A 3.0 Gb/s Throughput Hardware-Efficient Decoder for Cyclically-Coupled QC-LDPC Codes

Qing Lu, Jianfeng Fan, Chiu-Wing Sham, *Member, IEEE,* Wai M. Tam, and Francis C. M. Lau, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a new class of quasi-cyclic low-density parity-check (QC-LDPC) codes, namely cyclically-coupled QC-LDPC (CC-QC-LDPC) codes, and their RAM-based decoder architecture. CC-QC-LDPC codes have a simple structure and are constructed by cyclically-coupling a number of QC-LDPC sub-codes. They can achieve throughput and error performance as excellent as LDPC convolutional codes, but with much lower hardware requirements. They are therefore promising candidates for future generations of communication systems such as long-haul optical communication systems. In particular, a rate-$5/6$ CC-QC-LDPC decoder has been implemented onto a field-programmable gate array (FPGA) and it achieves a throughput of 3.0 Gb/s at 100 MHz clock rate with 10-iteration decoding. No error floor is observed up to an E_b/N_0 of 3.50 dB, where all 1.14×10^{16} transmitted bits have been decoded correctly.**

***Index Terms*—Cyclically-coupled QC-LDPC code, Decoder architecture, FPGA implementation, QC-LDPC code**

I. Introduction

With the growing need of advanced communication technologies, developing superior forward-error-correction (FEC) schemes has become imperative. Low-density parity-check (LDPC) block codes [1] have undoubtedly been one the most promising FEC classes recently due to their capability in approaching channel capacity. Quasi-cyclic LDPC (QC-LDPC) codes, having a regularized structure that reduces the encoder/decoder complexities, have further demonstrated the practical value of LDPC codes [2], [3], [4]. It has also been proved that QC-LDPC codes can achieve as excellent error performance as random LDPC codes [5].

By spatially-coupling consecutive LDPC block codes, LDPC convolutional codes (LDPCCCs) having an infinite code length are formed[1] [6], [7], [8]. LDPCCCs have been shown to possess lower thresholds and better error performance compared with LDPC block codes [9], [10]. However, when burst erasures occur and the erasures are beyond recovery, the chain decoding of LDPCCCs becomes seriously disrupted. Moreover, the infinite coupling structure of LDPCCCs incurs application challenges such as initial delay, termination difficulty and a gigantic memory requirement [9]. Subsequently, a tail-biting convolutional LDPC code has been proposed [11]. The tail-biting convolutional LDPC code is technically a block code. The overall length of the tail-biting code increases linearly with the number of times that a sliding convolutional structure is repeated (see Supplementary Materials for more details) and so is the memory requirement at the decoder. Also, if the decoder hardware is to be fully utilized, multiple codewords have to be decoded at the same time to fill the pipeline holes [12]. It implies much more memory will be needed at the decoder.

As to the decoding of LDPC (block or convolutional) codes, a method called sum-product algorithm (SPA) or belief propagation (BP) has been widely used [13], [14]. However, due to the complex process involved in updating the check-to-variable messages, the original SPA is usually applied with approximations. One of the most widely used substitutes is known as the min-sum (MS) algorithm [15], which executes the check-node updating by making simple comparisons. The min-sum algorithm hence has lowered the decoder complexity by a significant scale, but at the same time has caused a non-negligible bit-error-rate (BER) degradation [16], [17]. Normalized/offset MS algorithms have been proposed [18]. However, in practical implementations where messages are highly quantized, the accuracy degradation of MS techniques is still not negligible, especially when the check-node degree or code rate is high [16]. In another dimension, a layered decoding schedule is proposed to speed up the convergence of iterative decoding [19]. It has been shown that layered decoding can generally make the convergence rate two times faster compared to the flooding scheme [20], [21].

From time to time, the decoding practices of QC-LDPC codes, with either pragmatic or investigative purposes, are reported in literatures. A partially parallel decoder architecture has been developed using FPGA for a 9216-bit 1/2-rate (3,6)-regular LDPC code [22]. With a throughput of 54 Mbps, the decoder achieves a BER of 10^{-6} at $E_b/N_0 = 2$ dB over an additive white Gaussian noise (AWGN) channel. Then a modified SPA algorithm that balances the computation load between the two decoding phases has been implemented on a FPGA with both uniform and non-uniform 6-bit quantization [23]. For a 7/8-rate QC-LDPC code of length 8176 bits, a throughput of 175 Mbps is attained with a clock frequency of 193 MHz. Unlike the above practices, CMOS technology has been employed to complete a code-specific design characterized by a chunk-by-chunk check-node updating [24].

The work described in this paper was supported by a grant from the RGC of the Hong Kong SAR, China (Project No. PolyU 152088/15E).

The authors are with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Hong Kong (email: encm-lau@polyu.edu.hk).

[1]In general, spatially-coupled LDPC codes are constructed by re-connecting the edges among a large number of block codes and form a much larger code.

Min-sum algorithm together with layered decoding has been implemented using $0.18\ \mu m$ CMOS process [25]. Moreover, a bypassing scheme that effectively reduces the memory access and energy consumption has been proposed [24], [25]. However, without general applicability, this code-specific design may have a problem in migrating to other platforms.

As mentioned earlier, a great loss in BER performance is observed when the messages are quantized and min-sum decoding is used. The BER performance of quantized MS algorithm with offset is elaborated and optimized [26]. Despite a certain amount of BER compensation, implementation barriers exist. Another issue of common concern in literatures is memory requirement that relates to a great portion of hardware resources as well as power dissipation. Aware of the FPGA's popularity in evaluating LDPC codes, optimizations in memory-throughput tradeoff of modern FPGAs have been presented [27]. In addition, a synthesis tool is provided for architecture optimization of LDPC decoders.

In some specific applications such as future generations of optical transport networks, FEC schemes have to meet several characteristics. Firstly, a high code rate, say above 80%, is the key to achieving a high raw data rate. Secondly, no error floor should exist at or above a bit error rate of 10^{-15} so that no more than 10 bit errors exist per day in a 100 Gbps transmission link. Thirdly, a large net coding gain (NCG) is preferred considering the cost of installing repeaters. Last but not least, an acceptable complexity in the overall FEC design is a must due to practical concerns such as fabrication cost, power dissipation, etc.

With comparable decoder complexities, an LDPC convolutional code generally outperforms its block code counterpart in terms of bit error rate and net coding gain [6]. The advantages of using block code, however, keep spurring researchers to close the above gap. Intuitively we may borrow the idea of spatial-coupling from LDPCCC and apply it to the design of the block code such that a lower BER can be achieved. In this paper, we make use of the aforementioned idea and propose a specific type of QC-LDPC codes, namely *cyclically-coupled QC-LDPC (CC-QC-LDPC) codes*, as well as their decoder architecture.

Like QC-LDPCCC, a CC-QC-LDPC code can be constructed from an original QC-LDPC block code. However, CC-QC-LDPC codes differ from QC-LDPCCCs in the following aspects. First, a QC-LDPCCC is formed by re-connecting the edges among a number of consecutive QC-LDPC block codes; whereas our proposed CC-QC-LDPC code is constructed by combining variable nodes of consecutive block codes. Second, in a CC-QC-LDPC code, the coupling between consecutive block codes is performed in a cyclic manner, i.e., the first block couples with the second one, the second one with the third one, ..., and the last one with the first one (similar to tail-biting). Third, part of the original QC-LDPC block code does not couple with other block codes.

The contributions of this work can be summarized as follows. Firstly, we propose a new class of QC-LDPC codes called CC-QC-LDPC codes. Secondly, we propose a composite decoder architecture, which consists of subordinate QC-LDPC sub-decoders, for the decoding of CC-QC-LDPC codes. With mutual dependence between adjacent sub-decoders, the overall decoder gains a remarkable improvement in decoding capability. In addition, a concurrent operation is applied in the decoder resulting in a high decoding parallelism and a high throughput. Thirdly, specific arrangements are made to reduce the memory size of the random access memories (RAMs). Wherever possible, we assign only one RAM location for each variable-check connection. Thus the same location stores the variable-to-check (V2C) message and check-to-variable (C2V) message alternately. Compared to existing designs [25], [28], the required memory is substantially reduced. For some redundant messages not being able to be dynamically stored as such, we minimize the memory requirement by using two sets of RAMs — one for the odd layers and the other for the even layers. Fourthly, a check-node processor implementing the quantized SPA computation is applied. By using look-up tables (LUTs), the quantized SPA computation has a similar complexity as the quantized MS computation [12]. Also, we set up the LUTs in a parallel way to alleviate the delay. Last, we implement the decoder and evaluate its complexity under different code lengths. The BER results of a high rate, long CC-QC-LDPC code are attained under the AWGN channel. The proposed decoder has been experimentally demonstrated to achieve very low error rate, moderate complexity and high throughput.

The rest of this paper is organized as follows. Section II reviews the details of LDPC codes and the widely used SPA decoding algorithm. Section III describes the basic structure of our proposed CC-QC-LDPC codes. In Section IV, a CC-QC-LDPC decoder architecture with layered decoding is developed and presented. In Section V, an experiment to evaluate our decoders is presented with results and observations. Finally conclusions are drawn in Section VI.

