srNAND: A Novel NAND Flash Organization for Enhanced Small Read Throughput in SSDs

Jeongho Lee, Sangjun Kim, Jaeyong Lee, Jaeyoung Kang, Sungjin Lee, Nam Sung Kim, *Fellow, IEEE*, and Jihong Kim, *Member, IEEE*

***Abstract*—Emerging data-intensive applications with frequent small random read operations challenge the throughput capabilities of conventional SSD architectures. Although Compute Express Link enabled SSDs allow for fine-grained data access with reduced latency, their read throughput remains limited by legacy block-oriented designs. To address this, we propose srNAND, an advanced NAND flash architecture for CXL SSDs. It uses a two-stage ECC decoding mechanism to reduce read amplification, an optimized read command sequence to boost parallelism, and a request merging module to eliminate redundant operations. Our evaluation shows that srSSD can improve read throughput by up to 10.4× compared to conventional CXL SSDs.**

***Index Terms*—Compute Express Link, CXL.mem, flash read sequence, NAND flash architecture, partial flash read, SSD, NAND flash memory.**

I. Introduction

MODERN data-intensive applications, such as Deep Learning Recommendation Models (DLRM) and Vector Databases (VectorDB), depend on extremely large-scale datasets that exceed the capacity of host DRAM. As a result, these datasets are primarily stored in high-capacity SSDs optimized for high read bandwidth. However, accessing data on SSDs requires block I/O operations through the operating system kernel, followed by transfer to DRAM for processing. This approach introduces significant inefficiencies, particularly for small random data access patterns that are commonly observed both in DLRM and VectorDB. Frequent kernel context switches and block I/O overheads further exacerbate the challenges of handling small read operations.

A promising solution for efficiently supporting small reads for SSDs is the integration of the Compute Express Link (CXL) interface, specifically the CXL.mem protocol [1]. This interface enables direct access to 64-B data blocks without requiring kernel-level block I/O operations. SSDs equipped with the CXL interface (referred to as CXL SSDs in this letter) are thus better suited for handling small random reads.

However, simply integrating the CXL interface into conventional SSDs is insufficient to meet the high read throughput demands of data-intensive applications. Fig. 1 shows the variations in effective read bandwidth for DLRM and VectorDB in a baseline CXL SSD (baseSSD). The baseSSD assumes a traditional SSD internal organization with a large DRAM cache but supports a 64-B CXL.mem interface. To minimize the overhead of long NAND flash sensing time, we assume that baseSSD consists of ultra-low latency (ULL) NAND flash memory [2], which offers a faster sensing time of 3 μs and adopts the plane-independent read (PIR) scheme [3] to enhance read parallelism (i.e., up to 8 reads per chip in parallel). As dataset sizes grow beyond eight times the internal DRAM capacity, baseSSD achieves just 11% of its maximum theoretical read bandwidth.

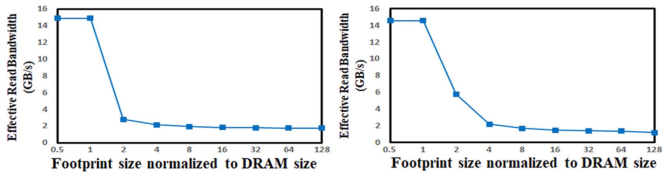

Fig. 1. Variations in effective read bandwidth of baseSSD across different dataset sizes.

The poor read throughput of baseSSD, as shown in Fig. 1, can be attributed to two key inefficiencies in handling small reads. First, baseSSD causes significant internal read amplification when serving a 64-B CXL.mem read request. This occurs because it must read a 2-KB page—the unit of ULL NAND flash memory read—resulting in a 32-fold amplification of data movement from the NAND flash chip compared to what was requested by the CXL.mem operation. Second, baseSSD is not optimized for efficiently reading small data from a NAND flash chip. Our evaluation shows that the data transfer utilization (DTU) of a NAND flash channel, which represents the proportion of the NAND flash channel bandwidth utilized for data transfer, can be as low as 6% . Since a large portion of the NAND flash channel bandwidth is not utilized for data transfer, achieving high read bandwidth for many small random reads becomes highly challenging.

