VLSI Architecture for Layered Decoding of QC-LDPC Codes With High Circulant Weight

Yang Sun and Joseph R. Cavallaro

***Abstract*— In this brief, we propose a high-throughput layered decoder architecture to support a broader family of quasicyclic low-density parity-check (QC-LDPC) codes, whose parity-check matrices are constructed from arrays of circulant submatrices. Each nonzero circulant submatrix is a superposition of K cyclic-shifted identity matrices, where the circulant weight $K \geq 1$. We propose a novel layered decoder architecture to support QC-LDPC codes with any circulant weight. We present a block-serial decoding architecture which processes a layer of a parity check matrix block by block, where each block is a $Z \times Z$ circulant matrix with a circulant weight of K. In the case study, we demonstrate an LDPC decoder design for the China Mobile Multimedia Broadcasting (CMMB) standard, which was synthesized for a TSMC 65-nm CMOS technology. With a core area of 3.9 mm^2, the CMMB LDPC decoder achieves a maximum throughput of 1.1 Gb/s with 15 iterations.**

***Index Terms*— ASIC, MIMO detection, shortest path algorithm, soft-output MIMO detector, VLSI architecture.**

I. Introduction

Low-density parity-check (LDPC) codes [1] have received tremendous attention in the coding community because of their remarkable error correction capability. In the LDPC code family, the structured QC-LDPC codes have been widely used in many practical systems due to their efficient hardware implementation and good error performance. Layered decoder architectures are often used to decode structured QC-LDPC codes [2]–[9].

The traditional layered decoder architecture handles only circulant-weight-1 LDPC codes. For example, the LDPC codes defined in IEEE 802.11 n and IEEE 802.16e standards are an array of circulant submatrices, where each submatrix is either a zero matrix or a $Z \times Z$ cyclic-shifted identity matrix. The circulant weight of each nonzero submatrix is 1, so that there are no data conflicts when updating the *a posteriori* probability (APP) log likelihood ratio (LLR) values. However, more generally, a nonzero submatrix can be a superposition of K cyclic-shifted identity matrices, i.e., the circulant weight K can be greater than 1. For example, the parity check matrices defined in the DVB-S2, DVB-T2, and CMMB LDPC codes consist of both circulant-weight-1 and circulant-weight-2 submatrices. One of the parity check matrices proposed for the wireless personal area network (WPAN) standard is composed of circulant-weight-4 submatrices [10]. Some custom-designed high-rate LDPC codes [11] in storage applications have a circulant weight of 3 or larger.

Due to the LLR memory updating conflicts, the traditional layered decoder architecture cannot be directly used to support QC-LDPC codes with a circulant weight of $K > 1$. Marchand *et al.* [12] present a solution to avoid the memory conflicts by splitting a circulant-weight-2 submatrix into two circulant-weight-1 submatrices. This scheme works for a specific LDPC code structure. But it has the disadvantage of reduced parallelism and is not very scalable to handle higher circulant weights. In [13], a specific solution to handle circulant-weight-2 submatrices is presented; it splits the LLR memory into two banks to avoid conflicts. This scheme does not scale very well to support higher circulant weights, i.e., $K > 2$, and the maximum row parallelism is limited. In [14], the authors propose a specific decoder architecture for $K = 2$ cases by assuming that there is always an identity matrix in any of the double-diagonal matrices. Therefore, it is not very straightforward for the decoder in [14] to handle a more general $K >= 2$ case with arbitrary shift values. The authors in [15] present a solution to handle circulant-weight-2 submatrices by changing the layer updating order. However, to make this scheme work, the matrix needs to be reordered in some way to avoid conflicts among layers. As the layer reordering pattern depends on the structure of the parity check matrix, it is not a general solution and the scheme only works for certain code structures.

The main contribution of this brief is a general layered decoder architecture that can support LDPC codes constructed from arbitrary high-circulant-weight ($K \geq 1$) submatrices. We propose a block-serial layered decoding architecture that processes the parity check matrix block by block, where each block is a $Z \times Z$ circulant submatrix with a circulant weight of K. Each block is processed in one clock cycle so that the decoding latency in terms of clock cycles is independent of the circulant weight K. In the case studies, we have implemented a layered LDPC decoder for the China Mobile Multimedia Broadcasting (CMMB) and DVB-S2 standards.

