Area-Efficient Min-Sum Decoder Design for High-Rate QC-LDPC Codes in Magnetic Recording

Hao Zhong, Wei Xu, Ningde Xie, and Tong Zhang

***Abstract*— This paper presents a silicon area efficient quasi-cyclic (QC) low-density parity-check (LDPC) code decoder design solution, which is geared to magnetic recording that demands high code rate and very high decoding throughput under stringent silicon cost constraint. The key of this proposed design solution is to transform the conventional formulation of the Min-Sum decoding algorithm in such a way that we can readily develop a hardware decoder architecture with several desirable features, including (i) silicon area saving potential inherent in the Min-Sum algorithm for high-rate codes can be fully exploited, (ii) the decoder circuit critical path may be greatly reduced, and (iii) check node processing and variable node processing can operate concurrently. For the purpose of demonstration, we designed ASIC (application-specific integrated circuit) decoders for four rate-8/9 regular-(4, 36) QC-LDPC codes that contain 512-byte, 1024-byte, 2048-byte, and 4096-byte user data per codeword, respectively. Synthesis results show that this proposed design solution can meet the beyond-2Gbps throughput requirement in future magnetic recording at minimal silicon area cost.**

***Index Terms*— Low-density parity-check (LDPC), decoder, VLSI, magnetic recording**

I. Introduction

Due to their excellent error correcting capability with relatively low-complexity iterative decoding algorithms, low-density parity-check (LDPC) codes have recently attracted much attention for their applications in magnetic recording systems. To be adopted by real-life hard disk drives, LDPC codes must not only achieve very low sector failure rate with high code rate (e.g., 8/9 and higher), but also be suitable for high-speed (e.g., 2Gbps and higher) VLSI implementation at minimal silicon area cost. Such stringent requirements make appropriate LDPC code construction a nontrivial task. Prior work has well demonstrated that quasi-cyclic (QC) LDPC codes, one special family of LDPC codes, are a promising candidate in this context. From the error-correcting performance perspective, researchers [1]–[3] have recently implemented high-speed dedicated hardware simulators using FPGA (field programmable gate array) devices to empirically demonstrate the significant coding gain of high-rate QC-LDPC codes over current practice at sector failure rates as low as $10^{-8} \sim 10^{-9}$. Meanwhile, the inherent structural regularity of QC-LDPC codes makes it relatively easy to design efficient partially parallel decoder architectures, as demonstrated by the large amount of work recently reported, e.g., see [4]–[11]. Partially parallel decoders map a certain number of variable nodes or check nodes to a single hardware unit in a time-division multiplexed mode subject to desirable trade-offs between silicon area and decoding throughput.

However, the silicon cost of high-speed decoder implementation still remains a big concern for using QC-LDPC codes in magnetic recording. This is mainly because of the long sector length in hard disk drives, e.g., in most current hard disk drives each sector contains 512-byte user data and recently IDEMA (International Disk Drive, Equipment, and Materials Association) announced a recommendation to make one sector contain 4096-byte user data. Generally, the longer sector size enables the use of longer ECC (error correction code) that can achieve the desired level of sector failure rate at less coding redundancy (or higher code rate). However, as the penalty, longer ECC tends to incur higher decoder implementation silicon cost, particularly when targeting on very high decoding throughput. This is particularly serious for LDPC codes since the amount of decoding messages to be stored (and hence the silicon cost for data storage) is proportional to the codeword length. This work concerns the development of QC-LDPC code decoder design solutions to push the silicon area efficiency envelope under the following magnetic-recording-oriented constraints: (i) code rate is 8/9 and higher, (ii) user data per codeword may range from 512-byte up to 4096-byte, and (iii) the decoder can achieve beyond-2Gbps throughput with a reasonable number of decoding iterations (e.g., 16 iterations). To the best of our knowledge, in spite of the large amount of recent work on LDPC decoder design, very few results are readily available on decoders satisfying the above constraints, particularly for very long LDPC codes such as 4096-byte user data per codeword that remains almost completely unexplored in the open literature.

