# Overcoming Capacitance Constraints in SSDs

Sanghyun Nam, Seungmin Shin
*Soongsil University*
{ssushnam, smshin}@ssu.ac.kr

Sungjin Lee
*DGIST*
sungjin.lee@dgist.ac.kr

Bryan S. Kim
*Syracuse University*
bkim01@syr.edu

Eunji Lee
*Soongsil University*
ejlee@ssu.ac.kr

***Abstract*—This paper presents `Hexa`-SSD, a novel SSD-internal DRAM management scheme that allows the SSD capacity to scale beyond the slow growth of capacitors. `Hexa`-SSD judiciously manages the dirty memory footprint within the SSD-internal buffer by using a low-overhead data reordering scheme on the deep queues available in today's storage interfaces. In doing so, our design guarantees crash consistency while using a fraction of the capacitors compared to the state-of-the-art designs. We implement our design in FEMU and demonstrate that `Hexa`-SSD delivers up to 1.4× higher IOPS and up to 49% less write amplification compared to the existing scheme under power constraints.**

***Index Terms*—energy efficiency, flash memory, reliability**

## I. Introduction

Charge-storing capacitors are central to the reliability of and data integrity in SSDs as they provide enough energy to safely persist data stored in volatile DRAM during a power crash. Without capacitors, an SSD cannot implement Power-Loss Protection (PLP) [11, 18, 23], and would require the SSD to perform a long and arduous recovery process by scanning all the pages in the SSD to rebuild the logical to physical mapping table. As shown in Table I, while client-class SSDs may forgo the implementation of PLP due to size and cost limitations, using capacitors is a must in enterprise-class SSDs.

However, the heavy reliance on capacitors is no longer sustainable as the increase in SSD far outpaces the increase in capacitor density. In 2011, a typical 2.5-inch SSD had 256GB capacity, but by 2018, a high-capacity SSD boasted 30TB, expanding by 100× over the past ten years [1, 21]. This remarkable growth in the SSD's capacity is thanks to the vertical stacking of layers that break the process scaling limit and multi-level cells that store multiple bits in a given transistor. On the other hand, Al(aluminum) and Ta(tantalum)-electrolytic capacitors used in SSDs have increased in density only by tenfold across four decades, approximately a 50× slower rate per year. The slow scaling of capacitors will eventually limit the percentage of DRAM that can be protected by PLP. Without a scheme to ensure the durability of data in a capacitor-constrained setting, this, in turn, will also limit the storage capacity as the size of DRAM and aggregate flash capacity proportionally scale [19].

This paper presents `Hexa`-SSD, a novel SSD-internal DRAM management scheme that allows the SSD capacity to scale beyond the slow growth of capacitors. `Hexa`-SSD uses a radically different approach to managing the SSD-internal DRAM. Rather than caching most of the mapping table and minimally buffering user writes [13], we buffer more user writes so that the number of modified mapping pages is small. This substantially reduces the amount of mapping table-related write traffic, and in turn, improves the overall performance. Our approach of buffering more user writes is enabled by the current trend of increasing the queue depth of the storage interfaces: NVMe has up to 65,535 queues with as many as 65,536 commands per queue. This extension allows SSDs to further optimize the internal activities by taking advantage of the outstanding request information.

To realize this design, `Hexa`-SSD maintains two data structures: first, *a zero-cost list* that holds the write requests whose mapping entry is already in a dirty translation page, and second, *a max binary heap* that maintains the indexes to translation pages sorted by the number of buffered user write requests associated with that page. When there is sufficient bandwidth at the underlying NAND flash subsystem for writes, `Hexa`-SSD first flushes user data from the zero-cost list, and then persists the dirty translation pages as ordered by the max binary heap. By doing so, each user write minimizes the number of eventual translation page write, and each translation page write maximizes the number of persisted mapping entries. We implement `Hexa`-SSD in FEMU, an open-source SSD development framework [17]. The performance evaluation with various workloads shows that `Hexa`-SSD offers 82% of IOPS of the full-protection SSD when a protected ratio is equal to or smaller than 10%, while a conventional SSD provides 74% of performance.

