One Gray Code Fits All: Optimizing Access Time with Bi-Directional Programming for QLC SSDs

Shaoqi Li*, Tianyu Wang*, Yongbiao Zhu*, Chenlin Ma*, Yi Wang*, Zhaoyan Shen†, and Zili Shao‡
*College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
†School of Computer Science and Technology, Shandong University, Qingdao, China
‡Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China
lishaoqi2023@email.szu.edu.cn, tywang@szu.edu.cn, zhuyongbiao2023@email.szu.edu.cn,
chenlin.ma@szu.edu.cn, yiwang@szu.edu.cn, shenzhaoyan@sdu.edu.cn, shao@cse.cuhk.edu.hk

***Abstract*—Gray code, a voltage-level-to-data-bit translation scheme, is widely used in QLC SSDs. However, it causes the four data bits in QLC to exhibit significantly different read and write performance with up to 8× latency variation, severely impacting the worst-case performance of QLC SSDs. This paper presents BDP, a novel Bi-Directional Programming scheme. Based on a fixed Gray code, BDP combines both the normal (forward) and reverse programming directions to enable runtime programming direction arbitration. Experimental results show that BDP can effectively improve the read and write performance of SSD compared to representative schemes.**

***Index Terms*—QLC SSD, Gray Code, Two-Step Programming**

I. Introduction

In high-density SSDs, the increased bit density within each flash cell offers an outstanding capacity-to-cost ratio but is accompanied by a rising bit error rate due to narrower voltage windows [1]–[3]. While Gray code mitigates this issue, it introduces significant performance disparities, with the four bits exhibiting up to 8× variation in read and write latency. As shown in Figure 1(a), existing approaches (e.g., FTSP [4]) typically use a unidirectional programming method that always programs fast bits first. However, under cold-then-hot workloads (Figure 1(c)), FTSP suffers from up to 31.5× higher access latency compared to our proposed BDP approach.

This performance degradation arises from the misplacement of hot data on slow pages. The state-of-the-art approach [5] attempts to alleviate this issue by employing multiple Gray codes to improve worst-case performance. However, as illustrated in Figure 1(d), MGC fails to fundamentally resolve the misplacement problem due to its reliance on a unidirectional programming sequence, resulting in suboptimal performance.

To address these limitations, we propose BDP, a novel Bi-Directional Programming scheme for QLC SSDs that introduces runtime programming order arbitration. BDP leverages both forward (fast bits first) and reverse (slow bits first) programming directions, ensuring that hot and cold data are allocated to appropriate pages (fast or slow) as they arrive (Figure 1(b)). Unlike MGC, BDP employs a single Gray code, eliminating the additional circuit overhead associated with multiple Gray codes. BDP incorporates hotness-aware data allocation and background migration schemes to improve performance. Experimental results demonstrate that BDP significantly enhances SSD read and write performance.

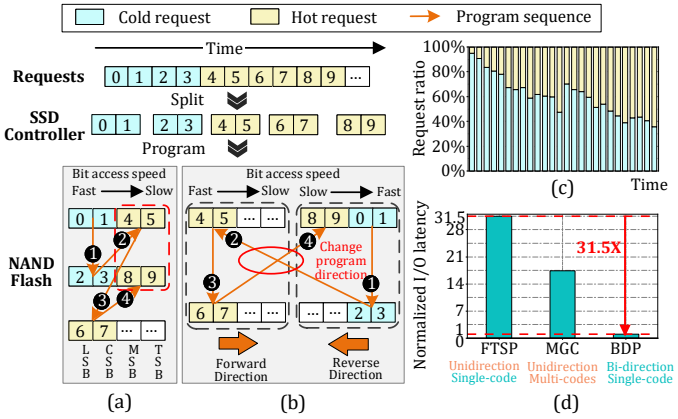

Fig. 1. Unidirectional programming method will cause performance degradation issues. (a) Cold data is stored on fast pages and hot data is stored on slow pages due to the programming direction constraints, causing long access latency. (b) BDP places both hot and cold data on proper pages. (c) Workload hm_0 follows a cold-then-hot pattern. (d) Performance comparison for different programming schemes.