II. Background

A. LDPC Codes

LDPC codes are a class of linear block codes that can be represented by a sparse parity-check matrix $\mathbf{H}$. In this matrix, each row serves as a constraint (check node) towards the specified received signal bits corresponding to columns (variable nodes) which contain elements of '1' in that row. The parity-check matrix of a QC-LDPC code is represented by

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}^{a_{1,1}} & \mathbf{I}^{a_{1,2}} & \dots & \mathbf{I}^{a_{1,L}} \\ \mathbf{I}^{a_{2,1}} & \mathbf{I}^{a_{2,2}} & \dots & \mathbf{I}^{a_{2,L}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{I}^{a_{J,1}} & \mathbf{I}^{a_{J,2}} & \dots & \mathbf{I}^{a_{J,L}} \end{bmatrix}, \tag{1}$$

where L and J denote the number of block columns and block rows, respectively, and $\mathbf{I}^{a_{j,l}}$, of size $z \times z$, is formed by cyclically shifting the columns of an identity matrix to the right by $a_{j,l}$ ($0 \leq a_{j,l} < z$) times. The codeword length is $L \times z$ and the code rate R is lower bounded by $R \geq 1 - J/L$.

B. Decoding Algorithm

In the realm of LDPC decoders, the sum-product algorithm (SPA) is the most widely used decoding scheme and is also the essence of many other variants such as min-sum (MS) decoding. Also known as belief propagation (BP), SPA carries out

Algorithm 1 Equivalent Layered Decoding for the QC-LDPC code in (1) using SPA

Initialization
set $\lambda_n = \beta_{mn} = 2y_n/\sigma^2$ and $\alpha_{mn} = 0,\ \forall m, n$
Iteration
for iteration $i = 1, 2, \cdots, I$ **do**
for layer $j = 1, 2, \cdots, J$ **do**
for group $g = 1, 2, \cdots, G$ **do**
for check node $m = (j-1)z + (g-1)\frac{z}{G} + 1, (j-1)z + (g-1)\frac{z}{G} + 2, \cdots, (j-1)z + g\frac{z}{G}$ **do**
for $\forall n \in \mathcal{N}(m)$ **do**
update the C2V messages α_{mn} in the current layer using

$$\alpha_{mn} = 2\tanh^{-1}\left(\prod_{n' \in \mathcal{N}(m)\backslash n} \tanh\left(\frac{\beta_{mn'}}{2}\right)\right) \quad (2)$$

update the APP β_n of the variable nodes using

$$\beta_n = \lambda_n + \sum_{m' \in \mathcal{M}(n)} \alpha_{m'n} \quad (3)$$

for check node $m'' \in \mathcal{M}(n)$ AND in the next layer, i.e., layer $(j+1) \mod J$ **do**
update the V2C messages $\beta_{m''n}$ in the next layer using

$$\beta_{m''n} = \beta_n - \alpha_{m''n} \quad (4)$$

end for
end for
end for
end for
end for
end for
Decision
set $\hat{x}_n = 0$ if $\beta_n \geq 0$, or set $\hat{x}_n = 1$ if $\beta_n < 0$, $\forall n$.

iterative message-passing processes to achieve convergence for all constrained variables. To speed up this convergence, a layered decoding schedule making an immediate use of the updated messages can also be employed.

Considering the parity-check matrix $\mathbf{H}$ in (1), we denote the check-to-variable (C2V) message with α_{mn} and variable-to-check (V2C) message with β_{mn}, where $m = 1, 2, \cdots, M$ and $n = 1, 2, \cdots, N$. We further define $\mathcal{N}(m)$ as the set of variable nodes connected to check node m; $\mathcal{M}(n)$ as the set of check nodes connected to variable node n; $\mathcal{N}(m)\backslash n$ as the set $\mathcal{N}(m)$ excluding variable node n; and $\mathcal{M}(n)\backslash m$ as the set $\mathcal{M}(n)$ excluding check node m. For each layer (i.e., block row), z check nodes can be divided into G groups that are processed in a sequential order. Finally we denote λ_n as the channel message (CM) and β_n as the a posteriori probability (APP) for variable n. The layered SPA decoding is summarized in Algorithm 1. In our design a binary input-additive white Gaussian noise (BI-AWGN) channel is assumed and hence λ_n is initialized as $2y_n/\sigma^2$, where y_n is the received value and σ is the standard deviation of noise samples. We compute β_n in the log-likelihood ratio (LLR) form and therefore its sign bit exactly symbolizes the bit state.

C. Modified SPA for Implementation

Due to the nonlinearity of (2), SPA cannot be implemented with simple circuits unless some algorithmic adjustments are made. An adder-based architecture is implemented by introducing extra log-domain mapping and demapping units [29]. The check-node processing function is then

$$\text{sign}\left(\alpha_{mn}\right) = \prod_{n' \in \mathcal{N}(m)\backslash n} \text{sign}\left(\beta_{mn'}\right) \quad (5)$$

$$|\alpha_{mn}| = \phi^{-1}\left[\sum_{n' \in \mathcal{N}(m)} \phi\left(|\beta_{mn'}|\right) - \phi\left(|\beta_{mn}|\right)\right] \quad (6)$$

where

$$\phi(x) = \phi^{-1}(x) = -\log\left(\tanh\frac{x}{2}\right). \quad (7)$$

III. Cyclically-Coupled QC-LDPC Codes

As demonstrated by LDPCCCs, using spatial coupling is an effective way of improving the loose constraints suffered by short codes such that an enhanced decoding capability is achievable [10]. In this paper we propose a structured subclass of QC-LDPC codes which are constructed by cyclically coupling a number of QC-LDPC sub-codes. Similar to QC-LDPCCCs, the proposed cyclically-coupled QC-LDPC (CC-QC-LDPC) block codes can be constructed from an original QC-LDPC sub-code. First, we divide the parity-check matrix of a QC-LDPC sub-code into three portions $\mathbf{H}_l$, $\mathbf{H}_m$ and $\mathbf{H}_r$. Hence, the parity-check matrix of the sub-code can be rewritten from (1) to

$$\mathbf{H_s} = \begin{bmatrix}\mathbf{H}_l & \mathbf{H}_m & \mathbf{H}_r\end{bmatrix}. \quad (8)$$

By cyclically-coupling K such sub-codes, we form a CC-QC-LDPC code whose parity-check matrix is represented as

$$\mathbf{H_{cc}} = \begin{bmatrix} \mathbf{H}_l^1 & \mathbf{H}_m^1 & \mathbf{H}_r^1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{H}_l^2 & \mathbf{H}_m^2 & \mathbf{H}_r^2 & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_l^3 & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{H}_r^{K-1} & \mathbf{0} \\ \mathbf{H}_r^K & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_l^K & \mathbf{H}_m^K \end{bmatrix}, \quad (9)$$

Note that in this paper, we assume that all the sub-codes are identical. In general, they can be different.

We assume that each sub-code $\mathbf{H_s}$ consists of $J \times L$ sub-matrices each of size $z \times z$. We use the following notation to relate adjacent sub-codes. We denote the sub-code to the cyclic left of the current sub-code as the preceding sub-code, and the one to its cyclic right as the following sub-code. For example, if the current sub-code is $[\mathbf{H}_l^3\ \mathbf{H}_m^3\ \mathbf{H}_r^3]$, the preceding sub-code will be $[\mathbf{H}_l^2\ \mathbf{H}_m^2\ \mathbf{H}_r^2]$, and the following sub-code will be $[\mathbf{H}_l^4\ \mathbf{H}_m^4\ \mathbf{H}_r^4]$. Referring to (9), we say that $\mathbf{H}_l$ of the current sub-code couples with $\mathbf{H}_r$ of the preceding sub-code because $\mathbf{H}_l$ and $\mathbf{H}_r$ correspond to the same block columns in (9) but they come from different sub-codes. Hence, we also can say that $\mathbf{H}_r$ of the current sub-code couples with $\mathbf{H}_l$ of the following sub-code. Since $\mathbf{H}_l$ and $\mathbf{H}_r$ correspond to the same block columns in (9), they must have the same number of block columns. We call the number of such coupled block columns as the coupling degree and denote it by W. In other words, both $\mathbf{H}_l$ and $\mathbf{H}_r$ consist of $J \times W$ sub-matrices and $\mathbf{H}_m$ consists of $J \times (L - 2W)$ sub-matrices. Then the rate of the CC-QC-LDPC code is easily shown to be lower bounded by $1 - \frac{J}{L-W}$. Note also that the check-node degree of the CC-QC-LDPC code is the same as those of the sub-codes. In other words, coupling the QC-LDPC sub-codes to form a CC-QC-LDPC code will not increase the check-node degree.

In summary, a CC-QC-LDPC code is constructed by cyclically coupling K QC-LDPC sub-codes. Moreover, the parity-check matrix corresponding to the original QC-LDPC sub-code consists of J block rows and L block columns, and each block is a circulant matrix of size $z \times z$. Among the L block columns, W of them (i.e., $\mathbf{H}_l$) share the same variable nodes with the preceding sub-code, another W of them (i.e., $\mathbf{H}_r$) share the same variable nodes with the following sub-code, and the remaining $L - 2W$ of them (i.e., $\mathbf{H}_m$) do not couple with any variable nodes of other sub-codes.

The structural features of CC-QC-LDPC codes shall benefit the corresponding decoders in a hardware sense compared with other codes having equivalent decoding capabilities. Firstly, the overall CC-QC-LDPC decoder architecture can be assembled from subordinate decoders of smaller size to reduce the complexity. In our architecture, the decoder for $\mathbf{H_s}$ after modification is used as a sub-decoder of $\mathbf{H_{cc}}$. Secondly, the decomposable structure endows the decoding with a chance of high parallelism, which is especially appreciated by circuits and critically relevant to the throughput. Thirdly, when the sub-codes are identical, some decoding circuits (e.g., multiplexers and controllers) can be shared and hence the hardware efficiency is improved. We make use of these advantages in our proposed decoder architecture, the details of which are presented in the following sections.

