On Array-based LDPC Codes in Channels With Varying Sampling Rate

Lara Dolecek
Venkat Anantharam

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2006-7
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-7.html

January 25, 2006

Acknowledgement

The authors would like to thank Marvell Semiconductor Inc. and California MICRO for supporting their research.

On Array-based LDPC Codes in Channels With Varying Sampling Rate

Lara Dolecek
EECS Department
University of California
Berkeley, CA 94720, USA
Email: dolecek@eecs.berkeley.edu

Venkat Anantharam
EECS Department
University of California
Berkeley, CA 94720, USA
Email: ananth@eecs.berkeley.edu

***Abstract*— We describe a method for enhancing synchronization error correction properties of an array-based LDPC code. The proposed method is based on code expurgation whereby a linear subcode is retained for message encoding and the remaining input bits are used for protection against synchronization errors. The method is easy to implement and incurs a minimal loss in the rate.**

I. Introduction

In many communication systems a binary input message $\mathbf{x}$ is encoded at the transmitter using substitution-error correcting code $C(n,k)$ into a coded sequence $\mathbf{c} = C(\mathbf{x})$. The modulated version of this sequence is corrupted by additive noise and arrives at the receiver as a waveform $r(t)$,

$$r(t) = \sum_i c_i h(t - iT) + n(t), \tag{1}$$

where c_i is the i^{th}bit of $\mathbf{c}$, $h(t)$ is the modulating pulse, and $n(t)$ is the noise introduced in the channel.

Upon receiving $r(t)$, the receiver samples it at time instances $\{kT_s + \tau_k\}$. The sequence of samples is fed into the decoder which produces the most likely input message. Since the traditional decoder is designed to overcome additive noise introduced in the channel, it is essential that the decoder is provided with samples taken at correct time instances. As the operating requirements under which timing recovery must be performed become more stringent, accurate synchronization becomes critical for the full utilization of the available coding gains.

Several authors have studied the problem of accurate timing recovery. Proposed solutions include building a more sophisticated timing recovery block [14], turbo-like approach to iteratively determine both sampling points and encoded data [16], and multiple hypothesis analysis of the sampling instances [11].

As an alternative to more complex and more expensive timing recovery schemes, we propose to employ the timing recovery block as is, and instead modify the decoding procedure and the code itself to compensate for the imperfect synchronization. The advantage of the proposed approach is that with systematically analyzing the robustness of the additive error correction code to synchronization errors, one could use a subcode of the original code that would be immune to both additive as well as sampling errors. The trade off would be in the incurred rate loss associated with the code modification versus the increased complexity and latency associated with the aforementioned approaches. The challenge of the proposed approach lies in determining the synchronization error correction capabilities of the code and in determining its subcode with desired properties.

To emphasize the issues that arise when adequate timing recovery is missing, assume that the modulation scheme in (1) is pulse-amplitude (PAM), and more importantly, that we are operating in the infinite SNR regime where the effect of $n(t)$ is negligible. As a consequence of the initial frequency error, say when $T_s < T$, or of the accumulated phase error in τ_k, some symbol may be sampled more than once (effectively repeated in the infinite SNR regime).

Assuming that the exact sampling instances are not known, a coded sequence $\mathbf{c}$ can give rise to a whole set of received sampled versions of $r(t)$. When two distinct sequences $\mathbf{c_1}$ and $\mathbf{c_2}$ result in the same sampled sequence, it is no longer possible to uniquely determine the coded sequence or its preimage $\mathbf{x}$ from the received sequence, *even in the noise-free environment*. We then say that the code $C(n,k)$ suffers from an *identification problem*. Specifically, let $S_1(\mathbf{c_1})$ be the set of all strings obtained by applying a repetition to $\mathbf{c_1}$. If $S_1(\mathbf{c_1}) \bigcap S_1(\mathbf{c_2})$, for $\mathbf{c_1} \neq \mathbf{c_2}$ is nonempty, we say that $\mathbf{c_1}$ (and $\mathbf{c_2}$) is an identification problem causing codeword.

Several authors have studied codes immune to a deletion or an insertion of a bit. For example, the so-called Varshamov-Tenengolts code proposed in [22] and popularized by Levenshtein in [12] has been further studied by Ferreira et al., [9], Levenshtein [13], Sloane [18], and Tenengolts [20]. Related constructions were proposed in [3], [4], [10] and [21]. Even though these constructions assure that the code is immune to a deletion or an insertion of a bit, they do not guarantee any other desirable properties of standard substitution error correcting codes, such as linearity and a good minimum Hamming distance. Concatenated codes that correct synchronization errors have been proposed in [5] and [6].

In this paper we first present a brief overview of the array-based low density parity check (LDPC) codes and discuss their identification error properties. In Section III we propose a general technique for constructing a code immune to repetitions.

Having established several useful ancillary results in Section IV, we then describe how the array-based LDPC code can be modified to eliminate the identification problem in Section V. A decoding algorithm appropriate for channels with repetition and additive errors is developed in Section VI along with preliminary simulation results on the modified array-based LDPC code.

II. Array-based LDPC codes

Array based LDPC codes are regular LDPC codes parameterized by integers j and p, where $j \leq p$, p is an odd prime, and have the parity check matrix $H_{p,j}$ given by

$$H_{p,j} = \begin{bmatrix} I & I & I & \dots & I \\ I & \sigma & \sigma^2 & \dots & \sigma^{p-1} \\ I & \sigma^2 & \sigma^4 & \dots & \sigma^{2(p-1)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ I & \sigma^{j-1} & \sigma^{(j-1)2} & \dots & \sigma^{(j-1)(p-1)} \end{bmatrix} \quad (2)$$

where σ denotes the following $p \times p$ permutation matrix,

$$\sigma = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}. \quad (3)$$

We let $C_{p,j}$ be the linear code with the parity check matrix given in (2).

Array-based LDPC codes have good performance [8], low encoding complexity [15], are suitable for efficient parallel decoding [2], [17], and have been proposed for a variety of applications, including digital subscriber lines [7] and magnetic recording applications [19], [23]. However, as explained below, such code suffers from an identification problem.

Lemma 1: Under a single repetition, there are at least $2^{p-1} - 2$ *identification problem causing codewords in* $C_{p,j}$ *for* $j < p$.

