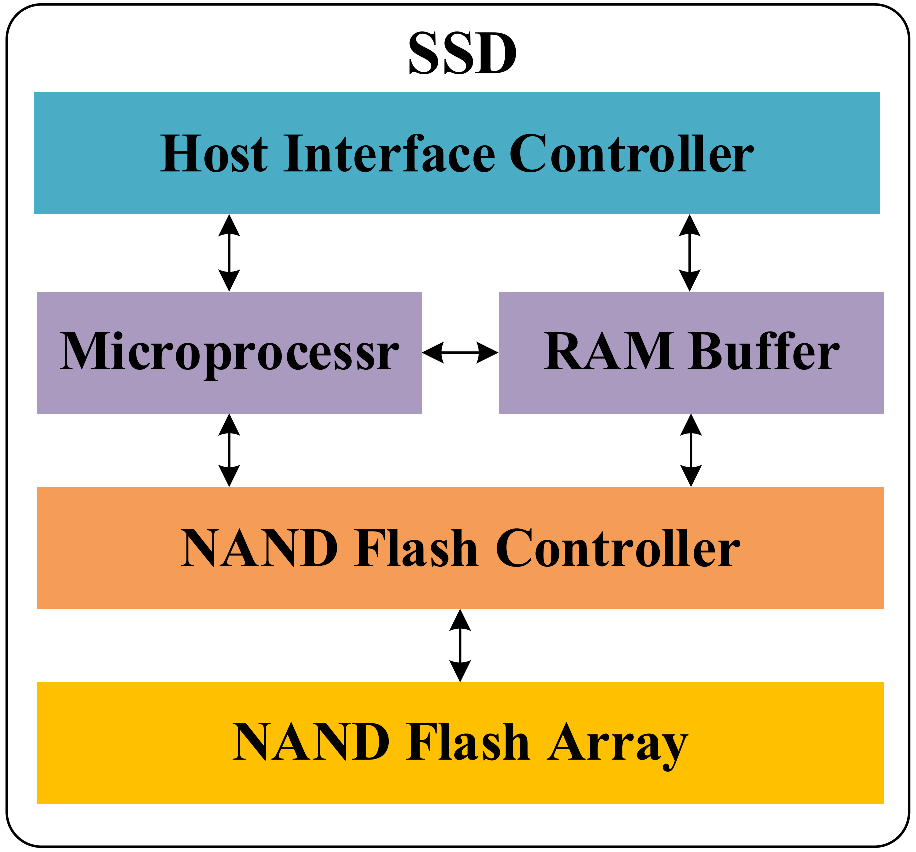
A High-performance Open-channel Open-way NAND Flash Controller Architecture

Yunhui Qiu, Wenbo Yin, Lingli Wang
State Key Laboratory of ASIC and System
Fudan University, Shanghai, China
FPL’21 2021-09-01

NAND-Flash-based SSD

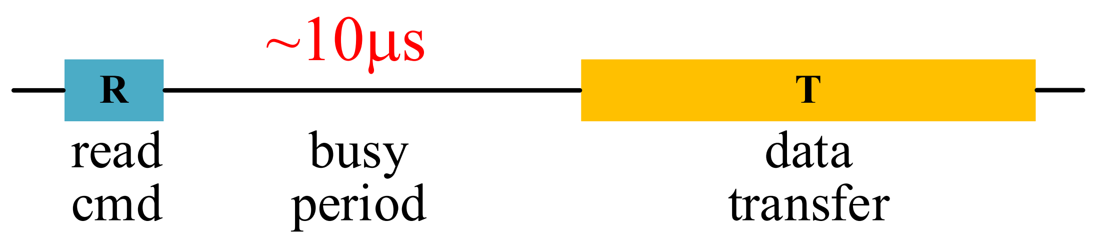


A typical SSD architecture

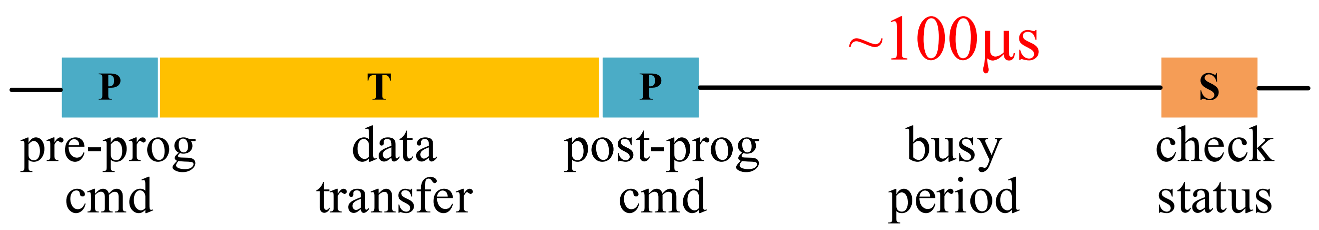
- **HIC**: connect to the host via PCIe, SATA…
- **MCU**: implement FTL to hide the peculiarities of NAND Flash
- **RAM**: cache data to fill the performance gap between the host interface and Flash media
- **NFC**: execute high-level commands according to the I/O protocol of Flash chips
- **NFA**: multi-channel multi-way Flash chips

*FTL: Flash Translation Level

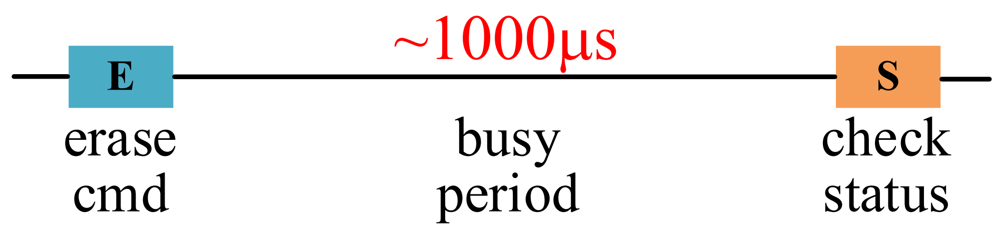
Flash Operations



(a) read page



(b) program page



(c) erase block

Challenges

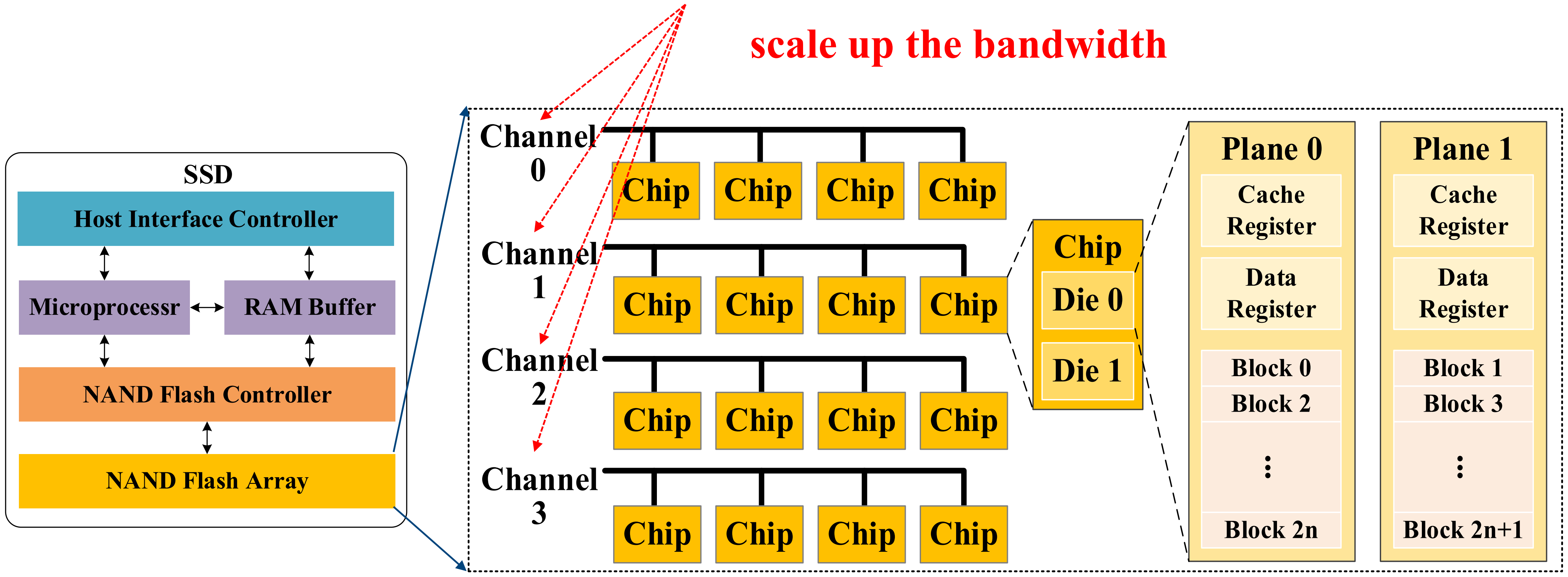
- **I/O speed:** ~10MT/s - ~1000MT/s, lower than the host interface
- **Slow execution**: serious under-utilization of I/O bandwidth

