IEEE Open Journal of
Circuits and Systems

Received 6 June 2022; revised 2 August 2022; accepted 23 August 2022. Date of publication 1 September 2022; date of current version 22 September 2022.

Digital Object Identifier 10.1109/OJCAS.2022.3203849

A 38.64-Gb/s Large-CPM 2-KB LDPC Decoder Implementation for NAND Flash Memories

LI-WEI LIU (Member, IEEE), MU-HUA YUAN, YEN-CHIN LIAO (Member, IEEE), AND HSIE-CHIA CHANG (Senior Member, IEEE)

Institute of Electronics, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan

This article was recommended by Associate Editor P. Amato.

CORRESPONDING AUTHOR: L.-W. LIU (e-mail: willvegapunk.ee06g@nctu.edu.tw)

This work was supported by the National Science and Technology Council (previously the Ministry of Science and Technology) under Grant 110-2622-8-A49-007-SB, Grant 110-2221-E-A49-157, and Grant 111-2221-E-A49-153.

ABSTRACT The routing congestion over a QC-LDPC decoder with a large circular permutation matrix (CPM) size has long been an obstacle to high throughput designs. This paper presents a large-CPM congestion-free decoder for (18396, 16416) quasi-cyclic Euclidean geometry low-density parity-check (QC-EG-LDPC) code in NAND flash application. Considering area efficiency in scheduling schemes and the array dispersion structure, the *Array-Disperse Based Dual Variable Node Unit (VNU) Cluster Architecture* fully leverages the code structure to support at least two physical channels of the Open NAND Flash Interface 5.0 (ONFI 5.0). In addition, the proposed congestion-aware analysis and implementation method achieve a highly parallel decoder at a 70% utilization ratio. Implemented in TSMC 28nm process, the presented decoder provides 38.64 Gbps throughput at RBER=1.456% Bi-AWGN channel with an area of 2.97 mm^2.

INDEX TERMS LDPC, decoder, NAND, SSD.

I. INTRODUCTION

SOLID-STATE DRIVE (SSD), a fast-growing nonvolatile storage device, progressively becomes an essential component in versatile applications such as data centers, family entertainment systems, and autonomous vehicles. To achieve high throughput while maintaining data reliability, an error correction code (ECC) plays a significant role in a NAND flash controller.

As time goes by, NAND flash manufacturers apply process scaling, multi-level cell, and 3D NAND structure to meet the growing demand for storage capacity and affordable prices. The reduction of data retention time [1], degrading endurance of program-erase cycle count [2], [3], and the layer-to-layer process variation [4] in high-density 3D-NAND structure drive the NAND a more reliable ECC in flash controllers. Consequently, the SSD system adopts the LDPC code [5] as a mainstream coding scheme and sets the raw bit error rate (RBER) to 1% as the ECC correctability threshold in the latest 3D TLC or QLC NANDs [6].

Featuring its soft-decision iterative decoding capability, the LDPC code design for the NAND flash application possesses its domain-specific consideration in code length, code rate, error floor, and decoding speed. On account of the smallest sector size as 512B supported by the file system, the user data ranges from 1KB, 2KB, and even 4KB in the corresponding LDPC codes. Considering the finite page size of the NAND flash, the parity size would be limited to the remaining space of each page, which constraints the code rate of the LDPC code. For example, given the TLC NAND flash page size in [7], the code rate should be at least 0.88 to accommodate 16KB of user data in one flash page. In addition, the error floor phenomenon [8], which stops improving error correction performance as expected in better channel conditions, is another concern in the LDPC code. References [9] and [10] propose a practical code construction method for the storage system to avoid trapping sets with short cycles and achieve uncorrectable bit error rate (UBER) $< 10^{-10}$ without an error floor.

Fig. 1 shows the general SSD controller's system architecture, which utilizes multiple independent physical channels to communicate individually with the NAND flash package. To comply with the UBER requirement for the lifetime of the

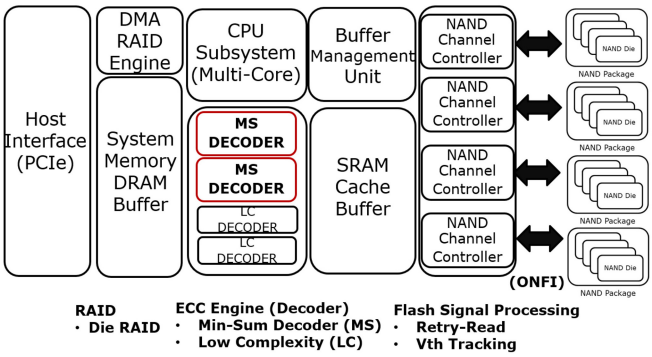


FIGURE 1. System Architecture of the SSD Controller.

SSD device in JEDEC-218 [11], a system designer applies flash signal processing, ECC, CRC check, and RAID recovery in the SSD controller. In the early stage of NAND Flash, the low complexity decoder engines in the hard-decision channel [12], [13], [14] are employed for high-speed and low-power concerns. When the channel condition degrades, the Min-Sum decoders are mainly activated for more robust error recovery. Moreover, the successfully decoded codewords are further streamed to CRC checker for data integrity. If the CRC-check fails or decode fails, the SoC applies flash signal processing such as retry read [15], [16] and Vth tracking [17], [18] to adjust the channel condition to re-decode. Finally, RAID recovery would be the final resort if all the combinations of mechanisms fail to recover the data.

In addition to the reliability issues mentioned earlier, the throughput requirement is the other primary concern for ECC engines nowadays. Considering interconnect network complexity among multiple engines to support different combinations of decoding flow while exploiting the parallelism of the NAND interface, a general SSD controller requires each min sum decoder engine to support at least two physical channels.

Accompanied by the increasing demand for high-throughput NAND flash devices, the Open NAND Flash Interface Working Group moves the Open NAND Flash Interface (ONFI) standard into the latest 5.0 [19] with a maximum speed of over 2400 MB/s per channel (i.e., the qualified two-channel decoder throughput should support at least 4800 MB/s). The high-speed LDPC min-sum decoder would become the design challenge over the early aging NAND channel and the increasing transmission rate. Although several publications address the high-speed decoder architecture, such as un-folding fully parallel architecture [20], [21] and frame-interleaving pipeline architecture [22], [23], the large code length and high code rate structure in NAND flash applications lead to formidable implementation costs.

Over the years, several publications have applied different optimization methods to the specific LDPC code to meet the NAND flash constraints. Reference [24] uses different saturation values and non-uniform quantization methods to implement a layer decoder. Reference [25] proposes the top-down approach to optimize the small sub-matrix size Array Dispersion LDPC (AD-LDPC) decoder with hybrid-storage architecture to meet the ONFI 2.3 interface throughput requirement. Reference [26] uses the graph-coloring method and linear programming method to achieve the low-area and minimized-stall pipeline block-parallel layer decoder. Reference [27] exploits the highly structured Dispersed Array LDPC code (DA-LDPC) to implement the column-based shuffle decoder without a barrel shifter. Reference [28] adopts the adaptive voltage frequency scaling scheme in the decoder architecture implemented in the 28nm process.

Although most of the literature on quasi-cyclic LDPC (QC-LDPC) decoders adopts a small circular permutation matrix (CPM) size in concern of routing congestion, we intend to leverage large CPM in quasi-cyclic Euclidean geometry LDPC (QC-EG-LDPC) code and evaluate the feasibility in decoder design for nowadays NAND flash applications. The contributions of this paper are listed as follows:

1) This paper provides a detailed quantitative analysis of two decoding scheduling methods to evaluate a suitable architecture for the target LDPC code.
2) An Array-Disperse Based Dual Variable Node Unit (VNU) Cluster Architecture is adopted for a higher parallelism level. It reveals the merit of scalability over the array dispersion method [29], [30] from a hardware perspective.
3) A congestion analysis would guide better floor planning and contribute to a congestion-free physical implementation over QC-LDPC code with a large CPM size.

The remainder of this paper is organized as follows. Section II introduces the background knowledge in this paper. Section III focuses on the decoding scheduling schemes and the quantitative analysis of the decoders. The proposed architecture and the congestion-aware physical implementation are addressed in Sections IV and V. Finally, Sections VI and VII present the implementation results and conclusion.

II. BACKGROUND

This section presents an overview of LDPC code and normalized min-sum algorithm (NMSA). Then, we briefly introduce the target QC-EG-LDPC code. The notation adopted in this paper is summarized in Table 1.

A. REVIEW OF LOW-DENSITY PARITY-CHECK CODE

An $M \times N$ sparse parity check matrix $\mathbf{H}$ defines an LDPC code, which resides in the null space of $\mathbf{H}$. Visualized in the bipartite graph (also known as Tanner graph), the ones in the parity check matrix can be interpreted as the connecting networks between M check nodes (CNs) in rows and N variable nodes (VNs) in columns. A cycle in a Tanner graph is deemed as a finite set of connected edges that starts and ends at the same node. A well-constructed LDPC code should avoid the cycle four structure, which limits the

TABLE 1. Notation table.

