COST-EFFECTIVE SCALABLE QC-LDPC DECODER DESIGNS FOR NON-VOLATILE MEMORY SYSTEMS

Ming-Han Chung, Yu-Min Lin, Cheng-Zhou Zhan, and An-Yeu(Andy) Wu

Graduate Institute of Electronics Engineering, National Taiwan University
Taipei, 106, Taiwan, R.O.C.

ABSTRACT

This paper presents a cost-effective scalable quasi-cyclic LDPC (QC-LDPC) decoder architecture for non-volatile memory systems (NVMS). A re-arranged architecture is proposed to eliminate the first-in-first-out (FIFO) memory in conventional decoders, where the FIFO size is linearly proportional to the codeword size. The area reduction is 18.5% compared to the conventional decoder architecture. The scalable datapaths of the proposed decoder reduce the re-design cost and enable the flexibility of using QC-LDPC codes for NVMS. A prototyping decoder with maximum codeword size of 9280 bits is implemented in TSMC 90nm CMOS technology, and the core area is only 2.52mm^2 at 138.8MHz.

***Index Terms*—**Non-volatile memory, scalable decoder, re-arranged architecture, QC-LDPC codes, TDMP algorithm.

1. INTRODUCTION

Non-volatile memory systems (NVMS) have been prevailed among many consumer electronic products, including mobile devices, computers, and the promising solid-state drives (SSDs). With the advanced manufacturing technology, the storage density of NVMS, particularly those consist of multi-level cell (MLC) flash memories, grows fast since more bits are able to be accommodated within a cell. However, the high-capacity flash memories suffer severe degradation of reliability and endurance due to higher error rate of MLC [1]. Recently, [2],[3] show the capabilities of using low-density parity-check (LDPC) codes as error-correcting codes (ECCs) to solve the reliability issues of NVMS. Compared to the conventional algebraic codes (e.g., BCH codes), LDPC codes provide much superior performance when the raw bit error rate (BER) is high in NVMS.

LDPC codes, which were invented by Gallager in 1962 [4], have received much attention in recent years for their excellent error correcting capabilities. QC-LDPC codes are an essential branch of LDPC codes since their regular cyclic structures significantly reduce the implementation complexity of encoder and decoder. Although QC-LDPC codes are regarded as to potential candidates of ECCs in NVMS, there are two issues for the decoder design. 1). In the conventional QC-LDPC decoder, the size of FIFO memory is linearly proportional to the codeword size, which is reached up to multiple kilo bytes (KB) in the NVMS. Hence, the hardware cost may be too high for practical application. 2). Contrary to the cases in advanced communication systems, there is no unified specification for QC-LDPC codes in NVMS. For example, QC-LDPC codes proposed in [2], [3] are very different in parameter setting. To reduce the re-design cost for different user-defined QC-LDPC codes in NVMS, a decoder with scalability is desirable.

In this paper, we analyze the cost of conventional QC-LDPC decoder when applied to NVMS. It is shown that the conventional architecture can be re-arranged for a FIFO-free architecture, and a large amount of area cost is reduced. To achieve the scalability, scalable datapaths are also designed for the proposed decoder architecture.

The rest of the paper is organized as follows. Section 2 introduces the fundamental of QC-LDPC codes and the decoding algorithms. Section 3 presents the proposed re-arranged decoder architecture and the scalable datapath design. Section 4 presents the implementation results and comparison. Section 5 concludes this paper.

2. QC-LDPC CODES AND DECODING ALGORITHMS

2.1 Fundamental of QC-LDPC Codes

A binary LDPC code is a linear block code defined by the null space of a sparse parity-check matrix $\mathbf{H}$ as $\mathbf{Hx}^T=\mathbf{0}$, where $\mathbf{x}$ is a codeword and $\mathbf{0}$ is a zero vector. LDPC codes can also be represented by a bipartite graph, where each column and row of $\mathbf{H}$ represents a bit node and a check node and is connected by an edge corresponding to the non-zero entries in $\mathbf{H}$.

