Analysis of Minimal LDPC Decoder System on a Chip Implementation

Tomáš PALENIK[1], *Peter FARKAS*[1], *Martin RAKUS*[1], *Jan DOBOS*[2]

[1] Dept. of Telecommunications, Slovak University of Technology, Ilkovicova 3, 81219 Bratislava, Slovakia
[2] Faculty of Informatics, Paneuropean University, Tematinska 10, 85105 Bratislava, Slovakia

tomas.palenik@ieee.org, p.farkas@ieee.org, rakus@ktl.elf.stuba.sk, jan.dobos@paneurouni.com

Abstract. *This paper presents a practical method of potential replacement of several different Quasi-Cyclic Low-Density Parity-Check (QC-LDPC) codes with one, with the intention of saving as much memory as required to implement the LDPC encoder and decoder in a memory-constrained System on a Chip (SoC). The presented method requires only a very small modification of the existing encoder and decoder, making it suitable for utilization in a Software Defined Radio (SDR) platform. Besides the analysis of the effects of necessary variable-node value fixation during the Belief Propagation (BP) decoding algorithm, practical standard-defined code parameters are scrutinized in order to evaluate the feasibility of the proposed LDPC setup simplification. Finally, the error performance of the modified system structure is evaluated and compared with the original system structure by means of simulation.*

Keywords

LDPC code shortening, System on a Chip, fixed nodes decoder, Adaptive Coding and Modulation.

1. Introduction

The shortening of Reed Solomon (RS) codes presents a known and widely used technique of code parameter adaptation for practical implementations. It has been well described in literature [1]. On the other hand, a similar method adapted for the use with the more modern Low-Density Parity-Check (LDPC) codes is a relatively recent topic. A theoretical approach regarding various shortening algorithms for selecting the optimal subset of variable nodes to be fixed has been thoroughly evaluated in [2] and [3]. This paper elaborates on this analysis by providing insight not just regarding the code structure, but also analyzing the effect of variable node value fixation on practical Belief Propagation (BP) decoding algorithms. Moreover, this text focuses on the even more practical approach – evaluation of potential LDPC decoder with respect to the limitations of a resource limited System on a Chip (SoC) implementation. Such devices usually implement IEEE 802.15.4 standard [4], with different Forward Error Correction (FEC) schemes – the current version of the standard specifies the use of RS code along with an optional internal Convolutional Code. Considering the experience with other standards, such as IEEE 802.16 [5] and IEEE 802.11 [6], it is reasonable to evaluate the possibility of future inclusion of the LDPC code family to FEC techniques used in low-resource standards and systems. Our theoretical method is in some aspects similar to the LDPC code puncturing method presented in [7] and further analyzed in [8], while being different in one key aspect – the fixed bits are always part of the information portion of the codeword, so their value is always known, thus enabling to initialize the prior Log Likelihood Ratios (LLRs) of the fixed bits to infinite confidence, instead of zero.

This paper focuses on preliminary evaluation of the potential of LDPC code utilization in the context of a very resource-constrained platform. The next section reintroduces the topic of LDPC codes with focus on the practical Quasi-Cyclic LDPC (QC-LDPC). The third section reviews the code shortening technique and proposes a detailed adaptation algorithm to be used with LDPC codes in both the transmitter and receiver. The third section also provides some insights into the operation of the Min-Sum decoding algorithm and the effects of node value fixation (necessary for code shortening) on the values being exchanged between the nodes during the message passing decoding algorithm. The fourth section discusses practical issues, such as the potential for simplification of the standard-defined [5] LDPC code set. The fifth section evaluates the decoding error performance by means of simulations and compares the achieved performance with the original system. The final section contains a brief summary and concludes the paper.

2. LDPC Codes Review

This section provides a brief recapitulation of known LDPC terminology required in the following sections. The LDPC code is a Linear Block Code (LBC) defined by its sparse parity check matrix **H** [9], such as the one depicted in Fig. 1. According to Fig. 1, for purpose of formulation of decoder equations, the columns of this matrix will be indexed by *col* and rows by *row*. As with any LBC, basic code parameters can be summarized in a triplet (n, k, d_{min}) or pair (n, k); where n denotes the codeword length, k the number of information symbols and d_{min} the minimum code distance. Symbols $N(row)$ and $M(col)$ define the so called neighborhoods – a set of nodes incident with a given check- or variable-node; where c_{row} denotes the *row*-th check-node and v_{col} the *col*-th variable-node.

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad N(row = 2) = N(c_2) = \{v_2, v_3, v_6\}$$

$$M(col = 4) = M(v_4) = \{c_3\}$$

Fig. 1. Example of a small parity check matrix of a (6, 3) code along with neighborhoods for variable node *col* = 4 and check node *row* = 2.

The graphical representation of the parity check matrix is called a Tanner graph. This is a bipartite graph consisting of two types on nodes – variable nodes each corresponding to codeword bits, and check nodes each associated with a parity equation. Tanner graph edges always connect a check node with variable nodes that participate in its parity equation. Tanner graph for the parity check matrix from Fig. 1 is depicted in Fig. 2.

