Computing and Informatics, Vol. 42, 2023, 157–190, doi: 10.31577/cai_2023_1_157

ADDING RLL PROPERTIES TO FOUR CCSDS LDPC CODES WITHOUT INCREASING THEIR REDUNDANCY

Peter Farkaš

Institute of Multimedia ICT, FEI STU
Ilkovičova 3
812 19 Bratislava, Slovakia
&
Institute of Applied Informatics
Faculty of Informatics, Pan-European University
Tematínska 10, 851 05 Bratislava, Slovakia
e-mail: p.farkas@ieee.org

Martin Rakús

Institute of Multimedia ICT, FEI STU
Ilkovičova 3
812 19 Bratislava, Slovakia
e-mail: martin.rakus@stuba.sk

Abstract. This paper presents the construction of Run Length Limited (RLL) Error Control Codes (ECCs) from four Low Density Parity Check (LDPC) Codes specified by Consultative Committee for Space Data Systems (CCSDS). The obtained RLL-ECCs present a practical alternative to the CCSDS codes with pseudo-randomizers. Their advantage is that the maximal runlengths of equal symbols in their codeword sequences are guaranteed, which is not the case if the common approach with pseudo-randomizers is used. The other advantages are that no additional redundancy is introduced into encoded codewords and that the encoding and decoding procedures of the original error control CCSDS codes do not have to be modified in the following cases. In the first case if hard decoding is used and the transmission channel can be modeled as a Binary Symmetric Channel (BSC) or in the second case if soft decoding and coherent Binary Phase Shift Keying (BPSK)

modulation is used and the appropriate transmission channel model is an Additive White Gaussian Noise (AWGN) channel.

Keywords: ECC codes, CCSDS codes, LDPC codes, RLL codes, modifier, reordering, redundancy, hard decoding, soft decision decoding, BSC, AWGN

Mathematics Subject Classification 2010: 94B05, 93-04, 94B10

1 INTRODUCTION

By introducing redundancy the purpose of ECCs is to protect transmitted or stored information against errors. The introduction of redundancy by line codes (LCs) on the other hand has the purpose to meet constraints or practical requirements of transmission channels or storage medium (therefore they are also named Constrained codes).

The construction methods of the ECCs [1, 2, 3, 4] and LCs [5, 6, 7, 8, 9, 10, 11, 12, 13, 14] is, in most cases, treated in literature separately. This tradition also continues nowadays. Examples of recent advances in constructing ECCs are found in [15, 16, 17, 18, 19, 20], while the up to date progress examples in constructing LCs are found in [21, 22, 23, 24, 25].

However, there is a fundamental problem how to order the Constrained codes and ECC in cascade. The question is which code should be the inner code and which the outer. It stems from the fact that Constrained codes are not suitable for error prone channels. In theory it is supposed that the constrained channels for which they are used are error free. In practice this assumption is not true and therefore ECCs are often used as inner codes. In this case the black box containing them and the real channel could be approximated in some cases as errorless. But the ECCs are not constructed with the goal to fulfill the constraints of the channels. From this point of view it seems that the last encoder and the first decoder connected to the real channel with constraints (the inner code) should be the Constrained code and not the ECC.

Under certain conditions a combined code that has error control capabilities as well as constraints can be obtained, which overcomes this problem. There are many methods how to construct such combined codes. Selected examples are [26, 27, 28, 29, 30, 31, 32, 33, 34]. One of the approaches to the construction of such combined codes is based on coset codes of linear ECCs. Because this paper follows this particular direction, in the following section the state of the art of this branch of development is given. The combined codes are named Transcontrol codes in [35].

2 STATE OF THE ART OF TRANSCONTROL CODES CONSTRUCTIONS BASED ON COSET CODES

The idea to use a coset code of a linear ECCs is not new. In [36] Lee and Heegard demonstrated that it is possible to limit the run length of zeros by adding a periodic fixed binary sequence to each convolutional code's codeword. The purpose was to improve clock synchronization. Calderbank et al. in [37] used this idea for the same purpose in order to construct codes for the partial response channel with transfer function $(1 - D^N)/2$. Ferreira et al. and Deng and Herro presented in [38, 39] DC-free coset codes with and without error correcting capabilities for fiber-based optical links. The codes were derived by partitioning linear block codes. The encoder and decoder structures remained the same as those of linear block codes with only slight modifications. A special class of DC-free coset codes were derived from BCH codes with specified bounds on minimum distance, disparity, and run length. The running disparity at the end of the actual codeword was used for controlling the selection of the next codeword in encoding and decoding procedures. Some detailed statements made in [38, 39] relating to disparity and runlength have been corrected and/or refined in [40]. Poplewell et al. in [41] exploited disparity to control whether the whole following codeword will be inverted or not in order to decrease the DC component in the multilevel transmitted sequence. In this paper an improved procedure exploiting disparity was also proposed which decreases low frequency spectral components in such sequences. In both procedures additional redundancy was introduced into the resulting transmitted sequence. In [42] two techniques for constructing DC-free codes with minimum distance 4 were presented. These constructions resulted in very low complexity implementation and allowed both hard and soft decision decoding to be applied. Construction 1 was based on adapting an algorithm developed for constructing a class of low complexity RLL-ECCs having a maximum runlength constraint of 6 and minimum distance 4. Construction 2 was based on utilizing simple array codes based on two parity check codes and mapping balanced words onto each row of the array (the same mapping was applied to every row) in place of the even parity word. In [35] it was proven that if the generator matrix of a linear binary block code exhibits certain properties then RLL coset codes of the corresponding ECCs could be obtained by the addition of a specific vector denoted as a modifier. The main difference when compared with previous results was that not only zero runs could be limited by this approach but also the runs of ones. A related property useful for construction of RLL-ECCs was identified later in [43], also in the parity check matrix.

Note 1. The Properties described in [35, 43] are closely related to the approaches used in this paper, therefore for convenience of the reader they will be completely repeated in Section 4.

Later different approaches were developed based on these properties and applied to numerous different ECCs [43, 44, 45, 46, 47, 48, 49, 50, 51, 52]. Recently it was

shown that related techniques could also be applied on standardized LDPC codes for 5G mobile networks [53]. Therefore a natural question arises as to whether there are also other standardized LDPC codes for which it could lead to obtaining RLLL ECCs.

In this paper it is shown that at least for the following four CCSDS codes the answer is positive:

- Three LDPC codes specified by CCSDS for the Telecommand (TC) Space Data Link Protocol in [54].
- One LDPC code specified by CCSDS for the Telemetry (TM) Space Data Link Protocol in [55].

However, the known methods presented in [35, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53] could not be applied without modifications to the standard CCSDS codes. Therefore, the purpose of this paper is to present a way how some of the mentioned techniques could be adapted to get RLL-ECCs from these codes.

The obtained RLL-ECCs could be of interest for practical applications in space exploration, and the approaches could be useful also for constructing RLL-ECCs for data storage systems.

The paper is organized as follows. In Section 3 theoretical background together with an overview of relevant CCSDS codes is given. In Section 4 the introduction to the method for obtaining RLL-ECCs and some comments on their advantages are presented. In Section 5 approaches for obtaining RLL-ECCs from four CCSDS codes are described in detail. In Section 6 the experimental results are given in a compact form. In Conclusions, some comments are made about the achieved results in this paper, and which results could be expected if further research is devoted to this area. In appendices the data necessary for constructing RLL-ECCs for particular CCSDS codes are given and also two examples of how the symbols could be ordered in RLL-ECCs obtained from two LDPC codes specified by CCSDS.

3 THEORETICAL BACKGROUND

In this section a rudimentary introduction to relevant ECC theory is given for the convenience of a broader readership. Only the families of codes related to the CCSDS codes for which the construction of RLL-ECCs was successful will be mentioned.

Linear block codes.

A linear block code C is defined as a k-dimensional subspace of an n-dimensional vector space over finite field $GF(q)$. It is denoted as an (n, k)-code. In this paper only binary block codes will be considered.

The subspace is spanned by k linearly independent vectors, which are usually given by rows of the so-called generator matrix $\mathbf{G}$. Each codeword $\mathbf{c}$ from C is

a linear combination of these rows. Therefore the encoding is simply a multiplication of the information vector $\mathbf{i}$ by $\mathbf{G}$:

$$\mathbf{c} = \mathbf{i}.\mathbf{G}. \tag{1}$$

The Hamming weight of a vector is the number of nonzero coordinates in the vector.

Another matrix by which the code is often given is the parity check matrix $\mathbf{H}$ for which

$$\mathbf{G}.\mathbf{H}^T = \mathbf{0}. \tag{2}$$

From its rows it is possible to deduce which symbols are participating in the corresponding check equations. For each codeword from the corresponding code C

$$\mathbf{c}.\mathbf{H}^T = \mathbf{0}. \tag{3}$$

$\mathbf{H}$ matrix is also often exploited in hard decoding of received vectors, for example if the so-called syndrome method is used [16]. In cases where the soft decoding is used the bipartite Tanner graph is useful, which in principle contains the same information as the $\mathbf{H}$ matrix [18].

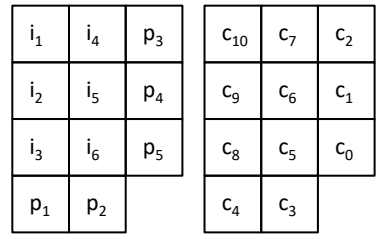


Figure 1. Example of binary $(11, 6)$ Product code obtained from single parity check codes a) symbols denoted from the point of view of parity check equations involving information and parity symbols, b) symbols denoted from the point of view of the resulting codeword symbols

From the practical point of view n is also interpreted as codeword length and k as the number of information symbols in codeword. In practice the coderate of the code is also very important:

$$R_c = \frac{k}{n}. \tag{4}$$

For example, the binary $(11, 6)$ Product code is illustrated in Figure 1. In it a single parity check is applied to each row and each column. It has the following

G matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

and the following **H** matrix.

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6}$$

Observing Figure 1 a) the following parity equations are valid:

$$p_1 = i_1 + i_2 + i_3, \tag{7}$$

$$p_2 = i_4 + i_5 + i_6, \tag{8}$$

$$p_3 = i_1 + i_4, \tag{9}$$

$$p_4 = i_2 + i_5, \tag{10}$$

$$p_5 = i_3 + i_6, \tag{11}$$

where p_i and i_i are parity and information symbols, respectively.

Observing Figure 1 b) the following check equations are valid:

$$c_{10} + c_9 + c_8 + c_4 = 0, \tag{12}$$

$$c_7 + c_6 + c_5 + c_3 = 0, \tag{13}$$

$$c_{10} + c_7 + c_2 = 0, \tag{14}$$

$$c_9 + c_6 + c_1 = 0, \tag{15}$$

$$c_8 + c_5 + c_0 = 0, \tag{16}$$

where c_i are codeword symbols. The codeword is denoted as a vector.

$$\mathbf{c} = (c_{10}, c_9, c_8, c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0). \tag{17}$$

The code in this example is a 6-dimensional subspace of the 11-dimensional vector space over $GF(2)$.

The LDPC codes discovered by Gallagher [56] are linear block codes defined by parity check matrices with a low density of nonzero elements, in binary case

ones. They belong to most practical ECC nowadays, not only because they have excellent error correcting capabilities but also because soft decoding methods are known for them with complexity which is not prohibitive for their application. For example, the belief propagation algorithm is often used for their decoding, which preferably operates with log likelihood ratios (LLR) [18]. Therefore it is not surprising that some of these codes are standardized by CCSDS.

LDPC codes recommended by CCSDS 131.0-B-4.