This paper presents a silicon area efficient QC-LDPC code decoder design solution geared to the above magnetic-recording-oriented constraints and demonstrates its practical potential through ASIC (application-specific integrated circuit) design at 65nm CMOS technology node. It is well-known that various LDPC decoding algorithms may fall into three categories, including Sum-Product algorithm (SPA) [12], Min-Sum algorithm [13], and layered decoding algorithm[1] [4], [9]. This work focuses on the Min-Sum algorithm because its check node processing approximation may potentially lead to significant silicon area savings from two perspectives: (i) the logic complexity may be reduced due to the elimination of the function $log[tanh(x/2)]$ that is typically implemented as look-up-table (LUT) in hardware, and (ii) more importantly, the size of memory for decoding message storage may be reduced due

Manuscript received March 2007.

H. Zhong is with LSI Corporation, San Jose, CA, USA.

W. Xu, N. Xie, and T. Zhang are with Rensselaer Polytechnic Institute, Troy, NY, USA (email: tzhang@ecse.rpi.edu)

[1]We categorize Turbo-Decoding Massage-Passing (TDMP) algorithm in [4] as layered decoding following the notation in [9].

to the possible compact representation of check-to-variable messages. However, this memory saving potential has not been fully exploited by existing high-speed partially parallel Min-Sum decoders [5], [6], although such potential has been pointed out in some serial Min-Sum decoding schemes [14], [15]. Moreover, for partially parallel decoder design, the conventional Min-Sum algorithm formulation results in explicit implementation of a sorter in each check node processing unit, which will make the potential of logic silicon area saving quickly diminish and result in an essential speed bottleneck as the code rate (or, more specifically, the row weight of parity check matrix) increases.

In this work, we developed a decoder design solution to fully exploit the potential advantages of Min-Sum decoding algorithm, particularly under the above magnetic-recording-oriented constraints. We transform (or re-formulate) the conventional Min-Sum algorithm formulation in such a way that it becomes relatively straightforward to develop a high-speed partially parallel decoder architecture with the following favorable features, including (a) the silicon area saving potential inherent in the Min-Sum algorithm for high-rate codes can be fully exploited, (b) the decoder circuit critical path may be greatly reduced, and (c) check node processing and variable node processing can operate concurrently. To demonstrate the effectiveness of this proposed design solution for magnetic recording, we designed decoders for four rate-8/9 regular-(4, 36) QC-LDPC codes that contain 512-byte, 1024-byte, 2048-byte, and 4096-byte user data per codeword, respectively. We note that QC-LDPC codes with column weight of 4 are used because prior work [1], [2], [16] showed that such codes may be much more immune to error floor than the codes with column weight of 3. All these decoders are designed using Synopsys tools and 65nm CMOS technology libraries. When the decoding messages are quantized using 4 bits, these decoders can achieve 2.1Gbps decoding throughput (with 16 decoding iterations) while occupying silicon areas of only 1.92mm^2, 2.32mm^2, 2.81mm^2, and 3.78mm^2, respectively.

The remainder of this paper is organized as follows. Section II presents the proposed Min-Sum algorithm transformation. Section III presents the developed partially parallel decoder architecture. The decoder ASIC design results are given in Section IV, and Section V draws the conclusion.

II. Proposed Min-Sum Algorithm Transformation

This section presents the proposed transformation of Min-Sum algorithm formulation. First, we introduce some conventional definitions and notations following the open literature. Given an $M \times N$ parity check matrix $\mathbf{H}$, we define the set of bits that participate in parity check m as $\mathcal{N}(m) = \{n : H_{m,n} = 1\}$, and the set of parity checks in which bit n participates as $\mathcal{M}(n) = \{m : H_{m,n} = 1\}$. We denote the set $\mathcal{N}(m)$ with bit n excluded by $\mathcal{N}(m) \setminus n$, and the set $\mathcal{M}(n)$ with check m excluded by $\mathcal{M}(n) \setminus m$. The channel message, variable-to-check message, check-to-variable message, and posterior Log-likelihood ration (LLR) are denoted as γ_n, $\alpha^i_{m,n}$, $\beta^i_{m,n}$, and λ^i_n respectively, where the superscript i is iteration index.