Multi-level Parallelism

- **Channel-level parallelism**

multiple channels with independent I/O buses

scale up the bandwidth

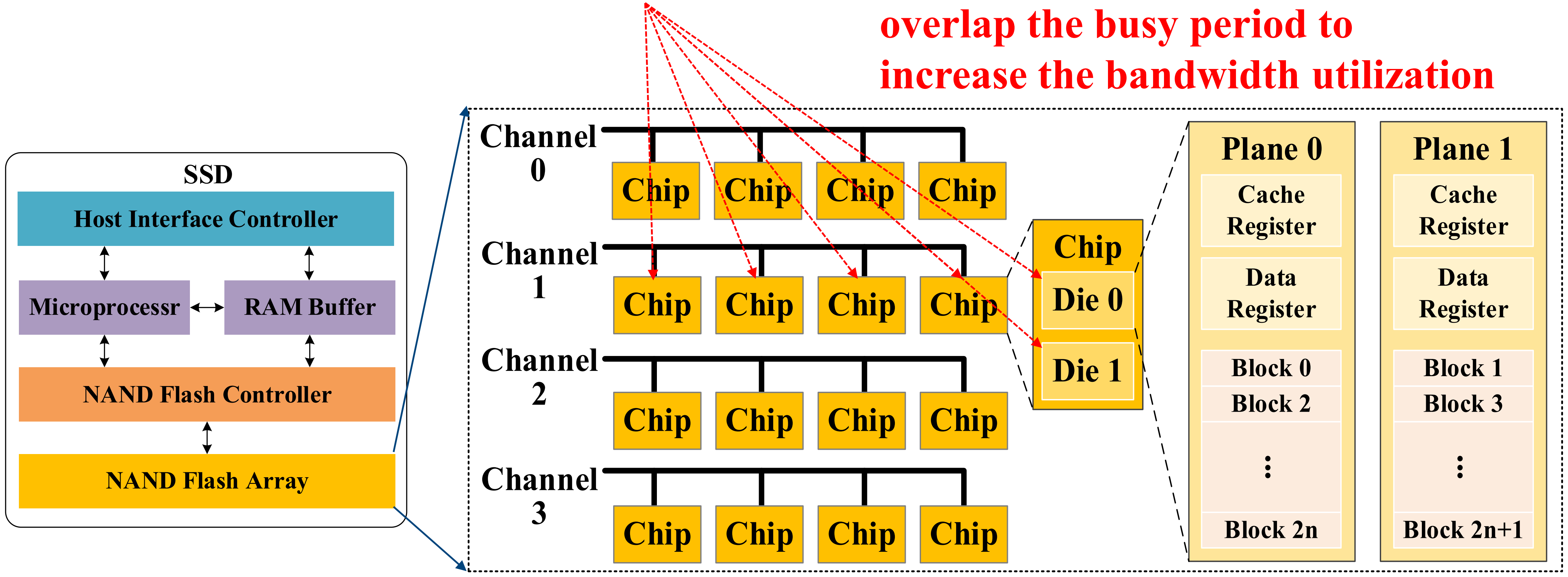


Multi-level Parallelism

- **Channel-level parallelism:** independent I/O buses, scale up the bandwidth
- **Way-level interleaving**

multiple ways (chips or dies) share an I/O bus

overlap the busy period to increase the bandwidth utilization

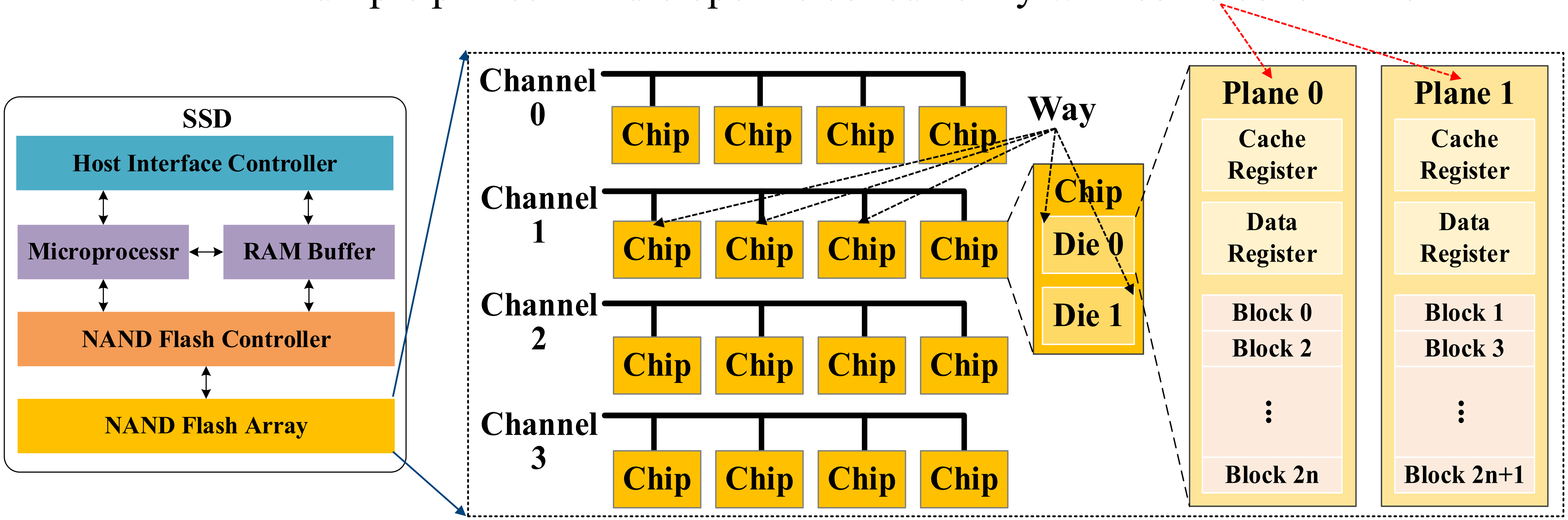


Multi-level Parallelism

- **Channel-level parallelism:** independent I/O buses, scale up the bandwidth
- **Way-level interleaving:** share an I/O bus, increase the bandwidth utilization
- **Plane-level interleaving**

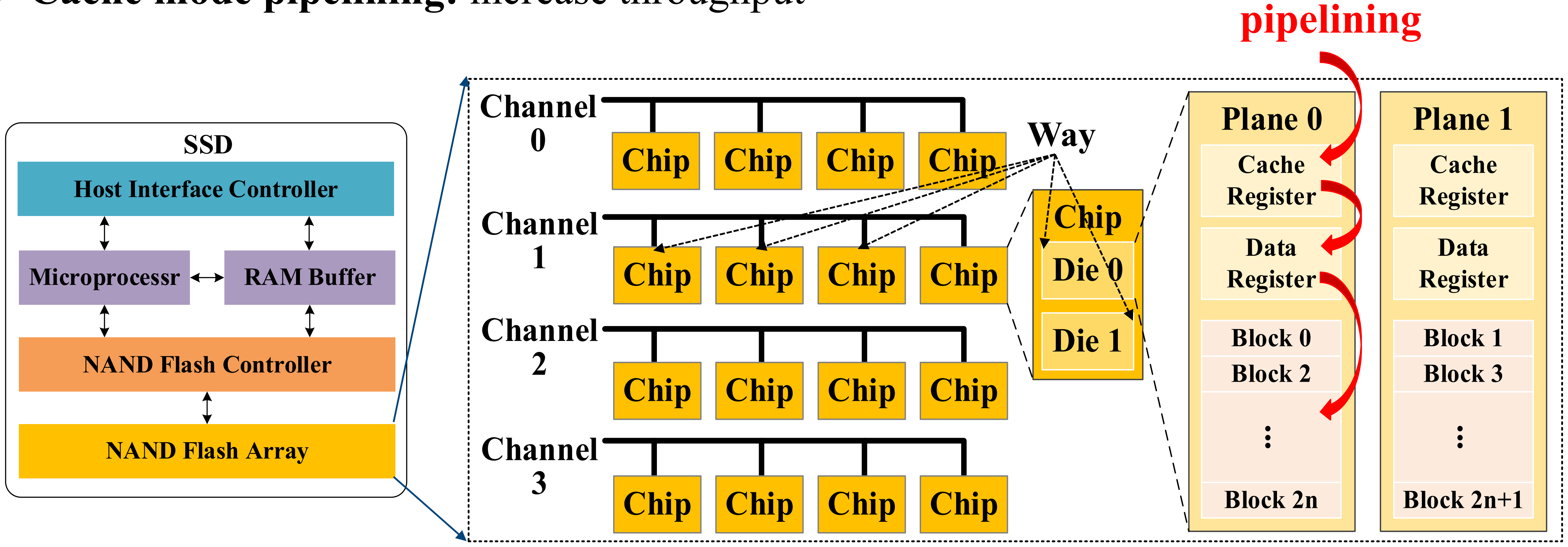
increase the bandwidth utilization

multiple planes in a die operate concurrently with some constraints



Multi-level Parallelism

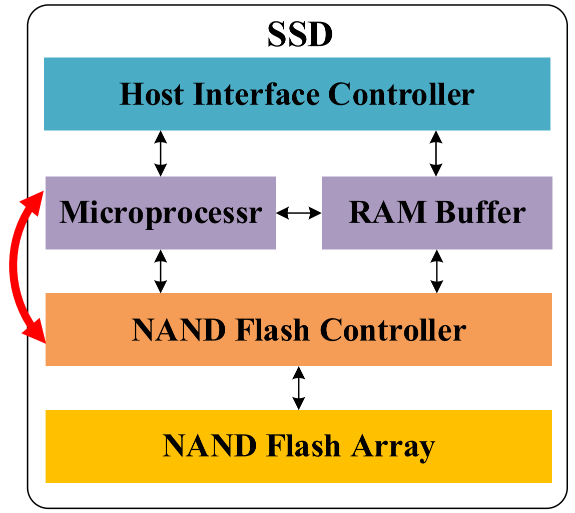
- **Channel-level parallelism:** independent I/O buses, scale up the bandwidth
- **Way-level interleaving:** share an I/O bus, increase the bandwidth utilization
- **Plane-level interleaving:** operate concurrently, increase the bandwidth utilization
- **Cache mode pipelining:** increase throughput



Motivation: exploit the multi-level parallelism

Related Work

1. **Flash simulators** [11], [13]-[18]
   - ☐ fast prototyping and verification
   - ☐ hard to accurately model hardware controllers
   - ☐ cannot be evaluated, verified, or tested in a practical environment
2. **Firmware-based I/O scheduling** [5], [8], [19]
   - ☐ tight coupling between FTL and Flash controller
   - ☐ frequent and time-consuming software-hardware interactions impose serious performance overhead
3. **Hardware-based I/O scheduling** [6], [12], [20]-[22]
   - ☐ auto-scheduling in Flash controller
   - ☐ out-of-order execution
   - ☐ not cover all levels of parallelism



Contributions

An open-source high-performance open-channel open-way Flash controller supporting **channel-way-plane** levels of parallelism and **cache mode** pipelining.

