A Modified Shuffling Method to Reduce Decoding Complexity of QC-LDPC Codes

Alireza Hasani (alireza.hasani254@gmail.com)
Leibniz-Institut fur innovative Mikroelektronik https://orcid.org/0000-0002-0946-0324
Lukasz Lopacinski
Leibniz-Institut fur innovative Mikroelektronik
Rolf Kraemer
Leibniz-Institut fur innovative Mikroelektronik

Research

Keywords: Quasi-cyclic low-density parity-check code, Layered decoding, Decoding complexity

Posted Date: September 8th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-69956/v1

RESEARCH

A Modified Shuffling Method to Reduce Decoding Complexity of QC-LDPC Codes

Alireza Hasani[1,2*], Lukasz Lopacinski[1] and Rolf Kraemer[1,2]

*Correspondence: hasani@ihp-microelectronics.com
[1] IHP - Leibniz-Institut für innovative Mikroelektronik, Frankfurt (Oder), Germany
Full list of author information is available at the end of the article

Abstract

Layered decoding (LD) of Low-Density Parity-Check (LDPC) codes is a decoding schedule that facilitates partially parallel architectures for performing Belief Propagation (BP)-based iterative algorithms. It has reduced implementation complexity and memory overhead compared to fully parallel architectures and higher convergence speed compared to both serial and parallel architectures. In this paper, we introduce a modified form of shuffling of the Parity-Check Matrices (PCMs) of Quasi-Cyclic LDPC (QC-LDPC) codes, which is basically an interleaving operation of the rows of the PCM. The modified shuffling method just like the conventional shuffling method results in a PCM in which each layer can be produced by the circulation of its above layer one symbol to the right. However, it additionally guarantees the weights of the columns in each layer to be either zero or one. Then, we show that due to these two properties, the number of occupied Look-Up Tables (LUTs) on a Field Programmable Gate Array (FPGA) is reduced by about 93% and consumed on-chip power by nearly 80%. Nevertheless, shuffling doesn't degrade Bit Error Rate (BER) performance compared with the non-shuffled case. Additionally, decoding throughput is not sacrificed for low SNR values and its degradation is negligible until the BER of 1e-6.

Keywords: Quasi-cyclic low-density parity-check code; Layered decoding; Decoding complexity

1 Introduction

With the advent of next-generation wireless networks the spontaneous adoption of many of the conventional transmission techniques and protocols may no longer be viable, since these networks have more stringent constraints in terms of throughput,

latency, reliability and energy efficiency. Forward-Error Correction (FEC) methods are one of the vital elements of such networks which are used to provide the demanded level of reliability in the transmission of information. Nevertheless, powerful FEC techniques like LDPC, Turbo or polar codes, known as capacity-achieving codes, bring about higher complexity and higher power consumption compared with traditional coding techniques. Among these codes, LDPC codes have been so far incorporated into several previous technologies and they are still seen as the potential candidate for the new standards like Fifth Generation (5G) and IEEE 802.11be.

The distinctive characteristics of LDPC codes such as their low-density PCM and Tanner graph lacking short cycles have facilitated the use of BP as the main decoding method for LDPC codes. As a matter of fact, their promising error-correcting performance occurs only if they are decoded with a BP algorithm. However, an iterative decoding algorithm based on BP is inherently costly and complex, and its use is not straightforward when next-generation wireless communication systems with tight constraints on throughput, latency and energy efficiency are in scope. Hence, trying to introduce BP-based decoding algorithms with reduced complexity has been a focal point of research.

In nutshell, BP algorithm is an attempt to obtain a more precise estimation of the primary estimates of the codeword bits which are exhibited by the soft-decision sequence received at the output of the demodulator. Toward this goal and dealing with PCM from the viewpoint of its Tanner graph, the BP algorithm employs a set of messages representing probabilities that a given symbol in a received codeword is either a one or zero [1]. These messages are successively passed between the nodes in the Tanner graph until they produce a sequence satisfying parity-check equations.

Over the years, many researchers have been trying to modify BP algorithm in different aspects. Numerous works address, for instance, the issue of computation of reliability messages [2–13]. Several other works investigate scheduling of the algorithm [14–16] which determines in what order the reliability messages should be exchanged between the nodes in the Tanner graph. Decoding schedule is generally associated with the implementation architecture of the decoding method. Flood schedule [17] for example facilitates a fully-parallel architecture in which all the Variable Nodes (VNs) and Check Nodes (CNs) in the Tanner graph pass messages concurrently to their neighbors in every iterations of the algorithm. Although hav-

ing high throughput this schedule demands as many functional units in hardware as the number of CNs and VNs, thus requiring a large silicon area with high interconnect complexity [18]. In a serial architecture, in contrast, a smaller number of functional units are re-used several times to perform each decoder iteration. In this way decoding complexity is lowered, although at the price of reduced decoding throughput.

Partially parallel decoding architecture is a good trade-off between hardware complexity and decoding throughput and it is best accomplished by Layered Decoding (LD) schedule [19–37]. In this schedule the rows of PCM are divided into several layers, and each iteration of the BP algorithm is likewise split into several sub-iterations, each running over one layer of the PCM. During each sub-iteration, reliability messages are exchanged between CNs of that layer and their neighbor VNs, and, ultimately, the updated reliability messages are handed to the next layer. Accordingly, in each sub-iteration only a subset of CNs, i.e. as many as the number of rows in each layer, participate in the decoding process. In this way, corresponding functional units in hardware can be re-used for the CNs of different layers and layers are processed successively from top to down the PCM. This causes a reduced hardware utilization of LD compared to flood schedule. Furthermore, LD schedule achieves better convergence performance than the flooding schedule due to the fact that the latest VTC messages are always used to update the CTV messages during a sub-iteration.

Since the introduction of LDPC codes many different ways for their construction have been proposed. Among them some generate a structured type of LDPC codes known as QC-LDPC codes which possess a cyclic property. By the leverage of this cyclic property their encoding and decoding process can be considerably simplified, while presenting comparable performance to random (or unstructured) LDPC codes [38, 39]. For the sake of decoding complexity it is highly desirable the number of ones in each column of each layer, i.e. the weight of the columns of each layer, be either one or zero when LD is adopted as the decoding schedule. QC-LDPC codes have inherently such layered structure with this property on the weight of columns and they are usually adopted as the FEC code when LD is the decoding method.

In an attempt to exploit the cyclic property of a QC-LDPC code to simplify its decoding, the authors in [27] introduced a novel shuffling method, which shuffles the

rows of the PCM of a QC-LDPC code prior to decoding and gives it a new layered format. After applying this shuffling each layer can be produced by circulating its above layer one symbol to the right. The authors then show this cyclic property can be exploited in order to simplify LD and speed up convergence rate. In particular, due to the cyclic property it is enough to realize only the first layer of PCM in hardware rather than the entire of that.

The downside of this shuffling idea is that it probably spoils the primary single-weight columns property of the PCM, meaning that it is unlikely for the shuffled PCM to maintain that property. In such cases, LD will no longer be a recommended decoding schedule and the conventional BP algorithm is the preferred choice.

We outlined a modified shuffling idea in our previous work [40] which was an extension to the primary shuffling idea. The modified shuffling method results in a shuffled PCM that has both the desired single-weight columns feature and also the cyclic property. This is in fact accomplished by introducing a set of offset values prior to performing the shuffling.

In this paper, the modified shuffling method of [40] is further investigated, the logic behind it is clearly expressed and its benefits and advantages are extensively highlighted. To be specific, [40] lacked any implementation results to verify the improvements promised by the modified shuffling method. Here, the results of implementing LD of several QC-LDPC codes when shuffled with the proposed technique is provided, and, for the sake of comparison, they are accompanied with the results associated with LD of non-shuffled QC-LDPC codes. The results in the former case reveal improvements in terms of number of occupied LUTs on FPGA and also power consumption. The target FPGA is a Xilinx-Virtex 7 and the tested codes are four QC-LDPC codes from IEEE 802.15.3c and one from IEEE 802.16e standards. It is shown that these improvements are achieved without sacrificing BER performance and decoding throughput for low values of E_b/N_0. Although provided simulation results and analysis indicate that with shuffling the decoder needs more sub-iterations to achieve a specific performance, we argue that this is not always translated into a lower decoding throughput. Implementation results reveal that the clock frequency with the shuffled PCM can be increased compared to the non-shuffled case, due to the simplifications made by the shuffling method in the decoding process. Therefore, slower convergence rate can be compensated to an extent by having faster clock.

This compensation is full for low values of E_b/N_0 and throughput degradation is negligible until the BER of 1e-6.

The organization of the paper is as follows. Section 2 presents preliminaries, including the fundamentals of QC-LDPC codes and their LD. Section 3 is devoted to the assessment of the novel shuffling method and its attributes. Implementation and simulation results along with necessary analysis come in section 4. Final conclusions are made in section 5.

2 Preliminaries

Before delving into the main proposal of the paper, a short introduction on QC-LDPC codes and LD schedule is necessary.

2.1 QC-LDPC Codes

A linear block code C is called an LDPC code if it is the null space of a PCM, denoted by $\boldsymbol{H}$, with following characteristics [41]: I- The number of ones in each row and column is small compared with n (the length of the code) and J (the number of rows in $\boldsymbol{H}$); II- the number of ones in common between any two columns (or rows) is not greater than one. The latter condition is usually referred to as the Row-Column (RC) constraint in the literature [42].

The PCM of an LDPC code is usually visualized by means of a graph known as the Tanner graph. The Tanner graph of a PCM consists of two sets of nodes. One set represents CNs, i.e. the parity-check sums (or equations), which are, in fact, the rows of $\boldsymbol{H}$, and the other set stands for VNs that are equivalent to columns of $\boldsymbol{H}$. A CN in the Tanner graph is connected to a VN if and only if the corresponding element of $\boldsymbol{H}$ is one.