II. Algorithm

A. QC-LDPC Codes and the Layered Decoding Algorithm

QC-LDPC codes are a special class of LDPC codes. The parity check matrix for a QC-LDPC code can be represented with an array of submatrices, where each submatrix is either a Z × Z zero matrix or a Z × Z circulant matrix. The cyclic-shifted identity matrix is a special case of the circulant matrix where the circulant weight is 1. However, more generally, the circulant weight of a sub-matrix can be greater than 1 by superimposing multiple cyclic-shifted identity matrices in. For example, K can be 2 for the LDPC codes defined in the DVB-S2, DVB-T2, and CMMB. K can be 3 or 4 for certain custom-designed high-rate LDPC codes in storage applications [11]. In this brief, we investigate the more general case where the circulant submatrix consists of K superimposed cyclic-shifted identity matrices, where $K \geq 1$.

We adopt the layered decoding algorithm [2] in our VLSI implementation. To facilitate the algorithm and architecture description, we define the following notation: The APP LLR of each bit n is defined as $L_n = \log(Pr(n = 0)/Pr(n = 1))$. The check node message from check node m to variable node n is denoted as $R_{m,n}$. The variable message from variable node n to check node m is denoted as $Q_{m,n}$.

The conventional layered decoding algorithm works only for LDPC codes with circulant-weight-1 submatrices ($K = 1$ case), where each variable node in a layer of $\mathbf{H}$ has at most one check node connected to it. However, for the case of $K > 1$, there could be multiple check nodes connected to the same variable node in a layer of $\mathbf{H}$. This will cause LLR updating conflicts when multiple check nodes are trying to update the check-to-variable messages for the same variable node. To resolve the access confliction, researchers have proposed to combine the contributions from every check node that connects to the same variable node [14], [16], [17]. Let $m_0, m_1, \ldots, m_{K-1}$ denote the check nodes that are connected to a variable node n in a layer of $\mathbf{H}$. Then, the LLR value for variable node n can be updated as [14],

Manuscript received October 15, 2011; revised June 6, 2012; accepted August 31, 2012. This work was supported in part by Renesas Mobile, Texas Instruments, Xilinx, Samsung, Huawei, and the U. S. National Science Foundation under Grant, ECCS-1232274, Grant EECS-0925942, and Grant CNS-0923479.

The authors are with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA (e-mail: ysun@rice.edu; cavallar@rice.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2012.2220388

