Exploit Asymmetric Error Rates of Cell States to Improve the Performance of Flash Memory Storage Systems

Congming Gao*, Liang Shi*§, Kaijie Wu*, Chun Jason Xue†, and Edwin H.-M. Sha*
*College of Computer Science, Chongqing University, Chongqing, China
† Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong
Email: {albertgaocm, shi.liang.hk, kaijie}@gmail.com, jasonxue@cityu.edu.hk, edwinsha@gmail.com

***Abstract*—The reliability of flash memory is getting worse with the introduction of Multiple Level Cell (MLC) and Triple Level Cell (TLC) technologies. To account for possible errors, each page in a flash memory is equipped with an Error Correction Code (ECC) scheme. The ECC scheme is chosen according to the worst-case error occurrences across all pages in the flash memory. Recent studies show that an MLC flash cell in different states exhibits diverse error rates and the difference is dramatic. Consequently, pages with different data will exhibit quite different error rates. Existing technologies that use one uniform ECC scheme for all pages in a flash memory is far from optimal. This paper exploits the asymmetric error rates exhibited by the pages with different data for write performance improvement. Before a page is programmed, its specific error rate, called Content-Dependent Bit Error Rate (CDBER), is estimated according to the content of the page. The margin between the CDBER of a page and the maximal error rate correctable by the uniform ECC code is exploited for write performance improvement. Simulation results show that the proposed approach leads to significant write performance improvement.**

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 market [1] and the number of bits per cell is increased from 1 bit to the most recent 6 bits [6]. However, as the bit density increases with technology scaling, the reliability of flash cells is shrinking, which induces significant performance degradation. This is because the more states represented by a cell, the smaller noise margin exists between adjacent states of the cell. 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 [16], which increases the programming latency [12] [8] [11]. In this work, we propose techniques to improve the write performance of MLC flash memory.

Flash memory uses floating gates to store charges. The voltages of the floating gates can be charged into multiple states. Different states represent different data. Previous works have shown that the error rate of a flash memory cell is highly dependent on the states - a cell in different states exhibits quite different error rates [5] [3] [4]. For example, Cai *et al.* [3] examined the error characteristics at 30-40nm technology nodes and found that the error sources have different impacts on the error rates of a flash memory cell in different states. The difference could be as large as 23 times. Wang *et al.* [17] also observed that a flash memory cell programmed to a state represented by a lower voltage has smaller Bit Error Rate (BER). Yaakobi *et al.* [19] found in their experiments that the erroneous voltage shifts at different states are different, and the difference will be enlarged if more bits (hence more states) are packed into a single cell [18]. 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 rate.

In this work, we propose techniques to improve flash write performance by exploiting the asymmetric error rates among the multiple states of MLC flash memory cells. Before a page is programmed, its specific error rate is estimated according to its data content. The error rate is referred to as Content-Dependent Bit Error Rate (CDBER). In this paper, the margin between a page's CDBER and the maximal error rates correctable by the uniform ECC code can be exploited to improve the write performance. Write performance can be improved by increasing the step size (ΔV_{pp}) used by the ISPP programming technique accordingly. This is based on the fact that a larger ΔV_{pp} corresponds to a faster write speed but introduces a higher error rate. Hence the write speed of a page can be increased as long as the increased error rate does not exceed the margin. In summary, this work makes the following contributions:

- Proposed a CDBER model with the understanding of the asymmetric state error rates of flash memory cells;
- Proposed a write speed improvement approach with CDBER exploiting;
- Presented an efficient implementation of the proposed

§ This work is supported by the Fundamental Research Funds for the Central Universities (Project No.106112014CDJZR185502). Liang Shi is corresponding author: shi.liang.hk@gmail.com.

approaches with negligible overhead. Extensive simulations show that the proposed approach achieves significant write performance improvement.