1. **open-way architecture**
   - ☐ expose multi-channel multi-way Flash topology to upper-level module
   - ☐ queue-based asynchronous interface for each way
2. **dual-level hardware scheduler**
   - ☐ increase multi-plane and cache mode operations
   - ☐ auto-interleave commands in fine granularity
3. **Flash command classification**
   - ☐ four groups: checking status, reading data, writing data, and others.
   - ☐ one FSM for each group, rather than each command

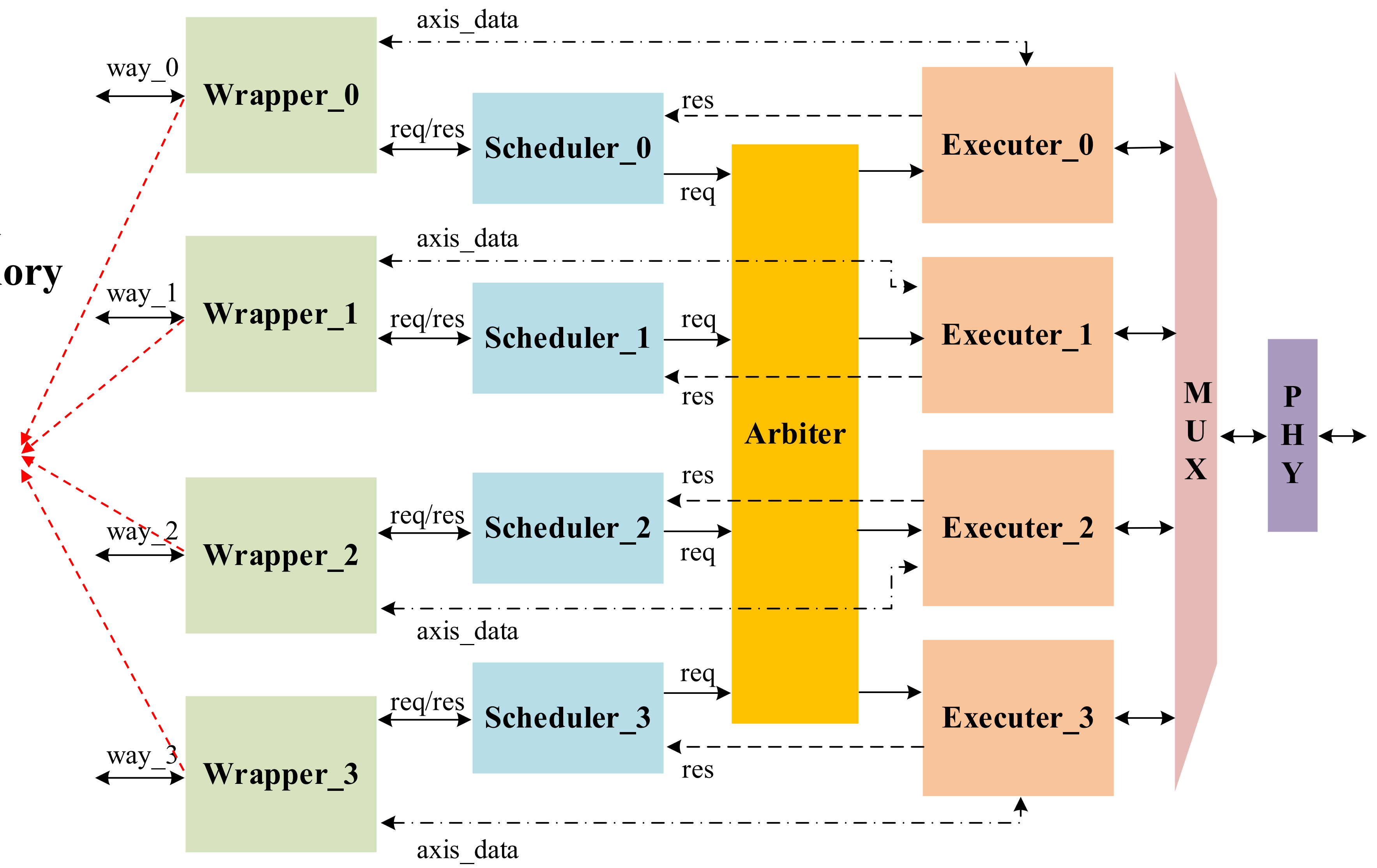
*open-source RTL codes: https://github.com/yhqiu16/OCOWFC

Flash Controller Design

Flash Channel Controller

Channel controller with four ways of Flash memory

Open-way: expose a queue-based interface for each way



Flash Command Classification

Flash commands

Reset Operations
RESET
SYNCHRONOUS RE-SET
RESET LUN
Identification Operations
READ ID
READ PARAMETERPAGE
READ UNIQUE ID
Configuration Operations
VOLUME SELECT
ODT CONFIGURE
GET FEATURES
SET FEATURES
GET FEATURES byLUN
SET FEATURES byLUN
Column Address Operations
CHANGE READ COL-UMN
CHANGE READ COL-UMN ENHANCED (ONFI)
CHANGE READ COL-UMN ENHANCED (JEDEC)
CHANGE WRITECOLUMN
CHANGE ROW AD-DRESS

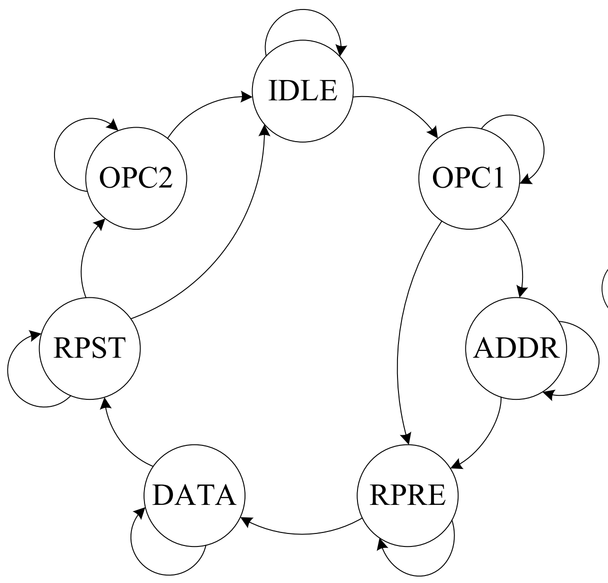
Status Operations
READ STATUS
READ STATUS EN-HANCED
Read Operations
READ MODE
READ PAGE
READ PAGE MULTI-PLANE
READ PAGE CACHESEQUENTIAL
READ PAGE CACHERANDOM
READ PAGE CACHELAST
Program Operations
PROGRAM PAGE
PROGRAM PAGEMULTI-PLANE
PROGRAM PAGECACHE
Erase Operations
ERASE BLOCK
ERASE BLOCK MULTI-PLANE (ON-FI)
ERASE BLOCK MUL-TI-PLANE (JEDEC)
Copyback Operations
COPYBACK READ
COPYBACK PRO-GRAM
COPYBACK PRO-GRAM MULTI- PLANE

Flash Command Classification

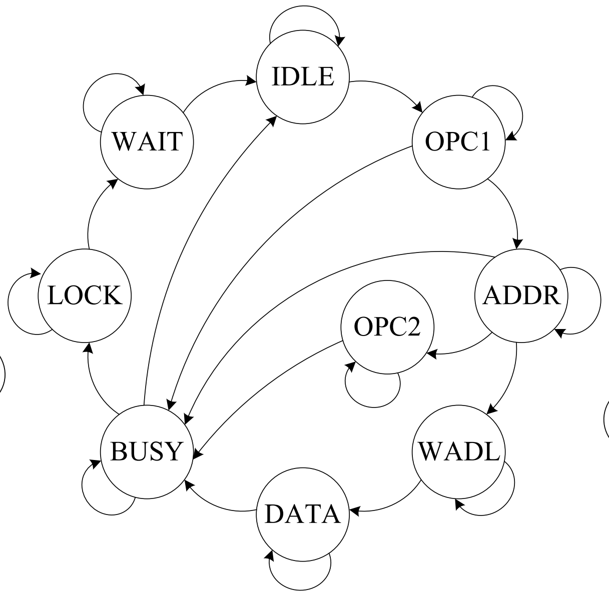
Flash commands classified into **four groups**: ***status***, ***erase***, ***read***, and ***program***

- ***status***: status checking commands, e.g. *read status* and *read status enhanced*.
- ***erase***: commands not using DQS signals, e.g. *erase block* and *reset*.
- ***read***: read data using DQS signals, e.g. *read page* and *read parameter page*.
- ***program***: write data using DQS signals, e.g. *program page* and *set feature*.