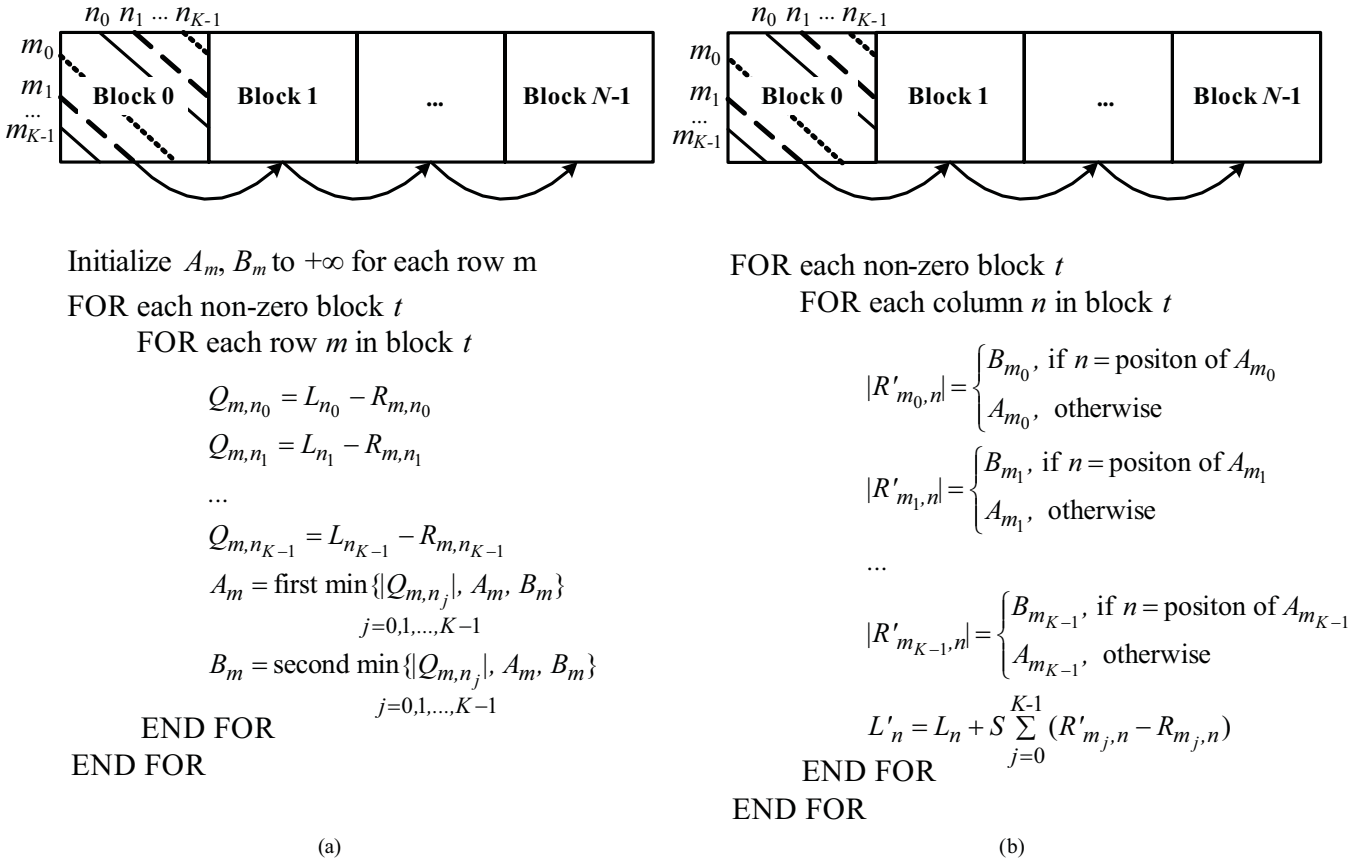

Fig. 1. Block-serial scheduling algorithm and its pseudocode implementation.

[16], [17]

$$L'_n = L_n + \sum_{j=0}^{K-1}(R'_{m_j,n} - R_{m_j,n}). \quad (1)$$

It should be noted that there is a small side effect of this LLR combining. For example, the performance degradation for $K = 2, 3, 4$ cases is less than 0.1 dB according to [17].

B. Proposed Block-Serial Scheduling Algorithm

To implement the extended layered algorithm as shown in (1), we developed a block-serial scheduling algorithm, which is shown in Fig. 1. In this algorithm, a parity check matrix is processed layer after layer. Inside each layer, each $Z \times Z$ submatrix is treated as a unit for parallel processing. The decoding algorithm is based on the scaled min-sum algorithm with a scaling factor of S. As shown in Fig. 1, the submatrices in a layer are serially scanned through twice. For a given row m, each submatrix has K nonzero column positions, i.e., $n_0, n_1, \ldots, n_{K-1}$. Before decoding, the LLR L_n for the bit n is initialized with the channel LLR value, and the check node messages $R_{m,n}$ are initialized with 0. The bottom portion of Fig. 1 shows the pseudocode for the proposed scheduling algorithm. In the algorithm, A_m and B_m are two variables for holding the first minimum and the second minimum values for parity check row m and are initialized to $+\infty$. The processing time for the proposed scheduling algorithm is independent of the circulant weight.

III. VLSI Architecture

We propose a flexible VLSI architecture for layered decoding of QC-LDPC codes with circulant weight (K). Unlike the methods in the existing solutions such as [14] and [15], which are specially designed for a particular code matrix, we aim at designing a more general decoder architecture to handle QC-LDPC codes with arbitrary circulant submatrices. The proposed LDPC decoder architecture is shown in Fig. 2. This architecture is a partial-parallel architecture by employing Z check node processing (CNP) units. Fig. 3 shows the block diagram of the CNP unit which is to compute the check node messages.

