This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2016.2626446, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Asymmetric Error Rates of Cell States Exploration for Performance Improvement on Flash Memory based Storage Systems

Edwin H.-M. Sha, Congming Gao, Liang Shi*, Kaijie Wu, Mengying Zhao, and Chun Jason Xue

***Abstract*—Recent studies show that an MLC flash cell in different states suffers from diverse error patterns in varying degree. That is, the error rates of each page are highly dependent on the data content. Consequently, pages with different data will exhibit quite different error rates. However, existing technologies equipped with one uniform ECC scheme for all pages in a flash memory do not take the different error rates of pages into consideration. In this paper, we propose to exploit the asymmetric error rates of flash memory exhibited by the flash pages with different data for performance improvement. Before a page is programmed, its specific error rates, called Content-Dependent Bit Error Rates (CDBER), are estimated according to the content of the page. The margin between the CDBER of a page and the maximal error rates correctable by the uniform ECC code is exploited for performance improvement. On one hand, a faster and suitable write operation is selected to speed up the progress of programming while the increased speed induced CDBER does not exceed the maximal correctable error rates. On the other hand, a light-weight ECC scheme can be chosen for a faster read operation since the page decoding process of a light-weight ECC scheme incurs less time overhead. Finally, a state mapping scheme, which further reduces the CDBER through mapping the high error rate states to the low error rate states of a page, is proposed. Simulation results show that the proposed approaches lead to significant write and read performance improvement.**

***Index Terms*—Asymmetric Error Rates, CDBER, ECC, BCH, State Mapping, Flash Memory.**

I. Introduction

During the past decades, the technology of flash memory advances with technology scaling and increasing bit density. Recently, 10nm flash memory has been delivered to the market [20] and the number of bits per cell is increased from 1 bit to the most recent 6 bits [16]. However, as the bit density increases with technology scaling, more states are represented by a cell where the noise margin between adjacent states of the cell is getting smaller. That is, the reliability of flash cell is getting worse under a smaller noise margin, which directly induces significant performance degradation [10] [14]. In order to realize a reliable program in such cells, one has to choose a smaller programming step size in the commonly used Incremental Step Pulse Programming (ISPP) scheme [32], which increases the programming latency [28] [19] [27]. In addition, read performance is also significantly related with the ECC scheme equipped in the flash memory. A heavier-weight ECC scheme is always demanded by the worse reliability incurs more decoding latency in the read process [35] [11] [36] [12] [17] [18]. In this work, we propose techniques to improve the write and read performance of MLC flash memory based storage systems.

Flash memory uses floating gates to store electrons. The number of electrons charged in the floating gates is represented as voltage of the floating gates, which determines the state of the cell. Each state of the cell represents different data. Previous works have shown that the error rates of a flash memory cell are highly dependent on the states - a cell in different states exhibits quite different error rates [3] [6] [5] [15]. For example, Cai et al. [3] examined the error characteristics at 30-40nm technology nodes and found that the error sources had different impacts on the error rates of a flash memory cell in different states. Wang et al. [34] also observed that a flash memory cell programmed to a state with a lower voltage threshold had smaller Bit Error Rates (BER). Yaakobi et al. [39] found in their experiments that the erroneous voltage shifts at different states were different, and the difference would be enlarged if more bits (hence more states) were packed into a single cell [38]. It can be derived that flash pages with different contents (thus different distributions of cells' states) exhibit quite different error rates. Despite the dramatic differences, however, the existing technology uses a uniform ECC scheme for all pages in a flash memory. The chosen ECC scheme must be powerful enough to handle the worst case - the case where all the data in a page are represented by the state with the highest error rates. However, the error rates of each page are different with each other according to various programmed data, which introduce a margin between the error rates of a page and the maximal error rates correctable by the uniform ECC code.

In this work, we propose techniques to improve flash performance by exploiting the error margin between the specific error rates of a page and the capability of ECC code. Before a page is programmed, its specific error rates, referred to as

An earlier version of this paper was presented at the 32nd IEEE International Conference on Computer Design (ICCD), 2014 [13].

L. Shi, C. Gao, K. Wu and E. Sha are with the Key Laboratory of Cyber Physical Society Credible Service Computing, Ministry of Education, and College of Computer Science, Chongqing University, Chongqing, P.R. China. E-mail: shi.liang.hk@gmail.com

M. Zhao is with the School of Computer Science and Technology, Shandong University, P.R. China.

C. Xue is with the City University of Hong Kong, Kowloon, Hong Kong.

This work is partially supported by the Fundamental Research Funds for Graduate Scientific Research and Innovation Foundation of Chongqing, China(Grant No.CYB14043), Graduate Scientific Research and Innovation Foundation of Chongqing, China (Grant No.CYS16019), the Fundamental Research Funds for the Central Universities (106112016CDJZR185512 and 106112014CDJZR185502), NSFC (61402059, 61472052 and 61572411), and National 863 Programs 2015AA015304.

Content-Dependent Bit Error Rates (CDBER), are estimated based on the data content. And hence, the margin between a page's CDBER and then maximal error rates correctable by the uniform ECC code is exploited to improve the performance of flash memory. In addition, the performance of flash memory can be further improved by applying a state mapping scheme. The basic idea is that if the high error rate states are the majority part of a page, the CDBER of the page can be reduced through mapping the states to lower error rate states. In summary, this work makes the following contributions:

- Observed that the error rates of the flash pages are highly related with the page content. The page content related error rates of flash pages are taken into consideration and exploited for performance improvement in this work;
- Proposed a CDBER with the understanding of the asymmetric state error rates of flash memory cells;
- Proposed a write speed improvement approach with exploiting CDBER;
- Proposed a read performance improvement approach through an ECC selection scheme;
- Proposed a state mapping scheme to further reduce the CDBER of each page;
- Presented an efficient implementation of the proposed approaches with negligible overhead. Experimental results show that the proposed approaches achieve significant performance improvement.

The rest of the paper is organized as follows. Section II is the background and related work. Section III presents the motivation of the proposed work. Section IV discusses the proposed CDBER exploiting approaches. Experiments and result analysis are presented in Section V. Finally, Section VI concludes the paper.

II. Background and Related Work

In this section, the background on flash memory and related works are presented.

A. Flash Memory

Different from the hard disk media, flash memory uses floating gate (FG) to store data. The data stored in the flash memory cells are represented by the amount of charges trapped in the FG. Based on the number of bits stored in one cell, flash memory can be classified into two types: single level cell (SLC) with one bit per cell and multi-level cell (MLC) with more than one bit per cell [16]. In this work, we focus on the MLC flash memory since its has higher bit density and lower cost.

Currently, three types of operations can be issued to flash memory: read, program, and erase. During read operations, the charges in the FG are sensed with a low voltage. If there are multiple bits in a cell, the read process would be long since it needs to go through several stages to identify the multiple bits in a cell. In this case, MLC flash memory has much longer read latency than that of SLC flash memory. During program operations, a high voltage is issued to charge the cell for the specific states. The program operations are always processed by an ISPP scheme [32], which is designed to do iteration charging and checking to make sure that the FG is charged to the specific states. In this case, MLC flash memory also has much longer write latency than that of SLC flash memory. During ease operations, the charges trapped in the cell are removed by applying a high voltage for a long time. Finally, the charges in the cell are cleaned. Note that read and write operations are executed in the unit of flash page, while erase operations are executed in the unit of flash block. Several flash pages compose one flash block. In this work, we propose approaches to reduce the read and write latencies for performance improvement.

B. Errors of Flash Memory and Modeling

In this section, we first give a brief discussion to the different errors of flash memory. Then, an introduction to the flash memory modeling is presented to show how to model the flash memory with considering several types of errors.

1) Errors of Flash Memory: Flash memory is prone to errors for several reasons. The key reason stems from the physical characteristic of flash memory cell. Recently, several types of errors have been identified and studied around the physical characteristic, including cell damages induced by program and erase cycling (P/E cycling) [22] [3] [28], current leakages during retention time [27] [19] [6] [4] [25], voltage increases introduced by cell-to-cell interferences [7], and voltage fluctuations with temperature variations [4] [8]. In the following, short introduction is presented for these error sources.

Firstly, for P/E cycling, charges removing during the P/E cycling can be trapped to the oxide layer, which would increase the cell's voltage [22] [3] [28]. In addition, the large voltage added to the cell during the P/E cycling would introduce damages to the cell. This is one of the key reasons for the limited P/E cycles for the flash memory [25]. Secondly, charges can be leaked from the cell during retention time, which was especially the case when there were more charges in the cell [34]. Thirdly, for the other types of error sources, such as cell-to-cell interference, when the other cells beside the target cell are programmed, the charges in the target cell would be increased [7].