Symbol Notation	Description
N	code length, which also determines the number of variable nodes
M	parity length, which also determines the number of check nodes
K	information length (payload)
CPM size	the size of the circular permutation matrix in a QC-LDPC code
VN	variable nodes
CN	check nodes
dv	column degree, the number of ones in each column
dc	row degree, the number of ones in each row
$N_v(j)$	the set which contains the ith VN's neighboring nodes
$N_c(j)$	the set which contains the jth CN's neighboring nodes
x_i	the ith estimated code bit
r_i	the ith received signal
s_j	the jth syndrome
T_{max}	maximum iteration
t	iteration count
$L_{ch}(r_i)$	log-likelihood ratio (LLR) of the i th received signal (also known as channel value)
$L^t_{v2c}(i,j)$	message from the ith VN to jth CN at t iteration (also known as V2C message in LLR format)
$L^t_{c2v}(j,i)$	message from the jth CN to ith VN at t iteration (also known as C2V message in LLR format)
$L^t_{post}(i)$	a posteriori LLR for the ith VN at t iteration
α	the scaling factor to compensate the approximation loss
min1(j)	the minimum value of all $\|L^t_{v2c}(i,j)\|$ where $i \in N_c(j)$
min2(j)	the 2nd minimum value of all $\|L^t_{v2c}(i,j)\|$ where $i \in N_c(j)$
min1-index(j)	the index of min1(j)
min2-index(j)	the index of min2(j)
global sign(j)	the product of all $sign(L^t_{v2c}(i,j))$ where $i \in N_c(j)$
V2C Sign	sign information of V2C message ($sign(L_{v2c})$)
Preq.L_{post}	prerequisite information to generate $L^t_{post}(i)$
Preq. Mag.	prerequisite magnitude information to generate $L^t_{c2v}(j,i)$ such as min1, min2, min1-index, and min2-index
Preq. Sign	prerequisite sign information to generate $L^t_{c2v}(j,i)$ such as global sign and V2C sign
BS	barrel shifting operation to align VNs related information to CNs
inv BS	inverted BS operation to align CNs related information to VNs
UBER	uncorretable bit error rate (the error rate after applying ECC)
RBER	raw bit error rate (the error rate before applying ECC)

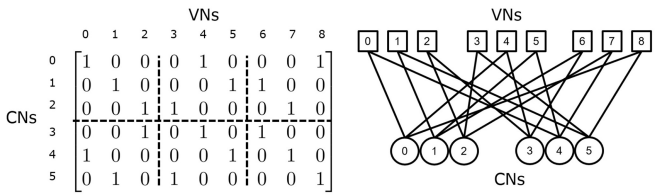


FIGURE 2. QC-LDPC Matrix & Tanner Graph.

information exchange between two sets of nodes. This rule is also known as the Row-Column constraint (RC-constraint). Column degree (dv) and row degree (dc) represent the number of edges in one column and row. If all row degree and column degree are fixed with constant dc and dv, we view these **H** matrices as regular LDPC and the other as irregular LDPC.

A QC-LDPC code introduces the quasi-cycle structure and divides the LDPC code into $B_M \times B_N$ of circular permutation matrices with size Z. It allows the decoder to operate a cluster of VNs or CNs with a size of Z as a unit for higher parallelism. The example in Fig. 2 shows a regular QC-LDPC code with CPM of size 3, $dc = 3$, and $dv = 2$.

B. REVIEW OF NORMALIZED MIN-SUM DECODING ALGORITHM

Since the normalized min-sum algorithm (NMSA) is one of the well-known approximate belief-propagation decoding algorithms for hardware implementation, the detail of NMSA is shown in algorithm 1.

Algorithm 1 Normalized Min Sum Algorithm

Initialize
1: All $L_{ch}(r_i) = \log(\frac{Pr(x_i=0|r_i)}{Pr(x_i=1|r_i)})$ and $L^{-1}_{c2v}(j,i) = 0$
2: **for** t= 0 to $T_{Max} - 1$ **do**
3: **for** $i = 0$ to $N - 1$ **do** //Variable Node Process
4: $L^t_{post}(i) = L_{ch}(r_n) + \sum_{j' \in N_v(i)} L^{t-1}_{c2v}(j', i)$
5: **if** $L^t_{post}(i) >= 0$ **then**
6: $x_i = 0$
7: **else**
8: $x_i = 1$
9: **for** $j \in N_v(i)$ **do**
10: $L^t_{v2c}(i,j) = L^t_{post}(i) - L^{t-1}_{c2v}(j,i)$
11: **for** $j = 0$ to $M - 1$ **do** //Check Node Process
12: $min1(j),\ min2(j) = min(\{|L^t_{v2c}(i,j)||i \in N_c(j)\})$
13: $min1\text{-}index(j) = argmin(\{|L^t_{v2c}(i,j)||i \in N_c(j)\})$
14: $global\ sign\ (j) = \prod_{j' \in N_c(j)} sign(L^t_{v2c}(i,j'))$
15: **for** $i \in N_c(j)$ **do**
16: **if** $i == min1\text{-}index(j)$ **then**
17: $Mag = \alpha \times min2(j)$
18: **else**
19: $Mag = \alpha \times min1(j)$
20: $Sign = global\ sign(j) \times sign(L^t_{v2c}(i,j))$
21: $L^t_{c2v}(j,i) = Mag \times Sign$
22: **for** $j = 0$ to M-1 **do** //Syndrome Calculation
23: $s_j = \bigoplus_{v \in N(c_j)} x_v$
24: **if** s **== 0** **then** //Early Termination
25: break

The decoding algorithm interprets the VNs and CNs as repetition code decoders and parity check code decoders, respectively. These two sets of distributed sub-decoders exchange soft information and achieve reliable performance in the iterative decoding process.

In the variable node process, the L_{post} combines the extrinsic L_{c2v} from connected CNs and an intrinsic L_{ch} message to determine the estimated code bit (x) from lines 3 to 8. Then, VNs generate the corresponding extrinsic L_{v2c} messages for each connected CN by excluding the incident L_{c2v} on line 10. In the check node process, the extrinsic L_{c2v} generation is separated into magnitude and sign operation from lines 12 to 20. The magnitude operation preserves the two minimum values (min1, min2) and the minimum index (min1-index) among connected L_{v2c} messages so that it could pick the smallest value excluding the incident L_{v2c} to represent the reliability of L_{c2v}. The global sign on line 14 is also preserved to implement the sign operation, which represents the L_{c2v}'s inclination, by excluding the incident sign of L_{v2c} as stated on line 20. In the end, the latest syndrome is updated and checked for early termination.

C. 2K QC-EG-LDPC CODES FOR NAND FLASH MEMORIES

With the advancement of the NAND technology, LDPC code construction becomes more challenging to support a large

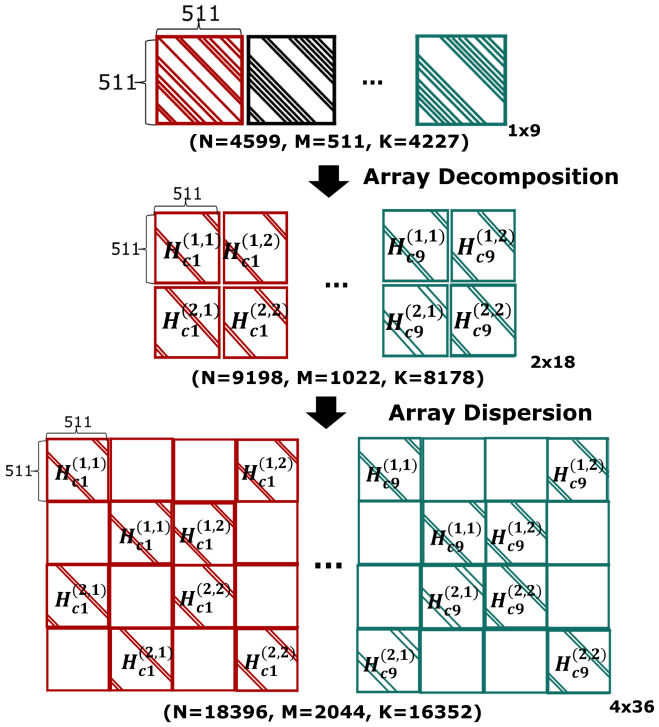


FIGURE 3. Construction of target 2KB QC-EG-LDPC Code.

user data size (2KB), high code rate (above 0.88), and a low UBER requirement. This subsection narrates the QC-EG-LDPC code with a large CPM size applied in this paper.