Note that CC-QC-LDPC codes differ from QC-LDPCCCs in several aspects. Firstly, a CC-QC-LDPC code is constructed by combining variable nodes of consecutive block codes. For example, $\mathbf{H}_r^2$ and $\mathbf{H}_l^3$ correspond to the same variable nodes. Secondly, in a CC-QC-LDPC code, the coupling between consecutive block codes is performed in a cyclic manner, i.e., the first block couples with the second one, the second one with the third one, ..., and the last one with the first one (similar to tail-biting). Thirdly, part of the original QC-LDPC block code, i.e., $\mathbf{H}_m$, does not couple with other block codes.

IV. CC-QC-LDPC Decoder Architecture

A. Overall Architecture

The overall architecture of our proposed CC-QC-LDPC decoder is illustrated in Fig. 1. In this architecture, there are K identical sub-decoders and a global controller. Moreover, the sub-decoders are connected in a cyclic way, similar to the coupling among the corresponding sub-codes. The components in each sub-decoder can be classified into the following three main categories.

1) The first category includes the computational logics (and registers for pipelining) consisting of (i) check-node processor (CNP), (ii) variable-node processor (VNP), and (iii) format converters (2's complement to sign-magnitude (C2S) and its inverse (S2C)). The function of these logics is to update the edge messages between the variable nodes (VNs) and check nodes (CNs). Since we adopt a layered decoding, we called this category of components a layered decoder.
2) The second category includes the random access memories (RAMs) which store the updated messages and channel messages. Each RAM contains memory locations that can be accessed easily based on their independent addresses.
3) The third category includes a switch network that connects the layered decoder and RAMs. The switch network is implemented with multiplexers and controlled by a controller.

The V2C messages are stored in sign-magnitude formats. These V2C messages are used to compute new C2V messages by the CNP that executes (2). Then other existing C2V messages and channel messages can be retrieved from the corresponding RAMs and added to the newly updated C2V messages by the VNP. The purpose is to evaluate the new V2C messages using (4) and the APP using (3). For convenience, all the updated C2V messages are converted into 2's complement format in advance. The newly generated V2C messages have to be first converted into sign-magnitude format by the C2S converters before being stored into the RAMs whereas the updated C2V messages are stored without further conversion. All the updated messages are then ready to be used for the decoding of the following layer in the layered decoder. After a pre-defined number of iterations has been performed, the VNP shall make decisions on the received bits based on the APP values (3) which have been updated according to the C2V messages and channel messages.

The layered decoder adopts a parallel structure that simultaneously processes a group of check nodes and their associated variable nodes. During the decoding of Layer j, the groups denoted by $g = 1, 2, \ldots, G$ are sequentially processed. When the memory locations of the messages have been appropriately allocated, the different groups can simply be controlled by counters that are incremented by one at each clock cycle. Another function of the controller is to switch the connections when the last group of Layer j has been processed and the first group of Layer $j + 1$ (or Layer 1 if $j = J$) is to begin.

Referring to the structure of a CC-QC-LDPC code, the C2V messages corresponding to $\mathbf{H}_l$ and the V2C messages corresponding to $\mathbf{H}_r$ are updated by the current sub-decoder. However, these two types of updated messages are to be used by the preceding sub-decoder and the following sub-decoder, respectively. Hence, these messages are stored with particular storage arrangements such that memory redundancies can be minimized. Details of the message storage will be discussed in Section IV-C.

B. LUT-based CNP with Parallel Routing

A CNP based on look-up tables (LUTs) is applied in our architecture to realize the quantized SPA [12]. In other words, the CNP consists of functional units performing $O(I_i, I_j) = \mathcal{Q}\left\{2\tanh^{-1}\left(\tanh\frac{I_i}{2}\tanh\frac{I_j}{2}\right)\right\}$, where $\mathcal{Q}$ denotes quantization. Then the C2V updating formula (2) can be completed by repeatedly using such LUTs. To reduce the gross usage of the LUTs, a recursive structure has been proposed in [30]. At first, the partial V2C products (in hyperbolic tangent domain) can be bidirectionally achieved with two sets of cascaded LUTs, each LUT taking in an untapped V2C input and the up-to-date partial product. Then the resultant C2V message to each

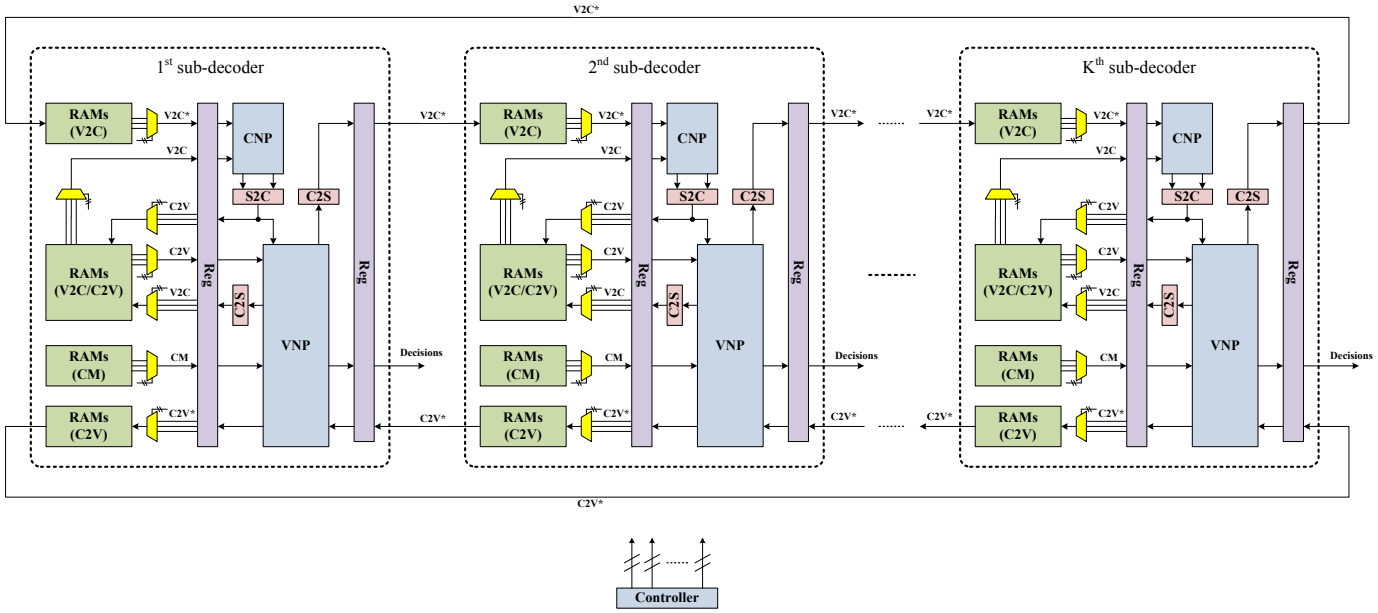


Fig. 1. The overall architecture of the CC-QC-LDPC decoder. The messages passing between adjacent sub-decoders are marked by asterisks.

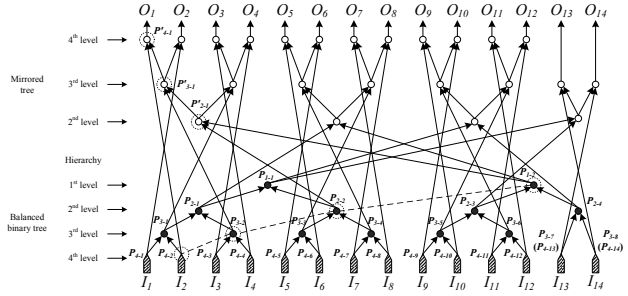


Fig. 2. The parallel structure of LUTs in the CNP with $d_c = 14$. $I_1 \sim I_{14}$ and $O_1 \sim O_{14}$ are the V2C inputs and C2V outputs, respectively. Solid/hollow nodes represent the LUTs and the arrows represent the connections among them. P_{i-j} denotes the partial product of V2C messages, and that with a prime denotes the partial product of a mirror node. Dashed circles and lines are, respectively, the LUTs and valid connections related to the computation of O_1.

variable node can be computed by feeding another LUT with two partial V2C products, which come from different sets of LUTs. For a check node with degree d_c, this type of CNP consumes a total of $3(d_c - 2)$ LUTs.

One notable drawback of this structure is that the cascaded LUTs may cause a significant delay, which equals $(d_c - 2)\tau$ where τ is the average delay caused by each LUT. Here we reroute these LUTs to make them operating in parallel. We denote the set of variable nodes connected to a degree-d_c check node c by $\{I_i : 1 \le i \le d_c\}$. Our target design can be derived by dividing these variable nodes in a hierarchical manner. First, we divide $\{I_i\}$ into 2 sub-sets. Next we consider a specific variable node $v \in \{I_i\}$ which must fall into one of these two sets. We use $\mathcal{S}^v/\mathcal{S}^{\bar{v}}$ to denote the sub-set that includes/excludes v. Then the C2V message passing from c to v is written as

$$\alpha_{cv} = 2\tanh^{-1}\left(\prod_{v' \in \mathcal{S}^v \setminus v} \tanh\left(\frac{\beta_{cv'}}{2}\right) \prod_{v'' \in \mathcal{S}^{\bar{v}}} \tanh\left(\frac{\beta_{cv''}}{2}\right)\right). \tag{10}$$

If we keep dividing the sub-set in which v exists, (10) can eventually be decomposed into

$$\alpha_{cv} = 2\tanh^{-1}\left(\prod_{1 \le t \le T} \prod_{v' \in \mathcal{S}_t^{\bar{v}}} \tanh\left(\frac{\beta_{cv'}}{2}\right)\right) \tag{11}$$

where T denotes the total number divisions made (or equivalently the number of levels) and $\mathcal{S}_t^{\bar{v}}$ represents the set of variable nodes separated from the set containing v at the tth division. Accordingly, we have $\mathcal{S}_1^{\bar{v}} = \mathcal{S}^{\bar{v}}$ and $\mathcal{S}_2^{\bar{v}} \bigcup \mathcal{S}_3^{\bar{v}} \cdots \bigcup \mathcal{S}_T^{\bar{v}} = \mathcal{S}^v \setminus v$.

