UNIVERSITY OF CALIFORNIA

Los Angeles

Low-Complexity Decoding of Low-Density Parity Check Codes Through Optimal Quantization and Machine Learning and Optimal Modulation and Coding for Short Block-Length Transmissions

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical Engineering

by

Linfang Wang

2023

ABSTRACT OF THE DISSERTATION

Low-Complexity Decoding of Low-Density Parity Check Codes
Through Optimal Quantization and Machine Learning
and Optimal Modulation and Coding for Short Block-Length Transmissions

by

Linfang Wang
Doctor of Philosophy in Electrical Engineering
University of California, Los Angeles, 2023
Professor Richard D. Wesel, Chair

(Abstract omitted for brevity)

The dissertation of Linfang Wang is approved.

Lara Dolecek

Gregory J. Pottie

Dariush Divsalar

Christina Fragouli

Richard D. Wesel, Committee Chair

University of California, Los Angeles

2023

To my parents, Genqi and Xiangqun ...

To my wife, my dear Hanzhi (Stephanie) ...

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

(Acknowledgments omitted for brevity.)

VITA

1974–1975 Campus computer center "User Services" programmer and consultant, Stanford Center for Information Processing, Stanford University, Stanford, California.

1974–1975 Programmer, Housing Office, Stanford University. Designed a major software system for assigning students to on-campus housing. With some later improvements, it is still in use.

1975 B.S. (Mathematics) and A.B. (Music), Stanford University.

1977 M.A. (Music), UCLA, Los Angeles, California.

1977–1979 Teaching Assistant, Computer Science Department, UCLA. Taught sections of Engineering 10 (beginning computer programming course) under direction of Professor Leon Levine. During summer 1979, taught a beginning programming course as part of the Freshman Summer Program.

1979 M.S. (Computer Science), UCLA.

1979–1980 Teaching Assistant, Computer Science Department, UCLA.

1980–1981 Research Assistant, Computer Science Department, UCLA.

1981–present Programmer/Analyst, Computer Science Department, UCLA.

PUBLICATIONS

MADHOUS Reference Manual. Stanford University, Dean of Student Affairs (Residential Education Division), 1978. Technical documentation for the MADHOUS software system used to assign students to on-campus housing.

CHAPTER 1

Introduction

For text, let's use the first words out of the ispell dictionary.

CHAPTER 2

Reconstruction-Computation-Quantization (RCQ): A Paradigm for Low Bit Width LDPC Decoding

2.1 Introduction

Low-Density Parity-Check (LDPC) codes [Gal62] have been implemented broadly, including in NAND flash systems and wireless communication systems. Message passing algorithms such as belief propagation (BP) and Min Sumare utilized in LDPC decoders. In practice, decoders with low message bit widths are desired when considering the limited hardware resources such as area, routing capabilities, and power utilization of FPGAs or ASICs. Unfortunately, low bit width decoders with uniform quantizers typically suffer a large degradation in decoding performance [LT05]. On the other hand, the iterative decoders that allow for the dynamic growth of message magnitudes can achieve improved performance [ZS14a].

LDPC decoders that quantize messages non-uniformly have gained attention because they provide excellent decoding performance with low bit width message representations. One family of non-uniform LDPC decoders use lookup tables (LUTs) to replace the mathematical operations in the check node (CN) unit and/or the variable node (VN) unit. S. K. Planjery *et al.* propose finite alphabet iterative decoders (FAIDs) for regular LDPC codes in [PDD13, DVP13], which optimize a *single* LUT to describe VN input/output behavior. In [PDD13] a FAID is designed to tackle certain trapping sets and hence achieves a lower error floor than BP on the binary symmetric channel (BSC). Xiao *et al.* optimize the parameters of FAID using a recurrent quantized neural network (RQNN) [XVT19, XVT20], and the simulation

results show that RQNN-aided linear FAIDs are capable of surpassing floating-point BP in the waterfall region for regular LDPC codes.

Note that the size of the LUTs in [PDD13, DVP13, XVT19, XVT20] describing VN behavior are an exponential function with respect to node degree. Therefore, these FAIDs can only handle regular LDPC codes with small node degrees. For codes with large node degrees, Kurkoski *et al.* develop a mutual-information-maximization LUT (MIM-LUT) decoder in [RK16], which decomposes a single LUT with multiple inputs into a series of concatenated 2×1 LUTs, each with two inputs and one output. This decomposition makes the number of LUTs linear with respect to node degree, thus significantly reducing the required memory. The MIM-LUT decoder performs lookup operations at both the CNs and VNs. The 3-bit MIM-LUT decoder shows a better FER than floating-point BP over the additive white Gaussian noise (AWGN) channel. As the name suggests, the individual 2×1 LUTs are designed to maximize mutual information [KY14].

Lewandowsky *et al.* use the information bottleneck (IB) machine learning method to design LUTs and propose an IB decoder for regular LDPC codes. As with MIM-LUT, IB decoders also use 2×1 LUTs at both CNs and VNs. Stark *et al.* extend the IB decoding structure to support irregular LDPC codes through the technique of message alignment [SLB18,Sta21]. The IB decoder shows an excellent performance on a 5G LDPC code [SBW20, SWB20]. In order to reduce the memory requirement for LUTs, Meidlinger *et al.* propose the Min-IB decoder, which replaces the LUTs at CNs with label-based min operation [MBB15, MM17, MMB20, GBM18].