The main difference between SPA and Min-Sum algorithm lies in the check node processing, i.e., the check node processing in SPA is realized as

$$\beta_{m,n} = \Phi\Bigg(\sum_{n' \in \mathcal{N}(m)\setminus n} \Phi(|\alpha_{m,n'}|)\Bigg) \prod_{n' \in \mathcal{N}(m)\setminus n} sign(\alpha_{m,n'}), \tag{1}$$

where $\Phi(x) \equiv -log[tanh(x/2)]$. The check node processing in Min-Sum algorithm is approximated as

$$\beta_{m,n} = \min_{n' \in \mathcal{N}(m)\setminus n} (\alpha_{m,n'}) \prod_{n' \in \mathcal{N}(m)\setminus n} sign(\alpha_{m,n'}). \tag{2}$$

Therefore, the function $\Phi(x)$, which is typically implemented as look-up tables (LUT) in hardware, is eliminated in Min-Sum algorithm. The conventional Min-Sum algorithm formulation is described as follows.

Algorithm 1: Min-Sum Decoding Algorithm

```
Initialization: $\alpha^0_{m,n} = \gamma_n$;
for $i$=0 to $i_{max}$ or convergence to codeword do
    forall check nodes $c_m$, $m \in \{1, ..., M\}$ do
        $|\beta^i_{m,n}| = \min_{n' \in \mathcal{N}(m)\setminus n}\{|\alpha^{i-1}_{m,n'}|\}$;
        $sign(\beta^i_{m,n}) = \prod_{n' \in \mathcal{N}(m)\setminus n} sign(\alpha^{i-1}_{m,n'})$;
    end
    forall variable nodes $v_n$, $n \in \{1, ..., N\}$ do
        $\lambda^i_n = \gamma_n + \sum_{m \in \mathcal{M}(n)} \beta^i_{m,n}$;
        $\alpha^i_{m,n} = \lambda^i_n - \beta^i_{m,n}$;
    end
end
Output the decoded bits as sign($\lambda^i_n$)
```

Due to the check node processing approximation, the check-to-variable messages from each check node only have 2 different magnitudes (i.e., the minimum and the 2nd minimum ones among the magnitudes of all the variable-to-check messages entering into this check node), no matter how large the check node degree is. Meanwhile, the check node processing approximation eliminates the function $\Phi(x)$. Intuitively, these two features may be leveraged to reduce the storage and logic silicon area. However, a direct realization of partially parallel decoders based on the above conventional Min-Sum algorithm formulation, e.g., the existing partially parallel Min-Sum decoders [5], [6], may not be able to effectively materialize such silicon area saving potential for two main reasons:

1) In spite of much less check-to-variable messages storage requirement, the total number of distinct variable-to-check messages always equals to the total number of 1s in the parity check matrix. A direct realization of partially parallel decoders may have to provide explicit storage for these variable-to-check messages, leading to the same (or similar) storage requirement as in its SPA decoder counterpart.
2) The direct realization of partially parallel decoders tends to implement parallel-input parallel-output check node processing units that use a sorter to search the 2 minimum ones among all the incoming variable-to-check messages.

As code rate increases, the silicon area overhead incurred by sorters will quickly increase.

To tackle the above two issues, we propose a transformed Min-Sum algorithm described in Algorithm 2. Although it is mathematically equivalent to the original Min-Sum algorithm, its formulation and execution order make it straightforward to realize silicon area savings at VLSI architecture level. In particular, this algorithm transformation has two key features, including (1) the check node processing and variable node processing are interleaved in such a way that each newly updated variable-to-check message may be directly absorbed by check node processing units without being intermediately stored, and (2) the check node processing is sequentialized so that the explicit implementation of a sorter is eliminated. To generate the outgoing check-to-variable messages from each check node (i.e., $\{|\beta_{m,n}|, n \in \mathcal{N}(m)\}$), the sequentialized check node processing only needs to keep track of the two minimum magnitudes, i.e., $min1_m$ and $min2_m$ where $min1_m \leq min2_m$, among the input variable-to-check messages, the sign $s_{m,n}$ of each input variable-to-check message and $S_m = \prod s_{m,n}$, and the variable node index I_m representing which variable node provides the message with the minimum magnitude.

