Hasani *et al. J Wireless Com Network* (2021) 2021:183
https://doi.org/10.1186/s13638-021-02056-5

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Reduced-complexity decoding implementation of QC-LDPC codes with modified shuffling

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

*Correspondence:
hasani@ihp-microelectronics.com
[2] Department of Electrical and Computer Engineering, Brandenburg University of Technology, Cottbus-Senftenberg, Cottbus, Germany
Full list of author information is available at the end of the article

Abstract

Layered decoding (LD) facilitates a partially parallel architecture for performing belief propagation (BP) algorithm for decoding low-density parity-check (LDPC) codes. Such a schedule for LDPC codes has, in general, reduced implementation complexity compared to a fully parallel architecture and higher convergence rate compared to both serial and parallel architectures, regardless of the codeword length or code-rate. In this paper, we introduce a modified shuffling method which shuffles the rows of the parity-check matrix (PCM) of a quasi-cyclic LDPC (QC-LDPC) code, yielding a PCM in which each layer can be produced by the circulation of its above layer one symbol to the right. The proposed shuffling scheme additionally guarantees the columns of a layer of the shuffled PCM to be either zero weight or single weight. This condition has a key role in further decreasing LD complexity. We show that due to these two properties, the number of occupied look-up tables (LUTs) on a field programmable gate array (FPGA) reduces by about 93% and consumed on-chip power by nearly 80%, while the bit error rate (BER) performance is maintained. The only drawback of the shuffling is the degradation of decoding throughput, which is negligible for low values of E_b/N_0 until the BER of 1e−6.

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

1 Introduction

Forward error correction (FEC) methods are one of the vital elements of next-generation wireless networks tasked to provide the required level of reliability. Nevertheless, powerful capacity-achieving FEC techniques like low-density parity-check (LDPC), turbo or polar codes have the downside of higher complexity and power consumption compared with traditional coding techniques. Among these codes, LDPC codes have been incorporated into several previous technologies and are seen as the potential candidate for the new standards like Fifth Generation (5G) and IEEE 802.11ax.

Belief propagation (BP) algorithm is commonly used as the decoding method for LDPC codes, as the parity-check matrix (PCM) of these codes is sparse and their Tanner graph representation lacks short cycles of length 4. One important issue with regard

to BP decoding is the schedule of the algorithm, i.e., the order in which the reliability messages are exchanged between the nodes of the Tanner graph. BP schedule is directly associated with the implementation architecture of the decoding method, and it falls into three main categories:

1. Fully parallel architecture which is realized by flood schedule [1] 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 yielding a high throughput, this schedule requires a large silicon area with high interconnect complexity [2]. This architecture is facilitated due to the inherent parallelizable feature of BP algorithm, which is in contrast to turbo codes whose decoding algorithms are inherently serial. However, several works have devised fully parallel architectures for decoding turbo codes [3–6].
2. Serial architecture in which 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.
3. Partially parallel architecture which is a good trade-off between hardware complexity and decoding throughput and it is best accomplished by layered decoding (LD) schedule [7–33].

In LD schedule, the rows of PCM are divided into a number of layers, and each iteration of the BP algorithm is likewise split into the same number of sub-iterations. Each sub-iteration runs over one layer of the PCM, during which the CNs of that layer exchange reliability messages with their neighbor VNs. At the end of a sub-iteration, updated reliability messages are delivered 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, causing a reduced hardware utilization of LD compared to flood schedule. Furthermore, LD schedule achieves a better convergence performance than the flood schedule due to the fact that the latest variable-to-check (VTC) messages are always used to update the check-to-variable (CTV) messages during a sub-iteration.

For the sake of LD complexity, it is highly desirable that the number of ones in each column of a layer be either one or zero. Quasi-cyclic LDPC (QC-LDPC) codes have inherently a layered structure with such a property, thus making them an appropriate candidate for LD. QC-LDPC codes are a special type of LDPC codes possessing a cyclic property which simplifies the encoding and decoding process of them, while preserving comparable performance to random (or unstructured) LDPC codes [34, 35].

2 Related work and contributions

Shuffling idea proposed in [15] shuffles the rows of the PCM of a QC-LDPC code prior to decoding, in the sense that the order of the rows of the PCM is totally changed. After shuffling, each layer can be produced by circulating its above layer one symbol to the right, leading to a simplified LD and sped-up convergence rate. In particular, due to the cyclic property, it is enough to realize only the first layer of a PCM in hardware rather than the whole PCM. The downside of this shuffling is that it may spoil the primary property of single weight columns in the PCM.

To workaround this shortcoming, we outlined a modified shuffling idea in our previous work [36] which results in a shuffled PCM that retains the desired property of single weight columns, and possesses the cyclic property too. This was accomplished by introducing a set of offset values prior to performing the shuffling. In this paper, this modified shuffling idea is further investigated. To be specific

1 The logic behind offset values applied for shuffling is clarified, aiming to elaborate how the offset values come into effect. The procedure for determining the offset values is also outlined.
2 Since [36] lacks implementation results to verify the improvements promised by the modified shuffling method, we provide in this work the implementation results for LD of several QC-LDPC codes when shuffled with the proposed technique. Improvements in terms of number of occupied look-up tables (LUTs) on a field programmable gate array (FPGA) and also power consumption are observed when compared with the case of non-shuffled LD. These improvements are achieved without sacrificing bit error rate (BER) performance. Although the decoding throughput deteriorates as E_b/N_0 rises, our analysis shows that if BER of 1e−6 is chosen as the target, throughput degradation will be insubstantial.