In this letter, we propose srNAND, a novel NAND flash architecture designed to optimize small read throughput in CXL SSDs. The design of srNAND focuses on overcoming the three key challenges of CXL SSDs in supporting small read operations. First, srNAND supports a 64-B partial read operation without large read amplification. To achieve this, we introduce a two-stage ECC decoding mechanism that splits the ECC decoding process into on-chip and off-chip stages. This scheme reduces the read amplification factor for a 64-B CXL.mem read from 32× to 1.3×.

Second, srNAND enhances its NAND flash read command sequence to improve the DTU for 64-B reads. We streamline the existing three-command read sequence into a two-command read sequence, enabling both commands in the new sequence to be interleaved with commands from other read requests. This optimization significantly

Received 11 February 2025; accepted 4 April 2025. Date of publication 19 May 2025; date of current version 30 June 2025. This work was supported in part by the Ministry of Science and ICT (MSIT), Korea under Grant RS-2024-00456287 (Global Scholars Invitation Program), and in part by the Institute for Information & communications Technology Planning & Evaluation (IITP) under Grant RS-2024-00459026 (Energy-Aware Operating System for Disaggregated Systems). *(Jeongho Lee and Sangjun Kim contributed equally to this work.) (Corresponding author: Jihong Kim.)*

Jeongho Lee, Sangjun Kim, Jaeyong Lee, and Jihong Kim are with the Department of Computer Science and Engineering, Seoul National University, Seoul 08826, Republic of Korea (e-mail: jihong@davinci.snu.ac.kr).

Jaeyoung Kang and Nam Sung Kim are with the Department of Electrical & Computer Engineering, University of Illinois, Urbana-Champaign, IL 61801 USA.

Sungjin Lee is with the Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang 37673, Republic of Korea.

Digital Object Identifier 10.1109/LCA.2025.3571321

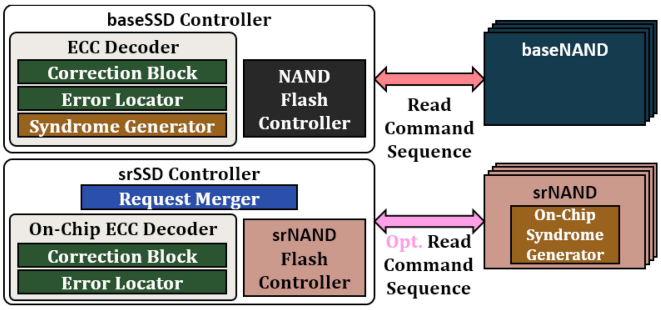

Fig. 2. An organizational comparison of baseSSD and srSSD.

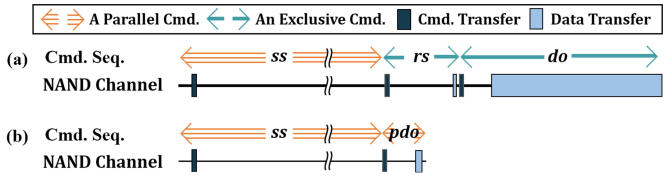

Fig. 3. Read command sequences of (a) baseNAND and (b) srNAND.

enhances parallelism among 64-B read requests, boosting the NAND flash channel's DTU from 6% to 32% .

Finally, we propose a request merging scheme at the SSD controller level, consolidating multiple CXL.mem requests targeting the same NAND flash page. This optimization avoids unnecessary NAND flash sensing operations when multiple CXL.mem requests are reading from the same NAND flash page. Such a pattern is commonly observed when a host issues medium-sized read requests (e.g., 256-B reads) via CXL.mem, as these requests are split into multiple 64-B CXL.mem requests before being sent to a CXL SSD.

To evaluate the effectiveness of srNAND, we built an SSD simulator, srSSD, that incorporates a performance model for srNAND chips with the proposed optimization schemes. Our evaluation results, using three real-world traces—embedding lookup [4], key-value store [5], and VectorDB search [6]—showed that srSSD was able to improve maximum read throughput by 10.4× compared to baseSSD.

