Adaptive 2D Scheduling based Nonbinary Majority-logic Decoding for NAND Flash Memory

Cheng Wang, Jun Li, *Senior Member*, *IEEE*, Lingjun Kong, *Member*, *IEEE*, Feng Shu, *Member*, *IEEE*, and Francis C.M. LAU, *Senior Member*, *IEEE*.

***Abstract*—This paper presents a nonbinary adaptive 2D scheduling-based majority logic decoding (NB-ATS-MLGD) algorithm for NAND flash memory. The proposed algorithms provide considerable tradeoff between error-correcting capability and decoding complexity, and make the NB-MLGD decoding more attractive for practical purposes in the multi-level cells (MLC) NAND flash memory. The most significant feature of the proposed NB-ATS-MLGD algorithm is the 2D layered scheduling strategy, where the decoding orders of check nodes (CNs) and variable nodes (VNs) are both adaptive. By leveraging on the MLC flash memory bit error patterns, an early-correcting (EC) criterion is incorporated into the NB-ATS-MLGD algorithm. Furthermore, the simplification and parallelization of proposed algorithms make them more practical in NAND flash memory. Simulation results show that the proposed algorithms increase the lifetime of MLC flash memory up to 3000 program-and-erase (PE) cycles and have desirable convergence speed compared with the conventional non-binary MLGD algorithm. When at low PE cycles, the NB-EC-ATS-MLGD algorithm improves frame error rate (FER) performance by more than 3 order of magnitudes compared with the conventional non-binary MLGD algorithm.**

***Index Terms*—Multi-level cell (MLC), NAND flash memory, non-binary low-density parity-check (NB-LDPC) codes, 2D scheduling, majority-logic decoding (MLGD).**

I. Introduction

RECENTLY, NAND flash memory has been widely used in solid state drives (SSDs), smart phones and other electronic devices due to the high access speed and low power consumption [1]. However, the performance of NAND flash memory is effected by the various electrical noises and interference components. To address this issue, error-correction codes (ECCs) such as low-density parity-check (LDPC) codes have been applied to storage systems, which significantly improve the reliability of NAND flash memory [2].

As non-binary (NB)-LDPC codes can achieve a superior error correcting capability compared with binary LDPC codes, many NB algorithms have been proposed, such as q-ary sum-product algorithm (QSPA) and extended min-sum (EMS) algorithm [3], [4]. To make a tradeoff between performance and complexity, iterative soft reliability-based majority-logic decoding (ISRB-MLGD) algorithm and other improved algorithm (IISRB-MLGD) have been proposed [5]–[7]. These NB-MLGD algorithms involve only finite field additions and multiplications as well as integer operations, which allow NB-LDPC codes to become attractive ECCs using in high-density flash memory.

For the past few years, the message-passing scheduling algorithms have attracted a lot of attention [8]–[10]. The flooding scheduling is the conventional scheduling scheme where all variable-to-check and check-to-variable messages are exchanged simultaneously. Compared with the flooding scheduling, informed dynamic scheduling (IDS), which uses the residual of message to get the latest available information, significantly improves the convergence speed [9].

To improve the error performance and accelerate the convergence of NB-MLGD algorithm, we first propose a non-binary adaptive 2D scheduling based majority-logic decoding (NB-ATS-MLGD) algorithm. Then, to pursue a superior error-correcting capability, an early-correcting (EC) criterion is incorporated into the NB-ATS-MLGD algorithm to form the NB-EC-ATS-MLGD. After that, a simplified version of NB-ATS-MLGD algorithm has been proposed by simplifying the message passing process of ATS, which dramatically reduces the complexity of NB-ATS-MLGD algorithm. Furthermore, in order to increase the practicality of the proposed algorithms in NAND flash memory, a parallel scheme has been added in by separating the check nodes (CNs) into several groups and then executing ATS, which provides considerable balance between the performance and hardware consumption.