The LDPC CCSDS codes for the Telecommand (TC) Space Data Link Protocol specified in [54] are (n, k)-codes with the following basic parameters: $(128, 64)$, $(256, 128)$, $(512, 256)$. The codes are specified using the following parity check matrices:

$$\mathbf{H}_{64\times128} = \begin{bmatrix} \mathbf{I}_M \oplus \mathbf{\Phi}^{7} & \mathbf{\Phi}^{2} & \mathbf{\Phi}^{14} & \mathbf{\Phi}^{6} & \mathbf{0}_M & \mathbf{\Phi}^{0} & \mathbf{\Phi}^{13} & \mathbf{I}_M \\ \mathbf{\Phi}^{6} & \mathbf{I}_M \oplus \mathbf{\Phi}^{15} & \mathbf{\Phi}^{0} & \mathbf{\Phi}^{1} & \mathbf{I}_M & \mathbf{0}_M & \mathbf{\Phi}^{0} & \mathbf{\Phi}^{7} \\ \mathbf{\Phi}^{4} & \mathbf{\Phi}^{1} & \mathbf{I}_M \oplus \mathbf{\Phi}^{15} & \mathbf{\Phi}^{14} & \mathbf{\Phi}^{11} & \mathbf{I}_M & \mathbf{0}_M & \mathbf{\Phi}^{3} \\ \mathbf{\Phi}^{0} & \mathbf{\Phi}^{1} & \mathbf{\Phi}^{9} & \mathbf{I}_M \oplus \mathbf{\Phi}^{13} & \mathbf{\Phi}^{14} & \mathbf{\Phi}^{1} & \mathbf{I}_M & \mathbf{0}_M \end{bmatrix}, \quad (18)$$

$$\mathbf{H}_{128\times256} = \begin{bmatrix} \mathbf{I}_M \oplus \mathbf{\Phi}^{31} & \mathbf{\Phi}^{15} & \mathbf{\Phi}^{25} & \mathbf{\Phi}^{0} & \mathbf{0}_M & \mathbf{\Phi}^{20} & \mathbf{\Phi}^{12} & \mathbf{I}_M \\ \mathbf{\Phi}^{28} & \mathbf{I}_M \oplus \mathbf{\Phi}^{30} & \mathbf{\Phi}^{29} & \mathbf{\Phi}^{24} & \mathbf{I}_M & \mathbf{0}_M & \mathbf{\Phi}^{1} & \mathbf{\Phi}^{20} \\ \mathbf{\Phi}^{8} & \mathbf{\Phi}^{0} & \mathbf{I}_M \oplus \mathbf{\Phi}^{28} & \mathbf{\Phi}^{1} & \mathbf{\Phi}^{29} & \mathbf{I}_M & \mathbf{0}_M & \mathbf{\Phi}^{21} \\ \mathbf{\Phi}^{18} & \mathbf{\Phi}^{30} & \mathbf{\Phi}^{0} & \mathbf{I}_M \oplus \mathbf{\Phi}^{30} & \mathbf{\Phi}^{25} & \mathbf{\Phi}^{26} & \mathbf{I}_M & \mathbf{0}_M \end{bmatrix}, \quad (19)$$

$$\mathbf{H}_{256\times512} = \begin{bmatrix} \mathbf{I}_M \oplus \mathbf{\Phi}^{63} & \mathbf{\Phi}^{30} & \mathbf{\Phi}^{50} & \mathbf{\Phi}^{25} & \mathbf{0}_M & \mathbf{\Phi}^{43} & \mathbf{\Phi}^{62} & \mathbf{I}_M \\ \mathbf{\Phi}^{56} & \mathbf{I}_M \oplus \mathbf{\Phi}^{61} & \mathbf{\Phi}^{50} & \mathbf{\Phi}^{23} & \mathbf{I}_M & \mathbf{0}_M & \mathbf{\Phi}^{37} & \mathbf{\Phi}^{20} \\ \mathbf{\Phi}^{16} & \mathbf{\Phi}^{0} & \mathbf{I}_M \oplus \mathbf{\Phi}^{55} & \mathbf{\Phi}^{27} & \mathbf{\Phi}^{56} & \mathbf{I}_M & \mathbf{0}_M & \mathbf{\Phi}^{43} \\ \mathbf{\Phi}^{35} & \mathbf{\Phi}^{56} & \mathbf{\Phi}^{62} & \mathbf{I}_M \oplus \mathbf{\Phi}^{11} & \mathbf{\Phi}^{58} & \mathbf{\Phi}^{3} & \mathbf{I}_M & \mathbf{0}_M \end{bmatrix}, \quad (20)$$

where $\mathbf{I}_M$, $\mathbf{0}_M$, $\mathbf{\Phi}^{\varepsilon}$ are $M \times M$ identity, all zero and ε^{th} right circular shifts of the identity matrix respectively and $M = n/8$. The operator $\oplus$ denotes modulo two additions.

The LDPC codes could be encoded using the generator matrices in systematic form:

$$\mathbf{G} = \left[\mathbf{I}_{4M} \vdots \mathbf{W}\right]. \quad (21)$$

The submatrix $\mathbf{W}$ could be obtained as follows [54]:

$$\mathbf{W} = \left[\mathbf{P}^{-1} \vdots \mathbf{Q}\right], \quad (22)$$

where **P** and **Q** are submatrices composed from the last and first $4M$ columns of corresponding parity check matrices given by (18), (19), (20), respectively.

LDPC codes recommended by CCSDS 131.0-B-3.

The LDPC CCSDS code for the Telemetry (TM) Space Data Link Protocol specified in [55] could be obtained from the parity check matrix of the $(8\,176, 7\,154)$ base code. The resulting LDPC code consists of two macro-rows, of 16 circulants (511×511 submatrices) each:

$$\mathbf{H}_{1\,022\times 8\,176} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \dots & \mathbf{A}_{1,16} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \dots & \mathbf{A}_{2,16} \end{bmatrix}, \tag{23}$$

where the circulants are defined via 1's positions in the first row in Table 1.

Circulants	Positions of 1's in 1st Row	
	in Each Circulant	in **H** ("Absolute")
$\mathbf{A}_{1,1}$	0, 176	0, 176
$\mathbf{A}_{1,2}$	12, 239	523, 750
$\mathbf{A}_{1,3}$	0, 352	1 022, 1 374
$\mathbf{A}_{1,4}$	24, 431	1 557, 1 964
$\mathbf{A}_{1,5}$	0, 392	2 044, 2 436
$\mathbf{A}_{1,6}$	151, 409	2 706, 2 964
$\mathbf{A}_{1,7}$	0, 351	3 066, 3 417
$\mathbf{A}_{1,8}$	9, 359	3 586, 3 936
$\mathbf{A}_{1,9}$	0, 307	4 088, 4 395
$\mathbf{A}_{1,10}$	53, 329	4 652, 4 928
$\mathbf{A}_{1,11}$	0, 207	5 110, 5 317
$\mathbf{A}_{1,12}$	18, 281	5 639, 5 902
$\mathbf{A}_{1,13}$	0, 399	6 132, 6 531
$\mathbf{A}_{1,14}$	202, 457	6 845, 7 100
$\mathbf{A}_{1,15}$	0, 247	7 154, 7 401
$\mathbf{A}_{1,16}$	36, 261	7 701, 7 926

Table 1. The position of ones in circulants

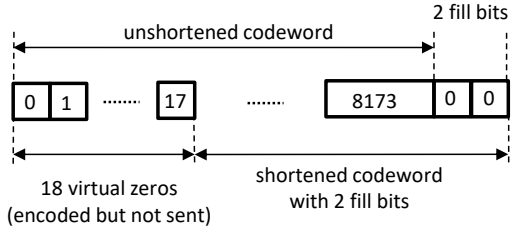


Figure 2. Relationship between the original codeword of the LDPC $(8\,176, 7\,154)$ base code and the shortened 8 160 bit long codeword

In actual applications the subcode $(8\,176, 7\,154)$ of the base code shall be used after shortening, and the extension as illustrated in Figure 2. The payload shall consist of 7 136 symbols which are prefixed by 18 zeros in order to get 7 154 bits. This message should be encoded via a generator matrix belonging to the $(8\,176, 7\,154)$ code. In the encoded codeword the first 18 symbols equal to zeros shall be deleted and two zeros shall be appended to the codeword. After this manipulation, the resulting message will have 8 160 bits.

4 INTRODUCTION TO THE METHOD FOR OBTAINING RLL-ECCS AND SOME COMMENTS ON ITS ADVANTAGES

The method for obtaining an RLL-ECC from a binary ECC is illustrated in Figure 3 for hard decoding and for transmission via BSC. The payload information is first encoded via the standard ECC into a sequence of codewords. Then the codeword symbols are reordered. In the following step some symbols are inverted by adding modulo two (mod 2) by a so-called modifier. The modifier is a binary vector with identical length to the codeword of the underlying ECC.

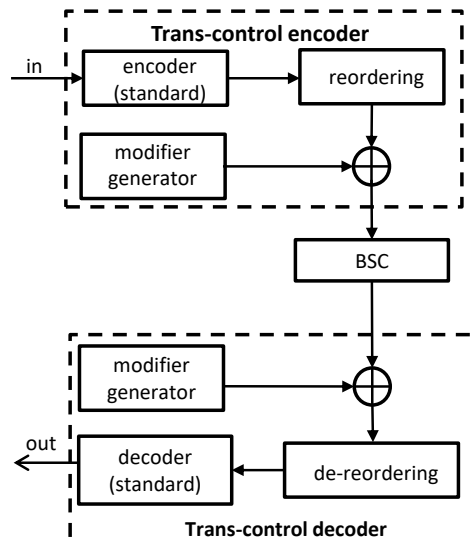


Figure 3. The principle of how a binary ECC can be modified in order to get RLL properties without additional redundancy and without the need to change its encoding and decoding algorithms in BSC. At the transmitter side the reordering (interleaving) and modifier addition are made. Their influence is eliminated by the addition of modifier and de-reordering (de-interleaving) on the receiving side.

On the receiving side, before the received symbols are input into the decoder,

the modifier adds mod 2 to them in order to eliminate its influence on decoding. Before standard decoding the order of the symbol also is restored. As a consequence, the modifications, namely two additions of modifiers and two reordering of symbols, are not visible from the point of view of the encoder and decoder.

Note 2. Further in this paper only the modifications on the side of transmitter will be described which could be interpreted as construction of RLL ECC codes from CCSDS LDPC codes. However, for the transmission it will be still supposed that the restoring modifications in the receiver will take place in real applications.

The main problem which has to be solved before the method could be applied in practice is to find a modifier and corresponding reordering of the codeword symbols, which will decrease the otherwise infinite run lengths in the codeword sequence of the original ECC.

There are different known approaches how to find a modifier and appropriate reordering connected with it [35, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53] based on the structural properties of the original ECCs.

For the reader's convenience only two from these will be mentioned here which are closely related to the methods used later in this paper. They are also most helpful for understanding the following explanation of how RLL-ECCs could be obtained from ECCs specified by CCSDS.

In [35] the following properties of matrix $\mathbf{G}$ were presented which allows obtaining RLL-ECCs from some ECCs using the modifiers obtained by approaches based on it.

Property 1. If the Hamming weight of all rows of generator matrix $\mathbf{G}$ of a linear binary block code C are even, then all codewords of C have even Hamming weights.

Proof. Let $w(\bar{g}_i)$ and $w(\bar{g}_j)$ be the Hamming weights of two rows $\bar{g}_i$ and $\bar{g}_j$ and let be m the number of coordinates in which both rows $\bar{g}_i$ and $\bar{g}_j$ have ones. Further, let us define $\bar{g}_k = \bar{g}_i + \bar{g}_j$. Then: $w(\bar{g}_k) = w(\bar{g}_i) - m + w(\bar{g}_j) - m = w(\bar{g}_i) + w(\bar{g}_j) - 2m$. Since we have assumed $w(\bar{g}_i)$ and $w(\bar{g}_j)$ as even any linear combination of rows of this matrix and, therefore, all codewords will have even weight. □

Property 2. If all codewords of a binary linear block code C with an even length n have an even Hamming weight, then the inversion of an odd number of symbols in the all codewords of C creates a new code C' in which no codeword has symbols which are all equal to zero or equal to one. (C' is a coset code of C).

Proof. Suppose that the ECC codeword vector $\bar{c}$ in C is transformed into $\bar{c}\,'$ in C' in which all bits are the same. Let w be the even weight of $\bar{c}$. To obtain $\bar{c}\,'$ from $\bar{c}$ either the w 1's or the $n - w$ 0's must be inverted. But since both w and n are even, this transformation would require an even number of inversions. □

This property could be sometimes applied also to submatrices of $\mathbf{G}$. In this case it is necessary to find disjoint subsets $G_1, G_2, \ldots, G_x$ of columns from $\mathbf{G}$ and form

the submatrices corresponding to these subsets $\mathbf{G}_1$, $\mathbf{G}_2$, ..., $\mathbf{G}_x$ each containing an even number of columns from $\mathbf{G}$ such that the corresponding submatrices have all rows with an even Hamming weight and:

$$G_1 \cup G_2 \cup \cdots \cup G_x \cup G_r = G, \tag{24}$$

where G is the set of all columns of $\mathbf{G}$ and G_r is the set of all columns not included in any of the submatrices $\mathbf{G}_1$, $\mathbf{G}_2$, ..., $\mathbf{G}_x$.

