LDPC Error Floor Prediction using Trapping Set aware Code Shortening

Bane Vasić
Codelucida, Inc. & Department of ECE
University of Arizona
vasic@codelucida.com

David Declercq
Shiva Planjery
Codelucida, Inc.
{declercq,planjery}@codelucida.com

Ben Reynwar
Vamsi Krishna Yella
Codelucida, Inc.
{reynwar,vyella1}@codelucida.com

***Abstract*—In this presentation, we address the problem of the error floor prediction of Low Density Parity Check (LDPC) decoders. Our approach is based on Monte-Carlo simulations on a shortened version of the original code. The shortening is performed such that the resulting code contains the most harmful trapping sets, giving rise to an error floor with the same slope as the original code. Our results show an accurate prediction of the error floor with a computation speed-up of 10^4.**

I. Introduction

It is now well recognized that the low-density parity check (LDPC) codes iterative decoders suffer from the so-called error-floor problem, *i.e.*, a sudden change in the frame error rate (FER) slope due to the presence of trapping sets (TS). With the growing interest in LDPC based error correction for very high-reliability applications, many works have addressed this shortcoming, either by smart LDPC code design avoiding the dominant TSs [1], or by improving the decoders such that they can correct errors located on some TSs [2]. Digital storage applications are especially demanding in terms of high-reliability. Nowadays NAND flash memories require FER as low as FER=10^{-11} without an occurrence of an error floor. For emerging persistent memory technologies, such as 3D XPoint, much lower reliability is targeted, down to FER=10^{-17}. This extremely low level of error is obviously out of the reach for any simulation platform, even FPGA based, and one needs to rely on other approaches to predict the performance, and validate the LDPC solutions.

The first method for estimating the error floor dates back to the work of Richardson [3]. He introduced a semi-analytical method based on importance sampling (IS) which relies on the knowledge of most harmful TSs. Numerous subsequent papers relied on the IS to estimate error floor for various channels and decoding algorithms. In spite of these efforts, it remains highly non-trivial to determine which TS are dominant and cause failures for a given decoder.

We propose in this paper an approach different from IS based on code shortening, but still relying on the characterization of TS harmfulness. In our method, the FER is estimated by the traditional MC simulation without noise amplification, but performed on a smaller Tanner graph corresponding to judiciously selected columns of the parity check matrix. As we demonstrate, the proposed method reliably predicts the location and slope of the error floor. The main advantage of the proposed method is that it is easy and practical to implement in an FPGA simulation platform and offers tremendous savings in simulation time.

II. Error Floor Prediction based on Code Shortening

Let the parity-check matrix (PCM) of a quasi-cyclic LDPC (QC-LDPC) code be denoted by $\mathbf{H}$. It is composed of square blocks or sub-matrices of size $L \times L$, which can be either *(i)* an all-zero $L \times L$ block, or *(ii)* a circulant permutation matrices (CPM). A block-row $\mathbf{H}_{j,:}$ is composed of N_b blocks, while a block-column $\mathbf{H}_{:,i}$ is composed of M_b blocks:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{:,1} & \mathbf{H}_{:,2} & \dots & \mathbf{H}_{:,N_b} \end{bmatrix} \quad (1)$$

We denote by block-shortening the process of removing one or several block-columns from $\mathbf{H}$, in order to obtain the PCM of a QC-LDPC code with lower rate. A specific shortening is obtained by specifying a vector of s block-column indices $\mathbf{i} = [i_1, \dots, i_s]$, with $M_b < s < N_b$ and $i_k \in \{1, \dots, N_b\}$, and extracting from $\mathbf{H}$ the corresponding block columns. The shortened code $\mathbf{H_s} = \begin{bmatrix} \mathbf{H}_{:,i_1} & \mathbf{H}_{:,i_2} & \dots & \mathbf{H}_{:,i_s} \end{bmatrix}$ has rate $R_s = 1 - M_b/s < 1 - M_b/N_b$.

The principle of our approach is to obtain a shortened code $\mathbf{H_s}$ which contains the same harmful TSs as the original code $\mathbf{H}$, and run classical Monte Carlo simulations on $\mathbf{H_s}$ to detect its error floor. If the decoder fails on the same structures for $\mathbf{H_s}$ and $\mathbf{H}$, both curves will have an error floor with the same slope, but since $\mathbf{H_s}$ has lower rate, the error floor will appear at a higher FER, resulting in computational savings.

Chosing properly the vector $\mathbf{i}$ is critical for the efficiency of our approach. When s is large, close to N_b, it is easier to ensure that the harmful TSs of $\mathbf{H}$ will remain in $\mathbf{H_s}$, but computational saving would be small. To detect the error floor much faster, we should try to lower the code rate by chosing a small value for s. However, with a too small value of s, there is a chance that $\mathbf{H_s}$ does not contain anymore the harmful TSs, resulting in an erroneous prediction of the error floor. Our strategy is then to minimize the number of indices in $\mathbf{i}$, while still ensuring that the harmful TSs are present in $\mathbf{H_s}$, using a precise criterion of TS harmfulness, presented in the next section.