II. Background

A. MLC NAND Flash Memory Channel Model

According to [11], there are many noises in flash memory, such as programming noise (PN), cell-to-cell interference (CCI), random telegraph noise (RTN) and data retention noise (DRN), which are all additive noises. So, we can build NAND flash memory channel model as Fig. 1. The detailed parameters of flash memory channel model are the same as the model given in [11]. Based on the MLC NAND flash memory model, we can get the soft threshold voltage by maximizing the mutual information (MMI) of flash channel and then obtain the log-likelihood ratio (LLR) information [11], [12]. Fig. 1

This work is supported in part by Key Program of NSFC (Grant No. U1501253) and NSFC (Grant Nos. 61727802, 61872184, and 61501250). Corresponding author: Lingjun Kong and Jun Li (email: ljkong@njupt.edu.cn; jun.li@njust.edu.cn).

Cheng Wang, Jun Li and Feng Shu are with School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, P. R. CHINA (e-mail: cheng.wang@njust.edu.cn; jun.li@njust.edu.cn; shufeng@njust.edu.cn). Jun Li is also with the School of Computer Science and Robotics, National Research Tomsk Polytechnic University, Tomsk, 634050, RUSSIA.

Lingjun Kong is with College of Telecommunication and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing, P. R. CHINA (e-mail: ljkong@njupt.edu.cn).

Francis C. M. Lau is with Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Hong Kong (e-mail: encm-lau@polyu.edu.hk).

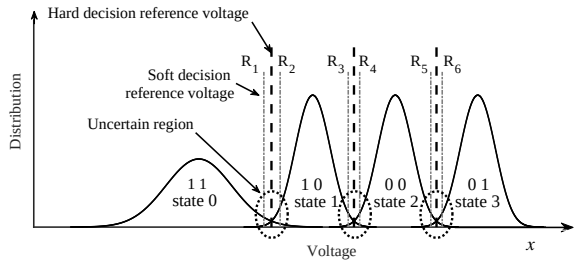

Fig. 1. Voltage distribution of an MLC NAND flash memory.

shows the 6-level MMI quantization schemes and the voltage distribution of an MLC NAND flash memory cell where PN, CCI, RTN and DRN are included.

B. Conventional ISRB-MLGD Algorithm

Assume $\mathcal{C}$ is a NB-LDPC code defined by the null space of an $m \times n$ sparse parity-check matrix $\mathbf{H}$ over finite field GF(q), where $q = 2^r$. Let $\mathbf{h}_i = (h_{i,1}, h_{i,2}, \ldots, h_{i,n})$ be the i-th row of $\mathbf{H}$ ($i = 1, 2, \ldots, m$). Denote the i-th CN ($i = 1, 2, \ldots, m$) and the j-th variable node (VN) ($j = 1, 2, \ldots, n$) as c_i and v_j, respectively. Define $\mathcal{N}_{c_i} = \{j : 1 \le j \le n, h_{i,j} \neq 0\}$ for $1 \le i \le m$ and $\mathcal{M}_{v_j} = \{i : 1 \le i \le m, h_{i,j} \neq 0\}$ for $1 \le j \le n$. For an information codeword $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{C}$, where $x_j = (x_{j,1}, x_{j,2}, \ldots, x_{j,r})$ for $1 \le j \le n$. Let $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ be the storage vector (for flash memory, the received vector is the LLR information), and $\mathbf{z} = (z_1, z_2, \ldots, z_n)$ be the hard decision vector. Let $\mathbf{q} = (q_1, q_2, \ldots, q_n)$ be the quantization vector of $\mathbf{y}$, where $q_j = (q_{j,1}, q_{j,2}, \ldots, q_{j,r})$ for $1 \le j \le n$. For $1 \le l \le q$, define $(a_{l,1}, a_{l,2}, \ldots, a_{l,r})$ as the binary representation of the GF(q) element a_l. These vectors can be extended into r-tuple over GF(2).

Denote $\varphi_{j,l}$ as the initial reliability measure, then it is defined by

$$\varphi_{j,l} = \sum_{t=1}^{r} (2a_{l,t} - 1) q_{j,t}. \tag{1}$$

Define $R_{j,l}^{(0)}$ as the reliability information and is given by

$$R_{j,l}^{(0)} = \lambda \varphi_{j,l}, \tag{2}$$

where λ is a scaling factor.