Among several code construction methods [31], [32], [33], [34], [35], Euclidean geometry-LDPC code is one of the algebraic LDPC codes with a low error floor. As a result, we adopt the approach in [34], [35] to construct a 1×9 array of eight-circulant permutation matrices as a prototype matrix (N=4599, M=511, K=4227, dv=8, dc=72, and CPM size=511) at the top of Fig. 3. The constructed QC-EG-LDPC code can be extended by decomposing [35] each eight-circulant permutation matrix into a 2×2 array of dual-circulant permutation matrices. The intermediate LDPC matrix (N=9198, M=1022, K=8178, dv=4, dc=36, and CPM size=511) is shown in the middle of Fig. 3.

Since the payload of the constructed code is still not large enough for 2KB user data encoding, we apply the array dispersion method in our previous works [29], [30] to further extend the code length twice larger. The resultant QC-EG-LDPC matrix, shown at the bottom of Fig. 3, consists of a 4×36 array of dual-circulant matrices with CPM size=511. Each row block comprises 18 dual circulant sub-matrices and 18 zero sub-matrices; each column block is composed of two dual-circulant sub-matrices and two zero sub-matrices. The extended code structure is arranged with the specific regular array pattern so that the extended QC-EG-LDPC could be viewed as the intertwining of two replicated intermediate LDPC matrices - one is dispersed in the first and fourth row blocks, and the other in the second and third row blocks.

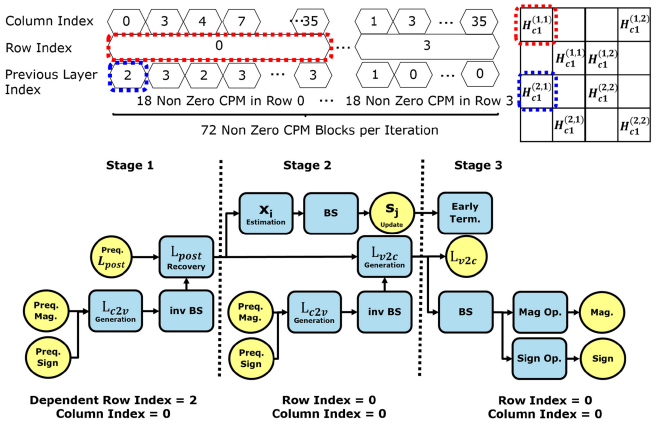


FIGURE 4. Block Parallel Layer Scheduling. The circle denotes storage element, and the square denotes combinational logic blocks.

In NAND flash applications, the payload is typically 2KB of user data with an additional 4 bytes for CRC (K=16384+32=16416). Since the null space of H gives a (18396, 16352) LDPC code, we need to purge the last 60 rows to construct the final (18396, 16416) QC-EG-LDPC code to satisfy our data size demand.

III. DECODING SCHEDULING AND ANALYSIS

The hardware architecture and decoder efficiency depends on the LDPC scheduling scheme. In this section, we introduce two well-known scheduling schemes and do the decoder evaluation with similar hardware complexity for the target QC-EG-LDPC code.

A. DECODING SCHEDULING SCHEME

The row-based layer scheduling scheme divides the LDPC matrix into four row layers. Each row layer's information exchange is referred to as the decoder's sub-task. For high code rate and long code length LDPC code, the block parallel layer decoder is an area-efficient design choice in row-based layer scheduling. The block parallel layer scheduling and the related block diagram are shown in Fig. 4, which illustrate an example of processing the sub-matrix with row index=0 and column index=0 in the red dash block.

In the QC-EG-LDPC code, the block-parallel decoder needs to access two blocks' prerequisite information (one from the non-zero CPM blocks in the previous dependent row and the other is the CPM block in the targeting row) to accomplish one block's information update for each cycle. Since only one block of information is exchanged, the block parallel layer decoder spends 18 cycles for each layer and 72 cycles for one iteration. Due to the data independence over the circulant blocks for each layer, the pipeline architecture becomes its advantage in achieving a high throughput design.

In the block parallel layer decoder, the first pipeline stage recovers the latest L_{post} from the previous dependent non-zero CPM block (the sub-matrix with previous layer index=2 and column index=0 in the blue dash block). The

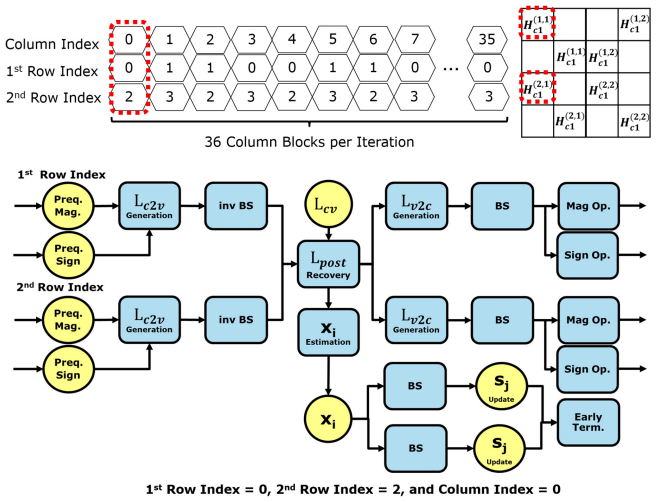


FIGURE 5. Shuffle Scheduling. The circle denotes storage element, and the square denotes combinational logic blocks.

block parallel decoder modifies the equation on line 10 in Algorithm 1 as

$$L_{post} = Preq.L_{post} + L_{c2v} \tag{1}$$

where the $Preq.L_{post}$ is redefined from L_{v2c} as the prerequisite information for L_{post}, and the L_{c2v} is generated from the prerequisite magnitude (*Preq. Mag.*) and sign (*Preq. Sign*) in the previous dependent layer. In the second stage, the decoder applies the same method to generate a block of L_{c2v} at layer index=0 and column index=0 in the red dash block. Then the previous stage's L_{post} is subtracted from the L_{c2v} to produce the latest L_{v2c} value, which corresponds to line 10 in Algorithm 1. The L_{post} also helps to estimate the code bits and update the intermediate syndrome values. In the last pipeline stage, the decoder consumes the latest L_{v2c} and exports the intermediate values (min1, min2, min1-index, and the global sign) to the temporary buffer. At the end of the row, these temporary values update the prerequisite magnitude and sign to finish the magnitude operation (Mag. Op.) and sign operation (Sign Op.) on lines 12-14 in Algorithm 1.

The column-based shuffle scheduling scheme divides the LDPC matrix into 36 column blocks. Each column block's information exchange is referred to as the decoder's subtask. The shuffle scheduling and the corresponding block diagram are shown in Fig. 5 as an example of processing the column block with column index=0.

Similar to the block-parallel layer decoder, the shuffle decoder also accesses two blocks' prerequisite information to generate the L_{c2v}. Since the shuffle scheduling updates two blocks of information at a time, the shuffle decoder spends one cycle for each column and 36 cycles for one iteration. In the variable node process, the decoder sums up all the connected L_{c2v} with L_{ch} to generate the latest L_{post}. Then, the L_{post} is used to estimate the code bit (x) and update the syndrome for an early termination check. Meanwhile, the latest L_{v2c} is immediately produced by subtracting the incident L_{c2v} from the L_{post} to accomplish line 10 in Algorithm 1. The resultant L_{v2c} is separated into the sign part and magnitude part to fulfill the corresponding operation on lines 12-14 in Algorithm 1. Unlike the layer decoder, the shuffle decoder immediately updates the prerequisite magnitudes and signs in the storage for the next column's decoding process.

TABLE 2. Decoder composition.

Type\Arch.		Shuffle Decoder	Layer Decoder
Non Comb. Logic	*Preq. Mag.	1983 × 18	1983 × 12
	Global Sign	1983	1983
	Syndrome	1983	1983
	Pipeline FF.	None	511 × (6 + 8)
	Temporary FF.	None	511 × 14
Reg. File Macro	V2C Sign	511 × 72 × 2	511 × 2 × 72
	Channel Value	18396 x 2	None
	Code word	18396	None
	Preq.L_{post}	None	18396 × 6
Comb. Logic	L_{c2v} Gen.	2 Blocks	2 Blocks
	L_{c2v}'s invBS	2 Blocks	2 Blocks
	L_{post} Recovery	1 Block	1 Block
	L_{v2c} Gen.	2 Blocks	1 Block
	L_{v2c}'s BS	2 Blocks	1 Block
	Syndrome's BS	2 Blocks	1 Block
	Mag. Op.	2 Blocks	1 Block
	Sign Op.	2 Blocks	1 Block

*Each check node's Preq. Mag. requires 18 bits (min1, min2, min1-index, and min2-index) for shuffler decoder and 12 bits (min1, min2, and min1-index) for layer decoder.