In this scalable decoder architecture, the memory structures are designed in such a way that they do not change as the circulant weight K changes. This is one of the important features that differentiate our decoder from the existing solutions. There are three types of storage: the LLR memory (LLR-Mem), the R memory (R-Mem), and the T register (T-Reg). The LLR-Mem is used for storing the initial and updated LLR values for each bit of a codeword. For LDPC codes with $M \times N$ submatrices, where each submatrix is a $Z \times Z$ circulant matrix, the LLR-Mem is organized such that Z LLR values are stored in the same memory word and there are N words in the memory. Each memory word has $Z \times W$ bits, where W is the bit width for each LLR. The R-Mem in the CNP is used to store the information for regenerating the $R_{m,n}$ values. Because of the min-sum algorithm, the check node messages can be stored in a compressed manner. For each row m, only the first minimum, the second minimum, the position of the first minimum, and the sign bits for all $Q_{m,n}$ values associated with row m are stored in the R-Mem. The R-Mem has M words in total. The T-Reg in the CNP is used to store the information for regenerating the new check node messages $R'_{m,n}$ for the current layer. The T-Reg has the same organization as one memory word in the R-Mem. As none of the memory structures in our decoder depends on the K value, the memory system is very flexible in supporting any $K \geq 1$ value.

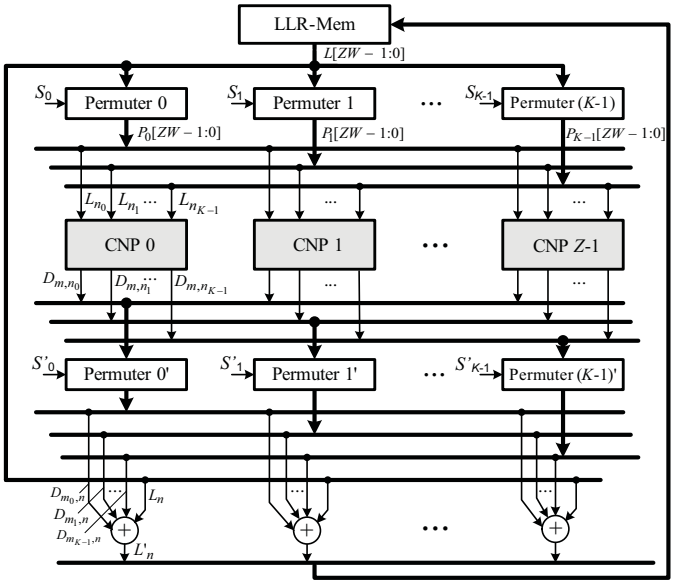

Fig. 2. Scalable LDPC decoder architecture for supporting high-weight circulant matrices.

The decoding data flow is based on the block-serial decoding algorithm described in Section II-B. During the first scan, Z LLR values $L[ZW-1:0]$ (the word length of each LLR value is W bits) are loaded from the LLR-Mem (one memory word at each clock cycle). And, then, the LLR values are passed to K permuters to produce K permuted LLR values $P_j[ZW-1:0]$, $j=0,1,\ldots,K-1$. Each signal $P_j[ZW-1:0]$ is sliced into Z subwords, which are passed to Z CNP units. The check node data computation takes place in the CNP units (see, Fig. 3). There are Z instances of such CNP units, where each CNP unit is responsible for processing one check row. For each row m, there are K columns $(n_0, n_1, \ldots, n_{K-1})$ having "1" in the current nonzero circulant submatrix. In each CNP unit, the K variable-to-check node messages Q_{m,n_j} from bit n_j to check node m, where $j=0,1,\ldots,K-1$, are computed simultaneously as

$$Q_{m,n_j} = L_{n_j} - R_{m,n_j}, \quad j=0,1,\ldots,K-1 \tag{2}$$

where R_{m,n_j} is the old check node message. Because the R values are stored in a compressed way in the R-Mem, we need to regenerate the R values by using a R-Gen unit. Let A_m and B_m denote the first minimum ("Min 0") and the second minimum ("Min 1") for row m. Let POS_m denote the position of the first minimum. Then, each R_{m,n_j} value is generated as

$$|R_{m,n_j}| = \begin{cases} 0.75 \times B_m, & \text{if } n_j = POS_m \\ 0.75 \times A_m, & \text{otherwise} \end{cases} \tag{3}$$

$$\text{sign}(R_{m,n_j}) = \prod_{j \in \mathcal{N}_m \backslash n_j} \text{sign}(Q_{m,j}). \tag{4}$$

Next, the Q_{m,n_0}, Q_{m,n_1}, …, and $Q_{m,n_{K-1}}$ values are compared against TA_m and TB_m read from the T-Reg, where TA_m and TB_m are the first minimum and second minimum temporary variables, which are initialized to the maximum positive values. The new two minimum values TA'_m and TB'_m are written to the T-Reg as TA_m and TB_m. The index of the first minimum Q value (POS_m) and sign bits for all Q values are also updated in T-Reg.