We use $\phi_{i,j}$ to represent the reliability measure of the extrinsic-information which can be computed by

$$\phi_{i,j} = \min_{j' \in \mathcal{N}(c_i) \backslash j} \max_{l} \varphi_{j',l}, \tag{3}$$

and can be regard as the *extrinsic weighting coefficient*.

Denote the decoded codeword during the k-th decoding iteration as $\mathbf{z}^{(k)} = (z_1^{(k)}, z_2^{(k)}, \ldots, z_n^{(k)})$, $\mathbf{s}^{(k)}$ is the syndrome vector corresponding to $\mathbf{z}^{(k)}$. Then the j-th *predicted* symbol z_j based on the i-th check syndrome, denoted by $\hat{z}_{i,j}^{(k)}$, is given by

$$\hat{z}_{i,j}^{(k)} = h_{i,j}^{-1} \sum_{j' \in \mathcal{N}_{c_i} \backslash j} h_{i,j'} z_{j'}^{(k)}. \tag{4}$$

Algorithm 1: ISRB-MLGD algorithm

Initialization: Set $\mathbf{z}^{(0)} = \mathbf{z}$, compute $\varphi_{j,l}$, $R_{j,l}^{(0)}$, and $\phi_{i,j}$.
1 **for** $k = 0$ *to* $I_{\max}$ **do**
2 Compute the syndrome $\mathbf{s}^{(k)} = \mathbf{z}^{(k)}\mathbf{H}^{\mathrm{T}}$;
3 **if** $\boldsymbol{s}_{1\times m}^{(k)} = (0, \ldots, 0)$ *or* $k = I_{\max}$ **then**
4 Output $\mathbf{z}^{(k)}$ as the decoded codeword and break.
5 **else**
6 Compute $\hat{\mathbf{z}}$ and $\psi_{j,l}^{(k)} = \sum_{\hat{z}_{i,j}^{(k)} = a_l, i \in \mathcal{M}_{v_j}} \phi_{i,j}$;
7 $R_{j,l}^{(k+1)} = R_{j,l}^{(k)} + \psi_{j,l}^{(k)}, 1 \le l \le q.$
8 $z_j^{(k+1)} = \arg \max_{l \in \mathrm{GF}(q)} R_{j,l}^{(k+1)}.$

The calculation of $\hat{z}_{i,j}^{(k)}$ is the message passing procedure. We also denote $\psi_{j,l}^{(k)}$ as the extrinsic-information vector used in updating the reliability measure $\mathbf{R}_j^{(k)}$, then the reliability measure is updated base on $\mathbf{R}_j^{(k+1)} = \mathbf{R}_j^{(k)} + \boldsymbol{\psi}_j^{(k)}$. The ISRB-MLGD algorithm is described in **Algorithm 1** [5].

III. Proposed NB-MLGD Algorithms

A. NB-ATS-MLGD Algorithm

In this section, a nonbinary adaptive 2D scheduling majority logic decoding (NB-ATS-MLGD) algorithm is proposed for the MLC NAND flash memory. For the sake of simplicity, the acronym "MLGD" will be omitted throughout this paper if there is no ambiguity. In the decoding algorithm, we define "message passing" as the algorithm passing the messages from one node (CN or VN) to another node (VN or CN).

We define the stability of the j-th VN v_j as

$$S_{v_j}^{(k)} = \max_{\alpha \in \mathrm{GF(q)}} R_{j,\alpha}^{(k)} - \max_{\beta \in \mathrm{GF(q)}} R_{j,\beta}^{(k)}, \tag{5}$$

where α is the symbol of the VN with maximum reliability and β is the symbol of the VN with second largest reliability. A VN is stable when its stability $S_{v_j}^{(k)}$ is large.

The stability of the i-th CN c_i denoted by S_{c_i} is defined by the minimum stability of the connected VNs, i.e.,

$$S_{c_i}^{(k)} = \begin{cases} \theta, & \min\limits_{j \in \mathcal{N}_{c_i}} S_{v_j}^{(k)} = 0, \\ \min\limits_{j \in \mathcal{N}_{c_i}} S_{v_j}^{(k)}, & \min\limits_{j \in \mathcal{N}_{c_i}} S_{v_j}^{(k)} \neq 0, \end{cases} \tag{6}$$

where θ is a default value to prevent the algorithm from errors.