In the following, we will give a brief introduction on the modeling of a flash memory device with considering these errors.

2) Flash Memory Modeling: In order to study the error characteristics of flash memory, several flash memory models haven been proposed [28] [27] [19]. For example, Pan et al. [28] [27] proposed to model the voltage distributions for an MLC flash memory with two bits per cell. The four states, "11" for the erase state, "10,", "10", "01", and "00" for the other three states were carefully modeled. Liu et al. [19] also proposed a flash memory model. Different from Pan et al. [28], they used unified distributions for all states. In this work, the flash memory model designed in [28] is applied because it has been widely used in previous works [40] [30] and the four states are clearly differentiated. In the following, a brief introduction to the flash memory model in [28] is presented.

Pan et al. [28] proposed to model the flash memory in two steps. The first step is to model the basic voltage distributions

of flash memory cell, where no error impact factors are considered. The second step is to add the error impact factors to the basic model. Pan et al. [28] added three types of error impact factors, including P/E cycling, current leakages during retention time and cell-to-cell interference, which were the dominate and common error sources for general flash memory devices. There are more details and parameters for references in the work [28].

C. Related Work

Several works have been proposed to exploit the error characteristics for improving the read and write performance of flash memory. Based on the types of exploited error factors, these works can be grouped at least into four types: retention time exploration [27] [19] [31], P/E cycling exploration [28] [6], cell-to-cell interference exploration [7] [5] [10] and temperature impact exploration [8] [21].

Pan et al. [27] and Liu et al. [19] proposed to exploit the retention time to improve the write performance of flash memory. They proposed to relax the retention time of flash memory and improved the write performance by increasing the program step size of ISPP. Shi et al. [31] further exploited the retention time during each P/E cycling process for lifetime improvement. Pan et al. [28] and Cai et al. [6] proposed to exploit the P/E cycles to improve the write performance and lifetime of flash memory. They proposed to use larger program step size or lower refresh frequency at the initial stages and increase them gradually to guarantee the reliability. For the cell-to-cell interference exploration, Cai et al. [7] [5] and Dong et al. [10] proposed approaches to identify it and use it for reliability or performance improvement. For the temperature exploration, Chen et al. [8] and Meza et al. [21] found that the temperature had big impact to the reliability of flash memory cell. They proposed to use the temperature for cell recovering or limit the impact of temperatures.

Different from the above mentioned works, this work is proposed based on a new type of error characteristic, asymmetric error rates of flash pages, which are not only related with the asymmetric error rates of four states, but also determined by the page content. Compared with previous works, the proposed work improves the performance via exploiting the asymmetric error rates of flash pages. Due to the various error rates of flash pages, the uniform error-correctable ability provided by ECC scheme can be exploited by accelerating the program speed of flash pages with lower error rates. In addition, the read performance can also be improved via encoding the flash pages which suffer from lower page error rates with lightweight ECC schemes.

III. Motivation

Based on the studies of previous works, we find that the BER of a flash memory page is not only determined by the error rate of each state, but also the distributions of the states of the cells in this page. In this paper, the error rate of a page is referred to as the page's Content-Dependent Bit Error Rates (CDBER). As shown in [3], a page with all its cells programmed to "01" or "00" states exhibits the worst-case error rates. In order to guarantee the reliability of flash memory under the case where all pages are programmed with the highest error rate states, a powerful ECC scheme, equipped with the error correctable ability for the worst case across all flash pages, is chosen. However, such a uniform powerful ECC scheme is overkill for the pages that are not filled with all the highest error rate states. In fact, these pages are quite common according to our preliminary study given below.

TABLE I: The statistics of eight typical application files.

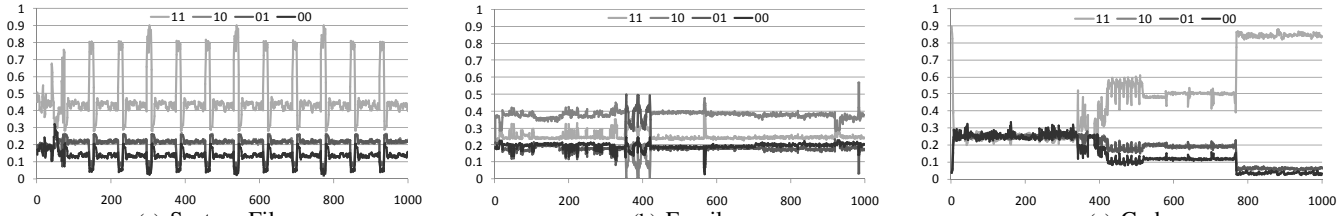
Applications	File Sizes	11 States (%)	10 States (%)	01 States (%)	00 States (%)
System Files	$28.2M$	13.67	17.78	19.57	48.98
Email	$9.15M$	19.15	36.12	19.50	25.23
Codes	$40.5M$	17.16	15.78	16.28	50.78
Movie	$194M$	28.24	23.73	23.61	24.42
DB Data	$45.6M$	20.73	19.74	19.48	40.05
Game File	$75.8M$	20.47	24.60	24.60	30.33
Music	$55.3M$	20.98	23.34	23.43	32.25
Office File	$21.2M$	22.74	20.80	20.25	36.21

In our preliminary study, we analyzed several kinds of files, including email, system files and codes, in order to examine the distributions of the cell states. In addition to these files, several other files, including songs, games, databases, movies and documents, were also evaluated in our experiments and found similar characteristics, which were not presented in the paper. And a four-state MLC flash memory cell was considered. Similar results could be expected for the flash memory with more states. The distribution of the four states in each page was collected. Figure 1 showed the distribution of the four states for the three typical application files. The x-axis represented the page number and the y-axis represented the percentages of the four states in each page. As shown in Figure 1, the distribution of the four states varied significantly from page to page and file to file. For example, system files had much more "11" compared with the other states. For emails, "10" was the dominant state. Table I shows the statistics of eight application files respectively. Similar to the presented three typical application file, the distributions of four states of the other five application files vary significantly as well. Based on these results, the CDBERs of different pages varied significantly as well, and most of them were much smaller than the worst-case error rates. The difference between the worst case and CDBERs was exploited for performance improvement in this work.

IV. Exploiting Asymmetric Error Rates of Cell States for Performance Improvement

In this section, the asymmetric error rates of cell states are exploited for performance improvement. Content-Dependent Bit Error Rates (CDBER) model is first proposed to build the relationship between the states of cells in a page and the estimated BER of this page. The margin between the CDBER of a specific page and the uniform error-correcting capability provided by current flash chips is then exploited for write (Section IV-B1) and read (Section IV-B2) performance improvement, respectively, while satisfying the original reliability requirements. In addition, a state mapping scheme (Section IV-B3) is proposed to reduce the CDBER of each page to further improve the performance of reads and writes. Finally,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2016.2626446, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



(a) System Files (b) Email (c) Codes

Fig. 1: State distributions of three typical application files: The x-axis represents the page number of the data and y-axis represents the percentages of each state.

an efficient implementation of the proposed approaches is presented.

A. Content-Dependent Bit Error Rate Model

Figure 2 shows the voltage distributions of an MLC flash memory cell. In this figure, x-axis represents the voltages, and y-axis represents the probability distributions of programmed voltages. Three reference voltages are predefined to differentiate the four states ($V_p^{(k)}, k = 0, 1, 2$). The width of the voltage distribution of each programmed state is ΔV_{pp}, which is the program step size applied in the process of program operation. Based on the voltage distributions of the flash memory cell, we first discuss the characteristics of flash errors. Then, the CDBER model is presented.

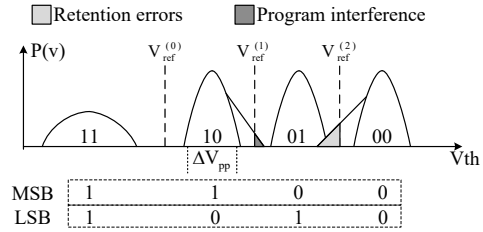


Fig. 2: The voltage distribution for MLC flash memory with voltage fluctuations and shifts due to different flash error sources.