During the second-scan, the LLR values are updated. The R'-Gen unit gets values from the T-Reg and generates the most recent check node messages for row m as

$$|R'_{m,n_j}| = \begin{cases} 0.75 \times TB_m, & \text{if } n_j = POS_m \\ 0.75 \times TA_m, & \text{otherwise} \end{cases} \tag{5}$$

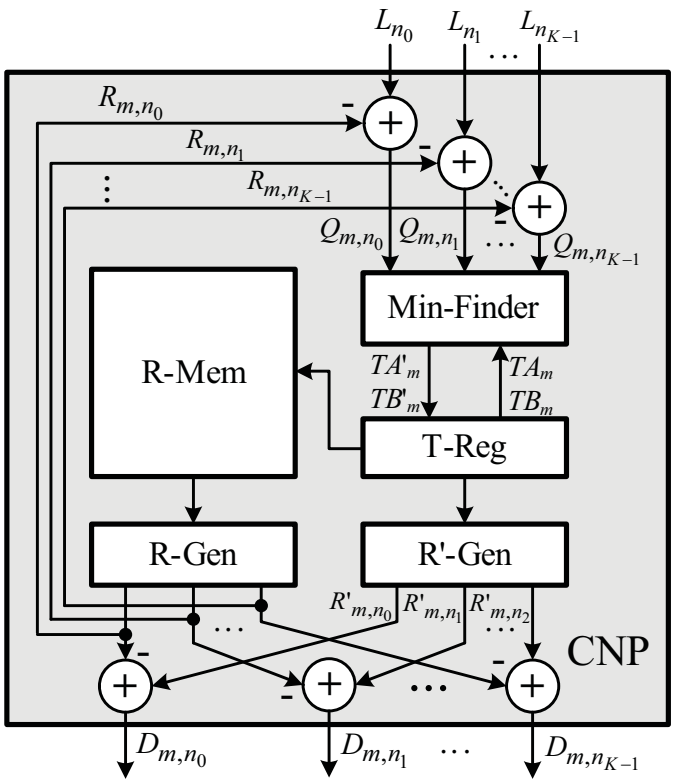

Fig. 3. Efficient implementation of the CNP unit.

$$\text{sign}(R'_{m,n_j}) = \prod_{j \in \mathcal{N}_m \backslash n_j} \text{sign}(Q_{m,j}). \tag{6}$$

At the same time, the R-Gen unit gets values from the R-Mem and generates the old K check node messages $R_{m,n_0}, R_{m,n_1}, \ldots, R_{m,n_{K-1}}$ for row m based on (3-4). Then, K delta values are computed as follows:

$$D_{m,n_j} = R'_{m,n_j} - R_{m,n_j}, \qquad j=0,1,\ldots,K-1. \tag{7}$$

Next, the delta values associated with the same shift value S_j are concatenated and are passed to permuters j'. After permutation, for each column n, K permuted delta values $D_{m_j,n}$ are obtained. Note that we only need one set of permuters which are time-shared between the first-scan process and the second-scan process to save area. For example, in Fig. 2, permuter j can be time shared with permuter j'. Therefore, K permuters would be required in this architecture. If using this shared architecture, in the first-scan, permuters take inputs from the LLR-Mem and produce outputs to CNPs, while in the second-scan they take inputs from CNPs and produce outputs to adders for updating LLRs. To support different submatrix sizes, the permuter needs to be flexible. In this brief, we adopt the flexible permuter proposed in [4].

After the delta values are permuted, the new LLR values are generated by adding the permuted delta values to the old LLR values as

$$L'_n = L_n + \sum_{j=0}^{K-1} D_{m_j,n} \tag{8}$$

where $D_{m_j,n}$ is the jth delta value related to column n. The new LLR values are then written back to the LLR-Mem. After the second-scan, the contents of the T-Reg are written to the R-Mem overwriting the values for the current layer.

