

# ***OpenCxD*: An Open Real-Device-Guided Hybrid Evaluation Framework for CXL-SSDs**

Hyunsun Chung<sup>1</sup>, Junhyeok Park<sup>1</sup>, Taewan Noh<sup>1</sup>, Seonghoon Ahn<sup>1</sup>, Kihwan Kim<sup>1</sup>,  
Ming Zhao<sup>2</sup>, Youngjae Kim<sup>1</sup>



The 33<sup>rd</sup> International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), Paris, France, October 21-23, 2025

# Contents



- Background
- Motivation
- Design
- Evaluation
- Conclusion

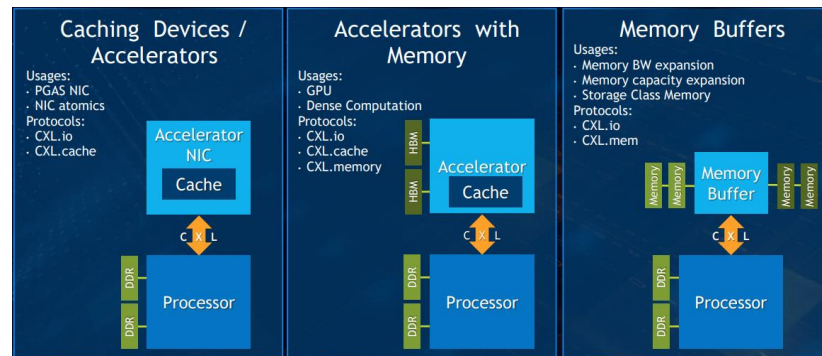
# Background

---

# CXL: A new memory interconnect



- Surge of memory intensive workloads in AI and LLMs have posed a new challenge in providing more memory capacity.
- **CXL**, a new memory interconnect, has emerged to answer the challenge.
- CXL enables additional memory space to host systems by exposing internal on-board DRAM via **PCIe-attached CXL devices**.



# CXL Devices: DRAM based



- Current forms of CXL mainly focus on memory expansion in the form of **DRAM-backed CXL memory devices**.
- Industry solutions include Samsung's CMM-D<sub>[1]</sub> and SK hynix's CMM-DDR5<sub>[2]</sub>.



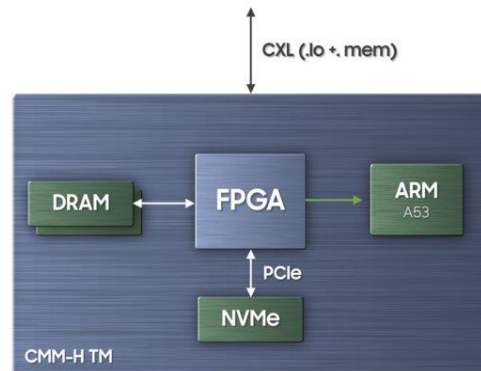
[1]: [CMM-D | CXL Memory | Samsung Semiconductor](#)

[2]: [SK hynix Completes Customer Validation of CXL 2.0-based DDR5](#)

# CXL Devices: the advent of CXL-SSDs

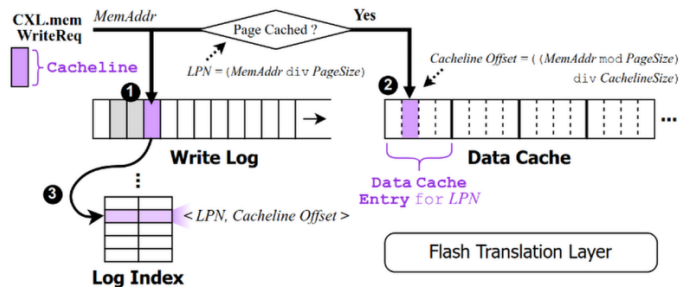


- Interest in utilizing NAND flash-based SSDs with DRAM as memory-semantic CXL devices has grown in academia and industry alike.
  - High capacity to cost ratio, NAND reuse, etc...
  - Example of industry interest includes Samsung's CMM-H<sub>[1]</sub>.

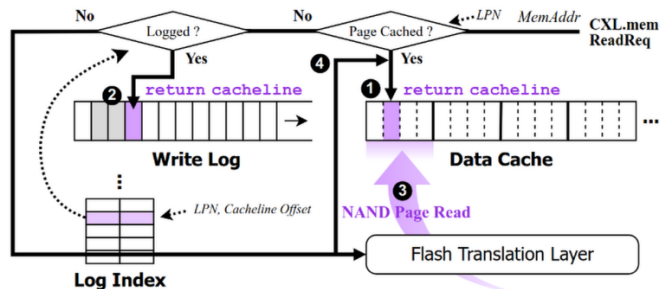


[1]: [Samsung CXL Solutions CMM-H or Memory Module- Hybrid Device](#)

# SkyByte<sub>[1]</sub>: SOTA CXL-SSD architecture



(a) Write operation

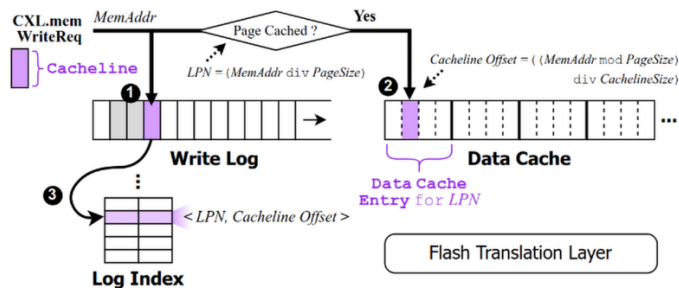


(b) Read operation

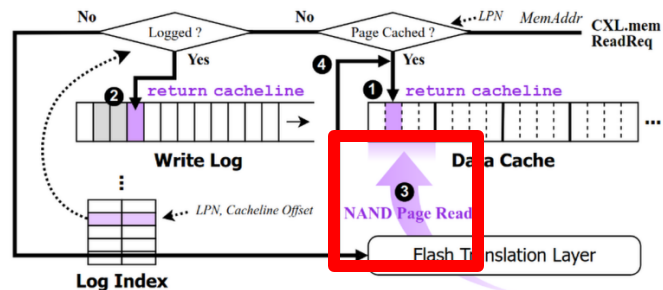
Fig. 2: Write/read flows of the state-of-the-art CXL-SSD [11], comprising a *Write Log*, *Data Cache*, and *Log Index*.

- Proposes a combination of a write log and data cache in SSD DRAM.
- Write log takes incoming write I/O of cacheline sizes (64B).
- Data Cache acts as a NAND page cache to enable faster read I/O.
- Both serve to enable both higher I/O performance as well as memory-semantic I/O.

# SkyByte<sub>[1]</sub>: SOTA CXL-SSD architecture



(a) Write operation



(b) Read operation

Fig. 2: Write/read flows of the state-of-the-art CXL-SSD [11], comprising a *Write Log*, *Data Cache*, and *Log Index*.

- SkyByte also proposes a context-switch policy where a CXL-SSD DRAM miss will trigger a system context switch to cover for the long NAND I/O time.
- SkyByte sets a 2μs threshold for the context switch to be triggered.