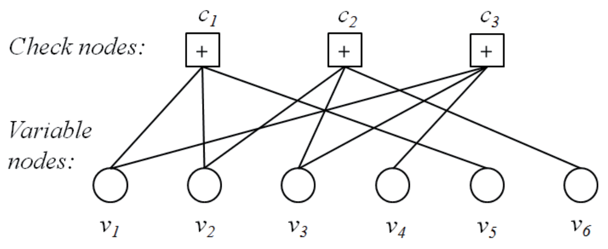

Fig. 2. Tanner graph for the code defined by H matrix in Fig. 1.

The main purpose of the Tanner graph is to visualize the LDPC code structure in order to support definition and visualization of various LDPC decoding schemes. More details regarding the LDPC code structure and decoding are provided by [9].

3. LDPC Code Shortening

The main idea of LDPC code shortening is similar to the known methods of RS code shortening [1]: the selected bits of the data words are assumed fixed, usually but not necessarily set to zero. Their value is defined in advance before the encoding in the transmitter and also known in advance to the receiver. These bits cannot be used to transmit useful information. Their purpose is merely to enable utilization of a code with given parameters (n, k) in situation where a code with different parameters (n', k') would be more appropriate. Fixed bits are inserted to data stream and after encoding they are discarded so that they are not transmitted at all.

This transformation is widely used with RS codes to overcome their limitations, where codeword length n is bound to the size of the underlying Galois Field [1]. This paper elaborates on the novel idea of using a similar technique with LDPC codes [2], [3]. In this context, the proposed idea can be utilized for practical purpose of system complexity reduction, and also potential improvement of the decoder error performance. These two potential improvements are analyzed in the following sections.

The focus of this section is the thorough description of technical steps necessary to implement the proposed modification. The channel models can be Binary Symmetric Channel (BSC) if hard-decision is made before decoding (used with bit-flipping decoders) or Adaptive White Gaussian Noise (AWGN) channel for a more advanced Soft-Input Soft-Output (SISO) log-likelihood BP decoding. The model shown in Fig. 3 represents a simple standard along with usual symbol notation [9].

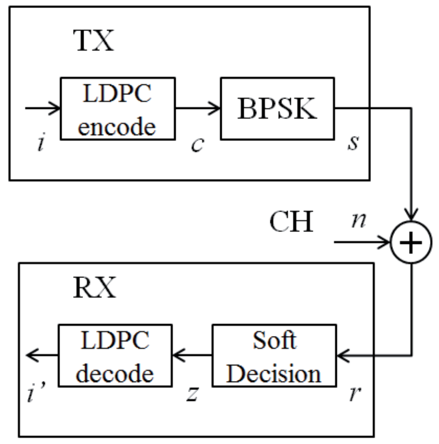

Fig. 3. The system model for a noisy AWGN channel with SISO decoder. TX-transmitter, RX-receiver, CH-channel, *i*-information word, *c*-codeword, *s*-transmitted signal, *n*-noise signal, *r*-received signal, *z*-codeword estimation, *i'*-information word estimation.

The process description of modified LDPC code shortening in the transmitter and corresponding inverse process in the receiver provided in the next subsection is referred to as Algorithm A1.

3.1 Algorithm A1:

1. TX: Set part of the data bits to zero in the transmitter before encoding. These bits are to be called fixed bits with f denoting their number.
2. TX: Encode the whole data word with a systematic LDPC code.
3. TX: Discard the redundant filler bits of the systematic part of the codeword – the all zero part.
4. TX: Transmit the shortened codeword.
5. RX: Insert appropriate filler values to the systematic part of the received noisy codeword in the receiver.

6. RX: Decode noisy codeword using original decoder with unchanged parameters.
7. RX: Discard redundant filler values.

For convenience, the process is also depicted in Fig. 4, with focus on visualizing the insertion and discarding of the filler bits before and after LDPC encoding, along with the equivalent processes in the receiver. The left side of the picture shows standard system operation – without code shortening, while the right side shows the process flow in a system implementing the described algorithm A1.