For a QC-LDPC code, the parity-check matrix is generally represented by an $M \times N$ array of circulants $\mathbf{A_{ij}}$ as

$$\mathbf{H} = \begin{bmatrix} \mathbf{A}_{11} & \cdots & \mathbf{A}_{1N} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{M1} & \cdots & \mathbf{A}_{MN} \end{bmatrix}. \quad (1)$$

A circulant matrix is a $b \times b$ square matrix, where each row is the cyclical right-shift of previous row. If the row weight and the column weight of a circulant are both zero, the circulant is an all-zero matrix. On the other hand, if the row weight and the column weight of a circulant are both one, the circulant is simply the cyclical right-shift of an identity matrix by an offset d. The structured parity-check matrix and the cyclical-shift property of circulant significantly simplify the hardware implementation of QC-LDPC decoder compared to other non-structure LDPC codes.

2.2 Decoding Algorithms of LDPC Codes

The iterative decoding algorithms of LDPC codes are base on the belief propagation algorithm, where the messages from the bit nodes and the check nodes are updated and passed along the edges iteratively. According to different bit nodes and check nodes scheduling techniques, the iterative decoding algorithms can be categorized into two types: 1) the two-phase message passing (TPMP) algorithm and 2) the turbo-decoding message passing (TDMP) algorithm [5].

In the TPMP algorithm, each iteration is composed of two phases, the check node updating and the bit node updating, and all of the bit nodes and check nodes participate in the updating operation in both phases. The sum-product algorithm (SPA) proposed by Gallager is a kind of the TPMP algorithm. To further reduce the complexity of hyperbolic function in the SPA, various kind of approximated algorithms are proposed, such as the min-sum algorithm and the normalized min-sum algorithm [6].

The TDMP algorithm executes the updating operations in a row-by-row fashion. Therefore, the check node updating and the bit node updating are executed locally within the corresponding row. The decoding of the TDMP algorithm is described as follows. Let the posterior message of kth bit in a codeword defined by the log-likelihood ratio (LLR) as

$$L(b_k) = \log\left(\frac{\Pr(b_k = 0 \mid y_k)}{\Pr(b_k = 1 \mid y_k)}\right), \qquad (2)$$

where y_k is the received signal for b_k. The extrinsic messages corresponding to the bit nodes (nonzero entries) in row i are denoted as $\boldsymbol{\lambda}^i = [\lambda_1^i, ..., \lambda_{r_i}^i]$ and initialized as zero, where r_i is the row weight of row i. $\boldsymbol{\gamma} = [\gamma_1, ..., \gamma_n]$ is denoted as the n posterior messages corresponding to the n bit nodes in **H,** where $n = N \times b$ in QC-LDPC codes, and γ_k is initialized with $L(b_k)$. In addition, the posterior messages of check node i are denoted as $\boldsymbol{\gamma}(\mathbf{I}_i)$, where $\mathbf{I}_i$ is the indexes of the neighboring bit nodes of check node i (i.e., the position of nonzero entries in the row i). In the TDMP algorithm, row updating operation over all rows of H constitutes iteration. Each row operation is composed of four steps as follows.

1). Read and subtract: The extrinsic messages vector $\boldsymbol{\lambda}^i$ and the posterior messages $\boldsymbol{\gamma}(\mathbf{I}_i)$ for row i are read. Then $\boldsymbol{\lambda}^i$ are subtracted from $\boldsymbol{\gamma}(\mathbf{I}_i)$ to generate prior messages $\boldsymbol{\rho} = [\rho_1, ..., \rho_{r_i}] = \boldsymbol{\gamma}(\mathbf{I}_i) - \boldsymbol{\lambda}_i$.

2). Decode: Decode prior messages $\boldsymbol{\rho}$ of row i using a soft-input soft-output (SISO) algorithm with $\boldsymbol{\rho}$ as input and $\boldsymbol{\Lambda}$ as output. In this work, we adopt check node updating equation of normalized min-sum algorithm

$$\Lambda_j = 0.5 \cdot \{XOR\{sgn(\rho_{i'})\} \cdot \min\{|\rho_{i'}|\}\}, \qquad (3)$$

where $j = 1, ..., r_i$ and $i' = 1, ..., r_i$ except j, as the SISO function.

3). Write back to check node: The extrinsic message vector $\boldsymbol{\lambda}^i$ are update with the newly generated $\boldsymbol{\Lambda}$ vector in step 2).

4). Write back to bit node: The partial posterior messages $\boldsymbol{\gamma}(\mathbf{I}_i)$ corresponding to the row i are updated by adding $\boldsymbol{\Lambda}$ to $\boldsymbol{\rho}$.