The organization of the paper is as follows. Section 3 presents necessary fundamentals of QC-LDPC codes and LD. Section 4 is devoted to assessment of the novel shuffling method and its attributes. Implementation and simulation results together with necessary analysis come in Sect. 5. Final conclusions are made in Sect. 6.

3 Preliminaries

3.1 QC-LDPC codes

The PCM of a QC-LDPC code is comprised of Circulant Permutation Matrices (CPMs) and zero matrices, wherein a CPM is a shifted identity matrix. Such a PCM could be represented as

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

in which $c \leq t$ and $\boldsymbol{A}_{i,j}$s are either $b \times b$ CPMs or $b \times b$ zero matrices. A codeword $\boldsymbol{\nu}$ is of length $t.b$ comprising t sections $\boldsymbol{\nu} = (\boldsymbol{\nu_1}, \boldsymbol{\nu_2}, \cdots, \boldsymbol{\nu_t})$ with each section $\boldsymbol{\nu_i}, 1 \leq i \leq t$ of length b. Codewords of a QC-LDPC code have sectionized cyclic structure, in the sense that with cyclic shifting of the t sections in a codeword, another valid codeword is obtained [37]. A compact way for representing PCM of a QC-LDPC code is known as the base matrix, denoted by $\boldsymbol{\mathcal{W}}$. In $\boldsymbol{\mathcal{W}}$, non-negative integers specify the shifting value with respect to an identity matrix in the corresponding CPM, and other entries, usually chosen to be -1, represent zero matrices in the PCM. 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

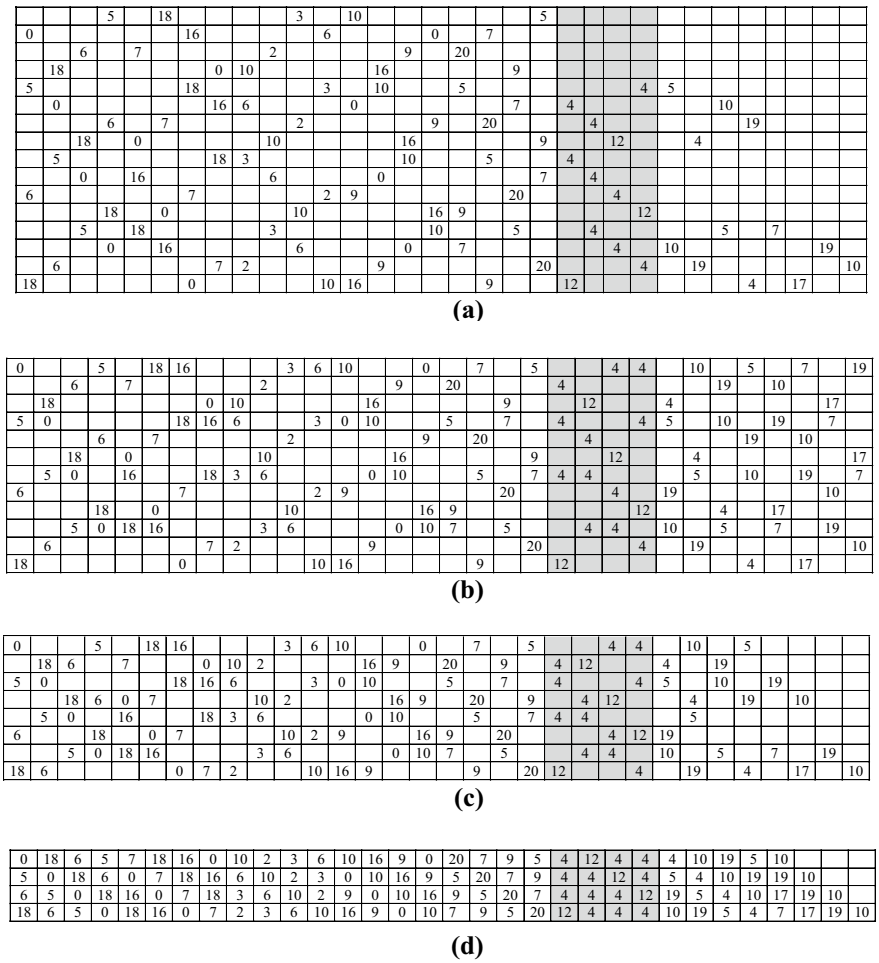

Fig. 1 Base matrices of IEEE 802.15.3c QC-LDPC codes [39]: **a** (672,336)-LDPC code; **b** (672-425)-LDPC code; **c** (672,504)-LDPC code; **d** (672,588)-LDPC code