For each submatrix $\mathbf{G}_i \in \{\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_x\}$ the Property 2 is fulfilled and therefore by inverting an odd number of corresponding coordinates, or in other words, using a modifier, which has an odd number of ones in the coordinates covering $\mathbf{G}_i$ we can eliminate all zeros or all ones in the resulting codeword in positions corresponding to G_i.

Later, in [43] a related property was also observed in parity check matrix $\mathbf{H}$ of linear block codes, which could be useful for obtaining RLL-ECCs from some ECCs using modifiers obtained by approaches based on it.

Property 3. If the check equation of a linear binary block code C contains a set E with an even number of symbols, then inverting an odd number of symbols from E in all the codewords of C creates a new code C' in which no codeword has symbols which are all equal to zero or all equal to one in E. (C' is a coset code of C).

Proof. The proof is similar to the proof of Property 2. Suppose that the codeword $\bar{c}$ from C is transformed by inverting some symbols from E (which we will denote $\bar{c}_E$) to a codeword $\bar{c}'$ from C' in such a way that all symbols corresponding to E in $\bar{c}_E{}'$ are the same. Let w be the even weight and n_E be the even length of $\bar{c}_E$ and also of $\bar{c}_E{}'$, respectively. To obtain $\bar{c}_E{}'$ from $\bar{c}_E$ either the w 1's or the $(n_E - w)$ 0's must be inverted. But since both w and n_E are even, this transformation would require an even number of inversions. It is obvious that in cases where $w(\bar{c}_E) = 0$ the inversion of an uneven number of bits in $\bar{c}_E$ will clearly cause not all symbols in $\bar{c}_E{}'$ to be the same. □

If multiple disjoint sets in a generator or parity check matrix fulfill one of the mentioned properties, then it is possible to invert the odd number symbols corresponding to each such subset and minimize the lengths of intervals of other symbols filling gaps between them. Please see Figure 4.

In Figure 4 there are three subsets of codeword symbols concatenated in three continuous intervals. The first interval from the left with length L_1 symbols and the last interval with length L_3 symbols correspond to sets in which the inversion of an odd number of symbols leads to not all symbols in each of these intervals being the same. The interval in the middle corresponds to any other symbols which are not members of any subsets allowing inversion of an odd number of symbols. Let the length of this interval between these subsests be L_2 symbols long. After application of the inversions the resulting codeword in the corresponding interval with overall length $L = L_1 + L_2 + L_3$ will have RLL properties and the worst case runlength or

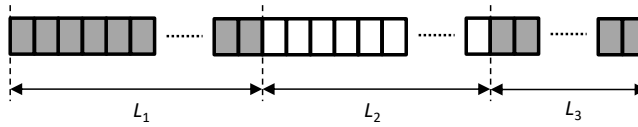


Figure 4. Illustration supporting explanation of how the upper bound (worst case) run length RL_{MAX} of equal symbols can be determined if two sets of symbols fulfilling Property 2 or Property 3 with one set of other symbols are concatenated in an alternating manner. Shadowed rectangles represent symbols fulfilling one of the properties, rectangles with no fill represent the other symbols.

the upper bound on it will be

$$RL_{MAX} = L_1 - 1 + L_2 + L_3 - 1. \tag{25}$$

The strategy of how to exploit the described properties from a high-level point of view, for example for matrix $\mathbf{H}$ is as follows. In the first step it is necessary to find as many disjoint subsets of symbols in different control equations as possible. As a rule, a brute force computerized search is necessary for this step, because the number of subsets could be very large. In the second step a redistribution of column positions in parity check matrix $\mathbf{H}$ is used with the goal to minimize the run lengths taking into account (25).

In the next example it is shown how the mentioned properties could be applied to the $(11, 6)$ Product code from Section 2.

The $\mathbf{G}$ matrix (5), could be reordered so that we get the following matrix:

$$\mathbf{G}' = \left[\begin{array}{cccc|ccccc|cc} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]. \tag{26}$$

(26) can be reordered further to (27). In (27) it is more clearly visible that the submatrix containing the first four columns from the left and the submatrix containing the next four columns respectively are both fulfilling Property 2. In order to minimize the run length after addition of the modifier, the following resulting form of the reordered $\mathbf{G}$ matrix could be used.

$$\mathbf{G}'' = \left[\begin{array}{cccc|c|cccc|cc} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right]. \tag{27}$$

Observing (27) and taking into account Property 2, it is obvious that it is possible to invert an uneven number of codeword symbols in positions corresponding to the subset of the first four columns from the left. As a result, not all symbols in the first four positions of the codeword will be the same. The same is also valid for the set of symbols starting in the 6^{th} position from the left and ending in the 9^{th} position if an uneven number of ones is in the modifier in these positions. For example the modifier could have the following form

$$\mathbf{m} = [\mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{0}, 0, \mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{0}, 0, 0]. \tag{28}$$

In modifier (28) the intervals of bold symbols correspond to the subsets which fulfill Property 2.

The sequence of codewords from the code used in this example will, after mod 2 addition of the modifier, contain runs no longer than 8 (please consult Figure 4, which illustrates the following explanation). This is because the gap between the locations of the two sets in which an uneven number of inversions could be applied has maximal length 2. Therefore, in two consecutive codewords there is an interval containing 10 symbols. In it the first four and the last four symbols are not identical. Therefore, in the worst case the first symbol is different from the next 3 symbols and the last symbol from the 10 is different from the previous 3 symbols. If the two triples and the gap contain identical symbols then it is obvious that the run length cannot be longer than 8.

Similar results could be obtained by exploiting Property 3. For example, in the first row from the top of the **H** matrix (6) there are four ones and also in the second row from the top there are four ones in different positions. Therefore, after reordering we can bring the **H** matrix to the following form

$$\mathbf{H}' = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{29}$$

Thus, one of the possible modifiers to limit the maximal run length to 8 could be the following

$$\mathbf{m} = [\mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{0}, 0, 0, \mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{1}, 0]. \tag{30}$$

In modifier (30) the intervals of bold symbols correspond to the subsets which fulfill Property 3.

The method illustrated in Figure 3 has the following advantages:

- The problem with ordering ECC and Translation codes mentioned in Section 1 of this manuscript is overcome;
- No additional redundancy is introduced;
- Practical implementation has low complexity;

- The encoding and decoding of the original ECC does not have to be modified in cases where some round conditions are valid;
- The RL_{MAX} is guaranteed, which is not the case if scramblers are used.

The first three advantages are obvious. The last two deserve some more detailed explanation. Especially in connection with the fourth one an important question arises if and under which round conditions the 4$^{\text{th}}$ advantage will remain valid if soft decoding is also used. Some answers were given recently in [57]. A short extract from [57] will be presented in the next text.

The argument why the 4$^{\text{th}}$ advantage is supported by the method when hard decoding is used is straightforward. It is obvious that in such cases the two mod 2 additions from the point of view of the encoder and decoder are transparent (not visible) if they are made in a synchronous manner.

"Synchronous" – for linear block codes means that block synchronization is present between the transmitter and receiver so that the mod 2 additions could be made for the same symbols in the transmitted and received blocks corresponding to codewords. The transparency of the additions is illustrated in Figure 5.

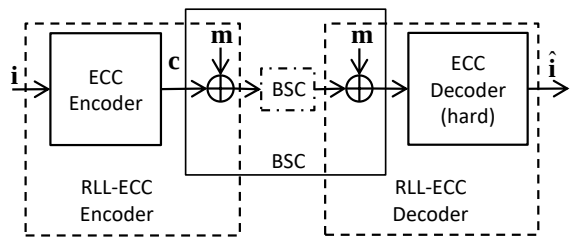


Figure 5. Illustration which shows why the encoding and hard decision decoding does not have to be modified if the modifiers are used. It is evident that the two modulo 2 additions of modifiers and BSC could be included in one box, which is essentially the same BSC because the two modulo 2 additions will cancel if applied in synchronization.

If soft decoding is used the 4$^{\text{th}}$ advantage is also valid if coherent BPSK modulation is used and the channel can be modeled as an AWGN channel. However, in this case the mod 2 addition on the receiving side has to be substituted by negation of the $L(z/s)$ values corresponding to symbols inverted on the transmitting side by ones of the modifier. A detailed explanation can be found in [57]. Here we only illustrate how the method is used in Figure 6 if soft decoding is applied for the ECC.

It has to be mentioned that the round conditions, namely exploitation of coherent BPSK modulation and modelling of the transmission channel as an AWGN channel, are favorable for many standardized CCSDS systems. In most standardized CCSDS systems coherent BPSK modulation is used and the AWGN channel model is appropriate for deep space communications [58].

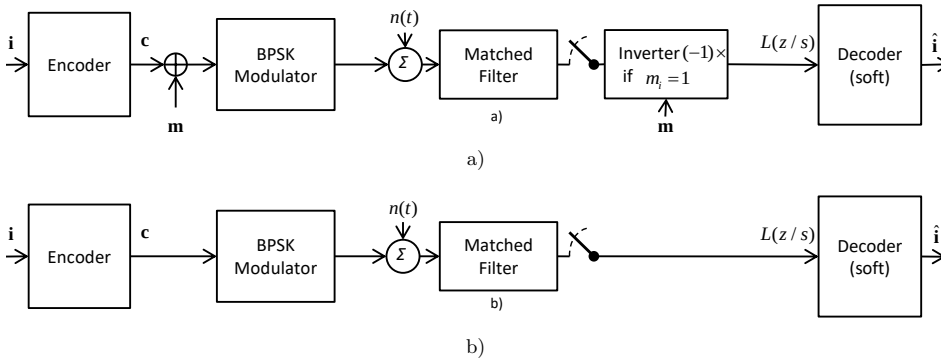


Figure 6. a) Illustration of how our method could be applied if soft decoding and coherent BPSK modulation are used in the AWGN channel. b) Shows that the addition of the modifier and multiplication of inverter cancel its influence and the encoder and decoder do not see these operations. Consequently, the encoding and decoding of the original ECC does not have to be modified. A detailed explanation of why the inversion of LLR is applied can be found in [57]. The $L(z/s)$ denotes the specific LLR obtained from the channel.

Concerning the 5^{th} advantage it is worth mentioning that in [54] the following note is present.

Note 3. LDPC coding, by itself, cannot guarantee sufficient bit transitions to keep receiver symbol synchronizers in lock. In order to have more bit transitions standard Pseudo-Randomizers (Scramblers) are used.

In general, the disadvantage of scramblers is that, as a rule, they cannot guarantee any RLL properties. They only increase the number of changes in a signal with high probability. By contrast the methods described in this paper guarantee the RLL properties of the constructed codes.

5 APPROACHES FOR OBTAINING RLL-ECCS FROM FOUR CCSDS CODES

In this section it will be explained how the method illustrated in Figure 3 can be applied to some CCSDS codes in order to construct RLL-ECCs. For simplicity the explanation will be done with the assumption that hard decoding is used and the transmission channel can be modelled as a BSC. However, the obtained RLL-ECCs could also be used if soft decoding is implemented in the decoder and the transmission is over the AWGN channel with coherent BPSK modulation. The only necessary change in this case is to use an inversion of $L(z/s)$ as illustrated in Figure 6 a) instead of a mod 2 addition. Again it has to be emphasized that only the modifications on the side of transmitter will be described which could be

interpreted as construction of RLL ECC codes from CCSDS LDPC codes. However, for the transmission it will be still supposed that the restoring modifications in the receiver will take place in real applications.

Approach for obtaining RLL-ECC from LDPC codes recommended by CCSDS 131.0-B-2.

In this subsection an approach is described how the RLL-ECCs could be obtained from standard LDPC codes specified in [54] by CCSDS and defined by the parity check matrices given in (18), (19), (20). The possibility to use the approach is based on the hypothesis that Property 3 can be exploited. In other words it is necessary to find as many disjoint rows with even Hamming weight in each of the parity check matrices as possible in order to minimize the RLL of the corresponding codes.