## II. Related work

The need to reduce the energy consumption for power-loss protection arises in different contexts. In a few studies, the total energy consumption was reduced past? present? by speeding up the backup process upon power failure using fast media [7, 10]. Guo et al. [7] reduce the capacitance requirement by writing back the volatile buffer data into Phase Change Random Access Memory (PRAM), which is faster and uses less power than NAND flash does. Smartbackup [10] proposes dynamic NAND channel allocation and single-level

TABLE I: Power Loss Protection in SSDs [11, 18, 23].

| SSD Model | Manufacturer | Class | PLP | Capacitor |
|---|---|---|---|---|
| 950Pro, 850Pro | Samsung | Client | None | - |
| M500 | Micron | Client | Partial | Ceramic |
| M500DC | Micron | Enterprise | Full | Tantalum |
| PM863, SM863 | Samsung | Enterprise | Full | Tantalum |
| DC1000B | Kingston | Enterprise | Full | Tantalum |
| DC S3700, S3500 | Intel | Enterprise | Full | Aluminum |

cell (SLC) mode programs to make the dump process shorter at sudden power-off.

Another approach to reducing the capacitor size is achieved by only protecting a small portion of the volatile buffer. A Dual-Region Write Buffer (DRWB) [14] divides the internal-SSD buffer into a small protected region and a large unprotected region. When the data on the unprotected region is updated, DRWB logs the delta for the page in the protected region. However, DRWB only performs this for the user data, having no consideration for the metadata such as mapping tables, despite that it actually accounts for most of the internal buffer of SSDs.

Prior work by Chen et. al [4] has also addressed the challenge of capacitance constraints for scalable SSDs. They observed that a small write buffer can be effective for reducing write traffic in particular applications that perform heavy journaling. Motivated by this observation, they present an application-SSD co-design to reduce the data writes buffered for heavy journaling applications. However, their approach is only applicable to systems that perform journaling and require changes to the application code.

SpartanSSD [16], which is most related to this work, pinpoints capacitance constraints in scalable SSDs and reduces capacitance requirements by making use of elastic journaling. SpartanSSD logs the mapping information updates into the in-device journal so that the writes to the mapping table can be buffered. They use a hybrid journal that is backed by a small DRAM and flash memory. This hybrid journal is highly flexible in terms of capacity; thus, it enables a timely checkpoint that reflects the log data to the mapping table and flushes dirty map pages into NAND flash chips. Although SpartanSSD also reduces translation-related writes under capacitance constraints, it has double write for mapping information updates, and more importantly, it increases the recovery time, which could be extremely harmful when the multiple SSDs are running simultaneously.

## III. Design and Implementation

Hexa-SSD partially protects a mapping table with limited capacitance. When the dirty pages of a mapping table reach more than the maximum number of protected pages, Hexa-SSD flushes them to flash memory based on the Least-recently Used (LRU) algorithm. Because the flush operation does not arise with SSD using PLP, mitigating the negative impact of flushing is critical to maintain performance under capacitance constraints. To this end, Hexa-SSD uses a cost-effective scheduling scheme for the in-storage buffer. Hexa-SSD selectively forces the user data to flash memory, which increases the dirtiness of the mapping table the least. This scheme reduces a dirty memory footprint of the mapping table at any given moment by enhancing a locality of updates. As a result, the frequency of flush operation for the mapping table can be greatly reduced.

Fig. 1 compares the flush overhead of FIFO-SSD and Hexa-SSD. In this example, there are seven write requests in the device queue sent from the host in the following order:

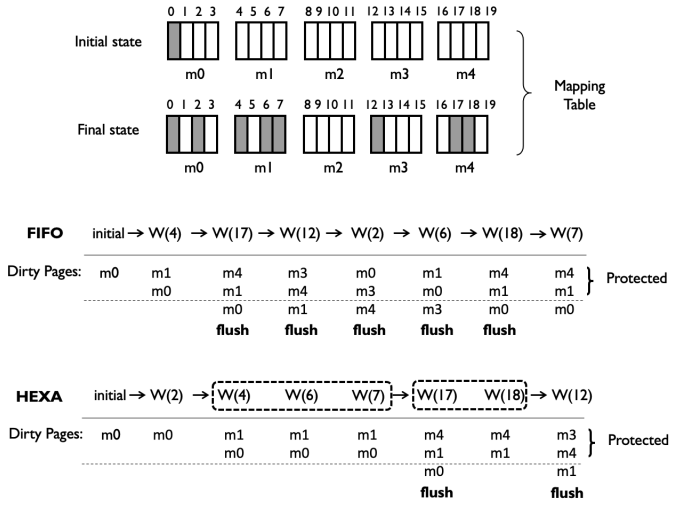



Fig. 1: **Hexa-SSD buffer management scheme.**

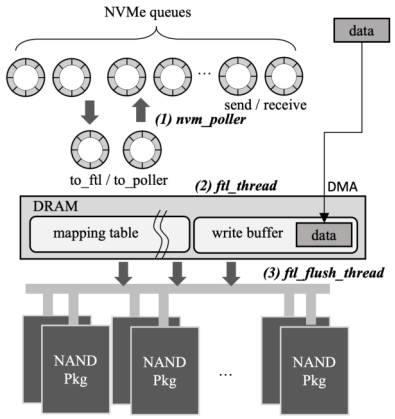



Fig. 2: **Hexa-SSD Architecture.**

W(4), W(17), W(12), W(2), W(6), W(18), and W(7). The mapping table has one dirty page (m0) at an initial state. We assume that 2 out of 5 pages of the mapping table are protected. FIFO-SSD writes the user data in the buffer to flash memory in arrival order. With this scheme, the mapping table would be randomly updated, generating a large number of dirty pages at a time window. Consequently, FIFO-SSD incurs a total of five flushes of the mapping table page during the write process.

In contrast, Hexa-SSD calculates the write cost for each datum that indicates an increase in the number of dirty pages of the mapping table when it is flushed, and it processes the request with minimum cost first. In this example, the write request W(2) has top priority because its associated mapping table page (m0) is already dirty, and thus it does not add to the number of dirty translation pages. Next, the write requests W(4), W(6), and W(7) are processed. Because their address mapping entries are on the same page of the mapping table, the cost of flushing them is reduced to one third. With this scheme, Hexa-SSD delivers only two flushes of the mapping table for the same task.

Fig. 2 shows the overall architecture of Hexa-SSD. Hexa-

SSD is implemented by extending FEMU, an open-source SSD development framework [17]. Hexa-SSD communicates with a host using NVMe storage interfaces and uses a small write buffer, which aggregates and batches user writes into the underlying flash memory.

Hexa-SSD maintains three different threads that are executing concurrently within SSDs. The nvm_poller takes a charge of transferring requests between NVMe queues and FTL-internal queues. The FTL-internal queue consists of a pair of sub-queues, named to_ftl and to_poller. This separation is intended to enable a non-blocking access to queues by allowing only a single writer for each queue. Second, the ftl_thread essentially handles the ingress requests from the internal queues. For write, it transfers data from the host memory to the SSD-internal write buffer with DMA and updates the associated entry in a translation page to point to the write buffer. Then, it notifies the completion of request to the nvm_poller by enqueueing the acknowledgement into the to_poller queue. Because Hexa-SSD protects the entire space of write buffer with capacitance, data persistency is guaranteed for all acknowledged writes. For read, the ftl_thread retrieves the requested data by consulting the mapping table and transfers them to the host.

The ftl_flush_thread materializes data from a DRAM-buffer into a flash memory. In FIFO-SSD, the user writes are issued to NAND flash memory in the order they arrive into the buffer. However, Hexa-SSD flushes buffered writes in the order such that it least increases the dirty memory footprint of the mapping table. To realize this design, Hexa-SSD maintains two data structures, as depicted in Fig. 2(b). First, *a zero-cost list* that holds the indexes to translation pages that are already in a dirty state, and second, *a max binary heap* maintains the indexes to translation pages sorted by the number of buffered user-write requests associated with that page.

When half of the write buffer becomes occupied, flushing is invoked. Hexa-SSD first flushes the user data on those translation pages in the zero-cost list, and then persists user data as their translation pages are ordered by the max binary heap. These data structures are updated by the ftl_thread when a write request arrives at SSD. To exploit the SSD internal parallelism, we send data to flash memory in batches by the number of NAND flash chips that can be written simultaneously.

Once the write operations of NAND flash memory complete, ftl_flush_thread updates the mapping table entries to point to the physical address of the data in a flash memory. At this moment, if the number of dirty mapping table pages goes beyond the protectable number of pages, ftl_flush_thread persists the mapping table page to flash memory. This is also conducted in batches by the number of NAND flash chips that can be written in parallel.