# Status quo of current CXL-SSD evaluation



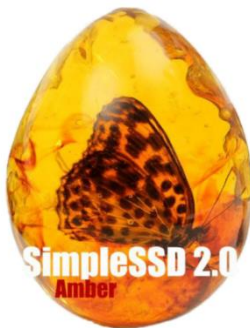
Due to the lack of CXL-SSD hardware, prior works<sub>[1-3]</sub> adopt an evaluation framework consisting of a x86 system simulator and an SSD simulator.

## x86 Simulator<sub>[4]</sub>:



Simulates the host-side CPU and memory operations.

## SSD Simulator<sub>[5]</sub>:



Simulates the CXL-SSD controller ops (write log, data cache, flush, NAND I/O, etc)

- [1]: [SkyByte: Architecting an Efficient Memory-Semantic CXL-based SSD with OS and Hardware Co-design](#), Haoyang Zhang et al. (HPCA 2025)
- [2]: [RomeFS: A CXL-SSD Aware File System Exploiting Synergy of Memory-Block Dual Paths](#), Yekang Zhan et al. (SoCC 2024)
- [3]: [ByteFS: System Support for \(CXL-based\) Memory-Semantic Solid-State Drives](#), Shaobo Li et al. (ASPLOS 2025)
- [4]: [gem5: The gem5 simulator system](#)
- [5]: [Amber: Enabling Precise Full-System Simulation with Detailed Modeling of All SSD Resources](#), Donghyun Gouk et al. (MICRO 2018)

# Status quo of current CXL-SSD evaluation



Due to the lack of CXL-SSD hardware, prior works<sub>[1-3]</sub> adopt an evaluation framework consisting of a x86 system simulator and an SSD simulator.

## x86 Simulator<sub>[4]</sub>:



Simulates the  
host-side CPU

## SSD Simulator<sub>[5]</sub>:



Simulates the CXL-  
SSD controller ops  
(write log, data cache)

→ This hybrid approach was acceptable for early-stage exploration of CXL-SSD concepts.

[1]: ByteFS: System Support for (CXL-based) Memory-Semantic Solid-State Drives, Shaobo Li et al. (ASPLOS 2025)

[4]: gem5: The gem5 simulator system

[5]: Amber: Enabling Precise Full-System Simulation with Detailed Modeling of All SSD Resources, Donghyun Gouk et al. (MICRO 2018)

# Motivation

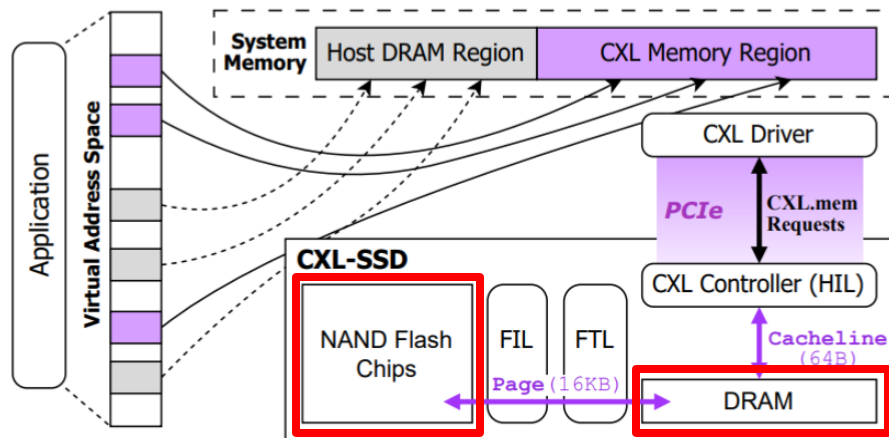
---

# Limitations of status quo evaluation



Moving from storage to memory brings new demands to the evaluation platforms of CXL-SSDs.

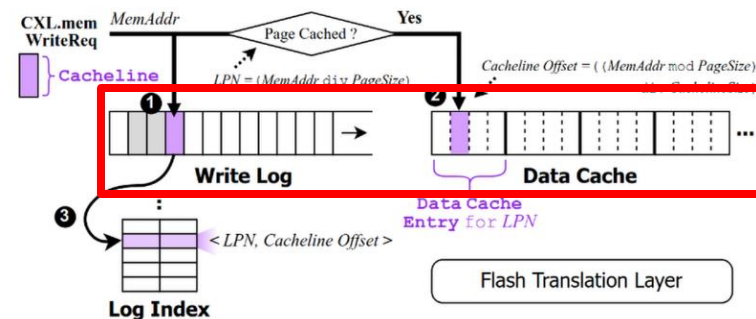
- Reflecting minute latency variations of **hardware components** (DRAM, NAND) become essential for portraying memory-semantic SSDs.



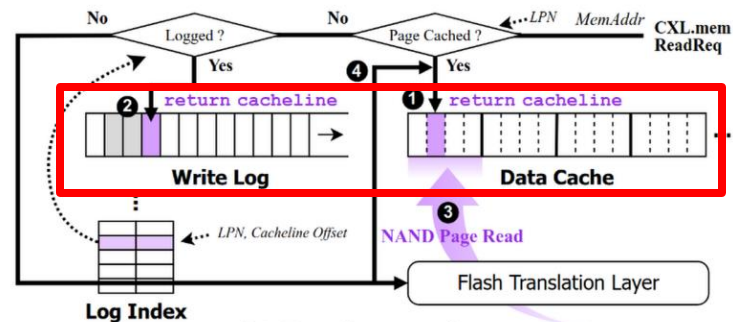
# Limitations of status quo evaluation

Moving from storage to memory brings new demands to the evaluation platforms of CXL-SSDs.

- Evaluation of impact of **new CXL-SSD features** (write log, data cache) in overall performance is also critical.



(a) Write operation



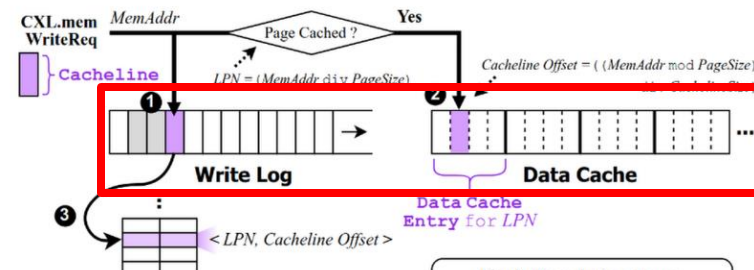
(b) Read operation

# Limitations of status quo evaluation



Moving from storage to memory brings new demands to the evaluation platforms of CXL-SSDs.

- Evaluation of impact of new CXL-SSD features



Such requirements in the evaluation platform are difficult to meet in SSD simulation, which rely on **static parameters**.

→ How does a parameter based SSD simulator compare to a real SSD device in terms of latency evaluation?

(b) Read operation

# Motivation experiment setup



- Comparison of NAND I/O latency between SimpleSSD (SSD simulation) and OpenSSD (real SSD prototype)
- Workloads: randread and randwrite.
  - SimpleSSD: Built-in I/O request generating benchmark based on fio<sub>[1]</sub>
  - OpenSSD: fio<sub>[1]</sub> benchmark.

TABLE I: Specifications of NAND flash modules used.