1) Flash Memory Error Characteristics: Due to various factors, the stored electrons in flash cells could be increased or decreased, which causes a right or left shift from its states, respectively. The left shift is dominated by charge leakage, and the right shift is dominated by the interferences from programming this and neighborhood cells and P/E cycling. Although the left shift and right shift exist in all four cell states, the impact caused by these error factors varies from state to state. That is, the error rate caused by the left shift in the state with larger charged electrons is higher than other states. On the contrary, the error rate caused by the right shift in the state with fewer or none electrons is also higher than other states. In this work, we use two vectors to represent the BERs caused by Left Shift and Right Shift, respectively, where $BER_L^{(b_1b_2)}$ ($BER_R^{(b_1b_2)}$) represents the BERs caused by Left (or Right) Shift at the state b_1b_2.

$$V(BER_L) = (BER_L^{(11)}, BER_L^{(10)}, BER_L^{(01)}, BER_L^{(00)}); \quad (1)$$

$$V(BER_R) = (BER_R^{(11)}, BER_R^{(10)}, BER_R^{(01)}, BER_R^{(00)}); \quad (2)$$

Note that $BER_L^{(11)}$ and $BER_R^{(00)}$ are 0 since left shift at state "11" and right shift at state "00" do not affect state reading. Then, the aggregated error rate of a flash memory cell with the four states is represented as follows;

$$V(BER) = V(BER_L) + V(BER_R); \quad (3)$$

The flash device model proposed in [28] can be extended to identify the error rates for $V(BER_L)$ and $V(BER_R)$, respectively. The extension for extracting $V(BER_L)$ and $V(BER_R)$ is as follows: First, during the extraction of $V(BER_L)$ and $V(BER_R)$, other error sources should be set with their worst cases to make sure that the error rates are exclusively obtained from the asymmetric states. Second, $V(BER_L)$ and $V(BER_R)$ are collected by setting the specific state. Finally, $V(BER_L)$ and $V(BER_R)$ are collected and can be varied with specific parameters, such as ΔV_{pp}, which will be exploited in the following.

2) Content-Dependent Bit Error Rate Model: CDBER are highly dependent on the error rates of each state in the flash cell and data stored in the flash page. As discussed in IV-A1, the error rates of each state are different with each other due to the various error factors. $BER_L^{(b_1b_2)}$ and $BER_R^{(b_1b_2)}$ are used to represent the error rates caused by state shifts at the state b_1b_2. For the distributions of the four states, the difference is commonly exist among pages and applications as shown in 1. Vector $V(P_i)$ is used to represent the distributions of the four states in a flash page i, where $V(P_i)$ =$(P_{11}, P_{10}, P_{01}, P_{00})$, and $P_{b_1b_2}$ ($b_1 \in \{0, 1\}$, $b_2 \in \{0, 1\}$) represent the percentages of the states b_1b_2 in the page.

An issue in the computation of the CDBER needs to be taken into consideration: Each word line in MLC flash memory stores the information of two data pages that are referred to as Most Significant Bit (MSB) and Least Significant Bit (LSB) pages, respectively. As shown in Figure 2, the LSB of a cell is identified by the threshold voltages $V_{ref}^{(0)}$, $V_{ref}^{(1)}$ and $V_{ref}^{(2)}$, and the MSB of the same cell is identified by the threshold voltage $V_{ref}^{(1)}$. The errors of the two bits originate from the erroneous shifts at different states. Hence the CDBER of the LSB page and the CDBER of the MSB page in a word line must be computed separately.

CDBER *Model of MSB Pages:* For the MSB of a cell, errors occur when the state of the cell shifts from "10" to "01" (right shift, $BER_R^{(10)}$) or from "01" to "10" (left shift, $BER_L^{(01)}$). In a word line, let us denote the percentages of the cells in the states "10" and "01" as P_{10} and P_{01}, respectively. Then, the CDBER of an MSB page is computed as follows:

$$CDBER_{MSB} = P_{10} \times BER_R^{(10)} + P_{01} \times BER_L^{(01)}; \quad (4)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2016.2626446, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

CDBER *Model of LSB Pages:* For the LSB of a cell, there are three cases: errors due to state shifting between "11" and "10", "10" and "01", and "01" and "00". Similar to an MSB page, the computation of the CDBER of an LSB page should take six cases into consideration, as shown below:

$$\begin{aligned} CDBER_{LSB} = {} & P_{11} \times BER_R^{(11)} + P_{10} \times BER_L^{(10)} \\ & + P_{10} \times BER_R^{(10)} + P_{01} \times BER_L^{(01)} \\ & + P_{01} \times BER_R^{(01)} + P_{00} \times BER_L^{(00)} \end{aligned} \tag{5}$$

We compare the default ECC capability, which is designed to support the worst-case BER of the LSB and MSB pages with the CDBER of pages in Figure 3. The CDBER of the application files are separately computed by the CDBER models of the LSB page and the MSB page. During the computation, the $V(P_i)$ is collected based on the specific files presented in Section III, and the two vectors $V(BER_L)$ and $V(BER_R)$ are collected with default settings, based on the models presented in Section II. The difference between the BER correctable by a popular ECC scheme and the CDBER of the application files is presented in Figure 3. The detail settings will be presented in the experiment section. The highest line in Figure 3 is the BER that can be corrected by the ECC scheme – BCH (34496, 32768, 55). The error correction capability is in the range of the codes used in [6] [35] [36] [19] [27]. The second and third highest lines are the worst-case CDBER of the LSB and MSB pages, respectively. The rest lines are the CDBER of the LSB and MSB pages of those application files. We can make the following observations:

- Observation 1: The CDBER of pages are far below the BER correctable by a popular ECC code;
- Observation 2: The CDBER of pages are far below the worst-case CDBER of MSB and LSB pages;
- Observation 3: The CDBER of pages vary significantly among different pages, which is consistent with the observations of the state distributions in Figure 1.

Based on these observations, three approaches are proposed to improve the write or read performance, respectively, in the following.

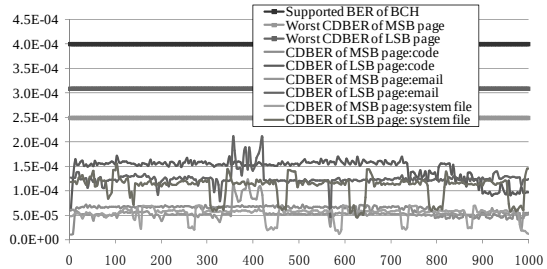


Fig. 3: CDBER of MSB and LSB pages against the default ECC capability for different files.

B. Exploiting CDBER for Performance Improvement

The large redundancy shown in the above observations leads us to two options on performance improvement: write performance and read performance. Figure 4 presents a flow for these approaches. The write and read performance improvement approaches will be first presented to exploit the large exploration space (Observation 1 and 2). Then, we discuss a state mapping scheme, which is an extension to the basic approaches by further exploiting the significant CDBER variations among different pages (Observation 3). The details are discussed as follows.

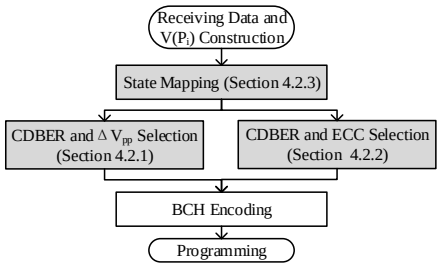
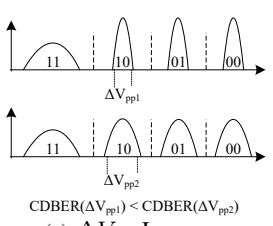


Fig. 4: The flow for the proposed approaches.

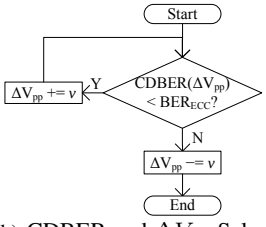
1) Write Performance Improvement: The first option is to improve the write performance by increasing the step size of program operation (ΔV_{pp}). The speed of program operation is related with the step size ΔV_{pp} in ISPP [32]. Compared with a smaller ΔV_{pp}, a larger one is able to program the cell to the specific voltage faster. However, the noise margin under applying the larger ΔV_{pp} is going to get smaller, which reduces the error tolerable ability for the error induced state shifts. Therefore, a larger ΔV_{pp} usually results in a higher BER, which in turn results in a higher CDBER of the programmed page. The increased CDBER is tolerable as long as it does not exceed the correctable BER by the ECC scheme.