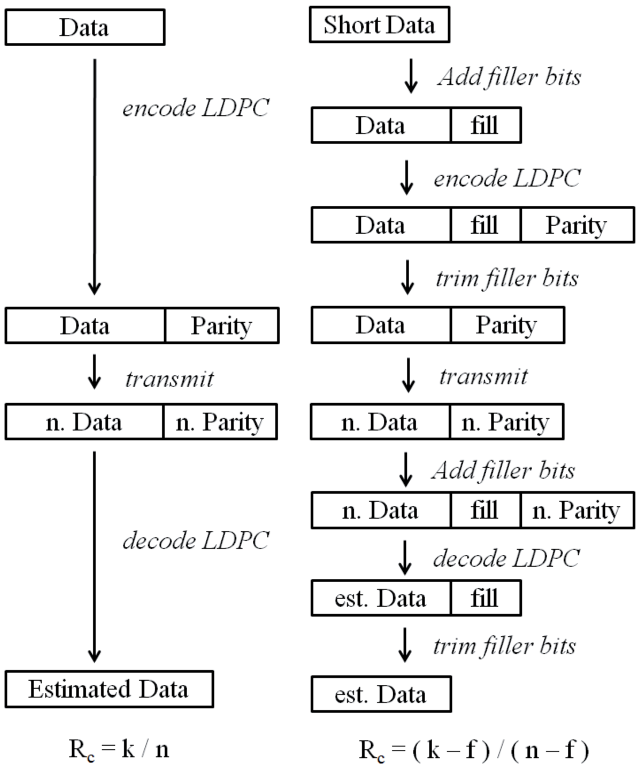

Fig. 4. Principle of LDPC code shortening by fill and removal of meaningless filler bits fixed in value. Standard system (left). System with LDPC shortening (right). The "*n.*" abbreviates "noisy" – for SISO decoding this means LLR values constructed from noisy channel observations, R_c – code rate.

There are some important implications rising from code shortening: regarding the useful data being transmitted, the new code has a smaller code rate: if the code rate of the original code is R_c, defined by the fraction k/n, then by using f filler bits, the new code rate R'_c will be equal to

$$R_c^{'} = \frac{k-f}{n-f} < R_c \,. \tag{1}$$

When comparing two codes similar in structure to the extent that all other concerns can be regarded insignificant, it is important to realize, that the code with smaller code rate should perform better in terms of error ratio – more specifically in higher Signal-to-Noise Ratio (SNR) range of the waterfall curve, where the coding gain offsets the drop of E_b (Energy per bit) associated with code rate drop (in case assuming the data rate remains constant).

3.2 LDPC Decoding

At first glance it would seem that the fixed filler bits hide a potential not only for improving the error capability of the code, but of the decoding as well. The error decoding properties of the code itself have been analyzed in detail in publications [2] and [3]. However, the decoding algorithm is a slightly different topic from the code itself. Therefore it is reasonable to evaluate this eventuality in detail. Intuitive approach unfolds as follows: since the fixed filler bits are not transmitted at all, no channel noise affects them, and so the perfect values of received symbols (+1 or -1 when using BPSK modulation - Binary Phase-Shift Keying) can be inserted in the receiver. When computing the channel LLR values, the variance of channel noise for these symbols σ_n^2 is zero which results in infinite values of their LLR metric:

$$LLR(r) = \frac{2r}{\sigma_n^2} \tag{2}$$

where r is the received channel sample as shown in Fig. 3. The infinite confidence seems promising, since based on the BP one might expect that this confidence will improve the decoding itself, by propagating some of this confidence to other variable nodes containing usable data. While this intuition seems interesting, a detailed analysis provided in the following subsection reveals that this improvement does not occur.

On the other hand, as also our simulations in later sections will confirm, there is indeed an improvement in the waterfall curve when using fixed nodes. However, this improvement originates from the lower code rate R_c', more specifically from the fact that there is a relatively higher number of parity bits per really transmitted data bit. (In absolute values, there are just few data bits transmitted while the number of parity bits remains the same).

3.3 Analysis of Min-Sum Decoding

While the intuition regarding the infinite confidence in the fixed filler nodes would suggest the propagation of this confidence to other nodes, thus improving the overall decoding, this section contradicts the intuition by means of a sophisticated example. First we review the operation of prominent and most widely used BP based (Sum-Product and Min-Sum) SISO decoding algorithm [9], [10]. The decoding is iterative, where each iteration consists of two steps: the horizontal step, conforming to one parity equation in which variable nodes send their extrinsic information $L_{row,col}$ to other variable nodes using the interconnections defined by a check node; the vertical step where these messages are collected across all parity equations defined by code structure to form a final posterior LLR estimate.

The Sum-Product algorithm defines the horizontal step equation [9]:

$$L_{row,col}^{(j)} = 2\,\text{arctanh}\left(\prod_{col' \in N(row)\backslash col} \tanh\left(\frac{Z_{row,col'}^{(j\text{-}1)}}{2} \right) \right) \tag{3}$$

with symbols $N(row)$ and $M(col)$ already defined in Sec. 2, (j) is the index of decoding iteration and symbols $L_{row,col}$ and $Z_{row,col}$ are the messages exchanged between variable nodes: $Z_{row,col}$ is the log-likelihood ratio defining that the col-th bit of the input data has the value 0 versus 1, given the information obtained via the check nodes other than check node row. $L_{row,col}$ is the LLR where the condition for check node row is satisfied when the input data bit col is fixed to value 0 versus value 1 and the other bits are independent with LLRs $Z_{row,col'}$, $col' \in N(row) \setminus col$ [10]. It is necessary to realize, that equation (3) is really a shorthand for many equations - one for each check node and each incident variable node set. This is elaborated in further subsections. For the majority of practical decoders this equation is often approximated by a computationally much simpler equation omitting the expensive hyperbolic tangent functions [9]:

$$L_{row,col}^{(j)} = \left(\prod_{col' \in N(row)\setminus col} \operatorname{sgn}\left(Z_{row,col'}^{(j\text{-}1)}\right) \right) \times \left(\min_{col' \in N(row)\setminus col} \left| Z_{row,col'}^{(j-1)} \right| \right). \quad (4)$$