Because the decoding requires only simple lookup operations, the LUT-based decoders deliver high throughput. However, the LUT-based decoders require significant memory resources when the LDPC code has large degree nodes and/or the decoder has a large predefined maximum decoding iteration time, where each iteration requires its own LUTs. The huge memory requirement for numerous large LUTs prevents these decoders from being viable options when hardware resources are constrained to a limited number of LUTs.

Lee *et al.* [LT05] propose the mutual information maximization quantized belief propagation (MIM-QBP) decoder which circumvents the memory problem by designing non-uniform quantizers and reconstruction mappings at the nodes. Both VN and CN operations are simple mappings and fixed point additions in MIM-QBP. He *et al.* in [HCM19] show how to systematically design the MIM-QBP parameters for quantizers and reconstruction modules. Wang *et al.* further generalize the MIM-QBP structure and propose a reconstruction-computation-quantization (RCQ) paradigm [WWS20] which allows CNs to implement either the min or boxplus operation.

All of the papers discussed above focus on decoders that use the flooding schedule. The flooding schedule can be preferable when the code length is short. However, in many practical settings such as coding for storage devices where LDPC codes with long block lengths are selected, the flooding schedule requires an unrealistic amount of parallel computation for some typical hardware implementations. Layered decoding [JF05], on the other hand, balances parallel computations and resource utilization for a hardware-friendly implementation that also reduces the number of iterations as compared to a flooding implementation for the same LDPC code.

2.1.1 Contributions

As a primary contribution, this work extends our previous work on RCQ [WWS20] to provide dynamic quantization that changes with each layer of a layered LDPC decoder, as is commonly used with a protograph-based LDPC code. The original RCQ approach [WWS20], which uses the same quantizers and reconstructions for all layers of an iteration, suffers from FER degradation and a high average number of iterations when applied to a layered decoding structure. The novelty and contributions in this chapter are summarized as follows:

- *Layer-specific RCQ Decoding structure.* This chapter proposes the layer-specific RCQ decoding structure. The main difference between the original RCQ of [WWS20] and

the layer-specific RCQ decoder is that layer-specific RCQ designs quantizers and reconstructions for each layer of each iteration. The layer-specific RCQ decoder provides better FER performance and requires a smaller number of iterations than the original RCQ structure with the same bit width. This improvement comes at the cost of an increase in the number of parameters that need to be stored in the hardware.

- *layer-specific RCQ Parameter Design.* This work uses layer-specific discrete density evolution featuring hierarchical dynamic quantization (HDQ) to design the layer-specific RCQ parameters. We refer to this design approach as layer-specific HDQ discrete density evolution. For each layer of each iteration, layer-specific HDQ discrete density evolution separately computes the PMF of the messages. HDQ designs distinct quantizers and reconstructions for each layer of each iteration.

- *FPGA-based RCQ Implementations.* This chapter presents the Lookup Method, the Broadcast Method and the Dribble Method, as alternatives to distribute RCQ parameters efficiently in an FPGA. This chapter verifies the practical resource needs of RCQ through an FPGA implementation of an RCQ decoder using the Broadcast method. Simulation results for a (9472, 8192) quasi-cyclic (QC) LDPC code show that a layer-specific Min SumRCQ decoder with 3-bit messages achieves a more than 10% reduction in LUTs and routed nets and more than a 6% reduction in register usage while maintaining comparable decoding performance, compared to a standard offset Min Sumdecoder with 5-bit messages.

2.1.2 Organization

The remainder of this chapter is organized as follows: Sec. 2.2 introduces the RCQ decoding structure and presents an FPGA implementation of an RCQ decoder. Sec. 2.3 describes HDQ, which is used for channel observation quantization and RCQ parameter design. Sec. **??** shows the design of the layer-specific RCQ decoder. Sec. **??** presents simulation results

including FER and hardware resource requirements. Sec. **??** concludes our work.

2.2 The RCQ Decoding Structure

Message passing algorithms update messages between variable nodes and check nodes in an iterative manner either until a valid codeword is found or the maximum number of iterations I_T is reached. The updating procedure of message passing algorithms contains two steps: 1) computation of the output message, 2) communication of the message to the neighboring node. To reduce the complexity of message passing, the computed message is often quantized before being passed to the neighboring node. We refer to the computed messages as the *internal messages*, and communicated messages passed over the edges of the Tanner graph as *external messages*.

For the uniform quantization decoder, the external messages are simply clipped internal messages, in order for a lower routing complexity. However, When external messages are produced by a uniform quantizer, low bit width external messages can result in an early error floor [ZS14b]. Non-uniform quantizers, on the other hand, address error floor issue by providing larger message magnitude range. Zhang *et al.* design a $q+1$ quasi-uniform LDPC decoder, where 2^q messages are allocated to uniform quantization, and the other 2^q messages correspond to exponentially growing quantization interval lengths [ZS14b]. Thorpe *et al.* introduced a non-uniform quantizer in [LT05]. Their decoder adds a non-uniform quantizer and a reconstruction mapping to the output and input of the hardware implementation of each node unit. This approach delivers excellent decoding performance even with a low external bit width. The RCQ decoder [WWS20] can be seen as a generalization of the decoder introduced in [LT05].

In this section, we provide detailed descriptions of the RCQ decoding structure. Three FPGA implementation methods for realizing the RCQ functionality are also presented.