When the PCM of an LDPC code is comprised of Circulant Permutation Matrices (CPMs) and zero matrices, the resultant code will be Quasi Cyclic (QC) as well. A circulant is a matrix in which each row is a cyclically rightward-shifted copy of its above row. If a circulant is a permutation as well, i.e. every rows and columns are single-weight, then the circulant is called a CPM. The PCM of a QC-LDPC code

could be represented as

$$\boldsymbol{H}_{qc} = \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \vdots \\ \boldsymbol{A}_c \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{1,1} & \boldsymbol{A}_{1,2} & \dots & \boldsymbol{A}_{1,t} \\ \boldsymbol{A}_{2,1} & \boldsymbol{A}_{2,2} & \dots & \boldsymbol{A}_{2,t} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{c,1} & \boldsymbol{A}_{c,2} & \dots & \boldsymbol{A}_{c,t} \end{bmatrix}, \quad (1)$$

in which c and t are two positive integers provided that $c \leq t$ and $\boldsymbol{A}_{i,j}$s are either $b \times b$ CPMs or $b \times b$ zero matrices. A codeword $\boldsymbol{v}$ of length $t.b$ of this code can be considered as a t-section sequence $\boldsymbol{v} = (\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots, \boldsymbol{v_t})$ in which each section $\boldsymbol{v_i}, 1 \leq i \leq t$ is of length b. In the resultant code, codewords have sectionized cyclic structure. That means, with cyclic shifting of all the t sections in a codeword individually, another valid codeword is generated [42].

Another way for representing the PCM of a QC-LDPC code which is more compact than (1) is known as the base matrix, denoted by $\boldsymbol{\mathcal{W}}$. Note that a CPM is in fact an identity matrix whose rows have been circularly shifted some samples to the left or right. $\boldsymbol{\mathcal{W}}$ is a matrix whose non-negative integer entries specify the shifting value in the corresponding CPM with respect to an identity matrix. The other entries, usually chosen to be -1, represent zero matrices in PCM. Transformation of a base matrix to its corresponding PCM is called dispersion. Fig. 1 shows the base matrices for the QC-LDPC codes utilized in IEEE 802.15.3c standard and Fig. 2 shows the 1/2-rate (2304,1152)-QC-LDPC code used in IEEE 802.16e. In these two figures, empty places are the locations of zero matrices.

2.2 LD of QC-LDPC Codes

Due to the RC constraint, the Tanner graph representation of the PCM of an LDPC code lacks short cycles of length 4. Therefore, the decoding method for them could be formulated as the computation of marginal probabilities on a factor graph [43]. An efficient way to solve these problems is BP algorithm which is exact when the factor graph is free of cycles, but approximate when it has cycles [43]. However, BP-based decoding algorithms have become the major decoding method for LDPC codes, although the Tanner graph of an LDPC code may have a few short cycles. In the BP algorithm the reliability messages are successively passed between VNs

and CNs in the Tanner graph till a codeword that has the maximum probability according to the received soft-decision sequence is found.

LD, as stated before, is a schedule which relies on the assumed layered structure of the PCM and is viewed as a way to realize a partially-parallel architecture for execution of the BP algorithm. In LD schedule, each iteration is split into several sub-iterations, running over successive layers of the PCM. During each sub-iteration, reliability messages are exchanged between CNs of that layer and their neighbor VNs, and at the end, the updated reliability messages are handed down to the next layer. Accordingly, only a subset of CNs and VNs participate in each sub-iteration, and layers are processed successively from top to down the PCM.

Let $\boldsymbol{y} = (y_0, \ldots, y_{n-1})$ be the soft-decision sequence at the output of the channel that is to be decoded. Assuming that the PCM has been divided into L layers each containing E consecutive rows of the PCM, successive layers of $\boldsymbol{H}_{qc}$ are traversed by the decoding algorithm in order from top to down. Specifically, the i-th layer $\boldsymbol{H}_{qc}^{(i)}, 1 \leq i \leq L$ contains rows $(i-1)E+1, \ldots, i.E$ of $\boldsymbol{H}_{qc}$. For $\boldsymbol{h}_j^{(i)}, 1 \leq j \leq E, 1 \leq i \leq L$, the j-th CN of $\boldsymbol{H}_{qc}^{(i)}$, its support is defined as all its neighbor VNs, i.e.

$$\mathcal{B}(\boldsymbol{h}_j^{(i)}) = \{l : h_{j,l}^{(i)} = 1, 0 \leq l < n\}. \tag{2}$$

Analogously, the support of l-th VN within the i-th layer is defined as all the CNs inside that layer connected to that VN. Mathematically,

$$\mathcal{A}_l^{(i)} = \{j : h_{j,l}^{(i)} = 1, 1 \leq j \leq E\}. \tag{3}$$

At the beginning of the algorithm, A Posteriori Probability (APP) values of VNs and Check-to-Variable (CTV) messages are initialized. APP values $Y_l, 0 \leq l < n$ are initialized with elements of $\boldsymbol{y}$ that serve as the a-priori probabilities. CTV messages corresponding to the i-th layer are denoted by $L_{j,l}^{(i)}, 1 \leq j \leq E, 0 \leq l < n, 1 \leq i \leq L$, and they are initially set to zero for all the layers. Once initialized, the iterative part of the algorithm begins and CN and VN operations for the layers are performed sequentially. The i-th sub-iteration that is associated with the i-th layer is further split into three steps:

1 Vertical step: Variable-to-Check (VTC) messages are updated as

$$Z_{j,l}^{(i)} = Y_l - L_{j,l}^{(i)}, 1 \leq j \leq E, 0 \leq l < n. \tag{4}$$

2 Horizontal step: CTV messages are updated as

$$L_{j,l}^{(i)} = \prod_{l' \in \mathcal{B}(\boldsymbol{h}_j^{(i)}) \backslash l} sgn(Z_{j,l'}^{(i)}) \times \min_{l' \in \mathcal{B}(\boldsymbol{h}_j^{(i)}) \backslash l} |Z_{j,l'}^{(i)}|. \quad (5)$$

Here, without loss of generality, min-sum scheme [5] has been used for computing CTV messages and $sgn(x)$ is a sign function which is equal to 1 when $x \geq 0$ and -1 otherwise. It should however be noted that other proposed schemes like the ones in [4, 7, 44–48] could rather be used for computation of the CTV messages.

3 Hard decision and stopping criterion test: APP values for all the VNs are computed as:

$$Y_l = y_l + \sum_{j \in \mathcal{A}_l^{(i)}} L_{j,l}^{(i)}, 0 \leq l < n. \quad (6)$$

At the end of each sub-iteration, the codeword bits are estimated according to the updated values of $Y_l, 0 \leq l < n$. In particular, $\hat{v}_l = 1$ if $Y_l > 0$ and $\hat{v}_l = 0$ otherwise. If the estimated codeword $\hat{\boldsymbol{v}} = (\hat{v}_0, \ldots, \hat{v}_{n-1})$ satisfies the check-sum condition $\boldsymbol{H}_{qc}\hat{\boldsymbol{v}}^T = 0$ then it will be accepted as the valid codeword and the algorithm terminates. Otherwise, it proceeds to the next sub-iteration and algorithm returns to the vertical step of the next layer. A threshold value I_{max} is often set to indicate the maximum number of iterations. If no valid codeword is found within I_{max} iterations, the algorithm halts and declares a failure.

The benefit of the condition of having only single-weight columns in each layer can now be clarified. By this condition realization of both (4) and (5) are simplified and the need for summation in (6) is also eliminated. It is worth emphasizing again that in the case of QC-LDPC codes, if each block-row in (1) is considered as a layer, then obviously all the columns in each layer have only one "1"-entry.

3 Modified Shuffling of QC-LDPC Codes

In this section, we propose a modified shuffling idea for LD of QC-LDPC codes. Shuffling is the act of swapping the rows of the PCM, in a manner that the complexity of the decoding algorithm reduces, while the error correction performance is preserved. The primary idea of shuffling method has already been proposed in [27]. For $i = 1, \ldots, b$, let $\boldsymbol{H}_{qc}^{sh,(i)}$ be a $c \times n$ matrix containing the rows

$i, i+b, i+2b, \ldots, i+(c-1)b$ of $\boldsymbol{H}_{qc}$. In this way, the $J = c.b$ rows of $\boldsymbol{H}_{qc}$ are partitioned between matrices $\boldsymbol{H}_{qc}^{sh,(i)}$, $i = 1, \ldots, b$. The shuffled PCM, $\boldsymbol{H}_{qc}^{sh}$, is then formed by having matrices $\boldsymbol{H}_{qc}^{sh,(i)}$, $i = 1, \ldots, b$ as its row-blocks:

$$\boldsymbol{H}_{qc}^{sh} = \begin{bmatrix} \boldsymbol{H}_{qc}^{sh,(1)} \\ \vdots \\ \boldsymbol{H}_{qc}^{sh,(b)} \end{bmatrix} \tag{7}$$

An example of such a shuffling is depicted in Fig. 3, which has been performed on a $(12, 4)$-QC-LDPC code. Now, in $\boldsymbol{H}_{qc}^{sh}$ each layer $\boldsymbol{H}_{qc}^{sh,(i)}$ is obtained by cyclically shifting its above layer $\boldsymbol{H}_{qc}^{sh,(i-1)}$ one symbol to the right, given that the circulation operation is performed on the t individual sections of a layer separately. For instance, in Fig. 3-(b), $\boldsymbol{H}_{qc}^{sh,(2)}$ can be derived from $\boldsymbol{H}_{qc}^{sh,(1)}$ by rightward cyclic shifting of the $t = 4$ sections of each row of that. Repeating this circulation one more time yields $\boldsymbol{H}_{qc}^{sh,(3)}$.

Due to this cyclic property provided by shuffling, it is no longer needed to implement the whole PCM in the target hardware (for example an FPGA) and define all the "1"-entries as connections in it. Instead, it suffices to implement only $\boldsymbol{H}_{qc}^{sh,(1)}$, the first layer of $\boldsymbol{H}_{qc}^{sh}$, and then perform a circulation on the memory blocks containing updated APP values at the end of each sub-iteration. This implementation benefit is further highlighted at the end of this section.

The shortcoming with this shuffling method is that in the shuffled matrix, layers may no longer be comprised of single-weight columns, but also columns of bigger weights. As a matter of fact, if an integer in the base matrix of a QC-LDPC code appears m times in a column, then, when the base matrix is dispersed and shuffled, in the corresponding block-column, columns of weight m emerge. For instance, in our studied QC-LDPC codes in Fig. 1 and 2, the shaded columns are the ones which have repetitive numbers. Consequently, after dispersion and shuffling, they bring about columns of weight 2 or 3 in each layer of their own shuffled matrix.