The second step in the iterative LDPC decoding algorithm is the simple vertical (per variable node) summation of extrinsic messages to produce final posterior LLR estimate:

$$Z_{col}^{(j)} = Z_{col}^{(0)} + \sum_{row' \in M(col)} L_{row',col}^{(j)} . \quad (5)$$

The effect of the infinite confidence of fixed-value nodes on Min-Sum decoding procedure can be nicely demonstrated by a simple example E1 in the following subsection.

3.4 Example E1

For the LDPC code defined in Fig. 1 and second row of parity check matrix **H** (row = 2), equation (4) can be rewritten in the following way: First, let variable nodes v_2 and v_3 contain real channel observations so that their LLR metrics, denoted z_2 and z_3, will be finite values. On the other hand, let v_6 be a fixed node with infinite absolute confidence z_6. This translates to the messages propagated between the variable nodes in horizontal step. First, value $row = 2$ is substituted into (4) which really defines three equations – each for one variable node in the role of receiver of the message $L_{row,col}$. For simplicity, we now ignore the signs of the messages:

$$\overset{\text{for } col \in N(2)=\{2,3,6\}}{L_{2,col}^{(j)} = \min_{col' \in N(2)\setminus col} \left| Z_{2,col'}^{(j-1)} \right|} . \quad (6)$$

Values $Z_{row,col}$ are initialized to channel observation z_{col} before the first iteration which makes all values $Z_{row,6} = z_6$ infinite.

Further by substituting values of col we get three equations:

$$\overset{col=2;\, col' \in N(2)\setminus 2=\{3,6\}}{L_{2,2}^{(j)} = \min_{col' \in N(2)\setminus 2} \left| Z_{2,col'}^{(j-1)} \right| = \min\left(\left|z_3\right|, Inf\right) = \left|z_3\right|} , \quad (7)$$

$$\overset{col=3;\, col' \in N(2)\setminus 3=\{2,6\}}{L_{2,3}^{(j)} = \min_{col' \in N(2)\setminus 3} \left| Z_{2,col'}^{(j-1)} \right| = \min\left(\left|z_2\right|, Inf\right) = \left|z_2\right|} , \quad (8)$$

$$\overset{col=6;\, col' \in N(2)\setminus 6=\{2,3\}}{L_{2,6}^{(j)} = \min_{col' \in N(2)\setminus 6} \left| Z_{2,col'}^{(j-1)} \right| = \min\left(\left|z_2\right|, \left|z_3\right|\right) \neq Inf} \cdot \quad (9)$$

From (7) to (9) it is now clear, that whenever the infinity value of $Z_{row,6}$ enters the minimum operator, it will be discarded in favor of smaller amplitude values. Thus the infinite confidence of the fixed nodes, that intuitively seemed very promising in delivering potentially extra error correcting capability, is effectively discarded right in the first half (horizontal step) of the first decoder iteration.

The horizontal step is followed by the vertical summation (5) for each variable node, independent of other variable nodes. Therefore the variable $Z_6^{(1)}$ remains infinity, while all others $Z_{col}^{(1)}$ remain unaffected by the infinite confidence of $Z_6^{(0)}$. This remains true for all other iterations, regardless of their number.■

Example E1 can be easily generalized for any LDPC code, provided that the degree of each check node is larger than 1, which is very much the case for all practical LDPC codes. This shows how and why the infinite confidence doesn't really translate to improve decoding under Min-Sum algorithm, which is later confirmed by simulations.

4. Practical Considerations

The simple design presented in the previous section has some interesting implications for communication stack implementations on a resource-constrained SoC. One of the goals of a modern Physical (PHY) layer implementation is to provide a good ACM – an Adaptive Coding and Modulation scheme that responds to dynamically changing mobile channel conditions by adjusting the parameters of error control code and modulation scheme, in order to provide a consistent Bit Error Rate (BER) and Frame Error Rate (FER) to upper layers. In industry-wide communication standards such as IEEE 802.16e [5] and IEEE 802.11ac [6] this is achieved by specifying several codes along with the set of their supported parameters. For instance, there are 6 different LDPC codes with 4 different code rates specified for use in [5], along with a set of 19 different codeword lengths. Each one of the codes is defined by its parity check matrix that needs to be stored in memory. Given the limited resources of a SoC, this may be a considerable problem. In this section we analyze the possibility of implementing just one of the LDPC codes and using the proposed method to obtain codes with different parameters by shortening of the single implemented code. The main purpose is to analyze all code parameters given in a communication standard and to evaluate what percentage of these codes can be replaced. A second goal is to provide a simple tabular overview of which exact code parameters can be implemented in this way.