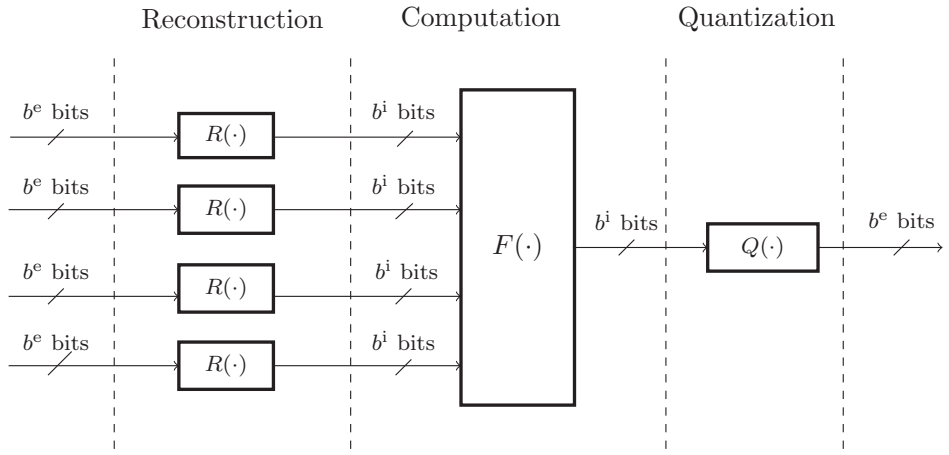


Figure 2.1: Illustration of a generalized RCQ unit which consists of three modules: *Reconstruction* that maps a b^{e}-bit value to a b^{i}-bit value, *Computation* that performs arithmetic operations, and *Quantization* that quantizes a b^{i}-bit value to a b^{e}-bit value.

2.2.1 Generalized RCQ Unit

A generalized RCQ unit as shown in Fig. 2.1 consists of the following three modules:

2.2.1.1 Reconstruction Module

The reconstruction module applies a reconstruction function $R(\cdot)$ to each incoming b^{e}-bit external message to produce a b^{i}-bit internal message, where $b^{\mathrm{i}} > b^{\mathrm{e}}$. We denote the bit width of CN and VN internal message by $b^{\mathrm{i,c}}$ and $b^{\mathrm{i,v}}$, respectively. For the flooding-scheduled RCQ decoder, $R(\cdot)$ is iteration-specific and we use $R_{\mathrm{c}}^{(t)}(\cdot)$ and $R_{\mathrm{v}}^{(t)}(\cdot)$ to represent the reconstruction of check and variable node messages at iteration t, respectively. In the layer-specific RCQ decoder, $R(\cdot)$ uses distinct parameters for each layer in each iteration. We use $R_{\mathrm{c}}^{(t,r)}(\cdot)$ and $R_{\mathrm{v}}^{(t,r)}(\cdot)$ to represent the the reconstruction of check and variable node messages at layer r of iteration t, respectively. The reconstruction functions are mappings of the input external messages to log-likelihood ratios (LLR) that will be used by the node. In this paper, these mappings are systematically designed by HDQ discrete density evolution, which will be

introduced in a later section.

For a quantizer $Q(\cdot)$ that is symmetric, an external message $d \in \mathbb{F}_2^{b^{\mathrm{e}}}$ can be represented as $[d^{\mathrm{MSB}}\ \tilde{d}]$, where $d^{\mathrm{MSB}} \in \{0,1\}$ indicates sign and $\tilde{d} \in \mathbb{F}_2^{b^{\mathrm{e}}-1}$ corresponds to magnitude. We define the magnitude reconstruction function $R^*(\cdot) : \mathbb{F}_2^{b^{\mathrm{e}}-1} \to \mathbb{F}_2^{b^{\mathrm{i}}-1}$, which maps the magnitude of external message, $\tilde{d}$, to the magnitude of internal message. Without loss of generality, we restrict our attention to monotonic reconstruction functions so that

$$R^*(\tilde{d}_1) > R^*(\tilde{d}_2) > 0, \quad \text{for } \tilde{d}_1 > \tilde{d}_2, \tag{2.1}$$

where $\tilde{d}_1$, $\tilde{d}_2 \in \mathbb{F}_2^{b^{\mathrm{e}}-1}$. The reconstruction $R(d)$ can be expressed by $R(d) = \left[d^{\mathrm{MSB}}\ \ R^*(\tilde{d})\right]$. Under the assumption of a symmetric channel, we have $R([0\ \tilde{d}]) = -R([1\ \tilde{d}])$.

2.2.1.2 Computation Module

The computation module $F(\cdot)$ uses the b^{i}-bit outputs of the reconstruction module to compute a b^{i}-bit internal message for the CN or VN output. We denote the computation module implemented in CNs and VNs by F_{c} and F_{v}, respectively. An RCQ decoder implementing the min operation at the CN yields a Min Sum(ms) RCQ decoder. If an RCQ decoder implements belief propagation (bp) via the *boxplus* operation, the decoder is called *bpRCQ*. The computation module, F_{v}, in the VNs is addition for both bpRCQ and msRCQ decoders.

If the RCQ decoder implements the *Min* operation at the check node yielding a MinSum (ms) decoder, i.e.:

$$F_c(h_1, \ldots, h_J) = \prod_j \operatorname{sign}(h_j) \times \min_j |h_j|, \tag{2.2}$$

where $h_j \in \mathbb{F}_2^{b^i}$, $j = 1, ..., J$ are internal messages, then we call the decoder an *msRCQ* decoder.