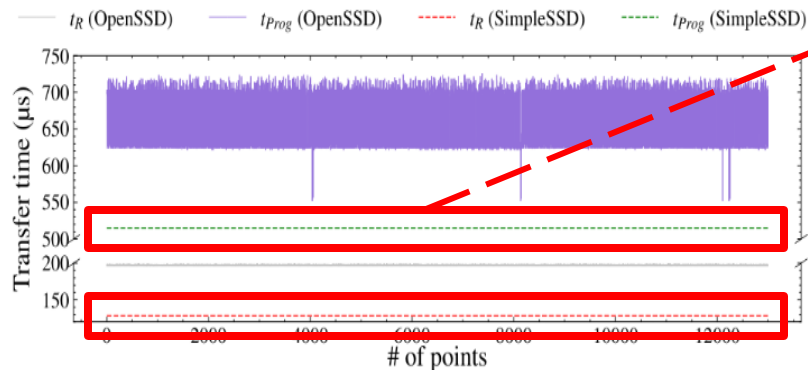
Manufacturer	Capacity	Parallelism Setup	NAND Flash Page Size
(a) SK Hynix	1 TiB	4 Channel, 8 Way	16 KiB
(b) Toshiba	256 GiB		

[1]: [axboe/fio: Flexible I/O Tester](#)

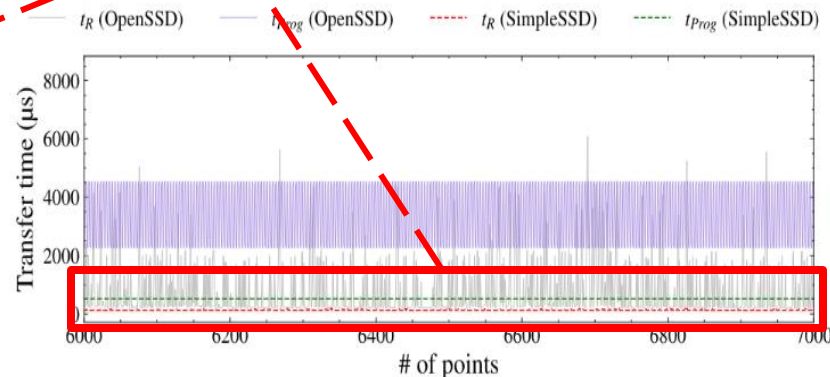
# Limitations of Software-Driven SSD Simulation: SimpleSSD vs OpenSSD



SimpleSSD provides NAND I/O latency as a single parameterized value.



NAND read/program I/O times of vendor (a)  
with iodepth=1.



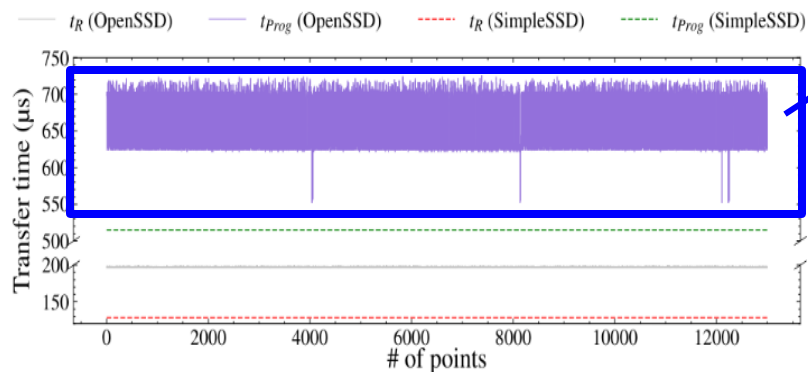
NAND read/program I/O times of vendor (a)  
with iodepth=8.



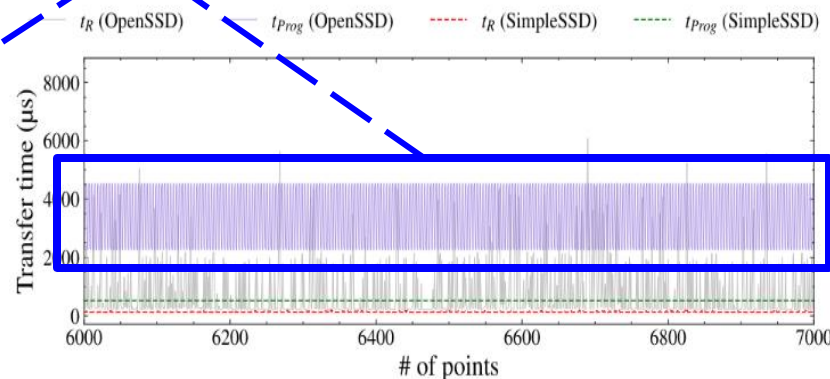
# Limitations of Software-Driven SSD Simulation: SimpleSSD vs OpenSSD



But OpenSSD's real NAND I/O latency shows it cannot be represented as a single unified value.



NAND read/program I/O times of vendor (a)  
with iodepth=1.

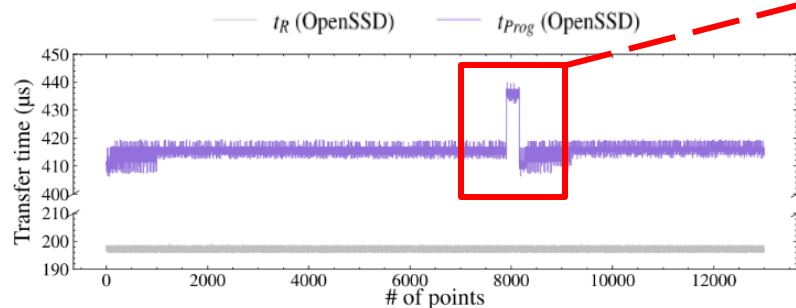


NAND read/program I/O times of vendor (a)  
with iodepth=8.

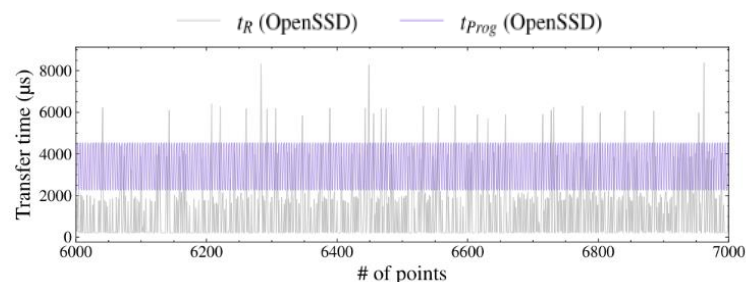
# Limitations of Software-Driven SSD Simulation: SimpleSSD vs OpenSSD



Depending on NAND vendor and workload, unexpected latency spikes can also be observed.



NAND read/program I/O times of vendor (b)  
with iodepth=1.



NAND read/program I/O times of vendor (b)  
with iodepth=8.

# Limitations of Software-Driven SSD Simulation: NAND Access Latency Breakdown



- Growing impact of the low-level NAND flash controller and firmware can be seen with more stressing workloads

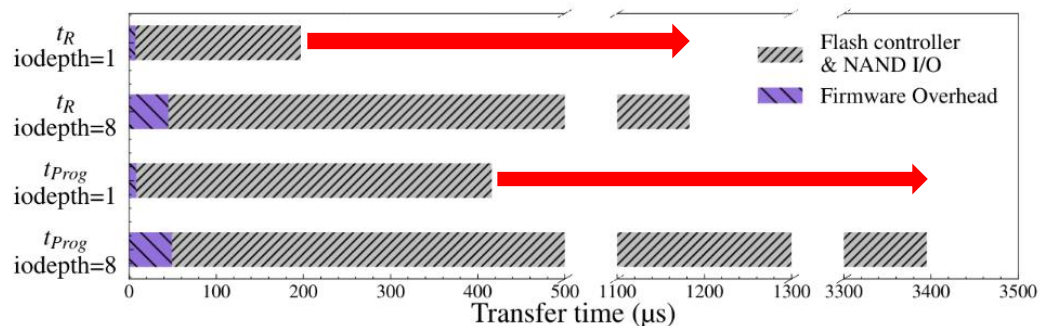


Fig. 5: Breakdown of NAND (b)'s average  $t_R$  and  $t_{Prog}$ .