First it is necessary to review the structure of LDPC codes used in modern communication standards. The **H** matrix's size is quite large, with number of variable nodes going to thousands. Therefore a compressed form of **H** matrix is usually required. The structure of the **H** matrix is defined by its partitioning to smaller square submatrices **P**, either all zero, or rotated identity matrices. **H** matrix can be easily written in the compressed form as show in Fig. 5 [5].

$$\mathbf{H}_{b,p} = \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \cdots & \mathbf{P}_{0,p-2} & \mathbf{P}_{0,p-1} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \cdots & \mathbf{P}_{1,p-2} & \mathbf{P}_{1,p-1} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \cdots & \mathbf{P}_{2,p-2} & \mathbf{P}_{2,p-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{P}_{b-1,0} & \mathbf{P}_{b-1,1} & \mathbf{P}_{b-1,2} & \cdots & \mathbf{P}_{b-1,p-2} & \mathbf{P}_{b-1,p-1} \end{bmatrix}$$

Fig. 5. Compressed format of **H** matrix using rotation submatrices **P**, *b*-number of submatrix rows, *p*-number of submatrix columns.

The whole code structure can then be expressed by a much smaller integer-valued compact model matrix $\mathbf{H}_{b,p}$, where each integer value represents a circular shift in the appropriate rotation matrix **P**. Six different LDPC codes and 19 different codeword lengths are standardized in [5]. Table 1 provides a basic overview of the codes, along with their compressed model matrix sizes. The different codeword (and also dataword) lengths are implemented by expanding these matrices by a factor *z* ranging from 24 to 96. This defines a total set of 114 similar LDPC codes with a very flexible range of parameters.

Code	$\mathbf{H}_{b,p}$ matrix size $[k_b \times n_p]$	H min. size $[k \times n]$	H max. size $[k \times n]$
Rate 1/2	12 × 24	288 × 576	1152 × 2304
Rate 2/3 A	8 × 24	192 × 576	768 × 2304
Rate 2/3 B	8 × 24	192 × 576	768 × 2304
Rate 3/4 A	6 × 24	144 × 576	576 × 2304
Rate 3/4 B	6 × 24	144 × 576	576 × 2304
Rate 5/6	4 × 24	96 × 576	384 × 2304

Tab. 1. Overview of basic codes in standard [5]. The minimum and maximum code sizes are determined by possible values of the expansion factor *z*.

While the flexibility of code parameters is appreciated, practical applications can perform quite well with only a small subset of this very large code parameter range. For the actual full size matrix **H** parameters *k* and *n* are determined from the size of the associated model matrix $\mathbf{H}_{b,p}$ and the size of the expansion factor *z* by (11).

$$\mathbf{H}^{(k\times n)} = \text{expand}\left(\mathbf{H}_{b,p}^{(k_b\times n_p)}\right), \qquad (10)$$

$$k = z\cdot k_b, n = z\cdot n_p \qquad (11)$$

where the conceptual expansion operation was described before. The possibility of various LDPC code parameters implemented by using just one code (with code rate R_{cb} = 5/6) was analyzed with complete result summarization covering all the 114 codes provided in Tab. 2. The table is organized in four composite-columns, each one giving code parameters *n, k*, $(n-k)$ or *f*, of a target code, with target code rates $R_c = \{5/6, 3/4, 2/3, 1/2\}$. The purpose of this table is to provide information whether or not the target code can be implemented by shortening of the standard-defined highest code rate code (R_{cb} = 5/6) defined in the first composite column. If it can be replaced, also the value *f* is nonzero and specifies the number of filler bits that must be shortened from the basis code.

For example: The first code of rate R_c = 3/4 would have parameters $n = 576$ and $k = 432$ defined by the value $z = 24$. As indicated in the table, this can be replaced by shortening of a basis code with R_{cb} = 5/6 and parameters $n_s = 864$ and $k_s = 720$. This must be shortened by $f = 288$ bits to get the desired (576, 432) code. Since such a basis code is part of the standard (such code parameters exist in the first composite column), it is possible to replace the original code with the shortened version of the basis code. The second code of rate R_c = 3/4 would have had parameters $n = 672$ and $k = 504$ defined by the value $z = 28$. To replace this code with an equivalent code by shortening a basis R_{cb} = 5/6 code with codeword size $n_s = 1008$ and $k_s = 840$ with value $f = 336$ bits would have to be used. However, such a code doesn't belong to the set of standard-mandated codeword sizes, and therefore the R_c = 3/4 code cannot be replaced. This is indicated by setting the numbers of n_s, k_s and *f* in the second row of the second composite column in Tab. 2 to zeros.

Given a target code with code rate smaller than the base rate R_{cb}, an appropriate basis code with R_{cb} = 5/6 can be found by a simple algorithm referred to as Algorithm A2.