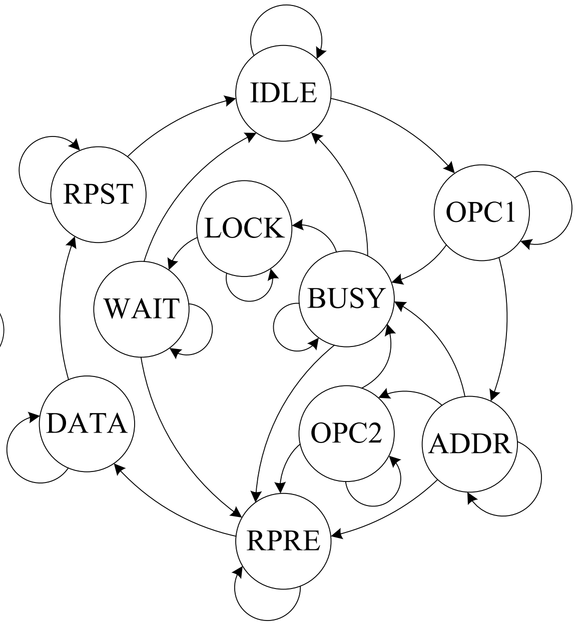
One FSM for each group, rather than for each command



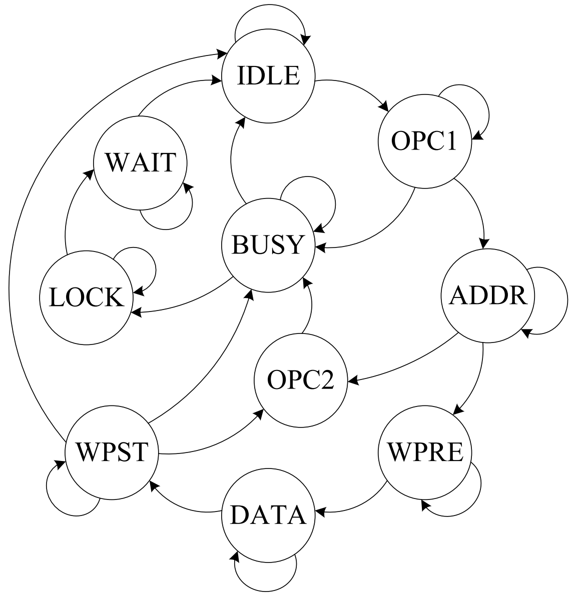
Status FSM



Erase FSM



Read FSM

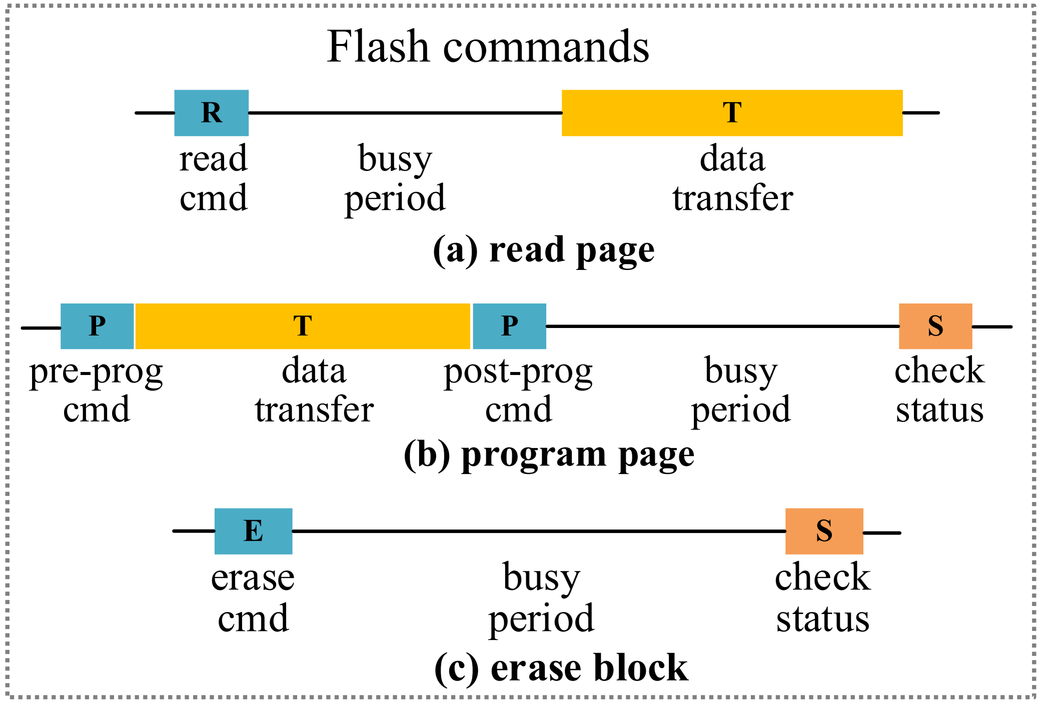


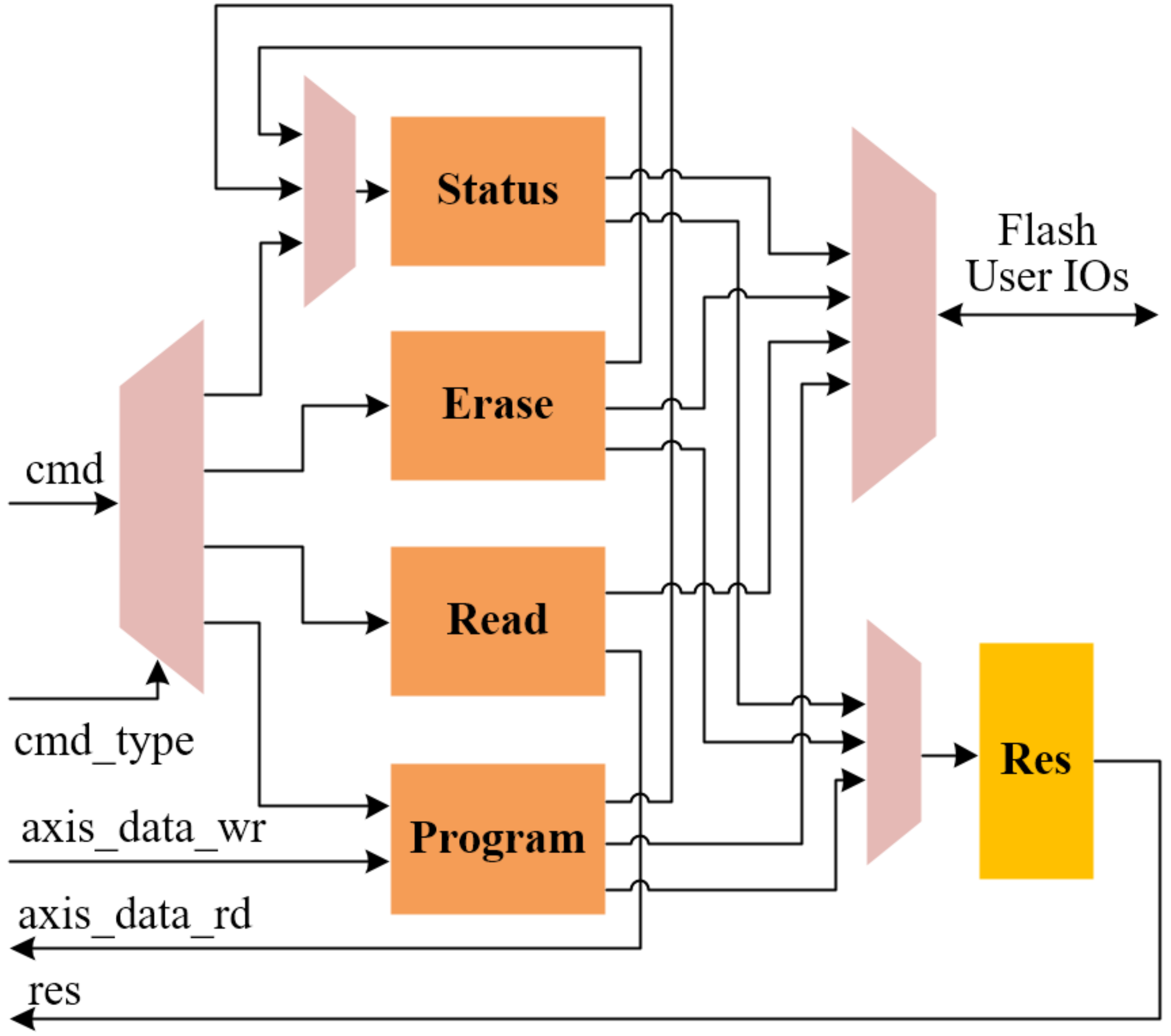
Program FSM

Flash Command Classification

Hardware-automated executer

Auto-schedule the command launching, data transferring, and status checking





Executer architecture

Fine-grained Interleaving

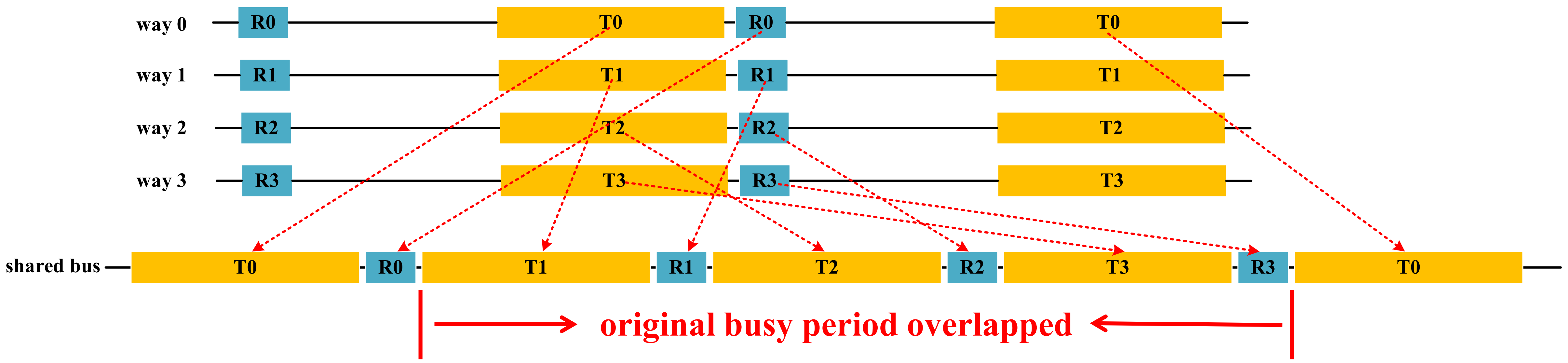
Long busy period & Commands executed in serial for a single way