Compared to the TPMP algorithm, the row-by-row updating in the TDMP algorithm leads to significant memory reduction since there is no intermediate message needed to be stored. Moreover, the refined posterior messages from earlier row as input of subsequent rows helps the decoding converges twice as fast as the standard TPMP algorithm [5]. As a result, we adopt the TDMP algorithm as the kernel of the proposed decoder design.

For decoding QC-LDPC codes with the TDMP algorithm, since there is no data dependency between rows within a circulant, the row operation can be executed in parallel. That is, b rows within a block row of circulant can be decoded at the same time.

3. DECODER ARCHITECTURE

3.1 Re-arranged Architecture

The conventional block-serial QC-LDPC decoder architecture of the TDMP algorithm [7],[8] is shown in Fig.1. The decoder process a block (i.e., a circulant in **H** of QC-LDPC codes) within a cycle and each block row updating is accomplished until N blocks are processed. Hence, there are total $M \times N$ cycles in each iteration. The posterior messages $\boldsymbol{\gamma}$ generated during decoding are stored in the posterior message memory. Note the LLR of input are only used to initialize the posterior message vector. There

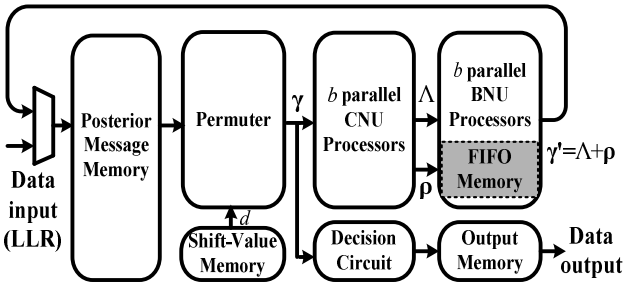

Fig 1. The block diagram of conventional QC-LDPC decoder.

are total b check node units (CNUs) and b bit node units (BNUs) work concurrently in the CNU processor and the BNU processor, respectively. When the decoding is finished, the hard-decision bits, and the decoded bit stream is stored in the output memory.

The timing diagram of CNU processor and BNU processor in the conventional decoder is shown in Fig. 2. It takes N cycles for CNU processor to perform the SISO function on $\boldsymbol{\rho}$ to generate $\boldsymbol{\Lambda}$ for each row in block row i. When the CNU processer finishes processing, the BNU processor starts to update the posterior messages by adding $\boldsymbol{\Lambda}$ to $\boldsymbol{\rho}$ for each row in block row i with other N cycles. The decoding schedule can be almost fully-overlapped (with an idle cycle for memory access) since the CNU processor is able to process next block row i+1 after the first circulant of block row i is updated by the BNU processor.

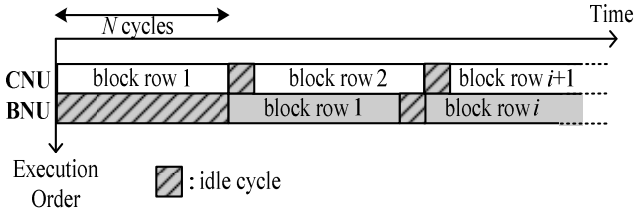

Fig 2. Timing diagram of the conventional decoder.

However, when the CNU processor in the conventional decoder is running, the prior message $\boldsymbol{\rho}$ has to be buffered in the FIFO memory for BNU processor to update later, as shown in Fig. 1. The size of the FIFO memory is equal to (*bit width of prior message*) × (*codeword size*). In the NVMS, the codeword size is generally equal to the size of a page, which is reached up to multiple KBs (1KB =1024 bits). Such a large size FIFO memory causes the high area cost of the conventional QC-LDPC decoder with the TDMP algorithm when applied in NVMS.

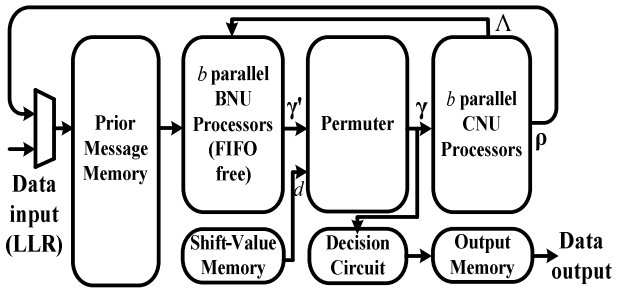

Fig 3. The block diagram of proposed FIFO-free QC-LDPC decoder with re-arranged architecture.