II. Design of srSSD

As shown in Fig. 2, the overall architecture of srSSD closely resembles those of conventional SSDs, with the primary distinction being the use of srNAND flash chips and a specialized controller tailored for them. The srNAND flash chip, built on the same ULL flash memory as the baseNAND flash chip to minimize NAND flash sensing time, integrates an on-chip syndrome generator to support a two-stage decoupled ECC (Section II-A). Unlike a conventional ECC decoder, which processes a 2-KB page, the off-chip ECC decoder handles only the requested 64-B data, along with an additional 30-B syndrome vector from the on-chip ECC decoder. The srNAND flash controller optimizes the read command sequence for the srNAND, mitigating the overhead that constitutes a critical bottleneck in achieving high throughput for small reads (Section II-B). Finally, the request merger module in srSSD eliminates redundant NAND flash sensing when successive reads target the same NAND flash page (Section II-C).

A. Read Amplification Optimization

Partial data read-out can enhance small read throughput; however, it presents challenges for ECC in SSDs. Typically, error correction in NAND flash memory requires large codewords units (e.g., 4 KB or larger), which constrains the use of partial reads. The reason for employing such large codewords is because they provide higher error correction capability with a lower parity overhead than small codewords. In srSSD, we utilize 256-B BCH ECC codes [7], tailored to address the error characteristics of srNAND, which is based on ULL NAND flash memory. To achieve the required reliability standards with *raw bit error rate* of 0.001 [8] for ULL NAND flash memory, 240 parity bits (= 30-B) are necessary. Codewords smaller than 256-B would require a higher proportion of parity, making them unsuitable for accommodation in the spare area of ULL flash memory. Yet, the 286-B codeword, consisting of 256-B data and 30-B of parity, still remains more than four times larger than the 64-B unit used in CXL.mem. A straightforward solution to minimize the read amplification factor for a 64-B read is to integrate a 256-B ECC decoder on-chip. Nonetheless, this approach demands significantly higher-performance ECC decoders to achieve the high read throughput required by srNAND chips. Unfortunately, integrating such high-performance ECC decoders on-chip is not feasible due to limited chip area.

To address this limitation, we propose a decoupled ECC approach. In BCH ECC codes, a vector known as *syndrome*, derived from the codeword, is used to identify the location of error bits. The proposed srNAND chip incorporates an on-chip syndrome generator (instead of an on-chip full ECC decoder), enabling the syndrome vector to be computed directly on-chip. The NAND flash chip transmits both the computed syndrome and the 64-B data requested by CXL.mem to the controller, where the error correction process is finalized using the error locator and correction block located within the controller. By separating an ECC decoder into on-chip and off-chip modules, a 64-B CXL.mem read requires 94-B of data transfer instead of 286-B of data transfer, lowering channel usage to roughly one-third compared to transmitting the full 256-B codeword. Additionally, syndrome generation is the least complex stage of the decoding process, making it highly suitable for integration within the constrained area of a NAND flash chip.

B. Command Sequence Optimization

Despite the effective mitigation of read amplification for small reads through the two-stage decoupled ECC decoder, the read bandwidth of srSSD does not significantly improve. Our analysis shows that the root cause of the poor read bandwidth is severely limited parallelism in handling 64-B read requests. This low parallelism stems from the existing three-command read sequence. As illustrated in Fig. 3(a), the three-command read sequence—*Sensing* (*ss*), *Read Status* (*rs*), and *Data-Out* (*do*). Among them, *do* and *rs* are *exclusive* commands, which cannot be issued in parallel with other commands that utilize the NAND flash channel. Only an *ss* command can execute concurrently with *rs* and *do* commands, significantly limiting the NAND flash channel's DTU.

