Multi-Layer Parallel Decoding Algorithm and VLSI Architecture for Quasi-Cyclic LDPC Codes

Yang Sun, Guohui Wang, and Joseph R. Cavallaro
Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005
Email: {ysun, gw2, cavallar}@rice.edu

***Abstract*—We propose a multi-layer parallel decoding algorithm and VLSI architecture for decoding of structured quasi-cyclic low-density parity-check codes. In the conventional layered decoding algorithm, the block-rows of the parity check matrix are processed sequentially, or layer after layer. The maximum number of rows that can be simultaneously processed by the conventional layered decoder is limited to the sub-matrix size. To remove this limitation and support layer-level parallelism, we extend the conventional layered decoding algorithm and architecture to enable simultaneously processing of multiple (K) layers of a parity check matrix, which will lead to a roughly K-fold throughput increase. As a case study, we have designed a double-layer parallel LDPC decoder for the IEEE 802.11n standard. The decoder was synthesized for a TSMC 45-nm CMOS technology. With a synthesis area of 0.81 mm^2 and a maximum clock frequency of 815 MHz, the decoder achieves a maximum throughput of 3.0 Gbps at 15 iterations.**

I. Introduction

Quasi-Cyclic LDPC (QC-LDPC) codes have been widely used in many practical systems, such as IEEE 802.11n WLAN and IEEE 802.16e WiMAX, due to their efficient hardware implementation and good error correcting performance. The parity check matrix of these QC-LDPC codes can be partitioned into block-rows or layers for efficient implementation by using the so-called layered decoding algorithm [1], [2]. The layered decoding algorithm has a much higher convergence speed (up to two times faster), and requires less memory compared to the standard two-phase flooding decoding algorithm. Thus, partial-parallel layered decoder architectures are often used to decode the QC-LDPC codes [3], [4], [5], [6], [7], [8], [9], [10].

The conventional layered decoder architecture [1], [2] is initially developed to process the parity check matrix layer by layer, where each layer corresponds to a block-row of the parity check matrix. Since the column-weight of each layer is typically 1 in many applications, such as IEEE 802.11n and IEEE 802.16e, this greatly simplifies the decoder design. To further improve the throughput, the two consecutive layers' data processing can be partially overlapped through a pipelined schedule [5], [3], where the data conflicts between two layers can be resolved by stalling the pipeline. The maximum row parallelism for the conventional layered algorithm is equal to the sub-matrix size Z, i.e. we can employ Z parallel check node processors to process Z rows in parallel. With this amount of parallelism, the conventional layered decoder can typically offer 100-1000 Mbps throughput [3], [4], [5], [6].

To go beyond 1-Gbps throughput, the layered architecture needs to be extended to provide higher parallelism. One natural extension of the conventional layered architecture is to design a multi-layer parallel architecture where multiple (K) layers of a parity check matrix are processed in parallel. Now the maximum row parallelism is increased to KZ, i.e. we can employ KZ check node processors to process KZ rows in parallel. It should be noted that the multi-layer parallel decoding algorithm would still require less memory than the two-phase flooding algorithm because there is still no need to store the variable node messages in the multi-layered algorithm.

In this paper, we propose a multi-layer parallel decoding algorithm and VLSI architecture for high throughput LDPC decoding. The data conflicts between layers are resolved by modifying the LLR update rules. As a case study, we describe a double-layer parallel decoder architecture for IEEE 802.11n LDPC codes.

II. Multi-Layer Parallel Decoding Algorithm

A. Introduction of The QC-LDPC Codes

Generally, a binary LDPC code is a linear block code specified by a sparse binary parity check matrix: $\mathbf{H}\cdot\mathbf{x}^{\mathbf{T}} = 0$, where $\mathbf{x}$ is a codeword and $\mathbf{H}$ can be viewed as a bipartite graph where each column and row in $\mathbf{H}$ represent a variable node and check node, respectively. QC-LDPC codes are a very important class of LDPC codes. The parity check matrix for a QC-LDPC code can be represented as an array of square sub-matrices, where each sub-matrix is either a $Z \times Z$ zero matrix or a $Z \times Z$ circulant matrix. As an example, Fig. 1 shows the parity check matrix for the block length 1944 bits, code rate 1/2, sub-matrix size $Z = 81$, IEEE 802.11n LDPC code. In this matrix representation, each square box with a label I_x represents an 81×81 cyclicly-shifted identity matrix with a shifted value of x, and each empty box represents an 81×81 zero matrix.