Proof: Consider two binary strings $\mathbf{c_1}$ and $\mathbf{c_2}$ of length p^2 such that $\mathbf{c_1}$ is $[a_2a_3...a_{p-2}a_{p-1}\overline{a}_pa_1]$ and $\mathbf{c_2}$ is $[a_3a_4...a_{p-1}a_p\overline{a}_1a_2]$ where a_i ($\overline{a}_i$) indicates a string of length p bits with a single 1 (0) in the i^{th} position and 0's (1's) everywhere else.

For example, for $p = 5$, $\mathbf{c_1}$ is [01000 00100 00010 11110 10000] and $\mathbf{c_2}$ is [00100 00010 00001 01111 01000].

First observe that both $\mathbf{c_1}$ and $\mathbf{c_2}$ give rise to the same string $\mathbf{c_{12}}$ after one repetition, where $\mathbf{c_{12}}$ is of length p^2+1, and is $[a_3a_4...a_{p-1}a_p\overline{a}_1a_20]$ (same as $[0a_2a_3...a_{p-2}a_{p-1}\overline{a}_pa_1]$). For example, for $p = 5$, $\mathbf{c_1}$ and $\mathbf{c_2}$ can both result in [00100 00010 00001 01111 010000].

We now prove that $\mathbf{c_1}, \mathbf{c_2}$ are in fact codewords of $C_{p,p-1}$.

Let $\mathbf{c_1}^{\langle kp \rangle}$ denote the string obtained by cyclically shifting $\mathbf{c_1}$ to the right by kp positions.

By the quasi-cyclic property of the code [15], it suffices to verify that $\mathbf{c_1}^{\langle 2p \rangle}=[\overline{a}_pa_1a_2a_3...a_{p-2}a_{p-1}]$ and $\mathbf{c_2}^{\langle 2p \rangle}=[\overline{a}_1a_2a_3a_4...a_{p-1}a_p]$ satisfy $\mathbf{c_1}^{\langle 2p \rangle}H_{p,p-1}^T = 0$ and $\mathbf{c_2}^{\langle 2p \rangle}H_{p,p-1}^T = 0$.

It easily follows that $\mathbf{c_1}^{\langle 2p \rangle}[II...I]^T = 0$. Now consider a row-wise submatrix of $H_{p,p-1}$, $[I\sigma^l\sigma^{2l}\dots\sigma^{l(p-1)}]$ for some l, $1 \leq l \leq p-2$. Write $\mathbf{c_1}^{\langle 2p \rangle}[I\sigma^l\sigma^{2l}\dots\sigma^{l(p-1)}]^T$ as $\overline{a}_p + \sum_{i=1}^{p-1} a_i(\sigma^{il})^T = \overline{a}_p + \sum_{i=1}^{p-1} a_{[i+il]_p}$, where $[x]_p$ indicates x mod p. Since $1 \leq i \leq p-1$ and $1 \leq l \leq p-2$, $i+il$ mod $p \neq 0$, and no term in the summation is a_p. In addition, all terms in the summation are distinct, as otherwise there would exist i, i', $i' < i$ such that $(i-i')(l+1) \equiv 0$ mod p, which is impossible for p prime, $i, i' \leq p-1$ and $l+1 \leq p-1$. Therefore, $\mathbf{c_1}^{\langle 2p \rangle}[I\sigma^l\sigma^{2l}\dots\sigma^{(p-1)l}]^T = 0$. The proof for $\mathbf{c_2}^{\langle 2p \rangle}H_{p,p-1}^T = 0$ is analogous.

Provided that both $\mathbf{c_1}^{\langle kp \rangle}$ and $\mathbf{c_2}^{\langle kp \rangle}$ have the same starting and ending bits, they too result in the identification problem under single repetition. This occurs as long as k is not congruent to 1 or to 2 mod p. Let R_1 (R_2) be the set of codewords obtained by cyclically shifting $\mathbf{c_1}$ ($\mathbf{c_2}$) by kp positions for k ranging from 3 to p. It easily follows that any nontrivial linear combination of elements in R_1 and the same linear combination of their counterparts in R_2 also results in the identification problem.

Since, by construction, $C_{p,j} \supseteq C_{p,j+1}$, in each $C_{p,j}$ code there are therefore at least $2^{p-2}-1$ pairs of codewords causing the identification problem. Since different linear combinations of the elements of R_1 and R_2 result in different codewords, there are therefore at least $2^{p-1}-2$ codewords that cause this problem. ■

III. Construction of a code for correcting multiple repetitions

In this section we show how to construct a code capable of correcting t repetitions. Let us first introduce a useful general transformation, in which we express the number of runs of the original codeword in terms of the weight of a string in the transformed domain.

For $C \subseteq \{0,1\}^n$ let $\tilde{C}$, $\tilde{C} \subseteq \{0,1\}^{n-1}$ be defined as a set of all distinct strings $\tilde{\mathbf{c}}$ obtained by multiplying each $\mathbf{c}$, $\mathbf{c} \in C$ with T_n over $GF(2)$, where T_n is a $n \times (n-1)$ matrix satisfying

$$T_n(i,j) = \begin{cases} 1, & \text{if } i = j, j+1 \\ 0, & \text{else.} \end{cases} \quad (4)$$

If C is a linear code of length n with a generator matrix G, its image under T_n is a linear code generated by $\tilde{G} = GT_n$.

If $\mathbf{c} \in C$ has r runs, then $\tilde{\mathbf{c}} \in \tilde{C}$ has weight $r-1$, and vice versa. Both $\mathbf{c}$ and its complement $\overline{\mathbf{c}}$ (if it exists) result in the same $\tilde{\mathbf{c}}$. In particular, if the code C is linear and the all-ones is not a codeword, the mapping under T_n is one-to-one, and $\tilde{G}$ is full rank.

In essence a repetition in $\mathbf{c} \in C$ corresponds to an insertion of a zero in its counterpart $\tilde{\mathbf{c}} \in \tilde{C}$. Therefore, to construct a code capable of correcting $t = 1$ repetition, it suffices to construct a code immune to a single insertion of a zero. Consider the set $S(m,w,a,r)$ defined as:

$$S(m,w,a,r) = \{ \quad \mathbf{s} = (s_1, s_2, ...s_m) \in \{0,1\}^m : \sum_{i=1}^m s_i = w, \sum_{i=1}^m i s_i \equiv a \bmod r\} \quad (5)$$

and let $S\left(m,(a_1,r_1),(a_2,r_2),...,(a_m,r_m)\right)$ be the union over distinct weights, i.e.

$$S\left(m,(a_1,r_1),(a_2,r_2),...,(a_m,r_m)\right)=\bigcup_{i=1}^{m}S(m,i,a_i,r_i). \tag{6}$$

Lemma 2: Provided that $r_i > i\ \forall i \in [0,m]$, the set $S\left(m,(a_1,r_1),(a_2,r_2),...,(a_w,r_w)\right)$ is single insertion of a zero correcting.

