number of messages exchanged per decoding iteration between the bit-node memory and the check-node block. The first aspect is related to the coding performance of the constructed codes; the other two aspects are related to hardware-efficiency of decoding.

As a proof of concept, a serial decoder applying the TDMP decoding algorithm was designed. A bit-accurate C++ simulator of the decoder was developed to analyze the quantization noise effect on the coding performance. The decoder was modeled in Verilog, and then synthesized using the Design Compiler tool from Synopsys. The power and area figures corresponding to each component of the decoder are obtained, and the figures are analyzed.

### 5.6.1 Code Category 1: $p$ -based Replication and Row-Splitting

Raptor code instances were constructed using the  $p$ -replication method to construct the source matrix, and row-splitting to generate the LT subcodes. The parameter  $p$  is set to 11, therefore,  $K_{LT} = p^2 \cdot (p - 1) = 1210$ . The rates considered are 0.4, 0.5 and  $\frac{2}{3}$ . The construction of the LT codes is done separately for each rate, therefore, the codes obtained are not rate-compatible but fixed-rate. Their performance evaluation is merely intended to show whether the corresponding code construction can yield good-performing fixed-rate codes. For each code-rate, the subcodes are generated by reading a circular partition table in a circular manner (see 5.4). The partition set stored in this table is chosen by simple search: several partition sets are tried and the one yielding the best performance is chosen. Table 5.2 shows the details of the codes compared.

Both the regular and irregular LDPC precodes are 4-cycle-free and have a minimum distance of 6. The irregular precode outperforms the regular precode at rate 0.66, and therefore was used there. The TPMP decoding of the codes was simulated assuming an AWGN channel and BPSK modulation. Three metrics were evaluated: the bit-error rate (BER), frame error rate (FER) and average number of iterations required for successful decoding, with a maximum of 100 iterations. The results are plotted in Figs. 5.11, 5.12, 5.12, and 5.14.

Table 5.2: Parameters of the simulated Raptor codes and LDPC codes.

Rate	Label	Type	Parameters	$(K, K_{LT}, N)$
0.4	RAP <sub>4</sub>	Raptor	Precode: Regular rate $(p - 2)/(p - 1) = 0.9$ LDPC code	(1089, 1210, 2723)
	LDPC <sub>4</sub>	LDPC	Random (3, 5), 4-cycle-free	(1089, -, 2723)
0.5	RAP <sub>5</sub>	Raptor	Precode: Regular rate $(p - 2)/(p - 1) = 0.9$ LDPC code	(1089, 1210, 2178)
	LDPC <sub>5</sub>	LDPC	IEEE 802.16 code [10]	(1104, -, 2208)
2/3	RAP <sub>6</sub>	Raptor	Precode: Irregular rate $1 - 1/p = 10/11$ LDPC code bit-degree = 3, check-degree distribution = $0.2x^2 + 0.3x^3 + 0.5x^4$	(1100, 1210, 1650)
	LDPC <sub>6A</sub>	LDPC	IEEE 802.16 code [10]	(1088, -, 1632)
	LDPC <sub>6B</sub>	LDPC	IEEE 802.16 code [10]	(1088, -, 1632)

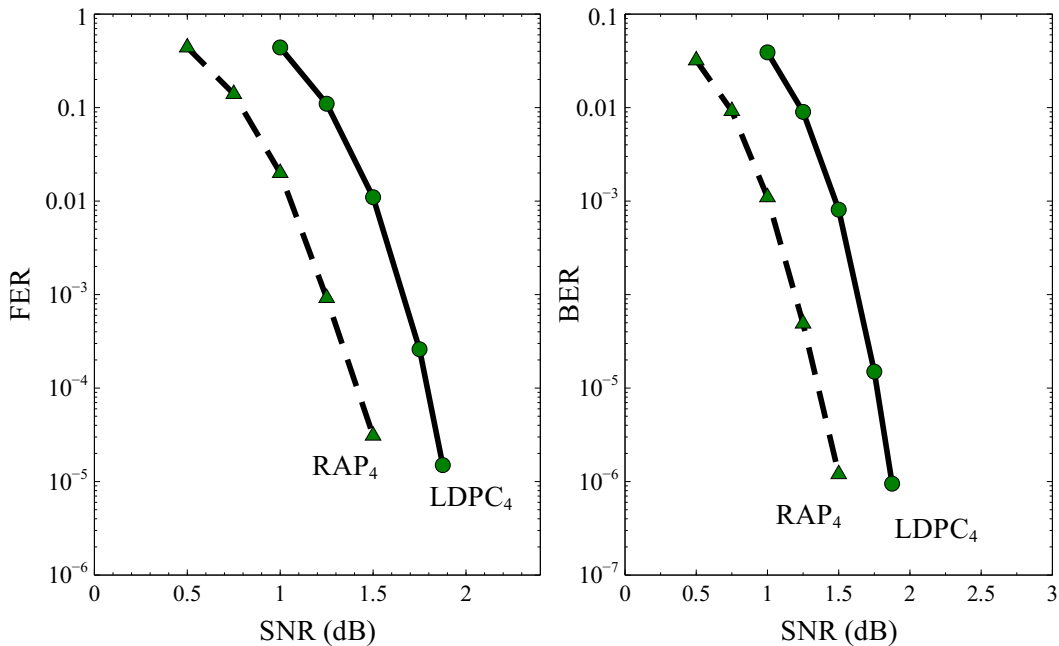


Figure 5.11: FER and BER vs.  $\text{SNR}(E_b/N_0)$  curves for rate-0.4 LDPC and Raptor codes.

The Raptor code outperforms the rate-0.4 LDPC code, compares favorably to the rate-0.5 LDPC code and unfavorably to the rate-2/3 LDPC codes. However, the relatively low minimum distance of the precode implies that an error floor would appear at low error rates (at FER of  $\sim 10^{-6}$  or lower, or equivalently BER of  $10^{-8}$ ). To get error floors at lower rates, the precode rate has to be decreased. No Raptor code of rate 3/4 or 5/6 could be constructed to have comparable performance to that of its LDPC counterpart.

As shown in Fig. 5.14, the decoding convergence speed of the Raptor and LDPC codes is comparable for rate 0.5. The average number of iterations in Raptor decoding is significantly higher than in LDPC decoding ( $\sim 3\times$ ) at rate 2/3. This is due to the nature of the LT code where the transmitted bits are modulo-2 sums of the LT input bits rather than input bits themselves. For rate 2/3, decoding of the constructed Raptor code requires 9 iterations to converge even in noiseless channel conditions. This is clearly related to the fact that 2/3 was the highest rate for which a Raptor code with performance comparable to that of LDPC could be attained.

For the TPMP algorithm, the number of messages (bit-to-check or check-to-bit) is equal to the number of edges in the corresponding LDPC and Raptor graphs. These numbers, compared in Table 5.3, indicate that the LDPC graphs are sparser by a factor of 4/3 than their Raptor counterparts.