As a solution to this weakness of shuffling, we propose to employ a set of offset values in order to modify the order of the rows of $\boldsymbol{H}_{qc}$ which are put in the same layer in $\boldsymbol{H}_{qc}^{sh}$. Suppose that $o_1, \ldots, o_c$ are some integer values such that $0 \leq o_m < b$, $1 \leq m \leq c$. According to the new shuffling method, the i-th layer of $\boldsymbol{H}_{qc}^{sh}$ is made up of rows $\{i+o_1, b+i+o_2, \ldots, (c-1)b+i+o_c\}$ of $\boldsymbol{H}_{qc}$ for $1 \leq i \leq b$. o_is serve as

the offset values, and they are carefully selected according to the $\boldsymbol{H}_{qc}$ which is to be shuffled, such that in $\boldsymbol{H}_{qc}^{sh}$ no column in a layer has the weight of greater than one.

The idea of employing offset values in shuffling can be viewed from another perspective. As stated, the shortcoming of the basic shuffling method stems from the presence of repetitive integers in a column of the base matrix. The offset values can be regarded as a means to eliminate repetitive values in a column. In other words, the modified shuffling technique with offset values is equivalent to the basic shuffling method if performed on $\boldsymbol{H}_{qc}$ whose base matrix no longer contains repetitive integers in a column because of the offset values. A possible offset set for each of our example codes is presented in table 1 and the resulting modified base matrices are shown in Fig. 4 for the IEEE 802.15.3c codes and Fig. 5 for the IEEE 802.16e code. With the introduced offset values in these examples the shaded columns no longer have repetitive integers. This perspective of the modified shuffling gives us the way for finding appropriate offset values for a specific QC-LDPC code. The straight way is to try all the possible values until one is found that it gives a base matrix in which all the integers in any column are distinct. In general, for a QC-LDPC code there are b^c possible offset sets. However, as soon as the first effective set is found the search can be halted. It should be noted that in some cases there may not exist a possible offset set, like the $(672, 588)$-QC-LDPC code of Fig. 1-d. This indicates that the modified shuffling is not necessarily applicable to all the QC-LDPC codes. It should also be emphasized that the desired cyclic property of $\boldsymbol{H}_{qc}^{sh}$ does exist in the case of modified shuffling as it does in the basic shuffling method, and hence each layer of $\boldsymbol{H}_{qc}^{sh}$ can still be produced from its previous layer by circular shifting.

The advantage of LD with a shuffled PCM over LD with a non-shuffled one from the implementation point of view is also worth noticing. Fig. 6 illustrates the LD architecture for $\boldsymbol{H}_{qc}^{sh}$ of Fig. 3. In this figure, VN Processing Unit (VNPU) and CN Processing Unit (CNPU) respectively stands for a processing unit responsible for computing VTC and CTV messages in (4) and (5) respectively. As illustrated, instead of having $J = 12$ CNPUs, $c = 4$ CNPUs delegating the CNs of the first layer are sufficient for decoding and the connections between them are determined by the 1-entries of the first layer. At the end of each sub-iteration, the computed APP values are cyclically shifted as shown by the figure. As a result of this circula-

tion, the existing connections between VNPUs and CNPUs remain valid and they will represent the connections for the next layer. Therefore, the next sub-iteration can be initiated instantly, without the need to redefine the connections between VNPUs and CNPUs. This is the main advantage delivered by shuffling the PCM of a QC-LDPC code, enabling us to make best use of their inherent cyclic feature.

4 Implementation and Experimental Results

We have implemented both LD with shuffled PCM and LD with non-shuffled PCM of the example codes of IEEE 802.16e and IEEE 802.15.3c standards. The hardware used for implementation was a Xilinx Virtex-7 FPGA. The acquired results have been shown in Table 2. As deduced from the figures in the table, with shuffled PCM, the design is considerably smaller and occupies nearly 93% less numbers of LUTs on FPGA. In terms of the on-chip power reported by the implementation tool, also, LD with shuffled PCM consumes about 80% less power. In summary, the superiority of the shuffling method in terms of hardware area and consumed power is apparent from the implementation results. Note that the design of the non-shuffled IEEE 802.16e is too big to fit in the FPGA, and hence the results are not available. It should also be noted that our main concern is to highlight the improvement introduced by the very shuffling method, but not the other elements of decoding like finding the first two minima.

Fig. 7 shows the performance simulation of the four codes, indicating that shuffling does not degrade performance and LD with (modified) shuffled PCM performs just as good as LD with non-shuffled PCM. This is due to the fact that shuffling interleaves the rows of the PCM and changes the order of them, while the overall PCM's characteristics remain intact. In other words, the determining attributes of the PCM such as distance property, cycles' distribution and rows' and columns' weight do not undergo any change. Nevertheless, the average number of iterations needed to achieve a specific BER performance depicted by Fig. 8 differs in two cases. This stems from the fact that layering is different in two cases. In the case of non-shuffled PCM, the J rows are divided into c layers, each of b rows, while in the case of shuffled PCM, they are divided into b layers, each of c rows. Since, c is usually much more smaller than b, in the first case, a bigger number of VNs are processed in a sub-iteration and hence it needs fewer sub-iterations in total. However, this further

number of sub-iteration to reach a specific BER is compensated to an extent with a bigger clock frequency allowed by the shuffling, as indicated by Table 2. To reach a better insight for that the average throughput is plotted for different codes in Fig. 9. The average throughput can be expressed as

$$\tau = \frac{k.f_{clk}}{N_{ave\_ite}.N_{clk}}. \tag{8}$$

Here, f_{clk} is the clock frequency specified in table 2, N_{ave_ite} is the average number of sub-iterations and N_{clk} is the number of clock cycles each sub-iteration needs. In our implementation, $N_{clk} = 8$ for the case of shuffled PCM and $N_{clk} = 7$ for the other case of non-shuffled PCM, noting that the one extra clock cycle in the former case is needed for cyclic shifting of the computed APP values. Fig. 9 shows that for low values of E_b/N_0 average throughput in the two cases overlap, indicating that the additional number of sub-iterations is compensated fully by the higher clock frequency. Although this is not true as E_b/N_0 grows, it can be claimed that the degradation of the throughput is negligible if BER=1e-6 is chosen as the targeted BER performance.

5 Conclusion

The novel shuffling method proposed in this paper is basically an interleaving the rows of the PCM of a QC-LDPC code with two objectives in mind. First, the columns in each layer of the shuffled PCM must remain single- or zero-weight. Second, each layer must be producible from the upper layer by a one-symbol circular shifting to the right. Though simple, this shuffling brings about considerable complexity reduction in the decoding implementation, while preserving the error-correcting capability of the code and its decoding throughput for BER values up to of 1e-6.

Abbreviations

APP	A Posteriori Probability
BER	Bit Error Rate
BP	Belief Propagation
CN	Check Node
CNPU	CN Processing Unit
CPM	Circulant Permutation Matrix
CTV	Check-to-Variable
FEC	Forward-Error Correction

FPGA	Field Programmable Gate Array
LD	Layered Decoding
LDPC	Low-Density Parity-Check
LUT	Look-Up Table
PCM	Parity-Check Matrix
QC	Quasi Cyclic
QC-LDPC	Quasi-Cyclic LDPC
RC	Row-Column
VN	Variable Node
VNPU	VN Processing Unit
VTC	Variable-to-Check
5G	Fifth Generation

Availability of data and materials

Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

Competing interests

The authors declare that they have no competing interests.

Funding

This work was supported by the German Research Foundation (DFG) project PSSS-FEC, project no. 442607813, application no. LO 2709/1-1.

Author's contributions

AH proposed and developed the new idea of the paper and drafted it. LL and RK have substantially revised it. All authors approved the submitted version.

Acknowledgements

This work was supported by the German Research Foundation (DFG) and conducted at IHP-microelectronics GmbH. The authors are also thankful to the support of Brandenburg University of Technology (BTU) Cottbus-Senftenberg, Prof. Dr. Jörg Nolte and Steffen Büchner.

Author details

[1] IHP - Leibniz-Institut für innovative Mikroelektronik, Frankfurt (Oder), Germany. [2]Department of Electrical and Computer Engineering, Brandenburg university of technology, Cottbus-Senftenberg, Cottbus, Germany.

References

1. Jones, C., Vallés, E., Smith, M., Villasenor, J.: Approximate-min constraint node updating for ldpc code decoding. In: IEEE Military Communications Conference, 2003. MILCOM 2003., vol. 1, pp. 157–162 (2003). IEEE
2. MacKay, D.J.: Good error-correcting codes based on very sparse matrices. IEEE transactions on Information Theory **45**(2), 399–431 (1999)
3. Eleftheriou, E., Mittelholzer, T., Dholakia, A.: Reduced-complexity decoding algorithm for low-density parity-check codes. Electronics letters **37**(2), 102–104 (2001)
4. Hu, X.-Y., Eleftheriou, E., Arnold, D.-M., Dholakia, A.: Efficient implementations of the sum-product algorithm for decoding ldpc codes. In: GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No. 01CH37270), vol. 2, pp. 1036–1036 (2001). IEEE
5. Chen, J., Fossorier, M.P.: Near optimum universal belief propagation based decoding of low-density parity check codes. IEEE Transactions on communications **50**(3), 406–414 (2002)
6. Wang, Z., Jia, Q.-w.: Low complexity, high speed decoder architecture for quasi-cyclic ldpc codes. In: 2005 IEEE International Symposium on Circuits and Systems, pp. 5786–5789 (2005). IEEE
7. Chen, J., Dholakia, A., Eleftheriou, E., Fossorier, M.P., Hu, X.-Y.: Reduced-complexity decoding of ldpc codes. IEEE transactions on communications **53**(8), 1288–1299 (2005)
8. Yazdani, M.R., Hemati, S., Banihashemi, A.H.: Improving belief propagation on graphs with cycles. IEEE Communications Letters **8**(1), 57–59 (2004)