4.1 Algorithm A2:

Compute the values *n, k*, $(n-k)$ based on basic matrix size dimension given in Tab. 1 and expansion factor *z*. Find compatible basis code parameters in the first composite column by comparing the $(n-k)$ column. If such a code exists in the first composite column of Tab. 2, use its parameters n_s and k_s. Compute the number of filler bits *f* by subtracting *k* from k_s. If such a code doesn't exist, indicate this by setting code parameters to zero.

Tab. 2 provides the results of similar computations for all the standard-defined codes. Out of the 76 combinations of code rate a codeword length, 35 are implemented using only one of the 6 codes specified – the basis code with R_c = 5/6. That means 46% of the standard required code parameters are covered while saving design complexity. This seems like a reasonable tradeoff to be considered in future standards.

		basis R_c = 5/6			**target R_c = 3/4 (A and B)**			**target R_c = 2/3 (A and B)**			**target R_c = 1/2**		
	z	***n***	***k***	***n-k***	n_s	k_s	***f***	n_s	k_s	***f***	n_s	k_s	***f***
1.	24	576	480	96	864	720	288	1152	960	576	1728	1440	1152
2.	28	672	560	112	0	0	0	1344	1120	672	2016	1680	1344
3.	32	768	640	128	1152	960	384	1536	1280	768	2304	1920	1536
4.	36	864	720	144	0	0	0	1728	1440	864	0	0	0
5.	40	960	800	160	1440	1200	480	1920	1600	960	0	0	0
6.	44	1056	880	176	0	0	0	2112	1760	1056	0	0	0
7.	48	1152	960	192	1728	1440	576	2304	1920	1152	0	0	0
8.	52	1248	1040	208	0	0	0	0	0	0	0	0	0
9.	56	1344	1120	224	2016	1680	672	0	0	0	0	0	0
10.	60	1440	1200	240	0	0	0	0	0	0	0	0	0
11.	64	1536	1280	256	2304	1920	768	0	0	0	0	0	0
12.	68	1632	1360	272	0	0	0	0	0	0	0	0	0
13.	72	1728	1440	288	0	0	0	0	0	0	0	0	0
14.	76	1824	1520	304	0	0	0	0	0	0	0	0	0
15.	80	1920	1600	320	0	0	0	0	0	0	0	0	0
16.	84	2016	1680	336	0	0	0	0	0	0	0	0	0
17.	88	2112	1760	352	0	0	0	0	0	0	0	0	0
18.	92	2208	1840	368	0	0	0	0	0	0	0	0	0
19.	96	2304	1920	384	0	0	0	0	0	0	0	0	0

Tab. 2. Potential for implementation of the standard defined codes with different code rates with a single code and proposed shortening scheme. Zero values indicate that this code parameters cannot be implemented by shortening of the basic R_{cb} = 5/6 code described in Sec. 4. Nonzero values provide the values of parameters of basic R_{cb} = 5/6 code that must be used to obtain an equivalent-parameters code, including the number of bits to shorten *f*. All desired code parameters are defined by the expansion factor *z*.

5. Simulation Results

As already mentioned in the previous sections, the infinite confidence of the filler nodes can lead to a wrong expectation of improved error performance. This was already demonstrated to be wrong for Min-Sum decoding algorithm and further analyzed from the code structure perspective in [2], [3]. Effects of shortening on different decoding schemes have to be analyzed separately. This analysis is to be performed in our future work. Despite this claim, there is indeed some shift in the waterfall curve present if the simulation is not designed carefully. This stems from a very simple fact that shortening a code by *f* data bits, while keeping the number of parity bits intact, the code rate of the new shortened code is lower than the code rate of the original code. This was already explicitly stated in equation (1). Therefore, in the following simulations, codes with same parameters are compared – one original code and one shortened code with the same parameter after shortening. Further simulation settings are classical: we compare the performance of a BPSK system in AWGN channel under the same Min-Sum decoding algorithm with 5 decoder iterations without any special decoder optimization, such as layered decoding.

Figure 6 provides comparison between $R_c = 3/4$A code, with an $R_c = 5/6$ code, shortened so that the parameters of each codes are ($n = 576$, $k = 432$) and Figure 7 compares a smaller rate $R_c = 2/3$B code with an equivalent shortened code again with final parameters ($n = 576$, $k = 432$). Note that because of the lower effective code rate, the waterfall curve in Fig. 7 is slightly shifted to the left, which is in accordance with coding theory. After 5 Min-Sum decoder iterations, it is evident, that the waterfall curves of these codes overlap. No special selection of

