Memory Efficient EMS Decoding for Non-Binary LDPC Codes

Leixin Zhou, Jin Sha
School of Electronic Science and Engineering
Nanjing University
Nanjing, China
llasionzhou@gmail.com; shajin@nju.edu.cn

Yun Chen
State Key Lab. of ASIC and System
Fudan University
Shanghai, China
chenyun@fudan.edu.cn

Zhongfeng Wang
Broadcom Corporation
Irvine, California, USA
zfwang@broadcom.com

Abstract—Non-binary low-density parity-check (NB-LDPC) codes are an extension of binary LDPC codes with significantly better performance when the code length is moderate. Previously, forward-backward schemes are used to implement check node processing, which need large amount of memory. In this paper, a novel approach-TCL-EMS is proposed for NB-LDPC decoding. Compared to original EMS decoding algorithm, the memory efficiency is improved and the average number of iterations is reduced significantly. Also, the overall decoder architecture is proposed.

I. INTRODUCTION

BINARY low-density parity-check (LDPC) codes, discovered by Gallager in 1962 [1], were rediscovered and shown to approach Shannon capacity in the late 1990s. Non-binary LDPC (NB-LDPC) codes, viewed as an extension of the binary ones, were first investigated by Davey and MacKay [2]. NB-LDPC codes have performance advantage over binary ones for channel impairments that corrupt multiple consecutive bits, since they are capable of correcting symbol-wise errors [3]. However, the performance gain brought by using non-binary LDPC codes comes together with a significant increase of the decoding complexity.

Lots of research efforts have already been spent on studying the efficient decoding algorithms for non-binary LDPC codes. Non-binary LDPC codes can be decoded with message passing algorithms as the extended sum-product algorithm (SPA) [2]. The computational complexity of the SPA is dominated by \( \Theta(q^2) \) sum and product operations for each check node processing, where \( q \) is the cardinality of the Galois field. Frequency-domain [4], log-domain [5], and mixed-domain SPA [6] have been proposed to reduce the complexity. In the frequency domain, the complexity will be dominated by \( \Theta \left( \frac{q^2}{\log_q q} \right) \) sum and product operations for each check node processing. However, Fourier transforms and a large number of multiplications are required. Log-domain SPA is less sensitive to quantization noise, and can convert multiplications to additions. Nevertheless, it is not easy to compute the Fourier transform in the log domain. Although the mixed-domain SPA can take advantage of both domains, it requires large look-up tables for the domain conversion. Furthermore, all these three algorithms need to keep all \( q \) messages on each edge of the Tanner graph.

Similar to the min-sum decoding algorithm of binary LDPC codes, a log-domain extended min-sum (EMS) algorithm has been proposed in [7] as an approximation of the non-binary SPA. This algorithm can reduce message memory requirement with little performance loss by keeping only the \( s_m < q \) most reliable messages at the input of the check node. To further reduce the computation complexity, the SUM in the check node processing is replaced by MAX in Min-Max algorithm [8]. In both the EMS and Min-Max decoding, the forward-backward (FB) scheme [5] is employed for the check node processing with the cost of extra memory for the intermediate results and long latency. Recently, based on path construction, an efficient trellis-min-max decoding algorithm is proposed in [9] [10].

In this paper, based on EMS, a truncated column layered EMS decoding (TCL-EMS) scheme is proposed to reduce the average number of iterations and improve memory efficiency. Meanwhile, an overall architecture for the TCL-EMS is also proposed. The rest of this paper is organized as follows: Section II introduces the original EMS decoding algorithm for non-binary LDPC codes as well as the FB scheme. Section III discusses the TCL-EMS decoding algorithm for NB-LDPC codes and presents the simulation results. The overall decoder architecture will be illustrated in Section IV. Finally, some conclusions are drawn in Section V.