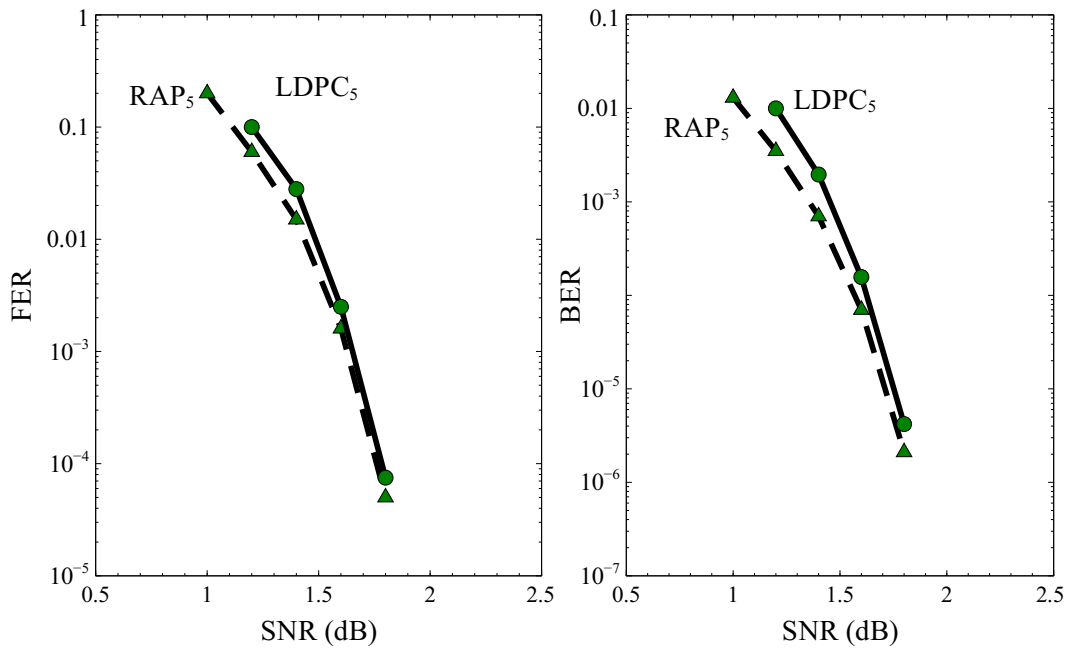


Figure 5.12: FER and BER vs.  $\text{SNR}(E_b/N_0)$  curves for rate-0.5 LDPC and Raptor codes.

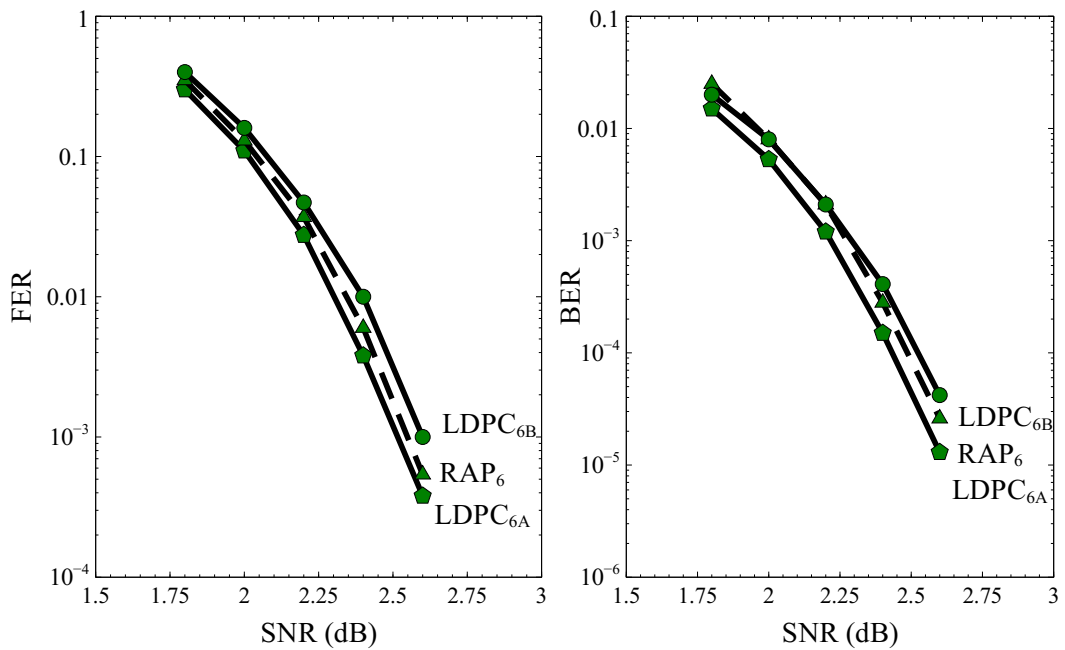


Figure 5.13: FER and BER vs.  $\text{SNR}(E_b/N_0)$  curves for rate-2/3 LDPC and Raptor codes.

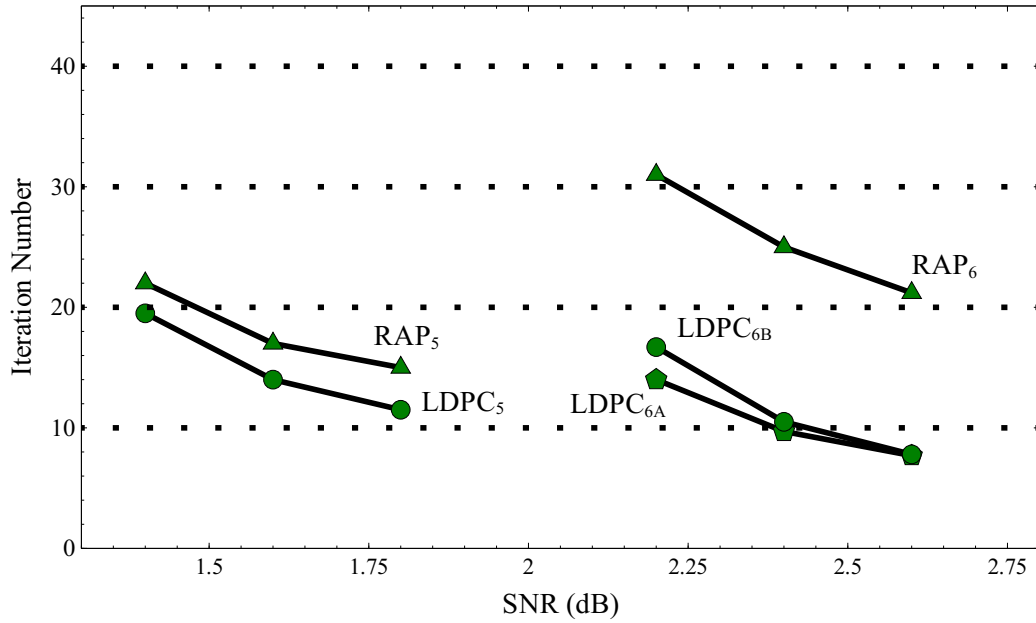


Figure 5.14: Average number of iterations required until decoding convergence vs. SNR for all codes in Table 5.2.

Table 5.3: Number of edges in the matching Raptor and LDPC tanner graphs

Rate	LDPC	Raptor	Sparsity Ratio (Raptor/LDPC)
0.4	8169	11056	1.35
0.5	6992	9076	1.3
2/3	5440,5508	7438	1.37

## 5.6.2 Code Category 2: Product-group Replication and Subcode Structuring

A rate-compatible Raptor code instance was constructed using the product-group replication method to construct the source matrix, and subcode structuring to generate the LT subcodes. A code instance with source matrix parameters  $(n, p, q, M) = (5, 3, 4, 12)$  is considered. The precode rate is chosen using the hardware-constrained rate criterion described in 5.5.2. Substituting  $(d_v, C, R_0, R'_0)$  by  $(2, 3, 0.9, 5/6)$ , the condition  $d'_0 \geq d_v \cdot (d_v \cdot r - C \cdot (1 - r))^{-1} \cdot d_0$ , reduces to  $\sim d'_0 \geq 4/3d_0$ , and the precode rate is set to 5/6, therefore  $(K, K_{LT}) = (1200, 1440)$ . Codes with rates down to 1/2 were constructed. Several LT-row subcode structures and precode row-split patterns were tried and those yielding the best coding performance chosen. The code frame error rate (FER) performance was simulated over an AWGN channel with BPSK modulation, and compared to that of the LDPC codes defined in IEEE802.16 [10] for rates  $\{5/6, 3/4, 2/3, 1/2\}$ , under two-phase message passing (TPMP) decoding. The results are shown in Fig. 5.15.