Algorithm 2: Transformed Min-Sum Algorithm

```
for i=0 to $i_{max}$ or convergence to codeword do
    forall variable nodes $v_n$, $n \in \{1, ..., N\}$ do
        if i=0 then
            $\alpha^i_{m,n} = \gamma_n$;
        else
            $\beta^i_{m,n} = \begin{cases} S^i_m \cdot s^i_{m,n} \cdot min1^i_m & n \neq I^i_m \\ S^i_m \cdot s^i_{m,n} \cdot min2^i_m & \text{otherwise} \end{cases}$
            $\lambda^i_n = \gamma_n + \sum_{m \in \mathcal{M}(n)} \beta^i_{m,n}$ ;
            $\alpha^i_{m,n} = \lambda^i_n - \beta^i_{m,n}$;
        end
        Initialize $min1^{i+1}_m = min2^{i+1}_m = +\infty, S^{i+1}_m = 1$;
        forall check nodes $c_m$, $m \in \mathcal{M}(n)$ do
            if $|\alpha^i_{m,n}| < min1^{i+1}_m$ then
                $min1^{i+1}_m = |\alpha^i_{m,n}|$;
                $min2^{i+1}_m = min1^{i+1}_m$;
                $I^{i+1}_m = n$ ;
            else
                if $|\alpha^i_{m,n}| < min2^{i+1}_m$ then
                    $min2^{i+1}_m = |\alpha^i_{m,n}|$;
                end
            end
            $s^{i+1}_{m,n} = sign(\alpha^i_{m,n})$ ;
            $S^{i+1}_m = S^{i+1}_m \cdot s^{i+1}_{m,n}$ ;
        end
    end
end
Output the decoded bits as sign($\lambda^i_n$)
```

Additionally, we note that the Min-Sum algorithm is generally less sensitive to quantization errors compared to SPA and hence may enable the use of smaller finite word-length. For example, it has been shown in [13] that a 4-bit quantization of decoding messages may be sufficient to avoid error floor, although it may result in about 0.2dB performance loss compared to 6-bit quantization.

III. Partially Parallel Decoder Architecture

This section presents the developed partially parallel QC-LDPC decoder architecture based on the above transformed Min-Sum algorithm formulation. The parity check matrix of a QC-LDPC code consists of an $m_b \times n_b$ array of $p \times p$ square circulant matrices[2]. Hence the $m_b \cdot p \times n_b \cdot p$ parity check matrix can be represented by a bipartite graph as shown in Fig. 1, where each group of consecutive p rows (or columns) are represented by a set of p check (or variable) nodes. Notice that the cyclic permutation blocks in Fig. 1 realize message passing between adjacent variable and check node groups through simple cyclic permutations based on the quasi-cyclic parity check matrix structure.

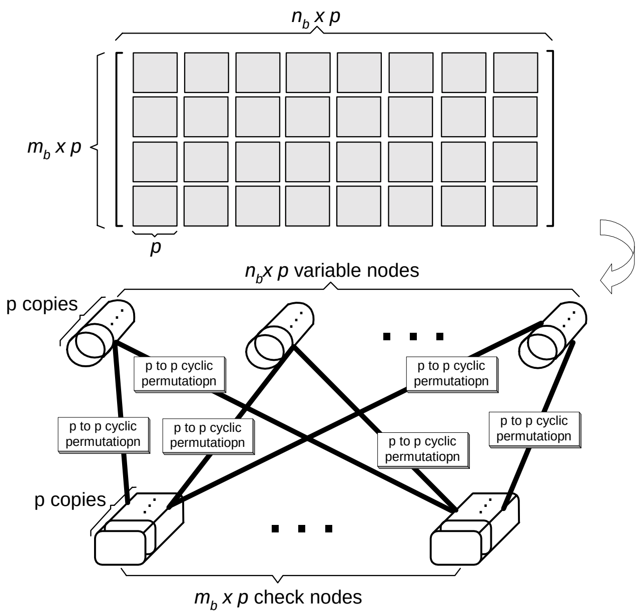

Fig. 1. Graph representation of QC-LDPC codes.