In our approach, the ΔV_{pp} is determined based on the identified CDBER of the page to be programmed using the model proposed in [28]. A larger ΔV_{pp} (hence faster write speed) is assigned for the pages with lower CDBER, and vice versa. Although it would be ideal if one could adjust ΔV_{pp} in a fine granularity, a more practical solution is to use a few discrete levels for ΔV_{pp}. When the CDBER of a page to be programmed is determined, we select a suitable ΔV_{pp} according to the discrete levels, where the range between zero and the worst-case CDBER of flash pages is divided among several parts.

The above discussions are illustrated in Figure 5. In Figure 5(a), with increased ΔV_{pp}, the noise margin is reduced, which introduces a higher CDBER. Figure 5(b) is the process for computing CDBER and selecting a suitable ΔV_{pp}. ΔV_{pp} is increased by a small step size *v* each time and checked against with the capability of ECC. With this approach, the CDBER can be fully exploited to speed up the write requests within the capability of ECC.



(a) ΔV_{pp} Increases



(b) CDBER and ΔV_{pp} Selection

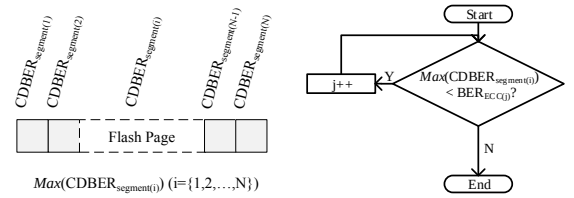
Fig. 5: Write Speedup with Increased ΔV_{pp}.

2) Read Performance Improvement: Alternatively, read latency can be reduced by exploiting the redundant error correction capability. Instead of using single ECC code for a whole

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2016.2626446, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

page, one can choose to partition the page into equal-length segments and protect each segment independently using a lighter-weight ECC code. During read operations, the decoding latency of a segment overlaps the fetching latency of the next segment, thereby reducing the overall read latency. However, using a lighter-weight ECC scheme for each individual segment delivers an inferior capability to the original ECC for the whole page, assuming that the total number of coding bits are same for both. In this subsection, we exploit CDBER to realize the use of lighter-weight ECC codes. Hence, we propose a smart ECC selection scheme to improve read performance without the cost of write performance.

In our approach, a page is also partitioned into multiple equal-size segments. The CDBER of each segment is calculated, and the worst-case CDBER among all the segments is used to find a matching ECC scheme. However, one should note that, on the one hand, the larger the number of segments, the less the number of check bits for each segment, hence the weaker error correction capability the applicable ECC schemes can provide. On the other hand, the larger the number of segments, the smaller the size of each segment, hence the higher the worst-case CDBER among segments. These two conflicting trends could eventually lead to the possibilities that, with the limited checking bits of each segment, there are no suitable ECC schemes that can support the worst-case CDBER of the segments! In this case, the proposed approach simply switches to a smaller number of segments while applying a stronger ECC scheme. During read operations, the data are read based on the selected ECC scheme. With this approach, the write speed of flash memory is not impacted and the read performance can be improved.



(a) $Max(CDBER_{segment(i)})$ (b) CDBER and ECC Selection

Fig. 6: Read Speedup with Light-Weight ECC Selection.

The above discussions are illustrated in Figure 6. In Figure 6(a), the flash page is partitioned into N equal-size segments. The required correction capability for the flash page should be the maximal CDBER of the N segments, $Max(CDBER_{segment(i)})$. Figure 6(b) is the process for the ECC selection scheme. Multiple ECC codes are checked to match $Max(CDBER_{segment(i)})$. With this approach, the CDBER can be fully exploited to select a lighter-weight ECC scheme.

3) State Mapping Scheme: Observation 3 in Section IV-A2 shows that the CDBERs of different pages vary significantly. The reason for the significant variations is mainly from that the distributions of the cell states in the pages are different. As shown in the CDBER model, the CDBER of a page is determined by two factors: the asymmetric error rates of different states and the percentages of different states. If the high error rate state takes the majority part of the page, CDBER is high and vice versa. Based on this observation, we propose a state mapping scheme to further reduce the CDBER of a page for performance improvement.

Previous works on state mapping mostly focus on emerging non-volatile memories, such as phase change memory (PCM) and spin-transfer torque magnetic random-access memory (STT-RAM) [9] [23] [33] [41] [24]. In this work, we propose to exploit the asymmetric error rate states of flash memory. In this case, the basic idea of the state mapping scheme is to reduce the percentages of the high error rate states. The approach is simple and presented as follows. First, the percentages of the four states are collected during the transferring of data from the host bus, as shown in Figure 4. Then, the state mapping is determined through mapping the high percentage state to the low error rate state and low percentage state to the high error rate state. Take Figure 7 as an example. The distributions of the four states of a flash page to be programmed is collected during the data transferring. Assume that the relationship for the percentages of the four states is $P_{01} > P_{10} > P_{11} > P_{00}$. Then, based on the error rates of each state and the percentage relationship, the mapping is as follows: "01" $\rightarrow$ "11", "10" $\rightarrow$ "10", "11" $\rightarrow$ "01", and "00" $\rightarrow$ "00". There are totally 24 mapping types for four states. This approach is different from previous works in non-volatile memories since the cost is insensitive for flash memory as secondary storage.

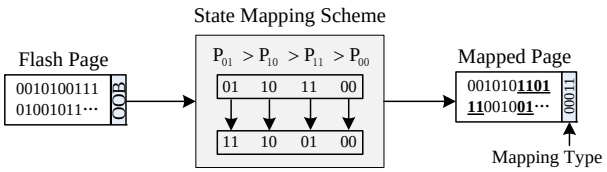


Fig. 7: State Mapping Scheme.

Comparing with the original flash page, the mapped page has a smaller CDBER, which enlarges the margin between the CDBER and error correction capability of the ECC scheme. Hence, there exist possibilities to further increase the step size of program operations and reduce the read latency by using a lighter-weight ECC scheme.

C. Implementation

This section presents the implementation details of the proposed approaches.

In the state-of-the-art FTL, the floating point computing capability is not widely supported. Hence, the CDBER models are relaxed in the implementation by removing the floating point computation. With computing the BERs of four states, we observed that the BER caused by left shift at the state 00 is greater than the BER caused by left and right shift at the state 01, which is far greater than the BER caused by right shift at the state 11, and the BER caused by right shift at the state 10 is the minimal one. Based on this important observation, the process in relaxing the computation of CDBER is presented as follows:

Relaxed CDBER Model of MSB Pages:

$$\begin{aligned} &CDBER_{MSB} \\ &= P_{10} \times BER_R^{(10)} + P_{01} \times BER_L^{(01)} \\ &< (1 - P_{01}) \times BER_R^{(10)} + P_{01} \times BER_L^{(01)} \\ &= BER_L^{(01)} - BER_R^{(10)} + P_{01} \times BER_L^{(01)} \end{aligned} \tag{6}$$

Relaxed CDBER Model of LSB Pages:

$$\begin{aligned} &CDBER_{LSB} \\ &= P_{11} \times BER_R^{(11)} + P_{10} \times (BER_L^{(10)} + BER_R^{(10)}) \\ &+P_{01} \times (BER_L^{(01)} + BER_R^{(01)}) + P_{00} \times BER_L^{(00)} \\ &< (P_{11} + P_{10}) \times BER_R^{(11)} + (P_{00} + P_{01}) \times BER_L^{(00)} \\ &= (1 - P_{00} - P_{01}) \times BER_R^{(11)} \\ &+(P_{00} + P_{01}) \times BER_L^{(00)} \\ &= BER_R^{(11)} + (P_{00} + P_{01}) \times (BER_L^{(00)} - BER_R^{(11)}) \end{aligned} \tag{7}$$

With the fixed BERs of four states, the relaxed CDBERs of flash pages are only determined by the percentages of "01" and "00" states. Hence, while the numbers of these two states are counted, the best ΔV_{pp} and ECC schemes can be determined by looking up the new write speedup and ECC scheme tables with comparing the numbers of "01" and "00" states. There are three components implemented in the flash controller: State Mapping Scheme, Four State Counter and Speedup Engine, as shown in Figure 8. State mapping scheme is designed to re-map the cell states for CDBER reductions. Then, Four State Counters record the numbers of four states. Finally, the Speedup Engine is designed to do either read or write performance speedup. Based on the numbers of four states, the best ΔV_{pp} is selected and applied to improve write performance, or the best ECC scheme is selected for future read latency reduction.