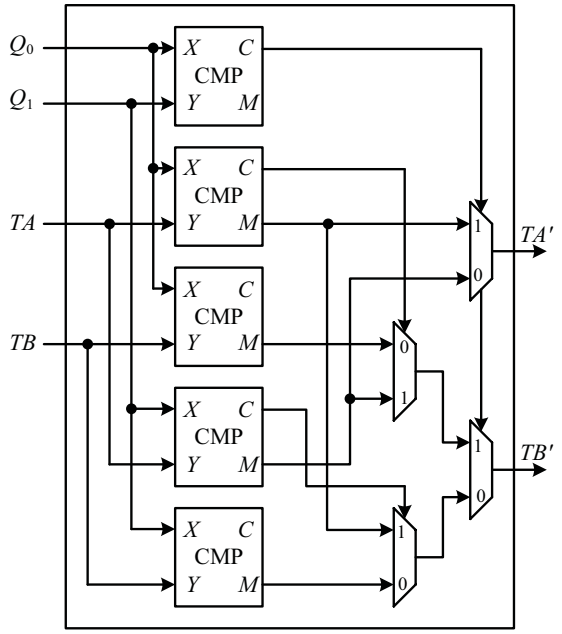

Fig. 4. Block diagram of the min-finder unit for $K = 2$.

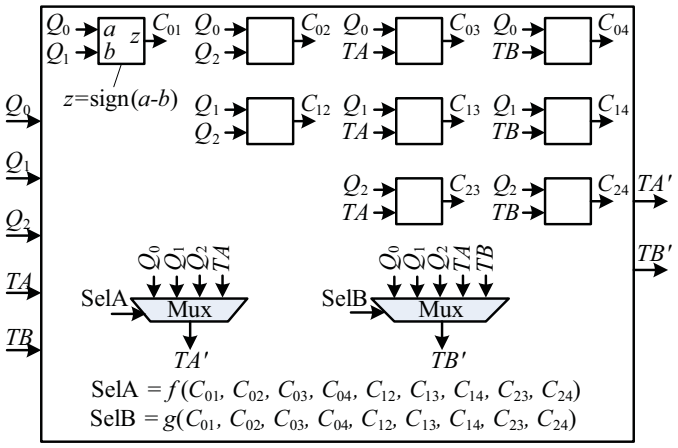

Fig. 5. Block diagram of the min-finder unit for $K = 3$.

This decoding process will be performed for all M layers and for multiple iterations. The data path of the decoder scales up as the circulant weight K increases, but the latency in terms of clock cycles remains constant as K increases. The architecture of the min-finder unit will change with the K value. Fig. 4 shows the block diagram of the min-finder for $K = 2$. This circuit takes four inputs: the first minimum TA, the second minimum TB, and two Q values (Q_0, Q_1). The circuit generates two outputs: the new first minimum TA', and the new second minimum TB'. As shown in Fig. 4, the min-finder unit consists of five comparison (CMP) units and four multiplexors. The CMP unit compares the two input data X and Y, and generates the minimum of X and Y: $M = \min(X, Y)$, and the sign of $X - Y$: $C = \text{sign}(X - Y)$. This min-finder for $K = 2$ has a critical path delay of one adder delay plus some multiplexor delays. Fig. 5 shows the block diagram of the min-finder for $K = 3$. The critical path of the min-finder for $K = 3$ is the sum of an adder delay, a combinational logic delay, and a multiplexor delay. Note that the min-finder for $K = 3$ shown in Fig. 5 can be generalized for any other K values.

With this architecture, the decoding throughput can be expressed as follows:

$$\text{Throughput} = \frac{N \times Z \times \text{Rate} \times \text{fclk}}{(2 \times E + M \times \Delta) \times \text{Iter}} \tag{9}$$

where M and N are the number of the block rows and block columns in $\mathbf{H}$, respectively, Z is the size of the submatrix, Rate is the code rate, E is the total number of nonzero submatrices in $\mathbf{H}$, Iter is the iteration number, and Δ is the pipeline delay in the data path for processing a layer of a parity check matrix. A typical pipeline delay is around 2 clock cycles. Note that the throughput is independent of the K value because we process each circulant submatrix as a unit.