# Limitations of Software-Driven SSD Simulation: NAND Access Latency Pattern CDF



- CDF shows NAND of similar specifications from different vendors show **differing latency characteristics that cannot be represented by a single fixed value.**

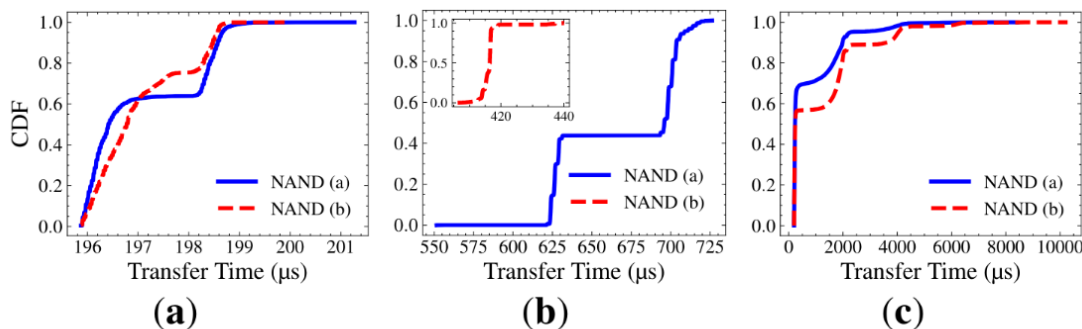


Fig. 6: NAND I/O latency Cumulative Distribution Function (CDF) of two different types of NAND in different workloads (a) randread, iodepth=1, (b) randwrite, iodepth=1, (c) randread, iodepth=8.

# Takeaway from motivation experiments



- NAND I/O latency is highly variant based on factors outside NAND specifications → A single value covering latency is not enough!
  - Even NAND with similar specifications between different vendors show differing performance characteristics.
- Such approximations are adequate for **average** performance metrics for storage devices but is lacking in portraying **per-request** performance required for **memory** devices.

**How can we improve evaluation platforms to more accurately portray CXL-SSD performance on a memory-semantic level?**

# **Proposed Solution: *OpenCXD***

---

# Overview of *OpenCxD*



- **x86 simulator (MacSim<sub>[1]</sub>)**
  - Provides cycle-accurate simulation of the entire memory hierarchy.
  - Replays memory traces from the target workload.
- **SSD platform (OpenSSD<sub>[2]</sub>)**
  - Real SSD hardware and configurable firmware. (NVMe interface, FTL, and NAND I/O scheduling)
  - Current implementation replicates the internal architecture of SkyByte.

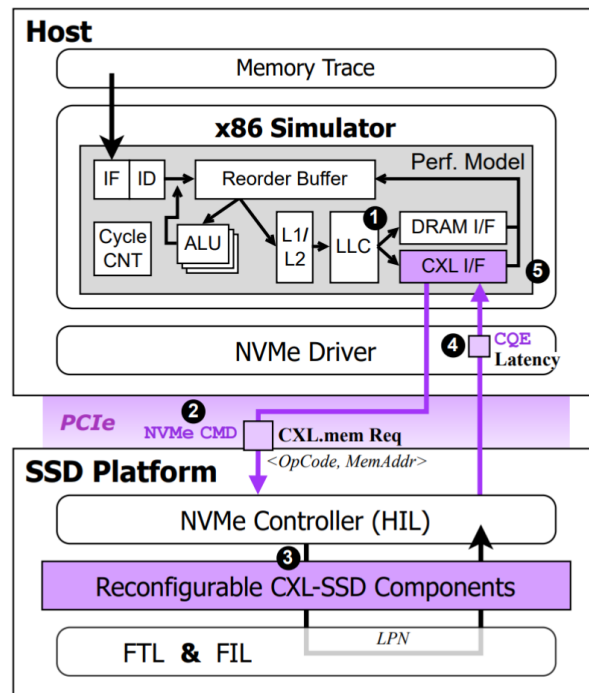


Fig. 7: Architecture of OPENCxD and its execution flow.

[1]: [gthparch/macsim: A heterogeneous architecture timing model simulator](#).



[2]: [The OpenSSD Project](#)

# Custom NVMe commands in *OpenCXD*



dword	description				
0	CID	P	Reserved	F	opcode
1	Namespace ID				
2	Reserved				
3					
4					
5	Metadata Pointer				
6	PRPentry#1 ( <i>not used</i> )				
7					
8	PRPentry#2 ( <i>not used</i> )				
9					
10	Reserved				
11	<b>CXL Memory Address</b>				
12	<b>Load or Store</b>				
13	Reserved				
14					
15					

(a) NVMe Command (SQE)

 NVMe Standard  
 OpenCXD Extension

※ *Device Op Latency = NAND  
 I/O Latency + Op Overhead*

dword	description				
0			R	SCT	Op Overhead R
1	<b>Device Op Latency</b>				
2	SQ ID		CQ ID		
3	Status Field		P	CID	

(b) NVMe CQE

Fig. 8: NVMe command and CQE for OPENCXD.

- Submission Queue Entry (SQE) includes the memory operation type (load/store) and memory address of said operation .
- Completion Queue Entry (CQE) returns operation latency and overhead values that is required for simulation integration.



# Execution Flow of *OpenCXD*

- ① Memory access detection
  - LLC cache miss detection - Checks whether the missing address falls within the memory-mapped region of the CXL-SSD.

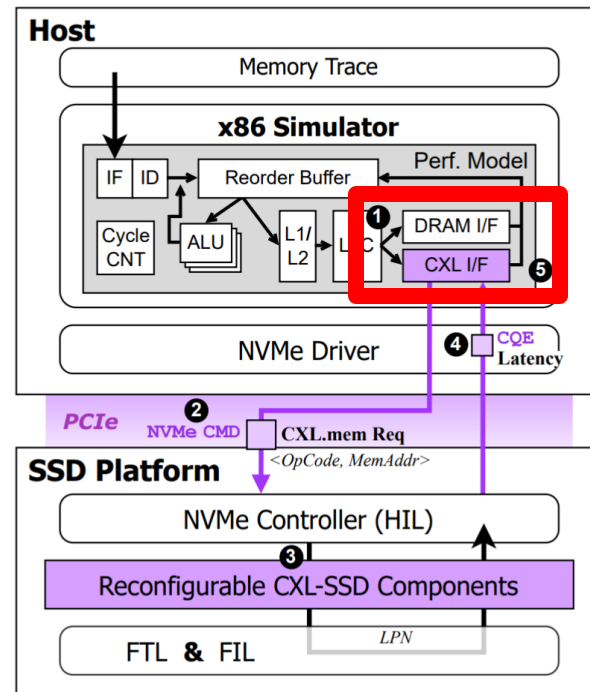


Fig. 7: Architecture of OPENCXD and its execution flow.

