ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/260672913

An RLL-Constrained LDPC Coded Recording System Using Deliberate Flipping and Flipped-Bit Detection

Article *in* IEEE Transactions on Communications · December 2012

DOI: 10.1109/TCOMM.2012.101712.110501

CITATIONS
5

READS
94

4 authors, including:

Hong-Fu Chou
Auckland University of Technology
5 PUBLICATIONS **22** CITATIONS
SEE PROFILE

Marc Fossorier
École Nationale Supérieure de l'Electronique et de ses Applications
305 PUBLICATIONS **12,234** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

RLL LDPC code View project

All content following this page was uploaded by Marc Fossorier on 28 October 2014.

The user has requested enhancement of the downloaded file.

An RLL-Constrained LDPC Coded Recording System Using Deliberate Flipping and Flipped-Bit Detection

Hong-Fu Chou, Yeong-Luh Ueng, Mao-Chao Lin, *Member, IEEE,* and Marc P. C. Fossorier, *Fellow, IEEE*

***Abstract*—In this paper, a low-density parity-check (LDPC) coded recording system is investigated, for which the run-length-limited (RLL) constraint is satisfied by deliberate flipping at the write side and by estimating the flipped bits at the read side. Two approaches are proposed for enhancing the error performance of such a system. The first approach is to alleviate the negative effect of incorrect estimation of the flipped bits by adjusting the soft information. The second approach is to increase the likelihood of the correct detection of flipped bits by designing a flipped-bit detection algorithm that utilizes both the RLL constraint and the parity-check constraint of the LDPC code. These two approaches can be combined to obtain significant improvement in performance over previously proposed methods.**

***Index Terms*—Constrained codes, low-density parity-check (LDPC) codes, recording, run-length-limited (RLL).**

I. Introduction

In recording systems, data bits are transformed into symbol sequences which must satisfy a run-length-limited (RLL) constraint. A (d, k) RLL sequence [1][2][3][4][5][6] is a sequence for which the number of zero symbols between two consecutive non-zero symbols is at least d and at most k in order to alleviate the inter-symbol interference (ISI), and to assist with timing recovery at the read side [7][8].

In order to increase storage reliability, error-correcting codes (ECC) are usually employed in recording systems. A low-density parity-check (LDPC) [9][10] coded recording system with RLL constraints can be constructed by sequentially encoding the user data through both the LDPC encoder and the RLL encoder [11][12]. At the read side, an equalizer followed by an RLL decoder and an LDPC decoder is applied. This is called the ECC-RLL scheme. However, this arrangement significantly increases the complexity in cases where turbo equalization [13] is needed, and is due to the insertion of the RLL decoder between the equalizer and the LDPC decoder. Various approaches have been proposed to overcome this problem [14][15][16][17][18][19]. In both [17] and [19], a reverse-concatenation scheme (or RLL-ECC scheme) is proposed to enable the construction of RLL codes with error-correcting capabilities, where the user data is first encoded through the RLL encoder and then the LDPC encoder. A major disadvantage of this approach is the rate loss resultant from the RLL code, although some novel arrangements can be used to reduce this loss. In [14], the authors propose to obtain a sequence that satisfies the RLL constraint by deliberately flipping those bits which violate the constraint in each LDPC codeword. In the flipping-based scheme, turbo equalization can still be performed since there is no RLL decoder at the read side. The scheme relies only on the capability of the LDPC code to correct the errors caused by the flipping operation. However, such a scheme can only be applied to systems with a very loose RLL constraint, otherwise the number of flipping bits becomes too large, which significantly degrades the error performance. In order to reduce the number of flipped bits, a selective flipping technique [15][16] is employed. The idea is that for each LDPC codeword, we obtain multiple bit-flipping versions (i.e., multiple candidates) and then choose the candidate which generates the fewest flipping errors when writing. This technique is efficient except for the need for side information, which is required in order to inform the read side which candidate has been chosen at the write side. The side information should be well-protected and should also satisfy the RLL constraint. In [18], the RLL constraint is applied to the output of the LDPC decoder in order to detect the flipped bits. Then, in the next iteration, the extrinsic LLRs (log likelihood ratios) which are to be passed to the equalizer and the LDPC decoder, respectively, are modified by reversing the polarity of the extrinsic LLR of each of the detected flipped bits. The detection process presented in [18] is not appropriate for flipping-based RLL recording systems that are not based on the selective flipping technique since the output of the LDPC decoder may contain too many errors, which will make the detection of the flipped bits extremely difficult.

In this paper, we focus on an LDPC-coded recording system with the $(0, k)$ constraint. We propose to modify the method presented in [18] using two approaches. For the first approach, the soft information of the detected flipped bits is adjusted to alleviate the effect of hard errors due to the incorrect estimation of flipped bits. The methods in consideration include

Paper approved by T.-K. Truong, the Editor for Coding Theory and Techniques of the IEEE Communications Society. Manuscript received August 1, 2011; revised May 3 and July 31, 2012.

This work was supported by the National Science Council of the R.O.C. under grants NSC 100-2221-E-007-071 and NSC 97-2221-E-002-145-MY3.

H.-F. Chou is with the Graduate Institute of Communication Engineering, National Taiwan University, Taipei 10617, Taiwan, R.O.C. (e-mail: d95942020@ntu.edu.tw).

Y.-L. Ueng is with the Department of Electrical Engineering and the Institute of Communications Engineering, National Tsing Hua University, Hsinchu, Taiwan, R.O.C. (e-mail: ylueng@ee.nthu.edu.tw).

M.-C. Lin is with the Department of Electrical Engineering and the Graduate Institute of Communication Engineering, National Taiwan University, Taipei 10617, Taiwan, R.O.C. (e-mail: mclin@cc.ee.ntu.edu.tw).

M. Fossorier is with ETIS ENSEA, UCP, CNRS UMR, Pontoise, France (e-mail: mfossorier2@yahoo.com).

Digital Object Identifier 10.1109/TCOMM.2012.101712.110501

0090-6778/12$31.00 © 2012 IEEE

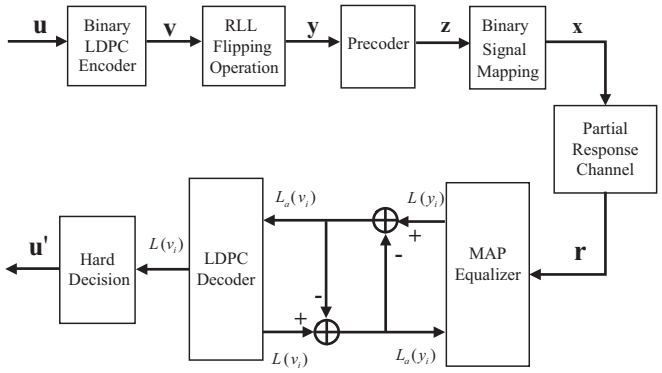


Fig. 1. An LDPC-coded binary recording system based on RLL constraints using deliberate flipping.

(i) erasing the soft information; (ii) using the average of the absolute values of the LLRs generated during each iteration as the magnitude of the soft information; and (iii) using the average of the absolute values of the LLRs generated during each iteration as the clipping threshold of the soft information. For the second approach, we consider a design for flipped-bit detection that properly utilizes both the RLL constraint and the parity-check constraint of the LDPC code in order to increase the likelihood of the correct detection of the flipped bits. We show that by appropriately examining both the RLL constraint and the parity check constraint at the output of the LDPC decoder for each iteration, we can significantly increase the probability of a correct detection of the flipped bits and, hence, reduce the overall error rate of the system. These two approaches can be combined to further improve the error performance.

The remainder of this paper is organized as follows. In Section II, a brief review of the flipping-based LDPC-coded system proposed in [14] and the method of detection for flipping bits in [18] is presented. In Section III, the approach for adjusting the soft information for the flipped bits is introduced. In Section IV, we present the approach for detecting flipped bits based on both the RLL constraint and the parity check constraint. Finally, conclusions are given in Section V.

II. Review of Flipping-based LDPC Coded Recording Systems with RLL Constraints

The idea of deliberately flipping some bits of the LDPC codeword to meet the RLL constraint at the write side, and using the error-correcting capability of the LDPC code to remove the flipped bits at the read side, was proposed in [14]. Improved versions were presented in [16] and [18].