I_{57}				I_{50}		I_{11}		I_{50}		I_{79}		I_1	I_0									
I_3		I_{28}		I_0				I_{55}	I_7				I_0	I_0								
I_{30}				I_{24}	I_{37}			I_{56}	I_{14}					I_0	I_0							
I_{62}	I_{53}			I_{53}			I_3	I_{35}							I_0	I_0						
I_{40}			I_{20}	I_{66}			I_{22}	I_{28}								I_0	I_0					
I_0				I_8		I_{42}		I_{50}		I_8							I_0	I_0				
I_{69}	I_{79}	I_{79}				I_{56}		I_{52}			I_0							I_0	I_0			
I_{65}				I_{38}	I_{57}			I_{72}		I_{27}									I_0	I_0		
I_{64}				I_{14}	I_{52}			I_{30}			I_{32}									I_0	I_0	
	I_{45}		I_{70}	I_0				I_{77}	I_9												I_0	I_0
I_2	I_{56}		I_{57}	I_{35}						I_{12}												I_0
I_{24}		I_{61}		I_{60}			I_{27}	I_{51}			I_{16}	I_1										I_0

Fig. 1. Parity check matrix for block length 1944 bits, code rate 1/2, sub-matrix size $Z = 81$, IEEE 802.11n LDPC code.

B. Review of The Layered Decoding Algorithm

A brief review of the conventional layered decoding algorithm [1] is provided to facilitate the discussion. We define the following notation. The *a posteriori* probability (APP) log-likelihood ratio (LLR) of each bit n is defined as: $L_n = \log \frac{Pr(n=0)}{Pr(n=1)}$, where L_n is initialized to be the channel input LLR. 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 algorithm, or single-layer algorithm, assumes that the rows are grouped into layers where the parity check matrix for this layer has at most a column-weight of one. The single-layer algorithm only handles one layer at a time, i.e. the maximum row parallelism is limited to the sub-matrix size Z. Each

layer is processed as a unit, one layer after another. For each non-zero column n inside the current layer, variable node messages $Q_{m,n}$ that correspond to a row m are formed by subtracting the check node message $R_{m,n}$ from the APP LLR message L_n:

$$Q_{m,n} = L_n - R_{m,n}. \tag{1}$$

For each row m, the new check node messages $R'_{m,n}$, corresponding to all variable nodes j that participate in this parity-check equation, are computed using the belief propagation algorithm. In this work, we use the scaled min-sum approximation algorithm (with scaling factor of S) to compute the R value:

$$R'_{m,n} \approx S \cdot \prod_{j \in \mathcal{N}_m \backslash n} \text{sign}(Q_{m,j}) \cdot \min_{j \in \mathcal{N}_m \backslash n} |Q_{m,j}|, \tag{2}$$

where $\mathcal{N}_m$ is the set of variable nodes that are connected to check node m, and $\mathcal{N}_m \backslash n$ is the set $\mathcal{N}_m$ with variable node n excluded. It should be noted that the equation above can be easily modified for offset min-sum algorithm. After the check nodes messages are computed, the new APP LLR messages L'_n are updated as:

$$L'_n = L_n + R'_{m,n} - R_{m,n}. \tag{3}$$

Then repeat equations (1)-(3) for each layer.

C. Proposed Multi-Layer Parallel Decoding Algorithm

To support layer-level parallelism, we propose a multi-layer (K-layer) parallel decoding algorithm, where the maximum row parallelism is increased to KZ. When using the conventional layered algorithm to process multiple layers at the same time, data conflicts may occur when updating the LLRs because there can be more than one check node connected to a variable node. Fig. 2 shows an example of the data conflicts when updating LLRs for two consecutive layers, where check node (or row) m_0 and check node m_1 are both connected to variable node (or column) n. To resolve the data conflicts, we use the following LLR update rule for a K-layer parallel decoding algorithm. For a variable node n, let m_k represents the k-th check node that is connected to variable node n. Then the LLR value for variable node n is updated as:

$$L'_n = L_n + \sum_{k=0}^{K-1} (R'_{m_k,n} - R_{m_k,n}). \tag{4}$$

Compared to the original LLR update rule (3), the new LLR update rule combines all the check node messages and adds them to the old LLR value. We can define a macro-layer as a group of K layers of the parity check matrix. The multi-layer parallel decoding algorithm is summarized as follows. For each layer k in each macro-layer l, do the following:

$$Q_{m_k,n} = L_n - R_{m_k,n} \tag{5}$$

$$R'_{m_k,n} = S \cdot \prod_{j \in \mathcal{N}_{m_k} \backslash n} \text{sign}(Q_{m_k,j}) \cdot \min_{j \in \mathcal{N}_{m_k} \backslash n} |Q_{m_k,j}| \tag{6}$$

$$L'_n = L_n + \sum_{k=0}^{K-1} (R'_{m_k,n} - R_{m_k,n}). \tag{7}$$

In the above calculation, the LLR values L_n are updated macro-layer after macro-layer. Within each macro-layer, all the check rows can be processed in parallel, which therefore leads to a K times larger parallelism than the conventional layered algorithm. For example, we can use KZ number of check node processors to process KZ rows in parallel.

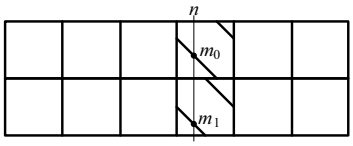

Fig. 2. Example of the data conflicts when updating LLRs for two layers.

III. Decoding Performance Evaluation

In the multi-layer parallel decoding algorithm, the layer-parallelism K will have some negative impact on the decoding convergence speed because the LLR updates occur less frequently than in the single-layer algorithm. To compare the performance of the multi-layer parallel decoding algorithm against the conventional layered decoding algorithm, we perform floating-point simulations for block length 1944 bits, code rate 1/2 IEEE 802.11n LDPC code. BPSK modulation is used for an AWGN channel. In the simulation, we collect at least 100 frame errors and the maximum iteration number is set to 15 for all the experiments. Fig. 3 compares the frame error rate (FER) performance of K-layer parallel decoders for $K = 1, 2, 3, 4, 6$. We also plot the FER curve for the two-phase flooding algorithm for comparison. As can be seen from the figure, the double-layer parallel decoder has shown a negligible performance loss, and the triple-layer parallel decoder has shown a small performance loss (< 0.1 dB). As K increases, the FER performance slowly degrades as expected. Note that the performance loss can be compensated by slightly increasing the iteration number. Nevertheless, the K-layer parallel decoder will have a roughly K-fold throughput increase compared to the conventional single-layer decoder if the same parallelism is used for processing each check row. Thus, a trade-off can be made between the layer-parallelism K, the error performance, and the throughput. The fixed-point performance will be discussed in Section V. For high rate codes, the performance degradation caused by the multi-layer algorithm is slightly worse because there are more overlapping non-zero blocks between layers.

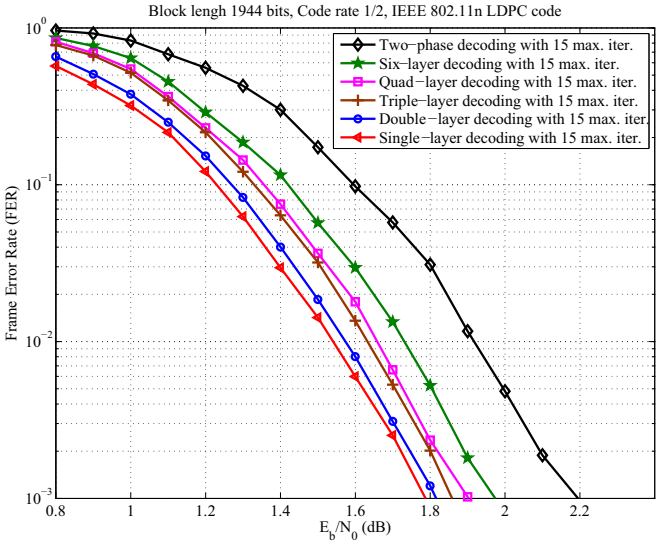

Fig. 3. Simulation results for multi-layer parallel decoding algorithm.

IV. Double-Layer Parallel Decoder Architecture for IEEE 802.11n LDPC Codes

As a case study, we have designed a double-layer parallel decoder for IEEE 802.11n LDPC codes. We propose a macroblock-serial (MB-serial) decoding algorithm. In this algorithm, a $Z \times Z$ sub-matrix is considered as a block and a macroblock (MB) contains $N_1 \times N_2$ blocks, where N_1 and N_2 are arbitrary integer numbers. Fig. 4(a) shows an example of an MB which contains 2×2 blocks: A, B, C, and D. Fig. 4(b) shows the MB view of the first two layers of the parity check matrix in Fig. 1. Because the rate 1/2 matrix is sparser than the high rate matrix, some blocks in an MB can be zero blocks. However, for a denser matrix, e.g. rate 5/6 matrix, all the four blocks in an MB are often non-zero blocks as shown in Fig. 4(c).