Each CN is classified into **satisfied** ($s_i = 0$) and **unsatisfied** ($s_i \neq 0$). We first select the unsatisfied CN which has largest stability and not selected before ($P_i = 0$). If all unsatisfied check nodes have been selected, then we select the satisfied check node which has largest check node stability and not selected before (**Algorithm 2** *line* 7-11).

We define the cumulative syndrome O_{v_j} as the amount of the unsatisfied CNs that v_j connected. If v_j is an erroneous VN, the CNs it connected have large probability to be unsatisfied. Similarly, if O_{v_j} is large, the v_j prone to be erroneous. Hence, we can rearrange the VN decoding order according to O_{v_j}. Using d_{c_i} to represent the degree of c_i. So the length of

Algorithm 2: NB-ATS-MLGD algorithm

Initialization: Set the maximum number of iterations to $I_{\max}$. Initialize $R_{j,l}^{(0)} = \lambda\varphi_{j,l}$, $\mathbf{s}^{(0)} = \mathbf{z}^{(0)}\mathbf{H}^{\mathrm{T}}$.

1 **for** $k = 0$ *to* $I_{\max}$ **do**
2 $\mathbf{P}_{1\times m}^{(k)} = (1, \ldots, 1)$;
3 **if** $\boldsymbol{s}_{1\times m}^{(k)} = (0, \ldots, 0)$ **then**
4 Output $\mathbf{z}^{(k)}$ as the decoded codeword and break.
5 **else**
6 **while** SUM($\mathbf{P}$) $\neq 0$ **do**
7 **if** $\exists s_p \neq 0 \cap P_p = 1$ **then**
8 Find the **unsatisfied** CN with largest $S_{c_i}^{(k)}$ and $P_i^{(k)} = 1$, return c_i;
9 **else**
10 Find the CN with largest $S_{c_i}^{(k)}$ and $P_i^{(k)} = 1$, return c_i;
11 Set $P_i^{(k)} = 0$;
12 Obtain the descending decoding order $\mathbf{j} = (j_1, j_2, \ldots, j_{d_{c_i}})$ of VNs;
13 **for** $\nu = 1$ *to* d_{c_i} **do**
14 Calculate the predicted symbol $\hat{z}_{i,j_\nu}^{(k)}$;
15 Update $R_{j_\nu,l}^{(k+1)}$ and $S_{v_{j_\nu}}^{(k+1)}$;
16 $z_{j_\nu}^{(k+1)} = \arg\max_l(R_{j_\nu,l}^{(k+1)})$;
17 **for** *every* $c_{\hat{i}} \in \mathcal{M}_{v_{j_\nu}}$ **do**
18 Update $s_{\hat{i}}^{(k+1)}$ and $S_{c_{\hat{i}}}^{(k+1)}$.
19 Update $\phi_{i,j}^{(k+1)}$.

Algorithm 3: NB-S-ATS algorithm

Initialization: Refer to **Algorithm 2**.

1 **for** $k = 0$ *to* $I_{\max}$ **do**
2 $\mathbf{P}_{1\times m}^{(k)} = (1, \ldots, 1)$;
3 **if** $\boldsymbol{s}_{1\times m}^{(k)} = (0, \ldots, 0)$ **then**
4 Output $\mathbf{z}^{(k)}$ as the decoded codeword and break.
5 **else**
6 Refer to **Algorithm 2** *line* 6-12;
7 **for** $\nu = 1$ *to* d_{c_i} **do**
8 Calculate the predicted symbol $\hat{z}_{i,j_\nu}^{(k)}$;
9 Update $R_{j_\nu,l}^{(k+1)}$;
10 $z_{j_\nu}^{(k+1)} = \arg\max_l(R_{j_\nu,l}^{(k+1)})$;
11 Update $\mathbf{s}^{(k+1)}$, $S_{v_j}^{(k+1)}$, $S_{c_i}^{(k+1)}$ and $\phi_{i,j}^{(k+1)}$;

decoding order vector is d_{c_i}(**Algorithm 2** *line* 12). This is a crucial step in our proposed algorithm. It helps the decoder correct the error as early as possible. We define a vector $\mathbf{P}$ to record the CN whether it has been selected during an iteration. $\mathbf{P}$ is initialized to all $1s$ in every iteration. If c_i has been selected for message passing, then P_i is set to 0.