# Execution Flow of *OpenCxD*

## ② Issue custom NVMe command

- Encapsulates memory request into a newly defined NVMe command for OpenCxD.
- The NVMe command is issued to the SSD platform via the NVMe passthrough interface.

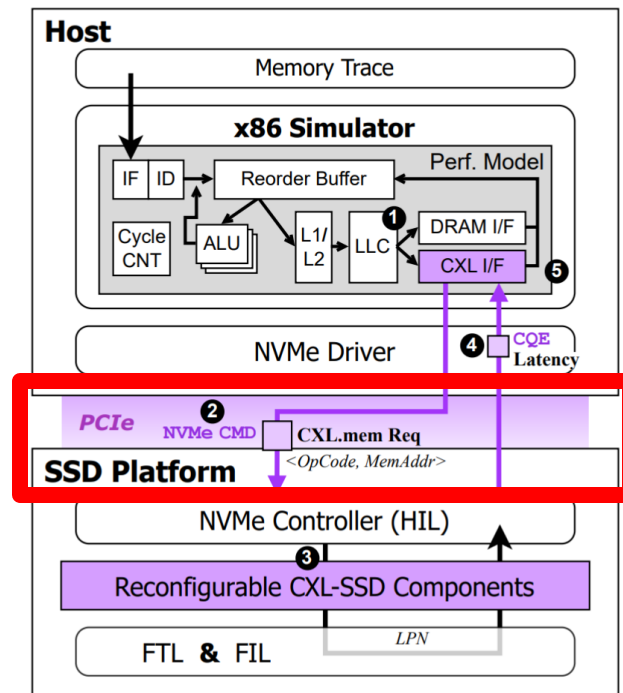


Fig. 7: Architecture of OPENCxD and its execution flow.

# Execution Flow of *OpenCxD*

## ③ Perform CXL-SSD

operations

- Fetches the NVMe command.
- Performs the corresponding CXL-SSD operations (e.g., Write Logging).

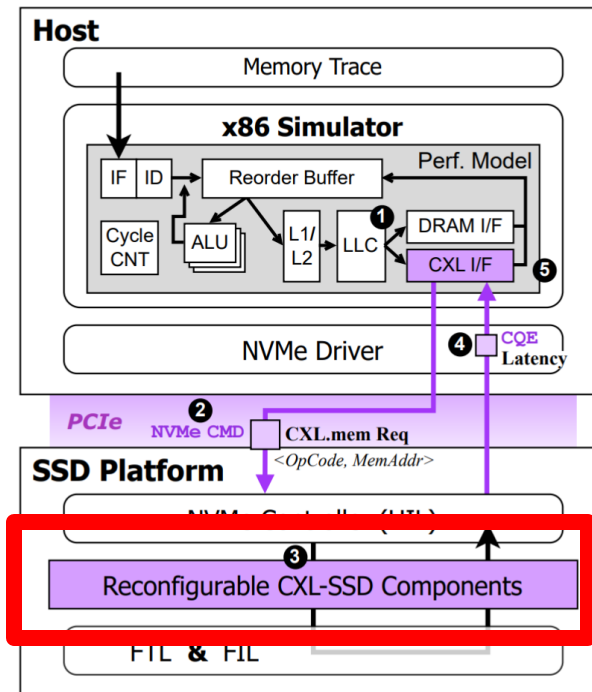


Fig. 7: Architecture of OPENCxD and its execution flow.

# Execution Flow of *OpenCxD*



- ④ Return timing information
- Returns the total processing time in device to the host by using reserved field of the NVMe Completion Queue Entry (CQE).

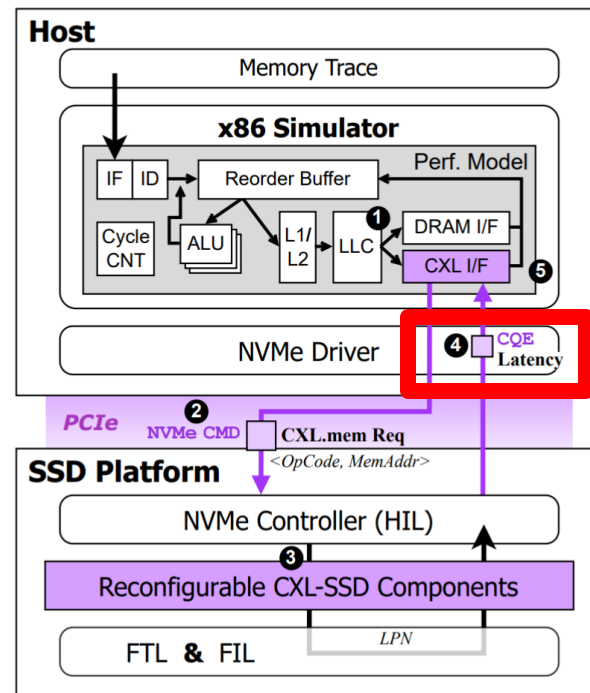


Fig. 7: Architecture of OPENCxD and its execution flow.

# Execution Flow of *OpenCXD*



- ⑤ Integration with simulation
- Extracts the device-side latency from the received CQE.
  - Adds the CXL.mem protocol overhead, which is recognized as consistent and reasonable<sub>[1]</sub>, as used in SkyByte (~40ns).
  - Method allows the NVMe overhead to be removed from the simulation equation.

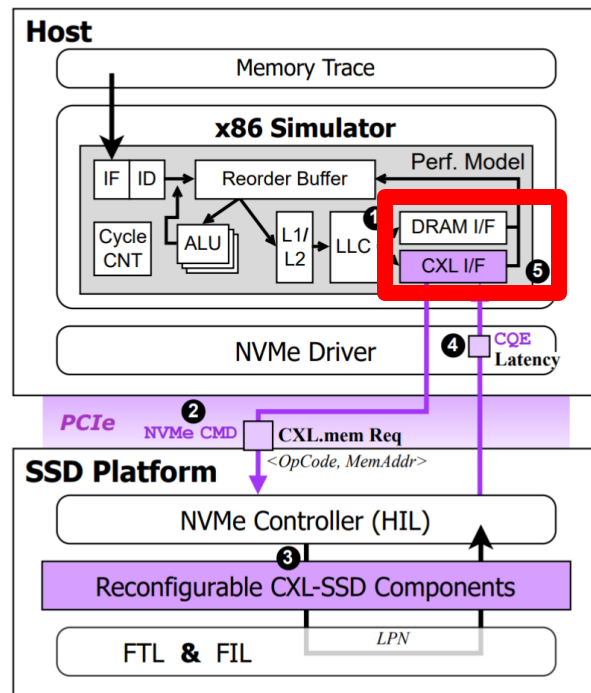


Fig. 7: Architecture of OPENCXD and its execution flow.