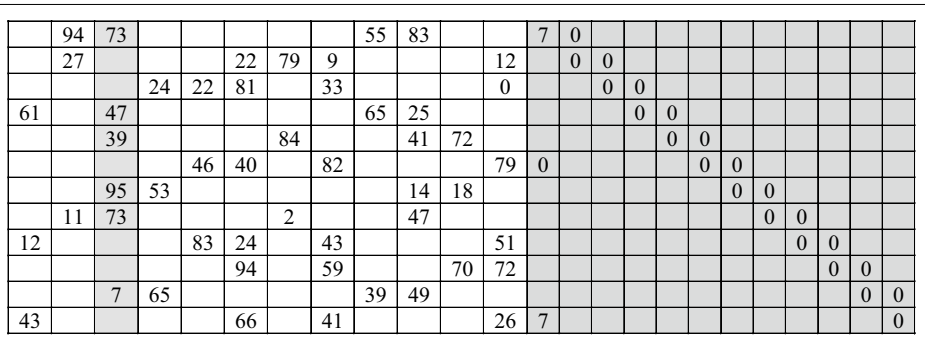

Fig. 2 Base matrix of a 1/2-rate (2304,1152)-QC-LDPC code utilized in IEEE 802.16e

(2304,1152)-QC-LDPC code used in IEEE 802.16e. In these two figures, empty places are the locations of zero matrices.

Tanner graph representation of a PCM is an important means to comprehend BP algorithm. It consists of two sets of nodes, where one set represents CNs, i.e.,

parity-check sums (equivalent to rows of $\boldsymbol{H}$), and the other set represents VNs (equivalent to columns of $\boldsymbol{H}$). A CN in a Tanner graph is connected to a VN if and only if the corresponding element of $\boldsymbol{H}$ is one.

3.2 LD schedule

One appropriate schedule which facilitates partially parallel architecture for execution of BP algorithm is LD. In this schedule, each iteration is split into several sub-iterations, each corresponding to a layer of the PCM. In each sub-iteration, CNs of the corresponding layer exchange reliability messages with VNs of that layer, and at the end, the updated reliability messages are provided to the next layer. Accordingly, fewer number of processing units for CNs and VNs are realized in hardware, and they are re-utilized in successive sub-iterations for successive layers.

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. The J rows of the PCM $\boldsymbol{H}_{qc}$ are divided into L layers each containing E consecutive rows. Hence, the i-th layer $\boldsymbol{H}_{qc}^{(i)}, 1 \leq i \leq L$ contains rows $(i-1)E+1, \ldots, i.E$ of $\boldsymbol{H}_{qc}$. The i-th sub-iteration of LD algorithm, associated with the i-th layer of the PCM, is split into three steps:

1 Vertical step: VTC messages are updated as

$$Z_{j,l}^{(i)} = Y_l - L_{j,l}^{(i)}, 1 \leq j \leq E, 0 \leq l < n, \tag{2}$$

where Y_l, $0 \leq l < n$ are A Posteriori Probability (APP) values of VNs initially set as $\boldsymbol{y}$, and $L_{j,l}^{(i)}, 1 \leq j \leq E, 0 \leq l < n, 1 \leq i \leq L$ are CTV messages corresponding to the i-th layer initially set to zero for all the layers.

2 Horizontal step: CTV messages are updated as

$$L_{j,l}^{(i)} = \prod_{l' \in \mathcal{B}(\boldsymbol{h}_j^{(i)}) \setminus l} sgn(Z_{j,l'}^{(i)}) \times \min_{l' \in \mathcal{B}(\boldsymbol{h}_j^{(i)}) \setminus l} |Z_{j,l'}^{(i)}|, \tag{3}$$

where

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

denotes the set of VNs neighbor to the CN $\boldsymbol{h}_j^{(i)}$.

3 Hard decision and early termination criterion: APP values are updated according to:

$$Y_l = y_l + \sum_{j \in \mathcal{A}_l^{(i)}} L_{j,l}^{(i)}, 0 \leq l < n. \tag{4}$$

with

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

analogous to $\mathcal{B}(\boldsymbol{h}_j^{(i)})$ representing the set of CNs in layer i connected to v_l.

In (3), min-sum scheme [38] has been used for computing CTV messages, wherein $sgn(x)$ is a sign function which is equal to 1 when $x \geq 0$ and -1 otherwise.

At the end of each sub-iteration, some codeword bits are estimated based on the updated APP values. In particular, $\hat{v}_l = 1$ if $Y_l > 0$ and $\hat{v}_l = 0$ otherwise. If the check-sum condition $\boldsymbol{H}_{qc}\hat{\boldsymbol{v}}^T = 0$ is satisfied by the estimated codeword $\hat{\boldsymbol{v}} = (\hat{v}_0, \ldots, \hat{v}_{n-1})$, it will be declared as the valid codeword leading to the termination of the algorithm. Otherwise, the algorithm continues starting from the vertical step of the next layer. The maximum number of iterations is however limited by a threshold $I_{\max}$. The algorithm declares a failure if decoding is not converged to a valid codeword within $I_{\max}$ iterations.

