usenix ASSOCIATION
ATC 2020

Fully Hardware Automated Open Research Framework for Future Fast NVMe Device

Myoungsoo Jung

Computer Architecture and Memory systems Laboratory

KAIST EE

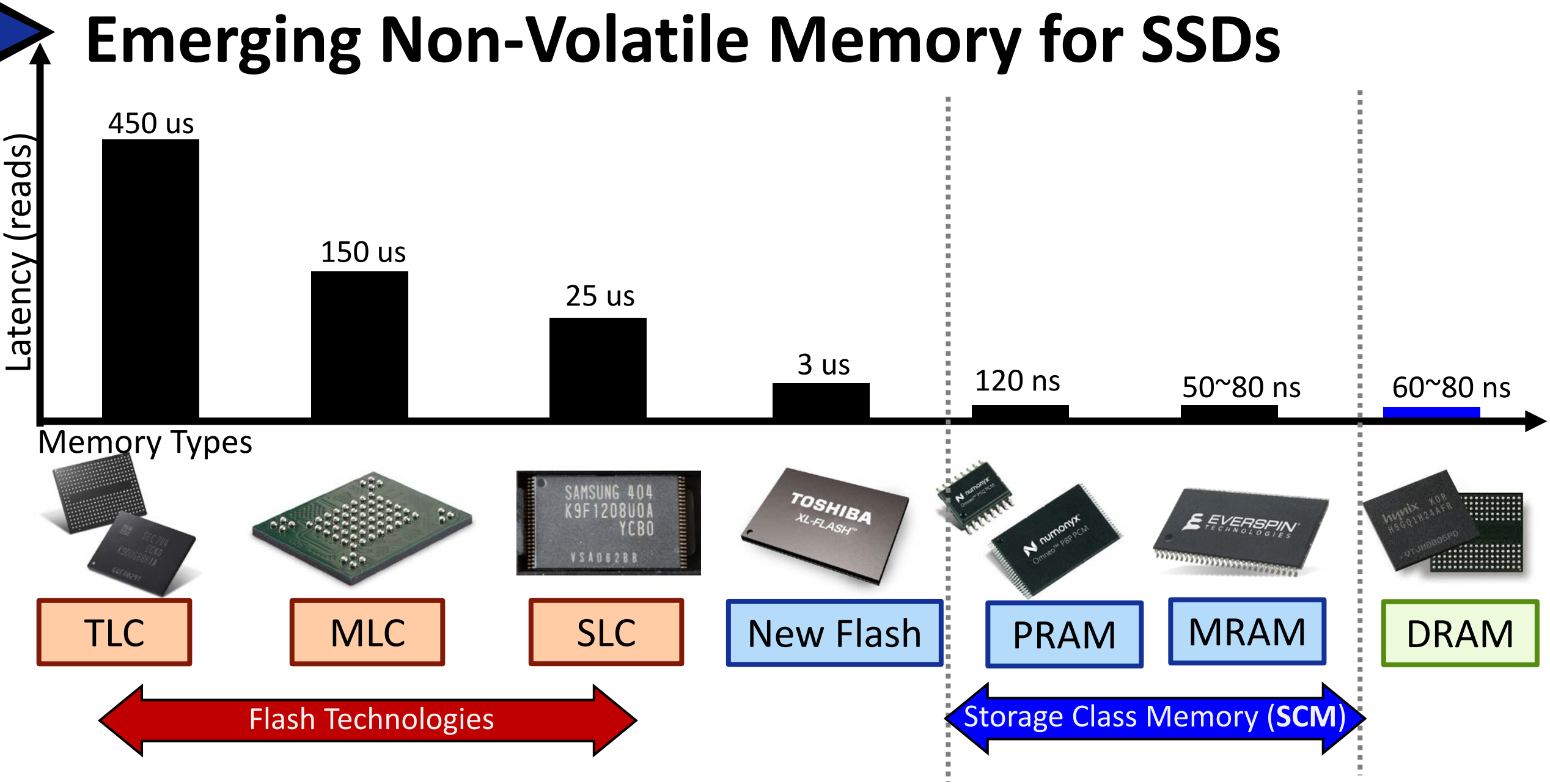