9. Mohsenin, T., Truong, D.N., Baas, B.M.: A low-complexity message-passing algorithm for reduced routing congestion in ldpc decoders. IEEE Transactions on Circuits and Systems I: Regular Papers **57**(5), 1048–1061 (2010)
10. Darabiha, A., Carusone, A.C., Kschischang, F.R.: A bit-serial approximate min-sum ldpc decoder and fpga implementation. In: 2006 IEEE International Symposium on Circuits and Systems, pp. 149–152 (2006). IEEE
11. Angarita, F., Valls, J., Almenar, V., Torres, V.: Reduced-complexity min-sum algorithm for decoding ldpc codes with low error-floor. IEEE Transactions on Circuits and Systems I: Regular Papers **61**(7), 2150–2158 (2014)
12. Declercq, D., Vasic, B., Planjery, S.K., Li, E.: Finite alphabet iterative decoders—part ii: Towards guaranteed error correction of ldpc codes via iterative decoder diversity. IEEE transactions on communications **61**(10), 4046–4057 (2013)
13. Ghanaatian, R., Balatsoukas-Stimming, A., Müller, T.C., Meidlinger, M., Matz, G., Teman, A., Burg, A.: A 588-gb/s ldpc decoder based on finite-alphabet message passing. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **26**(2), 329–340 (2017)
14. Mao, Y., Banihashemi, A.H.: Decoding low-density parity-check codes with probabilistic schedule. In: 2001 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (IEEE Cat. No. 01CH37233), vol. 1, pp. 119–123 (2001). IEEE
15. Chen, Y., Parhi, K.K.: High throughput overlapped message passing for low density parity check codes. In: Proceedings of the 13th ACM Great Lakes Symposium on VLSI, pp. 245–248 (2003)
16. Liao, E., Yeo, E., Nikolic, B.: Low-density parity-check code constructions for hardware implementation. In: 2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577), vol. 5, pp. 2573–2577 (2004). IEEE
17. Kschischang, F.R., Frey, B.J.: Iterative decoding of compound codes by probability propagation in graphical models. IEEE Journal on Selected Areas in Communications **16**(2), 219–230 (1998)
18. Liu, L., Shi, C.-J.R.: Sliced message passing: High throughput overlapped decoding of high-rate low-density parity-check codes. IEEE Transactions on Circuits and Systems I: Regular Papers **55**(11), 3697–3710 (2008)
19. Mansour, M.M., Shanbhag, N.R.: Turbo decoder architectures for low-density parity-check codes. In: Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE, vol. 2, pp. 1383–1388 (2002). IEEE
20. Mansour, M.M., Shanbhag, N.R.: High-throughput ldpc decoders. IEEE transactions on very large scale integration (VLSI) Systems **11**(6), 976–996 (2003)
21. Sankar, H., Narayanan, K.R.: Memory-efficient sum-product decoding of ldpc codes. IEEE transactions on communications **52**(8), 1225–1230 (2004)
22. Hocevar, D.E.: A reduced complexity decoder architecture via layered decoding of ldpc codes. In: IEEE Workshop onSignal Processing Systems, 2004. SIPS 2004., pp. 107–112 (2004). IEEE
23. Mansour, M.M., Shanbhag, N.R.: A 640-mb/s 2048-bit programmable ldpc decoder chip. IEEE Journal of Solid-State Circuits **41**(3), 684–698 (2006)
24. Gunnam, K.K., Choi, G.S., Wang, W., Kim, E., Yeary, M.B.: Decoding of quasi-cyclic ldpc codes using an on-the-fly computation. In: 2006 Fortieth Asilomar Conference on Signals, Systems and Computers, pp. 1192–1199 (2006). IEEE
25. Gunnam, K., Choi, G., Wang, W., Yeary, M.: Multi-rate layered decoder architecture for block ldpc codes of the ieee 802.11 n wireless standard. In: 2007 IEEE International Symposium on Circuits and Systems, pp. 1645–1648 (2007). IEEE
26. Rovini, M., Rossi, F., Ciao, P., L'Insalata, N., Fanucci, L.: Layered decoding of non-layered ldpc codes. In: 9th EUROMICRO Conference on Digital System Design (DSD'06), pp. 537–544 (2006). IEEE
27. Ueng, Y.-L., Cheng, C.-C.: A fast-convergence decoding method and memory-efficient vlsi decoder architecture for irregular ldpc codes in the ieee 802.16 e standards. In: 2007 IEEE 66th Vehicular Technology Conference, pp. 1255–1259 (2007). IEEE
28. Radosavljevic, P., de Baynast, A., Cavallaro, J.R.: Optimized message passing schedules for ldpc decoding. In: Conference Record of the Thirty-Ninth Asilomar Conference onSignals, Systems and Computers, 2005., pp. 591–595 (2005). IEEE
29. Yang, D., Yu, G., Zou, X., Deng, Y., Zhong, J.: The design and verification of a novel ldpc decoder with high-efficiency. In: 2014 International Symposium on Integrated Circuits (ISIC), pp. 256–259 (2014). IEEE

30. de Baynast, A., Radosavljevic, P., Sabharwal, A., Cavallaro, J.R.: On turbo-schedules for ldpc decoding. IEEE Communications Letters (2006)
31. Radosavljevic, P., Karkooti, M., de Baynast, A., Cavallaro, J.R.: Tradeoff analysis and architecture design of high throughput irregular ldpc decoders. IEEE Trans. Circuits Syst. I: Regular Papers (2006)
32. Brack, T., Alles, M., Kienle, F., Wehn, N.: A synthesizable ip core for wimax 802.16 e ldpc code decoding. In: 2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 1–5 (2006). IEEE
33. Gentile, G., Rovini, M., Fanucci, L.: Low-complexity architectures of a decoder for ieee 802.16 e ldpc codes. In: 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007), pp. 369–375 (2007). IEEE
34. Gunnam, K.K., Choi, G.S., Yeary, M.B., Atiquzzaman, M.: Vlsi architectures for layered decoding for irregular ldpc codes of wimax. In: 2007 IEEE International Conference on Communications, pp. 4542–4547 (2007). IEEE
35. Zhang, K., Huang, X., Wang, Z.: High-throughput layered decoder implementation for quasi-cyclic ldpc codes. IEEE Journal on Selected Areas in Communications **27**(6), 985–994 (2009)
36. Goldberger, J., Kfir, H.: Serial schedules for belief-propagation: analysis of convergence time. IEEE Transactions on Information Theory **54**(3), 1316–1319 (2008)
37. Cui, Y., Peng, X., Chen, Z., Zhao, X., Lu, Y., Zhou, D., Goto, S.: Ultra low power qc-ldpc decoder with high parallelism. In: 2011 IEEE International SOC Conference, pp. 142–145 (2011). IEEE
38. Wang, Z., Cui, Z.: Low-complexity high-speed decoder design for quasi-cyclic ldpc codes. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **15**(1), 104–114 (2007)
39. Fossorier, M.P.: Quasicyclic low-density parity-check codes from circulant permutation matrices. IEEE transactions on information theory **50**(8), 1788–1793 (2004)
40. Hasani, A., Lopacinski, L., Büchner, S., Nolte, J., Kraemer, R.: A modified shuffling method to split the critical path delay in layered decoding of qc-ldpc codes. In: 2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 1–6 (2019). IEEE
41. Lin, S., Costello, D.J.: Error Control Coding vol. 2. Prentice hall, ??? (2001)
42. Li, Z., Chen, L., Zeng, L., Lin, S., Fong, W.H.: Efficient encoding of quasi-cyclic low-density parity-check codes. IEEE Transactions on Communications **54**(1), 71–81 (2006)
43. Planjery, S.K., Declercq, D., Danjean, L., Vasic, B.: Finite alphabet iterative decoders—part i: Decoding beyond belief propagation on the binary symmetric channel. IEEE Transactions on Communications **61**(10), 4033–4045 (2013)
44. Mohsenin, T., Baas, B.M.: Split-row: A reduced complexity, high throughput ldpc decoder architecture. In: 2006 International Conference on Computer Design, pp. 320–325 (2006). IEEE
45. Mohsenin, T., Baas, B.M.: High-throughput ldpc decoders using a multiple split-row method. In: 2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07, vol. 2, p. 13 (2007). IEEE
46. Fossorier, M.P., Mihaljevic, M., Imai, H.: Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. IEEE Transactions on communications **47**(5), 673–680 (1999)
47. El Alami, R., Qjidaa, H., Mehdaoui, Y., Mrabti, M.: A thresholding algorithm for improved split-row decoding method of irregular ldpc codes. In: 2015 Intelligent Systems and Computer Vision (ISCV), pp. 1–5 (2015). IEEE
48. Mohsenin, T., Truong, D., Baas, B.: An improved split-row threshold decoding algorithm for ldpc codes. In: 2009 IEEE International Conference on Communications, pp. 1–5 (2009). IEEE
49. Yen, S.-W., Hung, S.-Y., Chen, C.-L., Chang, H.-C., Jou, S.-J., Lee, C.-Y.: A 5.79-gb/s energy-efficient multirate ldpc codec chip for ieee 802.15. 3c applications. IEEE journal of solid-state circuits **47**(9), 2246–2257 (2012)