Proof: Since no 1's are introduced in the transmission, it is sufficient to consider the strings having the same weight, i.e. if each set in the disjoint union in (6) is single insertion of a zero correcting, so is their (disjoint) union. Suppose a string $\mathbf{x}$ belonging to $S(m,i,a_i,r_i)$ for $r_i > i$ is transmitted. Its corrupted version (with an inserted zero) arrives at the receiver as a string $\mathbf{x}'$. If the receiver can uniquely determine $\mathbf{x}$ based on $\mathbf{x}'$, for all choices of $\mathbf{x}$ and $\mathbf{x}'$, then the set $S(m,i,a_i,r_i)$ is single insertion of a zero correcting.

Following the analysis of Sloane of a related single deletion correcting code [18], suppose the insertion of a zero occurs in the Lth position, which is unknown. The receiver computes $a' \equiv \sum_{i=1}^{m} i x_i' \bmod r_i$.

$$\begin{aligned} a' &\equiv \textstyle\sum_{i=1}^{m} i x_i' \bmod r_i \\ &\equiv \left(\textstyle\sum_{i=1}^{L-1} i x_i + \sum_{i=L}^{m}(i+1)x_i\right) \bmod r_i \\ &\equiv (a_i + R) \bmod r_i \end{aligned} \tag{7}$$

where R denotes the number of ones to the right of the inserted 0. Since $R \le i < r_i$, the offset R mod r_i can be uniquely determined from a_i and a' mod r_i, and $\mathbf{x}$ is recovered by deleting a zero immediately preceding the Rth 1 in $\mathbf{x}'$ counting from the right. ■

Therefore, if we wish to determine a subcode of a given code C of length n immune to a single repetition, it is sufficient to retain only those codewords which under transformation T_n result in strings that are the elements of $S\left(m,(a_1,r_1),(a_2,r_2),...,(a_m,r_m)\right)$ for $m = n-1$. Specifically, one can let $a_i = a$ and $r_i = r$, where $r-1$ denotes the upper bound on the weight of a string obtained by applying T_n to C. In the remainder we will use $S(m,a,r)$ as a shorthand for such a set.

The construction given in (5) and (6) can be generalized for the correction of multiple repetitions as follows:

Let w denote the weight of $\mathbf{s}$, and define $b_{i+1} = b_{i+1}(\mathbf{s})$, $1 \le i \le w$ to be the size of the run of zeros immediately following the i^{th} 1 in $\mathbf{s}$, and let b_1 be the size of the run zeros immediately preceding the leftmost 1, where by a run of zeros we mean the longest contiguous substring consisting of zeros only. If the i^{th} 1 is immediately followed by another 1, $b_i = 0$, and if the leftmost bit in $\mathbf{s}$ is 1, $b_1 = 0$. Moreover, if $\mathbf{s}$ consists only of zeros, $b_1 =$ length of $\mathbf{s}$. We call b_i the size of the i^{th} bin of zeros of $\mathbf{s}$.

Let $\mathbf{a} = (a_1, a_2, ..., a_t)$, and consider the set $\hat{S}(m,w,\mathbf{a},p)$ defined as

$$\begin{aligned} \hat{S}(m,w,\mathbf{a},p) = \{\ & \mathbf{s} = (s_1,s_2,...s_m) \in \{0,1\}^m : \\ & \textstyle\sum_{i=1}^{m} s_i = w, \\ & \textstyle\sum_{i=1}^{w+1} i b_i \equiv a_1 \bmod p, \\ & \textstyle\sum_{i=1}^{w+1} i^2 b_i \equiv a_2 \bmod p, \\ & \cdots\cdots\cdots \\ & \textstyle\sum_{i=1}^{w+1} i^t b_i \equiv a_t \bmod p\} \end{aligned} \tag{8}$$

and let $\hat{S}\left(m,(\mathbf{a_1},p_1),(\mathbf{a_2},p_2),...,(\mathbf{a_m},p_m)\right)$ be the union over different weights,

$$\hat{S}\left(m,(\mathbf{a_1},p_1),(\mathbf{a_2},p_2),...,(\mathbf{a_m},p_m)\right)=\bigcup_{l=1}^{m}\hat{S}(m,l,\mathbf{a_l},p_l). \tag{9}$$

Lemma 3: If each p_l is prime and $p_l > max(t,l)$, the set $\hat{S}\left(m,(\mathbf{a_1},p_1),(\mathbf{a_2},p_2),...,(\mathbf{a_m},p_m)\right)$ is t-insertions of zeros correcting.

Proof: It suffices to show that each set $\hat{S}(m,l,\mathbf{a_l},p_l)$ is t-insertions of zeros correcting. Suppose a string $\mathbf{x} \in \hat{S}(m,l,\mathbf{a_l},p_l)$ is transmitted. After experiencing t insertions of zeros, it is received as a string $\mathbf{x}'$. We now show that $\mathbf{x}$ is always uniquely determined from $\mathbf{x}'$.

Let $i_1 \le i_2 \le ... \le i_t$ be the (unknown) indices of the bins of zeros that have experienced insertions. For each j, $1 \le j \le t$, compute $a_j' \equiv \sum_{i=1}^{w+1} i^j b_i' \bmod p_l$, where b_i' is the size of the i^{th} bin of zeros of $\mathbf{x}'$,

$$\begin{aligned} a_j' &\equiv \textstyle\sum_{i=1}^{w+1} i^j b_i' \bmod p_l \\ &\equiv a_j + (i_1^j + i_2^j + ... + i_t^j) \bmod p_l, \end{aligned} \tag{10}$$

where a_j is the j^{th} entry in the residue vector $\mathbf{a_l}$ (to lighten the notation the subscript l in a_j is omitted).