II. NON-BINARY LDPC DECODING ALGORITHM

A. EMS Decoding Algorithm

A non-binary LDPC code is defined by a very sparse parity check matrix \( H \) whose components belong to a finite field \( GF(q) \). Matrix \( H \) has \( M \) rows and \( N \) columns. The non-zero elements of \( H \) are denoted \( h_{m,n} \). The codewords of a LDPC code are elements of the null space of \( H \), i.e., is a codeword iff \( cH^T = 0 \). Decoding algorithms of LDPC codes are iterative message passing based on a Tanner graph representation of the matrix \( H \).
In the EMS algorithm, a message is a vector of $q$ sub-messages. Let $x_j$ be the $j$-th code symbol of the code word. Let $\lambda_j = [\lambda_j(0), \lambda_j(1), \ldots, \lambda_j(q - 1)]$ be the a priori channel information for $x_j$. The sub-message $\lambda_j$ is a log-likelihood ratio (LLR) defined as $\lambda_j(d) = \log(\text{Prob}(x_j = z_j)/\text{Prob}(x_j = d_j))$, where $z_j$ is the most likely (ML) symbol for $x_j$. We denote $\alpha_{i,j}$ and $\beta_{i,j}$ as the V2C and C2V soft messages passed between the $i$-th check node (CN) and $j$-th variable node (VN) respectively. Let $x'_{i,j} = h_{i,j} \otimes x_i$. For the $i$-th CN, let the configuration $\mathcal{L}_i(x'_{i,j} = d_i)$ be the sequence such that $x'_{i,j} = d_i$ and the $i$-th check-sum is satisfied. Define an $s$-truncated configuration such for each $j' \in N_i \setminus \{j\}$, $\alpha_{i,j}(x'_{i,j'})$ is of the $s$ smallest sub-messages of $\alpha_{i,j}$ over all $d_i \in GF(q)$. Let $\mathcal{L}_i(x'_{i,j} = d_i)$ be the set of the $s$-truncated configurations. Taking $s < q$ requires additional operations called message truncation, in which the sub-messages are sorted with the $q - s$ largest ones ignored. Let $\kappa$ and $\kappa_{\text{max}}$ denote the iteration counter and the maximum number of iterations respectively. The pseudocode for EMS is shown in Algorithm 1.

**Algorithm 1** The Original EMS Decoding Algorithm

1: Initialization:
2: Set $z_j = \arg\min_{d\in GF(q)} \lambda_j(d)$. For all $i,j$ with $h_{i,j} \neq 0$, set $\alpha_{i,j}(h_{i,j} \otimes d) = \lambda_j(d)$. Set $\kappa = 0$
3: Main Loop:
4: Step 1: Compute the syndrome of $z \otimes H^T$. If $z \otimes H^T = 0$, stop decoding and output $z$ as the decoded codeword; otherwise go to Step 2
5: Step 2: If $\kappa = \kappa_{\text{max}}$, stop decoding and declare a decoding failure; otherwise, go to Step 3;
6: Step 3: With a proper scaling factor $0 < c \leq 1$ and $1 \leq s \leq q$, compute C2V messages by:
7: $\beta_{i,j}(d) = c \times \min_{i' \in M_j} \{ \alpha_{i',j}(x'_{i,j'}) | x'_{i,j'} = d_i\}$
8: Step 4: Set $\kappa \leftarrow \kappa + 1$. Compute the primitive messages by:
9: $\hat{\alpha}_{i,j}(h_{i,j} \otimes d) = \lambda_j(d) + \sum_{i' \in M_j} \beta_{i',j}(h_{i',j} \otimes d)$
10: Step 5: $\alpha_{i,j} = \arg \min_{d \in GF(q)} \hat{\alpha}_{i,j}(d)$
11: $\alpha_{i,j}(d) = \hat{\alpha}_{i,j}(d) - \hat{\alpha}_{i,j}(a_{i,j})$
12: Step 6: $\lambda_j(d) = \lambda_j(d) + \sum_{i \in M_j} \beta_{i,j}(h_{i,j} \otimes d)$
13: $z_j = \arg \min_{d \in GF(q)} \lambda_j(d)$
14: Go to Step 1

When we take $s < q$, the original EMS is denoted to be the truncated-EMS (T-EMS) algorithm.

**B. Forward-Backward Scheme**

The forward-backward scheme (FB) is widely used in the check node processing for kinds of NB-LDPC decoding algorithm [5] [8] [11]. The check node degree is denoted by $d_c$. Let $F_j$ and $B_j$ denote the intermediate message vector generated by the forward and backward computation, respectively. Meanwhile, $a$, $a'$ and $a''$ are possible code symbols. The FB contains three operations: forward, backward and merge, which are present in Algorithm 2.