Figures

Tables

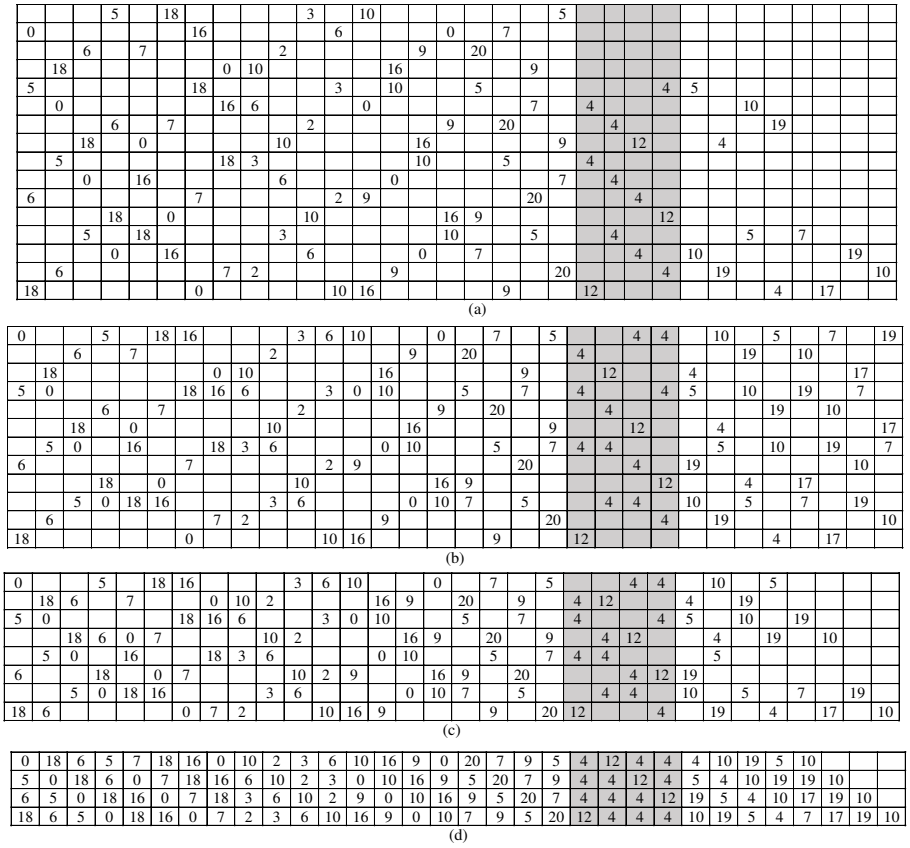


Figure 1: Base matrices of IEEE 802.15.3c QC-LDPC codes [49]: (a) (672,336)-LDPC code; (b) (672-425)-LDPC code; (c) (672,504)-LDPC code; (d) (672,588)-LDPC code.

Table 1: Possible offset values for the studied QC-LDPC codes

Code	Offset set
IEEE 802.16e (2304,1152)	$\{0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1\}$
IEEE 802.15.3c (672,336)	$\{0, 0, 0, 0, 9, 1, 2, 0, 0, 1, 2, 0, 0, 0, 0, 0\}$
IEEE 802.15.3c (672,425)	$\{14, 2, 0, 0, 3, 0, 17, 3, 0, 0, 19, 0\}$
IEEE 802.15.3c (672,504)	$\{0, 0, 1, 0, 10, 3, 19, 4\}$

	94	73						55	83			7	0										
	27				22	79	9				12		0	0									
			24	22	81		33				0			0	0								
61		47						65	25						0	0							
		39				84			41	72						0	0						
				46	40		82				79	0					0	0					
		95	53						14	18								0	0				
	11	73				2			47										0	0			
12				83	24		43				51									0	0		
					94		59			70	72										0	0	
		7	65					39	49													0	0
43					66		41				26	7											0

Figure 2: Base matrix of a 1/2-rate (2304,1152)-QC-LDPC code utilized in IEEE 802.16e.

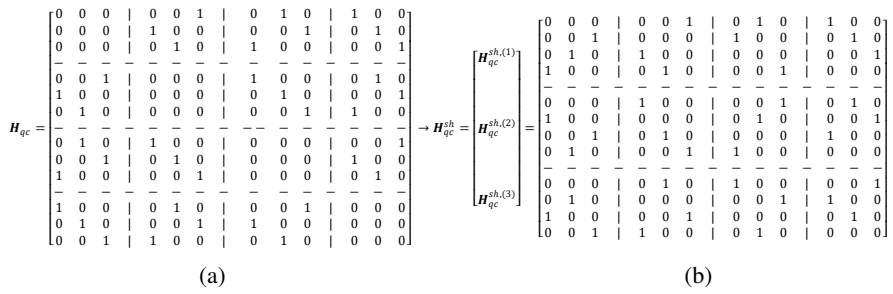
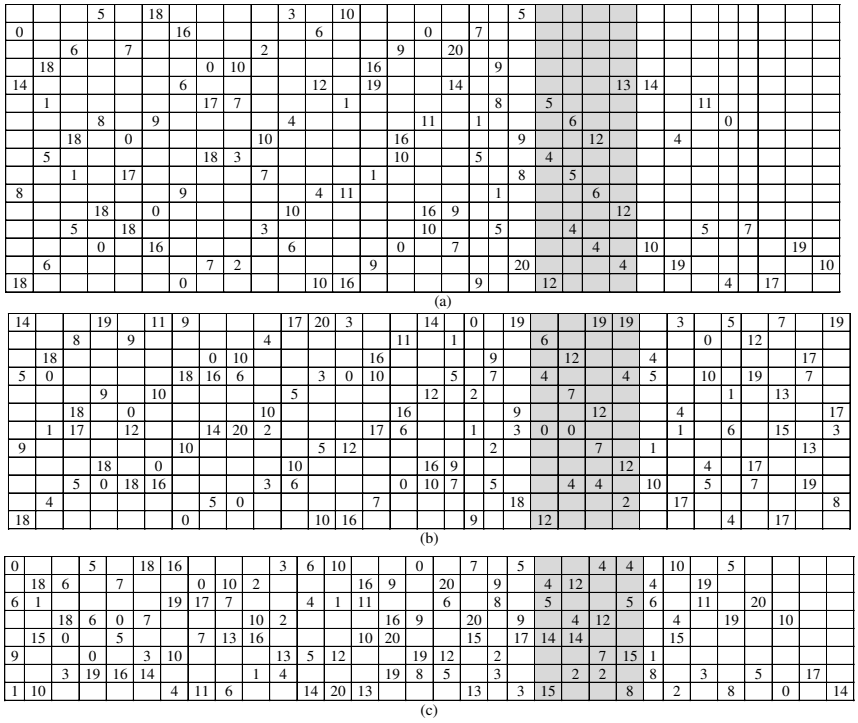


Figure 3: (a) PCM of a $(12, 4)$-QC-LDPC code; (b) its shuffled version.

Table 2: Implementation results of the example codes on a Xilinx Virtex-7 FPGA.

Code	Attributes	LD, shuffled	LD, non-shuffled
IEEE 802.15.3c (672,336)	Clk freq. [MHz]	209	169
	On-chip power [W]	0.840	6.693
	Area (# LUTs)	15299	262122
IEEE 802.15.3c (672,504)	Clk freq. [MHz]	175	144
	On-chip power [W]	0.710	5.461
	Area (# LUTs)	16007	218612
IEEE 802.15.3c (672,425)	Clk freq. [MHz]	200	185
	On-chip power [W]	0.900	4.786
	Area (# LUTs)	14779	185478
IEEE 802.16.e (2304,1152)	Clk freq. [MHz]	114	–
	On-chip power [W]	2.640	–
	Area (# LUTs)	185478	–



(a)

(b)

(c)

Figure 4: Base matrices of IEEE 802.15.3c QC-LDPC codes with offset values: (a) (672,336)-LDPC code; (b) (672-425)-LDPC code; (c) (672,504)-LDPC code.

	94	73						55	83			7	0														
	28				23	80	10				13		1	1													
			24	22	81		33				0			0	0												
62		48						66	26						1	1											
		39				84			41	72						0	0										
				47	41		83				80	1					1	1									
		95	53						14	18								0	0								
	12	74				3			48									1	1								
12				83	24		43				51								0	0							
					95		60			71	73									1	1						
		7	65					39	49												0	0					
44					67		42				27	8										1					

Figure 5: Base matrix of the 1/2-rate (2304,1152) IEEE 802.16e QC-LDPC code with offset values.

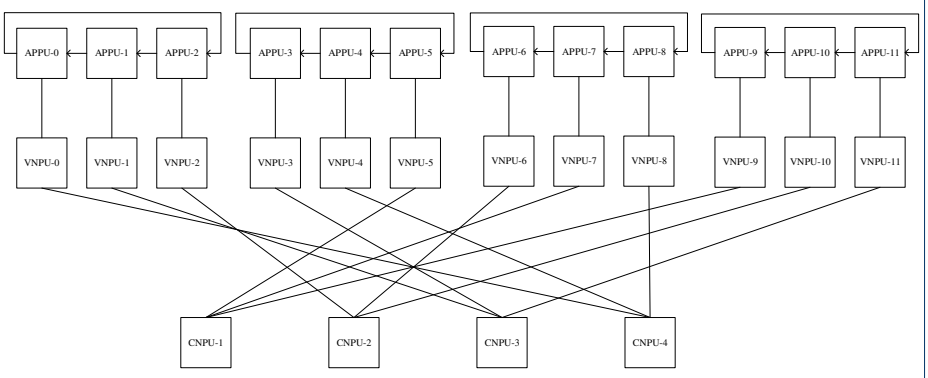


Figure 6: LD architecture for shuffled PCM of Figure 3.

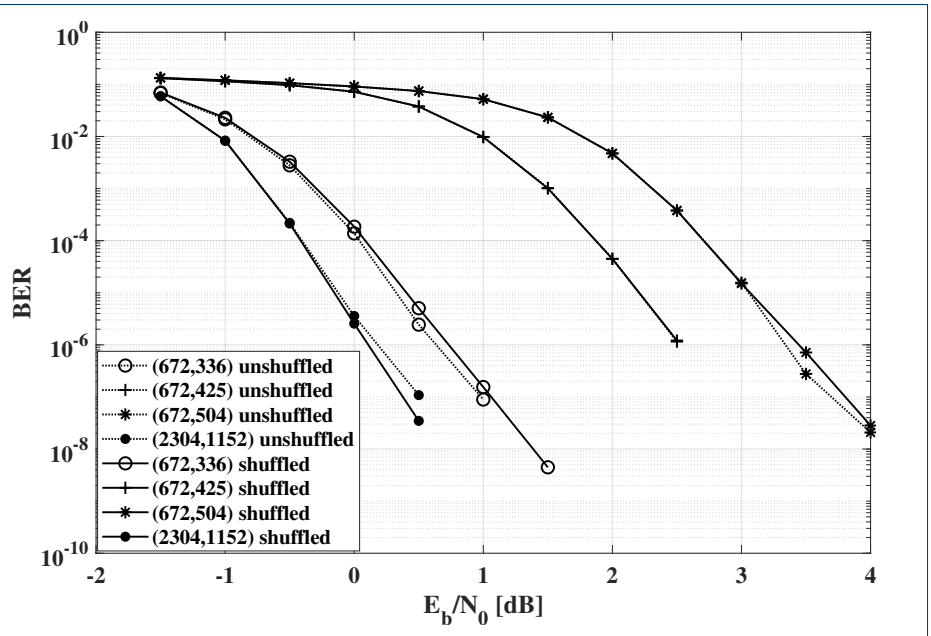


Figure 7: BER performance for the example QC-LDPC codes.