If an RCQ decoder implements belief propagation (bp) via the *boxplus* operation :

$$F_c(h_1, \ldots, h_J) = h_1 \boxplus h_2 \boxplus ... \boxplus h_J, \tag{2.3}$$

the decoder is called *bpRCQ*. The operator $\boxplus$ is defined as:

$$h_1 \boxplus h_2 = \log\left(\frac{1+e^{h_1+h_2}}{e^{h_1}+e^{h_2}}\right). \tag{2.4}$$

At variable node unit, both msRCQ and bpRCQ decoder sum up all incoming messages:

$$F_v(r_1,\ldots,r_J) = \sum_{j=1}^{J} r_j. \tag{2.5}$$

2.2.1.3 Quantization Module

The quantization module $Q(\cdot)$ quantizes the b^{i}-bit internal message to produce a b^{e}-bit external message. Under the assumption of a symmetric channel, we use a symmetric quantizer that features sign information and a magnitude quantizer $Q^*(\cdot)$. The magnitude quantizer selects one of $2^{b^{\mathrm{e}}-1}-1$ possible indexes using the threshold values $\{\tau_0,\tau_1,...,\tau_{\max}\}$, where $\tau_j \in \mathbb{F}_2^{b^{\mathrm{i}}}$ for $j \in \{0,1,...,2^{b^{\mathrm{e}}-1}-2\}$ and $\tau_{\max}$ is $\tau_{j_{\max}}$ for $j_{\max} = 2^{b^{\mathrm{e}}-1}-2$. We also require

$$\tau_i > \tau_j > 0, \quad i > j. \tag{2.6}$$

Given an internal message $h \in \mathbb{F}_2^{b^{\mathrm{i}}}$, which can be decomposed into sign part h^{MSB} and magnitude part $\tilde{h}$, $Q^*(\tilde{h}) \in \mathbb{F}_2^{b^{\mathrm{e}}-1}$ is defined by:

$$Q^*(\tilde{h}) = \begin{cases} 0, & \tilde{h} \le \tau_0 \\ j, & \tau_{j-1} < \tilde{h} \le \tau_j \\ 2^{b^{\mathrm{e}}-1}-1, & \tilde{h} > \tau_{\max} \end{cases}, \tag{2.7}$$

where $0 < j \le j_{\max}$. Therefore, $Q(h)$ is defined by $Q(h) = [h^{\mathrm{MSB}}\ Q^*(\tilde{h})]$. The super/subscripts introduced for $R(\cdot)$ also apply to $Q(\cdot)$.

2.2.2 Bit Width of RCQ decoder

The three tuple $(b^{\mathrm{e}}, b^{\mathrm{i,c}}, b^{\mathrm{i,v}})$ represents the precision of messages in a RCQ decoder. For the *msRCQ* decoder, it is sufficient to use only the pair $(b^{\mathrm{e}}, b^{\mathrm{i,v}})$ because $b^{\mathrm{i,c}} = b^{\mathrm{e}}$, we simply

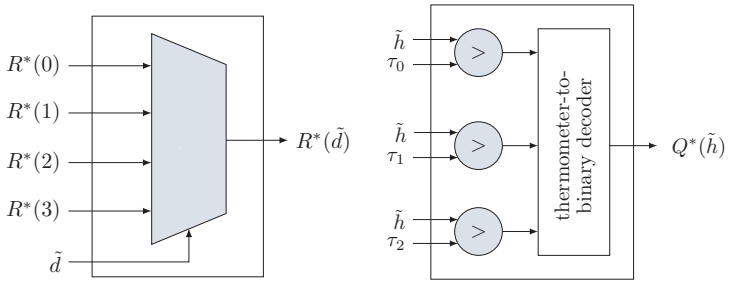
denote $b^{\mathrm{i,v}}$ by b^{v}. The CN min operation computes the XOR of the sign bits and finds the minimum of the extrinsic magnitudes. For a symmetric channel, the min operation can be computed by manipulating the external messages, because the external message delivers the *relative LLR meaning* of reconstructed values. Since we only use external messages to perform the min operation, $R^{\mathrm{c}}(\cdot)$ and $Q^{\mathrm{c}}(\cdot)$ are not needed for the msRCQ decoder. Finally, we use ∞ to denote a floating point representation.

2.2.3 FPGA Implementation for RCQ

The RCQ FPGA decoder may be viewed as a modification to existing hardware decoders based on the BP or MS decoder algorithms, which have been studied extensively [ZDN06, SH16, LZS17, AKK19]. The RCQ decoders require extra $Q(\cdot)$ and $R(\cdot)$ functions to quantize and reconstruct message magnitudes. To implement $Q(\cdot)$ and $R(\cdot)$ functions, we have devised the *Lookup*, *Broadcast*, and *Dribble* methods. These three approaches are functionally identical, but differ in the way that the parameters needed for the $Q(\cdot)$ and $R(\cdot)$ operations are communicated to the nodes.

2.2.3.1 Lookup Method

The quantization and reconstruction functions simply map an input message to an output message. Thus, a simple implementation uses lookup tables implemented using read-only memories (ROMs) to implement all these mappings. As an example, for the iteration-specific magnitude quantizer $Q^{*(t)}(\cdot)$, all iterations can be implemented by a single table indexed by the pair $(\tilde{x}, t)$, where $\tilde{x}$ is the internal message magnitude and t is the current iteration. This index forms an address into a ROM to produce an output $\tilde{y}$. The $Q(\cdot)$ and $R(\cdot)$ functions in every VN require their own ROMs, implemented using block RAMs. If block RAMs with multiple ports are available, then they can be shared by different VN banks to reduce the total amount required. If no ROM sharing occurs, then L VN unit with two ROMs each



(a) (b)

thermometer code	binary form
000	00
001	01
011	10
111	11

(c)

Figure 2.2: msRCQ magnitude reconstruction module (a) and magnitude quantization module (b). In FPGA, magnitude reconstruction module is realized by a multiplexer, and magnitude quantization is realized by comparison functions and a thermometer-to-binary decoder which realizes the mapping relationship shown in (c).

results in a total of $2L$ additional block RAMs used. This amount can be reduced with ROM sharing and other synthesis techniques. Because $Q(\cdot)$ and $R(\cdot)$ change with respect to different iterations and/or layers, one potential drawback of the Lookup method is a large block RAM requirement.