The code performance follows closely that of the LDPC codes, unfavorably for high rates, with improving performance as the rate decreases. The LDPC waterfall curve is slightly steeper. A rate-1/2 code with  $K_0 = 3600$  was constructed using LT and precode distributions nearly similar to that of its  $K_0 = 1200$  counterpart. Its FER curve shows the general trend of steeper waterfall and shift towards capacity as  $K$  increases, demonstrating the scheme's potential for application over a wide range of block lengths.

Structured LT row-encoding enhances the decoding convergence speed as shown in Table 5.4. The  $3\times$  gap reported in 5.6.1 for rate 2/3 is narrowed to  $2\times$  under TPMP decoding. Furthermore, a  $2.5\times$  to  $2.7\times$  reduction in the average number of iterations is achieved for Raptor codes at high rates by switching to TDMP decoding, as opposed to  $2\times$  reduction for LDPC. The gap is therefore narrowed further to 1.36 at rate 2/3.

Table 5.4: Average number of decoding iterations at FER of  $\sim 10^{-4}$ .

Rate	TPMP			TDMP		
	LDPC	Raptor	Ratio	LDPC	Raptor	Ratio
5/6	4	11.5	2.87	2.4	4.3	1.8
3/4	5.8	14	2.4	3.2	5.6	1.75
2/3	7.6	15.5	2	4.2	5.7	1.35
1/2	11.5	14.6	1.27	6.1	6.7	1.1

The number of messages stored in both LDPC and Raptor codes is compared in Table 5.5. The ratio of the LT message count to the LT edge count ranges between 0.67 and 0.75. Such reduction becomes more significant with low rates,

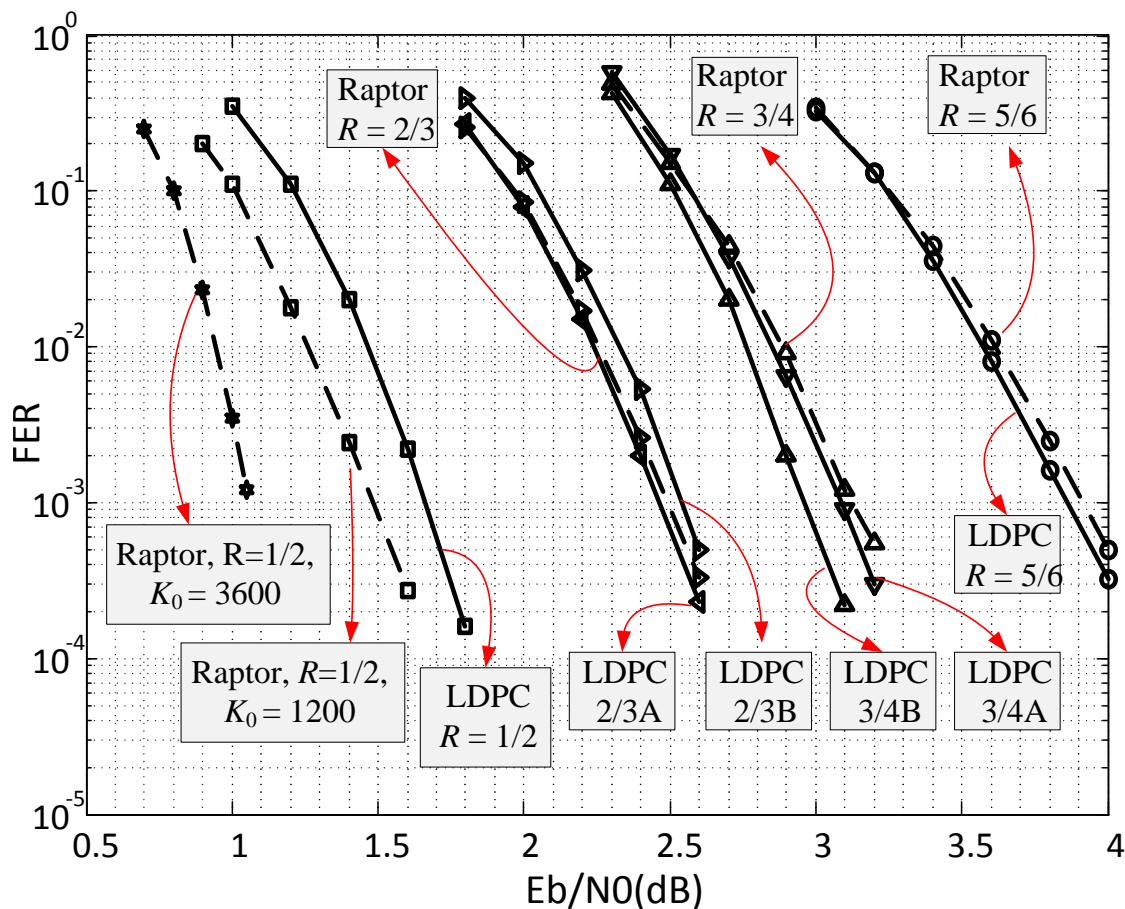


Figure 5.15: FER performance: Dashed curves correspond to Raptor codes, solid curves to LDPC. Unless otherwise stated,  $K_0 = 1200$ .

as the LT edge count percentage of the overall Raptor graph edge count increases. The Raptor message count, relative to LDPC, is reduced from  $1.3\times$  as reported in 5.6.1 to  $1.1\times$ , at rate  $1/2$ . It is noteworthy that since the memory size must be equal to the highest possible message count, the most significant figure, area-wise, is that of the low-rate codes with the largest block-size.

Simulation results demonstrate the effectiveness of subcode structuring in extending the rate-range of Raptor codes and in enhancing the decoding convergence speed, for AWGN channels with no intra-frame varying channel conditions.

### 5.6.3 Implementation

A serial decoder architecture for the Code category 1 (5.6.1) is designed. Some implementation details are described next.

The permuter and inverse permuter of the interconnect network (see Fig.5.1),



Table 5.5: Message Count Comparison:  $N_e$ , edge count per information bit;  $N_m$ , message count per information bit.

	LDPC	LT		Raptor		$N_m^{\text{Rapt}}/N_e^{\text{LDPC}}$
Rate	$N_e$	$N_e$	$N_m$	$N_e$	$N_m$	
5/6	4	2.3	1.7	5.9	5.3	1.3
3/4	4.72	2.83	2.1	6.43	5.7	1.2
2/3	5	3.45	2.4	7.05	6	1.2
1/2	6.33	5.12	3.45	8.72	7.05	1.11

used in the current design generates  $(p-1)^2$  possible permutations, and maps index  $x$  to index  $y$ ,  $0 \leq x, y \leq p-2$  as such:  $y = (q^a(x+1) \bmod p+b) \bmod (p-1)$ ,  $q$  being a primitive element of the group  $GF(p)$ . Parameters  $a$  and  $b$  are pseudo-randomly generated using linear feedback shift registers (LFSRs). Expressing  $q^a = q^{a_0} \cdot q^{2a_1} \cdot q^{4a_2} \dots$ , where  $a_0, a_1, \dots \in \{0, 1\}$ , the multiplication module can be implemented using  $(p-1)\lceil \log(p-1) \rceil$  2:1  $b$ -bit multiplexers, thus the whole permuter needs  $2(p-1)\lceil \log(p-1) \rceil$  [2:1]  $b$ -bit muxes.