## IV. Evaluation

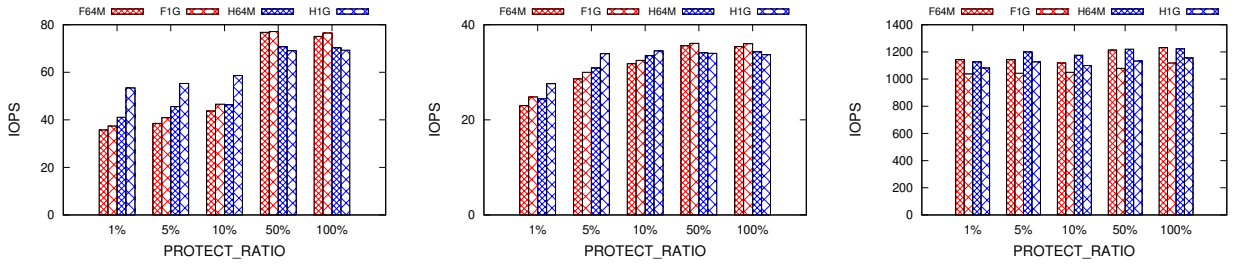
We perform the experiments on a machine with a 20-core Intel Xeon(R) Silver 4114 CPU running at 2.2GHz and 84GB memory. We run FEMU (QEMU-based SSD emulator) configured to use 10 cores, 4GB DRAM for main memory, and 16GB DRAM for SSD emulation. We use page-level mapping and caches all translation pages in DRAM. The NAND flash chips include 8 channels and 8 flash LUNs per channel. The page size is 8KB and there are 256 per-block pages. The read and write latency is set to 60 and 700$\mu$s, respectively [5]. We use the greedy algorithm for GC (Garbage-Collection) and mount an Ext4 file system on the device.

The performance evaluation is conducted using three workloads. The fio benchmark generates the 4KB of random writes and the skewed read-write mixed workload that follows JESD219 using 4 threads. A total of 64GB of data was written to the 4GB area. For the real workload, we use TPC-C [6] on MySQL, an online transactional processing benchmark. For TPC-C, we precondition an SSD so that 75% of the capacity is filled with data and perform 0.1 million write queries using 10 threads. For the performance comparison, we implemented a FIFO-SSD that processes write requests in arrival order.

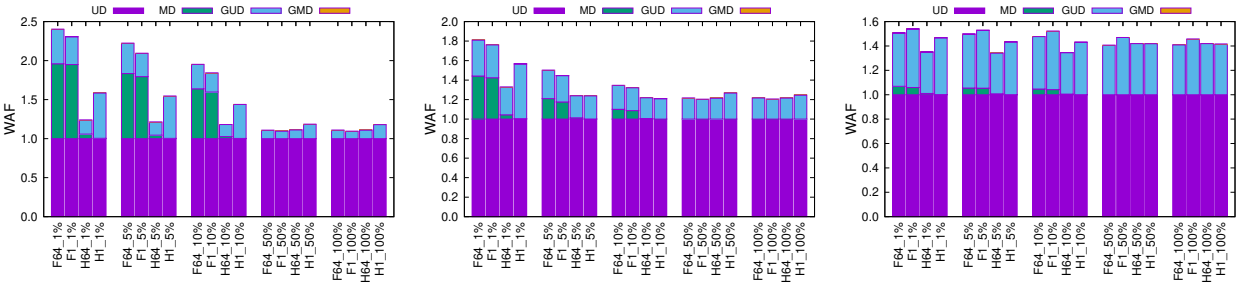
Fig.3 shows the IOPS and Write Amplification Factor (WAF) of the FIFO-SSD and Hexa-SSD (denoted with a prefix F and H, respectively) when varying the protected ratio of a mapping table from 1% to 100%. We study two different sizes of write buffer, 64MB and 1GB, to investigate the effectiveness of Hexa-SSD with respect to the queue depth. As the figure shows, the random workload improves the most with Hexa-SSD. This workload has low spatial locality innately, and thus it benefits enormously from the re-ordering of Hexa-SSD, in particular, when the buffer size is large. As a result, Hexa-SSD with 1GB buffer lowers WAF from 2.3 to 1.5 and enhances IOPS by 42.7% when the protection ratio is 1%.

For JESD and TPC-C, the result shows the same general trends as for the random workload, while the performance gain becomes smaller because they have more skewed access patterns. Hexa-SSD with 1GB write buffer improves IOPS by 10% and 5.6%, respectively, and reduces write amplification by 11.3% and 3% on average for a protection ratio under 10%. In particular, TPC-C achieves little improvement because the mapping table-related write originally accounts for only 5% of the total traffic. For this reason, TPC-C exhibits more sensitivity to the buffer size than the write traffic and it performs better with a smaller buffer. We suspect this effect is due to the increased software complexity as the buffer size becomes larger. This can have a great impact on the performance of highly skewed and latency-sensitive database workloads.