To eliminate the FIFO memory for a cost-effective decoder design in NVMS, we proposed a re-arranged architecture for QC-LDPC decoder as show in Fig. 3. The key idea is that instead of storing the posterior messages $\boldsymbol{\gamma}$ and the prior messages $\boldsymbol{\rho}$ as in the conventional decoder, only prior messages $\boldsymbol{\rho}$ are stored in the proposed architecture. The total memory saving is equal to (*bit width of posterior message*) × (*codeword size*). The position of permuter, CNU processor, and BNU processor is also re-arranged to keep consistent with the data flow in the TDMP algorithm. The timing diagram of the CNU processor and the BNU processor in the proposed decoder is shown in Fig. 4. When the CNU processor finishes processing block row i, the BNU processor loads the prior messages from prior message memory and updates the posterior messages of block row i. Without writing back, the posterior messages of block row i is directly passed to the permuter and CNU processor for row updating of block row i+1. Compared to the conventional decoder, the proposed architecture not only reduces the area cost of memory, but also let the timing schedule of decoder be fully overlapped.

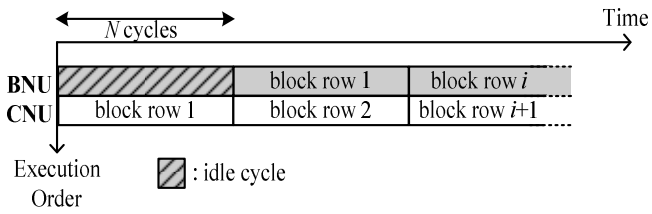

Fig. 4. Timing diagram of the proposed decoder.

3.2 Scalable Decoder Design

In QC-LDPC codes, the parameter M, N defines the size of the array of circulants, and b is the size of a circulant. In the proposed QC-LDPC decoder, the boundary conditional of control signal related to the size of M and N can be run-time programmable for different size of M and N. Scalable datapaths are design for the proposed decoder: each CNU and BNU in the CNU processor and BNU processor is turned on/off by an enable signal controlled by the actual value of b. We follow the approach in [9] to design a scalable permuter, which can cyclically shift the input data vector by an arbitrary shift-value d smaller than b. The shift value of each circulant is also run-time programmable and stored in a shift-value memory in permuter.

4. IMPLEMENTATION RESULTS

The fixed-point analysis of the TDMP algorithm is performed to determine the bit width of posterior messages γ, the prior message ρ, and the output of the SISO function Λ. After exhausted computer simulation, the messages, γ, ρ, and Λ, are quantized to 8 bits, 7 bits, and 6 bits, respectively. Fig.5 shows the error-correcting performance of a rate-0.896 LDPC code with (M, N, b) = (6, 58, 160) decoding by the floating-point and the fixed-point TDMP algorithm at 8 iterations.

A prototyping scalable decoder is designed with saving of the proposed decoder is equal to (*bit width of posterior message*) × (*maximum codeword size*) = 8 × 9280 = 74240 bits in total. The experimental scalable decoder is

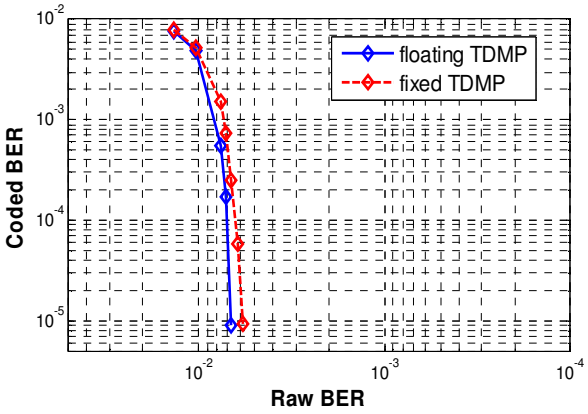

Fig. 5. Error-correcting performance of a rate-0.896 LDPC code with (M, N, b) = (6, 58, 160).

synthesized by TSMC 90nm 1P9M process. The gate count comparison of the scalable decoder is shown in Table II. The proposed re-arranged architecture achieves 18.5% area reduction compared to conventional architecture. The comparison of other state-of-the-art scalable QC-LDPC decoder is summarized in Table III. To make a fair comparison, we define the normalized area-efficiency (NAE), as a performance index.

$$NAE = \frac{Throughput}{Core\ Area \cdot Frequency} \cdot \left(\frac{Technology}{90nm} \right)^2 \ (\text{bits/mm}^2) \quad (4)$$

As shown in Table III, the proposed design has the highest NAE, and thus is a cost-effective design.