The function  $\psi(x) = -\log(\tanh(x/2))$ , in eqs. (3.1) and (3.2), is typically implemented using a lookup table (LUT), whose size grows exponentially with the word size  $b$ . This adds to the CFU size and may cause a delay bottleneck. In this implementation,  $\psi(\cdot)$  was approximated by the following linear-piecewise function:  $\psi(x) \approx a_i x - b_i \forall x \in R_i$ ,  $R_i$  being a subset of the input range. The factor  $a_i$  is chosen to be a positive or negative power of two and, therefore, mere shifting is needed to implement multiplication by  $a_i$ . The offset values  $b_i$  are stored in a LUT. The input range is decomposed into  $R_i = [2^{i-1}, 2^i - 1]$ ,  $i = 1, \dots, b$ . This range decomposition simplifies the decoding-circuitry of the LUT storing the  $b$  possible values of the offset. Better approximation can be obtained by cutting each sub-range size by half, thus doubling the LUT size. The piecewise-linear approximation is compared to the  $\psi(\cdot)$  function and the LUT approximation for  $b = 7$  in Fig. 5.16.

To satisfy the *TDMP timing condition*, described in 3.6, the rows of  $\mathbf{H}'_0$  are shuffled such that: any  $p$  consecutive rows of  $\mathbf{H}'_0$  have at most 1 nonzero entry per column.

Table 5.6 summarizes the hardware resources involved in the TDMP and TPMP serial Raptor decoders. Measured in iterations per second, the throughput of the two architectures is basically the same. The difference, in the throughput measured as the number of decoded frames per second, stems from the the fact that a smaller number of iterations is needed to decode a single frame using TDMP compared to TPMP. The TDMP and TPMP implementations involve nearly the same hardware resources. The interconnect network in the TDMP implementation includes a FIFO to organize forwarding the bit-to-check messages to the adder block where they are added to the corresponding computed check-to-bit messages.

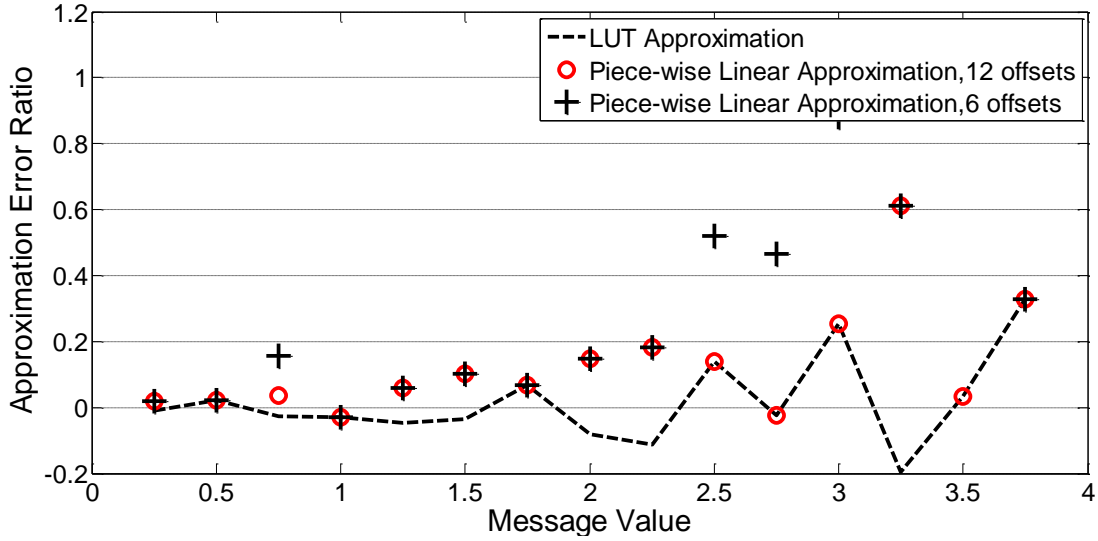


Figure 5.16: The ratio of the approximation error induced by the linear piecewise approximation compared to that of the LUT-based approximation, for input  $x < 4$ , and  $b = 7$  using 2's complement representation. The function  $\psi(x)$ , however, is approximated using 6 bits.

**Quantization Analysis:** A bit-accurate C++ simulator of the serial decoder was developed to analyze the quantization noise effect on the coding performance. The decoding procedure of a rate-0.4 Raptor code instance, was simulated for message quantization of 6, 7, and 21 (almost-ideal) bits, assuming an AWGN channel with BPSK modulation. The code parameters  $(p, K, K_{LT}, N)$  were set to (11,1089,1210,2717). As indicated by the simulation results, shown in Fig. 5.17, the BER has less steep waterfall regions compared to the floating point case. The message bit-width of  $b = 7$  is the smallest width to yield close performance to the floating point case up to an SNR of at least 1.5 dB. Using  $b = 7$ , the decoder performance was evaluated when using the linear-piecewise function to compute

Table 5.6: Complexity and throughput of the serial TDMP and TPMP Decoders. The blocks are  $b$ -bit wide.

Component	TDMP Serial Decoder	TPMP Serial Decoder
Memory	$(p - 1)((1 + f)p^3 + p^2)$	$(p - 1)((1 + f)p^3 + 2p^2)$
REGISTERS	$\sim p(p - 1)$	$\sim p(p - 1)$
Adders	$2(p - 1)$	$2(p - 1)$
[2 : 1] MUXs	$p^2/2 + 5p \lceil \log(p - 1) \rceil$	$p^2/2 + 5p \lceil \log(p - 1) \rceil + 2p$
FIFO	1	0
CFUs	1	1

the  $\psi(\cdot)$  function (2 approximations) and also compared to an equivalent two-phase message-passing (TPMP) decoder. Simulation results show moderate to slight performance loss resulting from the usage of linear piecewise approximation, depending on the level of approximation; In the two approximation cases, the loss is still smaller than the gap separating the decoder performance for  $b = 6$  and  $b = 7$ . The TPMP decoder slightly outperforms TDMP, under quantization, but this comes at the expense of nearly half convergence speed.