To address the limitations of the existing read command sequence, we propose a new read command sequence that eliminates exclusive commands from the sequence (Fig. 3(b)). In the proposed sequence, the first exclusive command, *rs*, is removed. The *rs* command checks whether a NAND flash chip is ready for data read-out and whether any failure occurred during *ss*. To avoid wasting NAND flash channel bandwidth during the execution interval of an *rs* command (Fig. 3(a)), we embed a sensing failure flag within the read-out data, thus eliminating an exclusive *rs* command. When a sensing failure occurs, we indicate the exception condition by using a disallowed data read-out response of

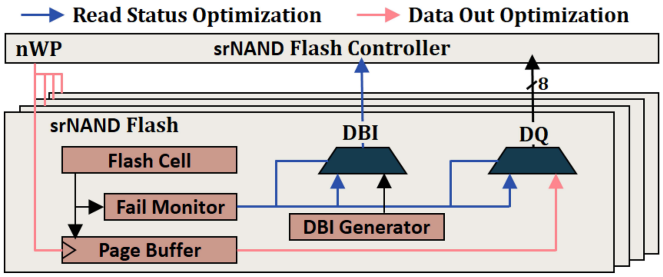

Fig. 4. Overview of an srNAND implementation.

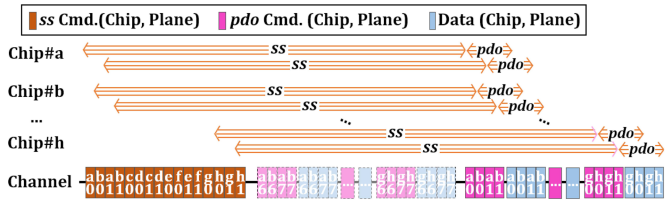

Fig. 5. Channel utilization improvement in srNAND.

the *do* command.[1] This approach not only removes the need for the *rs* command but also improves the overall efficiency of the read sequence.

The second exclusive command, the *do* command, is converted into a parallel command, also referred to as *pdo* command in this letter. In the existing read command sequence, *do* commands are scheduled exclusively by the NAND flash controller to avoid potential data corruption on a shared NAND flash channel. This exclusive scheduling ensures that only one NAND flash chip drives the DQ pins (i.e., transfers data from its page buffer to the channel and then to the controller) at any given time. When the controller broadcasts a read-enable signal to all NAND flash chips on the channel, only one chip (which executed the *do* command) is ready to perform data-out operations, thus preventing bus contention.

By contrast, *pdo* commands allow concurrent operations. The srNAND flash controller sets explicitly a data-out permission flag of a target srNAND chip, ensuring that only the selected chip can output data and thus avoiding conflicts on the shared channel. Consequently, while *do* commands cannot execute in parallel with other *do* commands, *pdo* commands can operate in parallel with other *pdo* commands as well as *ss* commands. For instance, as shown in Fig. 3, issuing a *do* command requires an 80-ns setup interval before data transfer begins—during which no other NAND flash chip can perform data-out although the NAND flash channel is idle. In the *pdo* command, however, this setup interval does not demand exclusivity, enabling multiple NAND flash chips to prepare for data transfer in parallel. The setup delay is used for internal preparations, such as enabling the NAND flash interface and configuring the I/O path. The data-out permission flag is implemented without additional pins. As shown in Fig. 4, the existing nWP (write protect) pin is reused as a data-out permission flag during read operations, facilitating efficient parallel *pdo* commands. With the optimized read command sequence, the latency of a 64-B read is reduced to 3.1 μs, which is 19% lower than the existing sequence (3.9 μs).

More critically, the optimized read command sequence significantly boosts small-read bandwidth. Fig. 5 illustrates this improvement in an srSSD system where eight NAND flash chips share a channel, and each chip can issue two[2] concurrent plane-independent reads. It clearly indicates that the NAND flash channel's DTU is significantly improved by enhanced parallelism in commands. The dotted-border boxes illustrate how DTU can be increased by overlapping idle periods on the NAND flash channel with *pdo* commands, as well as transferring data in other planes where the *ss* command has already completed. Because both *ss* and *pdo* commands in the optimized sequence can run in parallel, srNAND can perform much more 64-B data transfers within a single tR (the *ss* latency) compared to baseNAND.

[1] The data-bus invert (DBI) is set to 1 only when fewer 1's are transmitted via the DQ pins (to reduce power consumption) than 0's (Fig. 4). Therefore, transmitting eight 1's on the DQ pins with DBI at 1 is not allowed.

[2] For an illustration purpose, we assume two plane-independent reads.