TABLE. I. PARAMETER OF THE EXPERIMENTAL DECODER

Supported parameter space	$1 \leqq M \leqq 6$ $1 \leqq N \leqq 58$ $1 \leqq b \leqq 160$
Maximum codeword size	9280 bits

TABLE. II. GATE COUNT COMPARISON OF DIFFERENT ARCHITECTURE WITH SCALABILITY

Architecture	Gate Count	Reduction %
Conventional	748K	—
Proposed	609K	18.5

5. CONCLUSION

A cost-effective scalable QC-LDPC decoder with the TDMP algorithm for NVMS has been presented. The proposed re-arranged architecture for a cost-effective design achieves 18.5% area reduction. The scalable datapaths also make the decoder flexible to support various used-defined QC-LDPC codes in NVMS. A prototyping decoder is implemented in TSMC 90nm process with operating frequency of 138.8MHz. The core area is 2.52mm^2 with maximum throughput of 393Mbs at 8 iterations, which shows high area efficiency and is cost effective compared to other state-of-the-art scalable QC-LDPC decoders.

TABLE III. COMPARISON OF DIFFERENT SCALABLE QC-LDPC DECODERS

	Proposed	[8]	[10]
Process	TSMC90nm	TSMC0.18um	TSMC0.13um
Algorithm	TDMP	TDMP	TPMP
Core Area	2.52mm^2	11 mm^2	2.46mm^2
Frequency	138.8MHz	125MHz	125MHz
Max Throughput	393Mb/s @8iter	640Mb/s @10iter	86Mb/s @8iter
Max Codeword	9280 bits	2048 bits	1536 bits
Support Parameter Space	$1 \leqq M \leqq 6$ $1 \leqq N \leqq 58$ $1 \leqq b \leqq 160$	$1 \leqq M \leqq 16$ $1 \leqq N \leqq 32$ $1 \leqq b \leqq 64$	$1 \leqq M \leqq 6$ $1 \leqq N \leqq 12$ $1 \leqq b \leqq 128$
NAE	1.12 bits/mm^2	0.57 bits/mm^2	0.57 bits/mm^2

6. REFERENCES

[1] N. Mielke et al., "Bit Error Rate in NAND Flash Memories", in *IEEE Int. Reliability Physics Symposium. (IRPS)*, pp. 9-19, Apr, 2008.

[2] S.-L. Chen et al., "Reliability analysis and improvement for multi-level non-volatile memories with soft information," *ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp.753-758, June 2011.

[3] J. Wang et al., "Soft Information for LDPC Decoding in Flash: Mutual-Information-Optimized Quantization," in p*roc. IEEE Global Communications Conference (GLOBECOM)*, Houston, TX, Dec. 5-9, 2011.

[4] R. Gallager, "Low-Density Parity-Check Codes," *IRE Trans. Inf. Theory*, vol. 7, pp. 21–28, Jan. 1962.

[5] M. M. Mansour "A turbo-decoding message-passing algorithm for sparse parity-check matrix codes", *IEEE Trans. Signal Process*., vol. 54, pp.4376-4392, Nov. 2006.

[6] J. Chen et al, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun*., vol. 53, no. 8, pp. 1288-1299, Aug. 2005.

[7] K. Gunnam et al, "VLSI architectures for layered decoding for irregular LDPC codes of WiMax," in *Proc. IEEE ICC*, 2007, pp. 4542–4547.

[8] M. M. Mansour and N. R. Shanbhag, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," *IEEE J. Solid-State Circuits*, vol.41, no. 3, pp. 634–698, Mar. 2006.

[9] Bo Xiang et al., "An 847–955 Mb/s 342–397 mW Dual-Path Fully-Overlapped QC-LDPC Decoder for WiMAX System in 0.13 m CMOS," *IEEE J. Solid-State Circuits*, vol.46, no.6, pp.1416-1432, June 2011.

[10] X.-Y. Shih et al, "A Real-Time Programmable LDPC Decoder Chip for Arbitrary QC-Based Parity Check Matrices," in *Proc. IEEE Asian Solid-State Circuits Conf. (ASSCC)*, Taipei, Taiwan, pp. 369-372, Nov. 2009,