II. BDP: A Bi-Directional Programming Scheme

The system architecture of BDP is illustrated in Figure 2. BDP introduces runtime programming order arbitration, enabling the use of both forward and reverse programming algorithms. To minimize circuit overhead, BDP employs a single Gray code specifically designed for QLC SSDs. It supports two programming directions: forward programming (fast bits first, followed by slow bits) and reverse programming (slow bits first, followed by fast bits). BDP incorporates a hotness-aware data allocation scheme to ensure that hot and cold data are dynamically assigned to the appropriate pages (fast or slow) upon arrival. Additionally, BDP features a background data migration mechanism to address changes in data temperature levels over time, thereby mitigating performance degradation and maintaining SSD efficiency.

To enhance performance based on a single Gray code by combining two programming directions, we conduct an experimental analysis based on the heat-aware data placement strategy [6] to choose the optimal Gray code for BDP. The

978-3-9826741-0-0/DATE25/© 2025 EDAA

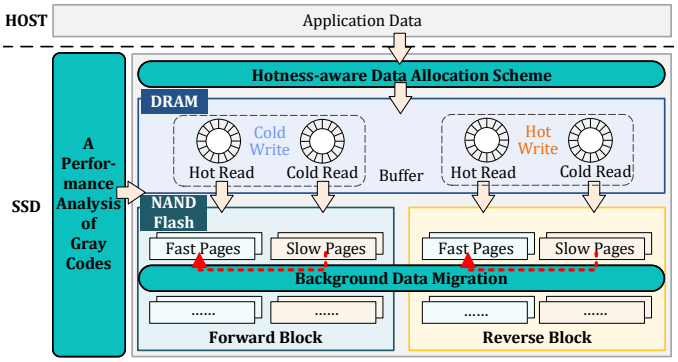

Fig. 2. The architecture of BDP in QLC-SSDs.

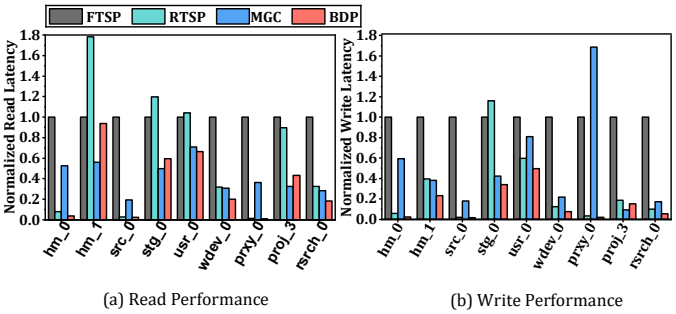

Fig. 3. I/O performance.

experimental results demonstrate that the 1-2-4-8 Gray code delivers the best-read performance. Therefore, BDP adopts the fixed 1-2-4-8 Gray code to minimize circuit complexity and avoid additional overhead.

To better align data characteristics with page performance, BDP adopts a hotness-aware data allocation scheme. Data is categorized into four types based on access characteristics: hot-read-hot-write, hot-read-cold-write, cold-read-hot-write, and cold-read-cold-write. Table I provides a summary of the target pages (LSB/CSB/MSB/TSB) and programming directions for each data type. Notably, RPI_L/C is identified as suitable for storing hot-read-hot-write data, similar to FP_L/C. Furthermore, given that RP_Blk offers an additional position for hot-write data storage (RP_M/T), RPI_L/C is prioritized as the primary location for hot-read-hot-write data.