# Timing Flow integration in *OpenCxD*

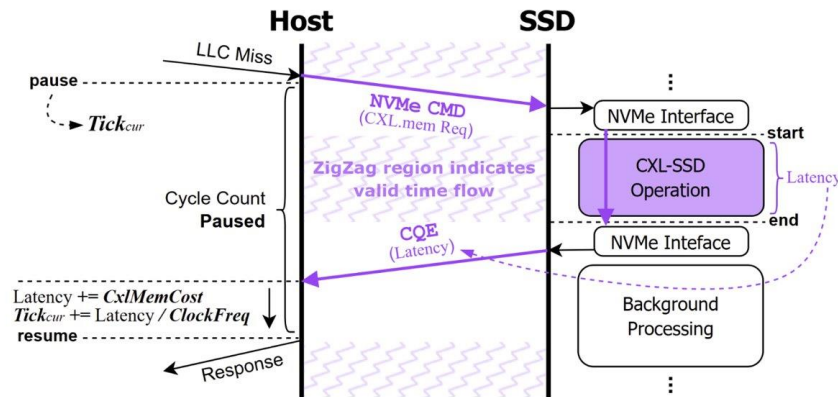


Fig. 8: Timing flow of a CXL memory request in OpenCxD.

- 1) Pause the x86 simulator during each CXL.mem access.
  - 2) Incorporate latency measured from firmware execution on the SSD.
- This implementation allows modeling of CXL.mem timing behavior while removing NVMe transport overhead from equation.

# Evaluation

---

# Evaluation Setup

- Testbed:  
**DaisyPlus  
OpenSSD  
Platform**<sup>[1]</sup>



TABLE II: Specifications of the OpenSSD platform.

SoC	Xilinx Zynq UltraScale+ ZU17EG, with ARM Cortex-A53 Core
NAND Module	256GB, 4 Channel & 8 Way
Interconnect	PCIe Gen3 × 16 End-Points
DRAM	2GB LPDDR4 @ 2400MHz

TABLE III: Specifications of the host system.

CPU	Intel(R) Core(TM) i7-14700K CPU @ 5.60GHz (28 cores)
Memory	32GB DDR5
OS	Ubuntu 24.04.2, Linux Kernel 6.11.0

[1]: DaisyPlus: <https://www.crz-tech.com/crz/article/DaisyPlus/>



# Evaluation Setup



- Workloads: 7 benchmarks (bc, bfs-dense, dlrn, radix, srar, tpcc, ycsb), traces using 8 cores with 3 threads each, taken from SkyByte's artifact.

**TABLE I:** Benchmarks used in our experiments.

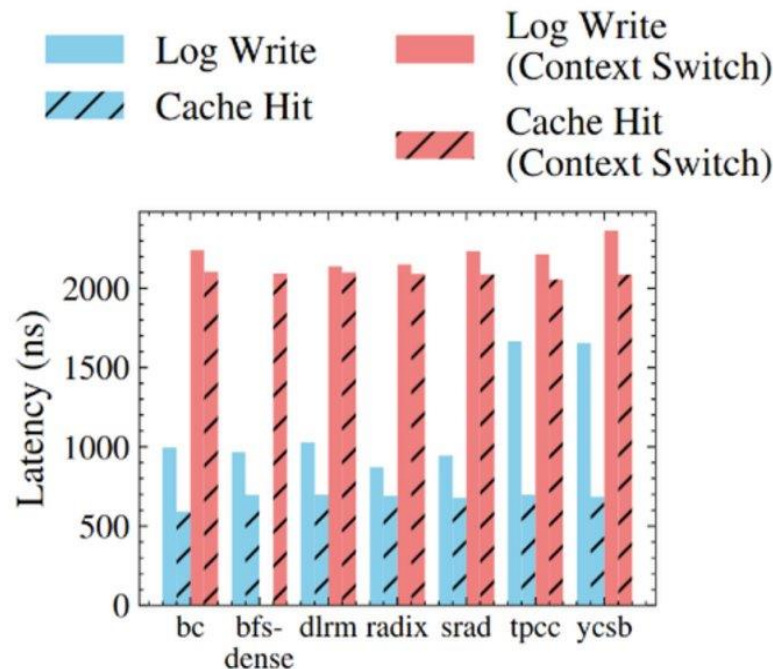
Category	Suite	Name	Memory Footprint	Write Ratio	LLC MPKI
Graph Processing	Rodinia [17]	bfs-dense	9.13GB	25%	122.9
	GAP [13]	bc	8.18GB	11%	39.4
HPC	Splashv3 [51]	radix	9.60GB	29%	7.1
Image Processing	Rodinia [17]	srar	8.16GB	24%	7.5
Database	WHISPER [45]	ycsb	9.61GB	5.0%	92.2
		tpcc	15.77GB	36%	1.0
Machine Learning	DLRN [46]	dlrn	12.35GB	32%	5.1

- Results from **OpenCXD** (evaluation integrating real SSD hardware) are compared with results from SkyByte artifact (evaluation using SSD simulation).

# Evaluation: CXL-SSD operation latency



- ***OpenCXL*** shows varying results in latency in all workloads.
  - SkyByte's log write and cache hit always seen to be 712ns and 640ns respectively.

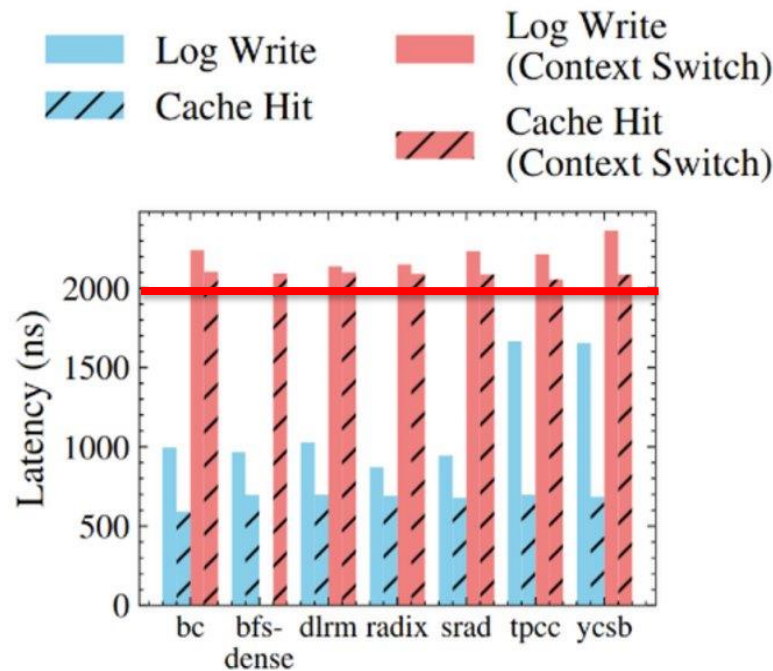


(a)

# Evaluation: CXL-SSD operation latency



- Some cache hit latencies even goes over the  $2\mu\text{s}$  threshold set in SkyByte, showing the effects of DRAM latency spikes.