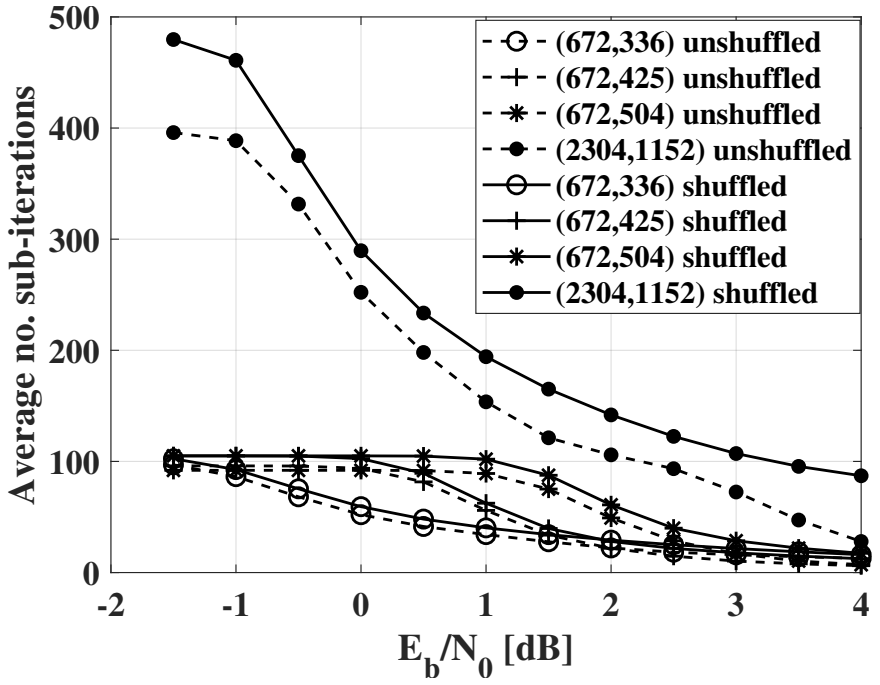


Figure 8: Average number of sub-iterations required for achieving the BER performance of Fig. 7.

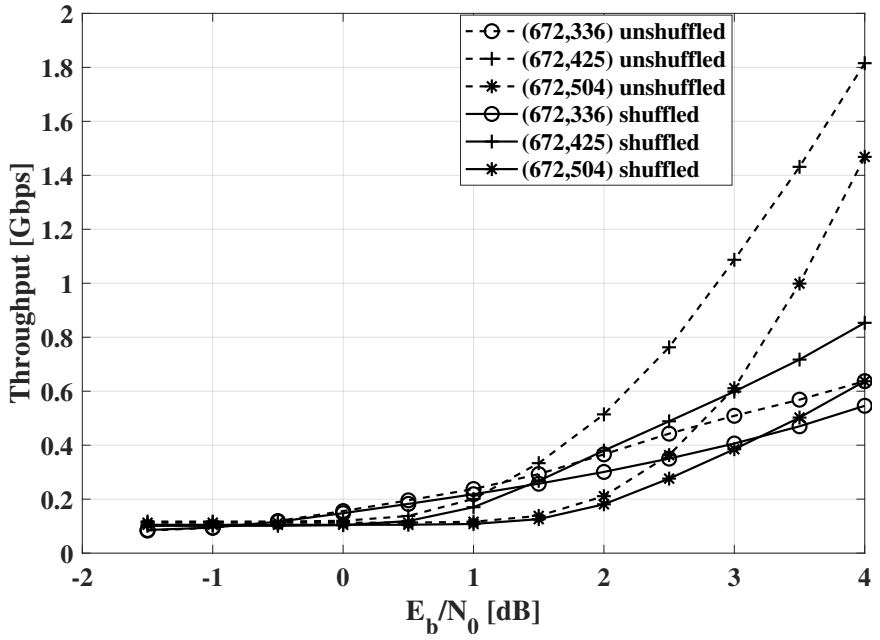


Figure 9: Average throughput.

Figures

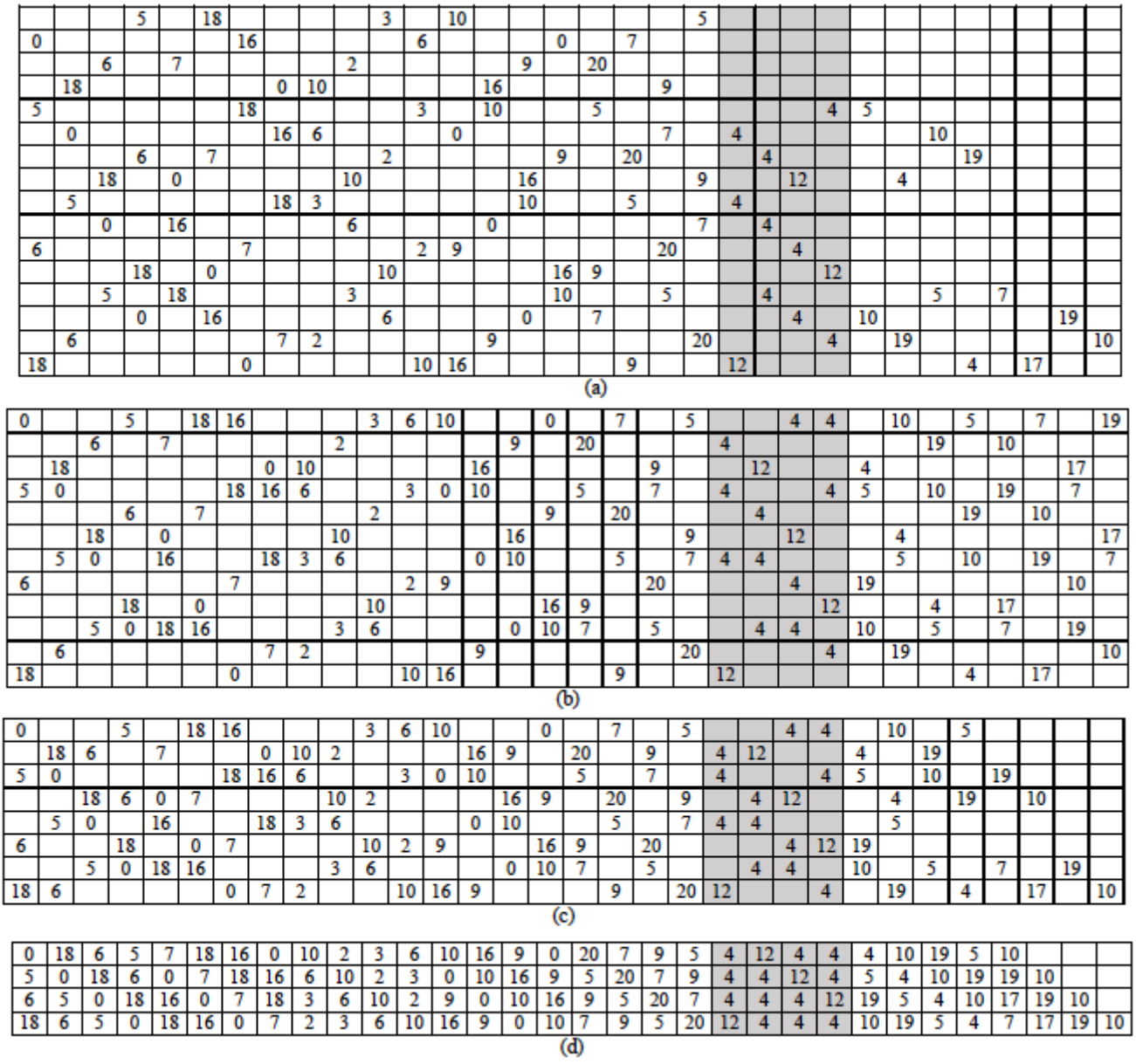


Figure 1

Base matrices of IEEE 802.15.3c QC-LDPC codes [49]: (a) (672,336)- LDPC code; (b) (672-425)-LDPC code; (c) (672,504)-LDPC code; (d) (672,588)- LDPC code.

	94	73						55	83			7	0										
	27				22	79	9				12		0	0									
			24	22	81		33				0			0	0								
61		47						65	25						0	0							
		39				84			41	72						0	0						
				46	40		82				79	0					0	0					
		95	53						14	18								0	0				
	11	73				2			47										0	0			
12				83	24		43				51									0	0		
					94		59			70	72										0	0	
		7	65					39	49													0	0
43					66		41				26	7											0

Figure 2

Base matrix of a 1/2-rate (2304,1152)-QC-LDPC code utilized in IEEE 802.16e.

$$\boldsymbol{H}_{qc} = \left[\begin{array}{ccc|ccc|ccc|ccc} 0&0&0&0&0&1&0&1&0&1&0&0\\ 0&0&0&1&0&0&0&0&1&0&1&0\\ 0&0&0&0&1&0&1&0&0&0&0&1\\ \hline 0&0&1&0&0&0&1&0&0&0&1&0\\ 1&0&0&0&0&0&0&1&0&0&0&1\\ 0&1&0&0&0&0&0&0&1&1&0&0\\ \hline 0&1&0&1&0&0&0&0&0&0&0&1\\ 0&0&1&0&1&0&0&0&0&1&0&0\\ 1&0&0&0&0&1&0&0&0&0&1&0\\ \hline 1&0&0&0&1&0&0&0&1&0&0&0\\ 0&1&0&0&0&1&1&0&0&0&0&0\\ 0&0&1&1&0&0&0&1&0&0&0&0 \end{array}\right] \rightarrow \boldsymbol{H}_{qc}^{sh} = \left[\begin{array}{c} \boldsymbol{H}_{qc}^{sh,(1)}\\ \boldsymbol{H}_{qc}^{sh,(2)}\\ \boldsymbol{H}_{qc}^{sh,(3)} \end{array}\right] = \left[\begin{array}{ccc|ccc|ccc|ccc} 0&0&0&0&0&1&0&1&0&1&0&0\\ 0&0&1&0&0&0&1&0&0&0&1&0\\ 0&1&0&1&0&0&0&0&0&0&0&1\\ 1&0&0&0&1&0&0&0&1&0&0&0\\ \hline 0&0&0&1&0&0&0&0&1&0&1&0\\ 1&0&0&0&0&0&0&1&0&0&0&1\\ 0&0&1&0&1&0&0&0&0&1&0&0\\ 0&1&0&0&0&1&1&0&0&0&0&0\\ \hline 0&0&0&0&1&0&1&0&0&0&0&1\\ 0&1&0&0&0&0&0&0&1&1&0&0\\ 1&0&0&0&0&1&0&0&0&0&1&0\\ 0&0&1&1&0&0&0&1&0&0&0&0 \end{array}\right]$$

(a) (b)

Figure 3

(a) PCM of a (12; 4)-QC-LDPC code; (b) its shuffled version.