A. Deliberate Flipping at the Write Side

Fig. 1 shows the block diagram for the system presented in [14]. A K-bit message $\mathbf{u}$ is forwarded to the encoder of a binary (N, K) LDPC code to produce an N-bit codeword $\mathbf{v} \equiv (v_1,v_2,\cdots,v_N)$. The codeword $\mathbf{v}$ is forced to satisfy the $(0,k)$ RLL constraint by flipping certain bits of $\mathbf{v}$ that violate the $(0,k)$ constraint, thereby obtaining a flipped version denoted as $\mathbf{y} \equiv (y_1,y_2,\cdots,y_N)$, where $v_i \in \{0,1\}$, $y_i \in \{0,1\}$, $i = 1,2,\cdots,N$. The flipping operation can be implemented as the summation of the codeword $\mathbf{v}$ and a binary location vector $\mathbf{q} \equiv (q_1,q_2,\cdots,q_N)$, i.e., $\mathbf{y} = \mathbf{v} \bigoplus \mathbf{q}$, where $\oplus$ represents the XOR operation. The location vector can be obtained using a sliding-window-based algorithm. For the case of the $(0,k)$ constraint considered in this work, the steps are described as follows:

Step 1: The window size is set to $k+1$ bits. The N-bit location vector $\mathbf{q}$ is set to the zero vector, and $i = 1$.

Step 2: If both vectors $(v_i, v_{i+1}, \cdots, v_{i+k})$ and $(q_i, q_{i+1}, \cdots, q_{i+k})$ are the zero $(k+1)$-tuple, then set q_{i+k} to 1.

Step 3: Repeat Step 2 for $i = 2,3\cdots,N-k$.

If $q_i = 1$, then v_i will be flipped before being stored in the recording media.

In Fig. 1, the output of the RLL flipping operation unit, $\mathbf{y}$, is processed by a precoder to yield $\mathbf{z} = (z_1, z_2, \cdots, z_N)$, where

$$z_i = y_i + z_{i-1} \text{ (mod 2)}.$$

Through a modulator (binary signal mapper), we have $\mathbf{x} = (x_1, x_2, \cdots, x_N)$, where

$$x_i = (-1)^{z_i}.$$

B. Channel Model

The signal $\mathbf{x}$ is written to the recording media. A practical magneto-optical (MO) recording channel presented in [20] and [21] is considered in this study. The channel impulse response $f(t)$ considered in [20] and [21] is given by

$$f(t) = \frac{2}{ST\sqrt{\pi}} exp\{-(\frac{2t}{ST})^2\} \tag{1}$$

where T is the bit duration and $S = 1.0$ is used. If we let $\prod(t)$ be the function which equals 1 for $0 \le t \le T$ and equals 0 otherwise, then the transition response $g(t)$ is the convolution of $\prod(t)$ and $f(t)$. The output signal of the MO recording channel is

$$r(t) = \sum_{i=-\infty}^{\infty} x_i g(t-(i+\Delta_i)T) + n(t), \tag{2}$$

where $\{\Delta_i\}$ is an independent and identically distributed random process representing the jitter effect, and $n(t)$ is the additive white Gaussian noise (AWGN) with one-sided power spectral density N_0. We assume that each Δ_i is a zero mean Gaussian random variable with variance σ_Δ^2. By assuming that $|\sigma_\Delta|T$ is small and $g(t)$ is sufficiently bandlimited [21], we have

$$g(t-(i+\Delta_i)T) \approx g(t-iT) - \Delta_i T f(t-iT). \tag{3}$$

By substitution, (2) then becomes

$$r(t) \approx \sum_{i=-\infty}^{\infty} x_i g(t-iT) - \sum_{i=-\infty}^{\infty} x_i \Delta_i T f(t-iT) + n(t), \tag{4}$$

where $\sum_{i=-\infty}^{\infty} x_i \Delta_i T f(t-iT)$ is the jitter noise. The average energy of the jitter noise for each received code symbol is

$$\begin{aligned} M_0 &= E\{x_i^2\}E\{\Delta_i^2\}E\{\sum_{j=\infty}^{\infty}[Tf(jT)]^2\} \\ &= \sigma_\Delta^2 \sum_{j=\infty}^{\infty}[Tf(jT)]^2. \end{aligned} \tag{5}$$

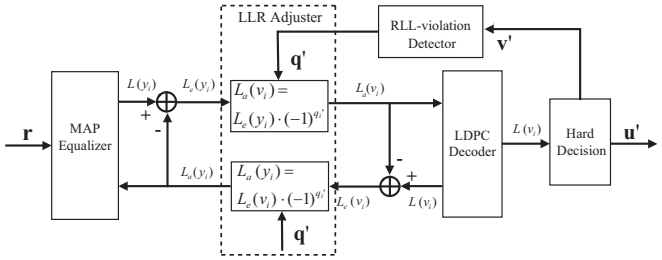


Fig. 2. Block diagram for the read side of the LDPC-coded binary recording system presented in [18].

The jitter noise effect is simulated according to (4) using a Gaussian distributed Δ_i. In the simulation, the energy of the signal x_i is fixed as 1 and the energy of the jitter noise M_0 is set to be proportional to N_0. We set $M_0 = \beta N_0$, where $\beta = 0.15$. In simulation, discrete time samples are required. From $f(t)$ and $g(t)$, we have $Tf(-2T) = Tf(2T) = 0.0962$, $Tf(-T) = Tf(T) = 0.1085$, $Tf(0) = 0.1128$ and $g(T) = g(4T) = 0.1114$, $g(2T) = g(3T) = 0.1028$. In addition, we set $f(iT) = 0$, for $\mid i \mid > 2$ and $g(iT) = 0$ for $i \leq 0$ and $i \geq 5$. Using (4), we can obtain the associated discrete-time output sequence denoted as $\mathbf{r}$.

C. Turbo Equalization at the Read Side

The sequence $\mathbf{r}$ suffers not only from noise (AWGN and jitter noise), but also from intersymbol interference (ISI). As shown in Fig. 1, the sequence $\mathbf{r}$ is forwarded to a SISO (soft-input soft-output) MAP (maximum *a posteriori*) equalizer combined with the precoder and a PR (partial response) channel in order to combat the effect of ISI. The MAP equalizer utilizes the BCJR algorithm [22] to produce the soft outputs (log likelihood ratios, LLRs) $L(y_i)$ for bits y_i, where $i = 1, 2, \cdots, N$. The *a priori* LLRs $L_a(v_i)$, where $i = 1, 2, \cdots, N$, for the LDPC decoder are calculated according to $L_a(v_i) = L(y_i) - L_a(y_i)$, where $L_a(y_i)$ is the *a priori* LLR for the MAP equalizer, and is produced from the soft output $L(v_i)$ of the LDPC decoder according to $L_a(y_i) = L(v_i) - L_a(v_i)$. The sum-product algorithm presented in [10] using U_i inner iterations is performed in the LDPC decoder. Moreover, U_o outer iterations are processed between the MAP equalizer and the LDPC decoder in order to realize the turbo equalization [13].

D. Detection of Flipped Bits at the Read Side

Fig. 2 shows the block diagram for the read side of the system proposed in [18], where turbo equalization with the consideration of flipped-bit detection is used. The MAP equalizer and the LDPC decoder sequentially process the sequence $\mathbf{r}$ to yield a decoded codeword $\mathbf{v}'$. Then, $\mathbf{v}'$ is forwarded to an RLL-violation detector so as to identify the bit positions which need to be flipped in order to meet the RLL constraint. This RLL-violation detector follows exactly the same procedure as described in Steps 1 to 3 in Section II-A to produce the resultant location vector denoted as $\mathbf{q}'$. Then, the polarities of the extrinsic LLRs of the bit positions corresponding to the non-zero bits in $\mathbf{q}'$ are reversed in the LLR adjuster before being passed to the decoder and the equalizer. Specifically,

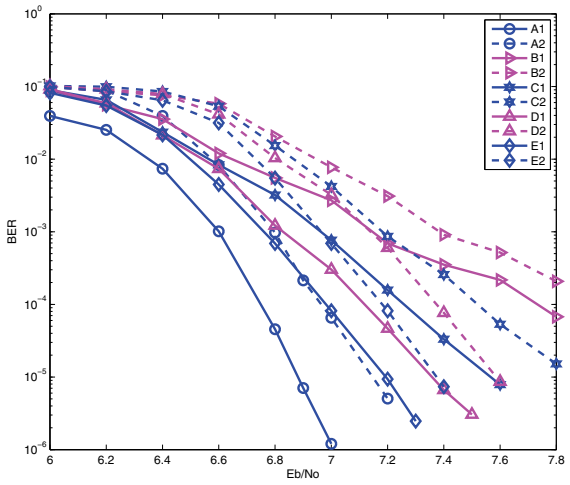


Fig. 3. BER results for LDPC-coded recording systems over the MO channel, where β = 0.15, (64,64)-regular (4095,3367) EG-LDPC code, and the (0,7) constraint are considered. Outer iteration $U_o = 15$. Inner iteration $U_i = 1$. (A1) Non-flipped system using a non-reset LDPC decoder; (A2) Non-flipped system using a reset LDPC decoder; (B1) Flipped system using the Soft-O method and a non-reset LDPC decoder; (B2) Flipped system using the Soft-O method and a reset LDPC decoder; (C1) Flipped system using the Soft-I method and a non-reset LDPC decoder; (C2) Flipped system using the Soft-I method and a reset LDPC decoder; (D1) Flipped system using the Soft-II method and a non-reset LDPC decoder; (D2) Flipped system using the Soft-II method and a reset LDPC decoder; (E1) Flipped system using the Soft-III method and a non-reset LDPC decoder; (E2) Flipped system using the Soft-III method and a reset LDPC decoder.

when $q'_i = 1$, the inversion operation used in [18] will change both the sign of the *a priori* LLR $L_a(y_i)$ of y_i for the MAP equalizer and the sign of the *a priori* LLR $L_a(v_i)$ of v_i for the LDPC decoder. The adjustment method described in [18] is denoted as method Soft-O.