4 Modified shuffling of QC-LDPC codes

4.1 Modified shuffling

LD schedule is suitably tailored for decoding QC-LDPC codes, since each block-row $\boldsymbol{A}_i$, $1 \le i \le c$ in (1) can be regarded as a layer. Accordingly, the columns in each layer are either single weight or zero weight. This property is useful for simplifying the decoding implementation. First and foremost, the summation in (4) is simplified as it exists only a single CTV message to be added to y_l. Moreover, the implementation of both operations in (2) and (3) are also simplified.

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. [15] proposes to partition the $J = c.b$ rows of $\boldsymbol{H}_{qc}$ between matrices $\boldsymbol{H}_{qc}^{sh,(i)}$, $i = 1, \ldots, b$ each containing the rows $i, i+b, i+2b, \ldots, i+(c-1)b$ of $\boldsymbol{H}_{qc}$. Accordingly, the shuffled PCM, $\boldsymbol{H}_{qc}^{sh}$, has matrices $\boldsymbol{H}_{qc}^{sh,(i)}$, $i = 1, \ldots, b$ as its row-blocks:

$$\boldsymbol{H}_{qc}^{sh} = \begin{bmatrix} \boldsymbol{H}_{qc}^{sh,(1)} \\ \vdots \\ \boldsymbol{H}_{qc}^{sh,(b)} \end{bmatrix} \tag{5}$$

An example of such a shuffling is depicted in Fig. 3, which has been performed on a (12, 4)-QC-LDPC code. 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, noting that the circulation must be performed separately on the t individual sections of a layer. Due to this cyclic property provided by shuffling, it is no longer needed to implement the whole PCM in the targeted 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 elaborated on in Sect. 4.2.

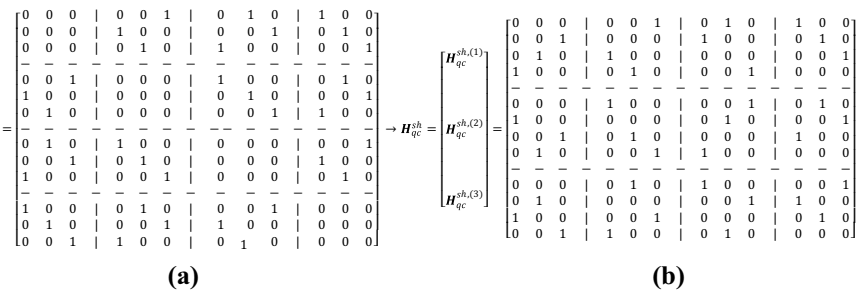

Fig. 3 **a** PCM of a (12, 4)-QC-LDPC code; **b** its shuffled version

The shortcoming with this shuffling method is that layers in the shuffled matrix may no longer be comprised of single weight columns. In particular, if the base matrix of a QC-LDPC code has repetitive numbers in a column, the corresponding block-column in the shuffled matrix will have columns of weight bigger than one in each layer. For instance, in Figs. 1 and 2, the shaded columns have repetitive numbers. Consequently, after 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 integer values $0 \leq o_m < b$, $m = 1, \ldots, c$ serving as an offset in order to modify the order of the rows of $\boldsymbol{H}_{qc}$ which are put in the same layer in $\boldsymbol{H}_{qc}^{sh}$. Accordingly, 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 are carefully selected so that in $\boldsymbol{H}_{qc}^{sh}$ no column in a layer has the weight bigger than one. 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 together with the resulting modified base matrices are shown in Fig. 4 and 5. As observed, with the introduced offset values in these examples, the shaded columns no longer have repetitive integers.