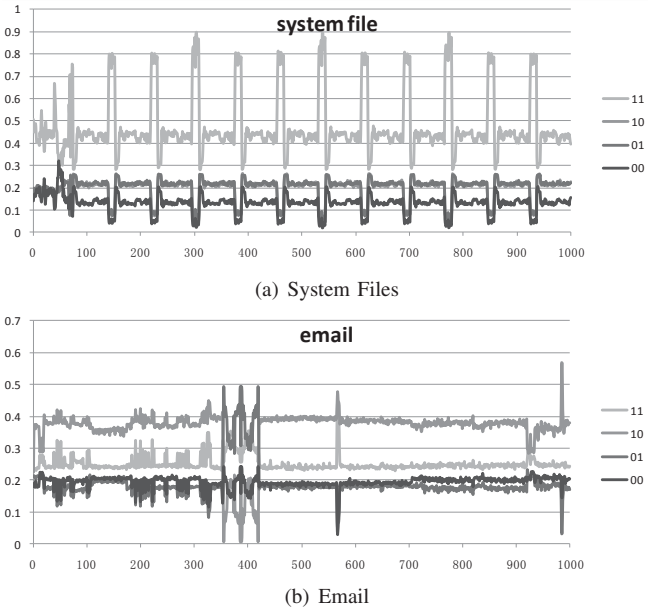
The rest of the paper is organized as follows. Section II presents the motivation of the proposed work. Section III discusses the proposed CDBER exploiting approaches. Experiments and result analysis are presented in Section IV. Finally, Section V concludes the paper.

II. Motivation

The integrity of the data stored in flash memory is affected by many factors, such as program and erase cycling (P/E cycling), leakage current, cell-to-cell interference, and temperature [4] [3]. Take MLC flash memory as an example. Each MLC flash memory cell hosts two bits information. Hence it has a total of four states: one Erase state "11", and three Program states "10", "01", and "00". The states are represented by depositing certain amount of charges in the floating gate of the cell. However, due to the factors mentioned above, the deposited charges may change unintentionally, which causes erroneous state shifts. For example, the deposited charges could be reduced due to leakage current, or increased due to repeatedly program-erase cycles and/or cell-to-cell interference [12] [11] [8]. We thereafter use Right Shifts to represent the increases in the cell voltage, and use Left Shifts to represent the reductions in the cell voltage. To ensure reliable storage, many approaches have been proposed, including ECC schemes [6], program speed reduction [7], and endurance relaxation [12] [4]. These works assume that the reliability characteristics of the four states of a flash memory cell are the same.

However, several recent works show that different states of a flash memory cell exhibit diverse reliability characteristics, and the difference is dramatic. For example, Cai *et al.* [3] found that the states "01" and "00" are much less reliable than the other two. Wang *et al.* [17] observed that a cell programmed to a lower threshold voltage induces less bit errors. Yaakobi *et al.* [19] found that the erroneous state shifts occurred at different states are not equal. Based on these observations, *the BER of a flash memory page should be determined by 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 Rate (CDBER). As shown in [3], a page with all its cells programmed to "01" or "00" states exhibits the worst-case error rates. Current approaches use the worst-case error rates as the requirement for their error-correcting capability, and apply to all pages uniformly. Such uniform error-correcting approaches, however, are overkill for the pages that are not filled with all "01" or "00" states. In fact, these pages are quite common according to our preliminary study given below.

In our preliminary study, we analyzed several kinds of files in order to examine the distributions of the cell states.



(a) System Files

(b) Email

Figure 1. Different state distributions of two typical application files.

A four-state MLC flash memory cell is considered. Similar results can be expected for the flash memory with more states. The files studied are partitioned into pages of 4KB. The distribution of the four states in each page is collected. In order to simulate the randomizer in the flash controller, we therefore obtain the pages from file system, and randomize them using a pseudo-noise (PN) random sequence generator, a cyclic redundancy check (CRC) generator, and the like, to simulate the internal randomization done at flash controller. Figure 1 shows the distribution of the four states for two typical application files. The x-axis represents the page number and the y-axis represents the percentages of the four states in each page. As shown in Figure 1, the distribution of the four states varies significantly from page to page and file to file. For example, system files have much more "11" compared with the other states. For emails, "10" is the dominant states. From the collected data, we find that the distribution of the four states varies significantly among pages. As a result, the CDBERs of different pages will vary significantly, and most of them are much smaller than the worst-case error rates. The difference between the worst case and CDBERs is exploited for performance improvement in this work.