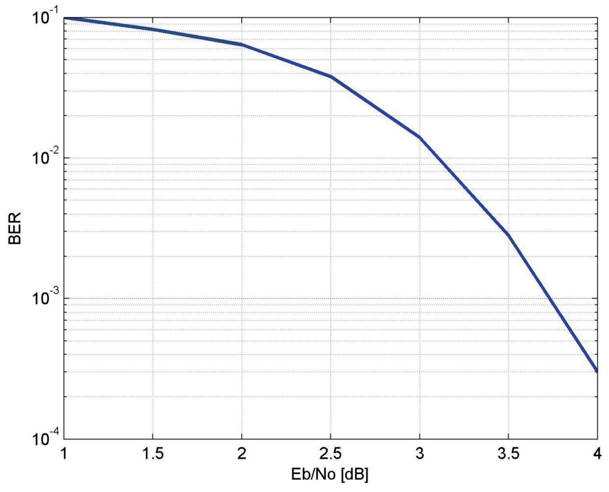

Fig. 6. Comparison of $R_c = 3/4$A LDPC code with equivalent shortened $R_c = 5/6$ LDPC code – two curves overlap.

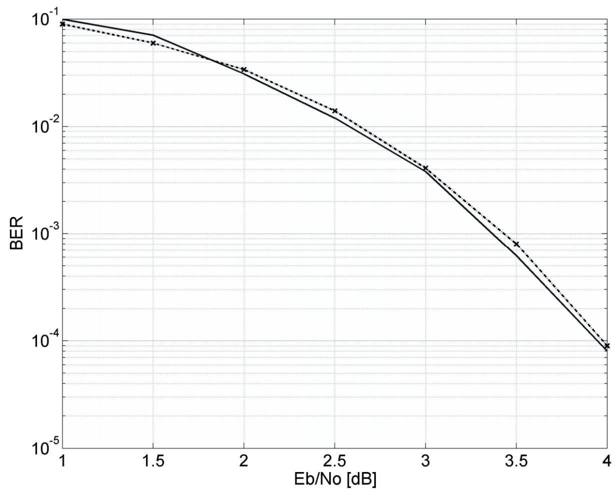

Fig. 7. Comparison of $R_c = 2/3$B LDPC (continuous line) code with equivalent $R_c = 5/6$ LDPC code (dashed) shortened to the same coderate – two curves almost overlap.

meaningless bit positions to be filled with zeros was implemented – the zero fill is one continuous block as shown in Fig. 4. Even without any optimization of fill positions, performance of the two codes is the same.

As shown in Fig. 7, the resulting waterfall curves almost overlap again. The minor difference comes from the slightly different code structures and may be considered negligible. This means that the shortened code is just as good as the original one, which is not surprising.

6. Conclusion

In this paper, we have shown how a time proven method of code shortening, used in the area of RS codes, can be successfully applied also in a novel different context of LDPC codes decoding. We expanded the existing effect analysis of shortening on code structure [2], [3] by providing a simple analysis of the effects of this code modification scheme on practical LDPC decoding algorithms, such as Min-Sum. A detailed analysis of potential simplification of LDPC code set currently used in modern communication standards is also provided, with results tabulated. Our analysis is then complemented by simulation results, confirming the theoretically expected effects of the proposed modification on system error performance.

Acknowledgments

This work was supported mainly by the Scientific Grant Agency of the Ministry of Education of the Slovak Republic and the Slovak Academy of Sciences under contract VEGA 1/0518/13; by EU RTD Framework Programme under ICT COST Action IC 1104; by the Slovak Research and Development Agency under contract SK-PT-0014-12 and SK-AT-0020-12; and by Visegrad Fund and National Scientific Council of Taiwan under IVF–NSC, Taiwan Joint Research Projects Program application no. 21280013 "The Smoke in the Chimney – An Intelligent Sensor-based TeleCare Solution for Homes".

References

[1] DEUTSCH, L. J. The effects of Reed-Solomon code shortening on the performance of coded telemetry systems. NASA, 1983. Available at: https://archive.org/details/nasa_techdoc_19840005335.

[2] LIU X., WU, X., ZHAO, C. Shortening for irregular QC-LDPC codes. *IEEE Communication Letters*, Aug. 2009, vol. 13, no. 8, p. 612–614.

[3] XU, Y., LIU, B., GONG, L., RONG, B., GUI, L. Improved shortening algorithm for irregular QC-LDPC codes using known bits. *IEEE Transactions on Consumer Electronics,* August 2011, vol. 57, no. 3, p. 1057–1063. DOI: 10.1109/TCE.2011.6018855

[4] LAN/MAN Standards Committee, IEEE Std 802.15.4^{TM}-2011: Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). Sep. 2011, New York, USA. ISBN: 978-0-7381-6683-4.

[5] LAN/MAN Standards Committee, IEEE Std 802.16™-2012: IEEE Standard for Air Interface for Broadband Wireless Access Systems. Aug. 2012, New York, USA. ISBN: 978-0-7381-7291-0.

[6] LAN/MAN Standards Committee, IEEE Std 802.11™-2012: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. March 2012, New York, USA, ISBN: 978-0-7381-7245-3.