A. Decoding Scheduling

In the developed partially parallel QC-LDPC decoder, the decoding scheduling within each decoding iteration directly follows the above transformed formulation. This can be illustrated in Fig. 2 and explained as follows. One out of the total n_b variable node sets is processed at one time, and all the check nodes are processed in a serial manner interleaved with the variable node processing. In Fig. 2(a) the first set of variable nodes is processed and feeds variable-to-check messages to all the check nodes for partial check node processing. Then the decoding moves to the second step as shown in Fig. 2(b) where the second set of variable nodes is processed and feeds variable-to-check messages to all the check nodes for further partial check node processing. Once all the variable nodes are processed within present iteration,

[2]A circulant matrix is such a matrix in which each row is obtained by cyclicly shifting the row above by one position.

as shown in Fig. 2(c), all the check nodes also receive all the input variable-to-check messages and finish the check node processing for present iteration. Then all the check-to-variable messages will be fed to the variable nodes for the next iteration. Fig. 2 clearly illustrates the two desirable features of this proposed Min-Sum algorithm transformation, i.e., the obviation of explicit storage of variable-to-check decoding messages and concurrent operation of check node processing and variable node processing.

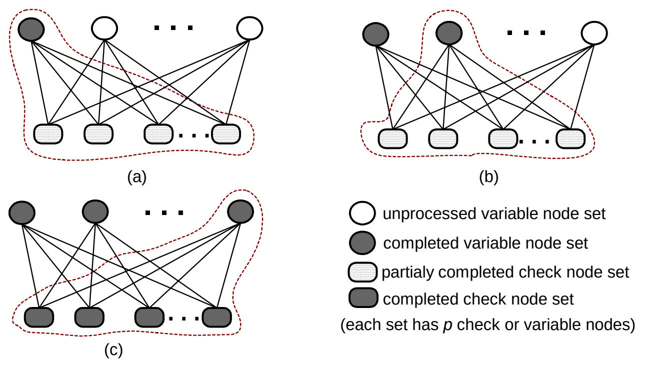
(a)
(b)
(c)
unprocessed variable node set
completed variable node set
partialy completed check node set
completed check node set
(each set has *p* check or variable nodes)

Fig. 2. Decoding scheduling within one decoding iteration.

B. Decoder Architecture

Let the QC-LDPC code parity check matrix be $m_b \cdot p \times n_b \cdot p$ and define a decoder parallelism factor s for partially parallel decoder design. Without the loss of generality, we assume p is divisible by s and denote $v = p/s$. Fig. 3 shows the principal architecture of our developed partially parallel decoder based on the proposed Min-Sum algorithm transformation. Given the decoder parallelism factor s, the decoder contains s variable node processing units (VNUs) and $m_b \cdot s$ serial check node processing units (SCNUs). The channel messages are stored in the channel message SRAM blocks (CMMB). Since all the variable-to-check messages are immediately absorbed by the concurrent check node processing as described in Section III-A, the decoder only stores the check-to-variable messages in the message storage block.