TABLE I
KEY CHARACTERISTICS OF COLLECTED I/O TRACES

Workload	Read Size(B)	Read Ratio	Footprint Size (MB)	Remarks
dlrm 1	128	1	2852	RM1 model
dlrm 2	256	1	2445	RM2 model
dlrm 3	128	1	1782	RM3 model
redis 1	64-192	0.92	1240	YCSB B
redis 2	64-192	0.94	1241	YCSB C
redis 3	64-192	0.77	1081	YCSB D
vectordb	64-128	0.98	539	PQ + HNSW

C. Request Merging Optimization

While CXL.mem requests are aligned to 64 B, the data used by the host applications is often larger than 64 B. For example, the size of embedding vectors used in our DLRM experiments ranged from 128 B to 256 B. This results in multiple CXL.mem requests for adjacent addresses, and such requests for adjacent addresses lead to redundant NAND flash sensing on the same NAND flash chip, which significantly reduces parallelism. In srSSD, we incorporate a request merger module that prevents redundant NAND flash sensing caused by multiple read requests to adjacent addresses.

As illustrated in Fig. 4, in the existing NAND flash read operation, the sensed data are stored in the page buffer after tR. Even though a CXL.mem request only requires specific 64-B data, the *ss* command loads an entire 2-KB page (containing the 64-B data) into the page buffer. The request merger module monitors incoming read requests and determines whether they can be serviced directly from the page buffer without issuing an additional *ss* command. If an *ss* command is not required, the srNAND flash controller issues a special *pdo* command, *pdo_pb*, which transfers the requested 64-B data directly from the page buffer without an *ss* command. In the current implementation, up to 32 successive read requests can be supported by the *pdo_pb*. The request merger module maintains the address of the last *ss* command to detect whether subsequent read requests can be serviced from the page buffer using *pdo_pb* commands.

III. EXPERIMENTAL RESULTS

Experimental Setup: To validate the effectiveness of the proposed srSSD architecture, we develop an srSSD simulator as an extension to the existing MQSim-E simulator [9]. In our evaluation, we use read-intensive I/O traces collected from three scenarios: (1) embedding table lookups in DLRM (`dlrm`), (2) key-value stores operating in in-memory Redis database (`redis`), and (3) search operations with product quantization in VectorDB (`vectordb`). Table I summarizes the key characteristics of these traces. We vary the internal DRAM cache size in the simulator relative to the memory footprint to evaluate the impact of different DRAM configurations.

Evaluation: We first measured the improvement of the read bandwidth over baseSSD when using srSSD. Fig. 6 summarizes the effective read bandwidth changes under varying DRAM sizes in selected workloads. Overall, srSSD is very effective in improving the read bandwidth

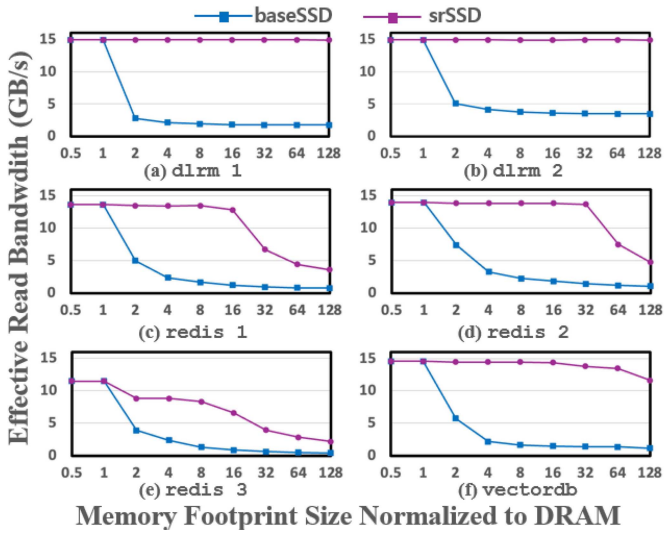

Fig. 6. Effective bandwidth improvement in `dlrm`, `redis`, and `vectordb` with srSSD.