Because the matrix $\mathbf{H}_{64\times128}$ (18) is the simplest and has similar properties to the other two defined by (19) and (20), it will be used in the following explanation. Observing (18) one can see that there are 4 submatrices 16×128 each composed of 8 submatrices 16×16 which we will denote as macro-rows. The scatter chart of the first macro-row from the top in (18) is illustrated in Figure 7.

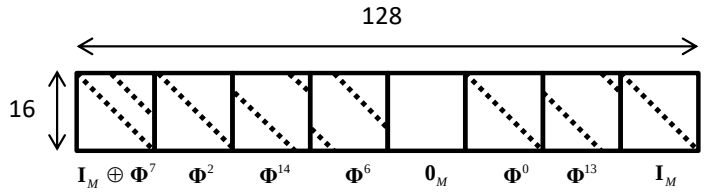


Figure 7. Scatter chart of the first macro-row from the top in the $\mathbf{H}_{64\times128}$ matrix

In this macro-row all the submatrices except the 1st and 5th from left have only one 1 in each row and each column because they are identity matrices or right shifts of the identity matrix. Therefore in this subset of six submatrices each row has a Hamming weight 6. The fifth submatrix has all elements 0. On the other hand the 1st submatrix has two ones in each row. Consequently each row of it can contribute to the overall Hamming weight 8 of the analyzed macro-row with Hamming weight 2. However, we cannot select all its rows in order to exploit Property 3 because the corresponding sets in these rows will not be all disjoint. (In other words there would be two ones in each column of the 1st submatrix.) To overcome this difficulty it is possible to choose only the rows with indexes 0, 2, 4, 6, 8, 10, 12, 14. The result is illustrated in Figure 8. The corresponding set of rows will be denoted as $P_{64\times128}$.

Note 4. The mathematical reason why there are no two ones in any column corresponding to positions of ones in rows with indexes from $P_{64\times128}$ is the following. The sequence of column indexes 0, 2, 4, ..., 14, corresponding to

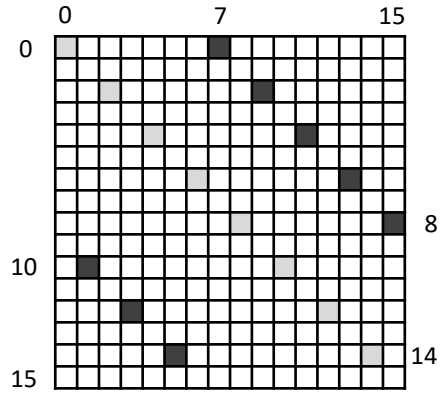


Figure 8. First submatrix from left in first macro-row from top in $\mathbf{H}_{64\times128}$ matrix with highlighted ones in the 8 selected rows with indexes 0, 2, 4, ..., 14. It is obvious that the Hamming weight of each of these selected rows is 2 and that each column corresponding to the selected rows has Hamming weight 1.

every second one from a circulant starting in the first row in position 0 (shadow squares in Figure 8) contains only even numbers. On the other hand the sequence 7, 9, ..., 13, 19 mod(16), 19 mod(16), 21 mod(16) corresponding to every second one in the circulant starting in the first row in position 7 (black squares in Figure 8) contains only odd numbers. Therefore, there is no overlap in these two sets of column indexes.

The corresponding set of rows will be denoted as $P_{64\times128}$. This selection of rows will contribute with Hamming weight 2 to the overall Hamming weight 8 in each selected row in the 1st macro-row of $\mathbf{H}_{64\times128}$.

The number of the selected rows in $P_{64\times128}$ (in which the Property 3 can be exploited) is 8. The found column indexes in which there are ones in the rows with indexes from $P_{64\times128}$ in 1st macro-row are listed in Appendix I (8.1), where the indexes from the remaining symbols are also presented. (The number of the remaining columns in the 1st macro-row of $\mathbf{H}_{64\times128}$ is $128 - (8 \times 8) = 64$).

It is possible to reorder the columns in the transmitter in such a way that there will be 8 symbols participating in each check equation corresponding to the rows, in continuous intervals with length 8 interleaved with continuous intervals with the other symbols with length 8 as illustrated in Figure 9.

Taking into account (25) and observing Figure 6 it is obvious that after using an appropriate modifier, (such as will invert an uneven number of columns in each interval of codeword symbols corresponding to the bold intervals in Figure 9), the maximal RLL of the resulting coset code will be equal to 22.

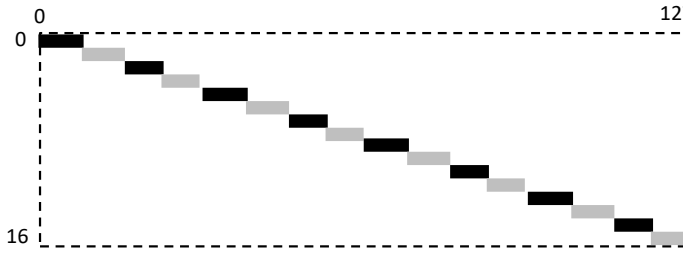


Figure 9. The scatter chart of the first macro-row from $\mathbf{H}_{64\times128}$ after the reordering of columns in $\mathbf{H}_{64\times128}$: bold lines are interval of 8 ones in which Property 3 could be exploited and shadow lines are intervals corresponding to the remaining columns

Note 5. After transmission of each codeword from the RLL ECC (coset code of the original ECC) obtained in this way the influence of the modifier addition and column reordering will be eliminated in the receiver so that the standard decoding of LDPC codes could be used in case the underlying channel could be modeled as a BSC or an AWGN with coherent BPSK modulation. Please see Section 4 or [57] for more details.

Similar analysis was performed for the $\mathbf{H}_{128\times256}$ parity check matrix defined by (20). The first submatrix of the first macro-row of it is illustrated in Figure 10. With the highlighted ones in the set $P^{(3)}_{128\times256} = \{0, 2, \ldots, 30\}$.

The mathematical reasoning in Note 4 is valid again for $P_{128\times256}$ with slight modification, namely that instead of modulo 16 counting modulo 32 counting is used for the sequences of every second column index for circulant starting in the first row in 0 (even) and 31 (odd) positions respectively.

Property 3 could be exploited in intervals of ones in rows corresponding to all indexes in $P_{128\times256}$ in the first macro-row of $\mathbf{H}_{128\times256}$. There will be 16 such rows with Hamming weight 8 each, therefore the number of remaining positions in which the Property 3 cannot be used will be $256 - (16 \times 8) = 128$. Consequently, a similar approach as before used for $\mathbf{H}_{64\times128}$ will also lead to a coset code with $RLL_{MAX} = 22$ in this case. It is because the 16 intervals with positions in which Property 3 can be exploited each containing 8 symbols will be interspersed with 16 intervals each containing 8 symbols in which it cannot be exploited after the appropriate reordering of columns similar to that illustrated in Figure 9. These sets are listed in Appendix II (8.2) (in this appendix the numbers correspond to columns in the $\mathbf{H}_{64\times128}$ matrix in order from left to right starting with 0 as a column index of the first column on the left).

Similar analysis was performed for the $\mathbf{H}_{256\times512}$ parity check matrix defined by (21). The first submatrix of the first macro-row of it is illustrated in Figure 10. With the highlighted ones in the set $P_{256\times512} = \{0, 2, \ldots, 62\}$.

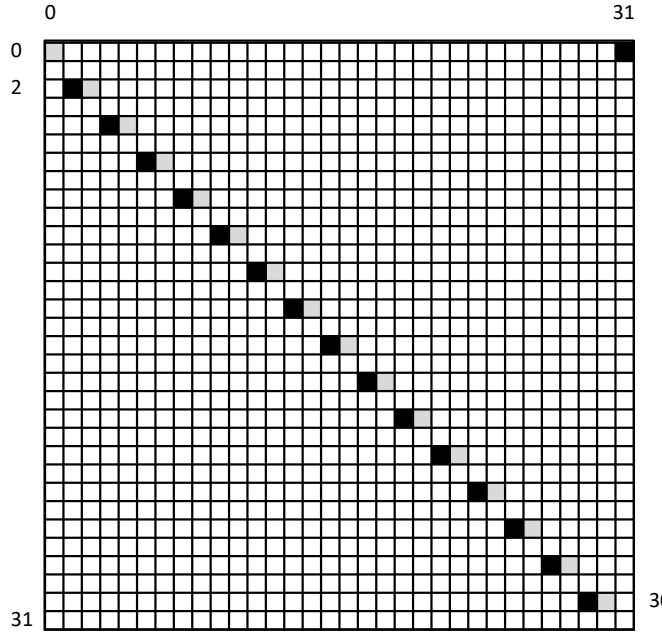


Figure 10. First submatrix from left in 1st macro-row scatter chart of $\mathbf{H}_{128\times256}$ matrix

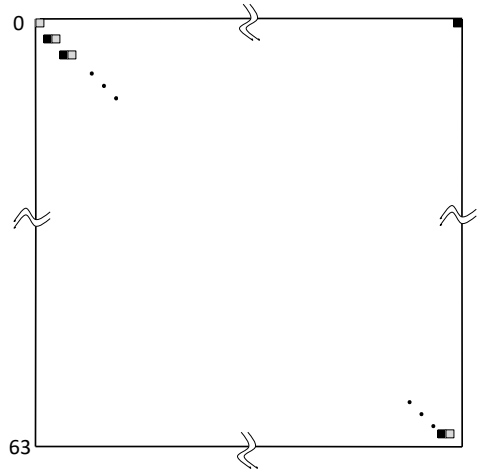


Figure 11. First submatrix from left in 1st macro-row scatter chart of $\mathbf{H}_{256\times512}$ matrix

The mathematical reasoning in Note 2 is valid again for $P_{256\times512}$ with slight modification, namely that instead of modulo 16 counting modulo 64 counting is used for the sequences of every second index for circulants starting in the first row in 0 (even) and 63 (odd) positions respectively.

After reordering, the first macro-row in matrix $\mathbf{H}_{256\times512}$ will contain 32 intervals with 8 symbols each in which Property 3 can be used interspersed with 32 intervals with 8 symbols each in which Property 3 cannot be exploited. These sets are listed in Appendix III (8.3). (In the appendix the numbers correspond to columns in the $\mathbf{H}_{256\times512}$ matrix in order from left to right starting with 0 as a column index of the first column on the left.)

Therefore, the number of remaining positions in which Property 3 cannot be used will be $512 - (32 \times 8) = 256$. Consequently a similar approach as used before for $\mathbf{H}_{256\times512}$ will also lead to a coset code with $RLL_{MAX} = 22$ in this case.

Note 6. The obtained results allow us to choose one of many possible reordering's of symbols in a codeword which will minimize the run lengths of equal symbols for each of the analyzed parity check matrices.

In Appendix IV (8.4) there is one example for the $\mathbf{H}_{128\times256}$ matrix in which the ordering of codeword symbols is given. In this sequence the numbers correspond to columns in the $\mathbf{H}_{128\times256}$ matrix in order from left to right starting with 0 as a column index of the first column on the left. Each continuous interval with bold numbers represents one set of symbols in which the inversion of an odd number of symbols can take place. In other words this corresponds to indexes of columns in which there are ones in the particular control equation in the parity check matrix (19). Observing the sequence and taking into account (25) it is obvious that after applying the odd number of inversions in each set represented by bold numbers in the sequence the maximal run-length of equal symbols will be $RL_{MAX} = 22$.

Approach for obtaining RLL-ECC from LDPC codes recommended by CCSDS 131.0-B-3.

In this section it will be presented how the RLL-ECC coset code can be obtained from the $(8\,166, 7\,156)$ LDPC code recommended by CCSDS in [55]. The construction of the original LDPC code starts with the $\mathbf{H}$ matrix (23). The analyses of the macro-rows in it revealed that it is not possible to use the same approach as it was used in the previous subsection for selection of rows in which the requirements, allowing exploiting Property 3 for inverting an uneven number of symbols in the corresponding subsets of symbols, are fulfilled. More specifically, it is not possible to select all rows with even indexes. It is because the submatrices in the macro-rows are 511×511 and its circulants start in the first row in pairs of indexes where each component can be even or uneven. In other words the mathematical reasons in Note 4 cannot

be applied. Please see Table 1. Therefore another approach has to be used for selection of the rows, which will exhibit the properties required in Property 3.