The operation of the decoder is described as follows. All the s VNUs perform the computation for s consecutive variable nodes in each clock cycle, hence each set of p variable nodes as illustrated in Fig. 2 is processed with v consecutive clock cycles. Therefore, the decoder finishes one round of variable node processing with $n_b \cdot v$ clock cycles. Upon receiving the variable-to-check messages from s variable nodes, the SCNUs partially update the decoding messages of the check nodes connected with those s variable nodes. Each group of s SCNUs are responsible to one set of p check nodes out of the total m_b sets. Once a time, each SCNU receives one incoming variable-to-check message and accordingly updates the intermediate decoding data $\{min1_m,\ min2_m,\ S_m,\ I_m\}$ (as defined in Section II). As shown in Fig. 3, one group of s SCNUs associates with one check-to-variable message storage block, each block contains one SRAM and two register arrays. In the run time, register array A stores the intermediate decoding data $\{min1_m,\ min2_m,\ S_m,\ I_m\}$ that are being updated by the SCNUs. Every $n_b \cdot v$ clock cycles, the data set $\{min1_m,\ min2_m,\ S_m,\ I_m\}$ in the register array A are completely updated and then are copied to the register array B that provides the input to all the VNUs as shown in Fig. 3. The SRAM stores the signs of incoming variable-to-check messages (i.e., all the $s_{m,n}$ as defined in Section II) that will be used to generate check-to-variable messages. Since the decoding datapath is naturally pipelined due to the use of two register arrays, all the VNUs and SCNUs can work concurrently and one decoding iteration takes $n_b \cdot v$ clock cycles. The access control for the data storage/retrieval to/from the SRAMs and register arrays can be readily derived based on the cyclic structure of the parity check matrix. Let f_{clock} denote the clock frequency of the decoder and r denote the number of decoding iterations, the decoding throughput can be estimated as

$$\frac{n_b \cdot p - m_b \cdot p}{n_b \cdot v \cdot r} \cdot f_{clock} = \frac{(n_b - m_b) \cdot s}{n_b \cdot r} \cdot f_{clock}. \tag{3}$$

The structures of each SCNU and VNU can be straightforwardly derived from the the above algorithm and architecture descriptions, as shown in Fig. 4 and Fig. 5, respectively. We use q to denote the finite word-length of each decoding message. Each SCNU receives one variable-to-check message at a time and accordingly updates the check-to-variable messages (i.e., sifts the 2 minimum magnitudes of incoming variable-to-check messages) in serial, while each VNU receives all the check-variable messages and generate the corresponding variable-to-check messages at once.

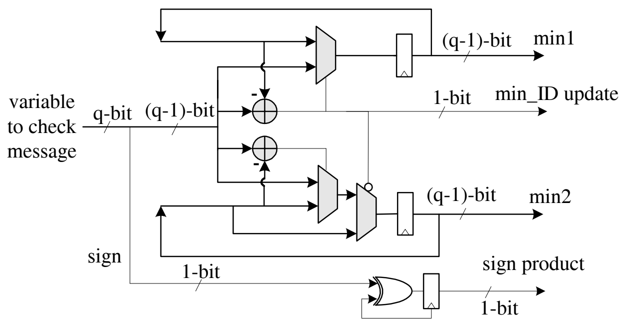

Fig. 4. Structure of one serial check node processing unit (SCNU).

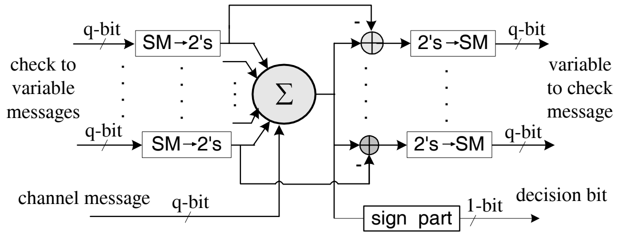

Fig. 5. Structure of one variable node processing unit (VNU).

IV. ASIC Design Results

This section presents ASIC decoder design results to demonstrate the effectiveness of the proposed decoder design