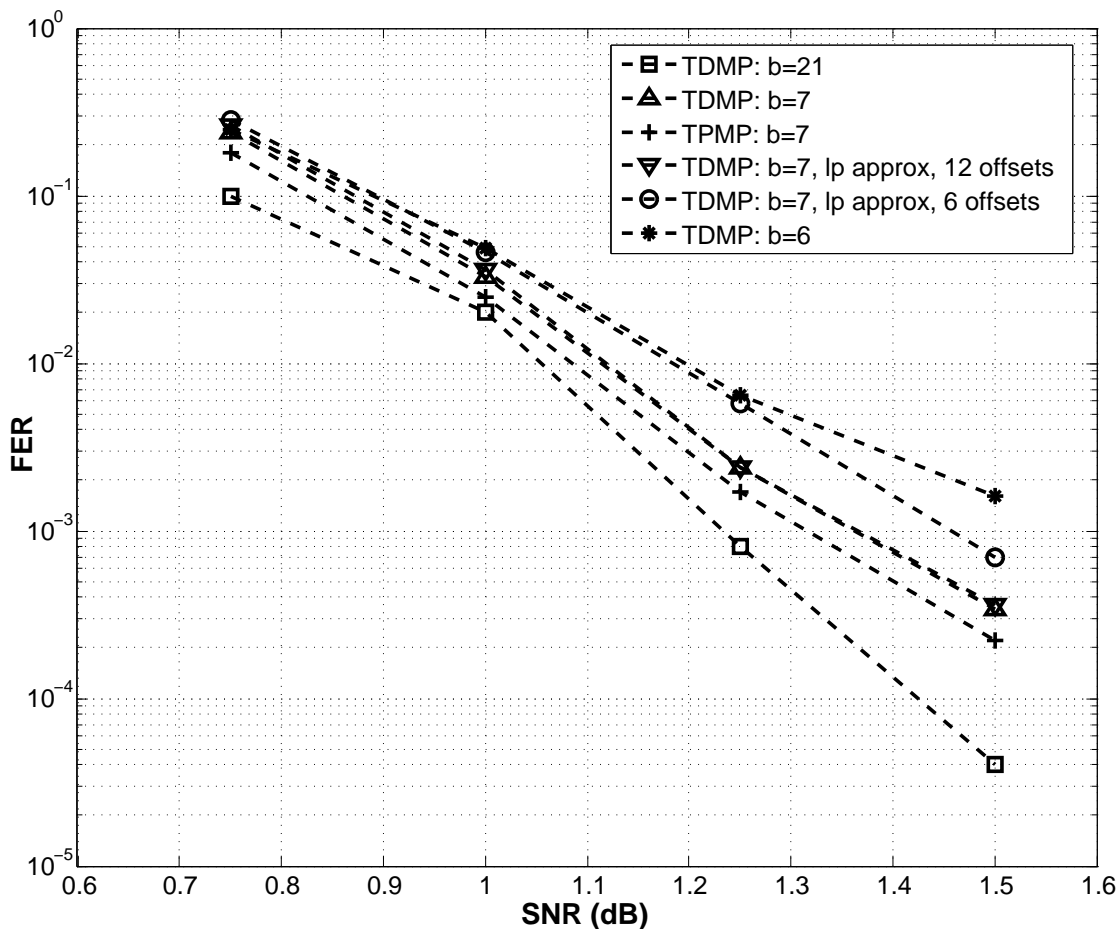


Figure 5.17: Coding performance of TDMP under various quantization scenarios: Near floating-point case,  $b = 7$  using lookup-table and linear piecewise approximations, and  $b = 6$ . TPMP performance at  $b = 7$  is shown for comparison.

**Synthesis Results:** The proposed decoder was modeled in Verilog, and then synthesized using the Design Compiler tool from Synopsys. The datapath bit-width,  $b$ , was set to 7, and the synthesis was done using a 65 nm, 1.2 V customized library (tcbn65lptc). The synthesized decoder has a critical delay of 3.21ns, and,

therefore, operates at a maximum frequency of 300 MHz. It occupies an area of  $0.55 \text{ mm}^2$  and dissipates an average power of 27 mW. For each decoding iteration, the decoder needs 741 cycles to process the LT rows and 363 cycles to process the LDPC rows of the rate-0.4 code instance, in addition to 45 extra cycles to switch between LDPC and LT decoding and to the next iteration, thus performing one iteration in 1149 cycles. For an SNR (signal to noise ratio) of 1.5 dB, the average number of iterations needed for convergence is 7.9 per frame, resulting in a maximum of 33050 processed frames per second under 300 MHz frequency, or equivalently a throughput of 36 Mb/s.

The area and power estimates of the different components of the decoder are detailed in Table 5.7. The power estimation assumes a default switching activity of 50%. The results show that memory dominates the decoder in both area and power consumption (89% and 58% respectively). The area dominance of memory blocks can be attributed to the low throughput of the decoder which processes a single row of  $\mathbf{H}'_0$  per cycle. Unlike memory, the number of processing elements replicates proportionally to the number of processed rows per cycle, being limited to one in this case. The check-node memory stores up to  $p^3(p-1) \times b$  bits, and performs  $p-1$  reads and  $p-1$  writes per cycle ( $\sim 65\%$  of the decoder memory cells and  $40\% \sim 50\%$  of the memory operations), which explains its large contribution in terms of area and power. Particularly, the high power cost of the check node memory, suggests the need to optimize it through partitioning and making use of the sequential access pattern in the check-node memory.

The area cost of the logic component follows the level of parallelism in the decoding procedure (i.e. the number of processed rows per cycle), which justifies the analysis and consequent optimization of the different logic sub components of the decoder. The CFU contribution to the logic area and power is 37% and 38% respectively, compared to 45% and 50% in the case of the interconnect network (composed of the permuters, IVM network, datapath registers and FIFO). The area occupied by the all pipeline registers in the decoder, intended primarily to increase throughput by decreasing the critical path delay, accounts for about 25% of the logic area.

Table 5.7: Power and Area estimates of each component of the synthesized decoder.

		Area( $\times 10^3 \mu m^2$ )	Area(%)	Power( $mW$ )	Power(%)
<b>LOGIC</b>	Memory Address Generation	10.7	1.95	1.3	4.9
	Permuter/Deprmuter/IVM Network	7.6	1.38	0.4	1.5
	Adders/Subtractors	2.6	0.47	0.3	1.1
	Datapath Registers	7.4	1.35	2.9	10.9
	FIFO	10.6	1.93	2	7.5
	CFU (except $\psi$ generators)	14.4	2.62	4	15
	CFU- $\psi$ generators	8.9	1.62	0.3	1.1
	<b>Total logic</b>	<b>62.2</b>	<b>11.3</b>	<b>11.2</b>	<b>42.1</b>
<b>MEMORY</b>	Bit-Node Memory	17	3.1	2.4	9
	Check-Node Memory	350	63.7	8.4	31.6
	Intrinsic Value Memory	120	21.85	4.6	17.3
	<b>Total memory</b>	<b>487</b>	<b>88.7</b>	<b>15.4</b>	<b>57.9</b>
<b>TOTAL</b>		<b>549.2</b>		<b>26.6</b>	

# Chapter 6

## Conclusion and Open Questions

Several conclusions can be made concerning the topics, methods, and results presented in this dissertation. Besides, the methods proposed here generate, themselves, a new set of questions and challenges that are yet to be approached.

In summary, the contribution of this dissertation can be divided into two parts. In the first part, it is shown that channel-to-rate matching can be applied on the receiver side, and that such matching is useful in the broadcast communication scenario to achieve significantly higher data rates for a limited hardware overhead. The matching is applied via a novel increment-based inter-frame (IIF) coding approach. The work in this part includes the algorithms, architectures, asymptotic analysis and optimization, and analysis of the enhancement brought to the achievable data rates.

In the second part, an architecture-aware Raptor code construction framework is proposed. The flow of the proposed framework is intended to achieve two “opposing” goals: 1) the constructed codes can pertain much of the irregularity features of Raptor codes, as well as 2) attain the appropriate underlying structure needed for hardware-efficiency of decoding. Simulations show the potential of these codes to produce good-performance as well as hardware-efficient decoder implementations. The work in this part includes the framework, the methods involved in each step of the framework, the resulting architecture and block design, and the study of the architectural implications of the different methods under consideration.

### 6.1 Conclusions

The first main conclusion(s) is related to the IIF scheme. First, channel-to-rate matching can be done on the receiver side. This means that while IIF coding is targeted for broadcast communication, it is appropriate whenever the feedback from the receiver(s) to the transmitter is costly or undesirable. Second, the graph-based

simplification and subsequent analysis show that the matching can be achieved using an iterative process that is a generalization of LT erasure decoding; besides, the process can be made “optimal”, under some assumptions, by the appropriate randomized construction of bi-partite graphs. Third, the optimal distribution of the decoding tasks over different layers and schemes follows from the organization of the hardware resources at hand (inter-frame versus intra-frame decoding), beside the different possible abstractions of the channel (PHY-layer soft-decoding versus APP-layer erasure decoding, etc.). In this regard, the IIF scheme outperforms other conventional schemes, such as the state-of-the-art two-stage scheme, while using hardware resources that are, for most, not dedicated/specific to the scheme.