2.2.3.2 Broadcast Method

The Broadcast method provides a scheme where all RCQ parameters are stored centrally in a control unit, instead of being stored in each VN. As an example, for the layered RCQ decoder

whose parameters update every layer and iteration, the pair (t, m), which corresponds to the current iteration and current layer, is used to index into ROMs in the control unit These ROMs output quantization thresholds $\{\tau_0^{(t,m)}, \tau_1^{(t,m)}, \ldots, \tau_{\max}^{(t,m)}\}$ and reconstruction values $\{R^{(i,l)}(0), R^{(t,m)}(1), \ldots, R^{(t,m)}(2^{b^c-1}-1)\}$, which are wired to the VN units. The $Q(\cdot)$ and $R(\cdot)$ blocks in the VN units only take in the parameters for each decoding iteration and layer, and use logic to perform their respective operations. Each VN only takes in the $Q(\cdot)$ and $R(\cdot)$ parameters necessary for decoding the current iteration and layer, and use logic to perform their respective operations. Fig. 2.2 shows an implementation for a 3-bit RCQ, which uses mere 2 bits for magnitude reconstruction and quantization. The 2-bit magnitude reconstruction module is realized by a 4×1 multiplexer. The 2-bit magnitude quantization consists of two steps, first a thermometer code [AV18], where the contiguous ones are analogous to mercury in a thermometer, is generated by comparing the input with all thresholds, and then the thermometer code is converted to the 2-bit binary form by using a thermometer-to-binary decoder, which realizes the mapping relationship in Fig. 2.2c. Two block RAMS are required in the control unit for the thresholds and reconstruction values. Small LUTs in each VN implement the $Q(\cdot)$ and $R(\cdot)$ functions. The main penalty of the Broadcast method is the additional wiring necessary to route the RCQ parameters from the central control unit to the VNs.

The main penalty of the Broadcast is the additional wiring necessary to route the L-msRCQ parameters from the control unit to VN banks. If w bits are used for each of the thresholds and reconstruction values of 3-bit L-msRCQ, a total of $7w$ additional wires need to be routed to each VN unit, w wires for each of the three thresholds and each of the four reconstruction values. With L VN units, the total amount of added routes is $7wL$. For a 4-bit L-msRCQ decoder, the total increase is $15wL$. The same parameters are routed to all the VN units. Thus shared wiring may be used in some cases.

2.2.3.3 Dribble Method

The Dribble method attempts to reduce the number of long wires required by the Broadcast method. Registers in the VNs save the current thresholds and reconstruction values necessary for the $Q(\cdot)$ and $R(\cdot)$ functions. Once again, quantization and reconstruction can be implemented using the logic in Fig. 2.2. When a new set of parameters is required, the bits are transferred (dribbled) one by one or in small batches from the control unit to the VN unit registers. Just as in the Broadcast method, two extra block RAMs and logic for the $Q(\cdot)$ and $R(\cdot)$ functions are required. But where the Broadcast method needs $7w$ additional wires routed to each VN bank for 3-bit L-msRCQ, the Dribble method requires only as many wires as the transfer batch size. The penalty of the Dribble method comes with the extra usage of registers in the VN units. A total of $7w$ bits stored in registers would be necessary in each VN bank to save the current threshold and reconstruction values for 3-bit L-msRCQ. In total, $7wL$ bits of register storage would be used for 3-bit L-msRCQ, and $15wL$ bits would be necessary for 4-bit L-msRCQ. This total can be reduced by having multiple VN units share sets of registers. We have implemented all methods and explored their resource utilization in [TWC21].

2.3 Hierarchical Dynamic Quantization (HDQ)

This section introduces the HDQ algorithm, a non-uniform quantization scheme that this paper uses both for quantization of channel observations and for quantization of internal messages by RCQ. Our results show, for example, that HDQ quantization of AWGN channel observations achieves performance similar to the optimal dynamic programming quantizer of [KY14] for the binary input AWGN channel, with much lower computational complexity.

2.3.1 Motivation

The quantizer plays an important role in RCQ decoder design. First, the channel observation is quantized as the input to the decoder. This section explores how to use HDQ to quantize the channel observations. Second, the parameters of $R(\cdot)$ and $Q(\cdot)$ are also designed by quantizing external messages according to their probability mass function (PMF) as determined by discrete density evolution. The use of HDQ to quantize internal messages is described in Section **??**.

The HDQ approach designs a quantizer that maximizes mutual information in a greedy or progressive fashion. Quantizers aiming to maximize mutual information are widely used in non-uniform quantization design [HCM19, WWS20, LB18, SLB18, SBW20, MBB15, MMB20, MM17, SWB20, GBM18, WLW19, WCS11, WVC14]. Due to the interest of this paper, the cardinality of quantizer output is restricted to 2^b, i.e., this paper seeks b-bit quantizers. Kurkoski and Yagi [TV13] proposed a dynamic programming method to find an optimal quantizer that maximizes mutual information for a binary input discrete memoryless channel (BI-DMC) whose outputs are from an alphabet with cardinality B, with complexity $\mathcal{O}(B^3)$. The dynamic programming method of [KY14] finds the optimal quantization, but the approach becomes impractical when B is large.