As a natural representation, a binary tree is adopted to illustrate our division strategy. We consider an example in which the check-node degree equals 14, i.e., $d_c = 14$. Referring to Fig. 2, there are 14 variable nodes and $T = 4$ levels. In the first level, the variable nodes $I_1 \sim I_8$ are assigned to the sub-set P_{1-1} while the remaining nodes $I_9 \sim I_{14}$ are assigned to the sub-set P_{1-2}. Note that we allow the larger sub-set (i.e., P_{1-1}) to contain 8 elements so that the number of hierarchical levels can be minimized. Next, P_{1-1} is further divided into P_{2-1} containing $I_1 \sim I_4$ and P_{2-2} containing $I_5 \sim I_8$; while P_{1-2} is divided into P_{2-3} containing $I_9 \sim I_{12}$ and P_{2-4} containing I_{13} and I_{14}. At this level the size of every sub-set is a power of 2 and hence equal divisions can be executed at the subsequent levels, as shown in Fig. 2. Note that we can decompose P_{2-4} at either the third level or the fourth level, and hence both I_{13} and I_{14} can have two different labels. In Fig. 2, each filled circle/node represents an LUT that outputs the partial product of a certain sub-set. These filled nodes reflect the division strategy and also make up a binary-tree structure. Note that the root node (at the 0-th level) is omitted because there is no need to compute the overall product of the variable set $\{I_i\}$.

After dividing the set $\{I_i\}$ into different sub-sets, the partial products of these sub-sets (at different levels) need to be multiplied together according to (11) in order to compute all the C2V messages. Without loss of generality, the output

(denoted by O_1) to the first variable node, which is computed using $I_2 \sim I_{14}$, is considered. According to the above divisions, the required partial products are P_{1-2}, P_{2-2}, P_{3-2}, and P_{4-2} (refer to nodes along the dashed line in Fig. 2). We will complete the multiplication among these partial products using three cascaded LUTs at three levels. As illustrated in Fig. 2, P_{1-2} and P_{2-2} are used to compute P'_{2-1}, which becomes the partial product of $I_5 \sim I_{14}$. Next, P'_{2-1} and P_{3-2} (the product of I_3 and I_4) are used to compute P'_{3-1}, which becomes the partial product of $I_3 \sim I_{14}$. Finally, P_{4-2} (i.e., I_2) combines with P'_{3-1} to compute the result P'_{4-1} (i.e., O_1), which is the partial product of $I_2 \sim I_{14}$. After taking all outputs into account, we minimize the total number of LUTs required by re-using all the partial products as much as possible. For example, P'_{3-1} can be combined with each of the elements (I_1 and I_2) in P_{3-1} to produce two outputs (O_2 and O_1). Hence, the path for producing O_2 differs by only one segment (LUT) compared with that for producing O_1.

As shown in Fig. 2, another LUT tree formed by the hollow circles/nodes and mirroring the tree formed by the filled circles (from the second level downwards) has been constructed. In this tree, the output of each hollow node will be shared by the subsets/variable nodes of the subset corresponding to its symmetric node. For example, the output of P'_{2-1} will combine with the respective subsets of P_{2-1} (i.e., P_{3-1} and P_{3-2}); and the output of P'_{3-1} will combine with the respective variable nodes of P_{3-1} (i.e., P_{4-1} and P_{4-2}).

Intuitively, a fewer number of levels in the (division) hierarchy implies a smaller delay of the critical path. It can be easily shown that the number of levels T is bounded by $T \geq \lceil \log_2 d_c \rceil$. Accordingly, the delay of the CNP can be minimized to $2\left(\lceil \log_2 d_c \rceil - 1\right)\tau$ with our proposed structure. Another observation is that a total of $3(d_c - 2)$ LUTs is employed and is the same as that required in the traditional structure. To construct our proposed symmetric-tree structure in a systematic way, we can make use of the following steps.

1) Forward Step: Place $d_c - 2$ LUTs in parallel to form a balanced binary tree so that the minimum number of levels is achieved. In this tree, the inputs of each node (LUT) are the outputs of its children (LUTs) or the leaves corresponding to the V2C inputs of the CNP.
2) Backward Step: Construct the mirror of the balanced binary tree generated in the Forward Step from the second level onwards. To every node at the second level of the mirrored tree, the two inputs come from the outputs of (i) the sibling of the corresponding node in the original balanced binary tree and (ii) the sibling of the parent of the corresponding node in the original balanced binary tree. For example, we consider P'_{2-1} at the second level of the mirrored tree. The corresponding node in the original balanced binary tree is P_{2-1}. Hence, the sibling of the corresponding node is P_{2-2} and the sibling of the parent of the corresponding node is P_{1-2}. The outputs of these two nodes therefore become inputs of P'_{2-1}.

C. Memory Arrangement

In our decoder, the message storage is fulfilled by RAMs that have been organized to fit the code structure. In each sub-code, the z check nodes forming one layer are connected to no common variable nodes. This characteristic allows the z check nodes to be split into G distinct groups such that z/G check nodes can be processed in parallel. As a result, a total of z/G RAMs are required to store the edge or channel messages corresponding to each sub-matrix of the sub-code[2] and each RAM should contain G entries.

Recall that for each sub-code of the CC-QC-LDPC code in (9), there is a corresponding sub-decoder in Fig. 1. Furthermore, each sub-code is divided into three portions, i.e., $\mathbf{H}_l$, $\mathbf{H}_m$ and $\mathbf{H}_r$. We consider the k-th sub-code ($k = 1, 2, \ldots, K$) and its corresponding sub-decoder. The sub-decoder is responsible for updating all the C2V messages of the sub-code. The variable nodes corresponding to $\mathbf{H}_m$ of the k-th sub-code are not shared by other sub-codes. Thus, these edge messages can only be updated by the k-th sub-decoder. However, variable nodes corresponding to $\mathbf{H}_l$ of the k-th sub-code are common with those corresponding to $\mathbf{H}_r$ of the $(k-1)$-th sub-code; and variable nodes corresponding to $\mathbf{H}_r$ of the k-th sub-code are common with those corresponding to $\mathbf{H}_l$ of the $(k+1)$-th sub-code. As a result, we can choose using the k-th sub-decoder to update the V2C messages corresponding to either $\mathbf{H}_l$ or $\mathbf{H}_r$ of the k-th sub-code, leaving the other V2C messages to be updated by the adjacent sub-decoder. In our design shown in Fig. 1, the k-th sub-decoder is responsible for updating the V2C messages corresponding to $\mathbf{H}_r$ of the k-th sub-code, leaving the V2C messages corresponding to $\mathbf{H}_l$ of the k-th sub-code to be updated by the $(k-1)$-th sub-coder. Hence, the k-th sub-decoder is responsible for updating (i) the C2V messages of the k-th sub-code; (ii) the V2C messages corresponding to $\mathbf{H}_m$ and $\mathbf{H}_r$ of the k-th sub-code; and (iii) the V2C messages corresponding to $\mathbf{H}_l$ of the $(k+1)$-th sub-code.

Given the behavioral differences of the messages under different sub-code partitions, the RAMs are classified into four different categories and storage schemes. They are described as follows.

1) The first category of RAMs performs a dynamic storage where the V2C and C2V messages corresponding to $[\mathbf{H}_m\ \mathbf{H}_r]$ of each sub-code, are alternately stored. During the decoding of Layer j ($j = 1, 2 \ldots, J$) of the k-th sub-code, only V2C messages of that layer and C2V messages of the other layers are required. Consequently, we can save the memory by assigning only one memory location for each variable-check edge. After a V2C message has been accessed by the layered decoder, the corresponding memory location will be immediately overwritten by the updated C2V message of the same variable-check edge. Each C2V message, moreover, will be accessed $J-1$ times in each decoding iteration and shall be replaced by the V2C message after the last use during the decoding of Layer $j-1$

[2]The channel messages can be considered as an additional layer of check nodes in which every sub-matrix is the identity matrix.

(or Layer J if $j = 1$). Numerically, the total number of RAMs in the first category for each sub-decoder is $J(L-W)z/G$.

2) The second category of RAMs are dedicated to storing the C2V messages corresponding to $\mathbf{H}_l$. These C2V messages, while updated by the current sub-decoder, will be used in the V2C and APP message-updating process of the preceding sub-decoder. In consequence, each RAM is continually read by the $(k-1)$-th sub-decoder (or K-th sub-decoder if $k = 1$) but written only when a specific layer of the k-th sub-code is processed. The total number of RAMs in this category for each sub-decoder is JWz/G.
3) The third category of RAMs are dedicated to storing the V2C messages corresponding to $\mathbf{H}_l$. As opposed to the second category, these messages are produced by the preceding sub-decoder and are used by the current sub-decoder to update its C2V messages. Among the C2V messages (i.e., corresponding to $\mathbf{H}_l$, $\mathbf{H}_m$ and $\mathbf{H}_r$) updated by the current sub-decoder, moreover, those corresponding to $\mathbf{H}_l$ will be returned to and used by the preceding sub-decoder. Since all sub-decoders are operating at the same time, the preceding sub-decoder will be writing V2C messages to the RAMs while the current sub-decoder is reading them. To ensure that there are no conflicts occurring when the sub-decoders are writing to/reading from the third category of RAMs, we allocate two sets of RAMs to alternately and respectively convey the V2C messages for the odd and even layers of each sub-code. Referring to Fig. 3, when Set 1 of RAMs is being accessed for the decoding of Layer j in the k-th sub-decoder, Set 2 of RAMs is written by the $(k-1)$-th sub-decoder (or K-th sub-decoder if $k = 1$) with the V2C messages for Layer $j+1$ (or Layer 1 if $j = J$) of the kth sub-code. Similarly, when Set 2 of RAMs is being accessed, Set 1 of RAMs is being written. Using this method, the memory efficiency is optimal in terms of the RAM usage. Altogether, there are $2Wz/G$ such RAMs required for each sub-decoder.
4) Distinct from the previous categories, the last category of RAMs stores the channel messages. During the decoding process, the channel messages are kept unchanged and continually output to the layered decoder for updating the V2C and APP messages. After the decoding of the current codeword has been completed, these RAMs are loaded with a new block of channel messages for the decoding of the next codeword. Totally, there are $(L-W)z/G$ RAMs in this category for each sub-decoder.