After visually analyzing the first macro-row of 511×511 submatrices in the $\mathbf{H}$ matrix (23)

$$\mathbf{R}_{511\times 8\,176} = \left[\begin{array}{cccc} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \ldots & \mathbf{A}_{1,16} \end{array} \right], \tag{31}$$

its scatter chart is sketched in Figure 12, the following observations could be made.

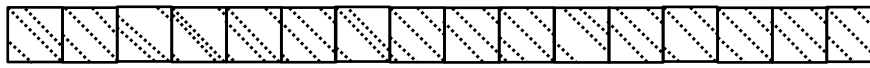


Figure 12. Scatter chart of the first macro-row of $\mathbf{R}_{511\times 8\,176}$ from (23)

Because each circulant in $\mathbf{R}_{511\times 8\,176}$ contains two ones in each row it is evident that each row in it contains 32 ones. Observing only one 511×511 circulant, $\mathbf{A}_{i,j}$ it is possible to concentrate on finding the maximal possible number of rows containing two ones in disjoint columns (each corresponding column will contain only one 1). In other words it is desirable to find a submatrix formed by a maximal number of rows selected from $\mathbf{A}_{i,j}$ in such way that the column Hamming weight will be only 1 (this is a consequence of the requirement of Property 3 that each such set of corresponding symbols have to be disjoint from others and each has even Hamming weight).

Further it can be argued that in submatrices in which the main diagonal contains ones, for example in $\mathbf{A}_{1,1}$, one simple approach is to start the selection of rows from the first row from the top and stop just before the requirement that the column Hamming weight must be equal to one for the resulting submatrix is broken. The number of rows which is possible to obtain in this way is dependent on the difference between the integers in Table 1 modulo 511.

Note 7. One can imagine that the scatter chart of each circulant is drawn on graph paper, which is then folded to form a cylinder. The smaller distance between ones in the first line will determine how many rows with the desired properties could be selected without violating the rule that in each column must be only one 1.

However, starting the selection from the first row would also give sets, which will contain the first 18 bits (column indexes) which are not transmitted. Please see Figure 2. This would result in no guarantee that the not transmitted symbol is the only one which is different from all others in its set. Consequently, it will not be possible to use 18 sets with these 18 symbols. Therefore it is necessary

to search for rows containing sets which fulfill Property 3, starting from rows with indexes higher than 17 in CCDSD notation.

In the other circulants the number of rows which can be obtained in this way is also dependent on the difference between the integers in Table 1 modulo 511. Please see Note 7. Circulant $\mathbf{A}_{1,4}$ has the smallest difference (distance) between ones in the first row. It is equal to 104. Therefore it contains a submatrix composed from rows of (31) with indexes from a set $S_L = \{18, 19, \ldots, 121\}$ has all rows with Hamming weight equal to 2 and all columns with Hamming weight equal to one. It is illustrated in Figure 13. The set S_L corresponds to the shaded area in this figure.

In all submatrices in the other circulants from the macro-row $\mathbf{R}_{511\times 8\,176}$ containing rows with indexes from the set the same conditions hold for each row and each column Hamming weight. As a consequence, the overall Hamming weight in a submatrix with dimension $104 \times 8\,176$ formed by rows with indexes S_L from $\mathbf{R}_{511\times 8\,176}$ is 32. In other words, the conclusion can be made that 104 rows fulfil Property 3 each having hamming weight equal to 32. They are listed in Appendix V (8.5).

Taking into account how the 8 160-bit long codewords are obtained in the standard [55] (illustrated in Figure 2) we can get one more set which fulfills Property 3. Namely the two zeros bits which are filled in the last two positions of the shortened codeword. They could very simply contribute to improving RLL properties. The modifier can invert one of these symbols and the two transmitted symbols will not be the same. Based on this selection of 105 sets which fulfill Property 3, it is possible to use any reordering which will bring the symbols in these sets together (in 32 consecutive positions) with 46-bit long gaps between them. In the gaps the other symbols not contained in the sets can be positioned. This follows from the following simple calculations. The 105 sets of symbols which fulfill Property 3 will consume $104 \times 32 + 2 = 3\,330$ symbols. There are $8\,160 - 3\,330 = 4\,830$ other symbols. These can be divided into 105 sets each containing 46 symbols. This allows concatenating the sets which fulfill Property 3 with the others in an alternating pattern. Consequently, in the worst case the maximal length of run of equal symbols $RL_{MAX} = 108$ (please see (23)). The resulting order of the indexes from the base code in the transmitted codeword after reordering could be, for example, as given in the Appendix VI (8.6). (Index there are in base code CCSDS.)

6 EXPERIMENTAL RESULTS

In this section we will, in compact form, present the experimental results obtained by our approaches together with an overview of the appendices which could be helpful for someone interested in implementing the system with the obtained RLL-CCSDS codes in practice.

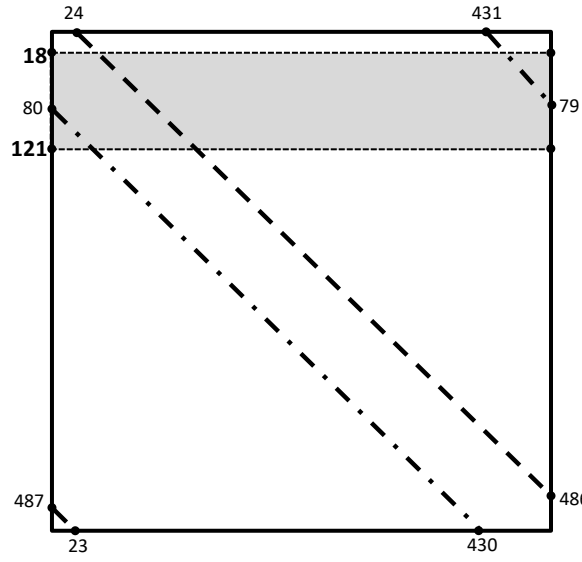


Figure 13. Scatter chart of the 4$^{\text{th}}$ macro-row of $\mathbf{R}_{511\times 8\,176}$ from (23)

For the LDPC codes for the TC Space Data Link Protocol specified by CCSDS in [54] with parameters: $(128, 64)$, $(256, 128)$, $(512, 128)$ the results are presented in Table 1 in the first 3 rows. For the $(8\,176, 7\,154)$ LDPC code [55] the result is presented in fourth row in Table 2.

CCSDS LDPC Code (n, k)	Ref.	RL_{MAX}	Data Needed for RLL-ECC and Modifier Construction
$(128, 64)$	[54]	22	Appendix I (8.1)
$(256, 128)$	[54]	22	Appendix II (8.2), IV (8.4)
$(512, 256)$	[54]	22	Appendix III (8.3)
$(8\,176, 7\,154)$	[55]	108	Appendix V (8.5), VI (8.6)

Table 2. Obtained results for the selected LDPC codes specified by CCSDS

In Table 2 the information is also present as to which Appendix provides the data needed for examples of possible constructions of the particular RLL-ECC codes and corresponding modifiers.

7 CONCLUSIONS

In this paper it was shown that RLL-ECC codes could be obtained from four selected LDPC codes specified by CCSDS using the method with modifiers depicted in Figure 3. The main advantages of these codes are that the run lengths of equal

symbols are restricted to corresponding values with the guarantee that no additional redundancy has to be introduced in encoding, and that the encoding and decoding of the original error control codes specified by CCSDS do not have not to be modified.

Further future research can reveal that it is possible that other RLL-ECC codes could be constructed from the standard CCSDS codes by the approach used in this paper.

For example in [59] the author proposed one related approach how to obtain RLL-ECCs from Reed Solomon (RS) using modifiers. However, its application to RS codes specified by CCSDS did not bring a significant decrease of RL_{MAX}. It remains for future research to find some other approach for CCSDS RS codes.

It is also possible that future research can bring new results with lower values of RL_{MAX} even for some of the 4 codes presented in this paper. The reason is that, for example, for $\mathbf{H}_{1\,022\times 8\,176}$ it was not possible to make a complete search which would reveal the optimal composition of row indexes subsets fulfilling Property 3.

This paper gave a positive answer to the question of whether the approach using modifiers can be used in order to obtain RLL-ECCs from some standard CCSDS codes.

Acknowledgment

This work was supported by the Slovak Research and Development Agency under the Contract No. APVV-19-0436 and it was also supported by the Scientific Grant Agency of the Ministry of Education of the Slovak Republic, and the Slovak Academy of Sciences (grant VEGA 1/0477/18).

8 APPENDICES

8.1 Appendix I

Indexes of subsets in which the modifier can invert an uneven number of symbols in a codeword of the CCSDS LDPC code defined by (18):

$$\mathbf{S}_1 = \{\mathbf{0}, \mathbf{7}, \mathbf{18}, \mathbf{46}, \mathbf{54}, \mathbf{80}, \mathbf{109}, \mathbf{112}\},$$

$$\mathbf{S}_2 = \{\mathbf{2}, \mathbf{9}, \mathbf{20}, \mathbf{32}, \mathbf{56}, \mathbf{82}, \mathbf{111}, \mathbf{114}\},$$

$$\mathbf{S}_3 = \{\mathbf{4}, \mathbf{11}, \mathbf{22}, \mathbf{34}, \mathbf{58}, \mathbf{84}, \mathbf{97}, \mathbf{116}\},$$

$$\mathbf{S}_4 = \{\mathbf{6}, \mathbf{13}, \mathbf{24}, \mathbf{36}, \mathbf{60}, \mathbf{86}, \mathbf{99}, \mathbf{118}\},$$

$$\mathbf{S}_5 = \{\mathbf{8}, \mathbf{15}, \mathbf{26}, \mathbf{38}, \mathbf{62}, \mathbf{88}, \mathbf{101}, \mathbf{120}\},$$

$$\mathbf{S}_6 = \{\mathbf{1}, \mathbf{10}, \mathbf{28}, \mathbf{40}, \mathbf{48}, \mathbf{90}, \mathbf{103}, \mathbf{122}\},$$

$$\mathbf{S}_7 = \{\mathbf{3}, \mathbf{12}, \mathbf{30}, \mathbf{42}, \mathbf{50}, \mathbf{92}, \mathbf{105}, \mathbf{124}\},$$

$$\mathbf{S}_8 = \{\mathbf{5}, \mathbf{14}, \mathbf{16}, \mathbf{44}, \mathbf{52}, \mathbf{94}, \mathbf{107}, \mathbf{126}\}.$$

The set of the remaining column indexes $\mathbf{H}_{64\times128}$ is the following:

$$S_R = \{17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 81, 83, 85, 87, 89, 91, 93, 95, 96, 98, 100, 102, 104, 106, 108, 110, 113, 115, 117, 119, 121, 123, 125, 127\}.$$