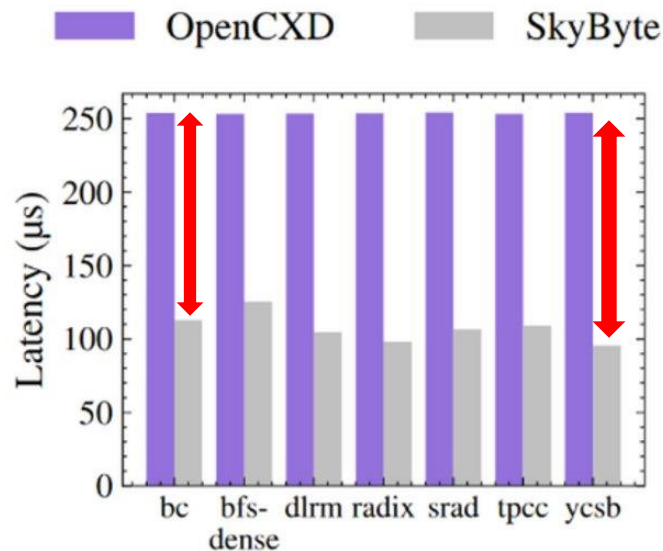


(a)

# Evaluation: Cache miss latencies



- Due to taking NAND controller and firmware overhead into consideration, **OpenCXD** shows a 2.4x higher average latency across all benchmarks over SkyByte.

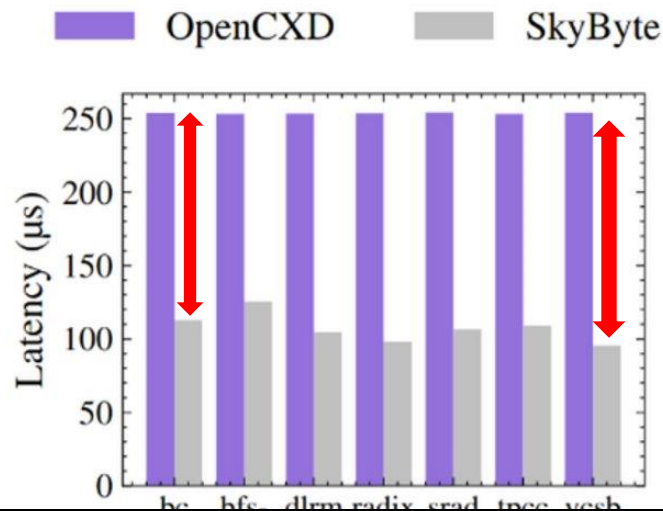


(b)

# Evaluation: Cache miss latencies



- Due to taking NAND controller and firmware overhead into consideration, **OpenCXD** shows a 2.4x higher average latency across all benchmarks over SkyByte.



→ **OpenCXD** reveals higher sensitivity to NAND performance and DRAM latency spikes, which differ based on each workload.

# Evaluation: Latency spread analysis



Histograms clearly show the skewed latency distribution in SkyByte.

- 87.2% of requests in `srad` and 94.3% in `ycsb` return the same fixed latency value of 99.72  $\mu$ s.

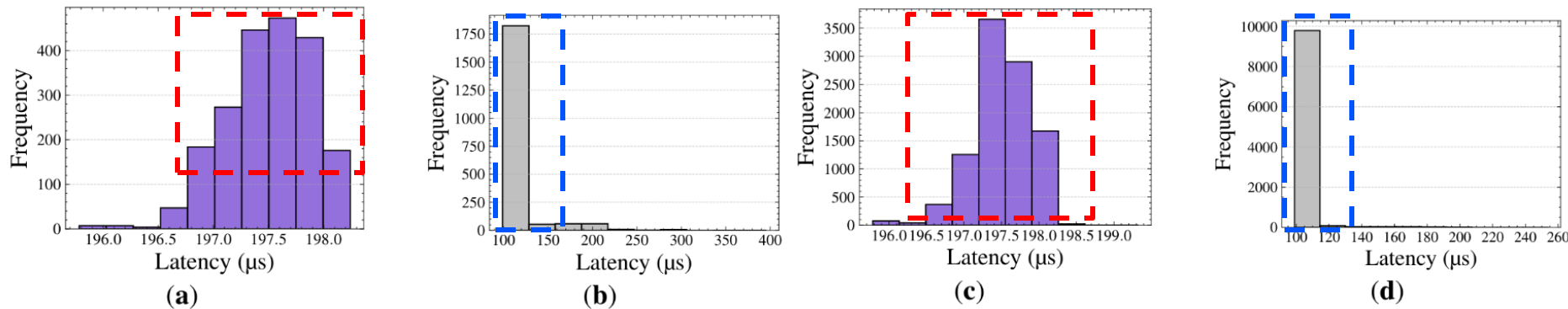


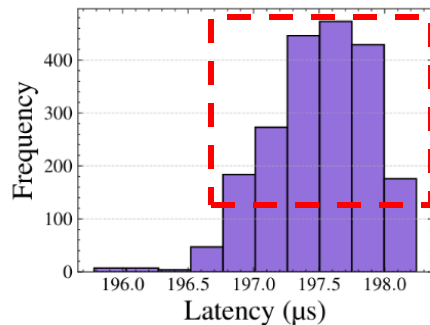
Fig. 11: Histograms of NAND read I/O latency during hit misses for `srad` ((a): OPENCXD, (b): SkyByte) and `ycsb` ((c): OPENCXD, (d): SkyByte).

# Evaluation: Latency spread analysis

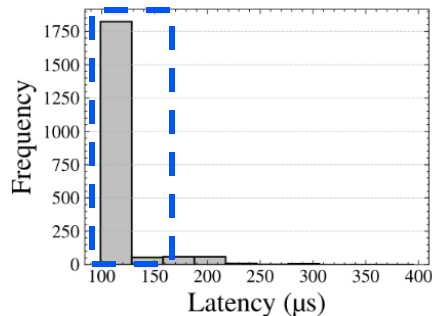


Histograms clearly show the skewed latency distribution in SkyByte.

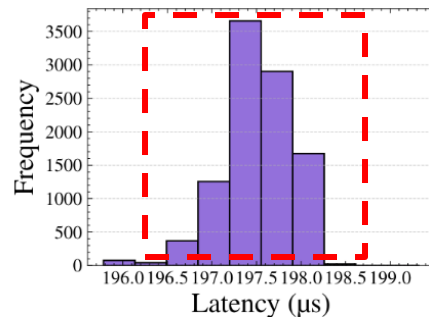
- 87.2% of requests in srad and 94.3% in ycsb return the same fixed latency value of 99.72  $\mu$ s.



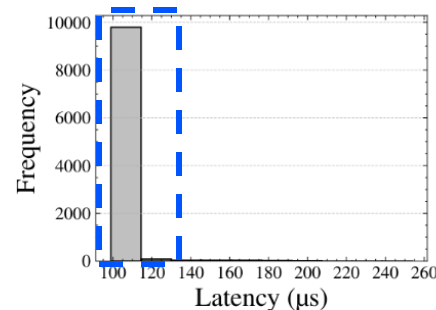
(a)



(b)



(c)



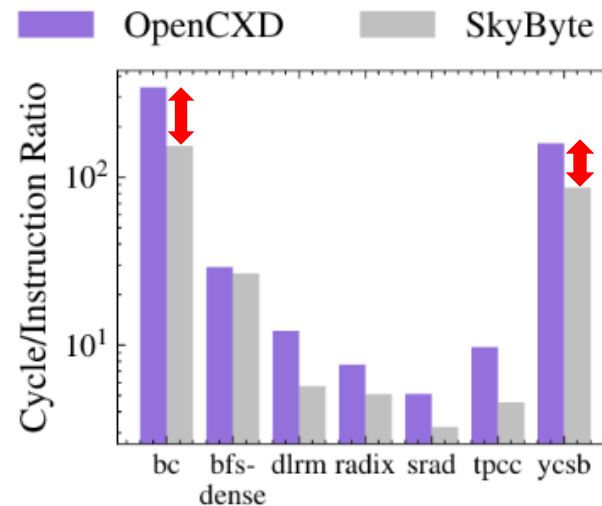
(d)

→ **OpenCXD** enables a more accurate study of CXL-SSD performance through a varied spread of latencies.