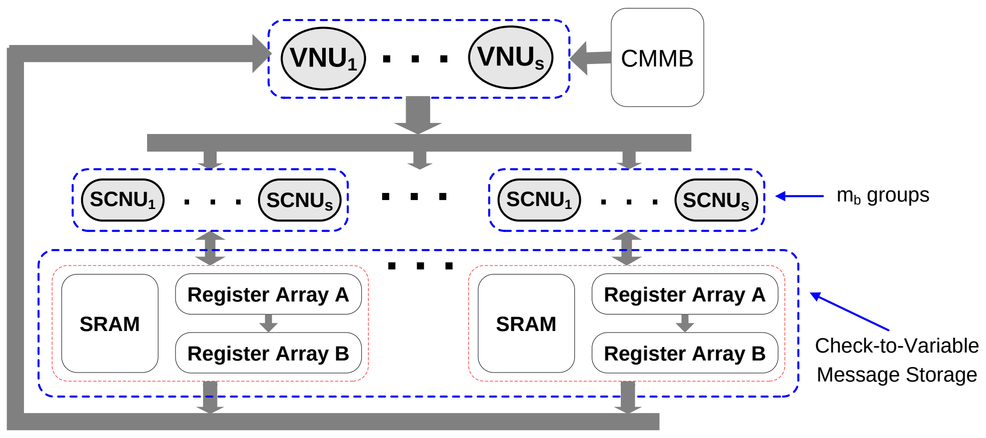

Fig. 3. Proposed partially parallel decoder architecture that contains s VNUs and $m_b \cdot s$ SCNUs.

TABLE I
DECODER DESIGN RESULTS FOR THE FOUR RATE-8/9 REGULAR-(4, 36) QC-LDPC CODES.

p	D (Bytes)	Silicon Area (mm^2)					Throughput
		VNU & SCNU	Register Array	SRAM	Others	Total	
128	512	0.26	1.21	0.25	0.20	1.92	2.1Gbps @ 300MHz
256	1024		1.62	0.29	0.15	2.32	
512	2048		1.96	0.40	0.19	2.81	
1024	4096		2.60	0.66	0.26	3.78	

solution and evaluate the trade-offs between the codeword length and silicon area cost. Prior work [1], [2], [16] demonstrated that regular QC-LDPC codes with column weight of 4 may be much less subject to error floor and be good candidates for magnetic recording. Therefore, this work focused on rate-8/9 regular-(4, 36) QC-LDPC codes whose parity check matrices are $4 \cdot p \times 36 \cdot p$ and each $p \times p$ circulant matrix has column weight of 1. Taking the circulant matrix size p as 128, 256, 512, and 1024, we constructed four QC-LDPC codes that contain 512-byte, 1024-byte, 2048-byte, and 4096-byte user data per codeword, respectively. We set the decoder parallelism factor s as 128, hence each decoder contains four SCNU groups, each group has 128 SCNUs, and one group of 128 VNUs. According to (3), the decoding throughput of these decoders is $\frac{64}{9} f_{clock}$ under 16 iterations. We designed these decoders using 65nm CMOS technology and Synopsys tools for simulation, synthesis, verification, and static timing analysis. All the decoders quantize the channel messages and decoding messages to be 4-bit. Let D denote the amount of user data in the code. Table I summarizes the ASIC design results. We note that the listed silicon area for Register Array and SRAM also includes the cost of the access controllers that control the data storage/retrieval.

Table II further shows the comparison with representative recent work on LDPC code partially parallel decoder design. For the first order approximation, when being compared against with decoders at 65-nm node, the areas of the decoders at 180-nm, 160-nm, and 130-nm listed in the table may roughly scale down by 8, 6, and 4, respectively. Considering the code length, decoding throughput (normalized with respect to the decoding iterations), and silicon area, this comparison further justifies the silicon area efficiency advantage of the proposed design solution.

V. Conclusion

In this paper, by appropriately transforming the conventional Min-Sum algorithm formulation, we develop an area efficient QC-LDPC code decoder design solution. It is particularly suitable for magnetic recording that demands long codeword length, high code rate, and very high decoding throughput. The decoder silicon cost for data storage is reduced by obviating the explicit storage of variable-to-check messages and storing the check-to-variable messages in a compact manner. The logic complexity reduction is achieved by obviating the explicit implementation of the resource-consuming data sorters in check node processing units. Furthermore, the decoder enables concurrent operation of variable node processing and check node processing, leading to further decoding throughput improvement. Its effectiveness has been demonstrated using ASIC decoder design at 65nm CMOS technology with the configurations geared to magnetic recording.

References

[1] L. Sun, H. Song, B.V.K.V. Kumar, and Z. Keirn, "Field-programmable gate-array-based investigation of the error floor of low-density parity check codes for magnetic recording channels," in *IEEE Transactions on Magnetics*, Oct. 2005, pp. 2983 – 2985.