8.2 Appendix II

Indexes of subsets in which the modifier can invert an uneven number of symbols in a codeword of the CCSDS LDPC code defined by (19):

$$\mathbf{S}_1 = \{\mathbf{0}, \mathbf{31}, \mathbf{47}, \mathbf{89}, \mathbf{96}, \mathbf{180}, \mathbf{205}, \mathbf{224}\},$$

$$\mathbf{S}_2 = \{\mathbf{1}, \mathbf{2}, \mathbf{49}, \mathbf{91}, \mathbf{98}, \mathbf{182}, \mathbf{207}, \mathbf{226}\},$$

$$\mathbf{S}_3 = \{\mathbf{3}, \mathbf{4}, \mathbf{51}, \mathbf{93}, \mathbf{100}, \mathbf{184}, \mathbf{209}, \mathbf{228}\},$$

$$\mathbf{S}_4 = \{\mathbf{5}, \mathbf{6}, \mathbf{53}, \mathbf{95}, \mathbf{102}, \mathbf{186}, \mathbf{211}, \mathbf{230}\},$$

$$\mathbf{S}_5 = \{\mathbf{7}, \mathbf{8}, \mathbf{55}, \mathbf{65}, \mathbf{104}, \mathbf{188}, \mathbf{213}, \mathbf{232}\},$$

$$\mathbf{S}_6 = \{\mathbf{9}, \mathbf{10}, \mathbf{57}, \mathbf{67}, \mathbf{106}, \mathbf{190}, \mathbf{215}, \mathbf{234}\},$$

$$\mathbf{S}_7 = \{\mathbf{11}, \mathbf{12}, \mathbf{59}, \mathbf{69}, \mathbf{108}, \mathbf{160}, \mathbf{217}, \mathbf{236}\},$$

$$\mathbf{S}_8 = \{\mathbf{13}, \mathbf{14}, \mathbf{61}, \mathbf{71}, \mathbf{110}, \mathbf{162}, \mathbf{219}, \mathbf{238}\},$$

$$\mathbf{S}_9 = \{\mathbf{15}, \mathbf{16}, \mathbf{63}, \mathbf{73}, \mathbf{112}, \mathbf{164}, \mathbf{221}, \mathbf{240}\},$$

$$\mathbf{S}_{10} = \{\mathbf{17}, \mathbf{18}, \mathbf{33}, \mathbf{75}, \mathbf{114}, \mathbf{166}, \mathbf{223}, \mathbf{242}\},$$

$$\mathbf{S}_{11} = \{\mathbf{19}, \mathbf{20}, \mathbf{35}, \mathbf{77}, \mathbf{116}, \mathbf{168}, \mathbf{193}, \mathbf{244}\},$$

$$\mathbf{S}_{12} = \{\mathbf{21}, \mathbf{22}, \mathbf{37}, \mathbf{79}, \mathbf{118}, \mathbf{170}, \mathbf{195}, \mathbf{246}\},$$

$$\mathbf{S}_{13} = \{\mathbf{23}, \mathbf{24}, \mathbf{39}, \mathbf{81}, \mathbf{120}, \mathbf{172}, \mathbf{197}, \mathbf{248}\},$$

$$\mathbf{S}_{14} = \{\mathbf{25}, \mathbf{26}, \mathbf{41}, \mathbf{83}, \mathbf{122}, \mathbf{174}, \mathbf{199}, \mathbf{250}\},$$

$$\mathbf{S}_{15} = \{\mathbf{27}, \mathbf{28}, \mathbf{43}, \mathbf{85}, \mathbf{124}, \mathbf{176}, \mathbf{201}, \mathbf{252}\},$$

$$\mathbf{S}_{16} = \{\mathbf{29}, \mathbf{30}, \mathbf{45}, \mathbf{87}, \mathbf{126}, \mathbf{178}, \mathbf{203}, \mathbf{254}\}.$$

The set of the remaining column indexes in $\mathbf{H}_{128\times256}$ is the following:

$$S_R = \{32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,$$

156, 157, 158, 159, 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 181, 183, 185, 187, 189, 191, 192, 194, 196, 198, 200, 202, 204, 206, 208, 210, 212, 214, 216, 218, 220, 222, 225, 227, 229, 231, 233, 235, 237, 239, 241, 243, 245, 247, 249, 251, 253, 255}.

8.3 Appendix III

Indexes of subsets in which the modifier can invert an uneven number of symbols in a codeword of the CCSDS LDPC code defined by (20):

$$\mathbf{S}_1 = \{\mathbf{0}, \mathbf{63}, \mathbf{94}, \mathbf{178}, \mathbf{217}, \mathbf{363}, \mathbf{446}, \mathbf{448}\},$$

$$\mathbf{S}_2 = \{\mathbf{1}, \mathbf{2}, \mathbf{96}, \mathbf{180}, \mathbf{219}, \mathbf{365}, \mathbf{384}, \mathbf{450}\},$$

$$\mathbf{S}_3 = \{\mathbf{3}, \mathbf{4}, \mathbf{98}, \mathbf{182}, \mathbf{221}, \mathbf{367}, \mathbf{386}, \mathbf{452}\},$$

$$\mathbf{S}_4 = \{\mathbf{5}, \mathbf{6}, \mathbf{100}, \mathbf{184}, \mathbf{223}, \mathbf{369}, \mathbf{388}, \mathbf{454}\},$$

$$\mathbf{S}_5 = \{\mathbf{7}, \mathbf{8}, \mathbf{102}, \mathbf{186}, \mathbf{225}, \mathbf{371}, \mathbf{390}, \mathbf{456}\},$$

$$\mathbf{S}_6 = \{\mathbf{9}, \mathbf{10}, \mathbf{104}, \mathbf{188}, \mathbf{227}, \mathbf{373}, \mathbf{392}, \mathbf{458}\},$$

$$\mathbf{S}_7 = \{\mathbf{11}, \mathbf{12}, \mathbf{106}, \mathbf{190}, \mathbf{229}, \mathbf{375}, \mathbf{394}, \mathbf{460}\},$$

$$\mathbf{S}_8 = \{\mathbf{13}, \mathbf{14}, \mathbf{108}, \mathbf{128}, \mathbf{231}, \mathbf{377}, \mathbf{396}, \mathbf{462}\},$$

$$\mathbf{S}_9 = \{\mathbf{15}, \mathbf{16}, \mathbf{110}, \mathbf{130}, \mathbf{233}, \mathbf{379}, \mathbf{398}, \mathbf{464}\},$$

$$\mathbf{S}_{10} = \{\mathbf{17}, \mathbf{18}, \mathbf{112}, \mathbf{132}, \mathbf{235}, \mathbf{381}, \mathbf{400}, \mathbf{466}\},$$

$$\mathbf{S}_{11} = \{\mathbf{19}, \mathbf{20}, \mathbf{114}, \mathbf{134}, \mathbf{237}, \mathbf{383}, \mathbf{402}, \mathbf{468}\},$$

$$\mathbf{S}_{12} = \{\mathbf{21}, \mathbf{22}, \mathbf{116}, \mathbf{136}, \mathbf{239}, \mathbf{321}, \mathbf{404}, \mathbf{470}\},$$

$$\mathbf{S}_{13} = \{\mathbf{23}, \mathbf{24}, \mathbf{118}, \mathbf{138}, \mathbf{241}, \mathbf{323}, \mathbf{406}, \mathbf{472}\},$$

$$\mathbf{S}_{14} = \{\mathbf{25}, \mathbf{26}, \mathbf{120}, \mathbf{140}, \mathbf{243}, \mathbf{325}, \mathbf{408}, \mathbf{474}\},$$

$$\mathbf{S}_{15} = \{\mathbf{27}, \mathbf{28}, \mathbf{122}, \mathbf{142}, \mathbf{245}, \mathbf{327}, \mathbf{410}, \mathbf{476}\},$$

$$\mathbf{S}_{16} = \{\mathbf{29}, \mathbf{30}, \mathbf{124}, \mathbf{144}, \mathbf{247}, \mathbf{329}, \mathbf{412}, \mathbf{478}\},$$

$$\mathbf{S}_{17} = \{\mathbf{31}, \mathbf{32}, \mathbf{126}, \mathbf{146}, \mathbf{249}, \mathbf{331}, \mathbf{414}, \mathbf{480}\},$$

$$\mathbf{S}_{18} = \{\mathbf{33}, \mathbf{34}, \mathbf{64}, \mathbf{148}, \mathbf{251}, \mathbf{333}, \mathbf{416}, \mathbf{482}\},$$

$$\mathbf{S}_{19} = \{\mathbf{35}, \mathbf{36}, \mathbf{66}, \mathbf{150}, \mathbf{253}, \mathbf{335}, \mathbf{418}, \mathbf{484}\},$$

$$\mathbf{S}_{20} = \{\mathbf{37}, \mathbf{38}, \mathbf{68}, \mathbf{152}, \mathbf{255}, \mathbf{337}, \mathbf{420}, \mathbf{486}\},$$

$$\mathbf{S}_{21} = \{\mathbf{39}, \mathbf{40}, \mathbf{70}, \mathbf{154}, \mathbf{193}, \mathbf{339}, \mathbf{422}, \mathbf{488}\},$$

$$\mathbf{S}_{22} = \{\mathbf{41}, \mathbf{42}, \mathbf{72}, \mathbf{156}, \mathbf{195}, \mathbf{341}, \mathbf{424}, \mathbf{490}\},$$
$$\mathbf{S}_{23} = \{\mathbf{43}, \mathbf{44}, \mathbf{74}, \mathbf{158}, \mathbf{197}, \mathbf{343}, \mathbf{426}, \mathbf{492}\},$$
$$\mathbf{S}_{24} = \{\mathbf{45}, \mathbf{46}, \mathbf{76}, \mathbf{160}, \mathbf{199}, \mathbf{345}, \mathbf{428}, \mathbf{494}\},$$
$$\mathbf{S}_{25} = \{\mathbf{47}, \mathbf{48}, \mathbf{78}, \mathbf{162}, \mathbf{201}, \mathbf{347}, \mathbf{430}, \mathbf{496}\},$$
$$\mathbf{S}_{26} = \{\mathbf{49}, \mathbf{50}, \mathbf{80}, \mathbf{164}, \mathbf{203}, \mathbf{349}, \mathbf{432}, \mathbf{498}\},$$
$$\mathbf{S}_{27} = \{\mathbf{51}, \mathbf{52}, \mathbf{82}, \mathbf{166}, \mathbf{205}, \mathbf{351}, \mathbf{434}, \mathbf{500}\},$$
$$\mathbf{S}_{28} = \{\mathbf{53}, \mathbf{54}, \mathbf{84}, \mathbf{168}, \mathbf{207}, \mathbf{353}, \mathbf{436}, \mathbf{502}\},$$
$$\mathbf{S}_{29} = \{\mathbf{55}, \mathbf{56}, \mathbf{86}, \mathbf{170}, \mathbf{209}, \mathbf{355}, \mathbf{438}, \mathbf{504}\},$$
$$\mathbf{S}_{30} = \{\mathbf{57}, \mathbf{58}, \mathbf{88}, \mathbf{172}, \mathbf{211}, \mathbf{357}, \mathbf{440}, \mathbf{506}\},$$
$$\mathbf{S}_{31} = \{\mathbf{59}, \mathbf{60}, \mathbf{90}, \mathbf{174}, \mathbf{213}, \mathbf{359}, \mathbf{442}, \mathbf{508}\},$$
$$\mathbf{S}_{32} = \{\mathbf{61}, \mathbf{62}, \mathbf{92}, \mathbf{176}, \mathbf{215}, \mathbf{361}, \mathbf{444}, \mathbf{510}\}.$$

The set of the remaining column indexes in $\mathbf{H}_{256\times512}$ is the following:

$$\begin{aligned} S_R = \{&65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99, 101, 103,\\ &105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 129, 131, 133, 135,\\ &137, 139, 141, 143, 145, 147, 149, 151, 153, 155, 157, 159, 161, 163, 165, 167,\\ &169, 171, 173, 175, 177, 179, 181, 183, 185, 187, 189, 191, 192, 194, 196, 198,\\ &200, 202, 204, 206, 208, 210, 212, 214, 216, 218, 220, 222, 224, 226, 228, 230,\\ &232, 234, 236, 238, 240, 242, 244, 246, 248, 250, 252, 254, 256, 257, 258, 259,\\ &260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275,\\ &276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291,\\ &292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307,\\ &308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 322, 324, 326,\\ &328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 348, 350, 352, 354, 356, 358,\\ &360, 362, 364, 366, 368, 370, 372, 374, 376, 378, 380, 382, 385, 387, 389, 391,\\ &393, 395, 397, 399, 401, 403, 405, 407, 409, 411, 413, 415, 417, 419, 421, 423,\\ &425, 427, 429, 431, 433, 435, 437, 439, 441, 443, 445, 447, 449, 451, 453, 455,\\ &457, 459, 461, 463, 465, 467, 469, 471, 473, 475, 477, 479, 481, 483, 485, 487,\\ &489, 491, 493, 495, 497, 499, 501, 503, 505, 507, 509, 511\}. \end{aligned}$$

8.4 Appendix IV

Example for possible order of symbols in the RLL ECC obtained from the CCSDS LDPC code defined by matrix (19):

0, **31**, **47**, **89**, **96**, **180**, **205**, **224**, 32, 34, 36, 38, 40, 42, 44, 46, **1**, **2**, **49**, **91**, **98**, **182**, **207**, **226**, 48, 50, 52, 54, 56, 58, 60, 62, **3**, **4**, **51**, **93**, **100**, **184**, **209**, **228**, 64, 66, 68, 70, 72, 74, 76, 78, **5**, **6**, **53**, **95**, **102**, **186**, **211**, **230**, 80, 82, 84, 86, 88, 90, 92, 94, **7**, **8**, **55**, **65**, **104**, **188**, **213**, **232**, 97, 99, 101, 103, 105, 107, 109, 111, **9**, **10**, **57**, **67**, **106**, **190**, **215**, **234**, 113, 115, 117, 119, 121, 123, 125, 127, **11**, **12**, **59**, **69**, **108**, **160**, **217**, **236**, 128, 129, 130, 131, 132, 133, 134, 135, **13**, **14**, **61**, **71**, **110**, **162**, **219**, **238**, 136, 137, 138, 139, 140, 141, 142, 143, **15**, **16**, **63**, **73**, **112**, **164**, **221**, **240**, 144, 145, 146, 147, 148, 149, 150, 151, **17**, **18**, **33**, **75**, **114**, **166**, **223**, **242**, 152, 153, 154, 155, 156, 157, 158, 159, **19**, **20**, **35**, **77**, **116**, **168**, **193**, **244**, 161, 163, 165, 167, 169, 171, 173, 175, **21**, **22**, **37**, **79**, **118**, **170**, **195**, **246**, 177, 179, 181, 183, 185, 187, 189, 191, **23**, **24**, **39**, **81**, **120**, **172**, **197**, **248**, 192, 194, 196, 198, 200, 202, 204, 206, **25**, **26**, **41**, **83**, **122**, **174**, **199**, **250**, 208, 210, 212, 214, 216, 218, 220, 222, **27**, **28**, **43**, **85**, **124**, **176**, **201**, **252**, 225, 227, 229, 231, 233, 235, 237, 239, **29**, **30**, **45**, **87**, **126**, **178**, **203**, **254**, 241, 243, 245, 247, 249, 251, 253, 255.