Example 1: Consider the MO channel with jitter noise as described in Section II-B. Suppose that the (0,7) RLL constraint is used. The LDPC code is a rate-0.82 $(64, 64)$-regular $(4095, 3367)$ EG-LDPC code [23]. The number of outer iterations is U_o = 15 and the number of inner iterations is U_i = 1. A syndrome check is used to determine whether or not the decoded codeword is valid. If a valid codeword is obtained, we will terminate the process even if the number of outer iterations does not reach the maximum number U_o = 15. Based on the flipping system described in [18] and also in this section, we can obtain the BER performance, which is shown as Curve (B1) in Fig. 3, where the non-reset LDPC decoder is used. For comparison, Curve (A1) in Fig. 3 depicts the BER performance of the non-flipped system, for which all the parameters are the same as that used for Curve (B1), except that the (0,7) constraint is removed. We see that Curve (B1) greatly deviates from Curve (A1), which implies that there is significant room for improvement in the flipping system considered in [18]. Curves (B2) and (A2) in Fig. 3 show the BER performances of the flipped system used in [18] and the non-flipped system, respectively, where the reset LDPC decoder is used, and it can be seen that Curve (B2) deviates markedly from Curve (A2). Moreover, we see that using the reset LDPC decoder will lead to inferior BER performances as compared to using the non-reset LDPC decoder. □

We propose two approaches designed to tackle the problems related to the possible incorrect detection of flipped bits. In Section III, we will present the first approach, which consists of methods to alleviate the negative effects of the incorrect detection of the flipped-bit locations. In Section IV, we will present the second approach, which consists of methods to enhance the likelihood of the detection of flipped bits.

III. Adjusting the Soft Information of Flipped Bits

In order to alleviate the negative effects of the incorrect detection of the flipped-bit locations in the system presented in [18], we propose modifying the LLR adjuster illustrated in Fig. 2 in such a way that the magnitudes of $L_a(y_i)$ and $L_a(v_i)$ will be adjusted if $q'_i = 1$, while the values of $L_a(y_i)$ and $L_a(v_i)$ will remain unchanged if $q'_i = 0$.

A. Proposed Adjustment Methods for Soft Information

Three methods, denoted Soft-I, Soft-II, and Soft-III, respectively, for implementing the LLR adjustment are presented as follows:

Soft-I: We apply erasure to each of the detected flipped bits. Hence, $L_a(y_i)$ and $L_a(v_i)$ are set to zero for $q'_i = 1$.

Soft-II: The inversion operation of the LLR adjuster for each of the detected flipped bits remains the same as in Soft-O [18]. However, the magnitude of the *a priori* LLR, $L_a(y_i)$ (and $L_a(v_i)$) for each of the detected flipped bits is adjusted to the average magnitude of all the $L_e(y_i)$ values (and all the $L_e(v_i)$ values) during the same iteration. The average of the extrinsic LLR values $|L_e(y_i)|$, $i = 1, 2, \cdots, N$, is denoted as γ_y, and the average of the extrinsic LLR values $|L_e(v_i)|$, $i = 1, 2, \cdots, N$, is denoted as γ_v.

Soft-III: The LLR adjuster illustrated in Fig. 2 is modified as follows. An average-based clipping operation is applied in addition to the inversion operation used in Soft-O. This method is also based on the average magnitude setting used in Soft-II. In this method, γ_y and γ_v are used as the clipping ratios for vectors $[L_e(y_1), L_e(y_2), \cdots, L_e(y_N)]$ and $[L_e(v_1), L_e(v_2), \cdots, L_e(v_N)]$, respectively. The clipping operation is activated if $q'_i = 1$. When the LLR magnitude $|L_e(y_i)|$ ($|L_e(v_i)|$) is larger than γ_y (γ_v), we reverse the polarity of the LLR value and set its magnitude to the value $\gamma_y(\gamma_v)$. On the other hand, when the LLR magnitude is smaller, then only the polarity of the LLR value is reversed.

B. Performance Evaluation

1) EXIT characteristics: We employ EXIT (extrinsic information transfer) characteristics [24] analysis to evaluate the information exchange between the LDPC decoder and the MAP equalizer (detector). Since there is an LLR adjuster inserted between the MAP equalizer and the LDPC decoder, the transfer functions of the MAP equalizer and the LDPC decoder in this paper is defined as follows. The transfer characteristic T_Z for the MAP equalizer takes one half of the LLR adjuster into consideration. Hence, the input mutual information is $I_{Z,in} = I_Z(L_a(y_i), y_i)$, and the output mutual information is $I_{Z,out} = I_Z(L_a(v_i), v_i) = T_Z(I_{Z,in})$. Since the

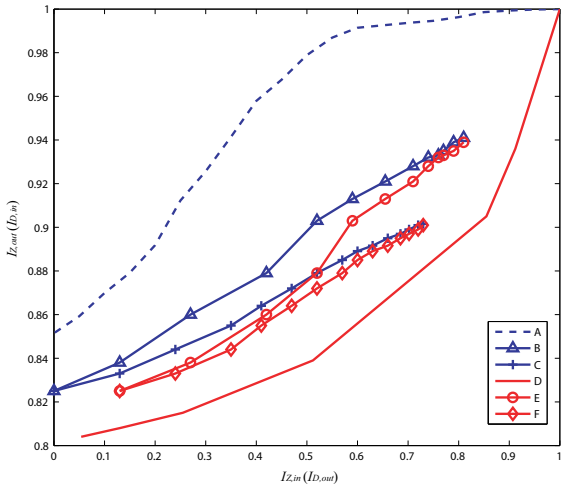


Fig. 4. EXIT curves for the MAP equalizer and the LDPC decoder over the MO channel, where β = 0.15, E_b/N_0 = 7.6 dB, (64,64)-regular (4095,3367) LDPC code, the Soft-O method, and the (0,7) constraint are considered. The transfer characteristic of the MAP equalizer is denoted as T_Z and the transfer characteristic of the LDPC decoder is denoted as T_D. (A) T_Z based on perfect $\mathbf{q}'$; (B) T_Z based on detected $\mathbf{q}'$ using a non-reset LDPC decoder; (C) T_Z based on detected $\mathbf{q}'$ using a reset LDPC decoder; (D) T_D based on perfect $\mathbf{q}'$; (E) T_D based on detected $\mathbf{q}'$ using a non-reset LDPC decoder; (F) T_D based on detected $\mathbf{q}'$ using a reset LDPC decoder.