Another counter-intuitive result is that Hexa-SSD has slightly lower IOPS than FIFO-SSD when the protected ratio is equal or above 50%. Our careful analysis reveals that the reordering of Hexa-SSD distorts the original write pattern generated by the host, which increases the possibility that pages with different lifetimes are stored in the same block. This subsequently increases the number of page-copy operations during GC, amplifying the write traffic. This is the case even when the workload is synthetic random as all host writes




(a) IOPS




(b) Write Traffic (UD: User Data, MD: Mapping Data, GUD/GMD: GC for User/Mapping Data)

Fig. 3: **IOPS and WAF.** From left: Random, JESD, and TPC-C.

transferred through a file system have a locality. Although the target environment of this paper is a case where the protected ratio is low, to improve the generality of Hexa-SSD, we will study the effect of Hexa-SSD on GC performance in more detail in the future.

## V. Conclusion

This paper presented a novel SSD design called Hexa-SSD. Hexa-SSD protects a fraction of the storage-internal buffer to overcome capacitance constraints in high-capacity SSDs. Hexa-SSD raises performance by reducing the dirty memory footprint of in-DRAM data through cost-effective re-ordering, which underlies the increasing queue depth of storage interfaces. Performance evaluation with various workloads shows that Hexa-SSD offers up to 42.7% higher IOPS and 49% less write traffic when the capacitance is highly limited.

## References


[1] AnandTech. Samsung 30.72 TB SSDs: Mass production of PM1643 begins. https://www.anandtech.com/show/12448/samsung-begins-mass-production-of-pm1643-sas-ssds-with-3072-tb-capacity, 2018.

[2] Jens Axboe. fio - flexible i/o tester. https://github.com/axboe/fio, 2021.

[3] Christoph Busche, Laia Vilà-Nadal, Jun Yan, Haralampos N Miras, De-Liang Long, Vihar P Georgiev, Asen Asenov, Rasmus H Pedersen, Nikolaj Gadegaard, Muhammad M Mirza, et al. Design and fabrication of memory devices based on nanoscale polyoxometalate clusters. *Nature*, 515(7528):545–549, 2014.

[4] Xubin Chen, Yin Li, and Tong Zhang. Reducing flash memory write traffic by exploiting a few MBs of capacitor-powered write buffer inside solid-state drives (ssds). *IEEE Trans. Computers*, 68(3):426–439, 2019. https://doi.org/10.1109/TC.2018.2871683.

[5] Wooseong Cheong, Chanho Yoon, Seonghoon Woo, Kyuwook Han, Daehyun Kim, Chulseung Lee, Youra Choi, Shine Kim, Dongku Kang, Geunyeong Yu, et al. A flash memory controller for 15$\mu$s ultra-low-latency ssd using high-speed 3d nand flash with 3$\mu$s read time. In *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 338–340. IEEE, 2018.

[6] Transaction Processing Performance Council. Tpc benchmark c standard specification, 1990.

[7] Jie Guo, Jun Yang, Youtao Zhang, and Yiran Chen. Low cost power failure protection for MLC NAND flash storage systems with PRAM/DRAM hybrid buffer. In *Design, Automation and Test in Europe, DATE 13, Grenoble, France, March 18-22, 2013*, pages 859–864, 2013. https://doi.org/10.7873/DATE.2013.181.

[8] Aayush Gupta, Youngjae Kim, and Bhuvan Urgaonkar. Dftl: a flash translation layer employing demand-based selective caching of page-level address mappings. *Acm Sigplan Notices*, 44(3):229–240, 2009.

[9] Jiaoying Huang, Liang Mei, and Cheng Gao. Life prediction of tantalum capacitor based on gray theory optimization model. In *2011 IEEE International Conference on Quality and Reliability*, pages 166–171. IEEE, 2011.

[10] Min Huang, Yi Wang, Liyan Qiao, Duo Liu, and Zili Shao. SmartBackup: An efficient and reliable backup strategy for solid state drives with backup capacitors. In *17th IEEE International Conference on High Performance Computing and Communications, HPCC 2015, 7th IEEE International Symposium on Cyberspace Safety and Security, CSS 2015, and 12th IEEE International Conference on Embedded Software and Systems, ICESS 2015, New York, NY, USA, August 24-26, 2015*, pages 746–751, 2015. https://doi.org/10.1109/HPCC-CSS-ICESS.2015.180.