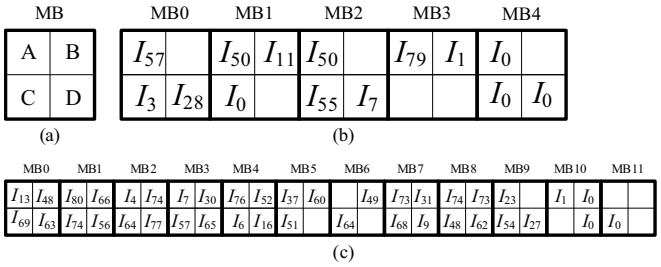

Fig. 4. (a) One MB with a dimension of $2Z \times 2Z$. (b) The MB view of the first two layers of the rate 1/2 matrix in Fig. 1. (c) The MB view of the first two layers of the matrix for rate 5/6, block length 1944 bits, 802.11n code.

We propose a partial parallel decoder architecture, where each MB is processed as a unit. Inside each macro-layer, MB is processed in serial, from left to right. Thus, we refer to this architecture as an MB-serial architecture. Fig. 5 shows the top level block diagram for the proposed MB-serial decoder architecture. In this architecture, the LLR memory is used for storing the initial and updated LLR values for each bit in a codeword. For LDPC codes with $M \times N$ sub-matrices each of which being a $Z \times Z$ shifted identity matrix, the LLR memory is organized such that Z LLR values are stored in the same memory word and there are N words in the memory. The LLR memory has two read-ports and two write-ports so that $2Z$ LLR values can be accessed at the same clock cycle. The decoding is a two-stage procedure. During the first stage, $2Z$ LLR values are read from the LLR memory at each clock cycle and are passed to four permuters A, B, C, and D, which correspond to four blocks in an MB (cf. 4(a)). Note that for zero blocks in an MB, the corresponding permuters and other related logic will be disabled.

The $2Z$ permuted LLR values L_{n_A} and L_{n_B} are fed to the even-layer's MB processing unit, and the other $2Z$ permuted LLR values L_{n_C} and L_{n_D} are fed to the odd-layer's MB processing unit. Each MB processing unit consists of $Z = 81$ min-sum units (MSUs) based on the maximum sub-matrix size defined in the IEEE 802.11n standard. Fig. 6 shows the block diagram for one MSU. Each MSU can process two LLR values at each clock cycle so that altogether Z MSUs can process $2Z$ LLR values at each clock cycle. During the first stage, Q values are computed by subtracting the R values from the LLR values based on (5). The R values are stored in a compressed way. The R-Regfile is used to store the information for restoring the $R_{m,n}$ values. Fig. 7 shows the organization of the R-Regfile. For each row m, only the first minimum (min0), the second minimum (min1), the position of the first minimum (pos), and the sign bits for all Q_{m,n_j} related to row m are stored in the R-Regfile. A R value generator (R-Gen) is used to restore the R values from

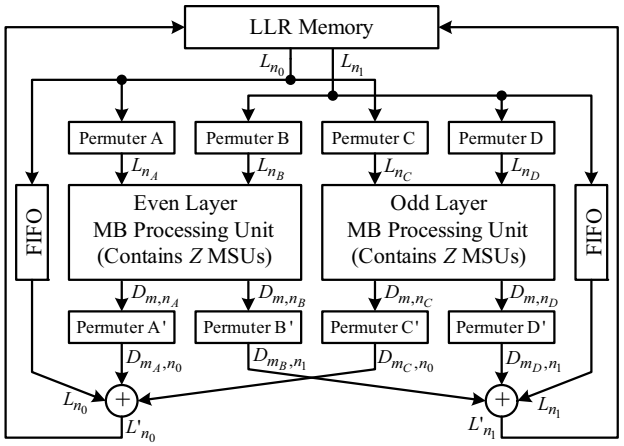

Fig. 5. MB-serial LDPC decoder architecture for the double-layer example.