$\Downarrow$ decrease

bandwidth utilization

$\Downarrow$ increase

Way-level fine-grained interleaving to overlap the busy period



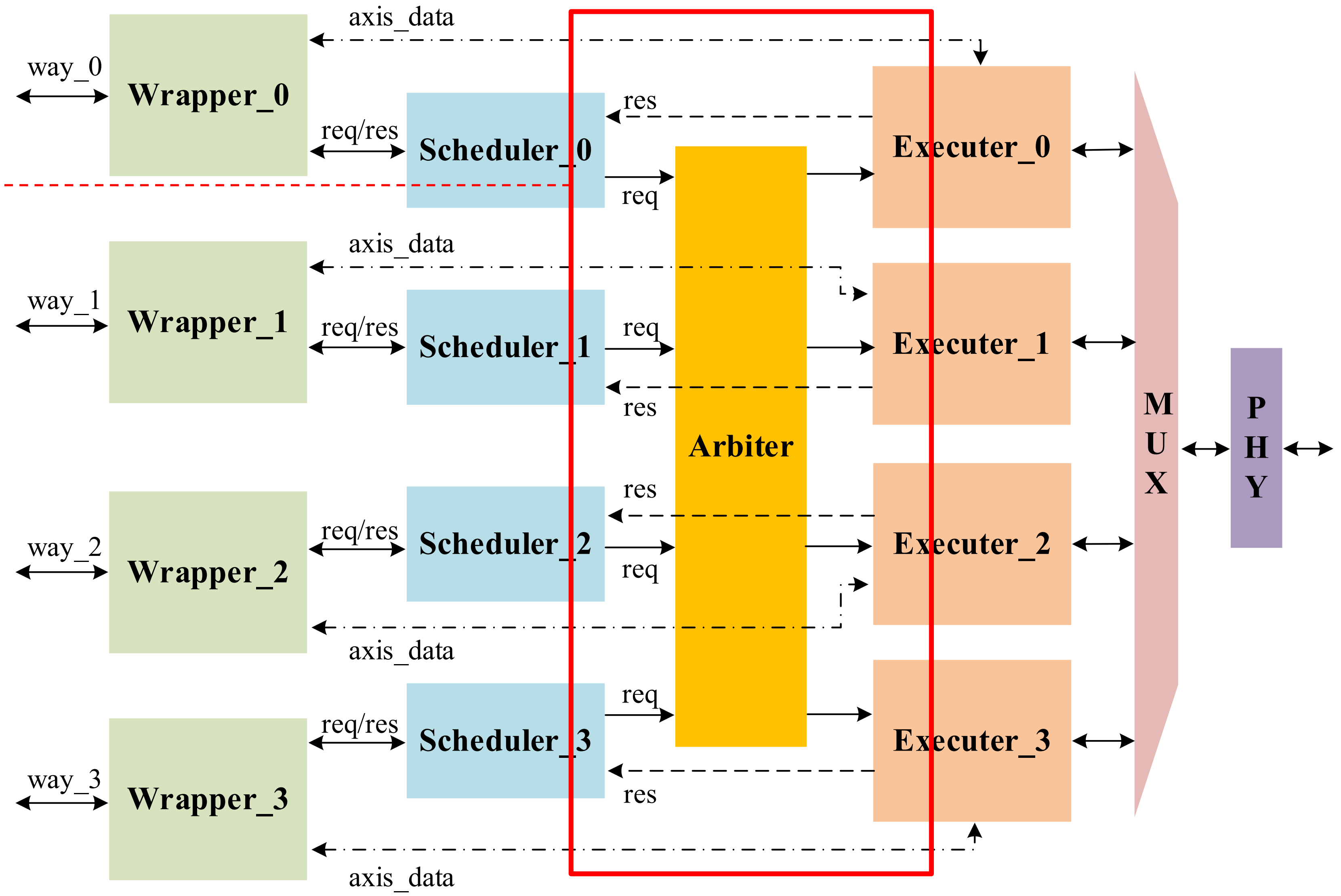
Fine-grained Interleaving

Hardware-automated interleaving implemented by Arbiter & Executers

Separate interface for each way rather than sharing interface

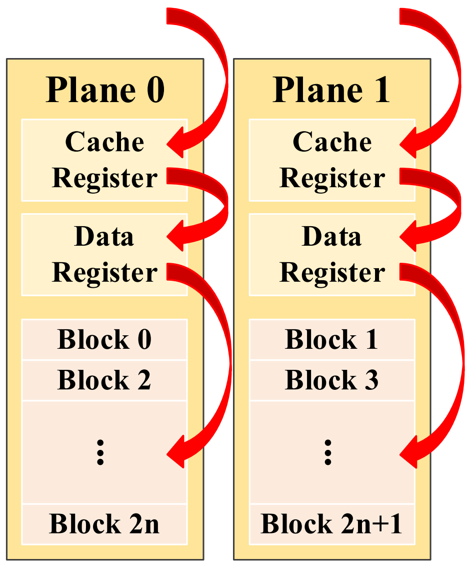
⬇

No need for out-of-order execution



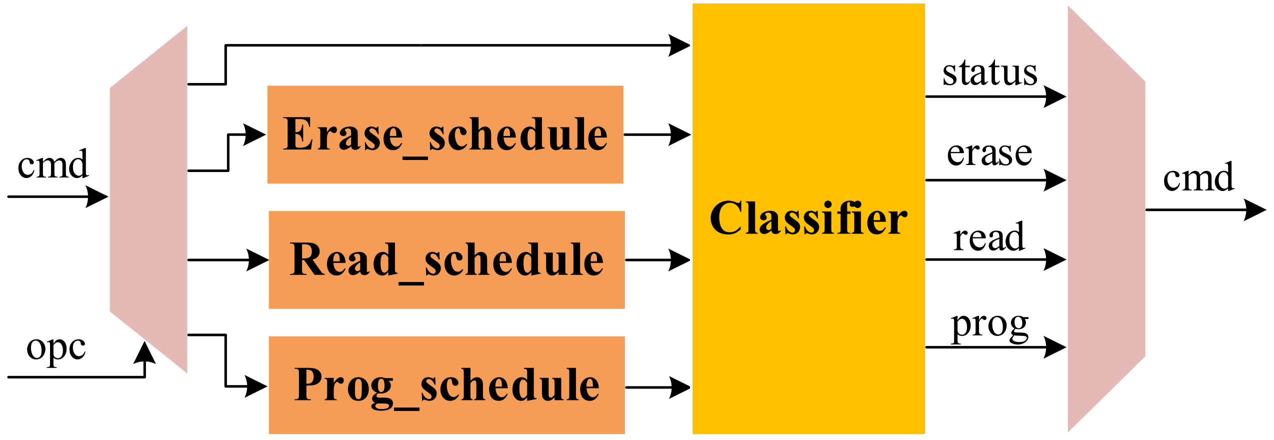
High-level Scheduling

- **Multi-plane operation**
  - planes within a die can operate simultaneously
  - same operation type, same die, block, and page addresses
- **Cache mode operation**
  - pipelining operations within a plane
  - consecutive operations of the same type



multi-plane & cache mode operations

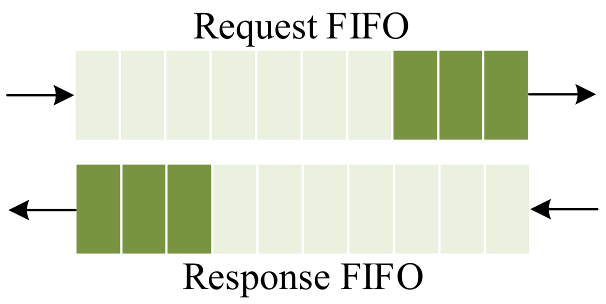
Scheduler generates more multi-plane and cache mode operations to **improve the bandwidth utilization**