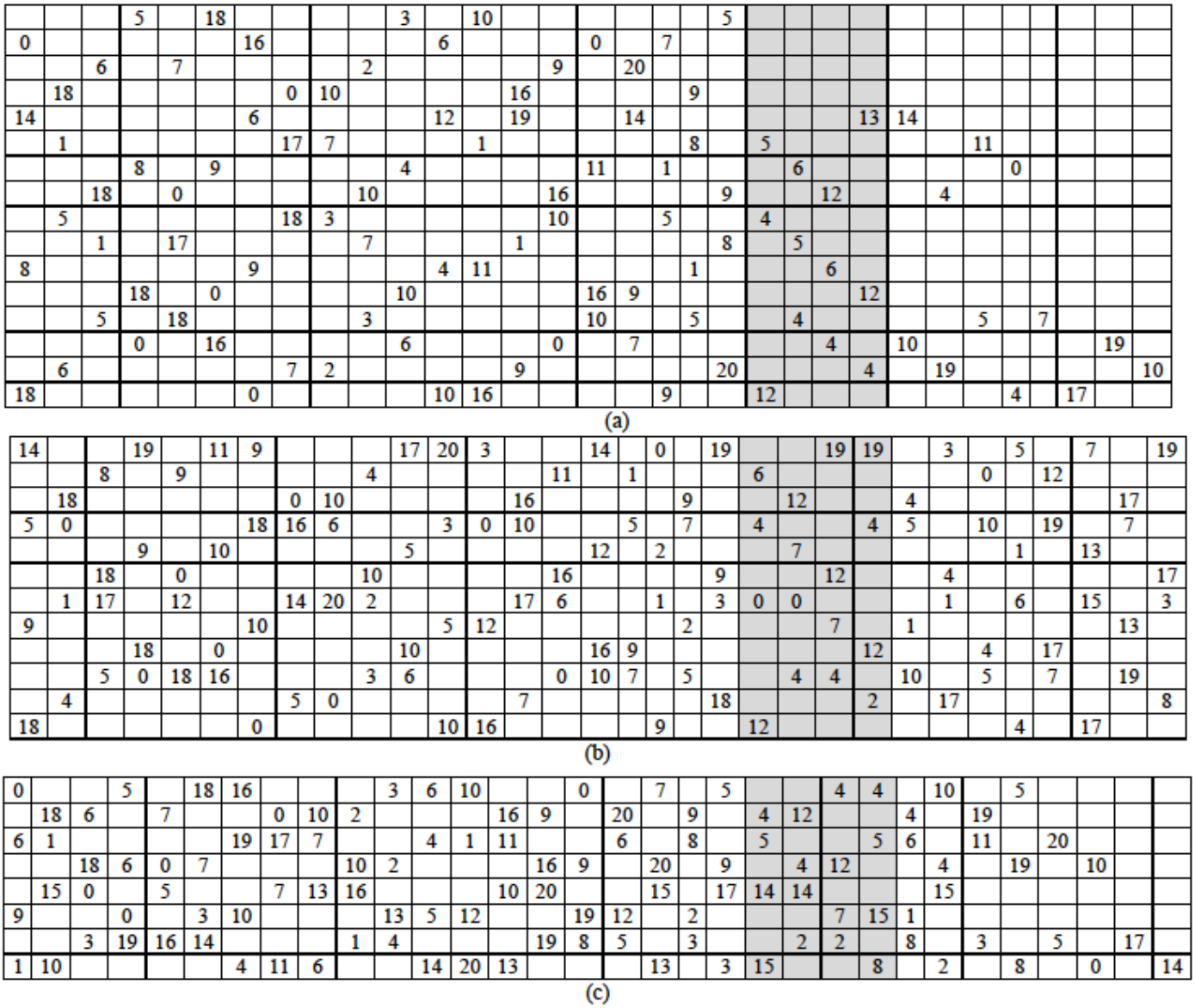


Figure 4

Base matrices of IEEE 802.15.3c QC-LDPC codes with offset values: (a) (672,336)-LDPC code; (b) (672-425)-LDPC code; (c) (672,504)-LDPC code.

	94	73						55	83			7	0										
	28				23	80	10				13		1	1									
			24	22	81		33				0			0	0								
62		48						66	26						1	1							
		39				84			41	72						0	0						
				47	41		83				80	1					1	1					
		95	53						14	18								0	0				
	12	74				3			48										1	1			
12				83	24		43				51									0	0		
					95		60			71	73										1	1	
		7	65					39	49													0	0
44					67		42				27	8											1

Figure 5

Base matrix of the 1/2-rate (2304,1152) IEEE 802.16e QC-LDPC code with offset values.

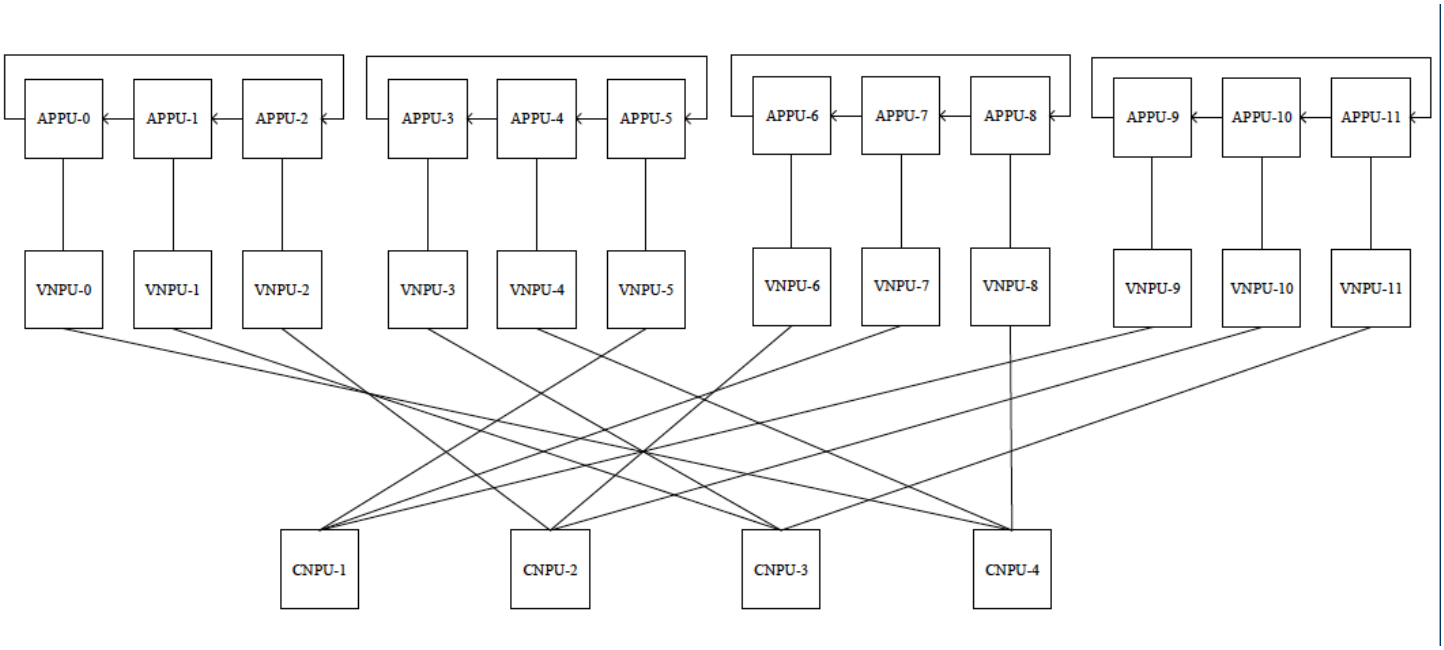


Figure 6

LD architecture for shuffled PCM of Figure 3.