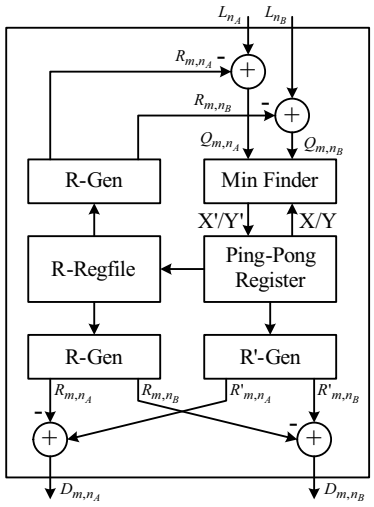

Fig. 6. Block diagram for the pipelined Min-sum unit (MSU).

the R-Regfile as:

$$|R_{m,n_j}| = \begin{cases} 0.75Y_m, & \text{if } n_j = P_m \\ 0.75X_m, & \text{otherwise,} \end{cases} \quad (8)$$

where X_m and Y_m denote the first minimum value and the second minimum value for row m, respectively, and P_m denotes the position of the first minimum value for row m. The sign bits of the R_{m,n_j} value are generated using the sign array. As the scaled min-sum algorithm is used, the R value is scaled by a factor of 0.75. A min finder unit (MFU) is used to compare the Q_{m,n_A} and Q_{m,n_B} values against X and Y read from the Ping-Pong register, where X and Y are the first minimum and the second minimum temporary variables and are initialized to be the maximum possible positive values. The two new minimum values X' and Y' are stored in the Ping-Pong register. The index of the minimum Q value and sign bits for all Q values are also updated in the Ping-Pong register. The Ping-Pong register consists of two registers (ping and pong registers), where each register has the same organization as one word of the R-Regfile. Two registers are required because we want to support pipelined decoding by overlapping two macro-layers' data processing. During the second stage, the R'-Gen unit gets values from the Ping-Pong register and restores the most recently updated R' values. Another R-Gen unit gets values from R-Regfile and restores the old R values. Then a

Index = Super-layer number				
0	Min 0	Min 1	Pos	Sign Array
1	Min 0	Min 1	Pos	Sign Array
⋮	⋯	⋯	⋯	⋯
M/2-1	Min 0	Min 1	Pos	Sign Array

Fig. 7. R-Regfile organization.

Delta-R value, denoted as D value, is formed by:

$$D_{m,n_j} = R'_{m,n_j} - R_{m,n_j}. \tag{9}$$

The R-Regfile has two read-ports so that it can be accessed simultaneously by two consecutive macro-layers. After the second stage, the contents of the Ping-Pong register is written to the R-Regfile overwriting the values for the current macro-layer, and the Ping and Pong registers switch role.

Now turning back to the top level decoder in Fig. 5, after the $2Z$ D values are produced by each MB processing unit, the D values are de-permuted and added to the LLR values from the FIFO to form the updated $2Z$ LLR values as:

$$L'_{n_0} = L_{n_0} + D_{m_A,n_0} + D_{m_C,n_0} \tag{10}$$

$$L'_{n_1} = L_{n_1} + D_{m_B,n_1} + D_{m_D,n_1}. \tag{11}$$

The new updated LLR values are then written back to the LLR memory.

To further increase the throughput, we can overlap the decoding process of two macro-layers. The pipelined data flow is illustrated in Fig. 8. The data dependencies between two macro-layers are avoided by using a scoreboard to keep track of the read and write sequences of the LLR values.

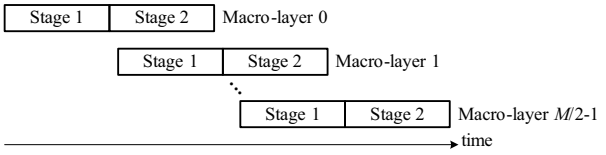

Fig. 8. Pipelined decoding data flow for the double-layer example.

It should be noted that the described double-layer parallel architecture shown in Fig. 5 and the pipelined data flow shown in Fig. 8 can be generalized for a K-layer parallel architecture by employing multiple MB processing units. The overall complexity and the throughput increase about linearly with the size of the macro-block.

V. VLSI Implementation and Comparison

A flexible double-layer parallel decoder which fully supports IEEE 802.11n LDPC codes was designed in Verilog HDL. The fixed-point design parameters are as follows. The channel input LLR is represented with 6-bit signed numbers with 2 fractional bits. The word lengths of the extrinsic R values and the APP LLR values are 6 bits and 7 bits, respectively. According to the computer simulation, this fixed-point implementation introduces a performance loss of 0.05 dB compared to the floating-point implementation at 10^{-4} FER for rate 1/2 code.