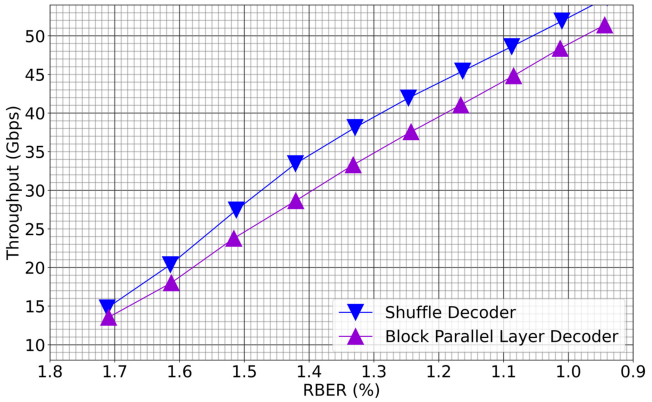
TABLE 3. Synthesis result of the TSMC 28nm.

	Shuffle Decoder	Layer Decoder
Clock Period (ns)	2.20 (454MHz)	1.20 (833MHz)
Macro (um^2)	362241.29	438739.35
Combinational (um^2)	523890.46	495167.09
Non Combinational (um^2)	118041.58	154142.99
Total Area (um^2)	1004173.33	1088049.43

B. QUANTITATIVE ANALYSIS OVER THE DECODER ARCHITECTURE

In addition to the scheduling scheme, the number of read levels in decoder inputs, size of the fixed-point message between nodes (L_{v2c} and L_{c2v}), and scaling factors (α) [36] also contribute to the decoding performance and hardware implementation cost. From a system point of view, since the multi-bit soft information would degrade the system throughput due to the NAND interface's limited bandwidth, our design adopts 2-bit soft values as decoder inputs. Based on the bit-true simulation model over the Bi-AWGN channel, our Monte-Carlo search suggests that $\alpha = 0.75$ and 4-bit message quantization in L_{v2c} and L_{c2v} provide the lowest marginal cost for decoding performances. Following the above configurations, we implement both architectures for quantitative analysis. Tables 2 and 3 list the decoder composition and the synthesis result in TSMC's 28nm HPM process.

Although the total area is close, the compositions of both decoder architectures are quite different. With respect to the non-combinational logic area, both architectures utilize an array of flip flops to store the prerequisite magnitude,



RBER (%)	1.70	1.61	1.51	1.42	1.33	1.24	1.16	1.08	1.01	0.94
Shuffle	13.18	18.17	24.49	29.85	34.03	37.43	40.55	43.38	46.33	49.05
Layer	11.61	15.44	20.37	24.54	28.52	32.19	35.17	38.41	41.44	44.04

FIGURE 6. Throughput Comparison @TSMC 28nm Synthesis and Average Iteration Counts over RBER Throughput = (Code Length × Clock Rate)/(Avg. Iter. × Cycle per Iteration) Cycle per iteration is 36 for shuffle decoder, and 72 for layer decoder. The average iteration counts are collected from the 10000 frames over the Bi-AWGN channel in the bit-true simulation.

global sign, and syndromes to avoid memory macros with extremely wide bit width and small word counts. Since the shuffle decoder requires an additional min2-index in the check node's magnitude operation (detailed in the next section), it consumes more area in prerequisite magnitude. However, the block parallel scheduling requires additional temporary buffers for intermediate values. Moreover, the pipeline registers in L_{post} and L_{v2c} contribute additional area overhead. As a result, the block parallel layer decoder requires more non-combinational logic than the shuffle decoder.

Due to the long code length property of the 2K LDPC code, the V2C signs are stored in register file macros in both architectures. The shuffle decoder requires the register file for channel value and estimated code word. The layer decoder stores all the $Preq.L_{post}$ values in the register file. The $Preq.L_{post}$'s storage capacity leads to larger block macros area than those in the shuffle decoder.

Regarding the combinational logics, both decoders access two blocks of prerequisite information to recover the L_{post} in variable node units, and the compositions for L_{c2v}-related logics are the same. Because the shuffle decoder simultaneously processes all the non-zero CPM blocks for each column, it requires twice computation units in L_{v2c} generation, barrel shifters, magnitude operation, and sign operation. As a result, the shuffle decoder consumes more area in combinational logics.

The column-base shuffle decoder runs at 454 MHz to complete one iteration of message exchanges for 36 cycles. The block parallel layer decoder can reach a clock rate of 833 MHz by pipeline architecture at the cost of 72 cycles for one iteration. The estimated throughput of both decoders in Fig. 6 shows that the shuffle decoder's small cycle count provides more competitiveness than the block-parallel layer decoder's high clock rate.

Based on the previous analysis of throughput and area, the column shuffle scheduling becomes the better choice for our 2K-QC-EG-LDPC code. However, the estimated throughput would be too optimistic in the lack of the wire load model in the 28nm synthesis library. In addition, the timing correlation issue between the synthesis and post-layout would be further enlarged in a high gate count design. As a result, the shuffle decoder requires more parallelism to support two ONFI 5.0 channels at RBER=1%.

IV. PROPOSED DECODER ARCHITECTURE

The overall architecture of our LDPC decoder design is shown in Fig.7, where the memories and logic blocks are colored for different types of configurations and usage. In the following subsections, we overview the decoder architecture and detail each submodule's design consideration.

A. THE ARRAY-DISPERSE BASED DUAL VNU CLUSTER ARCHITECTURE

The QC-EG-LDPC's array-dispersion pattern in Fig. 3 suggests that every two consecutive column blocks have independent dispersed CPM blocks in the matrix. This property not only allows the shuffle decoder to double the parallelism level but also simplifies the address control complexity without sacrificing the decoding performance. Therefore, we deploy dual variable unit (VNU) clusters in our architecture, where each cluster is in charge of 511 VNs' L_{post} recovery and L_{v2c} generation. The corresponding scheduling is shown in Fig. 8.

The storage for channel values and the estimated code word is also partitioned into two sets. Due to the dual-circulant structure in each CPM block, the proposed architecture requires eight barrel shifters and inverse barrel shifters for the variable node to check node (V2C) network and the check node to variable node (C2V) network. Each shifter is dynamically adjusted in each sub-iteration. Since all the check node units (CNU) should update the prerequisite information for L_{c2v} in each sub-iteration, we need to arrange 1983 CNUs in our decoder architecture. To use the inherent structure-property of QC-LDPC, we partition 1983 CNUs into four sets (Cluster A, B, C, and D) to fit into our shifting networks.

B. MEMORY & DATA PATH CONTROL

In the proposed architecture, each memory type is carefully-selected for the signal behavior. As mentioned in the previous section, the syndrome values, global signs, min1 and min2, and the corresponding indexes are all stored in the distributed registers (yellow block in Fig. 7). In the column shuffle decoding scheme, the channel values (CV) are pre-stored to compute the L_{post} for each column block. Therefore, we deploy single-port register files (red block in Fig. 7), which are only written in the first iteration. The register array is implemented as channel value LLR buffers to divide the timing path from the external circuit's delay. In order to accomplish the global sign update in (2), it requires both

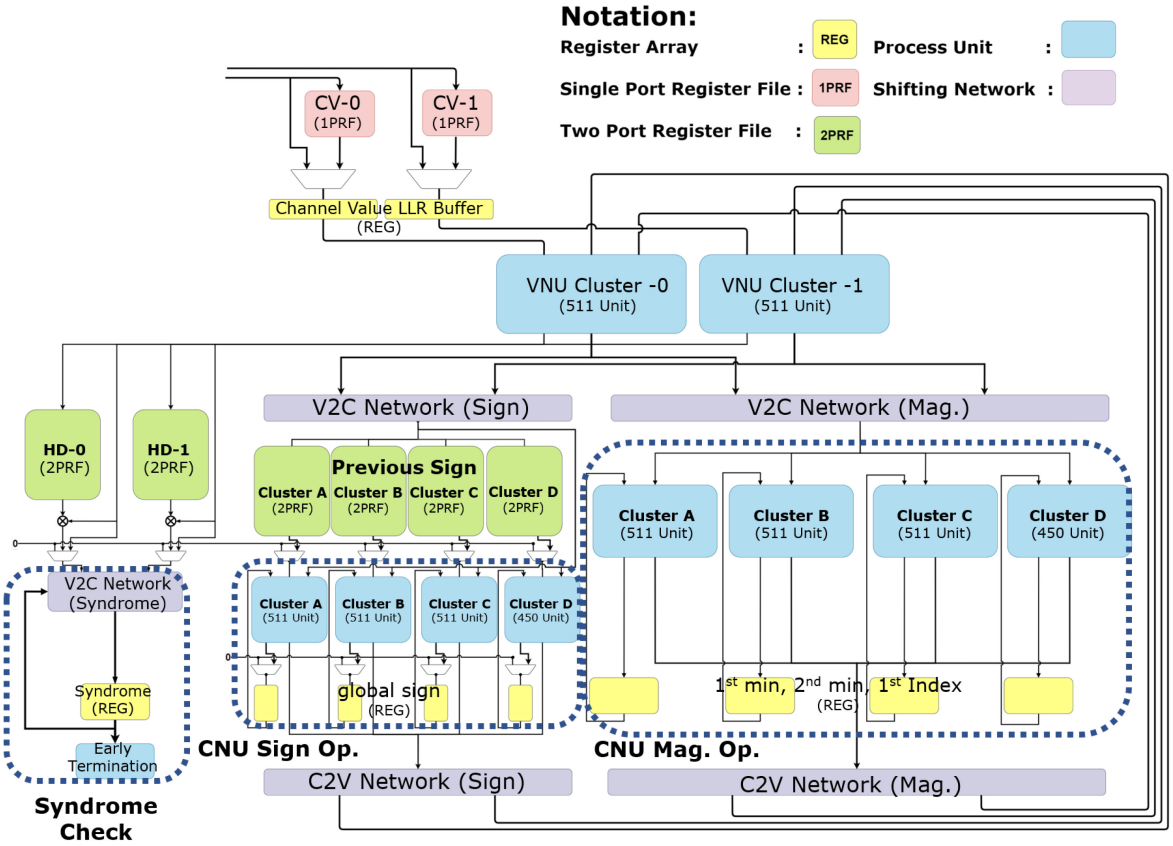


FIGURE 7. Hardware Architecture.