In order to quantize the outputs for a channel with large cardinality B when constructing polar codes, Tal and Vardy devised a sub-optimal greedy quantization algorithm with complexity $\mathcal{O}(B\log(B))$ [TV13]. In [LB18], Lewandowsky *et al.* proposed the modified Sequential Information Bottleneck (mSIB) algorithm to design the channel quantizer and LUTs for LDPC decoders. mSIB is also a sub-optimal quantization technique with complexity $\mathcal{O}(aB)$, where a is the number of trials. As a machine learning algorithm, multiple trials are required for good results with mSIB. Typical values of a range, for example, from 15 to 70.

HDQ is proposed in [WWS20] as an efficient b-bit quantization algorithm for the sym-

metric BI-DMC with complexity $\mathcal{O}\left(\frac{2^b}{\log(\gamma)}\log(B)\right)$. HDQ has less complexity than mSIB and also the Tal-Vardy algorithm. This section reviews the HDQ using symmetric binary input AWGN channel as an example. As an improvement to the HDQ of [WWS20], sequential threshold search is replaced with golden section search [Kie53].

2.3.2 The HDQ Algorithm

Let the encoded bit $x \in \{0, 1\}$ be modulated by Binary Phase Shift Keying (BPSK) and transmitted over an AWGN channel. The modulated BPSK signal is represented as $s(x) = -2x + 1$. We denote the channel observation at the receiver by y where

$$y = s(x) + z, \tag{2.8}$$

and $z \sim \mathcal{N}(0, \sigma^2)$. The joint probability density function of x and y, $f(x, y; \sigma)$, is:

$$f(x, y; \sigma) = \frac{1}{2\sqrt{2\pi\sigma^2}} e^{-\frac{(y-s(x))^2}{2\sigma^2}}. \tag{2.9}$$

HDQ seeks an b-bit quantization of the continuous channel output y, as in [WCS11]. In practice, often y is first quantized into B values using high-precision uniform quantization where $B \gg 2^b$, i.e., analog-to-digital (A/D) conversion. Let W be the result of the A/D output, where $W \in \mathcal{W}$ and $\mathcal{W} = \{0, 1, ..., B-1\}$. The alphabet of B channel outputs from the A/D converter is then subjected to further non-uniform quantization resulting in a quantization alphabet of 2^b values. We use D to represent the non-uniform quantizer output, which is comprised of the b bits $D = [D_1, ..., D_b]$. HDQ aims to maximize the mutual information between X and D.

For the symmetric binary input AWGN channel, a larger index w implies a larger LLR, i.e.:

$$\log \frac{P_{W|X}(i|0)}{P_{W|X}(i|1)} < \log \frac{P_{W|X}(j|0)}{P_{W|X}(j|1)}, \ \forall i < j. \tag{2.10}$$

Based on Lemma 3 in [KY14], any binary-input discrete memoryless channel that satisfies (2.10) has an optimal b-bit quantizer that is determined by $2^b - 1$ boundaries, which can be

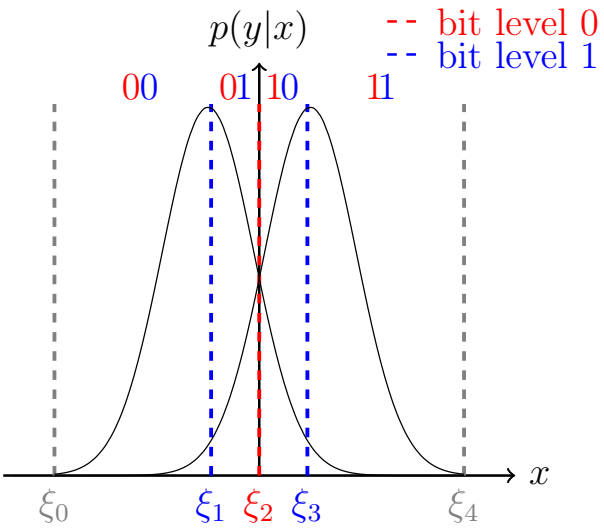


Figure 2.3: Given the conditional probability $p(y|x)$ of symmetric BI-AWGN channel, HDQ sequentially quantizing A/D output w into a 2-bit message by first finding the index ξ_2, then the indices ξ_1 and ξ_3.

identified by their corresponding index values. Denote the $2^b - 1$ index thresholds by $\{\xi_1, \xi_2, ..., \xi_{2^b-1}\} \subset \mathcal{W}$. Unlike the dynamic programming algorithm [KY14], which optimizes boundaries jointly, HDQ *sequentially* finds thresholds according to *bit level*, similar to the progressive quantization in [WLW19].

HDQ quantizes the symmetric BI-AWGN channel output using a progressive [WLW19] or greedy approach. The general b-bit HDQ approach is as follows:

1. We assume an initial high-precision uniform quantizer. For this case, set the extreme index thresholds $\xi_0 = 0$ and $\xi_{2^b} = B - 1$, which are the minimum and maximum outputs of the uniform quantization.