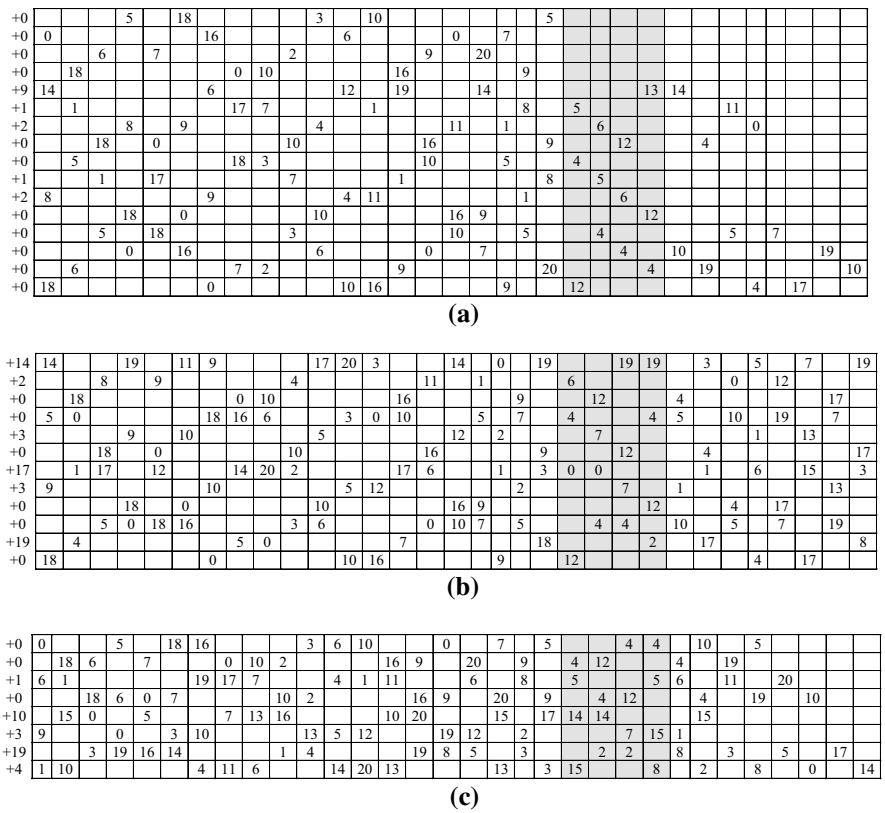

Fig. 4 Base matrices of IEEE 802.15.3c QC-LDPC codes with offset values: **a** (672,336)-LDPC code; **b** (672-425)-LDPC code; **c** (672,504)-LDPC code

	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

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

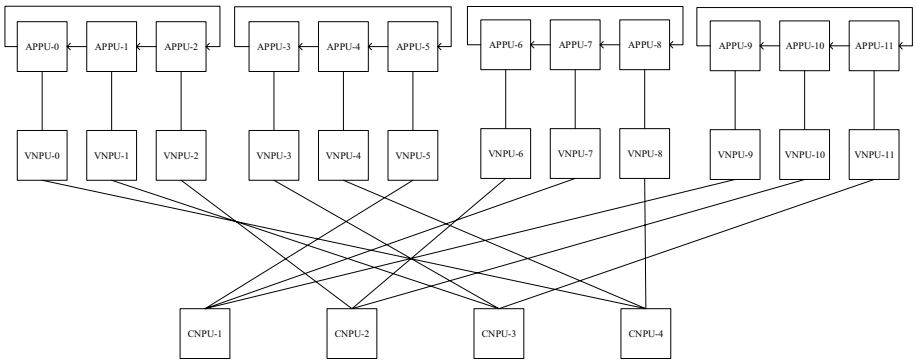

Fig. 6 LD architecture for shuffled PCM of Figure 3

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 finding the one that 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 which must be examined one by one until finding the one which results in a base matrix without repetitive integers in a column. Once such an offset set is found, the search is 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 be produced from its previous layer by a circular shifting.

4.2 LD implementation

The advantage of shuffling can be highlighted by an investigation of the implementation methodology. Specifically, 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) stand for a processing unit responsible for computing VTC and CTV messages in (2) and (3), 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

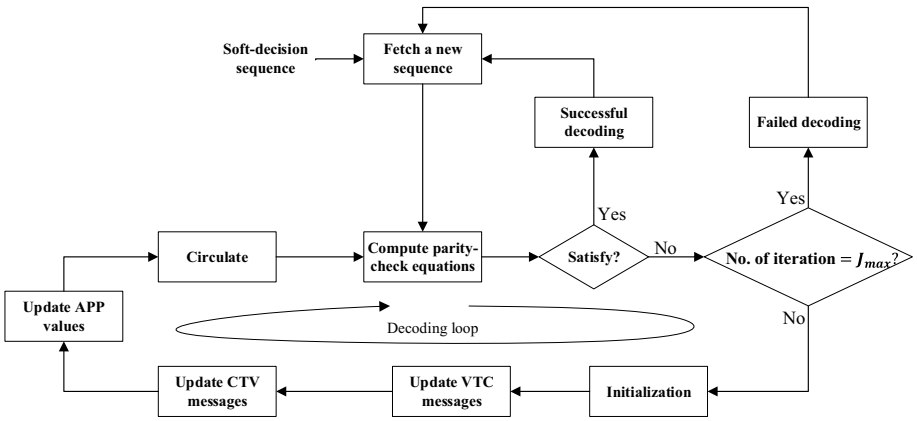

Fig. 7 Flowchart of the LD algorithm

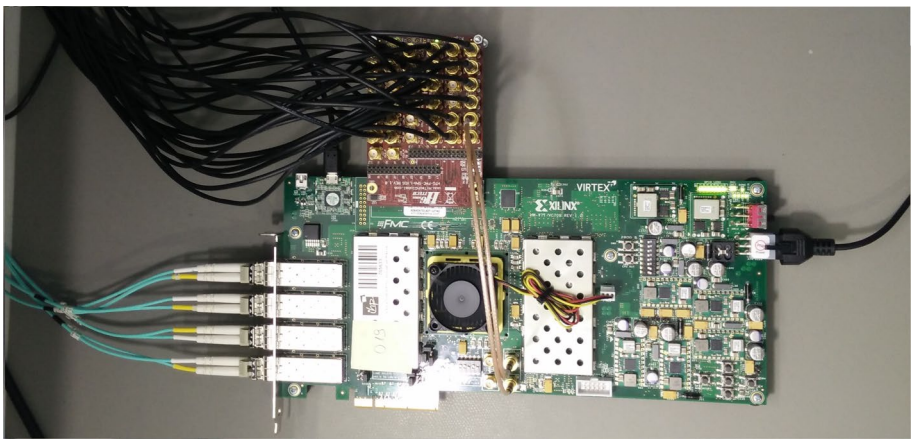

Fig. 8 Xilinx VC709 evaluation board with a Virtex-7 FPGA

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. This circulation serves as an alternative to redefining the connections between VNPUs and CNPUs, allowing the current connections between VNPUs and CNPUs to remain valid and to start the next sub-iteration immediately.