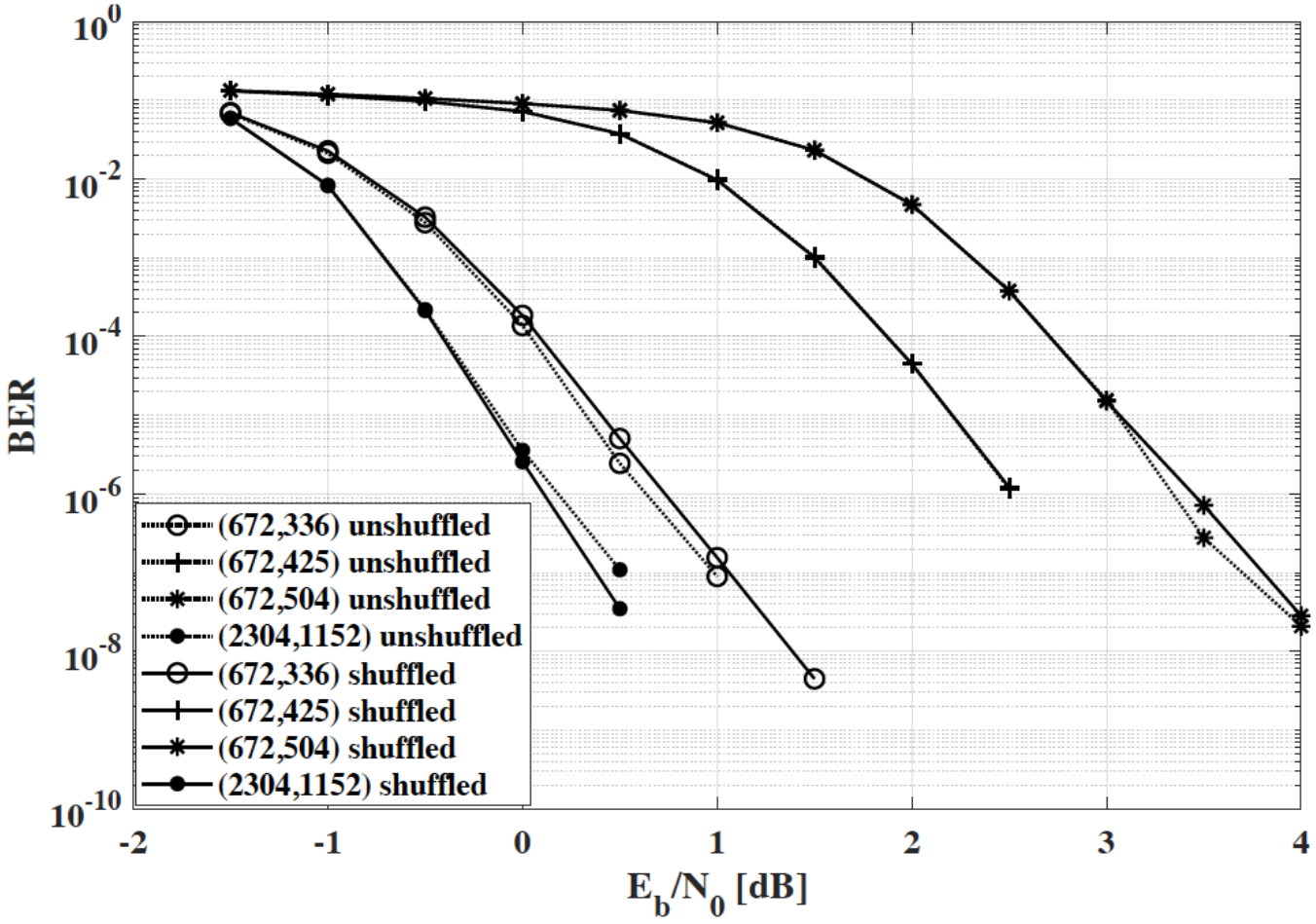


Figure 7

BER performance for the example QC-LDPC codes.

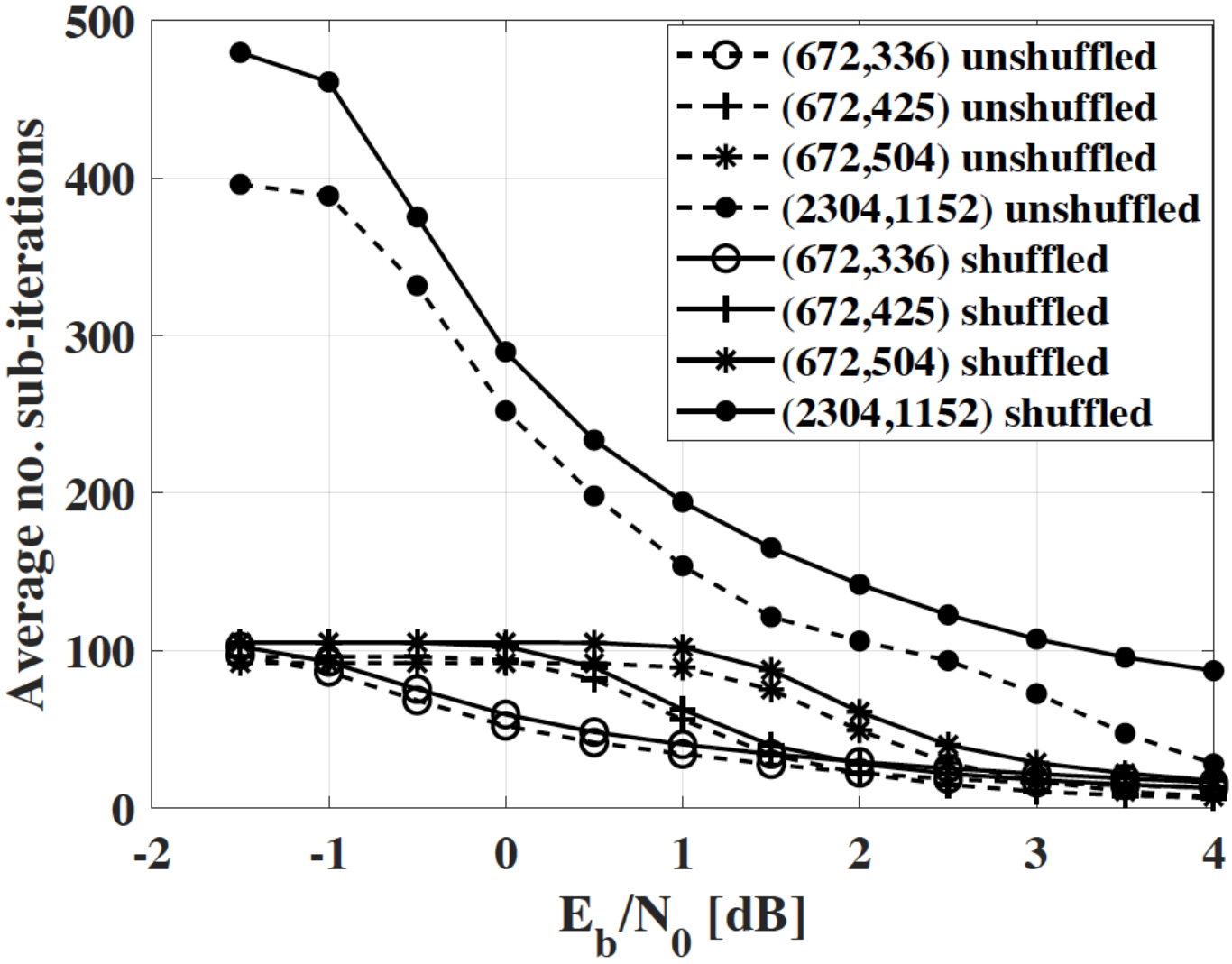


Figure 8

Average number of sub-iterations required for achieving the BER performance of Fig. 7.

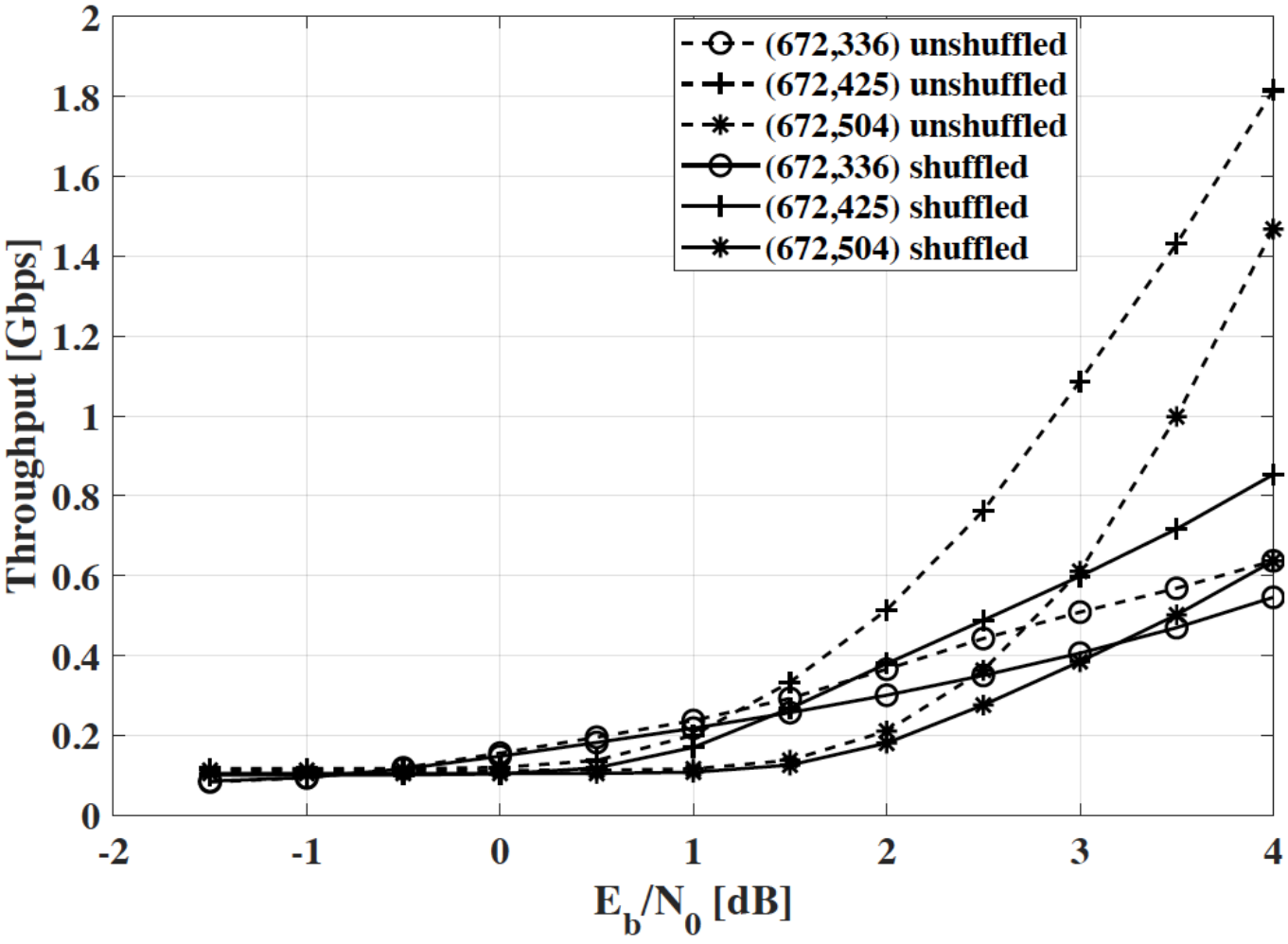


Figure 9

Average throughput.