The modifier in the above example could be, for example, the following: $\mathbf{m} = (0, 1, 3, 5, 9, \ldots, 29)$. The indexes denote positions of ones in the modifier.

8.5 Appendix V

Indexes of subsets in which the modifier can invert an uneven number of symbols in a codeword of the CCSDS LDPC code defined by (23):

$$S_{80} = \{\mathbf{80}, \mathbf{256}, \mathbf{603}, \mathbf{830}, \mathbf{1\,102}, \mathbf{1\,454}, \mathbf{1\,533}, \mathbf{1\,637}, \mathbf{2\,124}, \mathbf{2\,516}, \mathbf{2\,786}, \mathbf{3\,044}$$
$$\mathbf{3\,146}, \mathbf{3\,497}, \mathbf{3\,666}, \mathbf{4\,016}, \mathbf{4\,168}, \mathbf{4\,475}, \mathbf{4\,732}, \mathbf{5\,008}, \mathbf{5\,190}, \mathbf{5\,397}, \mathbf{5\,719}$$
$$\mathbf{5\,982}, \mathbf{6\,212}, \mathbf{6\,611}, \mathbf{6\,669}, \mathbf{6\,925}, \mathbf{7\,234}, \mathbf{7\,481}, \mathbf{7\,781}, \mathbf{8\,006}\}$$

$$S_{81} = \{\mathbf{81}, \mathbf{257}, \mathbf{604}, \mathbf{831}, \mathbf{1\,103}, \mathbf{1\,455}, \mathbf{1\,534}, \mathbf{1\,638}, \mathbf{2\,125}, \mathbf{2\,517}, \mathbf{2\,787}, \mathbf{3\,045}$$
$$\mathbf{3\,147}, \mathbf{3\,498}, \mathbf{3\,667}, \mathbf{4\,017}, \mathbf{4\,169}, \mathbf{4\,476}, \mathbf{4\,733}, \mathbf{5\,009}, \mathbf{5\,191}, \mathbf{5\,398}, \mathbf{5\,720}$$
$$\mathbf{5\,983}, \mathbf{6\,213}, \mathbf{6\,612}, \mathbf{6\,670}, \mathbf{6\,926}, \mathbf{7\,235}, \mathbf{7\,482}, \mathbf{7\,782}, \mathbf{8\,007}\}$$

$$\vdots$$

$$S_{183} = \{\mathbf{183}, \mathbf{359}, \mathbf{706}, \mathbf{933}, \mathbf{1\,046}, \mathbf{1\,205}, \mathbf{1\,636}, \mathbf{1\,740}, \mathbf{2\,108}, \mathbf{2\,227}, \mathbf{2\,636}, \mathbf{2\,889}$$
$$\mathbf{3\,089}, \mathbf{3\,249}, \mathbf{3\,608}, \mathbf{3\,769}, \mathbf{4\,169}, \mathbf{4\,271}, \mathbf{4\,578}, \mathbf{4\,600}, \mathbf{4\,835}, \mathbf{5\,293}, \mathbf{5\,500}$$
$$\mathbf{5\,822}, \mathbf{6\,085}, \mathbf{6\,203}, \mathbf{6\,772}, \mathbf{7\,028}, \mathbf{7\,337}, \mathbf{7\,584}, \mathbf{7\,884}, \mathbf{8\,109}\}.$$

The other sets between those explicitly given could be obtained by simply increasing each number in the previous set by 1. In case that one would like to use indexing

beginning with 0 and starting from left in the final word it is necessary to subtract 18 from all numbers listed in Appendix VI (8.6).

8.6 Appendix VI

Example of possible order of symbols in the RLL ECC obtained from the CCSDS LDPC code defined by matrix (23):

18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,

80, **256**, **603**, **830**, **1 102**, **1 454**, **1 533**, **1 637**, **2 124**, **2 516**, **2 786**, **3 044**, **3 146**,
3 497, **3 666**, **4 016**, **4 168**, **4 271**, **4 475**, **4 732**, **5 008**, **5 190**, **5 397**, **5 719**, **5 982**,
6 212, **6 669**, **6 925**, **7 234**, **7 481**, **7 781**, **8 006**,

63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 184, 185, 186, 187, 188,
189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202,

81, **257**, **643**, **831**, **1 103**, **1 455**, **1 534**, **1 638**, **2 125**, **2 517**, **2 787**, **3 045**, **3 147**,
3 498, **3 667**, **4 017**, **4 169**, **4 272**, **4 476**, **4 733**, **5 009**, **5 191**, **5 398**, **5 720**, **5 983**,
6 213, **6 670**, **6 926**, **7 235**, **7 482**, **7 782**, **8 007**,

203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 200, 201, 202,
236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248,

⋮

7 978, 7 979, 7 980, 7 981, 7 982, 7 983, 7 984, 7 985, 7 986, 7 987, 7 988, 7 989, 7 990, 7 991,
7 992, 7 993, 7 994, 7 995, 7 996, 7 997, 7 998, 7 999, 8 000, 8 001, 8 002, 8 003, 8 004, 8 005,
8 110, 8 111, 8 112, 8 113, 8 114, 8 115, 8 116, 8 117, 8 118, 8 119, 8 120, 8 121, 8 122, 8 123,
8 124, 8 125, 8 126, 8 127,

183, **359**, **706**, **933**, **1 046**, **1 205**, **1 534**, **1 636**, **1 740**, **2 108**, **2 227**, **2 636**, **2 889**,
3 089, **3 249**, **3 608**, **3 769**, **4 271**, **4 578**, **4 600**, **4 835**, **5 293**, **5 500**, **5 822**, **6 085**,
6 203, **6 315**, **6 772**, **7 028**, **7 337**, **7 584**, **7 884**, **8 109**,

8 128, 8 129, 8 130, 8 131, 8 132, 8 133, 8 134, 8 135, 8 136, 8 137, 8 138, 8 139, 8 140, 8 141,
8 142, 8 143, 8 144, 8 145, 8 146, 8 147, 8 148, 8 149, 8 150, 8 151, 8 152, 8 153, 8 154, 8 155,
8 156, 8 157, 8 158, 8 159, 8 160, 8 161, 8 162, 8 163, 8 164, 8 165, 8 166, 8 167, 8 168, 8 169,
8 170, 8 171, 8 172, 8 173,

8 174, **8 175**

The modifier in the base code indexing could be the following in this example $\mathbf{m} = (80, 81, \ldots, 183, 8\,174)$. The indexes denote positions of ones in the modifier. In case that one would like to use indexing beginning with 0 and starting from the left in the final word it is necessary to subtract 18 from all numbers listed in Appendix VI (8.6).

REFERENCES

[1] MacWilliams, F. J.—Sloane, N. J. A.: The Theory of Error-Correcting Codes. North-Holland Publishing Co., 1977.

[2] Rao, T. R. N.—Fujiwara, E.: Error-Control Coding for Computer Systems. Prentice-Hall, Inc., 1989.

[3] Pless, V.—Brualdi, R. A.—Huffman, W. C.: Handbook of Coding Theory. Elsevier Science Inc., 1998.

[4] Tomlinson, M.—Tjhai, C. J.—Ambroze, M. A.—Ahmed, M.—Jibril, M.: Error-Correction Coding and Decoding: Bounds, Codes, Decoders, Analysis and Applications. Springer, 2017, doi: 10.1007/978-3-319-51103-0.

[5] Myers, O.: Codes and Translations. Electrical Engineering, Vol. 68, 1949, No. 11, pp. 950–950, doi: 10.1109/EE.1949.6443215.

[6] Freiman, C. V.—Wyner, A. D.: Optimum Block Codes for Noiseless Input Restricted Channels. Information and Control, Vol. 7, 1964, No. 3, pp. 398–415, doi: 10.1016/S0019-9958(64)90486-3.

[7] Kautz, W.: Fibonacci Codes for Synchronization Control. IEEE Transactions on Information Theory, Vol. 11, 1965, No. 2, pp. 284–292, doi: 10.1109/TIT.1965.1053772.

[8] Gabor, A.: Adaptive Coding for Self-Clocking Recording. IEEE Transactions on Electronic Computers, Vol. EC-16, 1967, No. 6, pp. 866–868, doi: 10.1109/PGEC.1967.264753.

[9] Franaszek, P. A.: Sequence-State Coding for Digital Transmission. The Bell System Technical Journal, Vol. 47, 1968, No. 1, pp. 143–157, doi: 10.1002/j.1538-7305.1968.tb00034.x.

[10] Franaszek, P. A.: Efficient Code for Digital Magnetic Recording. IBM Technical Disclosure Bulletin, Vol. 23, 1981, No. 9, pp. 4375–4378.

[11] Patel, A. M.: Improved Encoder and Decoder for a Byte-Oriented Rate 8/9 (0, 3) Code. IBM Technical Disclosure Bulletin, Vol. 28, 1985, pp. 1938–1940.

[12] Siegel, P.: Recording Codes for Digital Magnetic Storage. IEEE Transactions on Magnetics, Vol. 21, 1985, No. 5, pp. 1344–1349, doi: 10.1109/TMAG.1985.1063972.

[13] Blahut, R. E.: Digital Transmission of Information. Addison-Wesley, 1990.

[14] Schouhamer-Immink, K. A.: Coding Techniques for Digital Recorders. Prentice Hall, 1991.

[15] Süral, A.—Sezer, E. G.—Ertuğrul, Y.—Arikan, O.—Arikan, E.: Terabits-per-Second Throughput for Polar Codes. 2019 IEEE 30$^{\text{th}}$ International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops), 2019, pp. 1–7, doi: 10.1109/PIMRCW.2019.8880815.

[16] Sanvicente, E.: Understanding Error Control Coding. Springer, 2019, doi: 10.1007/978-3-030-05840-1.

[17] Rakús, M.—Farkaš, P.—Páleník, T.—Daniš, A.: Five Times Extended Reed-Solomon Codes Applicable in Memory Storage Systems. IEEE Letters of the Computer Society, Vol. 2, 2019, No. 2, pp. 9–11, doi: 10.1109/LOCS.2019.2911517.

[18] Moon, T. K.: Error Correction Coding: Mathematical Methods and Algorithms. John Wiley & Sons, 2020.

[19] An, W.—Médard, M.—Duffy, K. R.: CRC Codes as Error Correction Codes. ICC 2021 – IEEE International Conference on Communications, 2021, pp. 1–6, doi: 10.1109/ICC42927.2021.9500279.

[20] Liu, W.—Chen, L.—Liu, X.: A Weighted Sum Based Construction of PAC Codes. IEEE Communications Letters, Vol. 27, 2023, No. 1, pp. 28–31, doi: 10.1109/LCOMM.2022.3209082.