Fig. 7 shows the decoding flowchart for the algorithm outlined in section 3.2. As shown, the last block in the decoding loop is a "circulate" operation performed on the updated APP values in order to prepare them for the next sub-iteration processing. The other blocks in the decoding loop are responsible for performing operations (2)–(4).

5 Implementation and experimental results

We implemented both LD with shuffled PCM and LD with non-shuffled PCM for the example codes of IEEE 802.16e and IEEE 802.15.3c standards. The utilized hardware was a Xilinx VC709 evaluation board, shown in Fig. 8 which possesses a Virtex-7 XC7VX690T-2FFG1761C FPGA. The implementation was conducted with 6-bit quantized messages and the synthesis tool was Vivado 2018.3.

The acquired results in terms of utilized LUTs, on-chip power and maximum clock frequency are shown in Table 1. The first two parameters are directly reported by the synthesis tool and the maximum clock frequency is estimated from the parameter of worst negative slack, also reported by the synthesis tool. As deduced from the figures in the table, the design with shuffled PCM is considerably smaller, and the consumed power is notably lower. For instance, the number of occupied LUTs reduces from 262122 to only 15299 in the case of (672,336) code, equivalent to $(1-\frac{15299}{262122})\times 100 \cong 94\%$ reduction in the numbers of LUTs on FPGA. Similarly, a $(1-\frac{0.84}{6.693})\times 100 \cong 87\%$ reduction in consumed power is also resulted. 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.

Figure 9 depicts the performance simulation of the three IEEE 802.15.3c codes, showing that shuffling does not degrade the BER performance. Indeed, LD with (modified) shuffled PCM performs as good as LD with non-shuffled PCM. Furthermore, the average number of iterations needed to achieve a specific BER performance depicted in Fig. 10 is quite the same in two cases, further confirming the similar performance of the two modes. This is due to the fact that shuffling just changes the order of the rows of the PCM, while the overall PCM's characteristics remain intact. In particular, the determining attributes of the PCM such as distance property, cycles' distribution and rows' and columns' weight do not undergo any change.

Comparing the two cases in terms of throughput can also be of interest. The average throughput for different codes is plotted in Fig. 11. Given that f_{clk} is the clock frequency specified in table 1, $N_{\text{ave\_ite}}$ is the average number of sub-iterations and N_{clk} is the number of clock cycles each sub-iteration needs, the average duration for decoding a sequence will be then $\frac{N_{\text{ave\_ite}}.N_{\text{clk}}}{f_{\text{clk}}}$, thus leading to the average throughput of

$$\tau = \frac{k.f_{\text{clk}}}{N_{\text{ave\_ite}}.N_{\text{clk}}}. \tag{6}$$

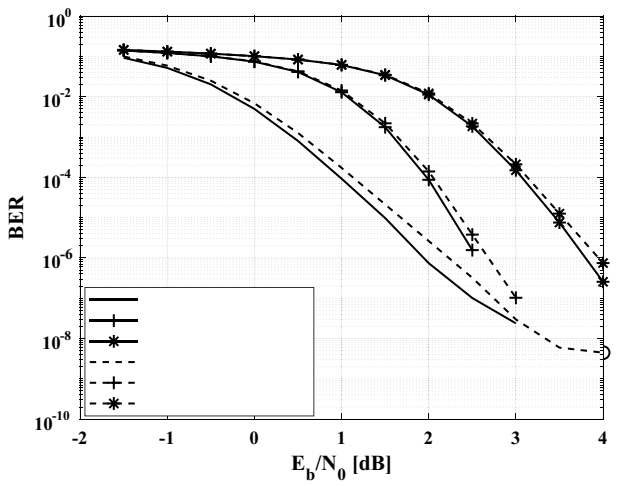

Fig. 9 BER performance for the example QC-LDPC codes

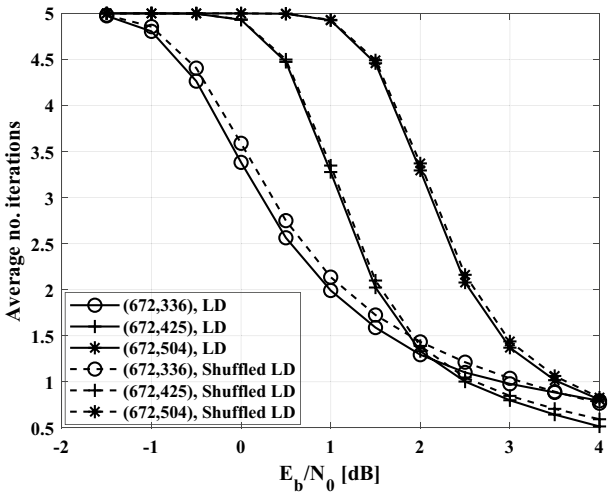

Fig. 10 Average number of sub-iterations required for achieving the BER performance of Fig. 9