The second main conclusion is that the design of rate-compatible codes has to be rethought as they are applied in the IIF scheme. Under this scheme, communication scenarios such as the *intra-frame varying channel-conditions*, already possible in IR-HARQ, become more frequent. The design of the code should be tailored to make its coding performance less sensitive to the order/indices of the transmitted bits. This is a challenging task because the latter sensitivity follows from the structure of the code, which in turn is crucial to obtain good coding performance using the iterative decoding methods.

The third conclusion is that architecture-aware short/moderate-length Raptor codes can be constructed. The corresponding Raptor construction process involves a combination of pseudo-random encoding and structuring. This combination is intended to preserve the features of the Raptor code which are projected to make this code outperform other conventional codes under some communication scenarios such as the *intra-frame varying channel-conditions*. As a result of the code construction method, the decoder architecture can be efficiently organized as a row-processor of a regular matrix. It can decode a large number of possible Raptor codes; besides, it can be further modified to decode any LDPC code which structure is compatible with that of the constructed Raptor codes. Preliminary results show the potential of the constructed Raptor codes to perform well under AWGN channels. Yet, issues related to the decoding convergence speed and the effective code-rate range have to be carefully considered.

## 6.2 Open Questions

Several questions, yet to be answered, arise from the methods proposed in this dissertation. Answering these questions makes it possible to attain a deeper understanding of the corresponding methods, their implications, and subsequently their applicability. The questions fall in two categories, one corresponding to the IIF scheme, the other to the proposed architecture-aware Raptor code construction.

Basic questions raised on the IIF scheme include its asymptotic performance, finite-length performance and optimal implementation. Concerning the asymptotic performance, two questions have to be considered: the first is on the existence and description of optimal IIF codes when the channel-characterizing probability distribution  $(\delta_\omega)_{\omega \geq 0}$  is not a geometric progression; the second, denoted here the *problem of designing universal inter-frame codes*, is on the minimum achievable  $\frac{K_S}{N_F}$  when the different channels corresponding to different receivers are described by different  $(\delta, \mu)$  pairs. Concerning finite-length performance, the main question is on the construction and the achievable performance of finite-length inter-frame codes. In the subject of implementation of the IIF encoder/decoder, three major questions arise: 1) how to organize the scheduling of the transmission of the subframes and frames? 2) how does the IIF code structure affect both the transmission scheduling and the hardware efficiency of the IIF decoding/encoding? 3) what is the optimal accuracy-complexity tradeoff involved in developing procedures that predict the success or failure of intra-frame decoding of a frame without performing the decoding procedure itself?

One interesting question is on the design of efficient rate-compatible IIF codes, defined similarly to rate-compatible PHY-layer codes. In such codes,  $\frac{K_S}{N_F}$  is increased by sending additional subframes, possibly upon request from the receivers.

Some questions arise from the asymptotic analysis process itself. For example, it can be seen that the optimal edge-degree distribution in the iterative decoding/matching process is a generalization of that in erasure decoding. A resulting question is then on the possible existence of some structure or generalized form of optimal degree distributions for a class of iterative processes and/or probability distributions  $(\delta_\omega)_{\omega \geq 0}$ . This problem can have applications in different fields in which such iterative matching processes may arise.

The major question yet to be answered on architecture-aware Raptor codes follows from the logic underlying the work on these codes in this dissertation, which is summarized here. First, the following observation is made: the peculiar Raptor features can yield codes that have good coding-performance in the different scenarios that may arise in unicast and broadcast communication. Second, motivated by this observation, it is shown that architecture-aware Raptor codes can be constructed, using a multi-stage construction flow, to have hardware-efficient decoder architectures. In addition, several methods and/or guidelines for each of the construction flow stages are considered so that the generated codes are good-performing. Simulation results show the potential of the constructed Raptor codes to yield coding-performance that is comparable to that of LDPC codes, over the AWGN channel. The question to be answered follows naturally: how can the LT subcodes and the precode parameters (rate, merging factor and possible split-after merge) be chosen to yield the best possible coding-performance, for a given sce-



nario? How does its performance compares to that of conventional Turbo and LDPC codes? Answering this question is crucial to validate the initial observation which has motivated the whole work on Raptor codes.

# Appendix A

## Abbreviations

APP-layer	Application-Layer
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BTC	Bit-to-Check (Message)
CFU	Check-Function Unit
CSI	Channel-State Information
CTB	Check-to-Bit (Message)
FER	Frame Error Rate
HARQ	Hybrid Automatic Repeat Request
IIF	Increment-Based Inter-Frame
LDPC	Low-Density Parity-Check (Code)
LLR	Log-likelihood Ratio
LT	Luby-Transform
PHY-layer	Physical-Layer
SNR	Signal-to-Noise Ratio
TDMP	Turbo-Decoding Message-Passing
TPMP	Two-Phase Message-Passing

# Bibliography

- [1] H. Zeineddine, M. M. Mansour, and R. Puri, “Construction and hardware-efficient decoding of Raptor codes,” *IEEE Trans. Signal Process.*, vol. 59, pp. 2943–2960, June 2011.
- [2] H. Zeineddine and M. M. Mansour, “A reconfigurable TDMP decoder for Raptor codes,” *J. Signal Process. Sys.*, vol. 69, pp. 293–304, Dec. 2012.
- [3] H. Zeineddine and M. M. Mansour, “Reconfigurable decoder architectures for Raptor codes,” in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process. (ICASSP)*, pp. 1669–1672, May 2011.
- [4] H. Zeineddine, L. Jalloul, and M. M. Mansour, “Hardware-oriented construction of a family of rate-compatible Raptor codes,” *IEEE Commun. Lett.*, vol. 18, pp. 1131–1134, July 2014.
- [5] H. Zeineddine and M. M. Mansour, “Inter-frame coding for broadcast communication - part I: Algorithms and architectures,” *submitted to IEEE Trans. Comm.*
- [6] H. Zeineddine and M. M. Mansour, “Inter-frame coding for broadcast communication - part II: Mathematical characterization,” *submitted to IEEE Trans. Comm.*
- [7] S. Sesia, I. Toufic, and M. Baker, eds., *LTE: The UMTS Long Term Evolution, From Theory to Practice*. John-Wiley, 2011.
- [8] D. Chase, “Code combining – A maximum-likelihood decoding approach for combining an arbitrary number of noisy packets,” *IEEE Trans. Commun.*, vol. 33, pp. 385–393, May 1985.
- [9] S. Lin and P. Yu, “A hybrid ARQ scheme with parity retransmission for error control of satellite channels,” *IEEE Trans. Commun.*, vol. 30, pp. 1701–1719, July 1982.

- [10] “IEEE standard for local and metropolitan area networks - Part 16: Air interface for broadband wireless access systems,” *IEEE 802.16*.
- [11] *3GPP TS 36.212 V8.8.0 (2009-2012) 3rd Generation Partnership Project, Technical Specification Group Radio Access Network*.
- [12] *3GPP TS 26.346, Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service (MBMS); Protocols and Codecs, 3GPP Technical Specification, Rev. V7.4.1, June 2007*.
- [13] “Digital video broadcasting (DVB); IP datacast over DVB-H: Content delivery protocols, ETSI technical specification,” 2006.
- [14] K. Wang, Z. Chen, and H. Liu, “A novel broadcasting scheme for LT-codes in wireless broadcasting systems,” *IEEE Commun. Lett.*, vol. 17, pp. 972–975, May 2013.
- [15] Q. Zhang, D. Lin, and G. Wu, “A novel decoding method for rateless codes basing on packet combining,” in *Proc. IEEE Int. Conf. Wireless Commun., Netw., Mobile Comp. (WiCOM)*, pp. 1–4, Sept. 2012.
- [16] D. Lin, M. Xiao, Y. Xiao, and S. Li, “Efficient packet combining based on packet-level coding,” *IET Electronics Letters*, vol. 47, pp. 444–445, Mar. 2011.
- [17] C. Berrou and A. Glavieux, “Near optimum error correcting coding and decoding: Turbo-codes,” *IEEE Trans. Commun.*, vol. 44, pp. 1261–1271, Oct. 1996.
- [18] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.
- [19] R. M. Tanner, “A recursive approach to low complexity codes,” *IEEE Trans. Inf. Theory*, vol. 27, pp. 533–547, Sept. 1981.
- [20] D. MacKay and R. Neal, “Near Shannon limit performance of low density parity check codes,” *Electronics Letters*, vol. 32, p. 1645, Aug. 1996.
- [21] T. Richardson, A. Shokrollahi, and R. Urbanke, “Design of capacity approaching irregular low-density parity-check codes,” *IEEE Trans. Inf. Theory*, vol. 47, pp. 619–637, Feb. 2001.
- [22] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews, “Capacity-approaching protograph codes,” *IEEE J. Sel. Areas Commun.*, vol. 27, pp. 619–637, Aug. 2009.