The reliability measure of extrinsic-information $\phi_{i,j}$ is calculated before the iterative decoding. Hence, it is a fixed value. As the mention above, it represents the *extrinsic weighting coefficient*. To get the dynamic reliability measure of extrinsic-information, the $\phi_{i,j}$ is revised and is given by

$$\phi_{i,j}^{(k)} = \min_{j' \in \mathcal{N}_{c_i} \backslash j} \max_l \lfloor R_{j',l}^{(k)} \cdot \frac{1}{\lambda} \rfloor, \tag{7}$$

where λ is the positive scaling factor used before, $\lfloor\rfloor$ means round down. At the end of iteration, $\phi_{i,j}^{(k)}$ is recalculated. This measure can also apply to ISRB algorithm and the revised algorithm is denoted by ISRBR algorithm.

The proposed NB-ATS is summarized in **Algorithm 2**.

B. NB-Simplified-ATS Algorithm

On account of 2D scheduling, the proposed NB-ATS algorithm needs additional computation to find the effective path for message passing. Consider the complexity and performance of the proposed algorithm, NB-ATS algorithm has been simplified to make it more applicable for NAND flash memory. The stability of CNs and VNs can be updated after the message passing procedure which will significantly reduce the complexity. Similarly, there is no need for calculating the check sum $\mathbf{s}$ immediately. The details of the NB-Simplified-ATS (NB-S-ATS) algorithm is given in **Algorithm 3**.

C. Early Correcting-ATS Algorithm

An adjacent state is defined as the state next to the initial hard-decision state of a flash memory cell. As shown in Fig. 1, if an initial hard-decision state symbol is 0, then the adjacent state symbol is 1. An adjacent state error occurs when a state is decoded as an adjacent state symbol, it more likely to happen compared with non-adjacent ones. The early-correcting (EC) criterion imposes restriction on the decoding results and helps the decoder to distinguish the results. In addition, if non-adjacent state errors occur, it means that there exists large noises in this flash memory cell which cannot be corrected by the EC criterion. Therefore, it is necessary to set a parameter 'I_{MLC}' for stopping the EC criterion. When the iterative number $k \leq I_{\mathrm{MLC}}$, the EC criterion is enabled. Otherwise, the tentative decoding results are adopted. For the MLC NAND flash memory, the EC criterion can be described as follows:

$$\mathcal{Z}_{\mathrm{adj}} = \begin{cases} \{1\}, & z_{\mathrm{ini}} = 0, \\ \{0, 2\}, & z_{\mathrm{ini}} = 1, \\ \{1, 3\}, & z_{\mathrm{ini}} = 2, \\ \{2\}, & z_{\mathrm{ini}} = 3, \end{cases} \tag{8}$$

$$\mathcal{Z}_{\mathrm{pos}} = \begin{cases} \mathcal{Z}_{\mathrm{adj}} \cup z_{\mathrm{ini}}, & k \leq I_{\mathrm{MLC}}, \\ \hat{z}, & k > I_{\mathrm{MLC}}, \end{cases} \tag{9}$$

where z_{ini} represents the initialized hard decisions, $\mathcal{Z}_{\mathrm{adj}}$ represents the results in adjacent states, $\mathcal{Z}_{\mathrm{pos}}$ represents the possible results of decoding and $\hat{z}$ is the predicted symbols.