III. Trapping Set Harmfulness

To classify a trapping set as harmful, it is not sufficient to rely only on trapping set structural properties, such as local expansions, distribution of cycles, or types of outside connexions. We rely on the concept of *noisy critical number* [2] to classify the TSs in different hamfulness classes.

Let us start with some definition to identify and classify TSs. We discuss only the case of variable node (VN) regular LDPC codes, with constant degree $d_v = 4$. Denote by $\mathcal{G}$ the Tanner graph of a given LDPC code, and let $\mathcal{T}$ be a subgraph of a TS, with a VNs and c check nodes (CN). The local expansion of $\mathcal{T}$ is defined as $e_{\mathcal{T}} = \frac{c}{a}$. Denote by $b < c$ the number of odd degree CNs. When the b odd degree CNs have all degre-1 in $\mathcal{T}$, and the $(c-b)$ even degree CNs have all degree-2, the TS is called *elementary*. For an elementary TS (ETS), if each and every VN has more even degree CN than odd degree CN, it is called an elementary *absorbing set* (EAS). Let us finally denote by $g_{2\ell}$ the number of cycles of length 2ℓ in $\mathcal{T}$. The cycle profile is defined as $\mathbf{g} = (g_{2\ell})_{\ell \geq 2}$, and is used to differentiate between different TSs with the same values of (a, b, c).

TABLE I
LIST OF **most harmful** TSS IN A RATE $R = 0.903$, $N = 2kB$ LDPC CODE

$\mathcal{T}(a,b)$	$e_{\mathcal{T}}$	$\mathbf{g}$	$\lvert\mathcal{T}\rvert$	type	$\mathcal{T}(a,b)$	$e_{\mathcal{T}}$	$\mathbf{g}$	$\lvert\mathcal{T}\rvert$	type
$\mathcal{T}(5,6)$	$\frac{13}{5}$	(03210)	124	ETS	$\mathcal{T}(6,6)$	$\frac{15}{6}$	(03441)	38	ETS
$\mathcal{T}(5,6)$	$\frac{13}{5}$	(03300)	16	ETS	$\mathcal{T}(6,6)$	$\frac{15}{6}$	(03342)	39	ETS
					$\mathcal{T}(6,6)$	$\frac{15}{6}$	(04331)	1	ETS

TABLE II
LIST OF **less harmful** TSS IN A RATE $R = 0.903$, $N = 2kB$ LDPC CODE

$\mathcal{T}(a,b)$	$e_{\mathcal{T}}$	$\mathbf{g}$	$\lvert\mathcal{T}\rvert$	type	$\mathcal{T}(a,b)$	$e_{\mathcal{T}}$	$\mathbf{g}$	$\lvert\mathcal{T}\rvert$	type
$\mathcal{T}(5,6)$	$\frac{13}{5}$	(02320)	280	ETS	$\mathcal{T}(6,6)$	$\frac{15}{6}$	(02542)	129	ETS
$\mathcal{T}(6,6)$	$\frac{15}{6}$	(03343)	40	ETS	$\mathcal{T}(6,6)$	$\frac{15}{6}$	(02363)	12	EAS

A first attemp in defining the harmfulness of a TS was proposed in [5] introducing the notion of critical number (CN) of a TS. It is defined as the minimum number of nodes in $\mathcal{T}$ (out of a) that have to be initially in error for the decoder to fail to converge, when $\mathcal{T}$ is isolated from the rest of the graph $\mathcal{G}$. In [2], we extended this notion to the noisy critical number vector (NCNV) of a TSs, denoted $\mathbf{u}_{\mathcal{T}}$. A noisy critical number is defined as the CN when the TS is not isolated anymore, and receives noisy messages from the outer edges that connect $\mathcal{T}$ to the rest of the graph. The NCNV collects the set of critical numbers for different outer noise configurations. In [2], we used the NCNV to discriminate between iterative decoders stengths on a given TS, but it could also be used as a criterion to rank the hamfulness of different TSs with same values of (a, b, c). Smaller values in $\mathbf{u}_{\mathcal{T}}$ imply a larger probability of decoding failure.

In Tables I-III, we list the elementary TSs found in a rate $R = 0.903$, length $N = 2304\,B$ (Bytes) LDPC code, using the algorithm in [6]. Only TSs with $a \leq 6$ and $b \leq 6$ are shown. We also indicate the number of non-isomorphic TSs with $|\mathcal{T}|$. Based on the NCNV of these TSs, we classified them into 3 categories: TSs in Table I are classified as the most harmful, TSs in Table II are classified as less harmful, and finally TSs in Table III are classified as not harmful. One interesting observation is that although EAS are usually designated as the dominant TSs, our analysis shows that it is not always the case, and that some ETS could be more harmful than EAS. We use this classification of TSs to select the shortening vectors $\mathbf{i}$.