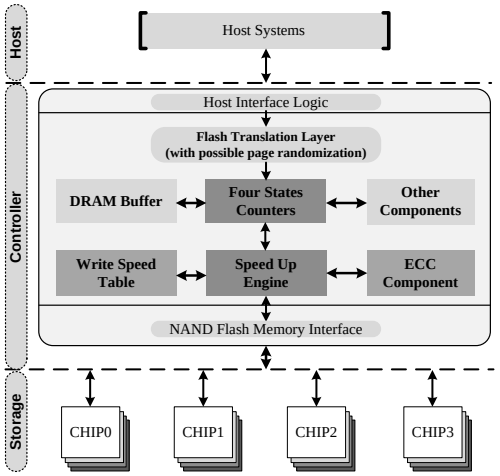


Fig. 8: The implementation of the proposed CDBER exploiting approach in the flash memory controller.

State Mapping Scheme The State Mapping Scheme is designed to smartly map the four states to achieve CDBER reduction. First, the numbers of cells in different states are constructed during the data transferring using four counters, one for each state. Then, the mapping type is determined. 5 bits are required to record the specific mapping type. During programming operation, the 5 bit mapping type recording is programmed to the OOB area of the flash page. Then, during read operation on the page, the data are read and translated back to the original data based on the mapping type.

States Counters: With the relaxed CDBER models, the best ΔV_{pp} and ECC scheme are selected through the numbers of "01" and "00" states. In addition, the state mapping scheme is also implemented based on the numbers of states. Hence, these state counters are in charge of counting the numbers of '01" and "00" states and then they can be used as the input of Speed Up Engine for performance improvement.

Speed Up Engine: Speed Up Engine is to improve either write or read performance. Lets first consider its function for *write speedup*. With the relaxed CDBER models, the possible range of numbers of "01" and "00" states is partitioned into a discrete set of levels and is stored in a table. This table is constructed based on the flash device model proposed in [28]. Note that other flash device models can also be applied since the asymmetric state error rates are the general characteristics of flash memory [6] [19]. Due to the difference between the relaxed CDBER models, the ΔV_{pp} selections for MSB pages only require the number of "01" states, and both numbers of "01" and "00" states are required for LSB pages. Lets then consider its function for *read speedup*. Similar to the write speedup, there is also a table used to record the correspondence between each ECC scheme and its supported ranges of numbers of "01" and "00" states. The supported range of a segment determined by the State Counters is used to select a matching ECC scheme.

D. Overhead Analysis

There are storage, hardware, latency and power overhead in the implementation of the proposed approaches.

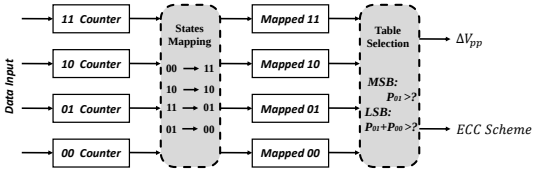


Fig. 9: The implementation of proposed approach within each component.

(1) In terms of storage overhead, two tables are maintained in the flash controller. The State Counters are added to count the numbers of states, which require 14 bits to record the number of each state. The Write Speed Table contains about six entries with each entry mainly comprising two 14-bit registers for storing the fixed values. The ECC Selection Table contains four entries with each entry mainly comprising two 14-bit registers for storing the fixed values. 5 bits are required for each flash page to record the mapping type, which is maintained in the OOB area of each flash page. In addition, a small buffer is used to temporally store the MSB or LSB pages. Overall, the storage overhead is negligible.

(2) In terms of hardware overhead, our approach requires multiple ECC schemes and multiple program voltages to support multiple ΔV_{pp}. Previous works suggest that these

additional components present little hardware overhead to the flash controller [27] [36]. Multiple BCH encoding schemes are required in the flash controller, which are *SHARED* by all the pages in an SSD. For the multiple ΔV_{pp}, using more than one ΔV_{pp} have been widely studied in previous work [28] [27] [19]. Similar to previous work, we also use this technique to improve the write speed. The circuity that provides multiple program voltages can be shared across the SSD drive, i.e., only one set for one drive. Overall, the hardware overhead is negligible.

(3) In terms of the latency overhead, the proposed approaches incur latency overhead in the following three additional processes, as shown in Figure 9. The first one is the process in counting the numbers of states for page *i*. In this case, only a few bytes are processed in pipeline so that the time cost at the hardware for counting the numbers of states can be hided and minimized [1]. The second one is the process of state mapping. This process can be realized by applying multiplexer, where these mapping types can be completed within 24 cycles. The last one is for selecting the best ΔV_{pp} and ECC schemes. This process can be realized by comparing the recorded numbers of states to the fixed values of each level. Hence, this process only need at most 5 or 3 compare operations, which are implemented in hardware. Due to the implementation in hardware, the time cost of these operations is at nanosecond level, which is negligible compared with the program and read latencies.

(4) In terms of the power overhead, the proposed approach is able to reduce the power consumption in the following three aspects. Firstly, with applying larger ΔV_{pp}, which can program cells to the fixed threshold values with less iterations, the number of issue times of peripheral circuits can be reduced. Secondly, with applying different ECC schemes, although the decoding and encoding time can be reduced, the whole page may be divided into several segments and decoded respectively. That is, the total power consumptions with applying different ECC schemes are similar. Thirdly, with applying state mapping scheme, more cells can be programmed into low-voltage states. Hence, the programmed voltages during runtime can be reduced.

V. Experiments and Analysis

In this section, we first present the experimental methodology. Then, the experimental results are presented, followed by the analysis of the improved performance.

A. Experiment Setup

The popular trace driven simulator Disksim 4.0 [2] and the widely used SSD model in [1] are extended to verify the proposed approaches. The simulated SSD contains 8 channels with 1 chip per channel. In each chip, there are 2 dies with 2 planes per die. In each plane, there are 2048 blocks with 64 4KB pages per block. In order to simulate an aged SSD, the number of program and erase cycles and the data retention time used in the error rate models are obtained from an aged SSD, where the number of program and erase cycles is set with 15K, and the data retention time is set with 1 year. With this setting, the calculated error rate of each state can be regarded as an aged SSD. Page based mapping is used as the default mapping scheme, and greedy garbage collection is applied. BCH(34496, 32768, 55) is used as the default ECC scheme.

Relaxed CDBER Models: The flash device model in [28] is extended to determine the error rates of different cell states. The rest settings are the same as the original work [28]. In order to present the data content of each flash page, four Gaussian distribution functions are constructed to simulate the distribution of the four states. And the parameters of Gaussian distribution functions are obtained from the above-mentioned files and applications.

Workloads: Several write intensive and read intensive workloads are selected from MSR traces [26], Financial traces [29], and postmark traces [1]. In this work, we carry out the simulation of the state distributions as follows: Several files from different applications, such as database files, emails, codes, operating system files, music, games, office and movies, are analyzed to identify the distribution of the cell states. Then, four Gaussian distribution functions are used to simulate the distribution of the four states based on the obtained distributions [27] [28]. The parameters for the Gaussian distribution functions are determined by the average percentages of the four states (μ) and the standard deviation (σ) of each state. The default BCH scheme used in this work is the widely used one [19].

ECC Selection Table: For the ECC selection scheme, four different BCH (n, k, t) schemes with different capabilities are implemented in the flash controller. Table III shows the four BCH schemes with their supported ranges of numbers of four states and decoding latencies. The decoding latencies are estimated by the hardware of [12] on 65nm CMOS cell and SRAM libraries. During program operations, BCH scheme is selected based on the numbers of "01" and "00" states and data are encoded using the selected BCH scheme. Then, during read operations, the data are read and decoded with the corresponding BCH scheme.

B. Experimental Results

In this subsection, we first evaluate write and read performance improvement by the speedup approaches without state mapping scheme. In the evaluation, five cases are evaluated. The first one is programmed with the default speed, which is referred as the traditional case. The second and third cases are on the improvement with only LSB pages or MSB pages speeded up. The third case is on the improvement with both LSB and MSB pages speeded up. The last one is programmed with the un-relaxed CDBER models, which are regarded as the optimal case for performance improvement. The second set of evaluation is for the state mapping scheme. The flash pages are speeded up based on the new state distributions obtained from the state mapping scheme.