The EC criterion is performed after the tentative decoding (**Algorithm 2** *line* 14) in the first several iterations, the tentative result is $\hat{z}$. We first screen the decoding result $\hat{z}_{i,j}^{(k)}$. If $\hat{z} \notin Z_{\mathrm{adj}}$, tt could be the wrong result. Therefore, we don't update $R_{j,l}^{(k+1)}$ and perform other propagation steps for

TABLE I
NUMBER OF OPERATIONS IN VARIOUS NON-BINARY DECODING ALGORITHMS

Algorithm	Computation Cost per Iteration				
	FA	FM	IA	IC	RM
ISRB	$2e-m$	$2e$	e	$n(q-1)$	
ISRBR	$2e-m$	$2e$	e	$n(q-1)+e(q-1)(\tau-2)$	n
NB-ATS	$e+e\xi(\tau-1)$	$e+e\xi\tau$	$e(\xi+1)+m$	$e\log_2\tau+e(q-1)(\tau-1)+\frac{(m-1)(m-2)}{2}+e(\log_2 q+q-2+\xi\tau)$	n
NB-S-ATS	$2e-m$	$2e$	$e\xi+m+n$	$e\log_2\tau+e(q-1)(\tau-1)+\frac{(m-1)(m-2)}{2}+n(\log_2 q+q-2)+e$	n
NB-G-ATS	$2e-m$	$2e$	$e\xi+m+n+nqg$	$e\log_2\tau+e(q-1)(\tau-1)+\frac{(m-1)(m-2)}{2}+n(\log_2 q+q-2)+e$	n

FA: Finite-Field Addition; FM: Finite-Field Multiplication; IA: Integer Addition;
IC: Integer Comparison; RM: Real Multiplication.

VN j. When the iteration number exceeds 'I_{MLC}', the decoder receives all the decoding results.

D. Parallel Algorithm

In order to make the proposed algorithms more competitive in NAND flash memory, the parallel ATS algorithm has been proposed. Before selecting the CNs for message passing, all the CNs are divided into g groups in order. For every group, the message passing procedures are the same as ATS. The only difference is that the reliability measure $R_{j,l}^{(k+1)}$ is replaced by $R_{j,l,g}^{(k+1)}$, which means the reliability information is generated by group g. After all the CNs of every group have been selected, then the reliability information of every group should be collected by

$$R_{j,l}^{(k+1)}=\sum_{g'=1}^{g}R_{j,l,g'}^{(k+1)}. \tag{10}$$

The rest of the steps are the same as **Algorithm 3** *line* 11.

E. Complexity Analysis

We analyze the computational complexity of the proposed algorithms. Assume that τ and ξ represent average CN degree and average VN degree, e is the number of elements in the parity-check matrix $\mathbf{H}$ and $e=m\tau=n\xi$, and q is the number of elements in the finite-field. The number of operations in various non-binary decoding algorithms is shown in Table I. In addition, we analyze the average computation in various decoding algorithms for one decoding procedure.

Example: A rate-0.9 length-4544 quasi-cyclic NB-LDPC code over GF(4) ($\xi=5$, $\tau=(50,51)$ and $e=22720$) is used in the simulation. Assume that PE = 17000, then we can compute the average computation of each algorithm in one decoding procedure. The main computation of our proposed algorithm is derived from integer comparison, real comparison and real addition while other operations are in the same order of magnitude compared with ISRB algorithm. The average number of these operations in one decoding procedure is shown in Table II. The computation of integer comparison (IC) almost comes from the calculation of $\phi_{i,j}$. Hence, the operations of these algorithms are the same magnitude in terms of integer comparison. In addition, the IC operations caused by ATS scheme are significantly reduced by NB-S-ATS algorithm and NB-G-ATS algorithm when compared with NB-ATS algorithm.

TABLE II
AVERAGE COMPUTATION IN VARIOUS DECODING ALGORITHMS FOR ONE DECODING PROCEDURE

Algorithm	Average Computation	
	IA	IC
ISRB	1.78×10^5	3.35×10^6
ISRBR	1.38×10^5	1.80×10^7
NB-ATS	3.81×10^5	2.68×10^7
NB-S-ATS	3.78×10^5	1.36×10^7
NB-G-ATS	6.47×10^5	1.48×10^7