IV. SIMULATION RESULTS

In this section, we demonstrate the effectiveness of our approach on a QC-LDPC code $\mathcal{C}$, with regular column degree $d_v = 4$, codeword length $N = 2304$ Bytes, and rate $R = 0.903$. Two shortened codes are designed from $\mathcal{C}$: $\mathcal{C}_{s_1}$, with rate $R = 0.8$, in which we kept a maximum number of harmful TSs from tables I and II ; and $\mathcal{C}_{s_2}$, with rate $R = 0.7$, in which we kept a maximum number of most harmful TSs from table I, regardless of the other TSs. We plot on Fig. 1 the performance of the original code $\mathcal{C}$ together with the shortened codes $\mathcal{C}_{s_1}$ and $\mathcal{C}_{s_2}$, on the BSC channel. The ratio between the numbers of most harmful trapping sets in the smaller and the original Tanner graphs is used for weighting the error-floor level in these plots. First, we simulated the three codes using the classical offset corrected Min-Sum (OMS) algorithm, which is known to have an error-floor due to the TSs. As we can see, the error floor prediction technique is very accurate using both shortened codes. The error floor appears around FER$= 10^{-9}$ for $\mathcal{C}$, and around FER$= 10^{-5}$ for $\mathcal{C}_{s_2}$, which provides a saving of 10^4 in computation time. In a second experiment, we simulated the Finite Alphabet Iterative Decoder (FAID), proposed in [2], on the shortened code $\mathcal{C}_{s_1}$. FAID is known to be insensitive to the attraction of dominant TSs, which results in improved performance in the error floor. Using our technique, we could then estimate that the error floor for FAID will appear around FER$= 10^{-15}$ on the original code. This level of FER is not reachable using Monte Carlo simulation, even with FPGA boards.

TABLE III
LIST OF **not harmful** TSS IN A RATE $R = 0.903$, $N = 2kB$ LDPC CODE

$\mathcal{T}(a,b)$	$e_{\mathcal{T}}$	$\mathbf{g}$	$\lvert\mathcal{T}\rvert$	type	$\mathcal{T}(a,b)$	$e_{\mathcal{T}}$	$\mathbf{g}$	$\lvert\mathcal{T}\rvert$	type
$\mathcal{T}(4,6)$	$\frac{11}{4}$	(02100)	926	ETS	$\mathcal{T}(6,6)$	$\frac{15}{6}$	(04321)	11	ETS
$\mathcal{T}(6,6)$	$\frac{15}{6}$	(03522)	11	ETS	$\mathcal{T}(6,6)$	$\frac{15}{6}$	(02640)	13	ETS

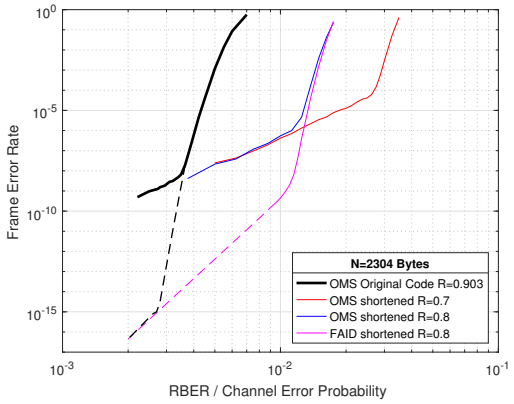

Fig. 1. Results of Error floor prediction on a N=2kB - R=0.903 LDPC code.

V. CONCLUSION

In this presentation, we propose a new an efficient prediction technique for the estimation of the error floor of LDPC decoders. The approach is based on code shortening, using the knowledge of the most harmful TSs present in an LDPC code. We validate our approach on a test code with length $N = 2304$ Bytes, and rate $R = 0.903$, using the OMS decoder, and then show that the error floor of FAID decoder is estimated to be 10^7 times lower than the OMS error floor.

ACKNOWLEDGMENTS

This work was supported in parts by the National Science Foundation under SBIR Phase II Grant 1534760.

REFERENCES

[1] D. V. Nguyen, S. K. Chilappagari, B. Vasić, and M. W. Marcellin, "On the construction of structured LDPC codes free of small trapping sets," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2280–2302, Apr. 2012.

[2] S. K. Planjery, D. Declercq, L. Danjean, and B. Vasić, "Finite alphabet iterative decoders, Part I: Decoding beyond belief propagation on the binary symmetric channel," *IEEE Trans. Commun.*, vol. 61, no. 10, pp. 4033–4045, Nov. 2013.

[3] T. J. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annual Allerton Conference on Communications, Control and Computing*, Monticello, IL, USA, Sept. 2003, pp. 1426–1435.

[4] L. Dolecek, Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 181–201, Jan. 2010.

[5] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasić, "Error floors of LDPC codes on the binary symmetric channel," in *Proc. IEEE International Conference on Communications (ICC '06)*, vol. 3, Istanbul, Turkey, 2006, pp. 1089–1094.

[6] M. Karimi and A. Banihashemi, "Efficient algorithm for finding dominant trapping sets of LDPC codes," *IEEE Trans. Inf. Theory,*, vol. 58, no. 11, pp. 6942–6958, Nov. 2012.