# Evaluation: CPU Cycles Comparison



- **OpenCXD** overall required more CPU cycles required to process one million memory accesses over SkyByte.
  - Result seen as OpenCXD reflects overall higher I/O latencies than SkyByte.
  - More threads are required to hide read latency through context switching in OpenCXD.
  - Results highlights the need for additional optimizations to effectively hide NAND I/O latency.

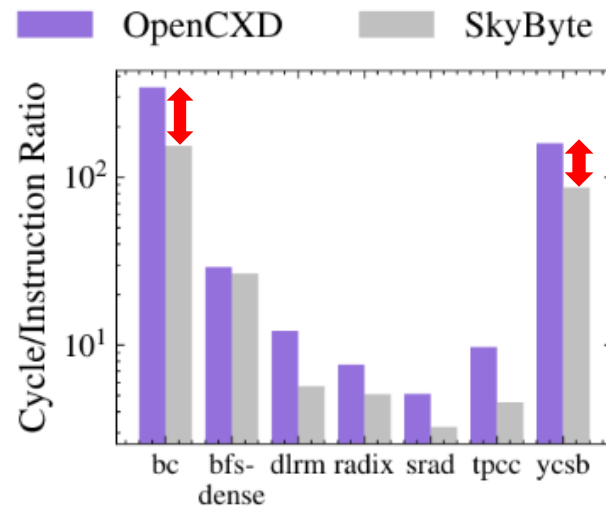




# Evaluation: CPU Cycles Comparison



- **OpenCXD** overall required more CPU cycles required to process one million memory accesses over SkyByte.
  - Result seen as OpenCXD reflects overall higher I/O latencies than SkyByte.
  - More threads are required to hide read latency through context switching in OpenCXD.
  - Results highlights the need for



→ Evaluations from **OpenCXD** reveal new work for CXL-SSD optimizations over the current state of the art.

# Conclusion

---

# Conclusion



- SSD simulation in an evaluation platform for CXL-SSDs is inadequate
  - Current SSD simulators are focused for storage devices, not memory-semantic devices.
- **OpenCxD** proposes a new evaluation method that incorporates real SSD hardware with CXL-SSD firmware in simulation results.
- **OpenCxD** demonstrates the effectiveness of such integration of real hardware in simulations.
  - More nuanced SSD and DRAM latency incorporation
  - Critical insights into CXL-SSD optimization requirements

# Thank you!

## • Contact

- Hyunsun Chung / [hchung1652@sogang.ac.kr](mailto:hchung1652@sogang.ac.kr)
- Junhyeok Park / [junttang@sogang.ac.kr](mailto:junttang@sogang.ac.kr)
- Youngjae Kim / [youkim@sogang.ac.kr](mailto:youkim@sogang.ac.kr)
- Data-Intensive AI Computing & Systems Laboratory <https://discos.sogang.ac.kr/>

<Camera-ready paper>  
Can be found on Google Scholar



## OPENCXD: An Open Real-Device-Guided Hybrid Evaluation Framework for CXL-SSDs

Hyunsun Chung<sup>1,\*</sup>, Junhyeok Park<sup>1,\*</sup>, Taewan Noh<sup>1</sup>, Seonghoon Ahn<sup>1</sup>  
Kilwan Kim<sup>1</sup>, Ming Zhuo<sup>2</sup>, Youngjae Kim<sup>1,†</sup>

<sup>1</sup>Sogang University, Seoul, Republic of Korea, <sup>2</sup>Arizona State University, Tempe, AZ, USA

**Abstract**—The advent of Compute Express Link (CXL) enables SSDs to participate in the memory hierarchy as large-capacity, byte-addressable memory devices. These CXL-enabled SSDs (CXL-SSDs) offer a promising new tier between DRAM and traditional storage, combining NAND flash density with memory-like access semantics. However, evaluating the performance of CXL-SSDs remains difficult due to the lack of hardware that natively supports the CXL mem protocol on SSDs. As a result, most prior work relies on hybrid simulators combining CPU models emulating the CXL mem semantics and SSD simulators that approximate internal flash behaviors. While effective for early-stage exploration, this approach cannot faithfully model firmware-level interactions and low-level storage dynamics critical to CXL-SSD performance. In this paper, we present OPENCXD, a real-device-guided hybrid evaluation framework that bridges the gap between simulation and hardware. OPENCXD integrates a cycle-accurate CXL mem simulator on the host side with a physical OpenSSD platform running real firmware. This enables fine-grained firmware execution triggered by simulated memory requests. Through these contributions, OPENCXD reflects device-level phenomena unobservable in simulation-only setups, providing critical insights for future hardware design tailored for CXL-SSDs.

**Index Terms**—Compute Express Link, Solid-State Drive, overhead and access latency while enabling a unified, tiered memory architecture that extends beyond DRAM's capacity limits [11]. Building on this capability, a new class of memory-sensitive devices, CXL-enabled SSDs (CXL-SSDs), has begun to emerge [10]. These devices leverage mature NAND flash technology to offer terabytes of byte-addressable capacity at a fraction of DRAM's cost per gigabyte [7], while maintaining access latencies in the microsecond range. Although slower than DRAM by several orders of magnitude, CXL-SSDs provide significantly lower latency than traditional SSDs accessed via the block interface (e.g., NVMe [12]).

The key challenge for CXL-SSDs is how to architect and evaluate these devices effectively, maximizing their performance potential while addressing inherent latency trade-offs. However, this evaluation remains difficult in practice, due to the lack of hardware that natively supports the CXL mem protocol on SSDs. To overcome this, recent research has adopted software-based hybrid simulators, combining CPU simulators (e.g., MacSim [13], GenS [14]) extended with CXL mem semantics and SSD simulators (e.g., SimpleSSD [15], FlashSim [16]) that model internal flash behaviors such as address translation and I/O scheduling. This methodology enables early-stage design exploration and has been widely used in prior work [17]–[19].

Notably, the CXL mem interface overhead itself has been characterized in prior studies [20], [21], and shown to be relatively consistent and bounded [10]. As such, injecting this CXL interface time overhead into the simulation as a parameter is generally considered a reasonable approach for modeling host-side access costs. However, the same cannot be said for modeling device-side behavior. Replacing a real SSD with a simulator introduces significant challenges in capturing the full complexity of CXL-SSD-specific firmware logic and storage interactions, which are limitations that fundamentally hinder accurate evaluation of CXL-SSD designs [10].

There are two key limitations of simulation-only evaluation. First, since CXL-SSDs function as memory rather than storage, they must handle fine-grained, cache-line-level memory accesses, making them highly sensitive to device-side performance fluctuations. However, simulators typically rely on static latency models, overlooking dynamic behaviors such as real-time NAND latency variability and firmware delays, resulting in latency estimation errors as high as 36% [22]. Second, CXL-SSDs introduce new firmware-managed mechanisms, such as write logging and log compaction [11], that do not exist in conventional SSDs. Simulating these new

\*They are first co-authors and have contributed equally.

†Y. Kim is the corresponding author.