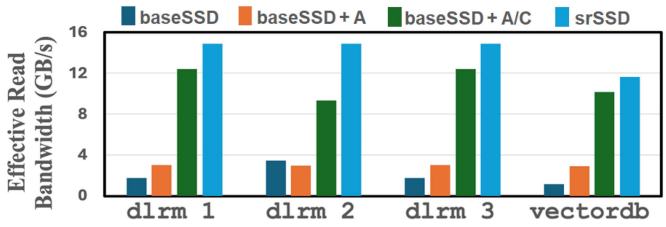

Fig. 7. Cumulative impact of optimization techniques on read bandwidth. (srSSD == baseSSD + A/C/M.).

of small reads. For example, in `dlrm`, srSSD improves 6.4 times over baseSSD under RM1 model. Fig. 6(c), (d), and (e) show srSSD achieves relatively lower improvements in `redis` compared to `dlrm` and `vectordb`. This is because of the `redis` application has a higher proportion of writes compared to other applications (Table I). The NAND flash programming caused by writes takes significantly longer than NAND flash reads, which greatly reduces the parallelism of the NAND flash channel. Notably, since YCSB D (`redis 3`) exhibits the highest write proportion, it offers the least performance improvement. In contrast, `vectordb`, with only a 2% write proportion, improves the performance by up to 10×.

To gain a better understanding of the impact of the three proposed optimizations for srSSD, the read bandwidth was evaluated by selectively applying each optimization. Fig. 7 compares four srSSD configurations with different combinations of enabled optimizations; we denote three optimizations as A, C, and M, respectively, for the read **a**mplification optimization, **c**ommand sequence optimization and request **m**erging optimization). For instance, baseSSD + A/C indicates that the read amplification optimization and the command sequence optimization are enabled. The most effective optimization was the command sequence optimization. Although the impact of the read amplification optimization is more straightforward to expect, the read bandwidth was improved mainly by the increased parallelism among read commands, which allowed the NAND flash channel to be more effectively utilized.

When the command sequence optimization is combined with the read amplification optimization, a total of 62 concurrent 64-B read operations can be processed within a single tR—more than a 20-fold increase compared to baseSSD, which manages only three 64-B transfers due to exclusive command execution and read amplification. Fig. 7's `dlrm 2` case demonstrates the effectiveness of the request merging optimization. Since each 256-byte embedding in the RM2 model splits into four 64-B CXL.mem operations, adding request merging combines these fragments with *pdo_pb* commands, boosting read bandwidth by 4.3× over baseSSD.

IV. Conclusion

We introduced srNAND—a novel extension to NAND flash chips that boosts read bandwidth for small random reads in CXL SSDs. With an on-chip syndrome generator, srNAND lets the SSD controller read 64-B data with low read amplification and high parallelism through a fully interleavable command sequence. A request merger module is also proposed for handling medium-sized reads. Experimental results show that srNAND and our SSD organization are effective for data-intensive applications with many small random reads.

Acknowledgment

The ICT at Seoul National University provided research facilities for this study.

References

[1] CXL Consortium, "CXL specification revision 1.1," 2019. [Online]. Available: https://computeexpresslink.org/wp-content/uploads/2024/02/CXL-1.1-Specification.pdf
[2] W. Cheong et al., "A flash memory controller for 15 μs ultra-low-latency SSD using high-speed 3D NAND flash with 3 μs read time," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2018, pp. 338–340.
[3] L. Heineck et al., "3D NAND flash status and trends," in *Proc. IEEE Int. Memory Workshop*, 2022, pp. 1–4.
[4] U. Gupta et al., "DeepRecSys: A system for optimizing end-to-end at-scale neural recommendation inference," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Architecture*, 2020, pp. 982–995.
[5] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proc. 1st ACM Symp. Cloud Comput.*, 2010, pp. 143–154.
[6] M. Douze et al., "The faiss library," 2024, *arXiv:2401.08281*.
[7] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inf. Control*, vol. 3, pp. 68–79, 1960.
[8] J. Park et al., "Flash-cosmos: In-flash bulk bitwise operations using inherent computation capability of NAND flash memory," in *Proc. 55th IEEE/ACM Int. Symp. Microarchitecture*, 2022, pp. 937–955.
[9] D. Lee, D. Hong, W. Choi, and J. Kim, "MQSim-E: An enterprise SSD simulator," *IEEE Comput. Archit. Lett.*, vol. 21, no. 1, pp. 13–16, Jan.-Jun., 2022.