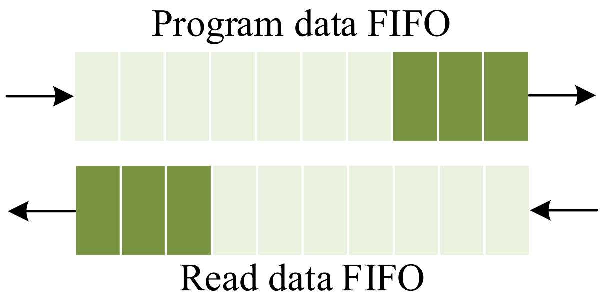
Scheduler architecture

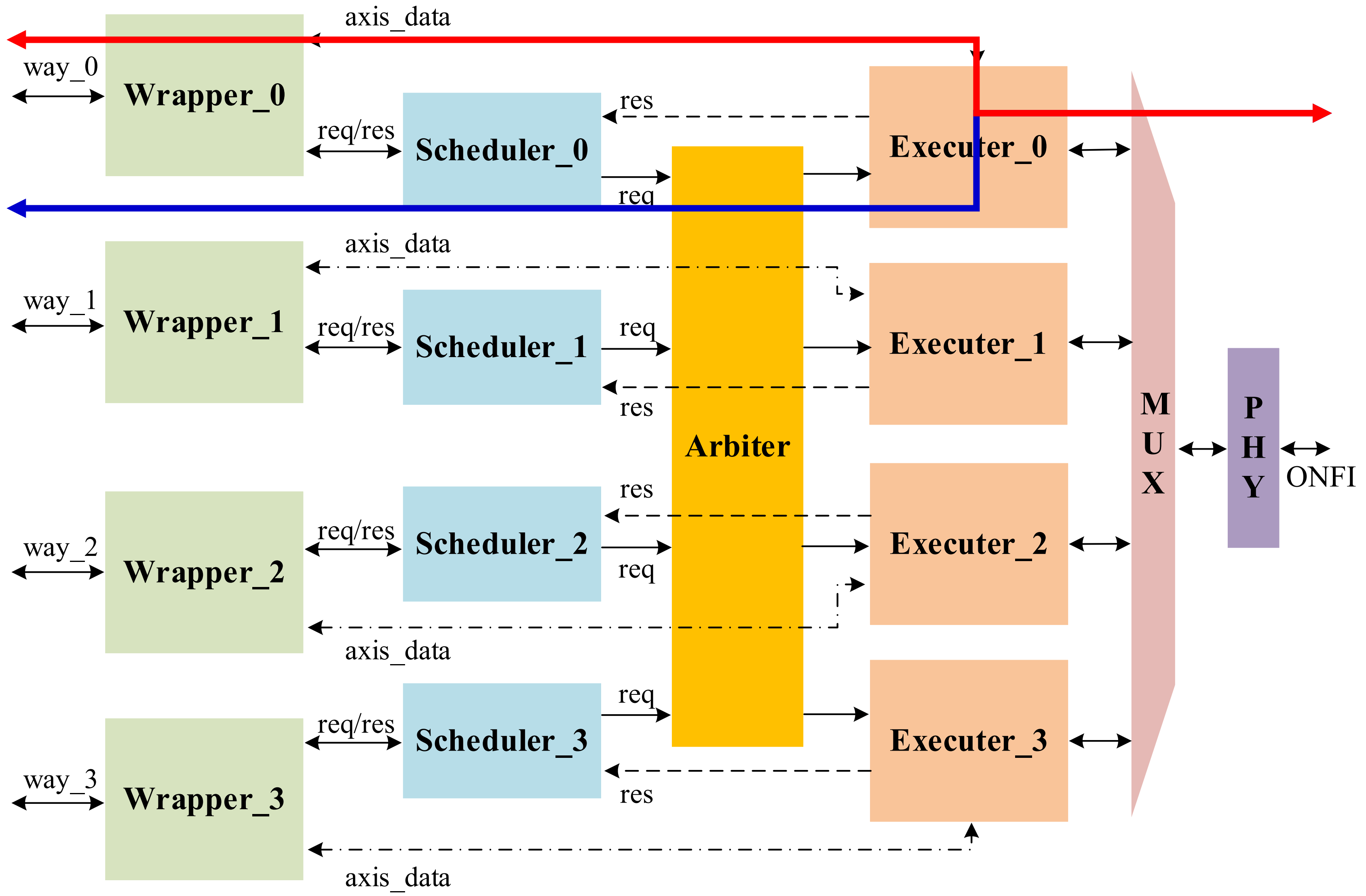
Open-way Architecture

Control Path:
multi-level (way-plane-cache), fine-grained scheduling



Data Path：
handshake-capable exploiting ONFI data pause mechanism, reduce data FIFO size





Evaluations

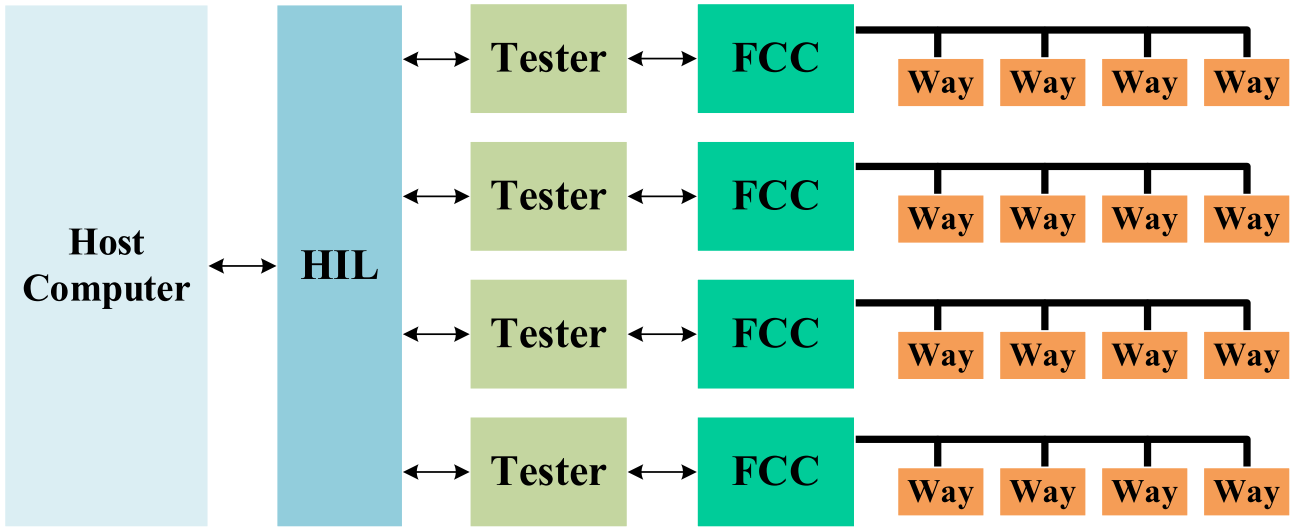
Experimental Setup

- Heterogeneous platform: FPGA + CPU
- Tester: generate various commands
- NFC: four channels, four ways per channel

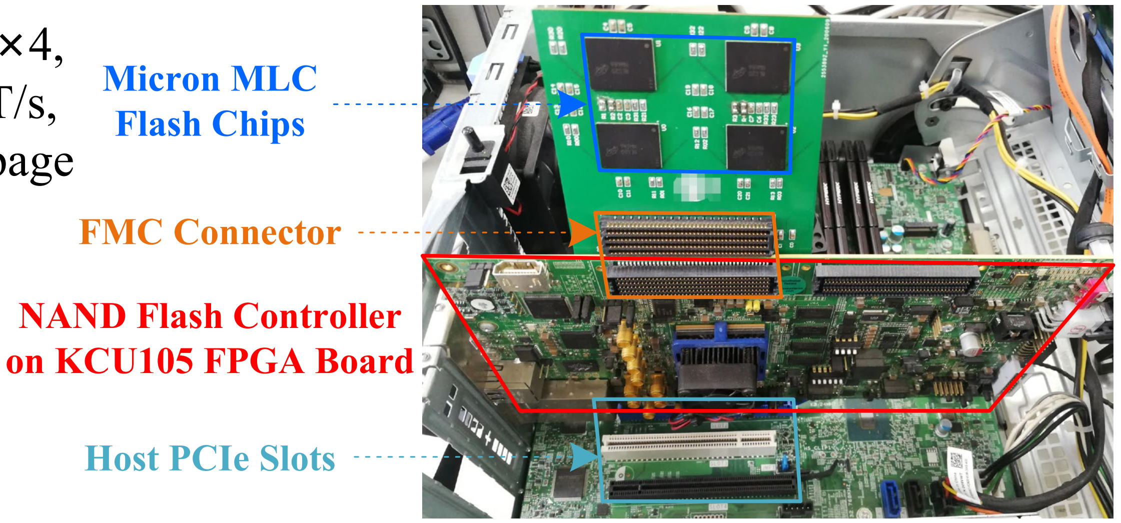
Overlap the busy period completely:
Given I/O speed (*f*), page size (*s*), and busy time (*t*),
the interleaving page number (*n*): $n = \frac{ft}{s}$
***n* = 2.3/61.0** for read/program
Max theoretical bandwidth: 1.33GB/s

Flash timing characteristics

Flash operations	**Typical**	**Max**
Read page time	--	**115μs**
Cache read busy time	26μs	115μs
Program page time	1600μs	**3000μs**
Cache program busy time	1100μs	3000μs
Erase block time	3ms	12ms