[2] H. Zhong and T. Zhang, "High-rate quasi-cyclic LDPC codes for magnetic recording channel with low error floor," in *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2006, pp. 3546–3549.

TABLE II
COMPARISON WITH REPRESENTATIVE RECENT WORK ON LDPC CODE PARTIALLY PARALLEL DECODER DESIGN.

Reference	This work				[4]	[5]	[6]	[7]
CMOS Technology	65nm, 0.9V				180nm, 1.8V	160nm, 1.5V	180nm, 3.3V	130nm, 1.2V
Decoding Algorithm	Min-Sum				TDMP	Min-Sum	Min-Sum	SPA
Code Rate	8/9				1/2 to 7/8	3/4	3/5	1/2
Average Column Weight	4				3	3	-	3
Quantization	4-bit				4-bit	6-bit	6-bit	-
Clock Frequency	300 MHz				125MHz	264MHz	-	200MHz
Throughput	2.1 Gbit/s				640Mbit/s	480Mbit/s	5.92Gbit/s	985Mbit/s
Decoding Iterations	16				16	10	-	8
Code Length	4608	9216	18432	36864	2048	600	1200	1024
Area (mm^2)	1.92	2.32	2.81	3.78	14.3	22.4	13.5	10.08

[3] X. Hu, B. V. K. V. Kumar, L. Sun, and J. Xie, “Decoding behavior study of LDPC codes under a realistic magnetic recording channel model,” *IEEE Transactions on Magnetics*, vol. 42, no. 10, pp. 2606–2608, Oct. 2006.

[4] M.M. Mansour and N.R. Shanbhag, “A 640-Mb/s 2048-bit programmable LDPC decoder chip,” *IEEE Journal of Solid-State Circuits*, vol. 41, pp. 684–698, March 2006.

[5] H. Liu et al, “A 480Mb/s LDPC-COFDM-based UWB baseband transceiver,” in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2005, pp. 444–609.

[6] C. Lin, K. Lin, H. Chan, and C. Lee, “A 3.33Gb/s (1200,720) low-density parity check code decoder,” in *Proceedings of the 31st European Solid-State Circuits Conference*, Sept. 2005, pp. 211–214.

[7] S. Kang and I. Park, “Loosely coupled memory-based decoding architecture for low density parity check codes,” *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC)*, pp. 703–706, Sept. 2005.

[8] H. Zhong and T. Zhang, “Block-LDPC: A practical LDPC coding system design approach,” *IEEE Transactions on Circuits and Systems I*, vol. 52, pp. 766–775, April 2005.

[9] D.E. Hocevar, “A reduced complexity decoder architecture via layered decoding of LDPC codes,” *IEEE Workshop on Signal Processing Systems (SIPS)*, pp. 107–112, March 2004.

[10] J.K.-S. Lee, B. Lee, J. Thorpe, K. Andrews, S. Dolinar, and J. Hamkins, “A scalable architecture of a structured LDPC decoder,” in *International Symposium on Information Theory*, July 2004, p. 293.

[11] Z. Wang and Q. Jia, “Low complexity, high speed decoder architecture for quasi-cyclic LDPC codes,” *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 6, pp. 5786–5789, May 2005.

[12] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, Feb. 2001.

[13] J. Chen, A. Dholakia, E. Eleftheriou, M.P.C. Fossorier, and Xiao-Yu Hu, “Reduced-Complexity Decoding of LDPC Codes,” *IEEE Transactions on Communications*, vol. 53, pp. 1288–1299, Agu. 2005.

[14] F. Guilloud, E. Boutillon, and J.L. Danger, “λ-Min Decoding Algorithm of Regular and Irregular Codes,” *Proceedings of the 3nd International Symposium on Turbo Codes and Related Topics*, Sept. 2003.

[15] Z. Wu and G. Burd, “Equation based LDPC decoder for inter-symbol interference channels,” *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP ’05)*, vol. 5, pp. 757–760, March 2005.

[16] L. Sun, H. Song, and B.V.K.V. Kumar, “Error floor investigation and girth optimization for certain types of low-density parity check codes,” in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, March 2005, pp. 1101–1104.