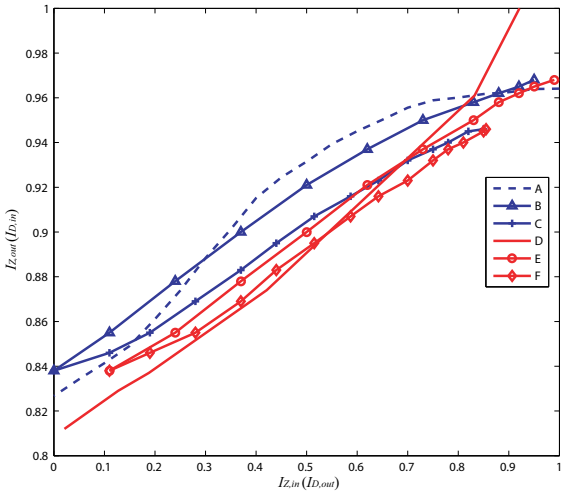


Fig. 5. EXIT curves for the MAP equalizer and the LDPC decoder over the MO channel, where β = 0.15, E_b/N_0 = 7.6 dB, (64,64)-regular (4095,3367) LDPC code, the Soft-I method, and the (0,7) constraint are considered. The transfer characteristic of the MAP equalizer is denoted as T_Z and the transfer characteristic of the LDPC decoder is denoted as T_D. (A) T_Z based on perfect $\mathbf{q}'$; (B) T_Z based on detected $\mathbf{q}'$ using a non-reset LDPC decoder; (C) T_Z based on detected $\mathbf{q}'$ using a reset LDPC decoder; (D) T_D based on perfect $\mathbf{q}'$; (E) T_D based on detected $\mathbf{q}'$ using a non-reset LDPC decoder; (F) T_D based on detected $\mathbf{q}'$ using a reset LDPC decoder.

MAP equalizer takes the channel condition into consideration, T_Z is dependent on E_b/N_0, β, and k, which are the AWGN, jitter noise, and RLL constraint parameters, respectively. The transfer characteristic T_D for the LDPC decoder takes the other half of the LLR adjuster into consideration. Hence, the input mutual information is $I_{D,in} = I_D(L_a(v_i), v_i)$, and the output mutual information is $I_{D,out} = I_D(L_a(y_i), y_i) =$

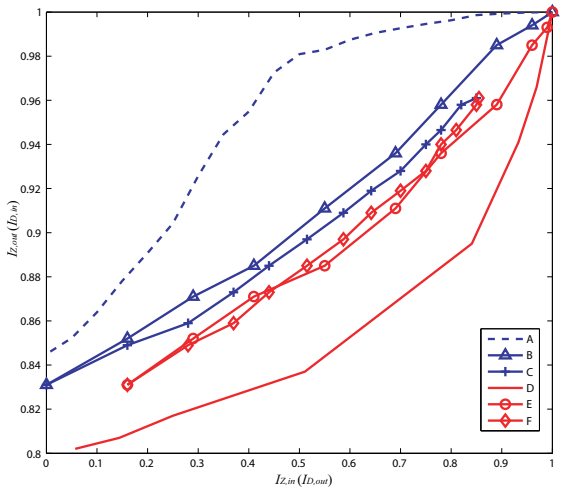


Fig. 6. EXIT curves for the MAP equalizer and the LDPC decoder over the MO channel, where β = 0.15, E_b/N_0 = 7.6 dB, (64,64)-regular (4095,3367) LDPC code, the Soft-II method, and the (0,7) constraint are considered. The transfer characteristic of the MAP equalizer is denoted as T_Z and the transfer characteristic of the LDPC decoder is denoted as T_D. (A) T_Z based on perfect $\mathbf{q}'$; (B) T_Z based on detected $\mathbf{q}'$ using a non-reset LDPC decoder; (C) T_Z based on detected $\mathbf{q}'$ using a reset LDPC decoder; (D) T_D based on perfect $\mathbf{q}'$; (E) T_D based on detected $\mathbf{q}'$ using a non-reset LDPC decoder; (F) T_D based on detected $\mathbf{q}'$ using a reset LDPC decoder.

$T_D(I_{D,in})$.

Based on the MO channel, the RLL constraint and the LDPC code shown in Example 1, the EXIT results using the Soft-O, Soft-I and Soft-II methods are obtained and are shown in Fig. 4, Fig. 5 and Fig. 6, respectively. Due to space limitations, the EXIT results for Soft-III are not shown in this paper.

Transfer characteristics can be obtained by using either the perfect $\mathbf{q}'$, i.e., $\mathbf{q}' = \mathbf{q}$, or the detected $\mathbf{q}'$. First, we examine the transfer characteristics obtained using the perfect $\mathbf{q}'$. We follow the approach described in [25] to obtain the transfer characteristics of the MAP equalizer and the LDPC decoder, where the perfect $\mathbf{q}'$, and $U_i = 1$ are used. The input mutual information $I_{Z,in}$ and $I_{D,in}$ for the equalizer and the decoder are obtained based on the assumption that the random variable $L_a(y_i)$ and $L_a(v_i)$ are Gaussian distributed. Using $L_a(y_i)$ as the input, we execute the BCJR algorithm in the equalizer to obtain the output LLR and then numerically compute the output mutual information $I_{Z,out}$. In a similar way, we execute the sum-product algorithm in the decoder to obtain the output LLR and then numerically compute the output mutual information $I_{D,out}$ using $L_a(v_i)$ as the input. The EXIT curve T_Z (T_D) for the equalizer (decoder) can be obtained by plotting $I_{Z,out}$ ($I_{D,out}$) versus $I_{Z,in}$ ($I_{D,in}$). In Fig. 4, Curve (A) and Curve (D) intersect at a mutual information value extremely close to 1, which indicates that the Soft-O method converges at $E_b/N_0 = 7.6dB$. In Fig. 5, Curve (A) and Curve (D) do not converge at $E_b/N_0 = 7.6dB$, since in the Soft-I method the erasure operation of the LLR value on the perfectly-detected flipped bits induces some loss of mutual information output compared to that of the Soft-O method. In Fig. 6, the convergence behavior of Curve (A) and Curve (D) for the Soft-II method is similar to that obtained using the Soft-O method.

Now, we examine the transfer characteristics obtained using the detected $\mathbf{q}'$. Turbo equalization using the detected $\mathbf{q}'$ can be analyzed by extracting the envelope curve of the trajectory characteristics. We rely on the simulation of iterative message passing between the MAP equalizer and the LDPC decoder to obtain the trajectory characteristics T_Z and T_D for the EXIT chart. We first take the envelopes of the staircase path for the trajectory of the iterative detection to obtain the transfer functions for the MAP equalizer and the LDPC decoder, respectively, based on each codeword $\mathbf{v}$. The final curves for T_Z and T_D are then obtained by averaging an ensemble of codewords. Both the reset and non-reset operations are considered in the LDPC decoder. Although the non-reset LDPC decoder needs more memory to store check-to-bit messages passed from the previous outer iteration, using the non-reset LDPC decoder can provide better performance compared to using the reset decoder, as can be seen from Figs. 4, 5 and 6.

In Fig. 4, Curves (B) and (E), using the Soft-O method, do not converge to a mutual information point with value close to 1 at E_b/N_0 of 7.6 dB. In contrast, Curves (B) and (E) in Fig. 5 at E_b/N_0 of 7.6 dB can converge to a mutual information point with value close to 1. This shows that the erasure operation of the Soft-I method reduces the negative effect of the incorrect detection of flipped bits present in the Soft-O method.

The Soft-II method adjusts the LLR magnitudes of the flipped bits, which prevents the spread of erroneous LLRs with large magnitudes. In Fig. 6, we can see that Curve (B) and Curve (E) intersect at a mutual information value close to 1.

2) BER performance: Based on the MO channel, the RLL constraint and the LDPC code shown in Example 1, we obtain the BER performances of the non-flipped system and the flipping-based systems using the Soft-O method and the proposed methods, where both the reset and the non-reset algorithms are considered in the LDPC decoder. The results are given in Fig. 3, where it can be seen that the proposed Soft-I and Soft-II methods can achieve an improved BER performance as compared to the Soft-O method. Moreover, using the Soft-II method can achieve a better BER performance than using the Soft-I method. It can also be observed that using the non-reset algorithm can provide a better performance compared to using the reset algorithm, which is consistent with the results implied by the EXIT curves shown in Figs. 4, 5 and 6.

Note that in the Soft-II method, the LLR magnitude is raised to the average value even if the magnitude of the incorrectly-detected LLR is smaller than the average value. However, this effect may deteriorate the detection process, although this undesired effect can be eliminated by applying the clipping operation employed in Soft-III. It can be seen from Fig. 3 that the proposed Soft-III method can further improve the performance as compared to the Soft-II method.