To summarize the above discussions, a total of $[(J+1)L+W]z/G$ RAMs each containing G entries are required for each sub-decoder. Since there are K sub-decoders, the overall memory size of our proposed decoder architecture equals $K[(J+1)L+W]zd$ bits where d is the width of each message in binary format.

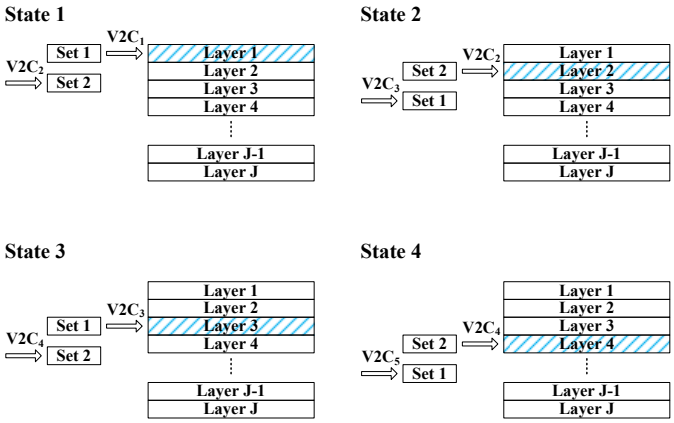


Fig. 3. Two sets of V2C-dedicated RAMs alternate to serve the odd or even layers of a sub-code. $\mathrm{V2C}_j$ is the messages from common (with the preceding previous sub-code assumed) variable nodes to the check nodes of Layer j.

D. Switch Network

In each sub-decoder, the layered decoder and the RAMs need to be connected perfectly in order to complete the processing of the J layers of the corresponding sub-code. To reduce the complexity, we make use of a network that switches layer-by-layer to correctly arrange the messages passed between the layered decoder and the RAMs. Specifically, we use multiplexers to fulfil this task. Denoting RAM(i) as one or more RAMs in the ith category as specified in Section IV-C, each input port selects data from P possible output ports where P is given as follows.

1) From the layered decoder to RAM(1), $P = 2$.
2) From the layered decoder to RAM(2), $P = 1$.
3) From the layered decoder to RAM(3), $P = J/2$.
4) From RAM(1) to the layered decoder, $P = J$.
5) From RAM(2) to the layered decoder, $P = J$.
6) From RAM(3) to the layered decoder, $P = 2$.
7) From RAM(4) to the layered decoder, $P = J$.

Note that the worst scenarios are mentioned here and P can be much smaller for a certain configuration or a specific code. For example, if $G = z$, there will be a fixed connection from RAM(4) to the layered decoder, i.e. $P = 1$.

There are also multiplexers connecting the RAMs and the address counters because the initial addresses vary for the decoding of different layers. To control the read and write addresses of the RAMs, G modulo-G counters with strong fan-out capabilities, each with a different output number, are needed. For the read/write address port of each RAM selecting output numbers from P counters, the following cases are concluded:

1) read operation of RAM(1), $P = J$;
2) read operation of RAM(2), $P = J$;
3) read operation of RAM(3), $P = 1$;
4) read operation of RAM(4), $P = J$;
5) write operation of RAM(1), $P = 2$;
6) write operation of RAM(2), $P = 1$;
7) write operation of RAM(3), $P = J/2$.

Note that the conclusions of the multiplexing cases are consistent with the arrangement of the RAMs.

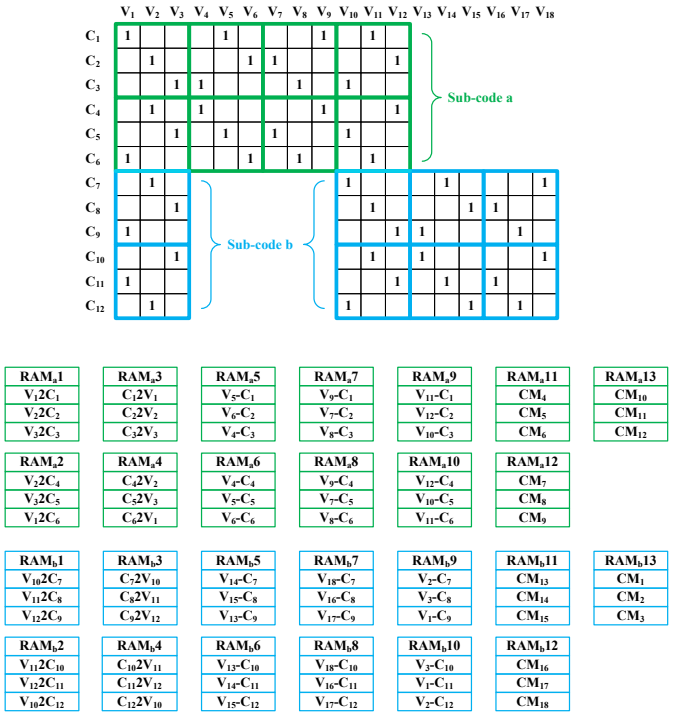


Fig. 4. A CC-QC-LDPC code with $K = 2$, $L = 4$, $J = 2$, $W = 1$ and $z = G = 3$.

E. Example

In this subsection, we make use of an example, where $K = 2$, $L = 4$, $J = 2$, $W = 1$ and $z = G = 3$, to elaborate the decoding process of the proposed decoder. As shown in Fig. 4, the code is constructed by two sub-codes, namely a and b, and therefore two sub-decoders are used in the decoder. Each sub-decoder, moreover, contains 13 RAMs: RAM1 and RAM2 for V2C messages corresponding to leftmost block column (i.e., $\mathbf{H}_l$); RAM3 and RAM4 for C2V messages corresponding to same block column; RAM5 to RAM10 for V2C/C2V messages corresponding to the other block columns (i.e., $\mathbf{H}_m$ and $\mathbf{H}_r$); and RAM11 to RAM13 for the channel messages corresponding to $\mathbf{H}_m$ and $\mathbf{H}_r$. We assume a discordant schedule as an example, i.e., when one sub-decoder is processing its first layer, the other sub-decoder is dealing with its second layer. To clarify the relevant timing terminology, the simultaneous decoding of the two layers described above shall be referred to as a state and each time slot within a state is called a stage. In our case, the decoding procedure of one complete iteration is comprised of two states and each state is comprised of three stages. Fig. 5 and Fig. 9[3] provide the step-by-step details of this process.

Referring to Fig. 5, the messages corresponding to the first layer of sub-code a and those corresponding to the second layer of sub-code b are updated in sub-decoders a and b, respectively.