By collecting the resulting expressions over all j, and setting $a_j'' \equiv a_j' - a_j \bmod p_l$, we arrive at

$$E_t = \begin{cases} a_1'' \equiv i_1 + i_2 + ... + i_t \bmod p_l \\ a_2'' \equiv i_1^2 + i_2^2 + ... + i_t^2 \bmod p_l \\ \cdots\cdots\cdots \\ a_t'' \equiv i_1^t + i_2^t + ... + i_t^t \bmod p_l. \end{cases} \tag{11}$$

The terms on the right hand side of the congruency constraints are known as power sums in t variables. Let S_k denote the k^{th} power sum mod p_l of $\{i_1, i_2, ..., i_t\}$,

$$S_k \equiv i_1^k + i_2^k + ... + i_t^k \bmod p_l, \tag{12}$$

and let Λ_k denote the k^{th} elementary symmetric function of $\{i_1, i_2, ..., i_t\}$ modp_l,

$$\Lambda_k \equiv \sum_{v_1<v_2<...<v_k} i_{v_1} i_{v_2} \cdots i_{v_k} \bmod p_l. \tag{13}$$

Using Newton's identities over $GF(p_l)$ which relate power sums to symmetric functions of the same variable set, and are of the type

$$S_k - \Lambda_1 S_{k-1} + \Lambda_2 S_{k-2} - ... + (-1)^{k-1}\Lambda_{k-1}S_1 + (-1)^k k\Lambda_k = 0, \tag{14}$$

for $k \leq t$, we can obtain an equivalent system of t equations:

$$\widetilde{E}_t = \begin{cases} d_1 \equiv \sum_{j=1}^{t} i_j \bmod p_l \\ d_2 \equiv \sum_{j<k} i_j i_k \bmod p_l \\ \cdots\cdots\cdots \\ d_t \equiv \prod_{j=1}^{t} i_j \bmod p_l, \end{cases} \tag{15}$$

where each residue d_k is computed recursively from $\{d_1, ..., d_{k-1}\}$ and $\{a_1^{''}, a_2^{''}, ...a_k^{''}\}$. Specifically, since the largest coefficient in (14) is t, and $t < p_l$ by construction, the last term in (14) never vanishes due to the multiplication by the coefficient k.

Consider now the following equation:

$$\prod_{j=1}^{t}(x - i_j) \equiv 0 \bmod p_l, \tag{16}$$

and expand it into the standard form

$$x^t + c_{t-1}x^{t-1} + ... + c_1 x + c_0 \equiv 0 \bmod p_l. \tag{17}$$

By collecting the same terms in (16) and (17), it follows that $d_k \equiv (-1)^k c_{t-k} \bmod p_l$. Furthermore, by Lagrange's Theorem, the equation (17) has at most t solutions. Since $i_t \leq p_l$ all incongruent solutions are distinguishable, and thus the solution set of (17) is precisely the set $\{i_1, i_2, ..., i_t\}$.

Therefore, since the system E_t of t equations uniquely determines the set $\{i_1, i_2, ..., i_t\}$, the locations of the inserted zeros (up to the position within the bin they were inserted in) are uniquely determined, and thus $\mathbf{x}$ is always uniquely recovered from $\mathbf{x}'$. ■

In particular, for $t = 1$, the constructions in (5) and (8) are related as follows.

Lemma 4: For p prime and $p > w$, the set $S(m, w, a, p)$ defined in (5) equals the set $\hat{S}(m, w, \hat{a}, p)$ defined in (8), where $\hat{a} = f_{m,w} - a$ mod p for $f_{m,w} = (w + 2)(2m - w + 1)/2 - (m + 1)$.

Proof: Consider a string $\mathbf{s} = (s_1, s_2, ..., s_m) \in S(m, w, a, p)$, and let p_i be the position of the i^{th} 1 in $\mathbf{s}$, so that $\sum_{i=1}^{m} i s_i = \sum_{i=1}^{w} p_i$. Observe that $p_k = \sum_{i=1}^{k} b_i + k$ where b_i is the size of the i^{th} bin of zeros in $\mathbf{s}$. Write

$$\begin{array}{l} \sum_{i=1}^{w} p_i + (m+1) = (b_1+1) + (b_1+b_2+2) + ...+ \\ (b_1+b_2+...+b_w+w) + (b_1+b_2+...+b_{w+1}+w+1) = \\ \sum_{i=1}^{w+1}(w+2-i)b_i + (w+1)(w+2)/2 = \\ (w+2)(m-w) + (w+1)(w+2)/2 - \sum_{i=1}^{w+1} i b_i = \\ (w+2)(2m-w+1)/2 - \sum_{i=1}^{w+1} i b_i. \end{array} \tag{18}$$

Thus, for $a \equiv \sum_{i=1}^{m} i s_i \bmod p$, the quantity $\hat{a} \equiv \sum_{i=1}^{w+1} i b_i \bmod p$ is $f_{m,w} - a \bmod p$. ■

Before showing in Section V how to systematically modify the array-based LDPC code based on the above approach while preserving the linearity of the parent code, we state several useful technical results.

IV. Auxiliary Results

Let P be the set of binary strings of length $n = p^2$ defined as $P = \{\mathbf{s} : \mathbf{s} = 0^{(p-t)p}1^{tp}$ or $\mathbf{s} = 1^{tp}0^{(p-t)p}\}$ where p is an odd prime, t is an even integer, $1 \leq t \leq p - 1$, and the notation 0^k1^l (1^l0^k) denotes a binary string of length $k + l$ consisting of a run of zeros of size k (run of ones of size l) followed by a run of ones of size l (run of zeros of size k).

Lemma 5: The set P forms a $p - 1$-dimensional set of linearly independent binary strings.

Proof: The proof is by contradiction. Suppose there exists a nontrivial linear combination (simply a sum over $GF(2)$) of elements of P which evaluates to an all-zero string. Let J be a non empty subset of P with the property that the sum of its elements is all zeros. Express J as a disjoint union of J_l and J_r where J_l (J_r) denotes the subset of elements of J which are of the type 1^l0^k (0^k1^l), where l is even and k is odd by construction. The cardinalities of J_l and J_r are both even, with at most one being zero. Let $j_l(j_r)$ be the string obtained by summing the elements of $J_l(J_r)$. If J_l (J_r) is empty, set j_l (j_r) to be an all-zeros string. By construction, if $|J_l| > 0$, j_l has both leftmost and the rightmost run being a run of zeros, with at least one more run (of ones) in between. Moreover, all runs in j_l except for its rightmost run are of even length. Likewise, if $|J_r| > 0$, the string j_r has at least three runs, and both its leftmost and rightmost runs are runs of zeros. The leftmost run in j_r is of odd length, and all other runs are of even length. Irrespective of whether J_l or J_r or neither of the two is empty, j_l and j_r are always different, and thus the sum of elements of J cannot be an all-zeros string. ■

Lemma 6: $\forall \mathbf{s} \in P$, $\mathbf{s}H_{p,j}^T$ = 0, for $H_{p,j}$ given in (2) and $j \leq p$.