IV. Improved Detection of Flipped Bits

From the EXIT curves shown in Fig. 4 and Fig. 6, we see that for both the Soft-O method and the Soft-II method, using

the perfect $\mathbf{q}'$, i.e., $\mathbf{q}' = \mathbf{q}$, can achieve a better convergence performance compared to using the detected $\mathbf{q}'$. From the EXIT curves not shown in this paper, the same conclusion was reached for the Soft-III method as the curve displayed a pattern that closely followed those of Soft-II. Note that the conclusion is not true for using the Soft-I method, as can be seen from Fig. 5. These results imply that improving the detection of $\mathbf{q}'$ can improve the BER performance of the flipping-based system, except for the Soft-I method. In Fig. 2, the estimated location vector $\mathbf{q}'$ is obtained from $\mathbf{v}'$ by checking the RLL constraint, while $\mathbf{v}'$ is obtained by applying a hard decision to the output LLR values of the LDPC decoder. In the flipping-based system, the received symbols are corrupted by a combination of the AWGN, the jitter noise and the flipping operation. The AWGN and the jitter noise may be combined to be approximated as the Gaussian-distributed noise. However, hard errors introduced by the flipping operation may not. Therefore, the LLR value can not closely reflect the likelihood of the associated symbol. Thus, there is room to improve the detection of $\mathbf{q}'$ by further examining the parity check constraint and the RLL constraint on $\mathbf{v}'$.

A. Parity-check Constraint for Estimating Unreliable Bits

For a binary linear code, the number of failed checks associated with each bit position can be employed in order to estimate the possible errors in the vector to be decoded. A well-known example is the simple bit flipping (BF) algorithm [23], which is designed such that some unreliable bits are identified by examining the parity check constraint, and then these unreliable bits are flipped part as the error correction operation. We denote the simple BF algorithm with only one iteration as Pcheck-1, which is used for identifying the unreliable bits of output $\mathbf{v}'$ in our LDPC decoder. In this subsection, two variations, denoted as Pcheck-2 and Pcheck-3, will be presented.

Let an $M \times N$ matrix, denoted as $\mathbf{H}$, be a parity-check matrix of an $(N,\ K)$ LDPC code, where $M \geq (N - K)$. The parity-check matrix $\mathbf{H}$ can be represented by a Tanner graph [26], which is a bipartite graph consisting of variable (bit) nodes and check nodes representing the columns and rows of $\mathbf{H}$, respectively. If there is a 1 at the i-th column and the j-th row of $\mathbf{H}$, then there is an edge between the i-th variable node and the j-th check node in its Tanner graph. For $j = 1, 2, \cdots, M$, let W_j denote the index set of variable nodes connected to the j-th check node and $\mathbf{h}_j$ denote the j-th row of $\mathbf{H}$. In Pcheck-1, the vector $\mathbf{p} \equiv (p_1, p_2, \cdots, p_N)$, which indicates the location of unreliable bits (i.e., the bits which are estimated to be in error) in $\mathbf{v}'$, is obtained using the following steps.

Step 1: Compute the syndrome vector $\mathbf{s}$ according to

$$\mathbf{s} \equiv [s_1, s_2, ..., s_M] = \mathbf{v}'\mathbf{H}^T \tag{6}$$

where $s_j = \mathbf{v}'\mathbf{h}_j^T$ is the parity-check sum for the j-th parity-check equation of $\mathbf{H}$, $j = 1, 2, ..., M$. It should be noted that the calculation of (6) is over GF(2).

Step 2: Compute a length-N vector $\mathbf{f}$, which is called the reliability profile, according to

$$\mathbf{f} \equiv (f_1, f_2, ..., f_N) = \sum_{j=1}^{M} s_j \mathbf{h}_j \tag{7}$$

where the summation is performed over the real field. The value f_i indicates the number of parity-check equations corresponding to the i-th bit node with a non-zero checksum.

Step 3: Write $S_F = \{f_i | i = 1, 2, \cdots, N\}$. Denote the maximum element in S_F as f_{M1}. Identify the elements of $\mathbf{f}$ which are equal to f_{M1}, and then set all the bits in $\mathbf{p}$ corresponding to those elements to 1. The other bits in $\mathbf{p}$ are set to zero.

The performance of Pcheck-1 algorithm and its variations described later depend on the construction of the parity check matrix (or the associated Tanner graph). It is assumed that all the variable nodes have identical degrees, and there is no cycle of length 4, where the degree of a variable node is the number of check equations which check the node. If the i-th bit node is the only bit of $\mathbf{v}'$ in error, then f_i has the largest value and the $\mathbf{p}$ vector will clearly indicate that p_i is the only component with value equal to 1.

In the case where the number of decoding errors in $\mathbf{v}'$ is more than one, it is possible that the positions of non-zero elements in $\mathbf{p}$ are different from the positions of the errors. In order to increase the probability of correctly identifying error locations, in the literature, there were many variations of the simple BF algorithm (with only one iteration), which are obtained by i) using a preset threshold δ such that p_i is set to 1 for $f_i \geq \delta$, ii) using a weighted reliable profile, or iii) iterative decoding which uses $\mathbf{v}' \oplus \mathbf{p}$ to replace $\mathbf{v}'$ in the next iteration.

In this paper, iterative decoding will be considered for the message passing between the LDPC decoder and the MAP equalizer. In addition, we consider two variations, denoted as Pcheck-2 and Pcheck-3, respectively. Steps 1 and 2 of both Pcheck-2 and Pcheck-3 are the same as those of Pcheck-1. The only difference lies in Step 3. In Pcheck-1, the location of elements in the reliability profile $\mathbf{f}$ with a value of f_{M1} are estimated to be unreliable, where f_{M1} is the maximum of all the elements in S_F. In Pckeck-2, the location of elements in $\mathbf{f}$ with values equal to not only f_{M1}, but also f_{M2}, are considered to be unreliable, where f_{M2} denotes the second largest element in S_F. In Pcheck-3, for each check node j with a non-zero checksum s_j, the location of elements in $\{f_i | i \in W_j\}$ equal to $\max_{i \in W_j}\{f_i\}$ are viewed as unreliable.

A more detailed description for Step 3 of both Pcheck-2 and Pcheck-3 is as follows.

Step 3 of Pcheck-2: Identify the elements of $\mathbf{f}$ equal to either f_{M1} or f_{M2} and then set all the bits in $\mathbf{p}$ corresponding to those elements to 1. The other bits in $\mathbf{p}$ are set to 0.

Step 3 of Pcheck-3:

(a) Initialize j as 1.

(b) When $s_j \neq 0$ for check node j, identify the elements in $\{f_i | i \in W_j\}$ which are equal to

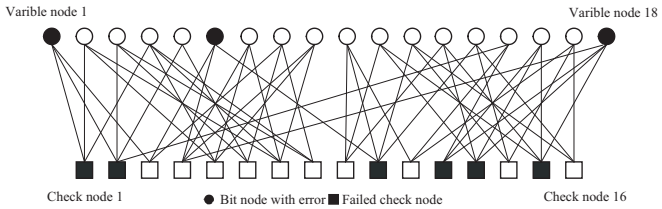


Fig. 7. The Tanner graph used in Example 2.

TABLE I
THE OUTCOMES FOR PCHECK-1, PCHECK-2, AND PCHECK-3.

	Outcome I	Outcome II	Outcome III	Outcome IV
Pcheck-1	93.1423%	3.7654%	0.03055%	3.06175%
Pcheck-2	72.1472%	17.8734%	0.0681%	9.9113%
Pcheck-3	93.4865%	3.34984%	0.0087%	3.15496%

$\max_{i\in W_j}\{f_i\}$ and then set all the bits in **p** corresponding to those elements to 1.

(c) Increase j by 1. If $j > M$, go to Step 3(d). Otherwise, go to Step 3(b).

(d) The bits in **p** that are not assigned with values in Step 3(b) are set to zero.

We illustrate the three methods for obtaining the **p** vectors using the following example.