2. The index threshold $\xi_{2^{(b-1)}}$ is selected as follows to determine the bit level 0:

$$\xi_{2^{(b-1)}} = \arg \max_{\xi_0 < \xi < \xi_{2^b}} I(X; D_1)\,, \tag{2.11}$$

Algorithm 1: Hierarchical Dynamic Quantization

input : $P(X,W), X \in \{0,1\}, W \in \{0,...,B-1\}$; b

output: $\{\xi_0, \xi_1, ..., \xi_{2^b-1}\}$, $P(X,T)$

$\xi_0 \leftarrow 0$, $\xi_{2^b} \leftarrow B-1$

for $i \leftarrow 0$ **to** $b-1$ **do**

  for $j \leftarrow 0$ **to** $2^i - 1$ **do**

    $\xi_{(j+0.5)2^{b-i}} = \texttt{GSS}(\xi_{j2^{b-i}}, \xi_{(j+1)2^{b-i}})$

  end

end

$P_{XT}(x,t) = \sum_{w=\xi_t}^{\xi_t+1} P_{XW}(x,w)$, $X \in \{0,1\}$, $T \in \{0,...,2^{b-1}\}$

where

$$D_1 = \mathbb{1}(W \geq \xi_2^{(b-1)}). \tag{2.12}$$

3. The index thresholds $\xi_{2^{(b-2)}}$ and $\xi_{3*2^{(b-2)}}$ are selected as follows to determine bit level 1:

$$\xi_{2^{(b-2)}} = \arg \max_{\xi_0 < \xi < \xi_{2^b-1}} I(X; D_2 | D_1 = 0), \tag{2.13}$$

$$\xi_{3*2^{(b-2)}} = \arg \max_{\xi_{2^b-1} < \xi < \xi_{2^b}} I(X; D_2 | D_1 = 1), \tag{2.14}$$

and

$$D_2 = \begin{cases} \mathbb{1}(W \geq \xi_{2^{(b-2)}}) & \text{if } D_1 = 0 \\ \mathbb{1}(W \geq \xi_{3*2^{(b-2)}}) & \text{if } D_1 = 1 \end{cases}. \tag{2.15}$$

4. In the general case, when the thresholds for k previous quantization bits have been determined, 2^k thresholds $\{\xi_{(j+0.5)2^{b-k}}, j = 0,..,2^k-1\}$ must be selected to determine the next quantization bit. Each threshold maximizes $I(X; D_{k+1} | D_k = d_k, \ldots, D_1 = d_1)$ for a specific result for the k previous quantization bits.

Fig. 2.3 illustrates how HDQ quantizes the symmetric binary input AWGN for the case where $b = 2$. First, the indices ξ_0 and ξ_4 of the extreme points are set. Then the

index ξ_2 is set to maximize $I(X; D_1)$. Finally, the indices ξ_1 and ξ_3 are set to maximize $I(X; D_2|D_1)$ by independently selecting ξ_1 to maximize $I(X; D_2|D_1 = 0)$ and ξ_3 to maximize $I(X; D_2|D_1 = 1)$.

Alg. 1 provides a full description of HDQ algorithm. The function $\texttt{GSS}(\xi_\ell, \xi_r)$ uses the golden section search algorithm described in Sec. **??** for thresholds search.

REFERENCES

[AKK19] Rajagopal Anantharaman, Karibasappa Kwadiki, and Vasundara Patel Kerehalli Shankar Rao. "Hardware Implementation Analysis of Min-Sum Decoders." *Advances in Electrical and Electronic Engineering*, **17**(2):179–186, June 2019.

[AV18] M P Ajanya and George Tom Varghese. "Thermometer code to Binary code Converter for Flash ADC - A Review." In *2018 Inter. Conf. on Control, Power, Comm. and Comp. Tech. (ICCPCCT)*, pp. 502–505, March 2018.

[DVP13] D Declercq, B Vasic, S K Planjery, and E Li. "Finite Alphabet Iterative Decoders—Part II: Towards Guaranteed Error Correction of LDPC Codes via Iterative Decoder Diversity." *IEEE Trans. Comm.*, **61**(10):4046–4057, October 2013.

[Gal62] Robert G. Gallager. "Low-Density Parity-Check Codes." *IRE Trans. on Info. Theo.*, **8**(1):21–28, January 1962.

[GBM18] R Ghanaatian, A Balatsoukas-Stimming, T C Müller, M Meidlinger, G Matz, A Teman, and A Burg. "A 588-Gb/s LDPC Decoder Based on Finite-Alphabet Message Passing." *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, **26**(2):329–340, February 2018.

[HCM19] X He, K Cai, and Z Mei. "On Mutual Information-Maximizing Quantized Belief Propagation Decoding of LDPC Codes." In *2019 IEEE Global Comm. Conf. (GLOBECOM)*, pp. 1–6, December 2019.

[JF05] J. Zhang and M. P. C. Fossorier. "Shuffled iterative decoding." *IEEE Trans. on Comm.*, **53**(2):209–213, 2005.

[Kie53] J Kiefer. "Sequential Minimax Search for a Maximum." *Proc. Am. Math. Soc.*, **4**(3):502–506, 1953.

[KY14] B M Kurkoski and H Yagi. "Quantization of Binary-Input Discrete Memoryless Channels." *IEEE Trans. Inf. Theory*, **60**(8):4544–4552, August 2014.

[LB18] J Lewandowsky and G Bauch. "Information-Optimum LDPC Decoders Based on the Information Bottleneck Method." *IEEE Access*, **6**:4054–4071, 2018.

[LT05] J K Lee and J Thorpe. "Memory-efficient decoding of LDPC codes." In *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, pp. 459–463, September 2005.

[LZS17] Yanhuan Liu, Chun Zhang, Pengcheng Song, and Hanjun Jiang. "A high-performance FPGA-based LDPC decoder for solid-state drives." In *2017 IEEE*

60th Inter. Midwest Symp. on Circuits and Systems (MWSCAS), pp. 1232–1235, August 2017.