III. Improving Performance By Exploiting Asymmetric Error Rates of Cell States

In this section, the asymmetric error rates of cell states are exploited for write performance improvement. A Content-Dependent Bit Error Rate (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 difference between the CDBER of a specific page and the uniform error-correcting capability provided by current flash chips

is then exploited for write performance improvement, while satisfying the original reliability requirements. Finally, an efficient implementation of the proposed approach 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. Based on the voltage distributions of the flash memory cell, we first discuss the characteristics of flash errors. Then, the CDBER model is presented.

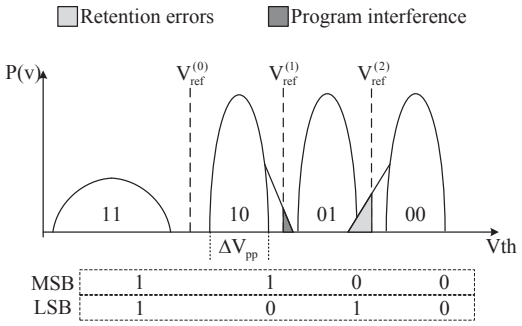


Figure 2. The voltage distribution for MLC flash memory with voltage fluctuations and shifts due to different flash error sources.

1) Flash Memory Error Characteristics: The error of a flash cell occurs when its stored charge changes unexpectedly that causes erroneous state transitions [3] [9]. Due to various factors, the stored charge could be increased or decreased, which causes a right or left shift from its states, respectively. Generally speaking, the left shift is dominated by charge leakage, and the right shift is dominated by the interferences from programming this and neighborhood cells. We use two vectors to represent the BERs caused by Left Shift and Right Shift, respectively, where $BER_L^{(b_1 b_2)}$ ($BER_R^{(b_1 b_2)}$) represents the BERs caused by Left (or Right) Shift at the state $b_1 b_2$.

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

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

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;

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

2) Content-Dependent Bit Error Rate Model: CDBER is highly correlated with the data stored in the flash page. As shown in Figure 1, the distributions of the four states in a page vary significantly among pages and applications. We use vector $V(P_i)$ to represent the distribution of the four states in a flash page i, where $V(P_i)$ =$(P_{11}, P_{10}, P_{01}, P_{00})$, and $P_{b_1 b_2}$ ($b_1 \in \{0, 1\}$, $b_2 \in \{0, 1\}$) represents the percentage of the state $b_1 b_2$ in the page. The CDBER of a page can then be computed by synthesizing the distributions of the four states and the error rate of each state.

An issue in the computation of the CDBER needs to be taken into consideration: Each word line in flash memory stores the information of two data pages that are referred to as MSB and LSB pages, respectively. Each cell in the word line thus contains one bit information (b_1) from the MSB page and and one bit (b_2) from the LSB page. b_1 and b_2 are referred to as the MSB and LSB of this cell, 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, error occurs 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 MSB page is computed as follows:

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

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

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

The difference between the BER correctable by a popular ECC scheme and the CDBERs of the application files presented in Figure 1 is shown 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 [4] [8] [11]. The second and third highest lines are the worst-case CDBER of the LSB and MSB pages, respectively.

The rest lines are the CDBERs of the LSB and MSB pages of those application files. We can make the following observations:

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

Based on these observations, write performance improvement approach is proposed in the following.

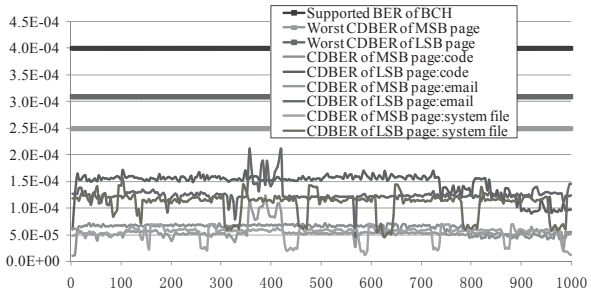


Figure 3. CDBER of MSB and LSB pages against the default BCH capability for different files.

B. Exploiting CDBER for Write Performance Improvement

The large redundancy shown in the Observation 1 leads us to improve the write performance of flash memory. Write performance of flash memory can be improved by increasing the step size of program operation (ΔV_{pp}). 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. We notice that there are a few other works on this area, such as Pan *et al.* [12] [11], Liu *et al.* [8], and Shi *et al.* [15]. However, they are based on different observations: the BER increases with P/E cycles and retention time. They hence suggest to use a larger ΔV_{pp} when P/E cycles are small or retention time is relaxed.