Example 2: Consider the LDPC code with the Tanner graph shown in Fig. 7. Let **v** be an all zero codeword and $\mathbf{v}' = [1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1]$, meaning that $\mathbf{v}'$ contains 3 bits in error. It can be calculated that $\mathbf{s} = [1,1,0,0,0,0,0,0,\ 0,1,0,1,1,0,1,0]$ and $\mathbf{f} = \mathbf{h}_1 + \mathbf{h}_2 + \mathbf{h}_{10} + \mathbf{h}_{12} + \mathbf{h}_{13} + \mathbf{h}_{15} = [2,1,1,1,1,1,0,0,0,1,2,1,2,1,2,3,0,3]$. Pcheck-1 yields $\mathbf{p} = [0,0,0,0,0,0,0,0,0,0,0,0,\ 0,0,0,0,1,0,1]$ while Pcheck-2 yields $\mathbf{p} = [1,0,0,0,0,0,0,0,0,0,\ 0,1,0,1,0,1,1,0,1]$. It can be seen from Fig. 7 that $W_1 = \{1,2,4\}$, $W_2 = \{1,3,5,15\}$, $W_{10} = \{6,10,12,14\}$, $W_{12} = \{15,16,18\}$, $W_{13} = \{11,13,16,18\}$, and $W_{15} = \{11,13,16,18\}$. Based on Pcheck-3, we have $\mathbf{p} = [1,0,0,0,0,1,0,\ 0,0,1,0,1,0,1,1,1,0,1]$. □

It can be seen from Example 2 that all three methods using the parity check constraint can not correctly identify all the erroneous bits. We need more statistics to evaluate these three methods. Hence, we perform simulation based on the MO channel, RLL constraint and the LDPC codes, as shown in Example 1. We compare the N-bit LDPC codeword **v** with the decoded codeword $\mathbf{v}'$ and observe the obtained **p** vector. Four outcomes, respectively called Outcomes I, II, III and IV, are observed. Outcome I represents the correctly decoded bits where $p_i = 0$, Outcome II represents the incorrectly decoded bits where $p_i = 0$, Outcome III represents the correctly decoded bits where $p_i = 1$ and Outcome IV represents the incorrectly decoded bits where $p_i = 1$. Outcomes I and IV are regarded as consistent outcomes, while Outcomes II and III are regarded as inconsistent. Table I shows the occurrence of these four outcomes for an LDPC-coded recording system using Pcheck-1, Pcheck-2, and Pcheck-3, respectively, at an E_b/N_0 of 7 dB with $U_o = 1$. The associated BER of the decoded codeword $\mathbf{v}'$ is around 10^{-3}.

From Table I, we observe that Pcheck-3 provides the highest percentage of consistent outcomes among the three methods using parity check constraints. A similar trend can be observed for other E_b/N_0 values in the simulation. We will see in the next subsection that Pcheck-3 can also help to obtain the best BER performance by considering the overall detection including using the parity check constraint and the RLL constraint.

B. Enhanced RLL Detection

Using the algorithm presented in [18], the locations of flipped bits $\mathbf{q}'$ can be estimated by searching for a segment of $k+1$ consecutive zeros in $\mathbf{v}'$. The procedure is also described in detail in the first paragraph of Section II-D and is denoted as Rcheck-O in this paper.

With the assistance of **p**, which indicates the unreliable bits of the decoded codeword $\mathbf{v}'$, it is possible to obtain a BER performance better than that presented in [18]. One possible way of employing the RLL constraint for obtaining $\mathbf{q}'$, denoted as Rcheck-I, is simply to replace $\mathbf{v}'$ with $\mathbf{v}' \oplus \mathbf{p}$ before checking the RLL constraint. However, the information for the unreliable bits provided by **p** is far from perfect, as demonstrated in Table I. Consequently, using Rcheck-I results in an obvious error floor, which will be seen in Section IV-C.

Statistics obtained from simulation show that, given $p_i = 1$, the probability that $v_i \neq v_i'$ is not much higher than the probability that $v_i = v_i'$. The high frequency of incorrectly estimated bit flipping is likely to cause error propagation in obtaining $\mathbf{q}'$. In particular, erroneous p_i bits in a cluster, such as the case where p_i and p_j are in error and are within $|i-j| \leq k$, is likely to cause error propagation. To reduce the occurrence of such conditions, we restrict the estimated bit flipping to two patterns. Then, we have Rcheck-II presented below.

Step 1: The window size is set to $k+1$ bits when the $(0,k)$ constraint is considered. Initialize the N-bit location vector $\mathbf{q}'$ with zeros and set i to 1.

Step 2: If $(q_i', q_{i+1}', \cdots, q_{i+k-1}')$ is not the all-zero k-tuple, go to Step 3. Otherwise, v_{i+k}' is viewed as a bit to be flipped if either pattern described in the following is matched.

Pattern I: If both vectors $(v_i', v_{i+1}', \cdots, v_{i+k}')$ and $(p_i, p_{i+1}, \cdots, p_{i+k})$ are the all-zero $(k+1)$-tuple, then set q_{i+k}' to 1.

Pattern II: For $0 \leq l \leq k$, if all the bits in vector $(v_i', v_{i+1}', \cdots, v_{i+k}')$ are zeros except that $v_{i+l}' = 1$, and all the bits in vector $(p_i, p_{i+1}, \cdots, p_{i+k})$ are zeros except that $p_{i+l} = 1$, then set q_{i+k}' to 1. In the case that $l = k$, the *a priori* information for y_{i+k} of the MAP equalizer obtained from the LDPC decoder shown in Fig. 2 is modified as

$$L_a(y_{i+k}) = L_e(v_{i+k}),$$

while the *a priori* information of v_{i+k} of the LDPC decoder obtained from the MAP equalizer is

$$L_a(v_{i+k}) = -L_e(y_{i+k}).$$

TABLE II
AN EXAMPLE ILLUSTRATING THE DETECTION BASED ON BOTH THE PARITY-CHECK CONSTRAINT AND THE RLL CONSTRAINT, WHERE THE (0,3) RLL CONSTRAINT IS CONSIDERED.

$\mathbf{v}$	1	1	1	0	0	0	0	0	0	0	0	0
$\mathbf{v}'$	1	1	1	0	0	0	1	0	1	0	0	0
$\mathbf{p}$	0	1	1	0	0	0	1	0	1	0	0	1
$\mathbf{v}' \oplus \mathbf{p}$	1	0	0	0	0	0	0	0	0	0	0	1
$\mathbf{q}$	0	0	0	0	0	0	1	0	0	0	1	0
$\mathbf{q}'_{Rcheck-O}$	0	0	0	0	0	0	0	0	0	0	0	0
$\mathbf{q}'_{Rcheck-I}$	0	0	0	0	1	0	0	0	1	0	0	0
$\mathbf{q}'_{Rcheck-II}$	0	0	0	0	0	0	1	0	0	0	1	0

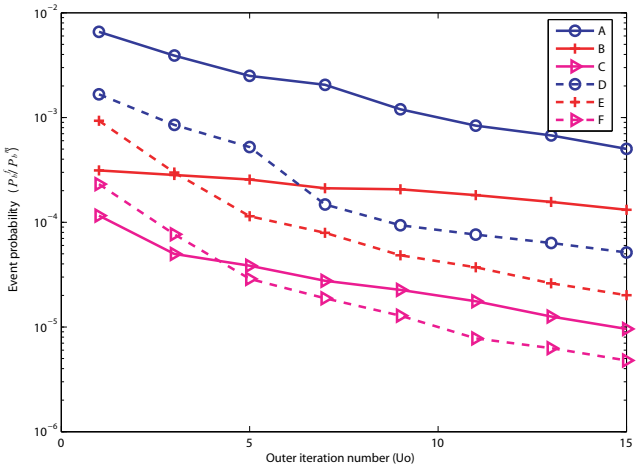


Fig. 8. Detection performance for LDPC-coded recording systems over the MO channel, where β = 0.15, E_b/N_0 = 7.6 dB, (64,64)-regular (4095,3367) EG-LDPC code, and the (0,7) constraint are considered. (A) P_b^f using Rcheck-O; (B) P_b^f using Rcheck-I and Pcheck-3; (C) P_b^f using Rcheck-II and Pcheck-3; (D) P_b^m using Rcheck-O; (E) P_b^m using Rcheck-I and Pcheck-3; (F) P_b^m using Rcheck-II and Pcheck-3.

Step 3 Repeat Step 2 for $i = 2, 3 \cdots, N-k$.

The reason that $L_a(y_{i+k}) = L_e(v_{i+k})$ and $L_a(v_{i+k}) = -L_e(y_{i+k})$ for Pattern II with $l = k$ is given as follows. In this case, we have $(v'_i, v'_{i+1}, \cdots, v'_{i+k}) = (0, \cdots, 0, 1)$, while the the correct version is $(v_i, v_{i+1}, \cdots, v_{i+k}) = (0, \cdots, 0, 0)$. The input to the LDPC decoder should be the corrupted version of $(0, \cdots, 0, 0)$. Hence, the polarity of the extrinsic information fed from the MAP equalizer to the LDPC decoder should be reversed, since $v'_{i+k} = 1$ must be reversed as $v_{i+k} = 0$. On the other hand, the input to the MAP equalizer should be the corrupted version of $(0, \cdots, 0, 1)$. Hence, the polarity of the extrinsic information fed from the LDPC decoder to the MAP equalization remains unchanged, since $v'_{i+k} = 1$ is the bit to be modulated and sent to the channel.