Proof: Since $H_{p,j}$ can be viewed as an array of permutation submatrices, each row of $H_{p,j}$ can be viewed as a concatenation of p substrings, where each such substring is of the type a_i, for a_i defined in Section II. Let $\mathbf{h}$ be a row of $H_{p,j}$. As $\mathbf{s}$ multiplies $\mathbf{h}^T$, the contribution to the product is either 1 or 0 per constituent substring, depending on whether the multiplying constituent substring in $\mathbf{s}$ consists of all ones or all zeros. Since the number of constituent substrings of $\mathbf{s}$ equal to all ones is always even by construction, the product $\mathbf{s}\mathbf{h}^T$ is zero for all $\mathbf{s} \in P$. ■

For $j < p$, as a consequence of the previous two Lemmas we can form a generator matrix $G_{p,j}$ with rank K where K is $p(p-j)+j-1$ ([15]) of the array-based LDPC code $C_{p,j}$, such that

$$G_{p,j} = \begin{bmatrix} G_p^s \\ G_{p,j}^m \end{bmatrix}, \tag{19}$$

where G_p^s is a $p-1 \times p^2$ matrix whose rows are all distinct elements of the set P. By applying only row manipulations to a generator matrix, the matrix $G_{p,j}^m$ (which is $K - p + 1 \times p^2$ and thus non empty for $j < p$) has each qp-th column, for $1 \leq q < p$, equal to the $(qp + 1)$-th column. Let $\tilde{G}_{p,j} = G_{p,j}T_{p^2}$ where T_{p^2} is given by (4), and observe that the top $p-1$ rows of $\tilde{G}_{p,j}$ are all distinct and are of the form $0^{tp-1}10^{p^2-tp-1}$, for $1 \leq t \leq p-1$, and that the bottom $K - p + 1$ rows have zeros in all positions that are multiples of p.

Let $\tilde{C}_{p,j}$ be the code generated by $\tilde{G}_{p,j}$. Since the all-ones string is not a codeword in $C_{p,j}$, the matrix $\tilde{G}_{p,j}$ has full rank.

Lemma 7: No codeword in $\tilde{C}_{p,j}$ has weight p^2-1 or p^2-2.

Proof: Suppose first that there exist a codeword $\mathbf{c} \in Cp, j$ for which $\mathbf{c}T_{p^2} = \tilde{\mathbf{c}}$ has weight p^2-1. It is then necessary that $\mathbf{c}$ itself consists of alternating bits. Since the top part of $H_{p,j}$ (cf. (2)) consists of an odd-sized array of identity matrices it is impossible to have $\mathbf{c}H_{p,j}^T = 0$.

Now suppose that there is a codeword $\mathbf{c}$ which produces $\mathbf{c}T_{p^2} = \tilde{\mathbf{c}}$ where $\tilde{\mathbf{c}}$ has weight p^2-2. It is then necessary that $\mathbf{c}$ has all runs of size 1 except for exactly one run of size 2. Suppose that the 2-bit run spans positions r and $r+1$. By the quasi-cyclic property of the code we can assume that $1 \leq r \leq p$. Let $\mathbf{c_l}$ denote the substring of $\mathbf{c}$ consisting of p leftmost bits, and let $\mathbf{c_r}$ denote the substring of $\mathbf{c}$ spanning the remaining $p+1$ through p^2 positions. Write $\mathbf{c}[II\dots I]^T = [\mathbf{c_l}\mathbf{c_r}][II\dots I]^T = \mathbf{c_l}$, and observe that the contribution from $\mathbf{c_r}$ to the overall result is either an all-ones string of length p (if $p-1 \equiv 2 \bmod 4$) or is an all-zeros string of length p (if $p-1 \equiv 0 \bmod 4$). Since $\mathbf{c_l}$ is neither all-zeros nor all-ones, $\mathbf{c}[II\dots I]^T \neq 0$, and $\mathbf{c}$ is not a codeword. ■

We conclude the section with the following result.

Lemma 8: For p an odd prime, the set $S = \{1, p+1, 2p+1, \dots, (p-1)p+1, p^2+1\}$ generates the complete residue class mod p^2 in that each $0 \leq b \leq p^2-1$ can be written as a sum of distinct elements of S mod p^2.

Proof: Write b, $0 \leq b \leq p^2-1$ as $b = tp+d \bmod p^2$ where $0 \leq t \leq p-1$ and $1 \leq d \leq p$. It is sufficient to show that each t, $0 \leq t \leq p-1$, can be written as a linear combination mod p of d distinct elements of the set $N_p = \{0, 1, 2, ..., p\}$ for each d, $1 \leq d \leq p$.

If indeed $t \equiv \sum_{i=1}^{d} n_i \bmod p$ for n_i being distinct elements of N_p, it follows that $tp \equiv \sum_{i=1}^{d} n_i p \bmod p^2$ and $tp+d \equiv \sum_{i=1}^{d} n_i p + d \bmod p^2 = \sum_{i=1}^{d}(n_i p+1) \bmod p^2$, where each term $n_i p+1$ in the last sum is a distinct element of S.

Let $R_{a,d}$ be the sum of d consecutive integers each taken mod p, starting with a mod p. For $1 \leq d \leq p-1$, it suffices to show that the entries in the p-term sequence

($R_{1,d}$, $R_{2,d}$, ... , $R_{p,d}$) = ($R_{1,d}$, $R_{1,d}+d$, ... , $R_{1,d}+(p-1)d$)

are all distinct mod p. Suppose there exist distinct k, l, $0 \leq k, l \leq p-1$ for which $R_{1,d}+kd \equiv R_{1,d}+ld \bmod p$. Then $(k-l)d \equiv 0 \bmod p$, which is impossible for p prime and $d < p$.