the previous iteration's V2C sign and the latest V2C sign on the data path.

$$\text{global sign}^{t}(j) = \text{global sign}^{t-1}(j) \oplus sign\left(L_{v2c}^{t-1}(i,j)\right) \oplus sign\left(L_{v2c}^{t}(i,j)\right) \quad (2)$$

Since the latest V2C sign must be written to the memory for the next iteration, the decoder requires two-port register files (green block in Fig. 7) to support reading and writing simultaneously. Similarly, the syndrome update process in (3) also requires two-port register files to read the previous codeword and write the current codeword in one cycle.

$$s^{t}(j) = s^{t-1}(j) \oplus x^{t-1}(i) \oplus x^{t}(i) \quad (3)$$

Fig. 9 shows the state transition and memory control over the proposed decoder. At the initial state (INIT_DEC), both sets of channel values go through the VNU cluster and V2C shifter and overwrite the V2C sign and syndrome with the early termination disable. After all the channel value has been loaded into the decoder, the decoder transits to a normal decoding state (DEC), and the syndrome check is activated for early termination. In the meantime, both the V2C sign and codeword memory simultaneously read and write to update global signs and syndromes, respectively. If the early termination signal is triggered, the decoder will transfer the codeword read control from decoding state control logic to the output state control logic. Because the codeword two-port

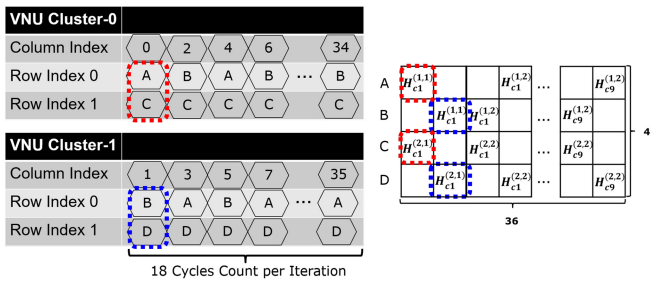


FIGURE 8. Dual VNU Scheduling.

register files' read and write ports are currently controlled by the two distinct control logics, the decoder could simultaneously load the next frame's channel value and output the decoded codeword to maximize the throughput.

C. SHIFTING NETWORK & SYNDROME CHECK

Due to the target QC-LDPC code's structure, the V2C and C2V shifting network are implemented using a cluster of multiplexers. The proposed decoder architecture consumes eight sets of independent multiplexer-based shifting networks, which dynamically route the L_{v2c} and L_{c2v} in each sub-iteration. According to (3), the code words in each column block must be aligned to the corresponding rows. As a result, the syndrome check circuit consists of a 1-bit

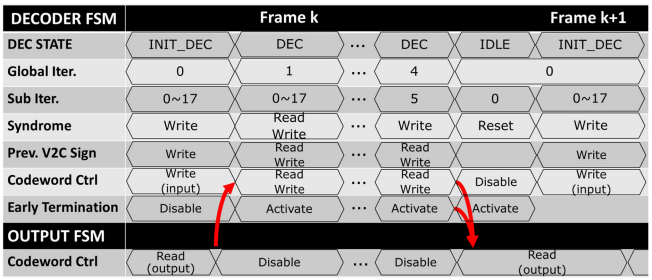


FIGURE 9. Data Flow Control.

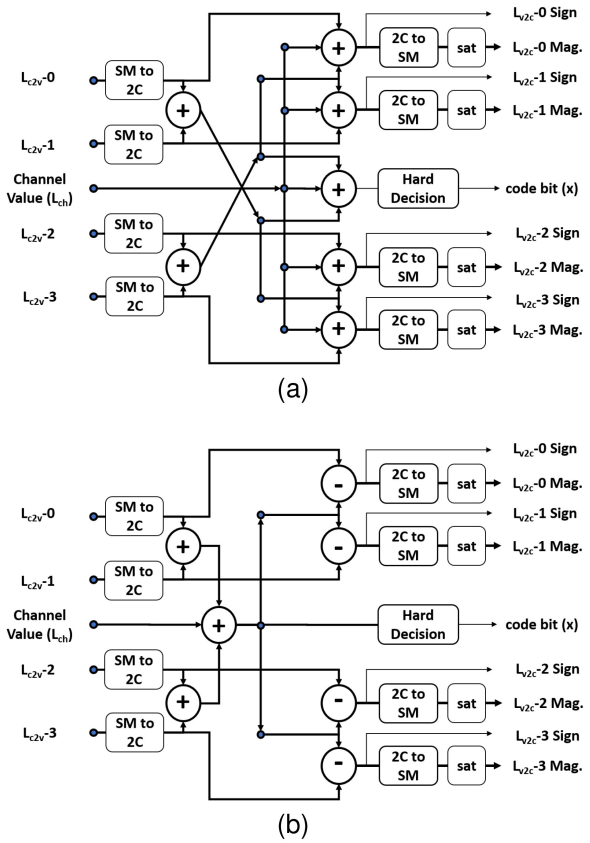


FIGURE 10. VNU Circuit Architecture: (a) Two-Stage High Speed Structure (b) Three-Stage Routing-Efficient Structure SM: sign-magnitude form, 2C: two's complement form.

V2C shifting network, two-port codeword register files, and syndrome registers, as shown in Fig. 7.

D. VARIABLE NODE UNIT

Fig. 10 presents different VNU circuit structures for L_{post} and L_{v2c} computation. The first stage is the sign-magnitude to two's complement converter, which combines sign and magnitude values to fulfill line 21 in Algorithm 1. The last stage is the two's complement to the sign-magnitude converter to separate L_{v2c}. The saturation circuit is deployed to keep the magnitude of LLR within 3-bit width. From the perspective of place and route, the routing-efficient structure [37] contributes less wire signal detour inside the VNU. However, all the internal connections are packed as a unit, and all VNUs are widely distributed over the design. Regardless

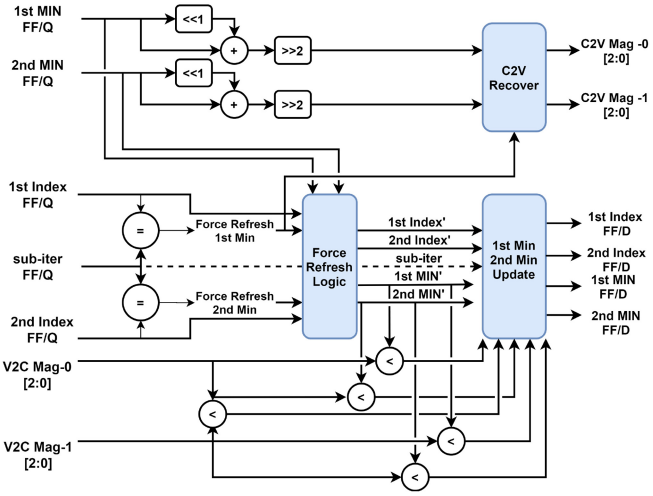


FIGURE 11. Magnitude Operation (Mag. Op.) Unit.

of the selected structure, the L_{v2c} and L_{c2v} message's bit-length and the number of connected check nodes are the two dominant factors in routing congestion. As a result, we adopt a two-stage high-speed structure [38] as our VNU for a high-throughput design.

E. CHECK NODE UNIT'S MAGNITUDE OPERATION

Fig. 11 shows the magnitude operation unit in the check node process. In the column shuffling scheduling scheme, the min1-and-min2 update mechanism differs from the conventional software simulation. Given that the decoder is currently at the i-th iteration and the k-th sub-iteration, the min1 and min2 of the L_{v2c} message are the temporary results from the k-th sub-iteration in the previous iteration to the current (k-1)-th sub-iteration. Suppose the current k-th sub-iteration V2C message is larger than the temporary min1 and min2, which also happens to be the same value in the previous k-th sub-iteration. In that case, the min1, min2, and min indices would not be changed, leading to performance degradation. Hence, we adopt the force-refresh logic to discard the corresponding min1 or min2, followed by the selection logic for the latest prerequisite magnitude update. In the case that the min1 is forced to refresh, the engagement of the min2 and min2 index could alleviate the performance loss.