In our approach, the ΔV_{pp} is determined based on the identified CDBER of the page to be programmed using the model proposed in [12]. 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}.

C. Implementation

This section gives the implementation details of the proposed approaches. There are two components implemented in the flash controller: CDBER Calculator and Speedup Engine, as shown in Figure 4. CDBER Calculator calculates the CDBER of each page. The Speedup Engine can be set to do write performance speedup. Based on the CDBER of page, it selects the best ΔV_{pp} to improve write performance. Note that these components are implemented under the FTL, where the proposed approach is transparent to the possible randomizer in the flash controller and the MSB and LSB pages are determined, as shown in Figure 4.

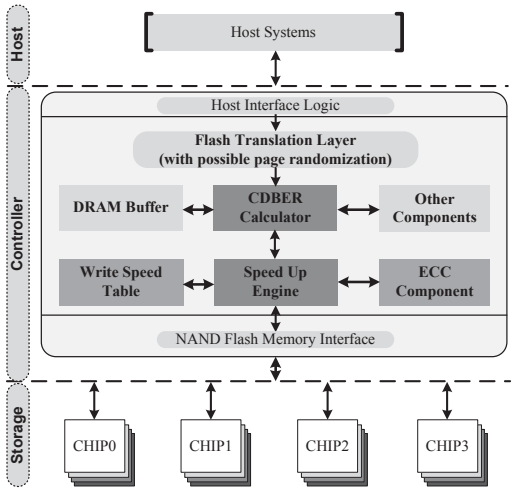


Figure 4. The implementation of the proposed CDBER exploiting approach in the flash memory controller.

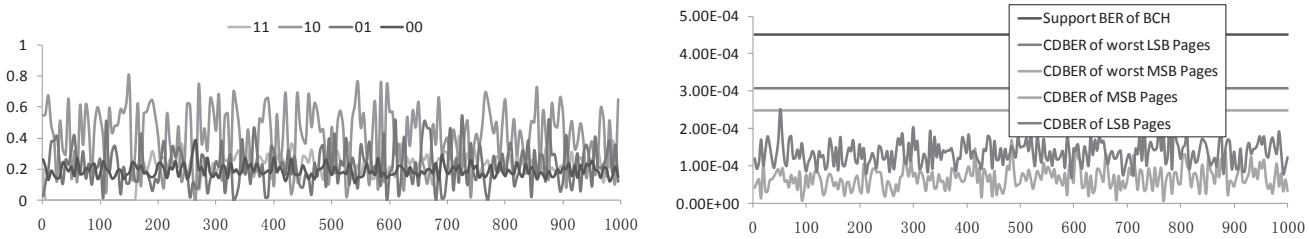
CDBER Calculator: CDBER Calculator computes the CDBER of the page to be programmed. It first counts the numbers of cells in different states using four counters, one for each state. Then, it calculates the CDBER of the page using the models developed in Section III-A.

Speed Up Engine: The Speed Up Engine is to improve write performance. The possible range of CDBERs, bounded by CDBER_{max} and CDBER_{min}, is partitioned into a discrete set of levels and is stored in a table as shown in Table I. This table is constructed based on the extended flash device model proposed in [12]. Each level of CDBER corresponds to a ΔV_{pp} such that it is the largest program step without letting the post-program error rate go beyond the deployed ECC's capability. The calculated CDBER is used as the input to index the table and the entry is the corresponding ΔV_{pp}.

Table I
MULTIPLE LEVELS OF CDBER RANGES WITH CORRESPONDING ΔV_{pp}.

Levels	Program Step Sizes	CDBER Ranges
0	$\Delta V_{pp}(0)$	CDBER_{max} – CDBER_1
1	$\Delta V_{pp}(1)$	CDBER_1 – CDBER_2
...	...	...
N-1	$\Delta V_{pp}(N-1)$	CDBER_{N-1} – CDBER_{min}

Overhead Analysis: There are storage, hardware, and power overhead in the implementation of the approach. (1) In terms of storage overhead, the Write Speed Table is maintained in the flash controller, which contains about a dozen of entries with each entry comprising three floating point



(a) Distribution of the four states in flash pages.

(b) CDBER of MSB and LSB pages.