1) At the beginning of State 1, the initial contents of $\{\text{RAM}_a5, \text{RAM}_a7, \text{RAM}_a9\}$ are V2C messages. In the first stage when check node C_1 is activated, the first entries of these RAMs are accessed for the messages $\{\text{V}_5\text{2C}_1, \text{V}_9\text{2C}_1, \text{V}_{11}\text{2C}_1\}$. They, together with $\text{V}_1\text{2C}_1$ at the first entry of RAM_a1, are fed into the CNP[4]. The updated C2V messages $\{\text{C}_1\text{2V}_5, \text{C}_1\text{2V}_9, \text{C}_1\text{2V}_{11}\}$ will overwrite the original V2C contents in the first entries of $\{\text{RAM}_a5, \text{RAM}_a7, \text{RAM}_a9\}$ and will be used in State 2; whereas the updated C2V message $\text{C}_1\text{2V}_1$ will replace the original one stored in the first entry of RAM_a3 and will be instantly read by the sub-decoder b in the next (second) stage.
2) With the updated C2V messages $\{\text{C}_1\text{2V}_5, \text{C}_1\text{2V}_9, \text{C}_1\text{2V}_{11}\}$, sub-decoder a further computes the V2C messages corresponding to variable nodes V_5, V_9 and V_{11}. The other C2V messages $\{\text{C}_5\text{2V}_5, \text{C}_4\text{2V}_9, \text{C}_6\text{2V}_{11}\}$ at the second, first and third entries of $\{\text{RAM}_a6, \text{RAM}_a8, \text{RAM}_a10\}$ are therefore retrieved. Moreover, sub-decoder b provides C2V messages concerning V_{11}. They are $\text{C}_8\text{2V}_{11}$ and $\text{C}_{10}\text{2V}_{11}$ located at the second and first entries of RAM_b3 and RAM_b4, respectively. (This shows the coupling effect between the sub-codes a and b.) Together with the channel messages $\{\text{CM}_5, \text{CM}_9, \text{CM}_{11}\}$ retrieved from $\{\text{RAM}_a11, \text{RAM}_a12, \text{RAM}_a13\}$, sub-decoder a updates the V2C messages $\{\text{V}_5\text{2C}_5, \text{V}_9\text{2C}_4, \text{V}_{11}\text{2C}_6, \text{V}_{11}\text{2C}_8\}$. The V2C messages $\{\text{V}_5\text{2C}_5, \text{V}_9\text{2C}_4, \text{V}_{11}\text{2C}_6\}$ replace the original C2V messages $\{\text{C}_5\text{2V}_5, \text{C}_4\text{2V}_9, \text{C}_6\text{2V}_{11}\}$ stored in $\{\text{RAM}_a6, \text{RAM}_a8, \text{RAM}_a10\}$; whereas $\text{V}_{11}\text{2C}_8$ is stored in the second entry of RAM_b1 in sub-decoder b. As shown in Fig. 9, in State 2 during which sub-decoder b decodes Layer 1, the contents in RAM_b1 will be used while the contents in RAM_b2 will be updated.
3) Referring to the first stage of Fig. 5, sub-decoder b updates the messages related to check node C_{10} and its associated variable nodes. $\{\text{V}_{11}\text{2C}_{10}, \text{V}_{13}\text{2C}_{10}, \text{V}_{18}\text{2C}_{10}, \text{V}_3\text{2C}_{10}\}$ are read from the first entries of $\{\text{RAM}_b2, \text{RAM}_b6, \text{RAM}_b8, \text{RAM}_b10\}$ and the CNP then updates the corresponding C2V messages, i.e., $\{\text{C}_{10}\text{2V}_{11}, \text{C}_{10}\text{2V}_{13}, \text{C}_{10}\text{2V}_{18}, \text{C}_{10}\text{2V}_3\}$. $\text{C}_{10}\text{2V}_{11}$ is stored at the first entry of RAM_b4 while $\{\text{C}_{10}\text{2V}_{13}, \text{C}_{10}\text{2V}_{18}, \text{C}_{10}\text{2V}_3\}$ replace the original V2C messages stored in the first entries of $\{\text{RAM}_b6, \text{RAM}_b8, \text{RAM}_b10\}$.
4) Similar to decoder a, decoder b then retrieves other C2V messages ($\{\text{C}_9\text{2V}_{13}, \text{C}_7\text{2V}_{18}, \text{C}_8\text{2V}_3\}$ from $\{\text{RAM}_b5, \text{RAM}_b7, \text{RAM}_b9\}$ and $\{\text{C}_3\text{2V}_3, \text{C}_5\text{2V}_3\}$ from $\{\text{RAM}_a3, \text{RAM}_a4\}$ of decoder a) and channel messages ($\{\text{CM}_{13}, \text{CM}_{18}, \text{CM}_3\}$ from $\{\text{RAM}_b11, \text{RAM}_b12, \text{RAM}_b13\}$). Using the updated C2V messages in the previous step, decoder b updates the V2C messages $\{\text{V}_{13}\text{2C}_9, \text{V}_{18}\text{2C}_7, \text{V}_3\text{2C}_8, \text{V}_3\text{2C}_5\}$. $\{\text{V}_{13}\text{2C}_9, \text{V}_{18}\text{2C}_7, \text{V}_3\text{2C}_8\}$ replace the original C2V messages in $\{\text{RAM}_b5, \text{RAM}_b7, \text{RAM}_b9\}$ while $\text{V}_3\text{2C}_5$ is stored in RAM_a2 of decoder a. By analogy, the second and third stages of State 1 and the three stages of State 2 can be deduced. The descriptions are therefore omitted here.

[3]Due to lack of space, Fig. 9 is shown in the Supplementary Materials and is also available on the authors' website http://www.eie.polyu.edu.hk/∼encmlau/ccqcldpc_supp_mat.pdf.

[4]The design for a degree-4 CNP can be easily completed with the proposed method and the result can be found identical to the traditional structure.

Comparing Fig. 5 and Fig. 9, an address-hopping can be found at the end of each State and, according to the discussion in Section IV-D, is fulfilled by switching the connected address counters. Another observation is that {RAM5, RAM7, RAM9} and {RAM6, RAM8, RAM10}, both belonging to the first category of RAMs, keep swapping their roles as storage of V2C and/or C2V messages in successive states. We refer to such type of storage as dynamic storage.

In Fig. 6, the switch network between the RAMs and the layered decoders is illustrated. Since there are two decoding states, 2-to-1 multiplexers are exclusively used for data selections at all input ports (except for the channel messages because $G = z$). Each multiplexer is controlled by the State signal S_0 that is negated by the controller every time three decoding stages have been completed. Since $J = 2$, most multiplexers are paired up and each multiplexer in a pair delivers the message for one of the two layers. In this diagram, the network used for routing messages internal to the sub-decoder is shown in the lower part while that for routing messages external to the sub-decoder is shown in the upper part. The routing reveals how the sub-codes are cyclically related and meanwhile possess respective autonomies.

The above example can also be used to show the scalability of the proposed architecture. If the code length needs to be extended, we can simply insert replicas of the sub-decoders in a ring-shape manner as illustrated in Fig. 1. Details within each sub-decoder are left unchanged and so are the incoming controlling signals. Then statistics will be scaled up accordingly including memory, logics and throughput. Similarly, the decoder can also be scaled down by extracting sub-decoders out and splicing the remaining parts together.

V. Implementation and Performance

The proposed decoder has been implemented on an Altera Stratix IV EP4SE530H35C2 FPGA which provides $424,960$ Adaptive Look-Up Tables (ALUTs) and $10,624$ memory logic array blocks (MLABs). CC-QC-LDPC codes of rate $5/6$ are constructed with $K = 4$, $L = 28$, $J = 4$, $W = 4$ and sub-matrix sizes of $z = 128, 256, 512, 1024$. Assuming that 4-bit quantization[5] is used together with 10 decoding iterations (i.e., $I = 10$), Table I shows the implementation details of our CC-QC-LDPC decoders. For code A, B and C, each sub-decoder adopts a degree-8 parallelism but for code D both 8 and 16 parallelism degrees have been attempted. It is observed that there is not much difference in the complexities of the first four cases except that their memory sizes hold a linear dependence on z. For Code D, when the parallelism degree is increased from 8 to 16, the number of ALUTs and registers are about doubled but the number of memory bits remains the same.

All the implementations are evaluated at a normalized clock rate (100 MHz) for an evident comparison and the information throughput (T) is related to other parameters as $T = \frac{Kz(L-W)}{IJ(G+N_p)} \times R \times f$, where N_p is the number of pipelining stages, R is the code rate and f is the clock frequency. We have arranged four stages in our design, which are tasked with (1) reading messages from the memory, (2) updating the C2V messages, (3) updating the V2C messages, and (4) writing messages into memory.

In Table I, we also list the complexity of the best-error-performing LDPC convolutional code (LDPCCC) decoder in [30]. Firstly, a strong memory size contrast can be noticed. Numerically, even the largest CC-QC-LDPC decoder consumes only 13.4% (2,359,296 bits versus 17,558,528 bits) of the LDPCCC decoder memory. Next we compare the CC-QC-LDPC decoder of code D having degree-16 parallelism with the LDPCCC decoder. It is shown that with a lower combinational ALUTs and registers requirement and a substantially lower memory requirement, the CC-QC-LDPC decoder can achieve a 3.0 Gbps[6] throughput, i.e., 50% higher than that of the LDPCCC decoder. Note however that the total number of iterations are different for the CC-QC-LDPC decoder and LDPCCC decoder. One feature of the LDPCCC decoder in [30] needed to be mentioned is that it adopts a fully pipelining structure so that all the iterations are performed by a series of processors. Therefore, its hardware size is proportional to the iteration number while its throughput is constant. Moreover, the better the target BER, the higher the implementation complexity. In contrast, the proposed CC-QC-LDPC decoder can flexibly trade between the BER performance and throughput by adjusting I with a given complexity.

Note that we select the parameters $K = 4$, $L = 28$, $J = 4$, $W = 4$ such that our CC-QC-LDPC code achieves the same code rate (i.e., $5/6$) as the LDPCCC in [30]. For a given sub-code $\mathbf{H_s} = [\mathbf{H}_l \ \mathbf{H}_m \ \mathbf{H}_r]$, adjusting the coupling degree W has the following effects. Generally, the larger the number of coupling columns (i.e., W), the more hardware required. A larger overlapping region calls for a larger number of adders to implement the VNP. It also consumes more memory locations. It will also more likely improve the BER. However, the code rate (given by $1 - \frac{J}{L-W}$) is reduced and so is the information rate. So the coupling degree should be the result of tradeoff between error performance and hardware complexity.

Based on the FPGA simulation on code D with degree-16 parallelism, an experiment is conducted to collect the BER results of our proposed CC-QC-LDPC decoder. In our simulations, binary phase-shift-keying (BPSK) modulation and AWGN channels are applied and the number of decoding iterations I is assumed to be 10. As shown in Fig. 7, no error floor is observed above a BER of 10^{-14}. Moreover, at $E_b/N_0 = 3.5$ dB, we are able to decode all the received 1.14×10^{16} bits correctly. Such evidence suggests that error floor might be bounded by 10^{-16}, although a hundred times more bits are required to be simulated to ascertain the statistical significance of this bound. The error performance of this CC-QC-LDPC decoder is compared with that of the aforementioned LDPCCC decoder. Referring to Fig. 7, we can observe that there is only a 0.02 dB difference at a BER of 10^{-12} and the gap is narrowing as E_b/N_0 increases. Next, we investigate the gain of the CC-QC-LDPC code due to coupling. Fig. 7 plots the BER curve of the QC-LDPC code without

[5]A larger number of quantization bits results in a higher complexity of, namely, CNP, VNP, memory size, etc., but a better error performance [31].

[6]The maximum clock rate based on our implementation model is slightly over 110 MHz, and hence the possible highest throughput is 3.3 Gbps.