We have synthesized the decoder for a TSMC 45nm CMOS technology. The maximum clock frequency is 815 MHz and the area is 0.81 mm^2 based on the Synopsys Design Compiler synthesis result. Table I summarizes the throughput performance of this double-layer parallel decoder for the decoding of IEEE 802.11n LDPC codes at 15 iterations, where the throughput is calculated by counting the number of the clock cycles for each iteration. The throughput increases with the block size and the code rate. Table II compares the implementation result of our decoder with existing 802.11n LDPC decoders from [3], [4], [6]. The solutions from [3], [4], [6] are all based on the conventional single-layer decoding architecture. As a fair comparison, the areas of those designs are all normalized to 45nm technology. Compared to those decoders, our pipelined double-layer parallel decoder achieves a much higher throughput at low complexity.

TABLE I
Throughput performance of the proposed decoder

Block length	Rate 1/2	Rate 2/3	Rate 3/4	Rate 5/6
648 bits	380 Mbps	520 Mbps	760 Mbps	1.0 Gbps
1296 bits	750 Mbps	1.1 Gbps	1.3 Gbps	2.0 Gbps
1944 bits	1.1 Gbps	1.7 Gbps	2.2 Gbps	3.0 Gbps

TABLE II
Comparison of LDPC Decoders for IEEE 802.11n

	This work	[3]	[4]	[6]
Technology	45 nm	65 nm	130 nm	180 nm
Area	0.81 mm^2	0.74 mm^2	1.85 mm^2	3.39 mm^2
Norm. area	0.81 mm^2	0.39 mm^2	0.22 mm^2	0.21 mm^2
Clock freq.	815 MHz	240 MHz	500 MHz	208 MHz
Num. of iter.	15	14	5	5
Max. throughput	3.0 Gbps	410 Mbps	1.6 Gbps	780 Mbps

VI. Conclusion

We have presented a high-throughput multi-layer parallel decoding algorithm and architecture for QC-LDPC codes. The conventional single-layer algorithm is extended to support multi-layer parallel decoding, which leads to a significant throughput improvement. A pipelined macroblock-serial decoder is described for double-layer parallel decoding of IEEE 802.11n codes. This decoder supports a maximum throughput of 3 Gbps with an area of 0.81 mm^2.

References

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

[2] M. M. Mansour and N. R. Shanbhag, "High-throughput LDPC decoders," *IEEE Tran. VLSI*, vol. 11, pp. 976–996, Dec. 2003.

[3] M. Rovini, G. Gentile, F. Rossi, and L. Fanucci, "A Scalable Decoder Architecture for IEEE 802.11n LDPC Codes," in *IEEE Global Telecommunications Conference*, 2007, pp. 3270–3274.

[4] K. Gunnam, G. S. Choi, M. B. Yeary, and M. Atiquzzaman, "VLSI Architectures for Layered Decoding for Irregular LDPC Codes of WiMax," in *IEEE Int. Conf. on Commun.*, June 2007, pp. 4542–4547.

[5] Y. Sun, M. Karkooti, and J. R. Cavallaro, "VLSI Decoder Architecture for High Throughput, Variable Block-size and Multi-rate LDPC Codes," in *IEEE Int. Symp. on Circuits and Syst.*, May 2007, pp. 2104–2107.

[6] C. Studer, N. Preyss, C. Roth, and A. Burg, "Configurable high-throughput decoder architecture for quasi-cyclic LDPC codes," in *IEEE Asilomar*, Oct 2008, pp. 1137–1142.

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

[8] W. Jun and Y. Shu-hui, "A Parallel Layered Decoding Algorithm for LDPC Codes in WiMax System," in *IEEE WiCom*, Sept. 2009, pp. 1–4.

[9] Z. Cui, Z. Wang, and Y. Liu, "High-Throughput Layered LDPC Decoding Architecture," in *IEEE Tran. VLSI*, Apr. 2009, pp. 582–587.

[10] B. Xiang and X. Zeng, "A 4.84 mm^2 847C955 Mb/s 397 mW dual-path fully-overlapped QC-LDPC decoder for the WiMAX system in 0.13 um CMOS," in *IEEE Symposium on VLSI Circuits*, June 2010, pp. 211–212.