IV. Case Studies

A. CMMB LDPC Decoder Design

We have implemented a high-throughput LDPC decoder for the CMMB standard using the proposed architecture with $Z = 256$ and $K = 2$. The CMMB LDPC codes have two code rates of 1/2 and 3/4. The two codes have the same block length of 9216 bits. The parity check matrix $\mathbf{H}$ is an array of 18×36 circulant submatrices for the rate 1/2 code, and an array of 9×36 circulant submatrices for the rate 3/4 code. Each nonzero submatrix is either a 256×256 cyclic-shifted identity matrix ($K = 1$), or a superposition of two cyclic-shifted identity matrices of dimension 256×256 ($K = 2$).

TABLE I
CMMB LDPC Decoder Comparison

	[13]	[15]	[14]	This brief
Matrix Kernel Size	256×256	256×256	128×128	256×256
Technology (nm)	90	130	130	65
Clock Frequency (MHz)	431	83.3	200	600
Scaled Clock Frequency[a] (MHz)	596	166.6	400	600
Core Area (mm^2)	4.4	7.59	4.75	3.9
Scaled Area[b] (mm^2)	2.3	1.9	1.19	3.9
Throughput (15 Iterations, rate 3/4)	342 Mb/s	90 Mb/s	243.8 Mb/s	1.1 Gb/s
Scaled Throughput[a]	474 Mb/s	180 Mb/s	487.6 Mb/s	1.1 Gb/s
Area Metric = $\frac{\text{Scaled Area}}{\text{Scaled Throughput}}$ ($\frac{\mu m^2}{\text{Mb/s}}$)	4.9	10.5	2.44	3.5

Frequency and throughput scaling factors are (feature size/65 nm) [18].
Area scaling factor is (65 nm/feature size)^2 [18].

In our implementation, the channel input LLR is represented with 6 bit signed numbers with 2 fractional bits. The word lengths of the R value and the LLR value are 6 bits and 7 bits, respectively. The decoder has been described using Verilog, synthesized using Synopsys Design Compiler, and placed and routed using Cadence SoC Encounter for a 1.0 V TSMC 65nm CMOS technology. The maximum clock frequency is 600 MHz based on the post-layout timing analysis. The maximum throughput is 1.1 Gb/s with 15 iterations.

Table I compares the VLSI implementation results of the proposed decoder with existing CMMB LDPC decoders from [13]–[15]. In the comparison, we scaled the area, clock frequency, and throughput of these decoders to a common 65-nm technology based on the technology scaling rule [18]. To compare the hardware efficiency, we define an area metric as (Scaled Area/Scaled Throughput). After technology scaling, we can see that [13] can achieve a throughput of 474 Mb/s (15 iter.) with a core area of 2.3 mm^2, [15] can achieve a throughput of 180 Mb/s

(15 iter.) with a core area of 1.9 mm^2, and [14] can achieve a throughput of 487.6 Mb/s (15 iter.) with a core area of 1.19 mm^2. In comparison, our decoder can achieve a high throughput of 1.1 Gb/s (15 iter.) with a core area of 3.9 mm^2. Compared with existing solutions, our decoder achieves higher throughput and good area-to-throughput performance. It should be noted that the decoder in [14] was implemented based on a smaller matrix kernel size of 128×128, whereas all the other decoders were designed based on a larger matrix kernel size of 256×256.

B. DVB-S2 LDPC Decoder Design

We have designed an LDPC decoder for DVB-S2 standard. As the submatrix size of the DVB-S2 LDPC code is very large (up to 360), we modified the proposed LDPC decoder architecture to support scalable parallelism. We fold the decoder so that a layer is split into a number of sections, where the APP messages are passing from sections to sections within a layer. We have implemented an eightfold decoder. The decoder has been synthesized for a 1.0 V TSMC 65-nm CMOS technology. The core area is 8.5 mm^2 and the throughput at 30 iterations is 210 Mb/s. We compare our decoder with the state-of-the-art DVB-S2 LDPC decoder from [8]. The decoder from [8] has a maximum throughput of 180 Mb/s with a core area of 6.03 mm^2 (65 nm). The throughput performance and the area efficiency, measured with (Area/Throughput), of our general decoder are comparable to those of the specialized decoder from [8]. The proposed decoder is more flexible to handle more different types of matrices in a unified and efficient way.