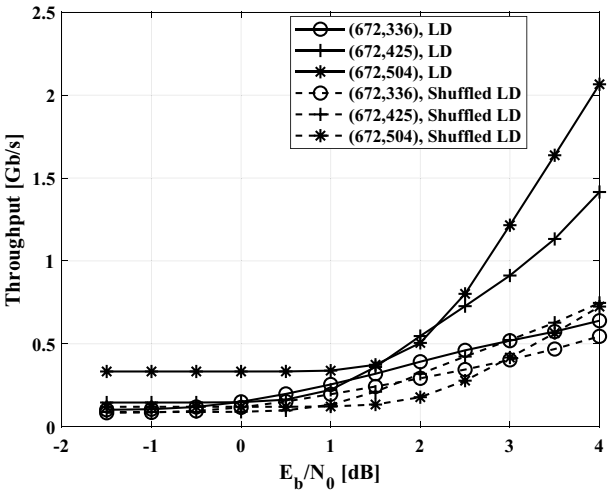

Fig. 11 Average throughput for the example QC-LDPC codes

Table 1 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	–

In our implementation, $N_{\text{clk}} = 8$ and 7 for the case of shuffled and non-shuffled PCM, respectively, noting that the one extra clock cycle in the former case is needed for cyclic shifting of the computed APP values. The average throughput in the two cases overlap for low values of E_b/N_0, indicating that the additional number of sub-iterations is compensated fully by the higher clock frequency. This is however not true when E_b/N_0 grows. If BER = 1e-6 is chosen as the targeted BER performance, the throughput degradation will be 0.1, 0.3, and 1.3 Gbps for the three codes, respectively. The degradation in throughput stems from the fact that layering is different in the 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.

6 Conclusion

The novel shuffling method proposed in this paper is basically a swapping of 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 weight or zero weight. Second, each layer must be producible from the upper layer by a one-symbol circular shifting. 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 of up to 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-LDPC: Quasi-Cyclic LDPC; VN: Variable Node; VNPU: VN Processing Unit; VTC: Variable-to-Check; 5G: Fifth Generation.

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 Dr. Steffen Büchner.

Authors' 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. All authors read and approved the final manuscript.

Funding

Open Access funding enabled and organized by Projekt DEAL. This work was supported by the German Research Foundation (DFG) project PSSS-FEC, project no. 442607813.

Availability of data and materials

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

Declarations

Competing interests

The authors declare that they have no competing interests.

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.

Received: 1 September 2020 Accepted: 22 September 2021
Published online: 09 October 2021