Moreover, the upper part of the circuit shows the L_{c2v} generation logic, where the scaling factor (α) is implemented with the compact shift and add unit. All the L_{c2v} generation logic's outputs depend on the signals from the storage, such as min1, min2, min1 index, min2 index, and sub-iteration count. This configuration cleverly separates the data path into two independent cycles and avoids the combinational loop.

V. CONGESTION-AWARE PHYSICAL IMPLEMENTATION

In this section, an analytical method over the synthesis netlist is applied, and the observation helps us regulate a routing-oriented floor plan to achieve the congestion-free LDPC decoder design implemented in the TSMC-28nm technology process.

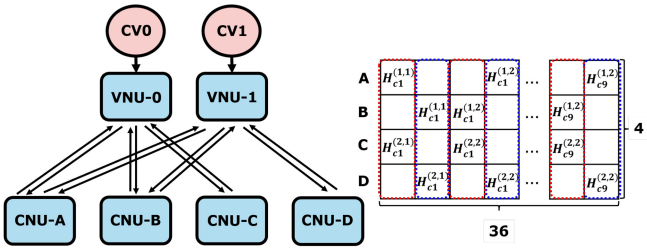


FIGURE 12. Bipartite Graph of Dual VNU Shuffle Scheduling.

TABLE 4. Pin density & congestion metrics.

	Area	# Pins	# Ports	PD	CM
VNU 0/1	206160.71	63606	17885	0.3085	9.847
***CNU Mag Op**	75173.32	28108	6132	0.3739	5.591
***CNU Sign Op**	6715.57	4088	2044	0.6088	6.235
Syndrome Check	60456.61	28992	4993	0.4795	5.076
V2C Network A/B	74730.42	24419	16352	0.3227	14.954
V2C Network C/D	63010.70	22680	8176	0.3599	8.142
C2V Network A/B	72000.99	23947	16352	0.3326	15.235
C2V Network C/D	60709.32	22518	8176	0.3709	8.295

The above information is collected from the Design Compiler's synthesis result.
*CNU Mag OP and CNU Sign Op is partitioned into 4 Clusters.
CM is denoted as Congestion Metric : (Num. of Ports)/$(4 \times \sqrt{\text{Area}})$
PD is denoted as Pin Density: (Num. of Pins)/(Area)

A. CONGESTION ANALYSIS ON TOP LEVEL DESIGN

The bipartite graph of the adopted VNU scheduling in Fig. 12 and the congestion-oriented analysis in Table 4 provide a clear view of the top-level design. The corresponding observations are listed as follows:

- The VNU cluster is the largest connected top-level module. The location of the two VNU clusters would significantly affect the neighboring instances' routing resources.
- Due to the array-dispersion pattern scheduling, the connections between VNU and CNU cluster A/B are more complicated than the other CNU clusters. As a result, the congestion metrics of the corresponding V2C network and C2V network are significantly higher than the others.
- In the merit of the decoder architecture in Fig. 7, the sign and magnitude operations are purposely divided into two separate blocks. This configuration allows sign operation units distributed close to sign register files, with no bound to the magnitude operation logic.
- The pin-density of the sign operation modules and syndrome check module is relatively higher than others, which would cause the detour for the fly-over nets.

Based on the above information, a well-considered floor planning of hard macros and top-level instances is applied, and the placement result is shown in Fig. 13(a).

B. CONGESTION-FREE FLOOR PLANNING

Considering that the VNU acquires the largest area and wire connection, we preserve most of the central region for VNU clusters. Because there is no direct connection between the VNUs, the VNU clusters are arranged in a diagonal format so that the limited horizontal and vertical routing wires are evenly utilized. Since both VNUs share the CNU operations of clusters A and B, we intentionally place the memory macros in the middle of the upper and right edges and leave the upper right corner as the reserved regions for the interleaving barrel shifters and the magnitude operation units. On the other hand, the last two row blocks (clusters C and D) are connected with the fixed VNU cluster, so the related shifters and storage units are deployed in the upper left and lower right corners. As shown in Fig. 7, the syndrome registers split the delay path between hard-decision logic and early termination. Since it provides more timing budgets for syndrome check operation, we can place the early termination logic and codeword register file in the lower-left corner of the floorplan to avoid contending routing resources. As for CNU magnitude operation, the low congestion metric (CM) in Table 4 suggests that these logic cells can be scattered all around the floorplan (green) and significantly diminishes the risk of congestion.

In addition to the guidance applied over the initial placement, we apply the following methods to alleviate the routing congestion further.

1) back-to-back macro placement: Due to the bandwidth of the register files being limited to 144 bits, the design consists of 52 memory macros over the physical layout. To save more area near the memory macros, we adopt the back-to-back placement of the V2C sign macros and leave enough space for sign operation units.
2) partial placement blockage under constraint: The dedicated partial placement blockage, which limits the utilization rate of logic inside the region, is set over the memory spacing to avoid the channel congestion issues between the macros.
3) power ground network alignment: Since the LDPC decoder is a power-consuming engine, the well-designed power ground network could also relieve the routing resources among layers of metals. This work aligns the power stripes in all layers to ensure all power vias in the upper layers can punch through the bottom layers without detouring. Because the number of power vias is reduced, that would leave more space for the signal routing and clock tree synthesis.

With previous approaches, the congestion map suggests no congestion issue under a 70% cell utilization ratio. Fig. 13(b) shows the layout view of the decoder with a 2.972 mm^2 core area.

VI. EXPERIMENT RESULTS

Our bit-true model's ECC performance over the AWGN channel in Fig. 14 shows that the proposed decoder outperforms the previous works and provides an ECC performance at UBER=10^{-5} under RBER=1.456% channel condition. In addition, the proposed decoder's UBER can reach nearly 1.178×10^{-10} without an error floor at RBER=1.245%. According to [6] and [9], the storage system based on the latest 3D TLC and QLC generally views RBER= 1% as their end of life, and the practical LDPC code's UBER is down

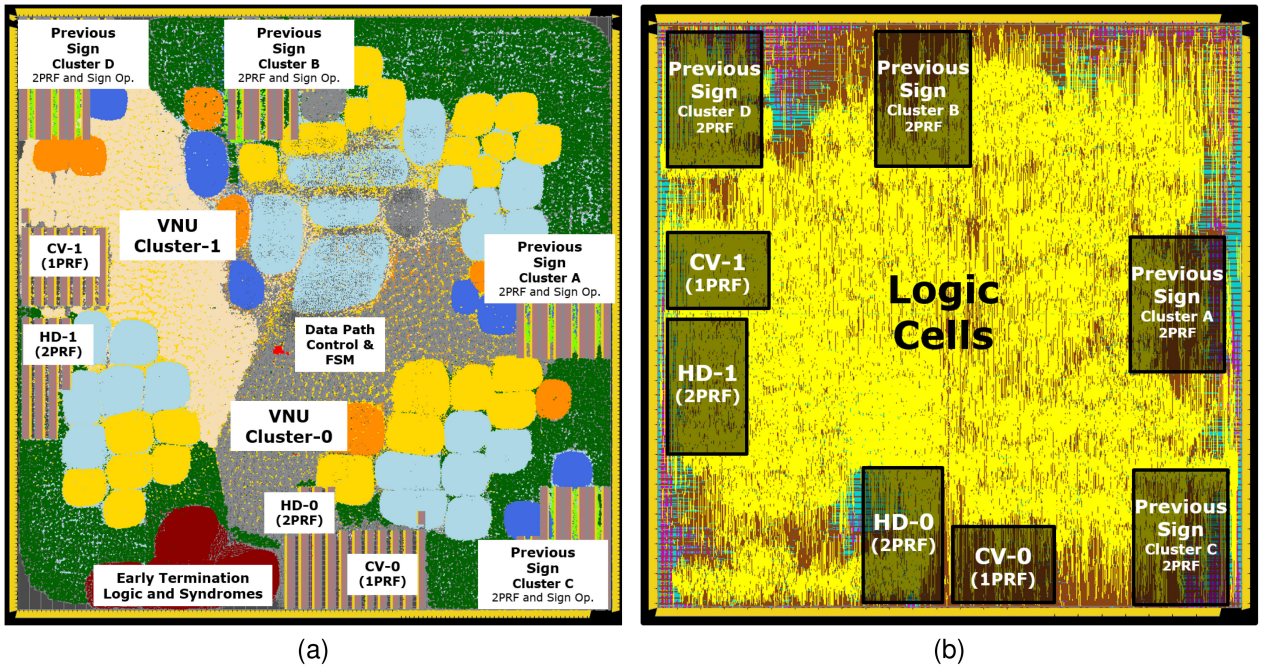


FIGURE 13. Physical Implementation: (a) Placement Result (b) Final Layout.