V. Conclusion

We presented a novel layered decoder architecture to support QC-LDPC codes with weight-K circulant submatrices. We successfully resolved the APP LLR updating conflicts by using a novel block-serial decoding architecture. The proposed LDPC architecture is more scalable than the traditional solution in terms of circulant weight.

References

[1] R. Gallager, "Low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[2] M. M. Mansour and N. R. Shanbhag, "High-throughput LDPC decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 6, pp. 976–996, Dec. 2003.

[3] H. Zhong and T. Zhang, "Block-LDPC: A practical LDPC coding system design approach," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 4, pp. 766–775, Apr. 2005.

[4] Y. Sun, M. Karkooti, and J. R. Cavallaro, "VLSI decoder architecture for high throughput, variable block-size and multi-rate LDPC codes," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 2104–2107.

[5] Z. Cui, Z. Wang, and Y. Liu, "High-throughput layered LDPC decoding architecture," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 4, pp. 582–587, Apr. 2009.

[6] K. Zhang, X. Huang, and Z. Wang, "High-throughput layered decoder implementation for quasi-cyclic LDPC codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 985–994, Aug. 2009.

[7] M. Rovini, F. Rossi, P. Ciao, N. L'Insalata, and L. Fanucci, "Layered decoding of non-layered LDPC codes," in *Proc. 9th EUROMICRO Conf. DSD*, Oct. 2006, pp. 537–544.

[8] S. Muller, M. Schreger, M. Kabutz, M. Alles, F. Kienle, and N. Wehn, "A novel LDPC decoder for DVB-S2 IP," in *Proc. Design, Autom. Test Eur.*, Apr. 2009, pp. 1308–1313.

[9] E. Boutillon, J. Tousch, and F. Guilloud, "LDPC decoder, corresponding method, system and computer program," U.S. Patent 7 174 495, Feb. 6, 2007.

[10] Y.-M. Chang, A. I. V. Casado, M.-C. F. Chang, and R. D. Wesel, "Lower-complexity layered belief-propagation decoding of LDPC codes," in *Proc. IEEE Int. Conf. Commun.*, May 2008, pp. 1155–1160.

[11] H. Zhong, "VLSI architectures of LDPC based signal detection and coding system for magnetic recording channel," Ph.D. dissertation, Dept. Electr., Comput. & Syst. Eng., Rensselaer Polytechnic Inst., Troy, NY, May 2006.

[12] C. Marchand, J.-B. Dore, L. Conde-Canencia, and E. Boutillon, "Conflict resolution by matrix reordering for DVB-T2 LDPC decoders," in *Proc. IEEE Global Telecommun. Conf.*, Mar. 2009, pp. 1–6.

[13] K. Zhang, X. Huang, and Z. Wang, "A dual-rate LDPC decoder for China multimedia mobile broadcasting systems," *IEEE Trans. Consum. Electron.*, vol. 56, no. 2, pp. 399–407, May 2010.

[14] C. Zhou, Y. Ge, X. Chen, Y. Chen, and X. Zeng, "An area-efficient LDPC decoder for multi-standard with conflict resolution," in *Proc. IEEE Int. Conf. Appl.-Specific Syst., Arch. Process.*, Sep. 2011, pp. 105–112.

[15] K. Guo, Y. Hei, and S. Qiao, "A parallel-layered belief-propagation decoder for non-layered LDPC codes," *J. Commun.*, vol. 5, no. 5, pp. 400–408, May 2010.

[16] Y. Zhu and C. Chakrabarti, "Aggregated circulant matrix based LDPC codes," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2006, pp. 916–919.

[17] Y. Sun, "Parallel VLSI architectures for multi-Gb/s MIMO communication systems," Ph.D. dissertation, Dept. Electr. & Comput. Eng., Houston, TX, Dec. 2010.

[18] J. R. Hauser, "MOSFET device scaling," in *Handbook of Semiconductor Manufacturing Technology*, R. Doering and Y. Nishi, Eds. Boca Raton, FL: CRC Press, 2008, ch. 1.3, pp. 8–21.