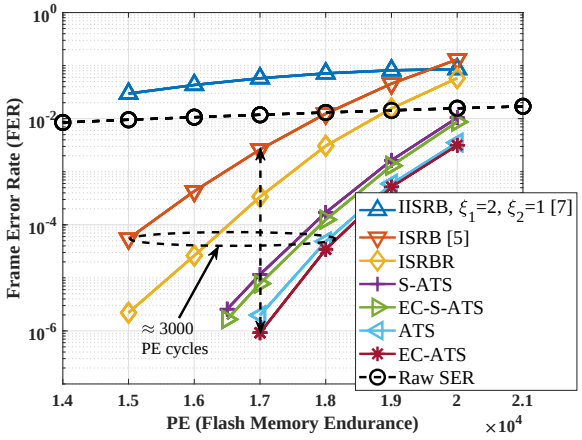

Fig. 2. FER performance comparison for different algorithms with the LDPC 4k-code when using 15-level MMI quantization scheme.

IV. SIMULATION RESULTS

In this section, we present the simulation results with NB-MLGD decoding algorithms proposed in Section III. In the simulations, we use 15-level and 6-level MMI quantization scheme and the LLR information is uniformly quantized by the 5 number of bits. We use the symbols of 4K-LDPC code over GF(4) to represent the 4 storage states of the MLC flash memory. The scaling factor λ is set to 5 and $\theta=10$. The maximum number of iterations $I_{\max}$ is set to 50. The EC parameter I_{MLC} is set to 1.

Fig. 2 shows the frame error rate (FER) curves of the conventional algorithms and the proposed algorithms using the 15-level MMI quantization scheme. With the adaptive 2D scheduling, the proposed algorithms have excellent performance compared with the ISRB algorithm and IISRB algorithm in terms of FER performance. Since the LDPC code we use in the simulation has large row-weight, the performance of IISRB algorithm is unsatisfactory. In addition, the *extrinsic*

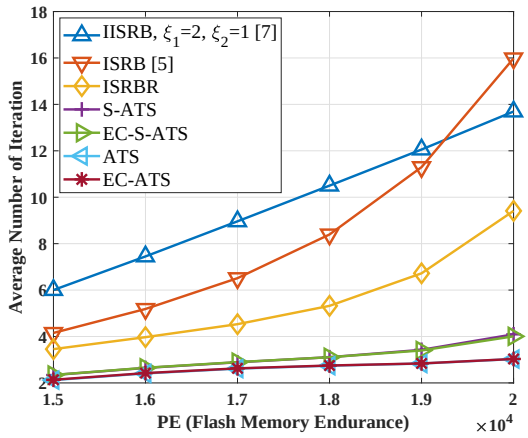

Fig. 3. The average number of iterations for different algorithms to converge with the LDPC 4k-code when using 15-level MMI quantization scheme.

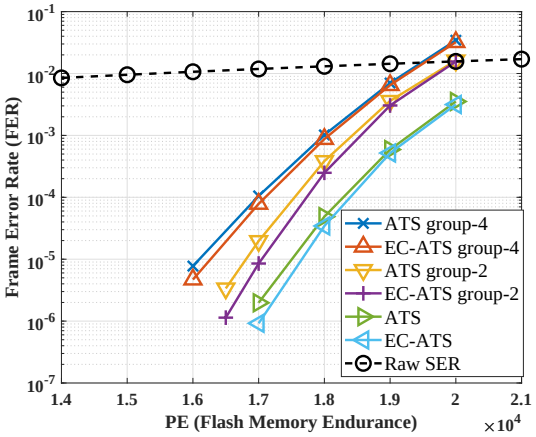

Fig. 4. FER performance comparison for different group-levels with the LDPC 4k-code when using 15-level MMI quantization scheme.

weighting coefficient updates after every iteration which provides more reliable information to the proposed algorithms. At FER of 10^{-4}, the proposed algorithms increase the lifetime of the MLC NAND flash memory by more than 3000 program-and-erase (PE) cycles. When PE $= 17000$, the NB-EC-ATS algorithm improves nearly 6 order of magnitudes compared with IHRB algorithm and more than 3 order of magnitudes compared with ISRB algorithm in FER performance. The NB-S-ATS algorithm and NB-EC-S-ATS algorithm are the simplified version of NB-ATS and NB-EC-ATS, respectively. They significantly reduce the complexity of decoding while loss a little error correcting capability.