TABLE 5. Comparison table.

	Proposed	ISCAS'21 [28]	TCAS-II'18 [27]	TCAS-I'17 [26]	TVSLI'16 [25]
State	Post-Layout	Synthesis	Post-Layout	Post-Layout	Post-Layout
Scheduling	Column Shuffle	Row Layer	Column Shuffle	Row Layer	Column Shuffle
Codes(N,K)	(18396, 16416)	(9216, 8192)	†(18300, 16416)	(18396, 16644)	(18900, 17010)
Code Rate	0.892	0.888	0.897	0.904	0.9
CPM Size	511	128	61	292	63
Utilization Ratio	0.70	N.A	0.80	0.62	0.57
Process (nm)	28	28	90	90	90
Area (mm^2)	2.972	0.26	1.44	4.19	2.56
Input Quant.	2	N.A.	2	5	2
Internal Quant.	4	N.A.	4	5	4
Clock Rate (MHz)	277.77	272	200	200	166
Support Early Termination	Yes	No	N.A.	No	Yes
Channel Condition (RBER%)	1.456	1.00	N.A.	1.28	0.8
Iteration	Avg. 7.34	Max 4	6	Max 8	Avg. 6
* Data Rate (Gbps)	34.504	1.553	1.83	4.246	1.58
Average Power (mW)	1115.6	50.5	N.A.	299	294

† According to [27], the (N,K) is modified from (18300,16470) to (18300,16416) due to non-full rank property.
* Data Rate is defined as $\frac{\text{Payload (K)} \times \text{Clock Rate}}{\text{Iteration} \times \text{Cycles per Iteration}}$, where the cycle per iteration is 18 in our case.

to 10^{-10} without an error floor. As a result, the proposed ECC performance is qualified for the reliability criterion in NAND flash applications.

Table 5 presents the implementation results of the proposed (18396,16416) QC-EG-LDPC decoder and the state-of-the-art works aiming at NAND flash applications. Under the ss corner ($125°C$, 0.81V), the proposed decoder can operate at 277.7 MHz. The critical path starts with min1's Flip-Flop's data output. After passing through CNU's L_{c2v} generation logic, C2V routing networks, VNU, V2C routing networks, and the CNU's magnitude operation unit, the critical path finally ends with the min1 index Flip-Flop's data input. At RBER $= 1.456\%$, the proposed decoder takes 7.342 iterations on average to complete the decoding process and achieves a data rate of 34.5 Gbps (equivalent to 38.64 Gbps in throughput). The decoder's 1115.6 mW power is measured by the Synopsys Prime Power with testing vectors under the same channel condition in the ff corner ($-40°C$, 0.99V). The proposed utilization ratio is 0.7, which is higher than most candidates with a small CPM size except [27], featuring barrel shifter free based on array disperse code construction method.

To select suitable ECC engines for a storage system, a system architect would compare all the decoder's data rates in the same channel conditions (e.g., RBER). After re-implementing our design in the 90nm process, we recompute our design's data rates at the aligned channel conditions. The area efficiency comparison in Fig. 15 suggests that our decoder provides higher area efficiency than the others.

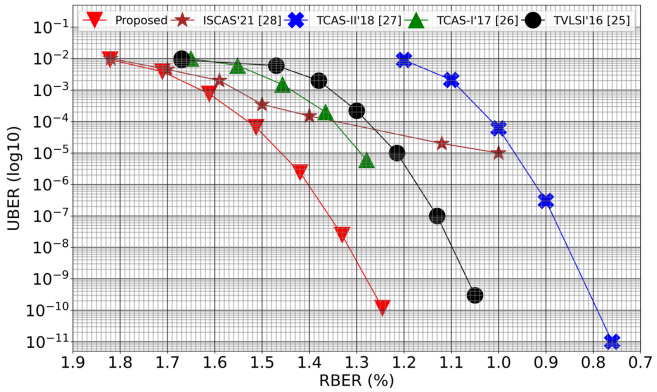
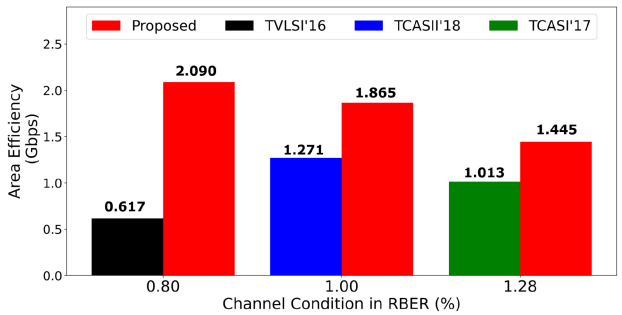


FIGURE 14. ECC performance plot (UBER vs RBER). 1) The waterfall of TCAS I'17 is transformed from SNR to RBER. 2) The code rate effect is compensated in the AWGN channel of our bit true simulation with variance = $(2R\frac{E_b}{N_0})^{-1}$.



Proposed Work @T90 Post Layout			
Channel Condition RBER (%)	0.80	1.00	1.28
Data Rate (Gbps)	25.64	22.88	17.72
Area (mm^2)	12.2675		
Clock Rate (MHz)	111.1		

FIGURE 15. Area Efficiency Comparison Area Efficiency = Data Rate (Gbps)/Area (mm^2) *Since there is no information to derive the channel condition in TCAS-II'18, we assume it operates at RBER=1.0% from Fig. 14.

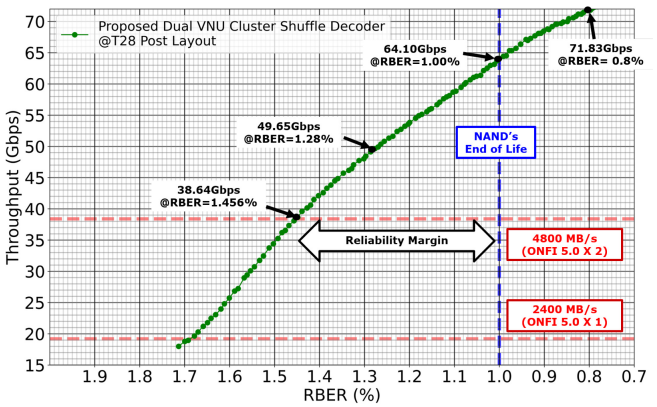


FIGURE 16. Throughput over RBER. Throughput = (Code Length=18396 × Clock Rate=277.77) / (Average Iteration × Cycle Count=18). The average iteration counts are collected from the 10000 frames over the Bi-AWGN channel in the bit-true simulation in each channel conditions.

Since the NAND Flash interface's speed contains the entire codeword inbound to the decoder, we provide the throughput-to-RBER chart over the Bi-AWGN channels in Fig. 16 to verify our implementation. The proposed decoder could reach 38.64 Gbps at RBER=1.456%, which implies that the decoder not only supports at least two ONFI 5.0 physical channels but also provides an additional 83 error bit capability (0.456% × 18396) from NAND's end of life. This additional error capability can act as the reliability margin to compensate for the discrepancy between AWGN and the real NAND channels.

VII. CONCLUSION

This paper presents the feasibility of the large CPM QC-EG-LDPC code for the severe channel condition and the demanding high throughput requirement. After considering decoding performance, scheduling schemes, hardware complexity, and target throughput, we adopt the Array-Disperse Based Dual VNU Cluster architecture to achieve a high throughput design with the column-shuffle scheduling scheme. Furthermore, in collaboration with the congestion analysis on top-level design and the congestion-free floor planning, the proposed decoder's physical implementation could reach a 0.7 utilization ratio. Implemented in TSMC 28nm process, the proposed decoder occupies a 2.97 mm^2 core area with 38.64 Gbps throughput under RBER=1.456% channel condition. It supports at least two physical channels of ONFI 5.0 with an additional 82-bit reliability margin.