512Gb×4,
333MT/s,
16KB-page



* HIL: host interface logic

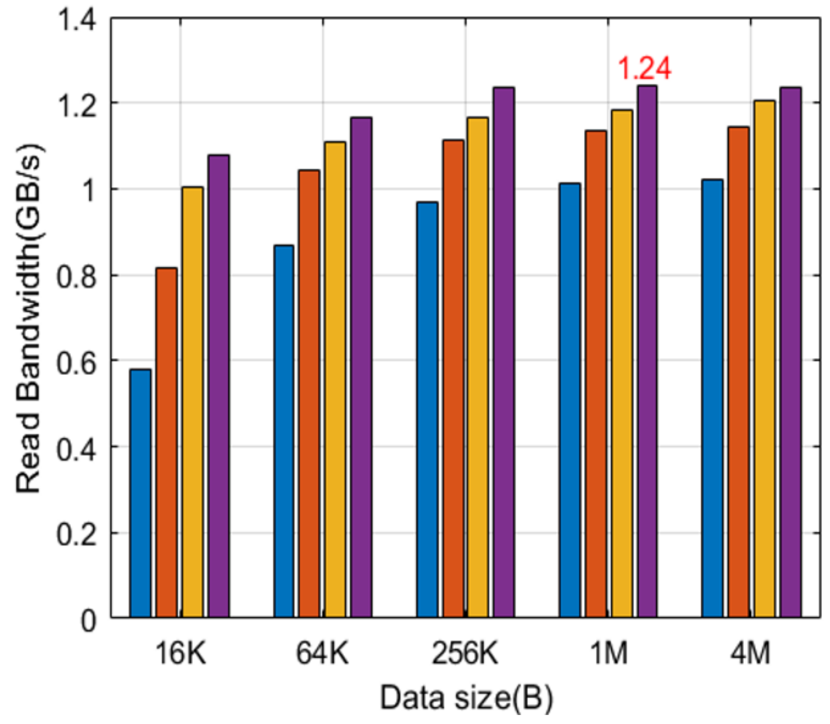
Bandwidth Evaluation

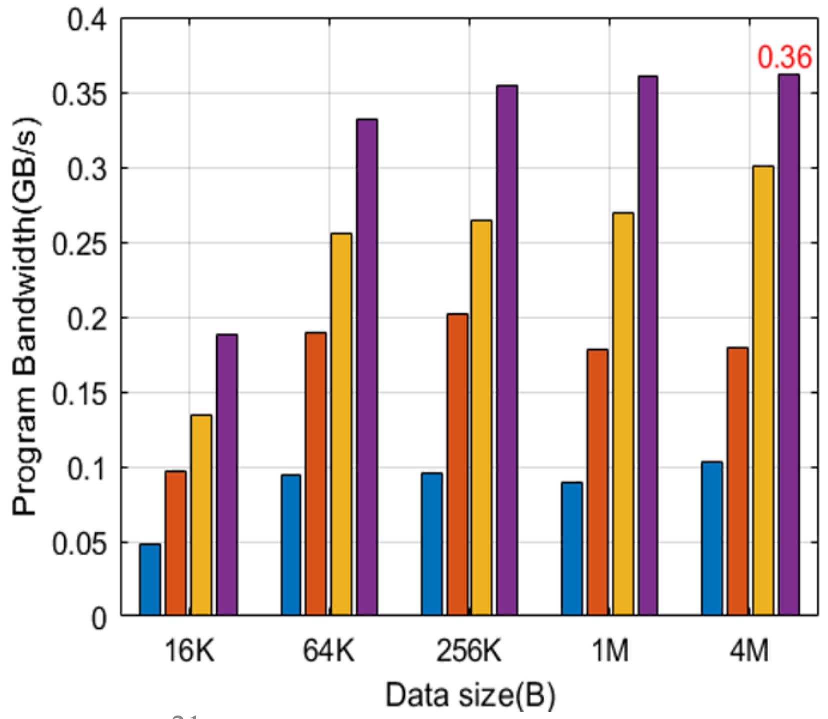
- the data size larger, the more multi-plane and cache mode operations.
- the bandwidth increases with the data size and the way number.

◆ **Four-way** configuration, max read & program bandwidth: 1.2 GB/s & 0.36 GB/s, 93% & 27% of theoretical bandwidth.

◆ **Eight-way** configuration: Cosmos+ OpenSSD [6] 80% and 39%, Gemini [21] 78% and 24%.

ours is 13% higher



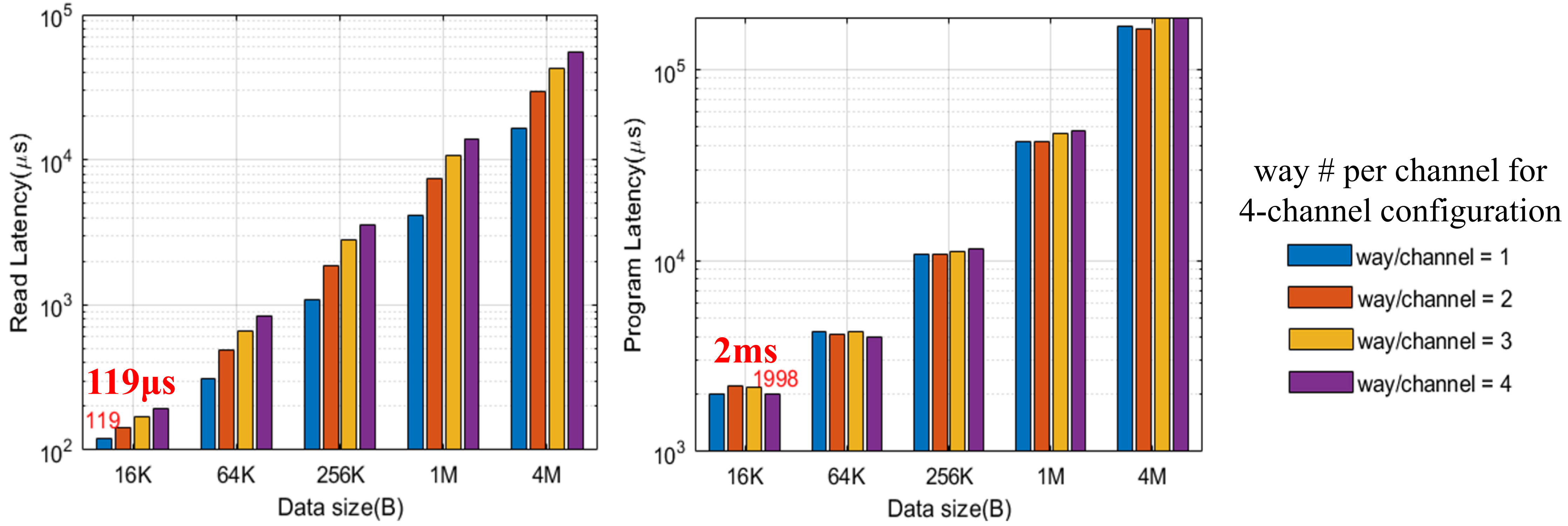


way # per channel for 4-channel configuration

🟦 way/channel = 1
🟧 way/channel = 2
🟨 way/channel = 3
🟪 way/channel = 4

Latency Evaluation

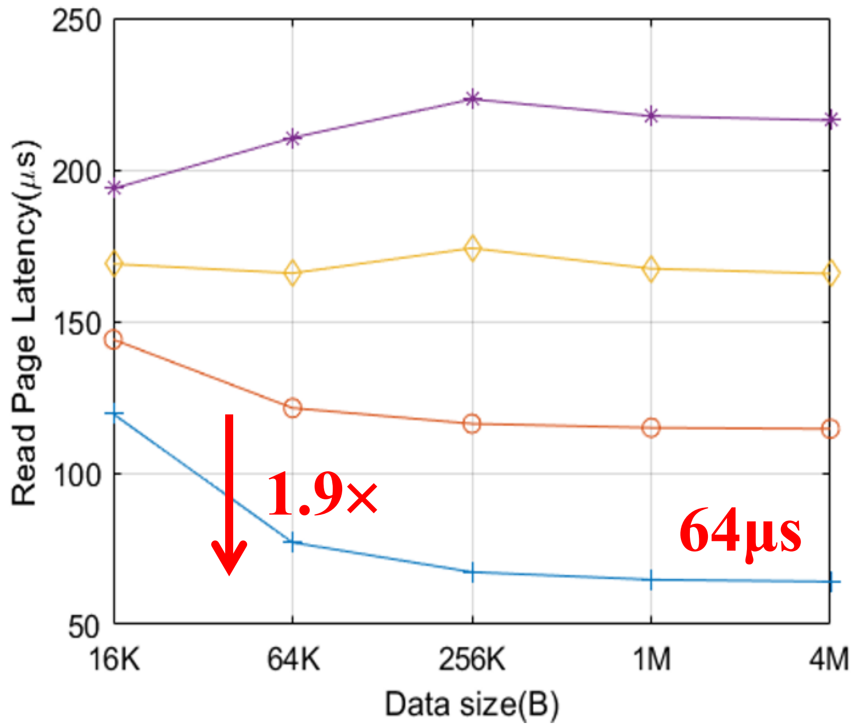
- Reading and programming **latencies rise with data size**.
- **Reading latency increases with way number**, since the data transferring time is comparable with the reading busy time.
- **Programming latencies almost the same for different way numbers**, since the data transferring time is much less than the programming busy time.