**Algorithm 2** The FB Scheme

1: Forward computation:
2: $F_1(a) = \alpha_{m,n_1}(a)$
3: $F_j(a) = \max_{a' + a'' = a} \{ \text{Sum}(F_{j-1}(a'), \alpha_{m,n_j}(a'')) \}$
4: Backward computation:
5: $B_d(a) = \alpha_{m,n_d}(a)$
6: $B_j(a) = \max_{a' + a'' = a} \{ \text{Sum}(B_{j+1}(a'), \alpha_{m,n_j}(a'')) \}$
7: Merge computation:
8: $\beta_{m,n_1} = B_2(a), \beta_{m,n_d} = F_{d-1}$
9: $\beta_{m,n_j} = \max_{a' + a'' = a} \{ \text{Sum}(B_{j+1}(a'), F_{j-1}(a'')) \}$

### III. TCL-EMS DECODING ALGORITHM FOR NB-LDPC CODES

**A. Proposed TCL-EMS Decoding Scheme**

The design in [12] utilized column-layered decoding algorithm to reduce the average of iterations in decoding of binary LDPC codes. In this paper, based on FB, we propose a truncated column layered EMS (TCL-EMS) decoding algorithm to reduce both storage requirement and average number of iterations.

In [13], the authors present many high performance non-binary quasi-cyclic LDPC codes (NB-QCLDPC). The parity-check matrix included in [13] can be divided into $G$ groups and each group has the same number of columns. Meanwhile, there exists at most one none-zero symbol for each row in a group.

Using codes in [13], we present the proposed column layered (CL) scheme in Algorithm 3. The performance simulation under BPSK modulation and AWGN channel for the proposed algorithm is shown in Fig.1. The code used is a $(6, 3)$ $(620, 310)$ NB-LDPC code generated over $GF(32)$ [13]. The $S$ is set to 16 and the maximum number of iteration is set to 50. The EMM means in MIN-MAX decoding, only the $S$ most reliable messages are used, just like the T-EMS. In addition, FP means floating point. The scaling factor of TCL-EMS FP and T-EMS is applied to the MIN-MAX, the performance is better than the original T-EMS. Moreover, when the CL scheme is applied to the MIN-MAX, the performance is better than the counterpart with two-phase schedule. However, when the FER is less than $2 \times 10^{-4}$, there appears an error floor.

**B. Quantization Selection**

Based on the simulation results, to get a good trade-off between memory consumption and performance, it is reasonable to select the 4-bit quantization scheme, the decoding gain of which is about 0.2 dB less than the EMS(FP)
Algorithm 3 The Proposed Column Layered Scheme

1: Initialization:
2: for each $g \in [1, G-1]$ do
3:    compute all $F_{j,m}$, $j \in \{1, 2, ..., d_c - 1\}$
4: end for
5: Main Loop:
6: for each $i \in [1, max_iter]$ do
7:    if $i$ is odd then
8:        for $g = G$ to 2 do
9:            update C2V messages in group $g$
10:           update V2C messages in group $g$
11:           compute $B_{j(g),m,j} \in \{2, 3, ..., d_c\}$
12:             end for
13:    else
14:        for $g = 1$ to $G - 1$ do
15:            update C2V messages in group $g$
16:           update V2C messages in group $g$
17:           compute $F_{j(g),m,j} \in \{1, 2, ..., d_c - 1\}$
18:             end for
19:     end if
20:     if the decoded codeword is valid then
21:         Quit
22: end if
23: end for
24: † Note: all $m \in \{1, 2, ..., M\}$

when the FER is about $10^{-5}$ and meanwhile the performance of which is almost as good as TCL-EMS(FP) and is much better than T-EMS. Because only $q/2$ messages are stored and meanwhile each message only needs 4 bits, lots of memory will be saved with acceptable performance loss. From our analysis, our algorithm is able to improve memory efficiency by 27% [11] and 86% [14], respectively.

IV. DECODER DESIGN FOR TCL-EMS ALGORITHM

A. Top Level Decoder Architecture

In this section, we present the overall architecture of the decoder for TCL-EMS. The top level decoder architecture for TCL-EMS algorithm is shown in Fig.2. Since the parity check matrix (PCM) is constructed from the circulate permutation matrix (CPM) by shifting, here we can assume its size is $k \times k$. Also the size of PCM is set to $m \times n$ over $GF(q)$. In addition, we assume the number of quantization bits $w$.