Example 3: Here, we illustrate the possible advantage of Rcheck-II over Rcheck-I. As indicated in Table II, decoding errors for $\mathbf{q}$ are located at the 7th and the 9th positions, respectively. In $\mathbf{p}$, there are 1's located at the 2nd, 3rd, 7th, 9th and 12th positions. Note that the $\mathbf{q}'$ vector obtained using Rcheck-II is less disturbed by the incorrect bits in $\mathbf{v}'$ and $\mathbf{p}$ than that obtained using Rcheck-I. From Table II, we can observe that the $\mathbf{q}'$ vector obtained using Rcheck-O is also not satisfactory. □

Basically, we are interested in correctly detecting the components with values of "1" in the location vector $\mathbf{q}$. The performance of the detector can be evaluated based on the probability of false alarms and missing detections. In the flipping-based system, a false-alarm event is defined as $q_i = 0$ and $q'_i = 1$, and a missing-detection event is defined as $q_i = 1$ and $q'_i = 0$. Let Q_0 (Q_1) denote the index set of variable nodes i where $q_i = 0$ ($q_i = 1$). The probability of a false alarm, denoted as P_b^f, can be calculated from $|\{q'_i = 1 | i \in Q_0\}|/|Q_0|$. Similarly, the probability of a missing detection, denoted as P_b^m, can be calculated from $|\{q'_i = 0 | i \in Q_1\}|/|Q_1|$. Based on the MO channel, the RLL constraint and the LDPC code used in Example 1, we perform simulation for the flipping-based system using Rcheck-O, Rcheck-I and Rcheck-II, respectively, to obtain the data for P_b^f and P_b^m, which are shown in Fig. 8. For Rcheck-I and Rcheck-II, vector $\mathbf{p}$ is obtained using Pcheck-3. It can be seen that as the number of outer iterations U_o increases, the values P_b^f and P_b^m decrease. Using either Rcheck-I or Rcheck-II can obtain lower P_b^f and P_b^m values as compared to using Rcheck-O. Moreover, Rcheck-II is superior to Rcheck-I in obtaining lower values for both P_b^f and P_b^m.

We also perform simulation for which vector $\mathbf{p}$ is obtained using Pcheck-1. Due to space limitations, the associated results are not plotted in any of figures in this paper. However, we observe from the simulated data that the combination of Rcheck-II and Pcheck-3 is superior to the combination of Rcheck-II and Pcheck-1 through the observation of lower values for both P_b^f and P_b^m.

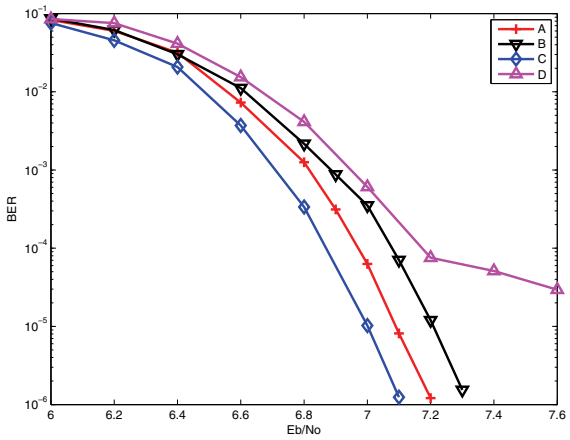


Fig. 9. BER results for LDPC-coded recording systems over the MO channel, where β = 0.15, (64,64)-regular (4095,3367) EG-LDPC code, the Soft-O method, a non-reset LDPC decoder, and the (0,7) constraint are considered. Outer iteration $U_o = 15$. Inner iteration $U_i = 1$. (A) Rcheck-II combined with Pcheck-1; (B) Rcheck-II combined with Pcheck-2; (C) Rcheck-II combined with Pcheck-3; (D) Rcheck-I combined with Pcheck-3.

C. BER Performances

Based on the MO channel, the RLL constraint and the LDPC code used in Example 1, we conducted further simulation to determine the BER performance of the flipped-based system by applying various parity check constraint and the RLL constraint methods, where the non-reset LDPC decoder and the Soft-O method was used. The results are provided in Fig. 9, where it can be seen that using Rcheck-

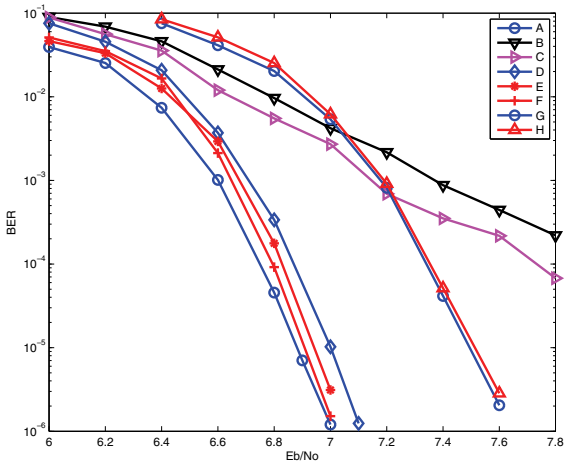


Fig. 10. BER results for LDPC-coded recording systems over the MO channel, where β = 0.15, a non-reset LDPC decoder, and the (0,7) constraint are considered. Outer iteration $U_o = 15$. Inner iteration $U_i = 1$. For Curves (A) to (F), the (64,64)-regular (4095,3367) EG-LDPC code is used while for Curves (G) to (H), the (4,64)-regular (16383,15363) RS-LDPC code is used. (A) Non-flipped system (rate-0.82 code); (B) Flipped system (rate-0.82 code) without using any flipped-bit detection technique; (C) Flipped system (rate-0.82 code) using Soft-O and Rcheck-O [18]; (D) Flipped system (rate-0.82 code) using Soft-O, Rcheck-II, and Pcheck-3; (E) Flipped system (rate-0.82 code) using Soft-III, Rcheck-II and Pcheck-1; (F) Flipped system (rate-0.82 code) using Soft-III, Rcheck-II and Pcheck-3; (G) Non-flipped system (rate-0.9376 code); (H) Flipped system (rate-0.9376 code) using Soft-III, Rcheck-II and Pcheck-3.

I combined with Pcheck-3 (Curve (D)) results in an error floor at about 4×10^{-5}, while using Rcheck-II combined with either Pcheck-1, Pcheck-2 or Pcheck-3 does not result in any error floor. Furthermore, using Rcheck-II combined with Pcheck-3 achieves a BER performance superior to that using either Rcheck-II combined with Pcheck-1 or using Rcheck-II combined with Pcheck-2. The trend of the BER performances shown in Fig. 9 is consistent with the trend implied by the data highlighted in both Table I and Fig. 8.

Finally, we can examine the overall BER performances by integrating the soft information adjustment approach and the improved detection for flipped bits approach, where the MO channel and the RLL constraint used in Example 1 and non-reset LDPC decoder are considered. The results are provided in Fig. 10. We see that by using the combination of Soft-III, Rcheck-II and Pcheck-3, a BER performance (Curve (F)) close to the performance of the ideal case, which is the non-flipped system (Curve (A)) can be achieved. The BER performance when using the combination of Soft-O, Rcheck-II and Pcheck-3 (Curve (D)) is slightly inferior. The disadvantage of using Soft-O as compared to using Soft-III, as illustrated in Fig. 10, is not as significant as that revealed in Fig. 3. This phenomenon is due to the fact that vector $\mathbf{q}'$ used in Fig. 10 is much improved in the sense that the correct rate of the estimated flipped bits is significantly increased. Fig. 10 also shows the BER performance when using the combination of Soft-III, Rcheck-II and Pcheck-1 (Curve (E)). We see that the BER represented by Curve (E) is slightly inferior to the BER represented by Curve (F).

For comparison purposes, the BER performances of the flipped system without using any flipped detection techniques (Curve (B)) and the flipped system using the detection technique (Curve (C)) presented in [18] (i.e., the combination of Soft-O, Rcheck-O without considering $\mathbf{p}$) are also provided in Fig. 10. Obviously, these BER performances are much inferior to those obtained using the proposed techniques.

In order to verify whether the proposed techniques are also effective for LDPC codes with code rates greater than 0.9, we also applied the proposed techniques to a rate-0.9376 $(4, 64)$-regular $(16383, 15363)$ RS-LDPC code [27]. The associated results are also included in Fig. 10. As indicated by Curve (H), using a combination of the the proposed techniques can also achieve a BER performance quite close to that of the non-flipped system (Curve (G)).