Figure 5. Data Content Simulation with distribution of cell states and CDBER of pages.

numbers. A small buffer is used to temporally store the MSB and LSB pages. Overall, the storage overhead is negligible. (2) In terms of hardware overhead, our approach requires multiple program voltages to support multiple ΔV_{pp}. There are six different ΔV_{pp} required in the flash controller. Using more than one ΔV_{pp} have been widely studied in previous work [12] [11] [8]. The only logic change needed is to provide multiple program voltages with one program voltage corresponding to one ΔV_{pp}. The circuity that provides multiple program voltages can be shared across the SSD drive. (3) In terms of the power overhead, a higher ΔV_{pp} indeed leads to higher power consumption. Based on the previous studies, the increased power consumption is manageable but the approach of increasing ΔV_{pp} is beneficial. The major reason is that the end-to-end voltage span between the Erase state and a Program State remains unchanged, and a higher ΔV_{pp} just means fewer steps to go from one end to the other.

IV. 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 [14] and the SSD model in [2] 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. 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.

CDBER Calculation: The flash device model in [12] is extended to determine the error rates of different cell states. The rest settings are the same as the original work [12]. Four Gaussian distribution functions are constructed to simulate the distribution of the four states ($V(P_i)$). CDBER is then determined using the CDBER model (CDBER_{MSB} and CDBER_{LSB}). The four Gaussian distribution functions are shown in the **workload** simulation, presented as follows.

Write Speedup Table: For the write speedup approach, the relationship between CDBER and ΔV_{pp} is first constructed. By default, the number of CDBER levels is set to 6. Table II shows the write speedup table used in the experiment.

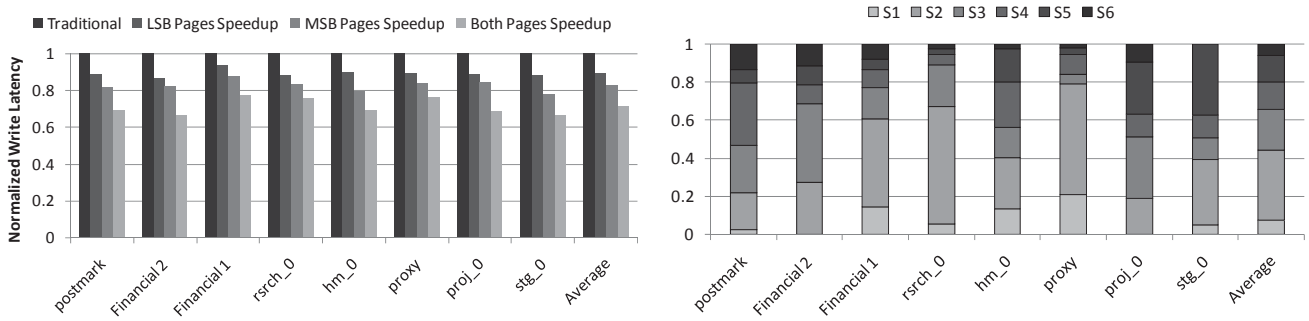
Table II
Correlation Between ΔV_{pp} and Supported CDBERs.

Levels	ΔV_{pp}	CDBERs Ranges	Program Lat.(μs)
1	0.3	$[4.0 \times 10^{-4}, 2.3 \times 10^{-4})$	200
2	0.38	$[2.3 \times 10^{-4}, 1.3 \times 10^{-4})$	157
3	0.46	$[1.3 \times 10^{-4}, 8.3 \times 10^{-5})$	130
4	0.54	$[8.3 \times 10^{-5}, 6.0 \times 10^{-5})$	111
5	0.6	$[6.0 \times 10^{-5}, 4.2 \times 10^{-5})$	100
6	0.68	$[4.2 \times 10^{-5}, 0)$	88.2

Workloads: Several write intensive and read intensive workloads are selected from MSR traces [10], Financial traces [13], postmark, and iozone traces [2]. However, there are no data content in these traces. 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. 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. Figure 5 presents one of the simulated workloads with the distribution of the four states and their corresponding CDBER for MSB and LSB pages. The distributions of the four states vary from page to page. The BCH scheme used in this work is the widely used one [8]. It can correct BER up to 4.09×10^{-4}, which is much higher than the worst-case CDBER of LSB and MSB pages. We believe that the simulation approach is able to simulate the real case of applications.