1) Write Performance Improvement: Figure 10 shows the write performance comparison. All of the latencies are normalized to the traditional case. Several observations can be made: By speeding up either LSB pages or MSB pages, the improvements are 9.1% and 16.6%, respectively. The

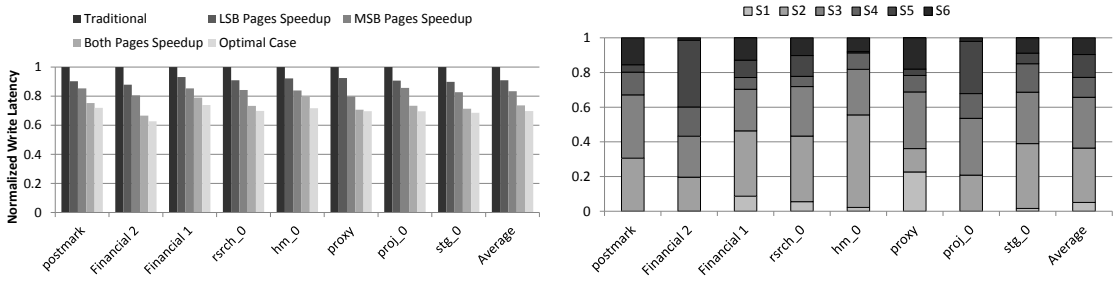
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2016.2626446, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE II: Correlation Between Number Ranges of "01" and "00" States and Corresponding ΔV_{pp}.

Levels	Program Step Sizes	Number Ranges for MSB Pages	Number Ranges for LSB Pages	Program Lat.(μs)
0	0.3	$num_{01} >= 15025$	$num_{01} + num_{00} >= 11682$	200
1	0.38	$num_{01} >= 7865$	$num_{01} + num_{00} >= 5817$	157
2	0.46	$num_{01} >= 4506$	$num_{01} + num_{00} >= 3048$	130
3	0.54	$num_{01} >= 2868$	$num_{01} + num_{00} >= 1704$	111
4	0.6	$num_{01} >= 1573$	$num_{01} + num_{00} >= 639$	100
5	0.68	$num_{01} >= 0$	$num_{01} + num_{00} >= 0$	88.2

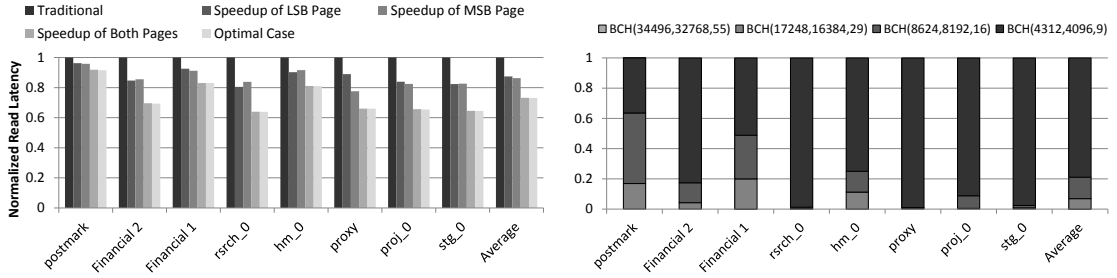
TABLE III: Correlation Between BCH Codes and Number Ranges of "01" and "00" States.

Levels	BCH (n, k, t)	Number Ranges for MSB Pages	Number Ranges for LSB Pages	Decoding Lat.(μs)
1	(34496,32768,55)	$num_{01} >= 16384$	$num_{01} + num_{00} >= 16221$	41.2
2	(17248,16384,29)	$num_{01} >= 11846$	$num_{01} + num_{00} >= 9143$	21.71
3	(8624,8192,16)	$num_{01} >= 4031$	$num_{01} + num_{00} >= 2704$	11.25
4	(4312,4096,9)	$num_{01} >= 0$	$num_{01} + num_{00} >= 0$	5.78



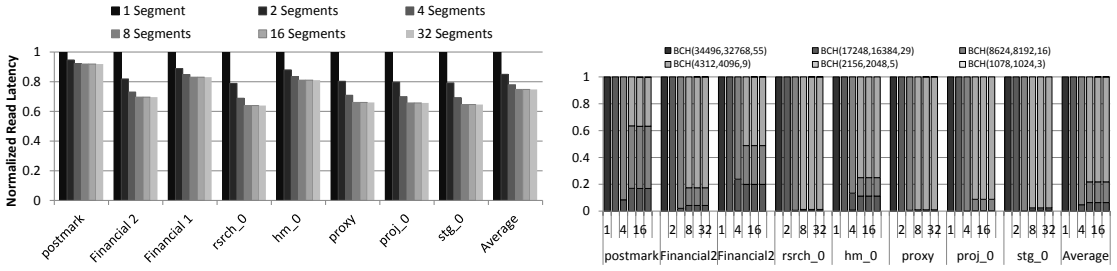
(a) Write Performance Improvement. (b) Request Distribution Among Different Levels.

Fig. 10: Write Performance Improvement Compared Against Traditional Case.



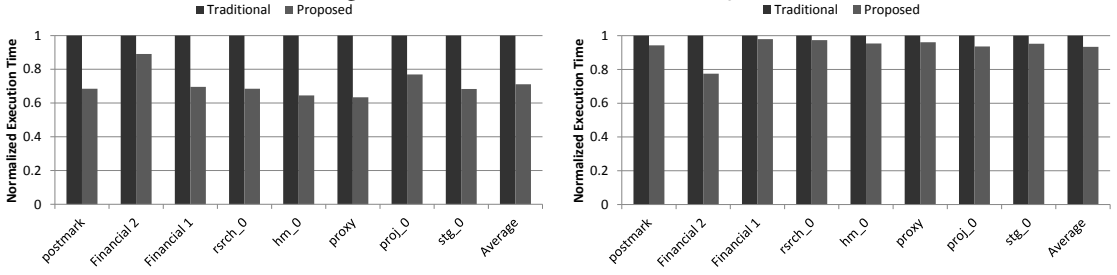
(a) Read Performance Improvement. (b) Request Distribution Among Different BCH Schemes.

Fig. 11: Read Performance Improvement Compared Against Traditional Case.



(a) Read Performance Improvement with Different BCH Schemes. (b) Request Distribution Among Different BCH Schemes.

Fig. 12: Evaluation of the number of segments.



(a) Execution Time Decrease With Write Speedup. (b) Execution Time Decrease With Smart ECC Selection.

Fig. 13: Execution Time Decrease With Write Speedup and Smart ECC Selection Compared Against Traditional Case.

performance difference between LSB pages and MSB pages stems from the different error rates between LSB pages and MSB pages as presented in Section IV-A. We can find that speeding up MSB pages has more benefits than LSB pages. By speeding up both pages, the write performance is significantly improved by 26.4%, on average.

In order to further understand the speedup reasons among the different workloads, the request distributions among different write speed levels are collected. Figure 10(b) shows the distributions of the requests among the 6 different write speed levels, $S1$ to $S6$, when both LSB and MSB pages are speeded up. $S1$ is the normal speed and gradually increased to $S6$ (The fastest one). As shown in Figure 10(b), most of write requests are speeded up. The reason comes from that majority of programmed data have much smaller error rates than the error correction capability of the default ECC scheme. Darker columns represent more write requests are speeded up by the faster speed levels, which induces better performance improvement. Furthermore, while there are 6 levels of write speeds available in Table II, 85.4% write requests are speeded up from levels $S2$ to $S5$, as shown in Figure 10(b).

In addition, the write performance improvement with applying relaxed CDBER models is close to the optimal case. Compared with the speeding up both pages, the optimal case only can reduce the write latency by 3.9%.

2) Read Performance Improvement: Read performance improvement is evaluated in a similar way. Four BCH schemes are selected as shown in Table III. Figure 11 shows the read performance improvement against the traditional case, which uses BCH(34496, 32768, 55) as the default ECC scheme for all pages. As shown in Figure 11, on average the read performance is improved by 26.8% if both LSB and MSB pages are partitioned into 8 segments and are protected using lighter-weight ECC codes. The improvement is 12.6% and 13.7% respectively if only one of the pages is speeded up.

In order to understand the different improvement among workloads, the request distributions among different BCH schemes are collected. Figure 11(b) shows the distributions of the requests among the 4 different BCH schemes when both LSB and MSB pages are speeded up. As shown in Figure 11(b), most of read requests are speeded up. The reason is similar to the write request improvement case. From the collected results, we can find that more requests are encoded by lighter-weight ECC schemes, more the read latency is reduced. Furthermore, as shown in Figure 11(b), BCH(8624, 8192, 16) and BCH(4312, 4096, 9) are the two mostly used lighter-weight ECC schemes for the simulated workloads.