V. Conclusions

We have presented two approaches for improving the error performance of a flipping-based system based on the technique presented in [18]. The idea of the first approach is to mitigate the performance degradation resulting from the incorrect detection of flipped bits by adjusting the soft information. The idea of the second approach is to properly employ the RLL constraint and the parity-check constraint of the LDPC code in order to increase the probability of the correct detection of flipped bits. By combining these two approaches, we can achieve error performance which is close to that of the non-flipped system. The results of this research imply that, based on the same coding rate, the BER degradation of the LDPC coded recording system using the RLL constraint can be very minor as compared to the LDPC coded recording system that does not include the RLL constraint if the RLL constraint is not very strict.

References

[1] A. Gallopoulos, C. Heegard, and P. H. Siegel, "The power spectrum of run-length-limited codes," *IEEE Trans. Commun.*, vol. 37, no. 9, pp. 906–917, Sep. 1989.

[2] J. W. M. Bergmans, S. Mita, M. Izumita, and N. Doi, "Partial-response decoding of rate l/2 modulation codes for digital storage," *IEEE Trans. Commun.*, vol. 39, no. 11, pp. 1569–1581, Nov. 1991.

[3] A. R. Calderbank, R. Laroiab, and S. W. McLaughlin, "Coded modulation and precoding for electron-trapping optical memories," *IEEE Trans. Commun.*, vol. 46, no. 8, pp. 1011–1019, Aug. 1998.

[4] B. H. Marcus, P. H. Siegel, and J. K. Wolf, "Finite-state modulation codes for data storage," *IEEE J. Sel. Areas Commun.*, vol. 10, no. 1, pp. 5–37, Jan. 1992.

[5] M. Jin, K. A. S. Immink, and B. Farhang-Boroujeny, "Design techniques for weakly constrained codes," *IEEE Trans. Commun.*, vol. 51, no. 5, pp. 709–714, May 2003.

[6] A. J. van Wijngaarden and K. A. S. Immink, "Construction of maximum run-length limited codes using sequence replacement techniques," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 200–207, Feb. 2010.

[7] J. Moon and L. R. Carley, "Efficient sequence detection for intersymbol interference channels with run-length constraints," *IEEE Trans. Commun.*, vol. 42, no. 9, pp. 2654–2660, Sep. 1994.

[8] K. A. S. Immink and H. D. L. Hollmann, "Prefix-synchronized run-length-limited sequences," *IEEE J. Sel. Areas Commun.*, vol. 10, no. 1, pp. 214–212, Jan. 1992.

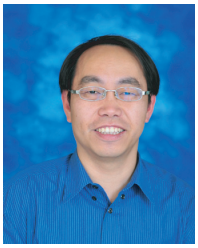
[9] R. G. Gallager, *Low-Density Parity-Check Codes*. MIT Press, 1963.

[10] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 5, no. 2, pp. 399–431, Mar. 1997.

[11] K. A. S. Immink, P. H. Siegel, and J. K. Wolf, "Codes for digital recorders," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2260–2300, Oct. 1998.

[12] H. Pozidis, G. Cherubini, A. Pantazi, A. Sebastian, and E. Eleftheriou, "Channel modeling and signal processing for probe storage channels," *IEEE J. Sel. Areas Commun.*, vol. 2, no. 2, pp. 143–157, Feb. 2010.
[13] K. A. S. Immink, J. Y. Kim, S. W. Suh, and S. K. Ahn, "Iterative correction of intersymbol interference: turbo equalization," *Europ. Trans. Telecommun.*, vol. 6, pp. 507–511, Sep. 1995.
[14] B. Vasic and K. Pedagani, "Run-length-limited low-density parity check codes based on deliberate error insertion," *IEEE Trans. Magn.*, vol. 40, no. 3, pp. 1738–1743, May 2004.
[15] Z. Li and B. V. Kumar, "An improved bit-flipping scheme to achieve run length control in coded systems," *IEEE Trans. Magn.*, vol. 41, pp. 2980–2982, Oct. 2005.
[16] ——, "Low-density parity-check codes with run length limited (RLL) constraints," *IEEE Trans. Magn.*, vol. 42, pp. 344–349, Feb. 2006.
[17] J. Lu and K. Boyer, "Novel RLL-ECC concatenation scheme for high-density magnetic recording," *IEEE Trans. Magn.*, vol. 43, no. 6, pp. 2271–2273, June 2007.
[18] H.-Y. Chen, M.-C. Lin, and Y.-L. Ueng, "Low-density parity-check codes with run length limited (RLL) constraints," *IEEE Trans. Magn.*, vol. 44, pp. 2235–2242, Sep. 2008.
[19] J. L. Fan and A. R. Calderbank, "A modified concatenated coding scheme with applications to magnetic storage," *IEEE Trans. Inf. Theory*, vol. 44, no. 4, pp. 1565–1574, July 1998.
[20] H. Song, B. V. K. V. Kumar, E. Kurtas, Y. Yuan, L. L. McPheters, and S. W. McLaughlin, "Iterative decoding for partial response (PR), equalized, magneto-optical (MO) data storage channels," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 4, pp. 774–782, Apr. 2001.
[21] H.-Y. Chen, H.-F. Chou, M.-C. Lin, and S.-K. Lee, "Capacity approaching run-length-limited codes for multilevel recording systems," *IEEE Trans. Magn.*, vol. 46, no. 1, pp. 95–104, Jan. 2010.
[22] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.
[23] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Trans. Inf. Theory*, vol. 47, pp. 2711–2763, Nov. 2001.
[24] S. Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 40, pp. 1727–1737, Oct. 2001.
[25] L. L. Hanzo, R. G. Maunder, J. Wang, and L.-L. Yang, *Near-Capacity Variable-Length Coding: Regular and EXIT-Chart-Aided Irregular Designs*, 1st edition. Wiley-IEEE Press, 2010.
[26] R. M. Tanner, "A recursive approach to low complexity," *IEEE Trans. Inf. Theory*, no. 5, pp. 533–547, Sep. 1981.
[27] L. Lan, L.-Q. Zeng, Y. Y. Tai, L. Chen, S. Lin, and K. Abdel-Ghaffar, "Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: a finite field approach," *IEEE Trans. Inf. Theory,*, vol. 53, no. 7, pp. 2429–2458, July 2007.

Hong-Fu Chou received the B.S. degree in electrical engineering from National Central University, Jhongli, Taiwan, in 2004. He received the M.S. degree in communication engineering from National Taiwan University, Taipei, Taiwan in 2006. He is working toward the Ph.D. degree in Graduate Institute of Communication Engineering, National Taiwan University, Taipei, Taiwan, Republic of China. His research interests include the communication IC, coding theory and coding for recoding systems.



Yeong-Luh Ueng received the Ph.D. degree in communication engineering from the National Taiwan University, Taipei, Taiwan, R.O.C., in 2001. From 2001 to 2005, he was with a private communication technology company, where he focused on the design and development of various wireless chips. Since December 2005, he has been a member of the faculty of the National Tsing Hua University, Hsinchu, Taiwan, where he is currently an Associate Professor with the Department of Electrical Engineering and the Institute of Communications Engineering. His research interests include coding theory, wireless communications, and communication ICs.

Mao-Chao Lin was born in Taipei, Taiwan, Republic of China, on December 24, 1954. He received the Bachelor degree and Master degree, both in electrical engineering, from National Taiwan University in 1977 and 1979, respectively. He also received the Ph.D. degree in electrical engineering from University of Hawaii in 1986. From 1979 to 1982, he was an assistant scientist of Chung-Shan Institute of Science and Technology at Lung-Tan, Taiwan. He is currently a Professor at the Department of Electrical Engineering, National Taiwan University. His research interests are in the area of coding theory and its applications.

Marc P. C. Fossorier received the B.E. degree from the National Institute of Applied Sciences (I.N.S.A.) Lyon, France in 1987, and the M.S. and Ph.D. degrees in 1991 and 1994, all in electrical engineering. His research interests include decoding techniques for linear codes, communication algorithms and statistics. Dr. Fossorier was a recipient of a 1998 NSF Career Development award and became IEEE Fellow in 2006. He served as Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY from 2003 to 2006, as Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS from 1996 to 2003, as Editor for the IEEE COMMUNICATIONS LETTERS from 1999 to 2007, and as Treasurer of the IEEE Information Theory Society from 1999 to 2003. From 2002 to 2007, he was an elected member of the Board of Governors of the IEEE Information Theory Society which he served as Second and First Vice-President. He was Program Co-Chairman for the 2007 International Symposium on Information Theory (ISIT), the 2000 International Symposium on Information Theory and Its Applications (ISITA) and Editor for the Proceedings of the 2006, 2003 and 1999 Symposium on Applied Algebra, Algebraic Algorithms and Error Correcting Codes (AAECC).

View publication stats