TABLE I
DATA ALLOCATION

Data Type	Hot Read Hot Write		Hot Read Cold Write	Cold Read Hot Write	Cold Read Cold Write
Bit Location	LSB /CSB (L/C)	LSB /CSB (L/C)	MSB /TSB (M/T)	LSB /CSB (L/C)	MSB /TSB (M/T)
Program Direction	Forward (FP)	Reverse with Invalid Program (RPI)	Reverse (RP)	Reverse (RP)	Forward (FP)

Given that workloads can change and cold data may become hot, if there is a mismatch between the hotness of the data and bit position, it is essential to transform the storage locations for such data. To address this issue, BDP proposes a background data migration strategy. Specifically, when a read operation is performed, the access characteristics of the data are re-evaluated. If the data temperature changes and the performance of its current location no longer meets the re-evaluated characteristics, a new write request is generated and added to the request queue.

III. EXPERIMENTS

BDP was evaluated using SSDsim [7], leveraging real workload traces from the MSR-Cambridge dataset [8]. We compared the performance of BDP against three baseline approaches: FTSP, RTSP, and MGC. FTSP represents a forward programming algorithm [4], RTSP employs a reverse programming order based on the TSP(4,16) algorithm [9], and MGC utilizes a multiple Gray codes programming scheme [5]. The experimental results, as shown in Figure 3, demonstrate that BDP significantly enhances SSD read and write performance.

ACKNOWLEDGMENT

The work described in this paper is supported in part by NSFC (U23B2040, 62122056, and 62102263), in part by Guangdong Basic and Applied Basic Research Foundation (2022A1515010180), in part by Shenzhen Science and Technology Program (RCJC20221008092725019, JCYJ20210324094208024, and 20220810144025001), in part by Guangdong Province Key Laboratory of Popular High Performance Computers (2017B030314073), in part by Guangdong Province Engineering Center of China-made High Performance Data Computing System and Shenzhen Key Laboratory of Service Computing and Applications, in part by the grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (GRF 14219422, GRF 14202123). Tianyu Wang is the corresponding author.

REFERENCES

[1] N. Shibata, H. Maejima *et al.*, "A 70nm 16Gb 16-level-cell NAND Flash Memory," *JSSC*, vol. 43, no. 4, pp. 929–937, 2008.

[2] T. Wang, Y. Zhu, S. Li, J. Xue, C. Ma, Y. Wang, Z. Shen, and Z. Shao, "Nice: A nonintrusive in-storage-computing framework for embedded applications," *TCAD*, vol. 43, no. 11, pp. 3876–3887, 2024.

[3] C. Ma, Z. Zhou, L. Han, Z. Shen, Y. Wang, R. Chen, and Z. Shao, "Rebirth-FTL: Lifetime optimization via approximate storage for NAND flash memory," *TCAD*, vol. 41, no. 10, pp. 3276–3289, 2022.

[4] P. Kalavade, "4 bits/cell 96 layer Floating Gate 3D NAND with CMOS under Array Technology and SSDs," in *IMW*. IEEE, 2020, pp. 1–4.

[5] Y. Lv, L. Shi, Q. Li, C. Gao, Y. Song, L. Luo, and Y. Zhang, "MGC: Multiple-Gray-code for 3d NAND Flash based High-Density SSDs," in *HPCA*. IEEE, 2023, pp. 122–136.

[6] Y. Lv, L. Shi, C. J. Xue, Q. Zhuge, and E. H.-M. Sha, "Latency Variation Aware Read Performance Optimization on 3D High Density NAND Flash Memory," in *GLSVLSI*, p. 411–414.

[7] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and C. Ren, "Exploring and Exploiting the Multilevel Parallelism Inside SSDs for Improved Performance and Endurance," *TC*, vol. 62, no. 6, pp. 1141–1155, 2013.

[8] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron, "Migrating server storage to SSDs: analysis of tradeoffs," in *EuroSys*, 2009, pp. 145–158.

[9] C.-C. Ho, Y.-C. Li, Y.-H. Chang, and Y.-M. Chang, "Achieving Defect-Free Multilevel 3D Flash Memories with One-Shot Program Design," in *DAC*, 2018, pp. 1–6.