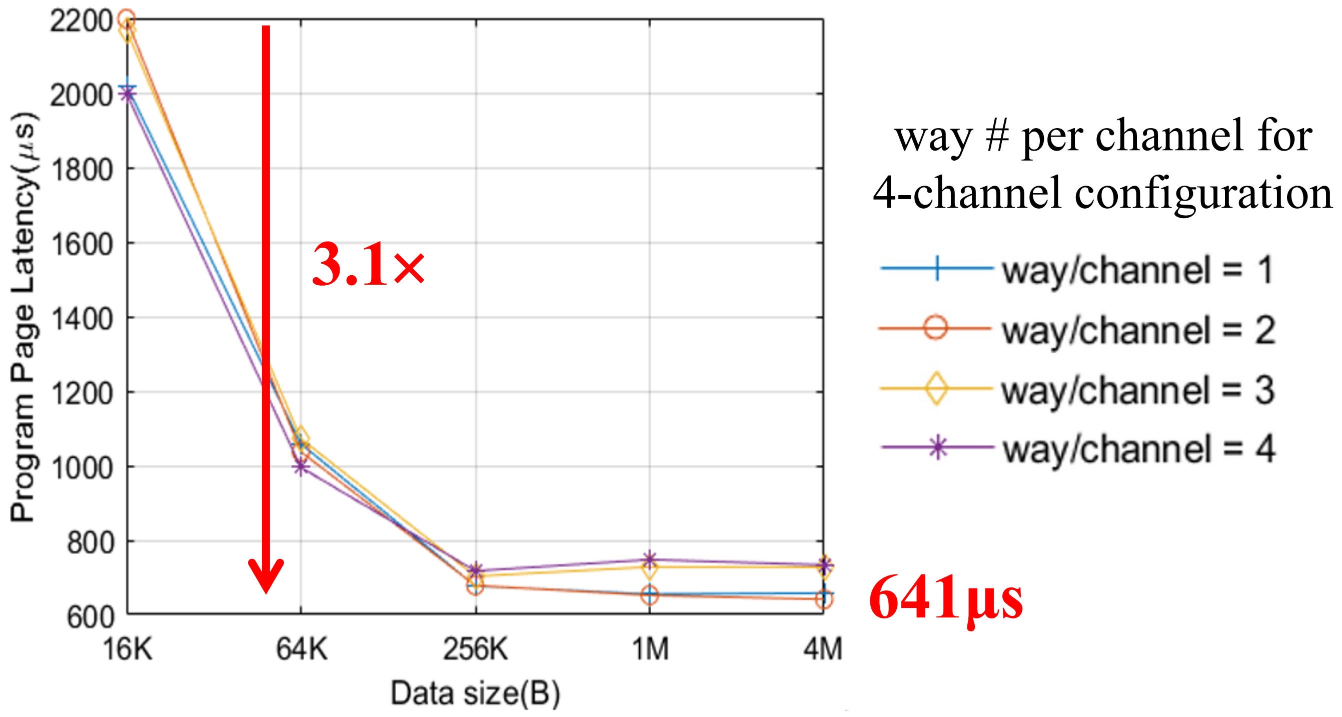


Average Latency Evaluation

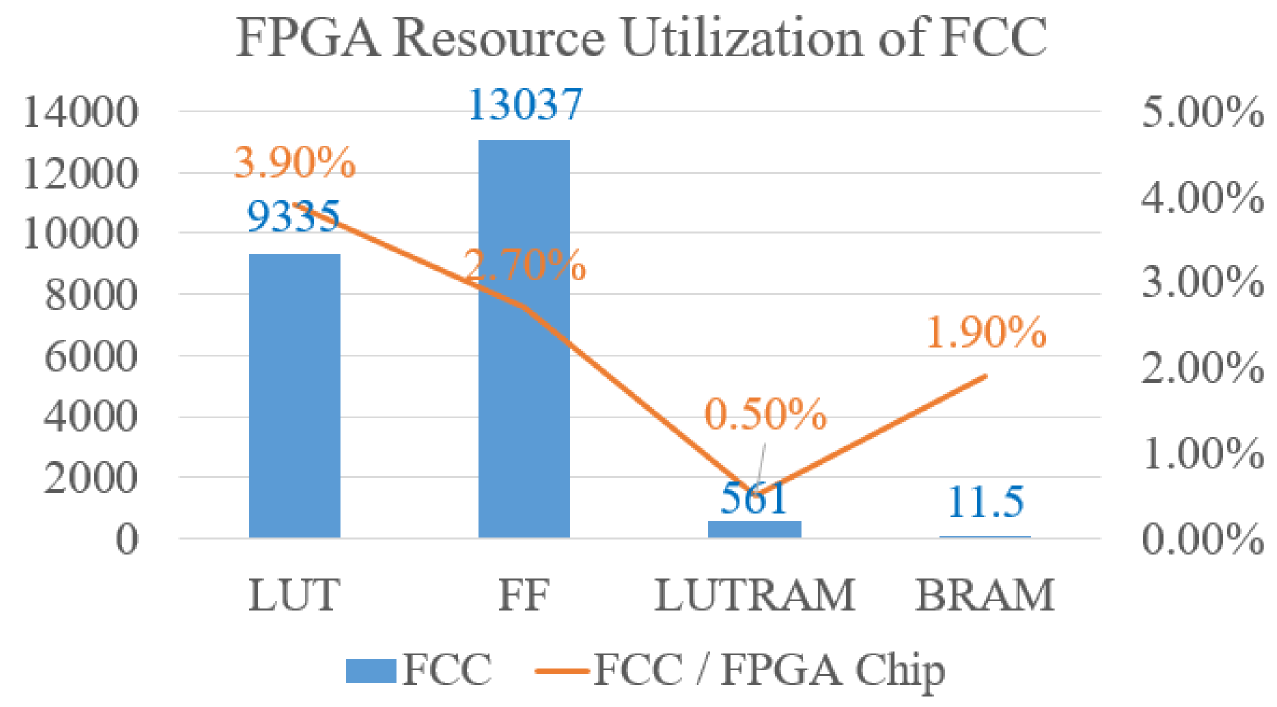
- **Different trends for read**: way-level interleaving can already saturate the I/O bandwidth when way number exceeds two.
- **Decreases obviously for program**: too long programming busy time

multi-level parallelism generally positive for programming but maybe negative for reading.

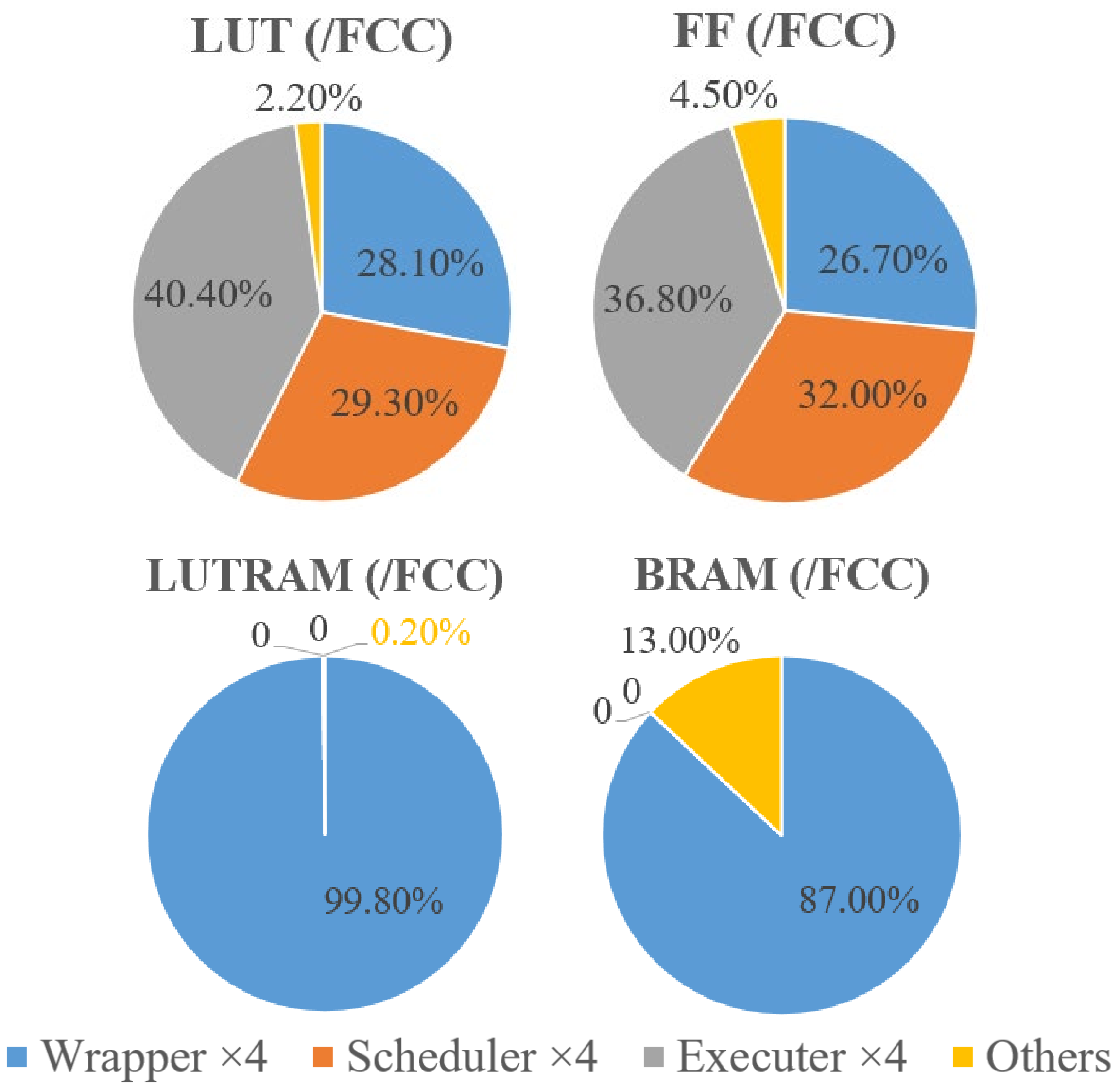




FPGA Resource Utilization



Xilinx KCU105 FPGA



Resource Utilization Comparisons

FPGA resource utilization comparisons between our FCC and other FCCs

Name	FPGA Device	Page Size	LUT	FF	BRAM
FCC in Gemini [21]	Vertex-6 XC6VLX240T	4KB	1673	2314	13
FCC in Cosmos+ OpenSSD [6]	Zynq-7000 XC7Z045-3FFG900	16KB	10988	7548	21
Our FCC	KCU105 XCKU040-2FFGVA1156E	16KB	**9335**	**13037**	**11.5**

Lowest BRAM usage: due to the **open-way** architecture and the **data pause mechanism** exploitation
Larger LUT & FF usage: our FCC supports **more levels of parallelism and more Flash operations**.

Summary

Conclusions

1. Multi-level parallelism
   - ☐ channel-way-plane-cache (parallel or interleaving or pipelining)
2. Flash controller design
   - ☐ open-way architecture to improve bandwidth utilization
   - ☐ dual-level hardware scheduler to improve bandwidth utilization
   - ☐ Flash command classification to reduce hardware resources
3. Evaluation
   - ☐ max. bandwidth: 1.2 GB/s, 93% of theoretical bandwidth, at least 13% higher than other controllers
   - ☐ min. latency for page read/program: 119µs/2ms
   - ☐ min. read/program latency with fine-grained interleaving: 64µs /641µs, speeding up by 1.9× and 3.1×

THANKS!

Yunhui Qiu, Wenbo Yin, Lingli Wang
State Key Laboratory of ASIC and System, Fudan University
FPL'21 2021-09-01

Open-source RTL codes: https://github.com/yhqiu16/OCOWFC