[11] Intel. Power loss imminent(PLI) technology. https://www.intel.com/content/www/us/en/solid-state-drives/ssd-power-loss-imminent-technology-brief.html, 2014.

[12] Song Jiang, Lei Zhang, XinHao Yuan, Hao Hu, and Yu Chen. S-FTL: An efficient address translation for flash memory by exploiting spatial locality. In *2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–12. IEEE, 2011.

[13] Woon-Hak Kang, Sang-Won Lee, Bongki Moon, Yang-Suk Kee,

and Moonwook Oh. Durable write cache in flash memory SSD for relational and nosql databases. In *ACM International Conference on Management of Data (SIGMOD)*, pages 529–540, 2014. https://doi.org/10.1145/2588555.2595632.

[14] Dongwook Kim and Sooyong Kang. Dual region write buffering: making large-scale nonvolatile buffer using small capacitor in SSD. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*, pages 2039–2046, 2015. https://doi.org/10.1145/2695664.2695830.

[15] Hyukjoong Kim, Dongkun Shin, Yun Ho Jeong, and Kyung Ho Kim. SHRD: Improving spatial locality in flash storage accesses by sequentializing in host and randomizing in device. In *15th USENIX Conference on File and Storage Technologies (FAST)*, pages 271–284, 2017.

[16] Hyeon Gyu Lee, Juwon Lee, Minwook Kim, Donghwa Shin, Sungjin Lee, Bryan S Kim, Eunji Lee, and Sang Lyul Min. Spartanssd: a reliable ssd under capacitance constraints. In *2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6. IEEE, 2021.

[17] Huaicheng Li, Mingzhe Hao, Michael Hao Tong, Swaminathan Sundararaman, Matias Bjørling, and Haryadi S Gunawi. The CASE of FEMU: Cheap, accurate, scalable and extensible flash emulator. In *16th USENIX Conference on File and Storage Technologies (FAST)*, pages 83–90, 2018.

[18] Micron. How Micron SSDs handle unexpected power loss. https://www.micron.com/-/media/client/global/documents/products/white-paper/ssd$_{p}ower_{l}oss_{p}rotection_{w}hite_{p}aper_{l}o.pdf$, 2014.

[19] Fan Ni, Chunyi Liu, Yang Wang, Chengzhong Xu, Xiao Zhang, and Song Jiang. A hash-based space-efficient page-level FTL for large-capacity SSDs. In *2017 International Conference on Networking, Architecture, and Storage (NAS)*, pages 1–6. IEEE, 2017.

[20] Jae-Woo Park, Doogon Kim, Sunghwa Ok, Jaebeom Park, Taeheui Kwon, Hyunsoo Lee, Sungmook Lim, Sun-Young Jung, Hyeongjin Choi, Taikyu Kang, Gwan Park, Chul-Woo Yang, Jeong-Gil Choi, Gwihan Ko, Jaehyeon Shin, Ingon Yang, Junghoon Nam, Hyeokchan Sohn, Seok-In Hong, Yohan Jeong, Sung-Wook Choi, Changwoon Choi, Hyun-Soo Shin, Junyoun Lim, Dongkyu Youn, Sanghyuk Nam, Juyeab Lee, Myungkyu Ahn, Hoseok Lee, Seungpil Lee, Jongmin Park, Kichang Gwon, Woopyo Jeong, Jungdal Choi, Jinkook Kim, and Kyo-Won Jin. 30.1 a 176-stacked 512gb 3b/cell 3d-nand flash with 10.8gb/mm2 density with a peripheral circuit under cell array architecture. In *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, volume 64, pages 422–423, 2021.

[21] Samsung. Samsung SSD 830 series. https://www.samsung.com/us/support/owners/product/128gb-ssd-830-series, 2011.

[22] Samsung. Samsung V-NAND SSD 860 QVO. https://www.samsung.com/semiconductor/global.semi.static/Samsung%20SSD%20860%20QVO%20Data%20Sheet$_{R}ev1.pdf$, 2013.

[23] Samsung. Power loss protection in ssds - how ssds are protecting data integrity. A Samsung Electronics White Paper, 2016.

[24] Sysbench. Sysbench. https://github.com/akopytov/sysbench.git, 2022.