For $d = p$, consider the p-term sequence

($R_{0,p+1}-0$, $R_{0,p+1}-1$, ... , $R_{0,p+1}-(p-1)$), where each term in the sequence is a sum of d distinct elements of N_p. Since $0, 1, ..., p-1$ are distinct mod p, so are all terms in this sequence. ■

V. Modified Encoding

Suppose the code $C_{p,j}$ with the generator matrix $G_{p,j}$ given in (19) and of rank K is used for transmission. Let $\mathbf{m_u}$ be the binary string of length $(K-p+1)$ bits provided by the user. Denote by $\mathbf{m_s}$ an auxiliary binary string of length $p-1$. Let $\mathbf{c} = [\mathbf{m_s}\mathbf{m_u}]G_{p,j}$ be the resulting codeword, and let s_1 and s_2 be additional auxiliary single bits.

The values of $\mathbf{m_s}$, s_1 and s_2 are chosen such that

$$f(\tilde{\mathbf{v}}) = \sum_{i=1}^{p^2+1} i\tilde{v}_i \equiv a \bmod p^2, \tag{20}$$

is satisfied for some arbitrary but fixed constant a, where $\tilde{\mathbf{v}}$ is defined as $[s_1\mathbf{c}s_2]T_{p^2+2}$ for T_{p^2+2} given in (4).

Discussion: By Lemma 7, the string $\tilde{\mathbf{v}}$ in (20) has weight at most p^2-1, and thus belongs to the set $S(p^2+1, a, p^2)$, where $S(p^2+1, a, p^2)$ is defined in Section III.

To show that it is always possible to find the appropriate values of s_1, s_2 and $\mathbf{m_s}$ such that (20) holds, let $\tilde{\mathbf{u}}$ be $[0^{p-1}\mathbf{m_u}]\tilde{G}_{p,j}$ and let $a' \equiv \sum_{i=1}^{p^2-1}(i+1)\tilde{u}_i \bmod p^2$. Also let $\tilde{\mathbf{s}}$ be $[\mathbf{m_s}0^{K-p+1}]\tilde{G}_{p,j}$, where $\tilde{\mathbf{c}} = \tilde{\mathbf{u}} + \tilde{\mathbf{s}}$, for $\tilde{\mathbf{c}} = \mathbf{c}T_{p^2}$. By construction every entry in $\tilde{\mathbf{u}}$ in position whose index is a multiple of p is precisely zero, and the only non-zero entries in $\tilde{\mathbf{s}}$ are in positions with indices that are multiples of p. Expand $f(\tilde{\mathbf{v}})$ as

$$\begin{aligned} f(\tilde{\mathbf{v}}) &= \hat{s}_1 + \textstyle\sum_{i=1}^{p^2-1}(i+1)\tilde{c}_i + (p^2+1)\hat{s}_2 \\ &= \hat{s}_1 + \textstyle\sum_{i=1}^{p^2-1}(i+1)\tilde{u}_i + \sum_{i=1}^{p^2-1}(i+1)\tilde{s}_i + \\ &\quad (p^2+1)\hat{s}_2 \end{aligned} \tag{21}$$

Then, $f(\tilde{\mathbf{v}}) \equiv a' + \sum_{i=0}^{p}(ip+1)z_i \bmod p^2$

where $\mathbf{z} = [\hat{s}_1\mathbf{m_s}\hat{s}_2]$, and $\hat{s}_1 = s_1 + c_1$, $\hat{s}_2 = s_2 + c_{p^2}$.

It follows by Lemma 8 that irrespective of the value a' (determined by the user's input message) it is always possible to choose $\mathbf{z}$ such that $f(\tilde{\mathbf{v}}) \equiv a \bmod p^2$. ■

Therefore, by reducing the dimensionality of the input space by $p-1$ and introducing the additional 2 guard bits, we are able to insure that the transmitted codeword (plus the guard bits) does not suffer from the identification problem. The overall rate loss is then $\frac{K}{p^2} - \frac{K-(p-1)}{p^2+2}$, which is asymptotically zero as the blocklength tends to infinity.

VI. Modified decoding algorithm

We conclude the paper with the outline of the message passing decoding algorithm appropriate for channels with varying sampling rate.

Suppose the sequence y of length $n+1$ bits is received as a result of transmitting a codeword of length n bits through a noisy channel, followed by a repetition error. For each coded bit x_i we wish to compute $P(x_i|y_1^{n+1})$. We introduce auxiliary variables G, which takes values $\in \{1, ..., n\}$, and L_i, for $\forall\, i \in [1, n]$, such that $L_i \in \{-1, 0, 1\}$. The variable G denotes the position of the repetition, and L_i denotes the relative location of the ith bit with respect to the repetition.

Write

$$P(x_i|y_1^{n+1}) = \sum_{G}\sum_{L_1^n}\sum_{x_1^n \setminus x_i} P(x_1^n, L_1^n, G|y_1^{n+1}). \tag{22}$$

Group the variables as shown in Fig. 1., for $1 \leq i \leq n$ and $1 \leq k \leq M$, where M is the total number of checks. Note that y_1^{n+1} is treated as evidence and not as variables.

The junction graph corresponding to these local domains is shown in Fig. 2, and has the bidirectional edges between:

local domain	local function $\varphi(\cdot)$
$\{G\}$	1
$\{G, L_i\}$	$1\,[L_i = 1 \cdot 1(L \leq i-1) +$ $0 \cdot 1(L = i) + (-1) \cdot 1(L \geq i+1)]$
$\{L_i, x_i\}$	$P(y_i\|x_i)1(L_i = -1) +$ $P(y_i\|x_i)P(y_{i+1}\|x_i)1(L_i = 0) + P(y_{i+1}\|x_i)1(L_i = 1)$
$\{x_i\}$	1
$\{c_k, (x_j, j \in \mathcal{N}_k)\}$	$1(c_k = \oplus_{j \in \mathcal{N}_k} x_j)$

Fig. 1. Local domains and functions.

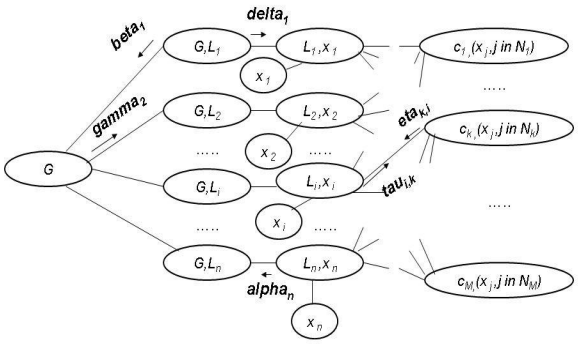

Fig. 2. Junction graph.