- [23] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, pp. 3051–3073, July 2009.
- [24] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, pp. 2551–2567, June 2006.
- [25] H. Pishro-Nik and F. Fekri, "Results on punctured low-density parity-check codes and improved iterative decoding techniques," *IEEE Trans. Inf. Theory*, vol. 53, pp. 599–614, Feb. 2007.
- [26] C.-H. Hsu and A. Anastasopoulos, "Capacity achieving LDPC codes through puncturing," *IEEE Trans. Inf. Theory*, vol. 54, pp. 4698–4706, Oct. 2008.
- [27] J. Ha, J. Kim, and S. W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 50, pp. 2824–2836, Nov. 2004.
- [28] J. Kim, A. Ramamoorthy, and S. W. McLaughlin, "Design of efficiently-encodable rate-compatible irregular LDPC codes," *IEEE Trans. Commun.*, vol. 57, pp. 365–375, Feb. 2009.
- [29] M. El-Khamy, J. Hou, and N. Bhushan, "Design of rate-compatible structured LDPC codes for hybrid ARQ applications," *IEEE J. Sel. Areas Commun.*, vol. 27, pp. 965–973, Aug. 2009.
- [30] J. Ha, J. Kim, D. Klinc, and S. W. McLaughlin, "Rate-compatible punctured low-density parity-check codes with short block lengths," *IEEE Trans. Commun.*, vol. 52, pp. 728–738, Feb. 2006.
- [31] T. V. Nguyen, A. Nosratinia, and D. Divsalar, "The design of rate-compatible protograph LDPC codes," *IEEE Trans. Commun.*, vol. 60, pp. 2841–2850, Oct. 2012.
- [32] T. V. Nguyen and A. Nosratinia, "Rate-compatible short-length protograph LDPC codes," *IEEE Commun. Lett.*, vol. 17, pp. 948–951, May 2013.
- [33] J. Cheng, "Coding performance of hybrid ARQ schemes," *IEEE Trans. Commun.*, vol. 56, pp. 1017–1029, June 2006.
- [34] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *IEEE Trans. Inf. Theory*, vol. 52, pp. 2033–2051, May 2006.
- [35] M. Luby, "LT-codes," in *IEEE Symp. Foundations of Comp. Science (FOCS)*, pp. 271–280, Nov. 2002.

- [36] E. Soljanin, N. Varnica, and P. Whiting, “Punctured vs rateless codes for hybrid ARQ,” in *Proc. IEEE Inf. Theory Workshop, (ITW)*, pp. 155–159, Mar. 2006.
- [37] D. Tse and P. Viswanath, eds., *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.
- [38] “Evolved universal terrestrial radio access (E-UTRA); physical channels and modulation.”
- [39] G. Yue and X. Wang, “Adaptive hybrid ARQ in Gaussian and Turbo coded systems,” in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, (New Orleans, LA), pp. 1–5, Dec. 2008.
- [40] A. Das, F. Khan, , and A. Nanda, “A<sup>2</sup>IR: An asynchronous and adaptive hybrid ARQ scheme for 3G Evolution,” in *Proc. IEEE Vehicular Technol. Conf. (VTC)*, (Rhodes, Greece), pp. 628–632, May 2001.
- [41] I. Andriyanova and E. Soljanin, “Optimized IR-HARQ schemes based on punctured LDPC codes over the BEC,” *IEEE Trans. Inf. Theory*, vol. 58, pp. 6433–6445, Oct. 2012.
- [42] *3GPP TS 36.212, Evolved Universal Terrestrial Radio Access (E-UTRA), Multiplexing and channel coding.*
- [43] G. Tan and T. Herfet, “Application layer hybrid error correction with Reed-Solomon code for DVB services over wireless LANs,” in *Proc. IEEE Int. Conf. Wireless Commun., Netw., Mobile Comp. (WiCOM)*, pp. 2952–2955, Sept. 2007.
- [44] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, “RaptorQ forward error correction scheme for object delivery,” in *RFC 6330, Internet Engineering Task Force*, Aug. 2011.
- [45] C. Bouras, N. Kanakis, V. Kokkinos, and A. Papazois, “Evaluating RaptorQ FEC over 3GPP multicast services,” in *Proc. Int. Wireless Commun. and Mobile Comp. Conf. (IWCMC)*, (Limassol, Cyprus), pp. 257–262, Aug. 2012.
- [46] A. Shokrollahi, S. Lassen, and R. Karp, “Systems and processes for decoding chain reaction codes through inactivation,” Feb. 2005. US Patent 6,856,263.
- [47] T. Mladenov, S. Nooshabadi, and K. Kim, “Implementation and evaluation of Raptor codes on embedded systems,” *IEEE Trans. Computers*, vol. 60, pp. 1678–1691, Dec. 2011.

- [48] M. Luby, M. Watson, T. Gasiba, and T. Stockhammer, “Mobile data broadcasting over MBMS tradeoffs in forward error correction,” in *ACM Proc. Int. Conf. Mobile Ubiquitous Multimedia (MUM)*, (Stanford, CA), 2006.
- [49] C. Berger, S. Zhou, Y. Wen, P. Willett, and K. Pattipati, “Optimizing joint erasure- and error-correction coding for wireless packet transmissions,” *IEEE Trans. Wireless Commun.*, vol. 7, pp. 4586–4595, Nov. 2008.
- [50] J. Hagenauer, “Rate-compatible punctured convolutional codes (RCPC codes) and their applications,” *IEEE Trans. Commun.*, vol. 36, pp. 389–400, July 1988.
- [51] R. Mantha and F. R. Kschischang, “A capacity approaching hybrid ARQ scheme using turbo codes,” in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*.
- [52] A. S. Barbulescu and S. S. Pietrobon, “Rate compatible turbo codes,” *IEE Electron. Lett.*, vol. 31, pp. 535–536, Mar. 1995.
- [53] D. N. Rowitch and L. B. Milstein, “On the performance of hybrid FEC/ARQ systems using rate compatible punctured turbo (RCPT) codes,” *IEEE Trans. Commun.*, vol. 48, pp. 948–959, June 2000.
- [54] J. Thorpe, “Low-density parity-check (LDPC) codes constructed from protographs,” *IPN Progress Report*, pp. 42–154, Aug. 2003.
- [55] T. Richardson and R. Urbanke, “Multi-edge type LDPC codes,” *EPFL, Technical Report*, 2004.
- [56] C. Shi and A. Ramamoorthy, “Design and analysis of  $E^2RC$  codes,” *IEEE J. Sel. Areas Commun.*, vol. 27, pp. 889–898, Aug. 2009.
- [57] M. Yazdani and A. Banihashemi, “On construction of rate-compatible low-density parity-check codes,” *IEEE Commun. Lett.*, vol. 8, pp. 159–161, Mar. 2004.
- [58] H. Y. Park, J. W. Kang, K. S. Kim, Senior, and K. C. Whang, “Efficient puncturing method for rate-compatible low-density parity-check codes,” *IEEE Trans. Wireless Commun.*, vol. 6, pp. 3914–3919, Nov. 2007.
- [59] B. N. Vellambi and F. Fekri, “Finite-length rate-compatible LDPC codes: A novel puncturing scheme,” *IEEE Trans. Commun.*, vol. 57, pp. 297–301, Feb. 2009.