REFERENCES

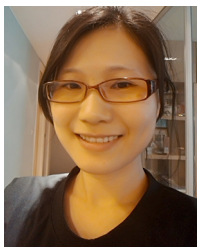
[1] R. Micheloni, *3D Flash Memories*. Dordrecht, The Netherlands: Springer, 2016.
[2] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Threshold voltage distribution in MLC NAND flash memory: Characterization, analysis, and modeling," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2013, pp. 1285–1290.
[3] W. Liu *et al.*, "Characterizing the reliability and threshold voltage shifting of 3D charge trap NAND flash," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2019, pp. 312–315.
[4] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch, and O. Mutlu, "Improving 3D NAND flash memory lifetime by tolerating early retention loss and process variation," in *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 2, 2018, pp. 1–48.
[5] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.
[6] Q. Li *et al.*, "Shaving retries with sentinels for fast read over high-density 3D flash," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchitect. (MICRO)*, 2020, pp. 483–495.
[7] "3D NAND flash memory flyer," Micron. 2016. [Online]. Available: https://www.micron.com/-/media/client/global/documents/products/product-flyer/3dnandflyer.pdf
[8] T. J. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annu. Allerton Conf. Commun. Control Comput.*, vol. 7, 2003, pp. 1426–1435.
[9] S.-H. Kuo, Z.-U. Liu, and J. Yang, "On practical LDPC code construction for NAND flash applications," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2017, pp. 191–195.
[10] X. Xiao, B. Vasić, S. Lin, K. Abdel-Ghaffar, and W. E. Ryan, "Reed-solomon based quasi-cyclic LDPC codes: Designs, girth, cycle structure, and reduction of short cycles," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5275–5286, Aug. 2019.
[11] "JESD218B," JEDEC, Arlington, VA, USA, 2016.
[12] Q. Huang, J. Kang, L. Zhang, S. Lin, and K. Abdel-Ghaffar, "Two reliability-based iterative majority-logic decoding algorithms for LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 12, pp. 3597–3606, Dec. 2009.
[13] O. A. Rasheed, P. Ivaniš, and B. Vasić, "Fault-tolerant probabilistic gradient-descent bit flipping decoder," *IEEE Commun. Lett.*, vol. 18, no. 9, pp. 1487–1490, Sep. 2014.
[14] K. Le, D. Declercq, F. Ghaffari, L. Kessal, O. Boncalo, and V. Savin, "Variable-node-shift based architecture for probabilistic gradient descent bit flipping on QC-LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 7, pp. 2183–2195, Jul. 2018.

[15] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch, and O. Mutlu, "HeatWatch: Improving 3D NAND flash memory device reliability by exploiting self-recovery and temperature awareness," in *Proc. IEEE Int. Symp. High Perform. Comput. Architect. (HPCA)*, 2018, pp. 504–517.
[16] Z. Fan, G. Cai, G. Han, W. Liu, and Y. Fang, "Cell-state-distribution-assisted threshold voltage detector for NAND flash memory," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 576–579, Apr. 2019.
[17] N. Papandreou, R. Agarwal, J. M. Cioffi, M. Qin, and P. H. Siegel, "Using adaptive read voltage thresholds to enhance the reliability of MLC NAND flash memory systems," in *Proc. GLSVLSI*, New York, NY, USA, 2014, pp. 151–156.
[18] B. Peleato, R. Agarwal, J. Cioffi, M. Qin, and P. H. Siegel, "Towards minimizing read time for NAND flash," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2012, pp. 3219–3224.
[19] "Open NAND Flash Interface Specification Revision 5.0." ONFI. 2021. [Online]. Available: http://www.onfi.org/specifications
[20] P. Schläfer, N. Wehn, M. Alles, and T. Lehnigk-Emden, "A new dimension of parallelism in ultra high throughput LDPC decoding," in *Proc. SiPS*, 2013, pp. 153–158.
[21] R. Ghanaatian *et al.*, "A 588-gb/s LDPC decoder based on finite-alphabet message passing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 2, pp. 329–340, Feb. 2018.
[22] M. Milicevic and P. G. Gulak, "A multi-gb/s frame-interleaved LDPC decoder with path-unrolled message passing in 28-nm CMOS," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 10, pp. 1908–1921, Oct. 2018.
[23] M. Li, V. Derudder, K. Bertrand, C. Desset, and A. Bourdoux, "High-speed LDPC decoders towards 1 tb/s," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 5, pp. 2224–2233, May 2021.
[24] M.-R. Li, H.-C. Chou, Y.-L. Ueng, and Y. Chen, "A low-complexity LDPC decoder for NAND flash applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2014, pp. 213–216.
[25] K. Ho, C.-L. Chen, and H. Chang, "A 520k (18900, 17010) array dispersion LDPC decoder architectures for NAND flash memory," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 4, pp. 1293–1304, Apr. 2016.
[26] H.-C. Lee, M. Li, J.-K. Hu, P.-C. Chou, and Y.-L. Ueng, "Optimization techniques for the efficient implementation of high-rate layered QC-LDPC decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 2, pp. 457–470, Feb. 2017.
[27] W. Shao, J. Sha, and C. Zhang, "Dispersed array LDPC codes and decoder architecture for NAND flash memory," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 8, pp. 1014–1018, Aug. 2018.
[28] C. Zhang, J. Mo, Z. Lian, and W. He, "High energy-efficient LDPC decoder with AVFS system for NAND flash memory," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2021, pp. 1–4.
[29] Y.-C. Liao, H.-C. Chang, and S. Lin, "Scalable globally-coupled low-density parity check codes," in *Proc. IEEE 10th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, 2018, pp. 1–5.
[30] H.-C. Chang, Y.-C. Liao, and S. Lin, "Low-density parity-check code scaling method," U.S. Patent 10 886 944, Jan.–May 2021.
[31] B. Vasic and O. Milenkovic, "Combinatorial constructions of low-density parity-check codes for iterative decoding," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1156–1176, Jun. 2004.
[32] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth tanner graphs," in *Proc. GLOBECOM*, vol. 2, 2001, pp. 995–1001.
[33] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, "A class of low-density parity-check codes constructed based on reed-solomon codes with two information symbols," *IEEE Commun. Lett.*, vol. 7, no. 7, pp. 317–319, Jul. 2003.
[34] Q. Huang, Q. Diao, S. Lin, and K. Abdel-Ghaffar, "Trapping sets of structured LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2011, pp. 1086–1090.
[35] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.
[36] J. Chen and M. P. C. Fossorier, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Commun. Lett.*, vol. 6, no. 5, pp. 208–210, May 2002.
[37] Y.-L. Ueng, C.-J. Yang, K.-C. Wang, and C.-J. Chen, "A Multimode shuffled iterative decoder architecture for high-rate RS-LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 10, pp. 2790–2803, Oct. 2010.
[38] Z. Cui, Z. Wang, and X. Zhang, "Reduced-complexity column-layered decoding and implementation for LDPC codes," *IET Commun.*, vol. 5, no. 15, pp. 2177–2186, 2011.

LI-WEI LIU (Member, IEEE) received the B.S. degree in electronics engineering from National Chiao Tung University (currently, National Yang Ming Chiao Tung University), Hsinchu, Taiwan, in 2018, where he is currently pursuing the Ph.D. degree in electronics engineering.

His current research interests include error correction coding, machine learning, and VLSI architecture for storage and communication system.

MU-HUA YUAN received the B.S. degree in engineering science from National Cheng-Kung University, Tanian, Taiwan, in 2017, and the M.S. degree in electronic engineering from National Chiao Tung University (currently, National Yang Ming Chiao Tung University), Hsinchu, Taiwan, in 2019. He is currently with the Chip Integration Team, Mediatek.



YEN-CHIN LIAO (Member, IEEE) received the B.S. and M.S. degrees in communication engineering and the Ph.D. degree in electronics engineering from National Chiao Tung University (currently, National Yang Ming Chiao Tung University), Hsinchu, Taiwan, in 2000, 2002, and 2008, respectively.

She was with ISSC Technology from 2002 to 2003, where she was involved in developing the IEEE802.11a/g products. From 2008 to 2010, she was with Ralink Technology, where she was working on the IEEE802.11n enhancement projects and submitting contributions to the IEEE802.11ac standard. From 2010 to 2012, she was recruited by MDV Technology, where she participated in the IEEE802.11d/e product development. She joined the Department of Electronics Engineering, National Chiao Tung University (currently, National Yang Ming Chiao Tung University), as a Postdoctoral Researcher in 2012. Her research interests include error correction coding, machine learning, signal processing, and communication systems.

HSIE-CHIA CHANG (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the Electronics Engineering Department, National Chiao Tung University (currently, National Yang Ming Chiao Tung University), Hsinchu, Taiwan, in 1995, 1997, and 2002, respectively.

He was with OSP/DE1, MediaTek Corporation from 2002 to 2003, where he was involved in decoding architectures for combo single chip. In 2003, he joined the Electronics Engineering Department, National Chiao Tung University (currently, National Yang Ming Chiao Tung University), as a Faculty Member, where he has been a Professor since 2010. He has served as the Deputy Director General with Chip Implementation Center, Taiwan, from 2014 to 2017. He has authored a book, over 120 IEEE journal/conference papers, and over 70 U.S./China/Taiwan patents. His research interests include algorithms and VLSI architectures in signal processing, in particular, error control codes and cryptosystems.

Dr. Chang was a recipient of the Outstanding Youth Electrical Engineer Award from the Chinese Institute of Electrical Engineering in 2010, and the Outstanding Youth Researcher Award from the Taiwan IC Design Society in 2011. He has also served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: REGULAR PAPERS since 2012, as well as a Technical Program Committee Member of the IEEE Asian Solid-State Circuits Conference from 2011 to 2013, and the International Solid-State Circuits Conference from 2018 to 2021.