Fig. 3 clearly shows that the proposed algorithms converge faster than other algorithms. The proposed algorithms have 2D scheduling which exploit the up-to-date information and find the right direction for message passing. Meanwhile, the reliability measure of extrinsic-information is dynamic for every iteration. These measures accelerate the convergence speed of the proposed algorithms.

In Fig. 4, we plot the FER curves of the parallel algorithms for different PE cycles. As the number of groups increases, the performance of the algorithms have some losses. Nevertheless, the parallel algorithms are still better than the conventional algorithms in terms of FER performance, which make the proposed algorithms more competitive in NAND flash memory. According to performance needs for NAND flash memory, we can adjust the group number of our proposed algorithms.

V. Conclusions

In this paper, several adaptive 2D scheduling NB-MLGD algorithms have been proposed for the MLC NAND flash memory. Based on ATS, the probability of the proposed algorithms passing message in a correct direction increases and the up-to-date information can be exploited in time. By observing the property of the MLC NAND flash memory, a NB-EC-ATS algorithm is proposed which exploits the MLC NAND flash memory property for correction in the early iteration. Compared with the conventional MLGD algorithms, the proposed algorithms significantly enhance the FER performance and accelerate the convergence speed. In addition, the EC criterion is able to extend to TLC NAND flash memory without a hitch. In order to make the proposed algorithms more practical in NAND flash memory, we simplify the proposed algorithms and make them more parallel, which improve the throughput and still achieve admirable FER performance.

References

[1] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error characterization, mitigation, and recovery in flash-memory-based solid-state drives," *Proc. IEEE*, vol. 105, no. 9, pp. 1666–1704, Sep. 2017.

[2] G. Dong, N. Xie, and T. Zhang, "On the use of soft-decision error-correction codes in NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 2, pp. 429–439, Feb. 2011.

[3] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE Trans. Commun.*, vol. 58, no. 5, pp. 1365–1375, May 2010.

[4] C. L. Lin, S. W. Tu, C. L. Chen, H. C. Chang, and C. Y. Lee, "An efficient decoder architecture for nonbinary LDPC codes with extended min-sum algorithm," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 63, no. 9, pp. 863–867, Sep. 2016.

[5] C. Y. Chen, Q. Huang, C. C. Chao, and S. Lin, "Two low-complexity reliability-based message-passing algorithms for decoding non-binary LDPC codes," *IEEE Trans. Commun.*, vol. 58, no. 11, pp. 3140–3147, Nov. 2010.

[6] S. Yeo and I. C. Park, "Improved hard-reliability based majority-logic decoding for non-binary LDPC codes," *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 230–233, Feb. 2017.

[7] C.Xiong and Z. Yan, "Improved iterative hard-and soft-reliability based majority-logic decoding algorithms for non-binary low-density parity-check codes," *IEEE Trans. Signal Processing*, vol. 62, no. 20, pp. 5449–5457, Oct. 2014.

[8] P. Radosavljevic, A. de Baynast, and J. R. Cavallaro, "Optimized message passing schedules for LDPC decoding," in *Proc. 39th Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA, USA, Oct. 2005, pp. 591–595.

[9] A. I. V. Casado, M. Griot, and R. D. Wesel, "LDPC decoders with informed dynamic scheduling," *IEEE Trans. Commun.*, vol. 58, no. 12, pp. 3470–3479, Dec. 2010.

[10] Y. Gong, X. Liu, W. Ye, and G. Han, "Effective informed dynamic scheduling for belief propagation decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 59, no. 10, pp. 2683–2691, Oct. 2011.

[11] C. A. Aslam, Y. L. Guan, and K. Cai, "Read and write voltage signal optimization for multi-level-cell (MLC) NAND flash memory," *IEEE Trans. Commun.*, vol. 64, no. 4, pp. 1613–1623, Feb. 2016.

[12] J. Wang, K. Vakilinia, T. Y. Chen, T. Courtade, G. Dong, T. Zhang, H. Shankar, and R. Wesel, "Enhanced precision through multiple reads for LDPC decoding in flash memories," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 880–891, May 2014.