[21] Kovačević, M.—Vukobratović, D.: Asymptotic Behavior and Typicality Properties of Runlength-Limited Sequences. IEEE Transactions on Information Theory, Vol. 68, 2022, No. 3, pp. 1638–1650, doi: 10.1109/TIT.2021.3134871.

[22] Mambou, E. N.—Tonnellier, T.—Gross, W. J.: Improved DC-Free Run-Length Limited 4B6B Codes for Concatenated Schemes. IEEE Access, Vol. 10, 2022, pp. 21847–21852, doi: 10.1109/ACCESS.2022.3152553.

[23] Park, S. J.—Lee, Y.—No, J. S.: Iterative Coding Scheme Satisfying GC Balance and Run-Length Constraints for DNA Storage with Robustness to Error Propagation. Journal of Communications and Networks, Vol. 24, 2022, No. 3, pp. 283–291, doi: 10.23919/JCN.2022.000008.

[24] Zhong, X.—Cai, K.—Song, G.—Wang, W.—Zhu, Y.: Constrained Coding and Deep Learning Aided Threshold Detection for Resistive Memories. IEEE Communications Letters, Vol. 26, 2022, No. 4, pp. 803–807, doi: 10.1109/LCOMM.2022.3148292.

[25] Immink, K. A. S.: Innovation in Constrained Codes. IEEE Communications Magazine, Vol. 60, 2022, No. 10, pp. 20–24, doi: 10.1109/MCOM.002.2200249.

[26] Herro, M. A.—Deng, R. H.: Error-Correcting DC-Free Binary Transmission Codes for Fiber Optic Digital Communications. Proceedings of the Conference on Information Sciences and Systems (CISS 1987), 1987, pp. 559–564.

[27] O'Reilly, J. J.—Popplewell, A.: Design and Spectral Characterisation of Error Correcting Line Codes. 1988 IEEE International Symposium on Information Theory (ISIT 1988), 1988.

[28] Popplewell, A.—O'Reilly, J. J.: Performance Aspects of Error Correcting Line Codes. Second IEE National Conference on Telecommunications 1989, 1989, pp. 47–52.

[29] Ferreira, H. C.—Hope, J. F.—Nel, A. L.: Binary Rate Four Eighths, Run-length Constrained, Error Correcting Magnetic Recording Modulation Code. IEEE Transactions on Magnetics, Vol. 22, 1986, No. 5, pp. 1197–1199, doi: 10.1109/TMAG.1986.1064518.

[30] Lee, P.—Wolf, J. K.: Combined Error Correction/Modulation Codes. IEEE Transactions on Magnetics, Vol. 23, 1987, No. 5, pp. 3681–3683, doi: 10.1109/TMAG.1987.1065193.

[31] Lin, Y.—Wolf, J. K.: Combined ECC/RLL Codes. IEEE Transactions on Magnetics, Vol. 24, 1988, No. 6, pp. 2527–2529, doi: 10.1109/20.92163.

[32] Ferreira, H. C.—Lin, S.: Error and Erasure Control (d,k) Block Codes. IEEE Transactions on Information Theory, Vol. 37, 1991, No. 5, pp. 1399–1408, doi: 10.1109/18.133257.

[33] Nasiri-Kenari, M.—Rushforth, C. K.: Some Construction Methods for Error-Correcting (d,k) Codes. IEEE Transactions on Communications, Vol. 42, 1994, No. 234, pp. 958–965, doi: 10.1109/TCOMM.1994.580204.

[34] Nguyen, T. T.—Cai, K.—Schouhamer Immink, K. A.—Kiah, H. M.: Capacity-Approaching Constrained Codes with Error Correction for DNA-Based Data Storage. IEEE Transactions on Information Theory, Vol. 67, 2021, No. 8, pp. 5602–5613, doi: 10.1109/TIT.2021.3066430.

[35] Farkaš, P.—Weinrichter, H.: Transcontrol Codes with Run-Length Limitation. International Journal of Electronics and Communications (AEÜ), Vol. 50, 1996, No. 6, pp. 353–356.

[36] Lee, T. A.—Heegard, C.: An Inversion Technique for the Design of Binary Convolutional Codes for the 1-DN Channel. 1985 Conference on Information Sciences and Systems, 1985.

[37] Calderbank, A.—Heegard, C.—Lee, T. A.: Binary Convolutional Codes with Application to Magnetic Recording. IEEE Transactions on Information Theory, Vol. 32, 1986, No. 6, pp. 797–815, doi: 10.1109/TIT.1986.1057245.

[38] Ferreira, H. C.—Wright, D. A.—Nel, A. L.: Hamming Distance Preserving Mappings and Trellis Codes with Constrained Binary Symbols. IEEE Transactions on Information Theory, Vol. 35, 1989, No. 5, pp. 1098–1103, doi: 10.1109/18.42229.

[39] Deng, R. H.—Herro, M. A.: DC-Free Coset Codes. IEEE Transactions on Information Theory, Vol. 34, 1988, No. 4, pp. 786–792, doi: 10.1109/18.9775.

[40] O'Reilly, J. J.—Popplewell, A.: A Further Note on DC-Free Coset Codes. IEEE Transactions on Information Theory, Vol. 36, 1990, No. 3, pp. 675–676, doi: 10.1109/18.54889.

[41] Popplewell, A.—Kokkos, A.—O'Reilly, J. J.: Aspects of Combined Coding and Linear Modulation and Its Application to Future Personal Communication Systems. 1991 Sixth International Conference on Mobile Radio and Personal Communications, IET, 1991, pp. 87–92.

[42] Popplewell, A.—O'Reilly, J.: A Simple Strategy for Constructing a Class of DC-Free Error-Correcting Codes with Minimum Distance 4. IEEE Transactions on Information Theory, Vol. 41, 1995, No. 4, pp. 1134–1137, doi: 10.1109/18.391256.

[43] Farkaš, P.—Schindler, F.: Run Length Limited Error Control Codes Construction Based on One Control Matrix Property. Journal of Electrical Engineering, Vol. 68, 2017, No. 4, pp. 322–324, doi: 10.1515/jee-2017-0046.

[44] Farkas, P.: Turbo-Codes with RLL Properties. IEE Colloquium on Turbo Codes in Digital Broadcasting – Could It Double Capacity?, IET, 1999, doi: 10.1049/ic:19990793.

[45] Sechny, M.—Farkas, P.: Some New Runlength-Limited Convolutional Codes. IEEE Transactions on Communications, Vol. 47, 1999, No. 7, pp. 962–966, doi:

10.1109/26.774834.

[46] Farkas, P.—Pusch, W.—Taferner, M.—Weinrichter, H.: Turbo-Codes with Run Length Contraints. International Journal of Electronics and Communications (AEÜ), Vol. 53, 1999, No. 3, pp. 161–166.

[47] Chomist, R.—Farkaš, P.: Some Extended Hamming Transcontrol Codes. Radioelektronika 2005 – 15th International Czech-Slovak Scientific Conference, VUT v Brně, 2005, pp. 342–345.

[48] Chomist, R.—Farkaš, P.: Extended Golay Code with Transcontrol Properties. The 6th International Conference on Digital Signal Processing and Multimedia Communications. Proceedings of the DSP – MCOM 2005, 2005, pp. 102–105.

[49] Farkas, P.—Chomist, R.: Reed Muller-Codes with Run Length Properties. SympoTIC'04. Joint 1st Workshop on Mobile Future and Symposium on Trends in Communications, 2004, pp. 66–69, doi: 10.1109/TIC.2004.1409500.

[50] Farkašová, K.—Farkaš, P.—Rakús, M.—Ružický, E.—Silva, A.—Gameiro, A.: Construction of Error Control Run Length Limited Codes Exploiting Some Parity Matrix Properties. Journal of Electrical Engineering, Vol. 66, 2015, No. 3, pp. 182–184, doi: 10.2478/jee-2015-0030.

[51] Farkaš, P.—Schindler, F.: Construction for Obtaining Trellis Run Length Limited Error Control Codes from Convolutional Codes. Journal of Electrical Engineering, Vol. 68, 2017, No. 5, pp. 401–404, doi: 10.1515/jee-2017-0074.

[52] Farkaš, P.—Janvars, T.—Farkašová, K.—Ružický, E.: On Run-Length Limited Error Control Codes Constructed from Binary Product Codes. Journal of Electrical Engineering, Vol. 69, 2018, No. 3, pp. 245–249, doi: 10.2478/jee-2018-0033.

[53] Farkaš, P.—Rakús, M.—Páleník, T.: A New Technique for Incorporating RLL Properties Into 5G LDPC Codes Without Additional Redundancy. Wireless Personal Communications, Vol. 119, 2021, pp. 749–762, doi: 10.1007/s11277-021-08235-3.

[54] TC Synchronisation and Channel Coding. Recommended Standard CCSDS 131.0-B-4 (Blue Book), Issue 4. Consultative Committee for Space Data Systems (CCSDS), 2021.

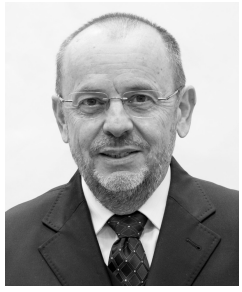
[55] TM Synchronisation and Channel Coding. Recommended Standard CCSDS 131.0-B-4 (Blue Book), Issue 4. Consultative Committee for Space Data Systems (CCSDS), 2022.

[56] Gallager, R. G.: Low Density Parity Check Codes. Ph.D. Thesis. Massachusetts Institute of Technology, 1960.

[57] Farkaš, P.—Páleník, T.: On Soft Decoding of Some Binary RLL-Transmission Codes in Systems with Coherent BPSK Modulation. 2022 Cybernetics & Informatics (K&I), IEEE, 2022, pp. 1–5, doi: 10.1109/KI55792.2022.9925949.

[58] Massey, J. L.: Deep-Space Communications and Coding: A Marriage Made in Heaven. In: Hagenauer, J. (Ed.): Advanced Methods for Satellite and Deep Space Communications. Springer, Berlin, Heidelberg, Lecture Notes in Control and Information Sciences, Vol. 182, 1992, pp. 1–17, doi: 10.1007/BFb0036046.

[59] Farkaš, P.—Rakús, M.: On Run Length Limitation in Codewords of Some Reed Solomon Codes. 2022 (Submitted to Wireless Personal Communications).



Peter Farkaš is with the Institute of Multimedia Information and Communication Technologies, Slovak University of Technology in Bratislava (STU) and with the Institute of Applied Informatics, Faculty of Informatics, Pan European University in Bratislava as a Professor. From 2002 until 2007 he was Visiting Professor at Kingston University, UK and Senior Researcher at Siemens PSE. In 2003 SIEMENS named him VIP for his innovations and patents. In 2004 he was awarded the Werner von Siemens Excellence Award for research results on two-dimensional Complete Complementary Codes. From 2008 to 2009 he worked also as a consultant in the area of Software Defined Radio for SandBridge Tech. (USA). He was the leader of a team from STU in projects funded by the European Community under the 5FP and 6FP "Information Society Technologies Programs": NEXWAY IST-2001-37944 (Network of Excellence in Wireless Applications and Technology) and CRUISE (Creating Ubiquitous Intelligent Sensing Environments) FP6 IST-2005-4-027738, (2006–2007). His research interests include Coding, Communications Theory and sequences for CDMA. He has published 1 book, about 45 papers in reviewed scientific journals and up to 100 papers in international conferences. He is the author or co-author of 7 patents. He is and was serving in TPC of about 60 international conferences and presented 12 invited lectures. As an IEEE volunteer, he was serving in the IEEE Czechoslovakia Section Executive Committee in different positions from 1992 to 2014 and from 2005 to 2006 he served as a chair of the Conference Coordinator Subcommittee in IEEE Region 8. He organized the IEEE R8 Conference EUROCON 2001 and was the chairman of SympoTIC'03, SympoTIC'04, SympoTIC'06 and co-organizer of the Winter School on Coding and Information Theory 2005. Since 2016 he has been serving as Vice-Chair of the computer chapter in the IEEE Czechoslovakia Section.

Martin Rakús studied radio electronics and graduated from the Slovak University of Technology in 2001. In 2004 he received his Ph.D. from the Slovak University of Technology and in 2020 he became Associate Professor at the same institute. Since 1995 he has been with the Institute of Multimedia Information and Communication Technology, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia. His primary research interests are error control coding and digital communication systems. He is a member of the IEEE.