B. Experimental Results

In this subsection, we evaluate write performance improvement approach. The speed is normalized to the traditional approach (where pages are programmed using the default speed). The first two evaluations are on the improvement when only LSB pages or MSB pages are speedup based on their CDBER and the other one is programmed using the



(a) Write Performance Improvement.

(b) Request Distribution Among Different Write Speed Levels.

Figure 6. Write Performance Improvement Compared Against Traditional Case.

default speed. The next evaluation is on the improvement when both LSB and MSB pages are speedup based on their corresponding CDBERs.

Figure 6 shows the write performance comparison. Several observations can be made: By speeding up either LSB pages or MSB pages, the improvements are 11% and 17.4%, respectively. The performance difference between LSB pages and MSB pages stems from the different error rates between LSB pages and MSB pages as presented in Section III-A. By speeding up both pages, the write performance is significantly improved by 28.8%, on average. Further, while there are 6 levels of write speeds ($S1$ to $S6$) available in Table II, 86.2% write requests are speeded up from levels $S2$ to $S5$, as shown in Figure 6(b).

V. Conclusions

In this paper, we proposed comprehensive approaches to improve write 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. Then, based on the proposed CDBER model, the write performance improvement approach increases the program step size of write operations to exploit the redundancy between the error correction capability of the deployed ECC scheme and the CDBER of the page to be programmed. An efficient implementation with negligible cost is then presented. Simulation results show that the proposed approach works effectively, with around 37.2% write performance improvement.

References

[1] Samsung Electronics Co. Ltd. Samsung Mass Producing High-Performance 128-gigabit 3-bit Multi-level-cell NAND Flash Memory. https://memorylink.samsung.com/.../detail.do?newsId=12761, 2013.

[2] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy. Design tradeoffs for SSD performance. In *ATC 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, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. S. Ünsal, and K. Mai. Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime. In *ICCD*, pages 94–101, 2012.

[5] 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.

[6] 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.

[7] J. Jeong, S. S. Hahn, S. Lee, and J. Kim. Improving nand endurance by dynamic program and erase scaling. In *Proceedings of the 5th USENIX Conference on Hot Topics in Storage and File Systems*, HotStorage'13, pages 4–4, 2013.

[8] R.-S. Liu, C.-L. Yang, and W. Wu. Optimizing NAND flash-based SSDs via retention relaxation. In *FAST 2012*.

[9] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. Nevill. Bit error rate in nand flash memories. In *IEEE International Reliability Physics Symposium, 2008.*, pages 9–19, April 2008.

[10] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating server storage to SSDs: analysis of tradeoffs. In *EuroSys 2009*.

[11] 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 2012*.

[12] Y. Pan, G. Dong, and T. Zhang. Exploiting memory device wear-out dynamics to improve NAND flash memory system performance. In *FAST 2011*.

[13] U. T. Repository. OLTP Application I/O. http://traces.cs.umass.edu/, 2007.

[14] J. S. Bucy, J. Schindler, S. W. Schlosser, and G. R. Ganger. The disksim simulation environment version 4.0 reference manual. In *Parallel Data Laboratory, Carnegie Mellon University*, 2008.

[15] L. Shi, K. Qiu, M. Zhao, and C. J. Xue. Error model guided joint performance and endurance optimization for flash memory. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 33(3):343–355, 2014.

[16] K.-D. Suh, B.-H. Suh, and et al. A 3.3 v 32 mb nand flash memory with incremental step pulse programming scheme. *IEEE Journal of Solid-State Circuits*, 30(11):1149–1156, Nov 1995.

[17] W. Wang, T. Xie, and D. Zhou. Understanding the impact of threshold voltage on mlc flash memory performance and reliability. *28th International Conference on Supercomputing*, 2014.

[18] E. Yaakobi, L. Grupp, P. Siegel, S. Swanson, and J. Wolf. Characterization and error-correcting codes for tlc flash memories. In *2012 International Conference on Computing, Networking and Communications (ICNC)*, pages 486–491, Jan 2012.

[19] E. Yaakobi, J. Ma, L. Grupp, P. Siegel, S. Swanson, and J. Wolf. Error characterization and coding schemes for flash memories. In *2010 IEEE GLOBECOM Workshops (GC Wkshps)*, pages 1856–1860, Dec 2010.