[MBB15] M Meidlinger, A Balatsoukas-Stimming, A Burg, and G Matz. "Quantized message passing for LDPC codes." In *2015 49th Asilomar Conference on Signals, Systems and Computers*, pp. 1606–1610, November 2015.

[MM17] M Meidlinger and G Matz. "On irregular LDPC codes with quantized message passing decoding." In *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, July 2017.

[MMB20] M Meidlinger, G Matz, and A Burg. "Design and Decoding of Irregular LDPC Codes Based on Discrete Message Passing." *IEEE Trans. Commun.*, **68**(3):1329–1343, March 2020.

[PDD13] S K Planjery, D Declercq, L Danjean, and B Vasic. "Finite Alphabet Iterative Decoders—Part I: Decoding Beyond Belief Propagation on the Binary Symmetric Channel." *IEEE Trans. Comm.*, **61**(10):4033–4045, October 2013.

[RK16] F J C Romero and B M Kurkoski. "LDPC Decoding Mappings That Maximize Mutual Information." *IEEE J. Sel. Areas Commun.*, **34**(9):2391–2401, September 2016.

[SBW20] M Stark, G Bauch, L Wang, and R D Wesel. "Information Bottleneck Decoding of Rate-Compatible 5G-LDPC Codes." In *ICC 2020 - 2020 IEEE Inter. Conf. on Comm. (ICC)*, pp. 1–6, June 2020.

[SH16] Ahmed M Sadek and Aziza I Hussein. "Flexible FPGA implementation of Min-Sum decoding algorithm for regular LDPC codes." In *2016 11th Inter. Conf. on Comp. Engi. Systems (ICCES)*, pp. 286–292, December 2016.

[SLB18] M Stark, J Lewandowsky, and G Bauch. "Information-Optimum LDPC Decoders with Message Alignment for Irregular Codes." In *2018 IEEE Glob. Comm. Conf. (GLOBECOM)*, pp. 1–6, December 2018.

[Sta21] Maximilian Stark. *Machine learning for reliable communication under coarse quantization*. PhD thesis, Technische Universität Hamburg, 2021.

[SWB20] M Stark, L Wang, G Bauch, and R D Wesel. "Decoding Rate-Compatible 5G-LDPC Codes With Coarse Quantization Using the Information Bottleneck Method." *IEEE Open Journal of the Communications Society*, **1**:646–660, 2020.

[TV13] I. Tal and A. Vardy. "How to Construct Polar Codes." *IEEE Trans. on Info. Theo.*, **59**(10):6562–6582, 2013.

[TWC21] Caleb Terrill, Linfang Wang, Sean Chen, Chester Hulse, Calvin Kuo, Richard Wesel, and Dariush Divsalar. "FPGA Implementations of Layered MinSum LDPC Decoders Using RCQ Message Passing." *arXiv preprint arXiv:2104.09480*, 2021.

[WCS11] Jiadong Wang, Thomas Courtade, Hari Shankar, and Richard D. Wesel. "Soft Information for LDPC Decoding in Flash: Mutual-Information Optimized Quantization." In *2011 IEEE Glob. Tele. Conf. (GLOBECOM)*, pp. 1–6, 2011.

[WLW19] Nathan Wong, Ethan Liang, Haobo Wang, Sudarsan V. S. Ranganathan, and Richard D. Wesel. "Decoding Flash Memory with Progressive Reads and Independent vs. Joint Encoding of Bits in a Cell." In *2019 IEEE Glob. Comm. Conf. (GLOBECOM)*, pp. 1–6, 2019.

[WVC14] Jiadong Wang, Kasra Vakilinia, Tsung Chen, Thomas Courtade, Guiqiang Dong, Tong Zhang, Hari Shankar, and Richard Wesel. "Enhanced Precision Through Multiple Reads for LDPC Decoding in Flash Memories." *IEEE J. Sel. Areas in Comm.*, **32**(5):880–891, 2014.

[WWS20] L Wang, R D Wesel, M Stark, and G Bauch. "A Reconstruction-Computation-Quantization (RCQ) Approach to Node Operations in LDPC Decoding." In *GLOBECOM 2020 - 2020 IEEE Glob. Comm. Conf.*, pp. 1–6, December 2020.

[XVT19] X Xiao, B Vasic, R Tandon, and S Lin. "Finite Alphabet Iterative Decoding of LDPC Codes with Coarsely Quantized Neural Networks." In *2019 IEEE Glob. Comm. Conf. (GLOBECOM)*, pp. 1–6, December 2019.

[XVT20] X Xiao, B Vasić, R Tandon, and S Lin. "Designing Finite Alphabet Iterative Decoders of LDPC Codes Via Recurrent Quantized Neural Networks." *IEEE Trans. Commun.*, **68**(7):3963–3974, July 2020.

[ZDN06] Zhengya Zhang, Lara Dolecek, Borivoje Nikolic, Venkat Anantharam, and Martin Wainwright. "GEN03-6: Investigation of Error Floors of Structured Low-Density Parity-Check Codes by Hardware Emulation." In *2006 IEEE Glob. Comm. Conf. (Globecom)*, pp. 1–6, 2006.

[ZS14a] X Zhang and P H Siegel. "Quantized Iterative Message Passing Decoders with Low Error Floor for LDPC Codes." *IEEE Trans. Commun.*, **62**(1):1–14, January 2014.

[ZS14b] X. Zhang and P. H. Siegel. "Quantized Iterative Message Passing Decoders with Low Error Floor for LDPC Codes." *IEEE Trans. on Comm.*, **62**(1):1–14, 2014.