Compared with the read performance improvement achieved by speeding both pages with un-relaxed CDBER models, the optimal case only can further reduce the read performance by 0.2%. This is because most read requests are encoded with the lighter-weight ECC schemes while the relaxed CDBER models are applied.

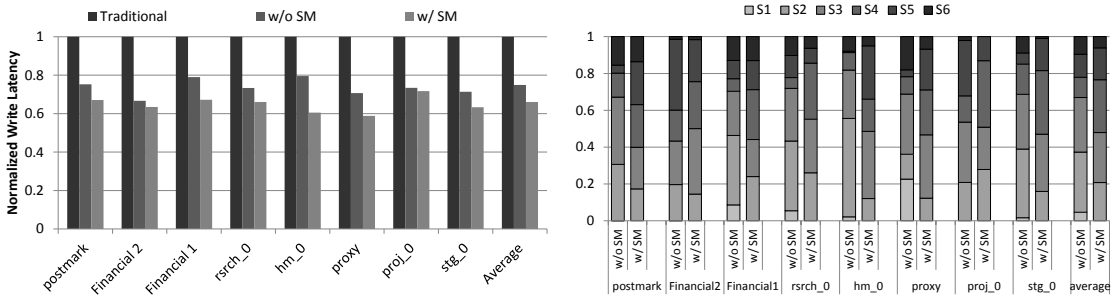
In addition to the default settings, the number of segments is evaluated in Figure 12. We have evaluated 1 to 32 segments to find the optimal number of segments, where 1 segment is the traditional case. Two additional lighter-weight BCH schemes are implemented to show the effectiveness: BCH(2156, 2048, 5) and BCH(1078, 1024, 3). As can be seen, the performance improvement increases significantly as the number of segments increases from 1 to 4, and peaks at 8 segments. As discussed in Section IV-B, the worst-case CDBER of segments increases rapidly with the number of segments, and eventually requires a heavier-weight ECC code. Figure 12(b) shows the distribution of the requests among the six BCH schemes with segments from 1 to 32. As can be seen, with increasing the number of segments, these two lightest-weight BCH schemes are not used. In addition, more heavier-weight BCH schemes, BCH(17248, 16384, 29), are applied as the number of segments increases. Based on this observation, the default number of segments is set to 8 in this work.

3) Execution Time Decrease: The execution time evaluation is presented in this section. As shown in Figure 15 and Figure 14, the write and read execution time at system level are presented. Compared with the traditional case, the proposed approach is able to reduce the write and read execution time by 28.9% and 6.6% on average. The achieved performance improvement at system level are contributed from the performance improvement at device level, where the time cost dominates the time cost of overall system [37]. On the other hand, the read execution time is only reduced by 6.6%. This is because the asymmetry of read and write requests. While the write latency at device level is greatly reduced, the overall system execution time also can be greatly decreased. On the contrary, there is little execution time decrease with little read latency decrease at device level.

4) State Mapping Scheme: State mapping scheme is evaluated in this section. The state mapping scheme is compared with the traditional case and the proposed approaches without state mapping scheme equipped. Figure 14(a) shows the write performance improvements, where *w/o SM* represents the the proposed approach without state mapping and *w/ SM* represents the state mapping equipped case. On average, the write performance is further improved by 7.9% compared with *w/o SM*. In order to understand the reason for the performance improvement, the request distributions among the 6 write speed levels are collected in Figure 14(b). As shown in this figure, we can find that more write requests are speeded up at the levels with faster write speeds. It means that CDBERs of most flash pages can be further reduced by the state mapping scheme. They can be utilized to further speed up the write requests.

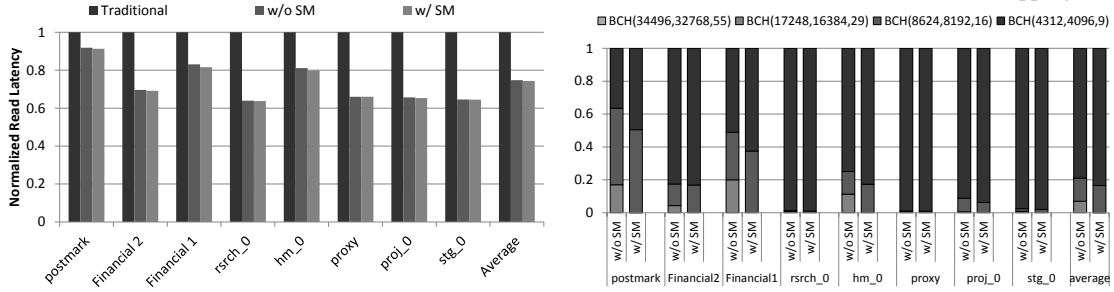
Figure 15(a) shows the read performance improvement compared with the traditional case and *w/o SM* case. Four BCH schemes are applied in the evaluation. Compared with the significant write performance improvement with state mapping scheme, the performance improvement achieved for read requests with state mapping scheme is insignificant. On average, the read performance is further improved by 0.4% compared with *w/o SM*. In order to understand the reason for the insignificant performance improvement, the request distributions among the 4 BCH schemes are collected in Figure 15(b). As shown in this figure, there are two observations: First, most requests have been encoded by the lightest-weight BCH schemes for *w/o SM*. Second, there are little improvement space for state mapping scheme. As show in this figure,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2016.2626446, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



(a) Write Performance Improvement With State Mapping. (b) Request Distribution Among Different Levels.

Fig. 14: Write Performance Improvement Among Traditional Case And The Cases With/Without State Mapping Scheme.



(a) Read Performance Improvement With State Mapping. (b) Request Distribution Among Different BCH Schemes.

Fig. 15: Read Performance Improvement Among Traditional Case And The Cases With/Without State Mapping Scheme.

the percentages of requests further encoded by the lightest-weight BCH scheme are small. In this case, even though state mapping scheme is able to further reduce the CDBERs of flash pages, the read performance is insignificantly improved.

VI. Conclusions

In this paper, we proposed comprehensive approaches to improve write or read performance of flash memory based storage systems. Different from previous works, the asymmetric error rates of cell states of flash memory are exploited. A Content-Dependent Bit Error Rate (CDBER) model is proposed to determine the error rates of a page according to the page's content. Simulation results show that the proposed approach works effectively, achieves significant performance improvement.

References

[1] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. S. Manasse, and R. Panigrahy. Design tradeoffs for ssd performance. In *ATC*, pages 57–70, 2008.

[2] J. S. Bucy, J. Schindler, S. W. Schlosser, and G. R. Ganger. The disksim simulation environment version 4.0 reference manual. *Parallel Data Laboratory*, page 26, 2008.

[3] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai. Error patterns in mlc nand flash memory: Measurement, characterization, and analysis. In *DATE*, pages 521–526, 2012.

[4] Y. Cai, Y. Luo, E. F. Haratsch, K. Mai, and O. Mutlu. Data retention in mlc nand flash memory: Characterization, optimization, and recovery. In *HPCA*, pages 551–563, 2015.

[5] Y. Cai, O. Mutlu, E. F. Haratsch, and K. Mai. Program interference in mlc nand flash memory: Characterization, modeling, and mitigation. In *ICCD*, pages 123–130, 2013.

[6] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. S. Unsal, and K. Mai. Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime. In *ICCD*, pages 94–101, 2012.

[7] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, O. Unsal, A. Cristal, and K. Mai. Neighbor-cell assisted error correction for mlc nand flash memories. In *SIGMETRICS*, pages 491–504, 2014.

[8] R. Chen, Y. Wang, and Z. Shao. Dheating: Dispersed heating repair for self-healing nand flash memory. In *CODES+ISSS*, page 7, 2013.

[9] P. Chi, C. Xu, X. Zhu, and Y. Xie. Building energy-efficient multi-level cell stt-mram based cache through dynamic data-resistance encoding. In *ISQED*, pages 639–644, 2014.

[10] G. Dong, S. Li, and T. Zhang. Using data postcompensation and predistortion to tolerate cell-to-cell interference in mlc nand flash memory. *TCAS-I*, pages 2718–2728, 2010.

[11] G. Dong, N. Xie, and T. Zhang. On the use of soft-decision error-correction codes in nand flash memory. *TCAS-I*, pages 429–439, 2011.