CAMELab

CAMEL

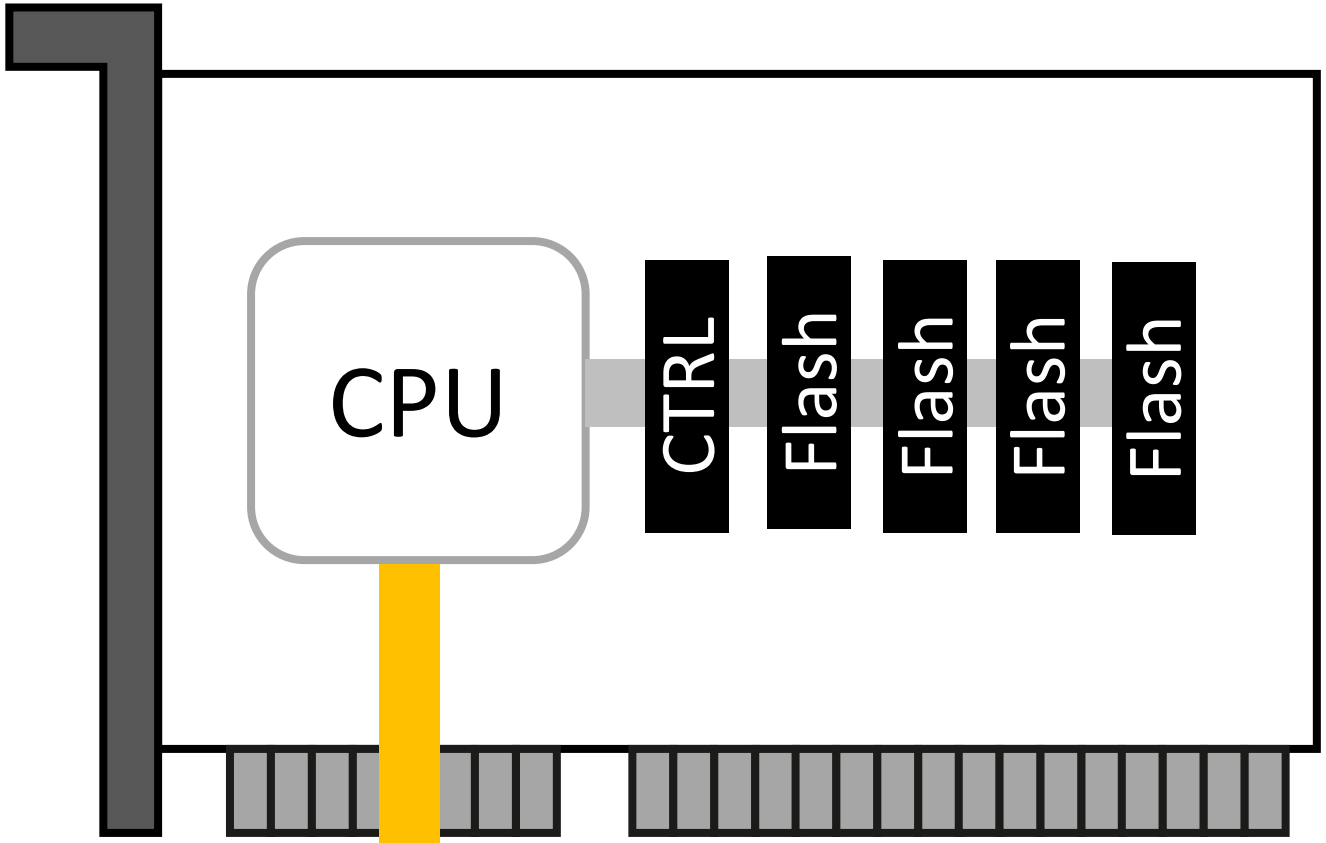
Sponsored by

MEMRAY

Emerging Non-Volatile Memory for SSDs



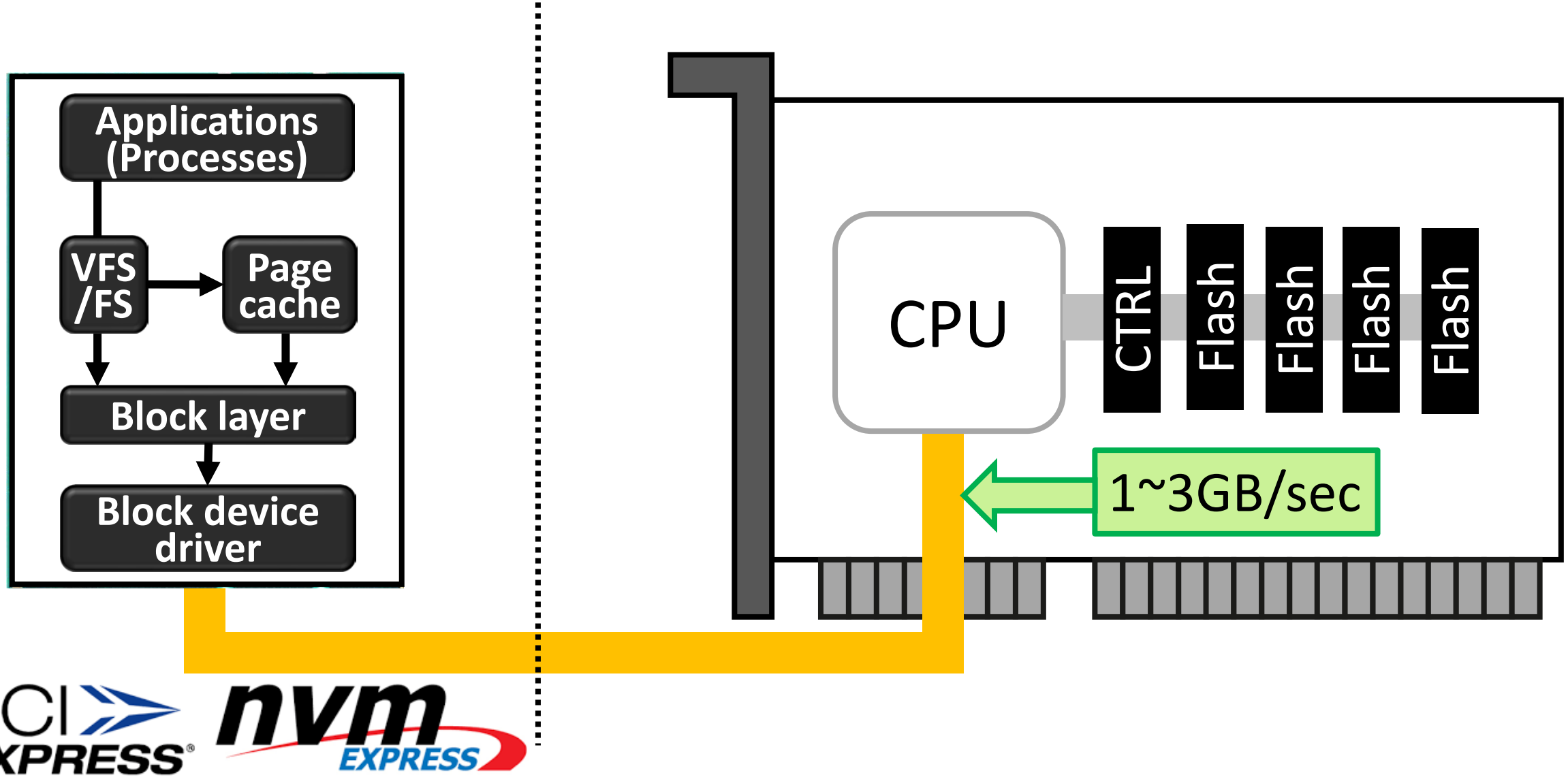
NVMe Internals and Interfaces