[7] HA, J., KIM, J., KLINC, D., MC LAUGHLIN, S. W. Rate-compatible punctured low-density parity-check codes with short block lengths. *IEEE Transactions on Information Theory*, Feb. 2006, vol. 52, no. 2, p. 728–738. DOI: 10.1109/TIT.2005.862118

[8] KWON, J., KLINC, D, HA, J., MC LAUGHLIN, S. W. Fast decoding of rate-compatible punctured LDPC codes. In *Proceedings of IEEE International Symposium on Information Theory ISIT 2007.* Nice (France), June 2007, p. 216–220. DOI: 10.1109/ISIT.2007.4557229

[9] MOON, T. K. *Error Correction Coding - Mathematical Methods and Algorithms*. New Jersey: Wiley, 2005. ISBN 978-0471648000.

[10] HUANG, X. Single-scan Min-Sum algorithms for fast decoding of LDPC codes. In *IEEE Information Theory Workshop ITW '06*. Chengdu (China), October 2006, p. 140–143. DOI: 10.1109/ITW2.2006.323774

[11] HALFHILL, T. R. Parallel processing with CUDA. In *Microprocessor Report*. Arizona, USA, January 2008. Available at: http://www.nvidia.com/docs/IO/55972/220401_Reprint.pdf.

About the Authors...

Tomáš PÁLENÍK (Ing., Ph.D.) was born in 1980. He received his Ing. degree in Telecommunications in 2006 from the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava (FEI STU), Slovakia. In 2010 he received his Ph.D. degree in Telecommunications also from FEI STU. Currently, he is an assistant at the Dept. of Telecommunications, FEI STU in Bratislava, Slovakia. He is a Member of the IEEE. His research interests include digital communication systems simulations, Orthogonal Frequency Division Multiplexing, Software Defined Radio, Error-Control Coding and design and implementation of massively parallel algorithms for programmable GPUs.

Peter FARKAŠ (M'93) received the Ing. (MSc.) degree in Telecommunications from STU in 1980, Ph.D. degree from St. Petersburg State Polytechnic University in 1987 and DrSc. degree from STU in 1997. He is with the Inst. of Telecommunications at STU and with the Inst. of Applied Informatics, Faculty of Informatics, Pan European University in Bratislava as a Professor. He has visited and worked at the Dept. of Informatics IV RWTH Aachen, Computing Center of RWTH Aachen, Dept. of Communications and Radio-Frequency Engineering, Vienna University of Technology, Dept. of Mathematics, Statistics and Operational Research of the Nottingham Trent University, Dept. of Communication Systems in Lancaster University, Cooperative Research Center for Broadband Telecommunications and Networking at Dept. of Computer and Communication Engineering, Edith Cowan University, Australian Telecommunications Research Institute at Curtin University of Technology. From 2002 till 2007 he was a Visiting Professor at Kingston University, UK and a senior researcher at SIEMENS PSE. In 2003 SIEMENS named him VIP for his innovations and patents. In 2004 he was awarded with 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 for SANDBRIDGE Tech. He was responsible leader of a team from STU in project NEXWAY IST - 2001-37944 (Network of Excellence in Wireless Applications and Technology) and also project leader on behalf of STU in CRUISE (Creating Ubiquitous Intelligent Sensing Environments) FP6 IST-2005- 4- 027738, (2006-2007). (Projects funded by the European Community under the 5FP and 6FP "Information Society Technologies" Programs.) His research interests include Error Control Coding, Communications Theory and sequences for CDMA. He published 3 chapters in books, 44 papers in reviewed scientific journals, 92 papers in international conferences and is an author or coauthor of 7 patents. Prevailing part of these publications is dealing with Coding theory, coding constructions and their decoding. He is and was serving in TPC of about 60 international conferences and presented 11 invited lectures.

Prof. Farkaš organized IEEE R8 Conference EUROCON 2001 and was chairman of SympoTIC'03, SympoTIC'04, SympoTIC'06 and co-organizer of Winter School on Coding and Information Theory 2005. From 1992 till 2014 he was serving in IEEE Czechoslovakia Section Executive Committee in different positions, now as a chair and from 2005 to 2006 he served as a chair of Conference Coordinator Subcommittee in Region 8 IEEE.

Martin RAKÚS (Ing., Ph.D.) graduated in Radio Electronics from the Slovak University of Technology in 2001. Since 1995 he is with the Dept. of Telecommunications, currently as an assistant professor. In 2004 he received Ph.D. degree from the Slovak University of Technology. He is a Member of the IEEE. His primary research interests are Error Control Coding and communication systems.

Ján DOBOŠ (Ing.) received the master's degree in Telecommunications from the Slovak University of Technology in Bratislava, Faculty of Electrical Engineering and Information Technology in 2013. He is currently working as an assistant in the field of information security and programming at the Faculty of Informatics of Pan-European University, Slovakia and pursuing his Ph.D. degree in the Dept. of Telecommunications, Slovak University of Technology in Bratislava. His current research interests include construction of error-control codes over finite fields, web page development, information security, and data mining.