- $\{G\}$ and $\{G, L_i\}$ for each $1 \leq i \leq n$,
- $\{G, L_i\}$ and $\{L_i, x_i\}$ for each $1 \leq i \leq n$,
- $\{L_i, x_i\}$ and $\{c_k, (x_i, i \in \mathcal{N}_k)\}$, for each pair (i, k) such that $i \in \mathcal{N}_k$, and
- $\{x_i\}$ and $\{L_i, x_i\}$ for each $1 \leq i \leq n$.

Observe that

$$P(x_1^n, L_1^n, G|y_1^{n+1}) \propto \varphi(G) \prod_i \varphi(G, L_i) \prod_i \varphi(L_i, x_i) \prod_i \varphi(x_i) \prod_k \varphi(c_k, (x_j, j \in \mathcal{N}_k)). \tag{23}$$

We may thus use a message passing algorithm as in [1] to try to find $P(x_i|y_1^{n+1})$.

Let all messages be initialized to 1, and let $\alpha_j(L_j)$ be the message sent from $\{L_j, x_j\}$ to $\{G, L_j\}$ at some stage.

The message $\beta_j(G)$ from $\{G, L_j\}$ to $\{G\}$ is then $\beta_j(G) = \sum_{L_j} \varphi(G, L_j)\alpha_j(L_j)$, and the message $\gamma_i(G)$ sent from $\{G\}$ to $\{G, L_i\}$ is $\prod_{j \in \{1,n\} \setminus i} \beta_j(G)$. Finally, the message from $\{G, L_i\}$ to $\{L_i, x_i\}$ is $\delta_i(L_i) = \sum_G \varphi(G, L_i)\gamma_i(G)$.

The message $\tau_{j,k}(x_j)$ sent from $\{L_j, x_j\}$ to $\{c_k(x_j, j \in \mathcal{N}_k)\}$ is $\sum_{L_j} \varphi(L_j, x_j)\delta_j(L_j) \prod_{l \in \mathcal{N}_j \setminus k} \eta_{l,j}(x_j)$, where $\eta_{l,j}(x_j)$ is the message from $\{c_l, (x_j, j \in \mathcal{N}_l)\}$ to $\{L_j, x_j\}$ and is $\sum_{x_i, i \in \mathcal{N}_l \setminus j} \varphi(c_l, (x_i, i \in \mathcal{N}_l)) \prod \tau_{i,l}(x_i)$.

The message $\alpha_j(L_j)$ is updated to $\sum_{x_j} \varphi(L_j, x_j) \prod_{k \in \mathcal{N}_j} \eta_{k,j}(x_j)$, and message exchange continues as above.

As a result, from (22) one gets

$$P(x_i|y_1^{n+1}) \approx \sum_{L_i} \varphi(L_i, x_i)\delta_i(L_i) \prod_{k \in \mathcal{N}_i} \eta_k(x_i), \tag{24}$$

and we also have

$$P(G = g|y_1^{n+1}) \approx \prod_{i=1}^{n} \beta_i(g). \tag{25}$$

Even though there are $O(n)$ computations each involving $O(n)$ variables per global iteration step, computational complexity can be reduced from $O(n^2)$ to $O(n)$ with organized calculations, as we now show.

We have

$$\beta_i(g) = \begin{cases} \alpha_i(1), & \text{for } g \leq i-1 \\ \alpha_i(0), & \text{for } g = i \\ \alpha_i(-1), & \text{for } g \geq i+1 \end{cases} \tag{26}$$

and

$$\gamma_i(g) = \prod_{\hat{i} \neq i} \beta_{\hat{i}}(g) \ . \tag{27}$$

The message $\delta_i(L_i)$ is

$$\delta_i(L_i) = \begin{cases} \sum_{g \geq i+1} \gamma_i(g), & \text{for } L_i = -1 \\ \gamma_i(i), & \text{for } L_i = 0 \\ \sum_{g \leq i-1} \gamma_i(g), & \text{for } L_i = 1 \end{cases} \tag{28}$$

Now the trick is to directly compute all the δ_i directly from the α_i.

Let us write

$$\begin{aligned} \delta_i(-1) &= \sum_{g \geq i+1} \gamma_i(g) \\ &= \sum_{g \geq i+1} \prod_{\hat{i} \neq i} \beta_{\hat{i}}(g) \\ &= \sum_{g \geq i+1} \frac{B(g)}{\beta_i(g)} \\ &= \frac{\sum_{g \geq i+1} B(g)}{\alpha_i(-1)} \ , \end{aligned} \tag{29}$$

where

$$\begin{aligned} B(g) &= \prod_i \beta_i(g) \\ &= \left(\prod_{i \leq g-1} \beta_i(g)\right) \beta_g(g) \left(\prod_{i \geq g+1} \beta_i(g)\right) \\ &= \left(\prod_{i \leq g-1} \alpha_i(-1)\right) \alpha_l(0) \left(\prod_{i \geq g+1} \alpha_i(1)\right) \ . \end{aligned} \tag{30}$$

Similarly,

$$\begin{aligned} \delta_i(0) &= \gamma_i(i) \\ &= \prod_{\hat{i} \neq i} \beta_{\hat{i}}(i) \\ &= \frac{B(i)}{\beta_i(i)} \\ &= \frac{B(i)}{\alpha_i(0)} \ , \end{aligned} \tag{31}$$

and

$$\begin{aligned} \delta_i(1) &= \sum_{g \leq i-1} \gamma_i(g) \\ &= \sum_{g \leq i-1} \prod_{\hat{i} \neq i} \beta_{\hat{i}}(g) \\ &= \sum_{g \leq i-1} \frac{B(g)}{\beta_i(g)} \\ &= \frac{\sum_{g \leq i-1} B(g)}{\alpha_i(1)} \ . \end{aligned} \tag{32}$$

Given α_i for $1 \leq i \leq n$, we can compute $\prod_{i \leq g-1} \alpha_i(-1)$ and $\prod_{i \geq g+1} \alpha_i(1)$ for $1 \leq g \leq n$ in $O(n)$ steps and from these and the $\alpha_g(0)$, $1 \leq l \leq n$, we can compute the $B(g)$, $1 \leq g \leq n$ in another $O(n)$ steps. Now, from the α_i, $1 \leq i \leq n$ we can compute the δ_i, $1 \leq i \leq n$, in another $O(n)$ steps.