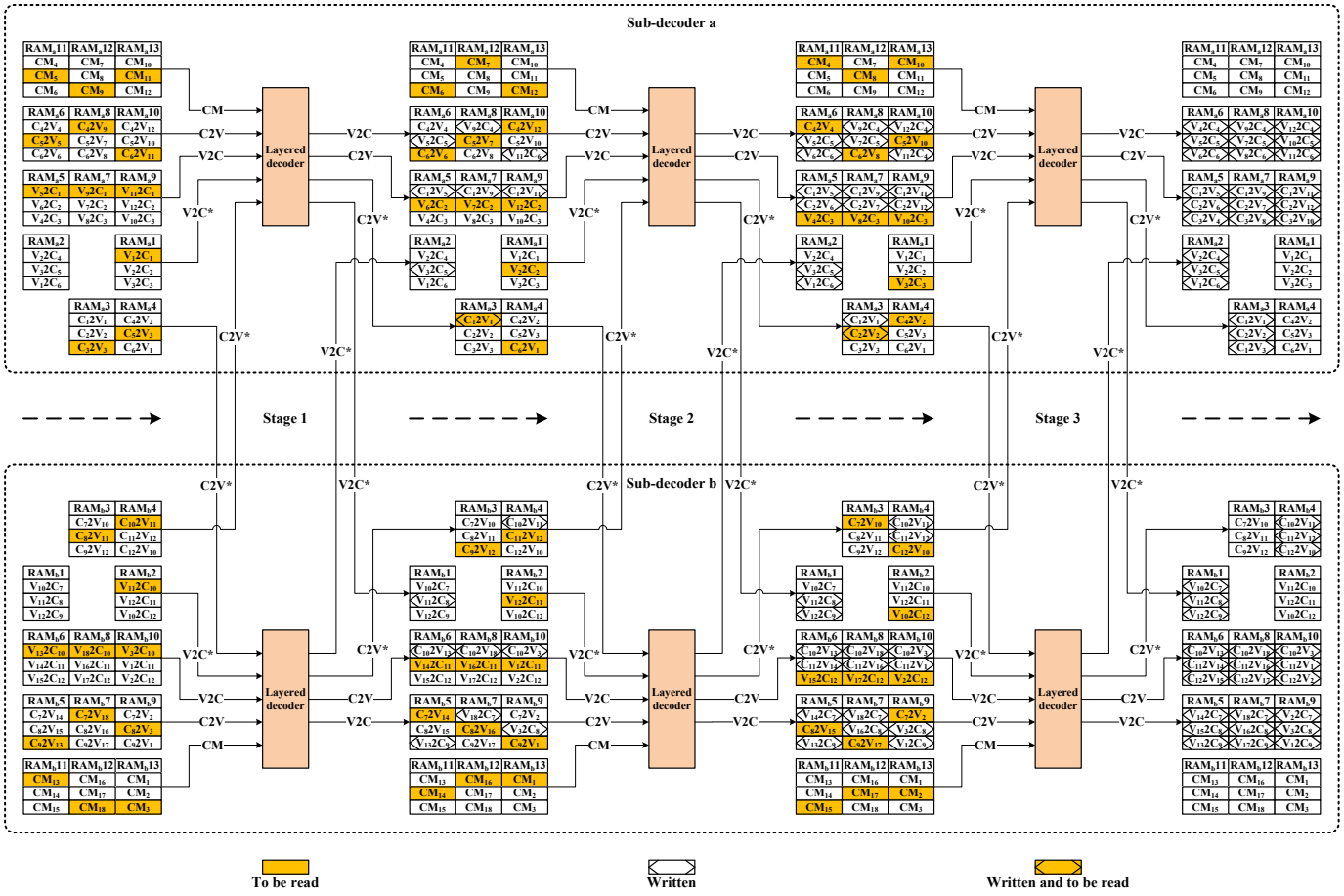


Fig. 5. State 1 of the decoding procedure in the example.

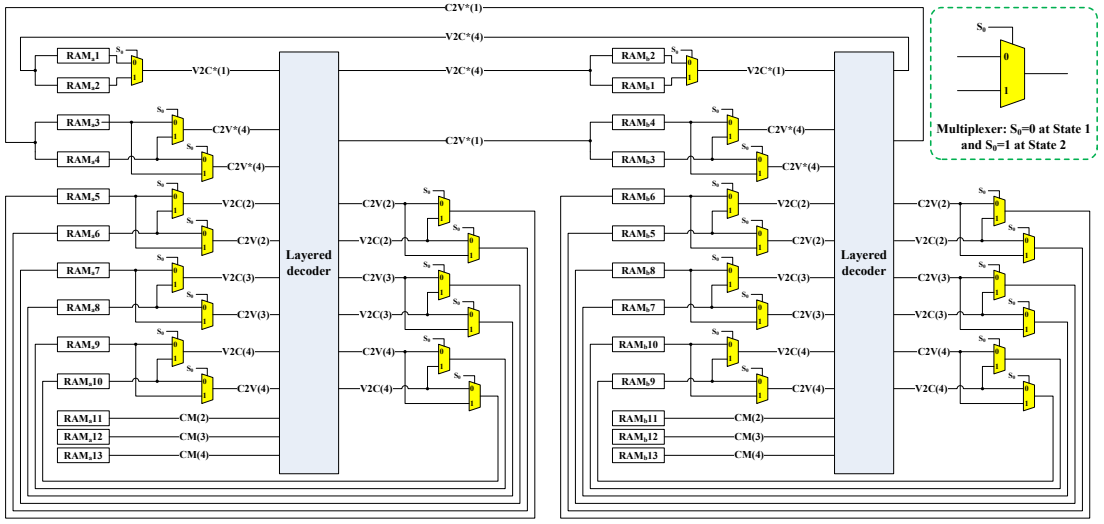


Fig. 6. The switch network for the decoder example. The numbers in parentheses indicate the order of block columns in the sub-code to which the messages correspond.

coupling (i.e., using only one sub-code in code D). We can observe that our proposed CC-QC-LDPC code outperforms the original QC-LDPC code (the one without coupling) by more than 0.25 dB at a BER of 10^{-7}. Furthermore, a larger gain is expected at a lower BER.

Lastly, we examine the performance of our proposed check-node processor (CNP) that implements quantized SPA (QSPA) based on LUTs. Two other CNPs, namely (i) approximate SPA (ASPA) based on mappers and adders and (ii) min-sum based on comparators, are also studied for comparisons. The BER performance of the QC-LDPC decoder is plotted in Fig. 8 when different types of CNPs are used. The floating-

TABLE I

HARDWARE INFORMATION OF THE DECODER IMPLEMENTATIONS. CODE A: $z = 128$; CODE B: $z = 256$; CODE C: $z = 512$; CODE D: $z = 1024$. ALL THE CC-QC-LDPC CODES ARE CONSTRUCTED BY FOUR 4×28 SUB-CODES WITH A COUPLING DEGREE OF 4. FOR COMPARISON, THE INFORMATION OF A LDPCCC DECODER IS IMPORTED FROM [30] (CODE 2-P). ALL DECODERS ARE DESIGNED UNDER 4-BIT QUANTIZATION.

	z	Stage No. G	Combinational ALUTs	Registers	Memory bits	Clock	Iteration No. I	Throughput (info bits)
Code A	128	16	66,285	43,798	294,912	100 MHz	10	1.55 Gbps
Code B	256	32	66,974	43,799	589,824	100 MHz	10	1.55 Gbps
Code C	512	64	67,878	43,800	1,179,648	100 MHz	10	1.55 Gbps
Code D	1024	128	70,324	43,801	2,359,296	100 MHz	10	1.55 Gbps
		64	134,170	87,575	2,359,296	100 MHz	10	3.00 Gbps
LDPCCC [30]	512	512	170,102	105,505	17,558,528	100 MHz	18	2.00 Gbps

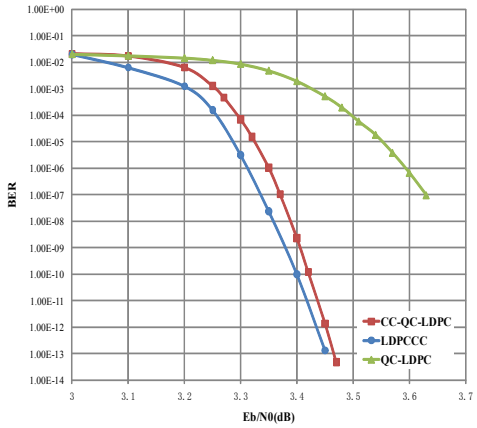


Fig. 7. The BER comparison of different decoders. The CC-QC-LDPC code is code D and the QC-LDPC code is the sub-code of code D. All the results are obtained from FPGA simulation under AWGN channels and 4-bit quantization.

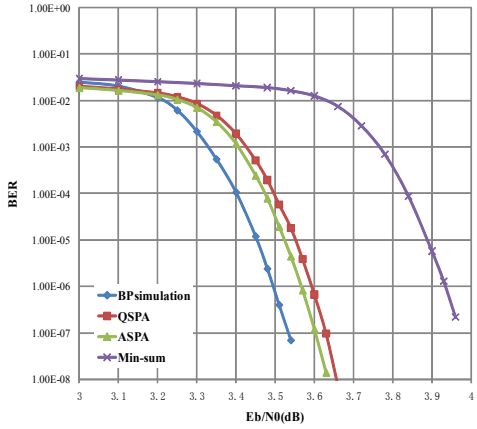


Fig. 8. The BER comparison of various decoders using different CNPs: QSPA using the LUTs, ASPA using mappers and adders, and min-sum using comparators. All the hardware simulations are under 4-bit quantization. BP is the result of computer-based simulation with double-precision floating-point data. The simulations are all based on the same QC-LDPC code with a check-node degree of 28.

point computer simulation result is also shown. With 4-bit quantization, the LUT-based architecture (QSPA) degrades by only 0.1 dB compared with the double-precision floating-point decoder at a BER of 10^{-7}. Also at this level, there is only a 0.03 dB difference between the ASPA decoder and the QSPA decoder. The min-sum decoder, however, is notably

TABLE II

COMPARISON OF THE CNPS USING DIFFERENT METHODS. THE RESULTS ARE COLLECTED BY SYNTHESIZING THE CNP AND A PARTITION WITH REGISTERED INPUTS AND OUTPUTS.