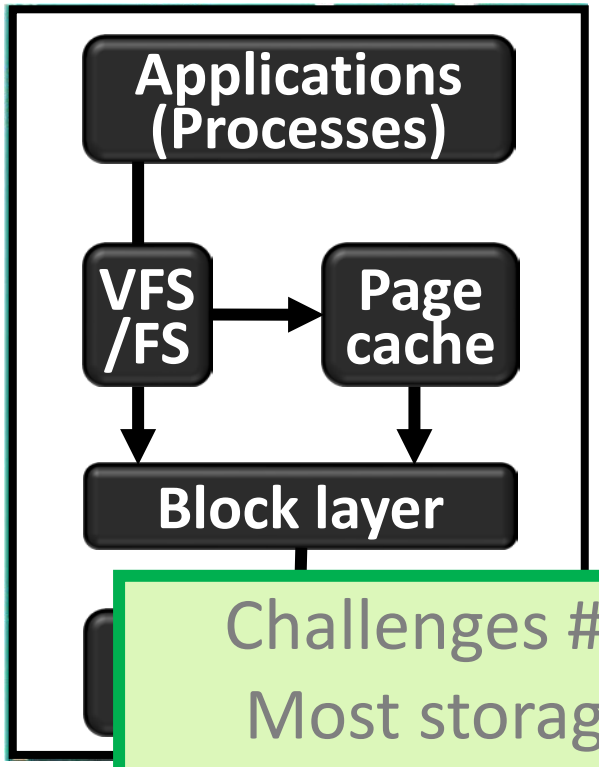
PCI EXPRESS®

nvm EXPRESS

NVMe Storage Stack



NVMe Storage Stack Redesign

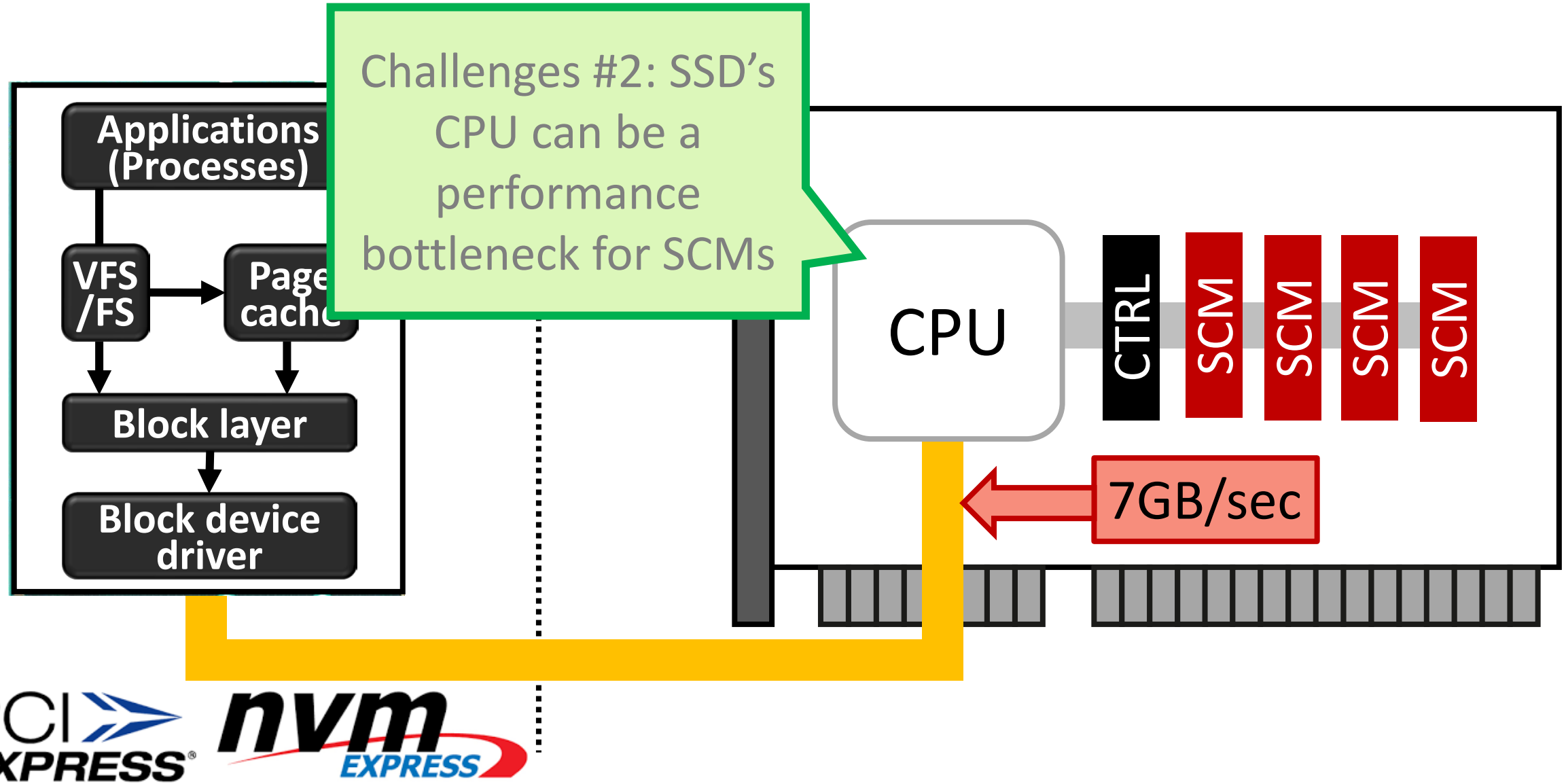


Challenges #1: Most storage research relies on simulation/kernel-level emulation