- [60] R. Asvadi and A. H. Banihashemi, “Rate-compatible puncturing scheme for finite-length LDPC codes,” *IEEE Commun. Lett.*, vol. 17, pp. 147–150, Jan. 2013.
- [61] K. Zhang, X. Huang, and Z. Wang, “A high-throughput LDPC decoder architecture with rate compatibility,” *IEEE Trans. Circuits and Systems*, vol. 58, pp. 839–846, Apr. 2011.
- [62] D. Klinc, J. Ha, J. Kim, and S. McLaughlin, “Rate-compatible punctured low-density parity-check codes for ultra wide band systems,” in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, pp. 3856–3860, Dec. 2005.
- [63] J. Kwon, D. Klinc, J. Ha, and S. McLaughlin, “Fast decoding of rate-compatible punctured LDPC codes,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pp. 216–220, June 2007.
- [64] J. Li and K. Narayanan, “Rate-compatible low density parity check codes for capacity-approaching ARQ scheme in packet data communications,” in *Proc. Int. Conf. Commun., Internet, and Inf. Technology (CIIT)*, Nov. 2002.
- [65] N. Jacobsen and R. Soni, “Design of rate-compatible irregular LDPC codes based on edge growth and parity splitting,” in *Proc. IEEE Vehicular Technol. Conf. (VTC)*, pp. 1052–1056, Sept. 2007.
- [66] T.-Y. Chen, D. Divsalar, J. Wang, and R. Wesel, “Protograph-based Raptor-like LDPC codes for rate compatibility with short blocklengths,” in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, pp. 1–6, Dec. 2011.
- [67] G. Liva and M. Chiani, “Protograph LDPC codes design based on EXIT analysis,” in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, pp. 3250–3254, Nov. 2007.
- [68] B. K. Butler and P. H. Siegel, “Bounds on the minimum distance of punctured quasi-cyclic LDPC codes,” *IEEE Trans. Inf. Theory*, vol. 59, pp. 4584–4597, July 2013.
- [69] R. Palanki and J. S. Yedidia, “Rateless codes on noisy channels,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, p. 37, July 2004.
- [70] M. M. Mansour and N. R. Shanbhag, “High-throughput LDPC decoders,” *IEEE Trans. VLSI Syst.*, vol. 11, pp. 976–996, Dec. 2003.
- [71] B. Xiang *et al.*, “An area-efficient and low-power multirate decoder for quasi-cyclic low-density parity-check codes,” *IEEE Trans. VLSI Syst.*, vol. 18, pp. 1447–1460, Oct. 2010.



- [72] X. H. Kai Zhang and Z. Wang, “High-throughput layered decoder implementation for quasi-cyclic LDPC codes,” *IEEE Trans. VLSI Syst.*, vol. 27, pp. 985–994, Aug. 2009.
- [73] B. Xiang, D. Bao, S. Huang, and X. Zeng, “An 847-955 Mb/s 342-397 mw dual-path fully-overlapped QC-LDPC decoder for WiMAX system in 0.13 $\mu$ m CMOS,” *IEEE J. Solid State Circuits*, vol. 46, pp. 1416–1432, June 2011.
- [74] C. Sham, X. Chen, F. C. M. Lau, Y. Zhao, and W. M. Tam, “A 2.0 Gb/s throughput decoder for QC-LDPC convolutional codes,” *IEEE Trans. Circuits and Systems*, vol. 60, pp. 1857–1869, July 2013.
- [75] N. Jiang, K. Peng, J. Song, C. Pan, , and Z. Yang, “High-throughput QC-LDPC decoders,” *IEEE Trans. Broadcasting*, vol. 55, pp. 251–259, June 2009.
- [76] M. M. Mansour and N. R. Shanbhag, “A 640 Mb/s 2048-bit programmable LDPC decoder chip,” *IEEE J. Solid State Circuits*, vol. 41, pp. 684–698, Mar. 2006.
- [77] M. M. Mansour and N. R. Shanbhag, “Architecture-aware low-density parity-check codes,” in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, vol. 2, pp. II–57 – II–60, May 2003.
- [78] M. M. Mansour and N. R. Shanbhag, “A novel design methodology for high-performance programmable decoder cores for AA-LDPC codes,” in *Proc. Int. Workshop Signal Processing Systems(SiPS’03)*, pp. 29–34, Aug. 2003.
- [79] M. M. M. Fossorier and H. Imai, “Reduced complexity iterative decoding of low-density parity-check codes based on belief propagation,” *IEEE Trans. Commun.*, vol. 47, pp. 673–680, May 1999.
- [80] X.-Y. Hu, E. Eleftherious, D.-M. Arnold, and A. Dholakia, “Efficient implementation of the sum-product algorithm for decoding LDPC codes,” in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, pp. 1036–1036E, Nov. 2001.
- [81] J. Chen and M. Fossorier, “Density evolution for two improved BP-based decoding algorithms of LDPC codes,” *IEEE Commun. Lett.*, vol. 6, pp. 208–210, May 2002.
- [82] M. M. Mansour, “VLSI design for high-speed sparse parity-check matrix decoders,” in *Proc. Asilomar Conf. Signals, Systems and Computers (Asilomar)*, pp. 708–712, 2005.

- [83] M. M. Mansour, “A turbo-decoding message-passing algorithm for sparse parity-check matrix codes,” *IEEE Trans. Signal Process.*, vol. 54, pp. 4376–4392, Nov. 2006.
- [84] T. Okamura, “A Hybrid ARQ scheme based on shortened low-density parity-check codes,” in *Proc. IEEE Wireless Commun. and Netw. Conf. (WCNC)*, (Las Vegas, NV), pp. 82–87, Apr. 2008.
- [85] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, “Efficient erasure correcting codes,” *IEEE Trans. Inf. Theory*, vol. 47, pp. 569–584, Feb. 2001.
- [86] M. A. S. D. S. M. Luby, M. Mitzenmacher and V. Stemann, “Practical loss-resilient codes,” in *Proc. 29th Symp. on Theory of Computing*, pp. 150–159, 1997.
- [87] M. Luby, M. Mitzenmacher, and M. A. Shokrollahi, “Analysis of random processes via and-or tree evaluation,” in *Proc. 9th Annu. ACM-SIAM Symp. Discrete Algorithms*, pp. 364–373, 1998.
- [88] A. J. Felstrom and K. Zigangirov, “Time-varying periodic convolutional codes with low-density parity-check matrices,” *IEEE Trans. Inf. Theory*, vol. 45, pp. 2181–2191, Sept. 1999.
- [89] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Trans. Inf. Theory*, vol. 20, pp. 284–287, Mar. 1974.
- [90] A. Venkiah, C. Poulliat, and D. Declercq, “Rate splitting issue for finite length Raptor codes,” in *Proc. IEEE Sarnoff Symp.*, pp. 1–5, Apr. 2008.
- [91] A. Venkiah, C. Poulliat, and D. Declercq, “Analysis and design of Raptor codes for joint decoding using information content evolution,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pp. 421–425, June 2007.