References

1. F.R. Kschischang, B.J. Frey, Iterative decoding of compound codes by probability propagation in graphical models. IEEE J. Sel. Areas Commun. **16**(2), 219–230 (1998)
2. L. Liu, C.-J.R. Shi, Sliced message passing: High throughput overlapped decoding of high-rate low-density parity-check codes. IEEE Trans. Circuits Syst. I Regul. Pap. **55**(11), 3697–3710 (2008)
3. R.G. Maunder, A fully-parallel turbo decoding algorithm. IEEE Trans. Commun. **63**(8), 2762–2775 (2015)
4. Y. Sun, J.R. Cavallaro, Efficient hardware implementation of a highly-parallel 3g pp LTE/LTE-advance turbo decoder. Integration **44**(4), 305–315 (2011)
5. S.K. Chronopoulos, V. Christofilakis, G. Tatsis, P. Kostarakis, Preliminary BER study of a TC-OFDM system operating under noisy conditions. J. Eng. Sci. Technol. Rev. **9**(4), 13–16 (2016)
6. S.K. Chronopoulos, V. Christofilakis, G. Tatsis, P. Kostarakis, Performance of turbo coded OFDM under the presence of various noise types. Wirel. Pers. Commun. **87**(4), 1319–1336 (2016)
7. M.M. Mansour, N.R. Shanbhag. Turbo decoder architectures for low-density parity-check codes, in *Global Telecommunications Conference. GLOBECOM'02. IEEE*, vol. 2 (IEEE, 2002), pp. 1383–1388
8. M.M. Mansour, N.R. Shanbhag, High-throughput LDPC decoders. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **11**(6), 976–996 (2003)
9. H. Sankar, K.R. Narayanan, Memory-efficient sum-product decoding of LDPC codes. IEEE Trans. Commun. **52**(8), 1225–1230 (2004)
10. D.E. Hocevar. A reduced complexity decoder architecture via layered decoding of LDPC codes, in *IEEE Workshop on Signal Processing Systems, 2004. SIPS 2004* (IEEE, 2004) pp. 107–112
11. M.M. Mansour, N.R. Shanbhag, A 640-mb/s 2048-bit programmable LDPC decoder chip. IEEE J. Solid-State Circuits **41**(3), 684–698 (2006)
12. 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* (IEEE, 2006), pp. 1192–1199
13. K. Gunnam, G. Choi, W. Wang, M. Yeary. 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* (IEEE, 2007), pp.1645–1648
14. 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)* (IEEE, 2006), pp. 537–544
15. Y.-L. Ueng, C.-C. Cheng. 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* (IEEE, 2007), pp. 1255–1259
16. P. Radosavljevic, A. de Baynast, J.R. Cavallaro. Optimized message passing schedules for LDPC decoding, in *Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers, 2005* (IEEE, 2005), pp.591–595
17. D. Yang, G. Yu, X. Zou, Y. Deng, J. Zhong. The design and verification of a novel LDPC decoder with high-efficiency, in *2014 International Symposium on Integrated Circuits (ISIC)* (IEEE, 2014), pp. 256–259
18. A. de Baynast, P. Radosavljevic, A. Sabharwal, J.R. Cavallaro. On turbo-schedules for LDPC decoding. IEEE Commun. Lett. (2006)
19. P. Radosavljevic, M. Karkooti, A. de Baynast, J.R. Cavallaro, Tradeoff analysis and architecture design of high throughput irregular LDPC decoders. IEEE Trans. Circuits Syst. I: Regul. Pap. **1**(1), 1 (2006)
20. T. Brack, M. Alles, F. Kienle, N. When. 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* (IEEE, 2006), pp. 1–5
21. G. Gentile, M. Rovini, L. Fanucci. 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)* (IEEE, 2007), pp. 369–375
22. K.K. Gunnam, G.S. Choi, M.B. Yeary, M. Atiquzzaman. VLSI architectures for layered decoding for irregular LDPC codes of WIMAX, in *2007 IEEE International Conference on Communications* (IEEE, 2007), pp. 4542–4547
23. K. Zhang, X. Huang, Z. Wang, High-throughput layered decoder implementation for quasi-cyclic LDPC codes. IEEE J. Sel. Areas Commun. **27**(6), 985–994 (2009)
24. J. Goldberger, H. Kfir, Serial schedules for belief-propagation: analysis of convergence time. IEEE Trans. Inf. Theory **54**(3), 1316–1319 (2008)
25. Y. Cui, X. Peng, Z. Chen, X. Zhao, Y. Lu, D. Zhou, S. Goto. Ultra low power qc-LDPC decoder with high parallelism, in *2011 IEEE International SOC Conference* (IEEE, 2011), pp. 142–145
26. J. Zhang, M.P. Fossorier, Shuffled iterative decoding. IEEE Trans. Commun. **53**(2), 209–213 (2005)
27. Y.-L. Ueng, C.-J. Yang, C.-J. Chen. A shuffled message-passing decoding method for memory-based LDPC decoders, in *2009 IEEE International Symposium on Circuits and Systems* (IEEE, 2009), pp. 892–895
28. J. Zhang, M. Fossorier. Shuffled belief propagation decoding, in *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers, 2002*, vol. 1 (IEEE, 2002), pp. 8–15
29. J. Zhang, Y. Wang, M.P. Fossorier, J.S. Yedidia, Iterative decoding with replicas. IEEE Trans. Inf. Theory **53**(5), 1644–1663 (2007)
30. Z. Cui, Z. Wang, X. Zhang, Q. Jia. Efficient decoder design for high-throughput LDPC decoding, in *APCCAS 2008–2008 IEEE Asia Pacific Conference on Circuits and Systems* (IEEE, 2008), pp. 1640–1643
31. Y.-L. Ueng, C.-J. Yang, K.-C. Wang, C.-J. Chen, A multimode shuffled iterative decoder architecture for high-rate RS-LDPC codes. IEEE Trans. Circuits Syst. I Regul. Pap. **57**(10), 2790–2803 (2010)
32. F. Guilloud, E. Boutillon, J. Tousch, J.-L. Danger, Generic description and synthesis of LDPC decoders. IEEE Trans. Commun. **55**(11), 2084–2091 (2007)
33. Y.-L. Ueng, B.-J. Yang, C.-J. Yang, H.-C. Lee, J.-D. Yang, An efficient multi-standard LDPC decoder design using hardware-friendly shuffled decoding. IEEE Trans. Circuits Syst. I Regul. Pap. **60**(3), 743–756 (2013)
34. Z. Wang, Z. Cui, Low-complexity high-speed decoder design for quasi-cyclic LDPC codes. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **15**(1), 104–114 (2007)
35. M.P. Fossorier, Quasicyclic low-density parity-check codes from circulant permutation matrices. IEEE Trans. Inf. Theory **50**(8), 1788–1793 (2004)

36. A. Hasani, L. Lopacinski, S. Büchner, J. Nolte, R. Kraemer. 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)* (IEEE, 2019), pp. 1–6
37. Z. Li, L. Chen, L. Zeng, S. Lin, W.H. Fong, Efficient encoding of quasi-cyclic low-density parity-check codes. IEEE Trans. Commun. **54**(1), 71–81 (2006)
38. J. Chen, M.P. Fossorier, Near optimum universal belief propagation based decoding of low-density parity check codes. IEEE Trans. Commun. **50**(3), 406–414 (2002)
39. S.-W. Yen, S.-Y. Hung, C.-L. Chen, H.-C. Chang, S.-J. Jou, C.-Y. Lee, A 5.79-gb/s energy-efficient multirate LDPC codec chip for IEEE 802.15. 3c applications. IEEE J. Solid-State Circuits **47**(9), 2246–2257 (2012)

Publisher's Note