[12] G. Dong, N. Xie, and T. Zhang. Enabling nand flash memory use soft-decision error correction codes at minimal read latency overhead. *TCAS-I*, pages 2412–2421, 2013.

[13] C. Gao, L. Shi, K. Wu, C. J. Xue, and E. H. Sha. Exploit asymmetric error rates of cell states to improve the performance of flash memory storage systems. In *ICCD*, pages 202–207, 2014.

[14] L. M. Grupp, J. D. Davis, and S. Swanson. The bleak future of nand flash memory. In *FAST*, pages 2–2, 2012.

[15] J. Guo, Z. Chen, D. Wang, Z. Shao, and Y. Chen. Dpa: A data pattern aware error prevention technique for nand flash lifetime extension. In *ASP-DAC*, pages 592–597, 2014.

[16] K.-C. Ho, P.-C. Fang, H.-P. Li, C.-Y. M. Wang, and H.-C. Chang. A 45nm 6b/cell charge-trapping flash memory using ldpc-based ecc and drift-immune soft-sensing engine. In *ISSCC*, pages 222–223, 2013.

[17] J. Li, K. Zhao, J. Ma, and T. Zhang. Realizing unequal error correction for nand flash memory at minimal read latency overhead. *TCAS-II*, pages 354–358, 2014.

[18] R.-S. Liu, M.-Y. Chuang, C.-L. Yang, C.-H. Li, K.-C. Ho, and H.-P. Li. Improving read performance of nand flash ssds by exploitingerror locality. *TC*, pages 1–1, 2015.

[19] R.-S. Liu, C.-L. Yang, and W. Wu. Optimizing nand flash-based ssds via retention relaxation. In *FAST*, pages 11–11, 2012.

[20] S. E. C. Ltd. High-Performance 128-gigabit 3-bit Multi-level-cell NAND Flash Memory. http://www.samsung.com/global/business/semiconductor/news-events/press-releases/detail?newsId=12761, 2013.

[21] J. Meza, Q. Wu, S. Kumar, and O. Mutlu. A large-scale study of flash memory failures in the field. pages 177–190, 2015.

[22] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. R. Nevill. Bit error rate in nand flash memories. In *IRPS*, pages 9–19, 2008.

[23] A. Mirhoseini, M. Potkonjak, and F. Koushanfar. Coding-based energy minimization for phase change memory. In *DAC*, pages 68–76, 2012.

[24] A. Mirhoseini, M. Potkonjak, and F. Koushanfar. Phase change memory write cost minimization by data encoding. *JESTCS*, pages 51–63, 2015.

[25] V. Mohan, S. Sankar, S. Gurumurthi, and W. Redmond. refresh ssds: Enabling high endurance, low cost flash in datacenters. *Univ. of Virginia, Tech. Rep. CS-2012-05*, 2012.
[26] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating server storage to ssds: analysis of tradeoffs. In *EuroSys*, pages 145–158, 2009.
[27] Y. Pan, G. Dong, Q. Wu, and T. Zhang. Quasi-nonvolatile ssd: Trading flash memory nonvolatility to improve storage system performance for enterprise applications. In *HPCA*, pages 1–10, 2012.
[28] Y. Pan, G. Dong, and T. Zhang. Exploiting memory device wear-out dynamics to improve nand flash memory system performance. In *FAST*, pages 18–18, 2011.
[29] U. T. Repository. OLTP Application I/O. http://traces.cs.umass.edu/, 2007.
[30] L. Shi, K. Qiu, M. Zhao, and C. J. Xue. Error model guided joint performance and endurance optimization for flash memory. *TCAD*, pages 343–355, 2014.
[31] L. Shi, K. Wu, M. Zhao, C. J. Xue, and E. H. Sha. Retention trimming for wear reduction of flash memory storage systems. In *DAC*, pages 1–6, 2014.
[32] K.-D. Suh, B.-H. Suh, Y.-H. Lim, J.-K. Kim, Y.-J. Choi, Y.-N. Koh, S.-S. Lee, S.-C. Kwon, B.-S. Choi, J.-S. Yum, et al. A 3.3 v 32 mb nand flash memory with incremental step pulse programming scheme. *JSSC*, pages 1149–1156, 1995.
[33] J. Wang, X. Dong, G. Sun, D. Niu, and Y. Xie. Energy-efficient multi-level cell phase-change memory system with data encoding. In *ICCD*, pages 175–182, 2011.
[34] W. Wang, T. Xie, and D. Zhou. Understanding the impact of threshold voltage on mlc flash memory performance and reliability. In *ICS*, pages 201–210, 2014.
[35] G. Wu, X. He, N. Xie, and T. Zhang. Diffecc: improving ssd read performance using differentiated error correction coding schemes. In *MASCOTS*, pages 57–66, 2010.
[36] G. Wu, X. He, N. Xie, and T. Zhang. Exploiting workload dynamics to improve ssd read latency via differentiated error correction codes. *TODAES*, page 55, 2013.
[37] Q. Xu, H. Siyamwala, M. Ghosh, T. Suri, M. Awasthi, Z. Guz, A. Shayesteh, and V. Balakrishnan. Performance analysis of nvme ssds and their implication on real world databases. In *SYSTOR*, page 6, 2015.
[38] E. Yaakobi, L. Grupp, P. H. Siegel, S. Swanson, and J. K. Wolf. Characterization and error-correcting codes for tlc flash memories. In *ICNC*, pages 486–491, 2012.
[39] E. Yaakobi, J. Ma, L. Grupp, P. H. Siegel, S. Swanson, and J. K. Wolf. Error characterization and coding schemes for flash memories. In *GLOBECOM Workshops*, pages 1856–1860, 2010.
[40] K. Zhao, W. Zhao, H. Sun, X. Zhang, N. Zheng, and T. Zhang. Ldpc-in-ssd: Making advanced error correction codes work effectively in solid state drives. In *FAST*, 2013.
[41] M. Zhao, Y. Xue, C. Yang, and C. J. Xue. Minimizing mlc pcm write energy for free through profiling-based state remapping. In *ASP-DAC*, pages 502–507, 2015.

Edwin H.-M. Sha (S'88-M'92-SM'04) received the Ph.D. degree from the Department of Computer Science, Princeton University, Princeton, NJ, in 1992. From August 1992 to August 2000, he was with the Department of Computer Science and Engineering at the University of Notre Dame, Notre Dame, IN. Since 2000, he has been a Tenured Full Professor at the Department of Computer Science at the University of Texas at Dallas, Richardson, TX. Since 2012, he has been serving as the Dean of the College of Computer Science at Chongqing University, Chongqing, China.

Congming Gao received BS degree in Computer Science and Technology from Chongqing University in China in 2013. He is now a Ph.D candidate in the College of Computer Science, Chongqing University. His research interests include embedded and real-time systems, non-volatile memory and architecture optimizations.

Liang Shi received the B.S. degrees in Computer Science from Xi'an University of Post & Telecommunication, Xi'an, Shanxi, China, in July, 2008, Ph.D. degree from University of Science and Technology of China, Hefei, China, in June, 2013. He is now a full-time teacher in the College of Computer Science at the Chongqing University. His research interests include flash memory, embedded systems, and emerging non-volatile memory technology.

Kaijie Wu received the BE degree from Xidian University, Xi'an, China, in 1996, the MS degree from the University of Science and Technology of China, Hefei, China, in 1999, and the Ph.D. degree in electrical engineering from Polytechnic University (Now Polytechnic Institute of New York University), Brooklyn , New York, in 2004. He then joined University of Illinois, Chicago, USA as an Assistant Professor. Since 2013, he becomes a professor at the College of Computer Science, Chongqing University, China. His research is on the big area of trustworthy computing with special interest on dependable computing and hardware security.

Mengying Zhao received B.E. degree in School of Computer Science and Technology from Shandong University, China in July 2011, and Ph.D. degree in Department of Computer Science, City University of Hong Kong in July 2015. She is now an Assistant Professor in School of Computer Science and Technology in Shandong University. Her research interests include computer architecture, embedded and real-time systems, and non-volatile memory.

Chun Jason Xue received BS degree in Computer Science and Engineering from University of Texas at Arlington in May 1997, and MS and PhD degree in computer Science from University of Texas at Dallas, in Dec 2002 and May 2007, respectively. He is now an Associate Professor in the Department of Computer Science at the City University of Hong Kong. His research interests include memory and parallelism optimization for embedded systems, software/hardware co-design, real time systems and computer security.