The TCL-EMS decoding is applied as follows. At the initial step, the a priori information is loaded into the LLR memory from channel. During the decoding, $k$ columns in one group are processed in parallel. The corresponding $p$ check nodes processing units (CNU) will read messages from V2C memory and update the C2V messages of one group. These updated C2V messages will be permuted and then loaded by $k$ variable node processing units (VNU). The message for tentative decision are generated by adding $\lambda_j(d)$ to all $\hat{a}_{i,j}(h_{i,j} \otimes d)$, while the V2C messages can be generated by normalizing and scaling the results of subtracting $\lambda_j(d)$ from tentative messages. The updated V2C messages of one group will be written to the V2C message memory and used right away in the CNUs for the next group.

Because of the characters of the column layered decoding, there are $k \times d_c$ CNUs and $k$ VNUs. Besides the V2C message memory and the LLR memory, there still exists a block of intermediate memory in each CNU. Messages of each symbol occupy $s \times (w + \log_q a)$ bits. The LLR memory stores a posteriori messages $L_{av}$. Therefore the size of LLR memory is $n \times q \times w$ bits. The V2C memory contains $k \times s \times (w + \log_q a)$ bits.

B. CNU Architecture

The VNU can be easily extended from the VNU architecture in [11], and all the change we need is the adding of a scaling block to scale the V2C messages.

The elementary step in CNUs, which is implemented by ECU (elementary computation unit), has two input messages and one output message. Following the strategy and ECU architecture proposed by [11], which provides a minimum number of operations to process the sorted values of the output vector $V$, the main component is a sorter of size $s$, which is
used to fill in the output message. The algorithm starts with an initialization loop which consists in inserting into the sorter a first list of values corresponding to the first input message. Then, the algorithm walks through the s largest values among all the \( s^2 \) combinations.

The CNU architecture is shown in Fig. 3. The CNU processes messages serially. Each FB memory module, which contains \( d_r - 1 \) blocks, denote the intermediate message memory for FB computation. \( F_B^j \) memory block stores \( F_j \) at the beginning of odd iteration, and then filled by updated \( B_{ij} + 1 \). In the even iteration, the case is contrary. The control signal \( S_0 \) and \( S_1 \) are used to choose the two input message vector of ECU. When the ECU performs forward or backward computation, the intermediate results will be sent to corresponding \( F_B^j \) memory block. The control signal \( S_2 \) are used to select \( F_B^j \) block to store the intermediate computation results of ECU. \( \beta_{m,n} \) will be computed in the merge step and sent to permutation network. It should be noted that \( F_B^j \) memory block needs to store immediate messages for \( p \) rows because of the characters of the CL processing. For a \((620, 310),(6, 3)\) QC-NBLDPC code over \( GF(32) \), they store forward/backward messages for 3 rows. Therefore the size of each memory block is \( 3 \times s \times (w + \log_2 q) \)-bits.

C. Comparison with Other Implementation Schemes

It would be meaningful to compare the proposed scheme with others’ schemes.

For the \((620, 310),(6, 3)\) QC-NBLDPC code over \( GF(32) \), our implementation scheme will reduce the storage requirement of intermediate messages for FBC and the message memory. For fair comparison, we define memory efficiency (ME) as the ratio of parallelism and total memory consumption. The detailed comparison results are listed in Table I. Decoders in [11] [14] both adopt 5-bits quantization. It is apparent that our TCL-EMS algorithm has much better memory efficiency.

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Consumption and Efficiency of Different Schemes.</td>
</tr>
<tr>
<td>proposed</td>
</tr>
<tr>
<td>LLR memory(bit)</td>
</tr>
<tr>
<td>Message memory(bit)</td>
</tr>
<tr>
<td>FBC memory(bit)</td>
</tr>
<tr>
<td>Total memory(bit)</td>
</tr>
<tr>
<td>Parallelism</td>
</tr>
<tr>
<td>Normalized ME</td>
</tr>
</tbody>
</table>

V. CONCLUSION

In this paper, a low-complexity decoding algorithm for NB-LDPC codes is proposed. The proposed scheme can level up the convergence speed and meanwhile improve memory efficiency. The corresponding decoder architecture is also presented.

REFERENCES