The belief at node $\{G\}$ is the vector $\prod_i \beta_i$ whose coordinates are the $B(g)$, $1 \leq g \leq n$, and are already computed.

The messages $\tau_{j,k}$ and $\eta_{k,j}$ are analogous to messages computed in a traditional message passing algorithm on a bipartite graph, so their complexity is also $O(n)$.

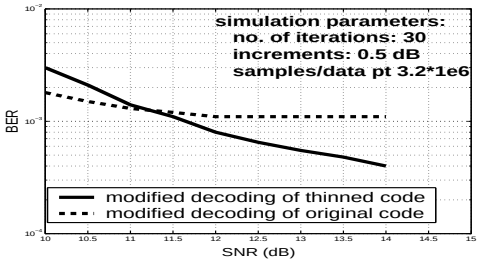

Fig. 3. Performance of LDPC (25,12) over AWGN with one repetition

When a code of interest, such as the array-based LDPC code, suffers from an identification problem, we replace standard encoding with that of Section V, and impose the overall congruency constraint (21) on the encoded binary string.

For illustration purposes we provide simulation results in Fig 2. from an earlier version of the presented decoding algorithm.

VII. Concluding remarks

In this paper we presented a technique for modifying array-based LDPC codes for channels in which the varying sampling rate causes repetition of symbols. While suffering a minimal loss in the rate, the proposed technique systematically expurgates the code such that the resulting code has significantly improved synchronization error correction properties. Besides array based LDPC codes, the presented methodology shows promise for a wider range of permutation matrix based LDPC constructions. Future work involves extending the proposed method to other LDPC families as well as incorporating the correction capabilities for multiple synchronization errors.

Acknowledgment

The authors would like to thank Marvell Semiconductor Inc. and California MICRO for supporting their research.

References

[1] S. Aji and R. McEliece, “The generalized distributive law”, *IEEE Trans. on Information Theory* vol. 46(2), pp. 325–343, March 2000.

[2] P. Bhagawat, M. Uppal and G. Choi, ”FPGA based implementation of decoder for array low-density parity check codes”, In *Proc. of ICASSP 2005*, Philadelphia, PA, USA.

[3] P.A.H. Bours, ”Construction of fixed-length insertion/deletion correcting runlength-limited code”, *IEEE Trans. on Information Theory* vol. 40(6), pp. 1841–1856, Nov. 1994.

[4] L. Calabi and W.E. Hartnett, ”A family of codes for the correction of substitution and synchronization errors”, *IEEE Trans. on Information Theory* vol. 15(1), pp. 102–106, Jan. 1969.

[5] G. Chen, M. Mitzenmacher, C. Ng and N. Varnica, ”Concatenated codes for deletion channels,” In *Proc. of the IEEE International Symposium on Information Theory* 2003, Yokohama, Japan.

[6] M.C. Davey and D.J.C. MacKay, ”Reliable communication over channels with insertions, deletions and substitutions”, *IEEE Trans. on Information Theory* vol. 47(2), pp. 687-698, Feb. 2001.

[7] E. Eleftheriou and S. Ölçer, ”Low density parity check codes for digital subscriber lines”, *Proceedings of the IEEE International Conference on Communications* 2002, New York, NY, pp. 1752–1757.

[8] J. L. Fan, ”Array-codes as low-density parity-check codes”, in *Proc. of the 2nd International Symposium on Turbo Codes and Related Topics*, Brest, France, Sept. 2000.

[9] H.C. Ferreira, W.A. Clarke, A.S.J. Helberg, K.A.S. Abdel-Gaffar and A.J. Han Vinck, ”Insertion/deletion correction with spectral nulls”, *IEEE Trans. on Information Theory* vol. 43, pp. 722–732, March 1997.

[10] T. Kløve ”Codes correcting a single insertion/deletion of a zero or a single peak-shift”, *IEEE Trans. on Information Theory* vol. 41(1), pp. 279–283, Jan. 1995.

[11] P. Kovintavewat, J. R. Barry, M. F. Erden and E. Kurtas, “Per-survivor timing recovery for uncoded partial response channels,”*Proceedings of the IEEE International Conference on Communication*s 2004, Paris, France.

[12] V. I. Levenshtein, ”Binary codes capable of correcting deletions, insertions and reversals”, *Sov. Phys.-Dokl.*, vol. 10(8), pp. 707–710, Feb. 1966.

[13] V. I. Levenshtein, ”On perfect codes in deletion and insertion metric”, *Discrete Math. Appl.*, vol. 2(3), pp. 241–258, 1992.

[14] J. Liu, H. Song and B.V.K.V. Kumar, “Symbol timing recovery for low-SNR partial response recording channels,” In *Proc. GLOBECOM 2003*, Nov. 2002, pp. 1141 – 1145, San Francisco, CA, USA.

[15] T. Mittelholzer, “Efficient encoding and minimum distance bounds of Reed-Solomon-type array codes”, In *Proc. of the IEEE International Symposium on Information Theory* 2003, p. 282, Lausanne Switzerland.

[16] A. R. Nayak, J. Barry and S. W. McLaughlin, “Joint timing recovery and turbo equalization for coded partial response channels,” *IEEE Trans. On Magnetics*, vol. 38(5), pp. 2295–97, Sept. 2002.

[17] S. Ölçer, ”Decoder architecture for array-code based LDPC codes”, In *Proc. GLOBECOM 2003*, Dec. 2003, San Francisco, CA, USA.

[18] N.J.A. Sloane, ”On single deletion correcting codes”, 2000. Available at http://www.research.att.com/˜njas/doc/dijen.pdf

[19] H. Song and B.V.K. Kumar, ”Low density parity check codes for partial response channels”, *IEEE Signal Processing Magazine*, vol. 21(1), pp. 56–66, Jan 2004.

[20] G. Tenengolts, ”Nonbinary codes correcting single deletion or insertion”, *IEEE Trans. on Information Theory* vol. 30, pp. 766–769, Sept. 1984.

[21] J.D. Ullman, ”Near optimal single synchronization error correcting code”, *IEEE Trans. on Information Theory* vol. 26(2), pp. 288–292, 1965.

[22] R. R. Varshamov and G.M. Tenengolts, “Codes which correct single asymmetric errors,” *Avtomatika i Telemehkanika*, vol. 26, no. 2, pp. 288–292,1965.

[23] B. Vasic and E. Kurtas, ”Coding and Signal Processing for Magnetic Recording Systems”, CRC press, 2005.