CNP	Combinational ALUTs	Maximum clock
Min-sum [15]	309	210 MHz
ASPA [29]	1068	140 MHz
QSPA (cascaded) [30]	274	80 MHz
QSPA (parallel)	277	220 MHz

seen as much worse than the other decoders. We further compare the complexity and clock frequency of these CNPs by implementing them as independent processors. Referring to Table II, we observe that the ASPA CNP, despite having the best BER performance, is much more complex than the others in terms of the number of ALUTs. As a matter of fact, the min-sum decoding can also be considered as LUT-based. Thus its usage of combinational ALUTs is close to QSPA's. This is also the reason why the proposed parallel structure can work for the min-sum decoder to achieve a possibly high clock rate[7]. More importantly, the delay improvement of the parallel structure from the cascade structure has been measured. By rearranging the LUTs, the clock frequency of the proposed LUT-based CNP has increased by over 140% without any degradations in the complexity and computation accuracy. Although the adder-based CNP (ASPA) has a parallel structure, its delay is still significant mainly because the output of its mapping unit that executes (7) has a larger width than the input. In summary, if a slight error degradation is tolerable, our proposed parallel LUT-based architecture is an optimal choice for implementing a reduced-complexity check-node processor.

VI. CONCLUSION

A cyclically-coupled QC-LDPC (CC-QC-LDPC) code and its decoder architecture are proposed and implemented using a FPGA platform. The decoder for a rate 5/6 CC-QC-LDPC code has been implemented. It achieves a throughput of 3.0 Gbps. The BER results show that CC-QC-LDPC codes have a high potential to outperform LDPCCC in decoding capability with lower complexities. Moreover, at $E_b/N_0 = 3.50$ dB, the CC-QC-LDPC decoder can decode all the 1.14×10^{16} received bits correctly. The evidence suggests that error floor might be

[7]The maximum clock of the min-sum decoder appearing in Table II is actually the result of using the traditional (cascaded) structure with min-sum tables (comparators) in the CNP.

bounded by 10^{-16}, although a hundred times more bits would be required to be simulated by the proposed system.

REFERENCES

[1] R. Gallager, "Low-density parity-check codes," *IRE Trans. on IT*, vol. 8, pp. 21–28, 1962.

[2] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. on IT*, vol. 50, pp. 1788–1793, 2004.

[3] Z. Li, L. Chen, L. Zeng, S. Lin, and W. H. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. on Commun.*, vol. 54, pp. 71–81, 2006.

[4] W. M. Tam, F. C. M. Lau, and C. K. Tse, "A class of QC-LDPC codes with low encoding complexity and good error performance," *IEEE Commun. Lett.*, vol. 14, pp. 169–171, 2010.

[5] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near-Shannon-limit quasi-cyclic low-density parity-check codes," *IEEE Trans. on Commun.*, vol. 52, pp. 1038–1042, 2004.

[6] A. Jimenez Felstrom and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. on IT*, vol. 45, pp. 2181–2191, 1999.

[7] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. on IT*, vol. 50, pp. 2966–2984, 2004.

[8] E. Pisek, D. Rajan, and J. Cleveland, "Trellis-based QC-LDPC convolutional codes enabling low power decoders," *IEEE Trans. on Commun.*, vol. 63, pp. 1939–1951, 2015.

[9] D. J. Costello Jr, A. E. Pusane, S. Bates, and K. S. Zigangirov, "A comparison between LDPC block and convolutional codes," in *Proc. ITAW*, 2006.

[10] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Threshold saturation via spatial coupling: why convolutional LDPC ensembles perform so well over the BEC," *IEEE Trans. on IT*, vol. 57, pp. 803–834, 2011.

[11] D. Mitchell, A. Pusane, K. Zigangirov, and D. Costello, "Asymptotically good LDPC convolutional codes based on protographs," in *ISIT*, 2008, pp. 1030–1034.

[12] C.-W. Sham, X. Chen, W. M. Tam, Y. Zhao, and F. C. M. Lau, "A layered QC-LDPC decoder architecture for high speed communication system," in *APCCAS*, Dec. 2012, pp. 475–478.

[13] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, pp. 58–60, 2001.

[14] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. on IT*, vol. 47, pp. 599–618, 2001.

[15] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. on Commun.*, vol. 47, pp. 673–680, 1999.

[16] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. on Commun.*, vol. 53, pp. 1288 – 1299, 2005.

[17] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes," *IEEE Trans. on Commun.*, vol. 53, pp. 549–554, 2005.

[18] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Trans. on Commun.*, vol. 50, pp. 406–414, 2002.

[19] E. Sharon, S. Litsyn, and J. Goldberger, "An efficient message-passing schedule for LDPC decoding," in *23rd IEEE Convention of EEE in Israel*, 2004, pp. 223–226.

[20] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *IEEE Workshop on SPS*, 2004, pp. 107–112.

[21] E. Sharon, S. Litsyn, and J. Goldberger, "Efficient serial message-passing schedules for LDPC decoding," *IEEE Trans. on IT*, vol. 53, pp. 4076–4091, 2007.

[22] T. Zhang and K. K. Parhi, "A 54 Mbps (3,6)-regular FPGA LDPC decoder," in *SIPS '02*, 2002, pp. 127–132.

[23] Z. Wang and Z. Cui, "Low-complexity high-speed decoder design for quasi-cyclic LDPC codes," *IEEE Trans. on VLSI Systems*, vol. 15, pp. 104–114, 2007.

[24] L. Liu and C.-J. R. Shi, "Sliced message passing: high throughput overlapped decoding of high-rate low-density parity-check codes," *IEEE Trans. on Circ. and Syst. I*, vol. 55, pp. 3697–3710, 2008.

[25] J. Jin and C. ying Tsui, "An energy efficient layered decoding architecture for LDPC decoder," *IEEE Trans. on VLSI Systems*, vol. 18, pp. 1185–1195, 2010.

[26] D. Oh and K. K. Parhi, "Optimally quantized offset min-sum algorithm for flexible LDPC decoder," in *42nd Asilomar Conf. on Signals, Systems and Computers*, 2008, pp. 1886–1891.

[27] X. Chen, J. Kang, S. Lin, and V. Akella, "Memory system optimization for FPGA-based implementation of quasi-cyclic LDPC codes decoders," *IEEE Trans. on Circ. and Syst. I*, vol. 58, pp. 98–111, 2011.

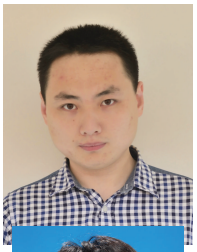
[28] Z. Wang, Z. Cui, and J. Sha, "VLSI design for low-density parity-check code decoding," *IEEE Circuits and Systems Magazine*, vol. 11, pp. 52–69, 2011.

[29] A. J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 404–412, 2002.

[30] C.-W. Sham, X. Chen, F. C. M. Lau, Y. Zhao, and W. M. Tam, "A 2.0 Gb/s throughput decoder for QC-LDPC convolutional codes," *IEEE on Trans. Circ. and Syst. I*, vol. 60, pp. 1857–1869, 2013.

[31] B. Xiang, R. Shen, A. Pan, D. Bao, and X. Zeng, "An area-efficient and low-power multirate decoder for quasi-cyclic low-density parity-check codes," *IEEE Trans. on VLSI Systems*, vol. 18, pp. 1447–1460, 2010.

[32] A. E. Pusane, R. Smarandache, P. O. Vontobel, and D. J. Costello, "Deriving good LDPC convolutional codes from LDPC block codes," *IEEE Trans. on IT*, vol. 57, pp. 835–857, 2011.



Qing Lu received the B.E. degree from Dalian University of Technology, China, and M.Sc. degree from The Hong Kong Polytechnic University (HKPolyU), Hong Kong, respectively. He is currently a research assistant at HKPolyU. His research interests include digital implementation of communication systems, VLSI design optimization and system development based on FPGAs.

Jianfeng Fan received the B.E. degree from Heilongjiang University, China, and the M.Sc. degree from HKPolyU. From 2013 to 2014, he was a research assistant at HKPolyU. His research interests include hardware design for coding.

Chiu-Wing Sham received the B.Eng., M.Phil. and Ph.D. degrees from The Chinese University of Hong Kong, Hong Kong. Before joining HKPolyU in 2006, he was a Research Engineer with Synopsys, China, and an Electronic Engineer with ASM (HK). He received the Best Paper Award in ISQED 2013. His research interests include design automation of VLSI, design optimization of digital VLSI systems and embedded systems.

Wai M. Tam received B.Sc. degree from Jinan University, China, and M.Phil. and Ph.D. degrees from HKPolyU, Hong Kong. She is currently a Research Fellow at HKPolyU. Her research interests include channel coding, mobile cellular systems, complex networks and chaos-based digital communications.

Francis C. M. Lau (M'93–SM'03) received the B.Eng. (Hons) degree and Ph.D. degree from King's College London, University of London, U.K. He is a Professor and Associate Head at the Department of Electronic and Information Engineering, HKPolyU, Hong Kong. His main research interests include applications of complex network theories, channel coding, cooperative networks, wireless sensor networks, chaos-based digital communications, and wireless communications.