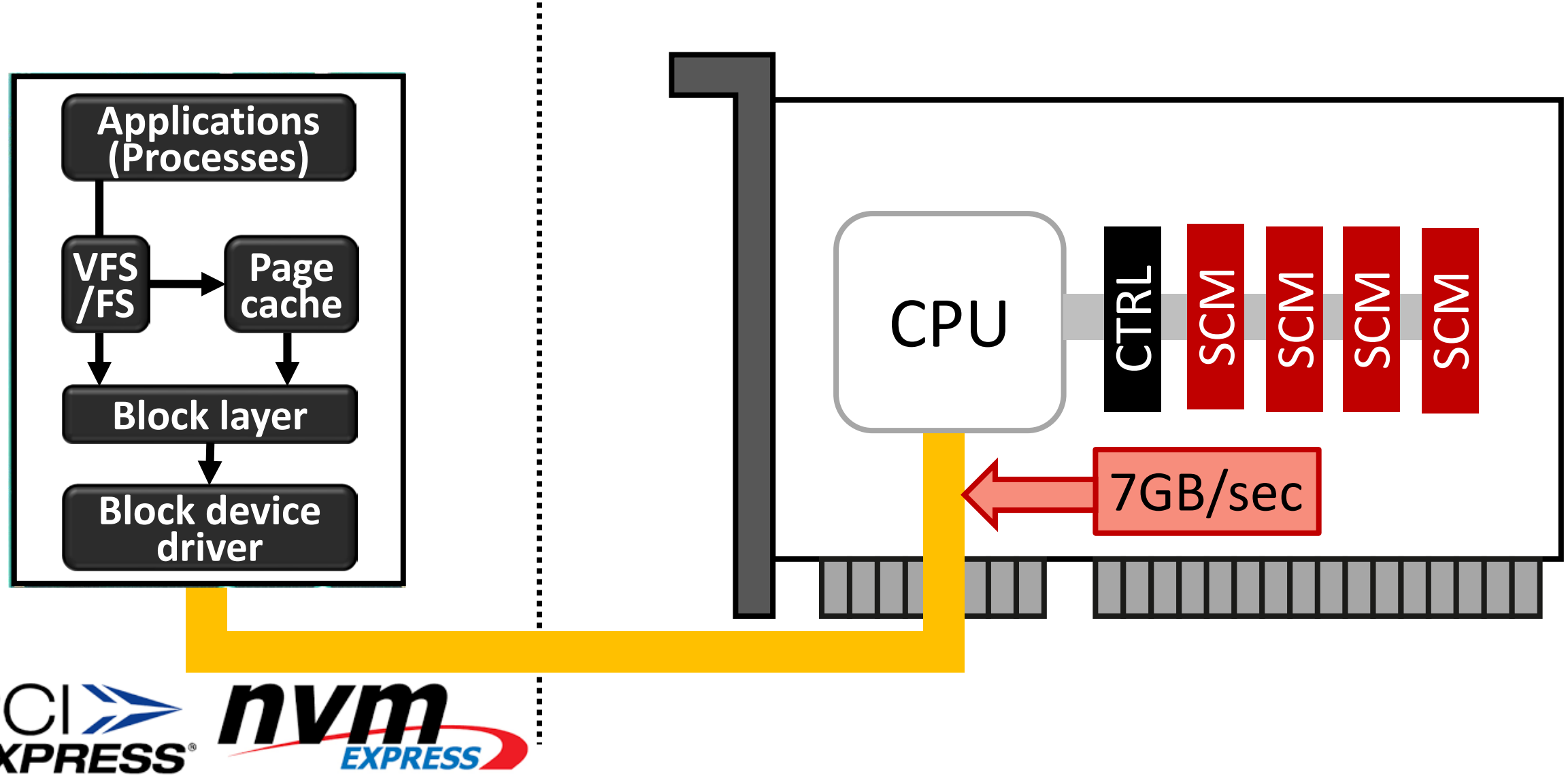
- FlashShare: Punching Through Server Storage Stack from Kernel to Firmware for Ultra-Low Latency SSDs (OSDI'18)
- De-indirection for Flash-Based SSDs with Nameless writes (FAST'12)
- Towards SLO Complying SSDs Through OPS Isolation (FAST'15)
- The case of FEMU: Cheap, Accurate, Scalable and Extensible Flash Emulator (FAST'18)
- There're more and more!

PCI EXPR

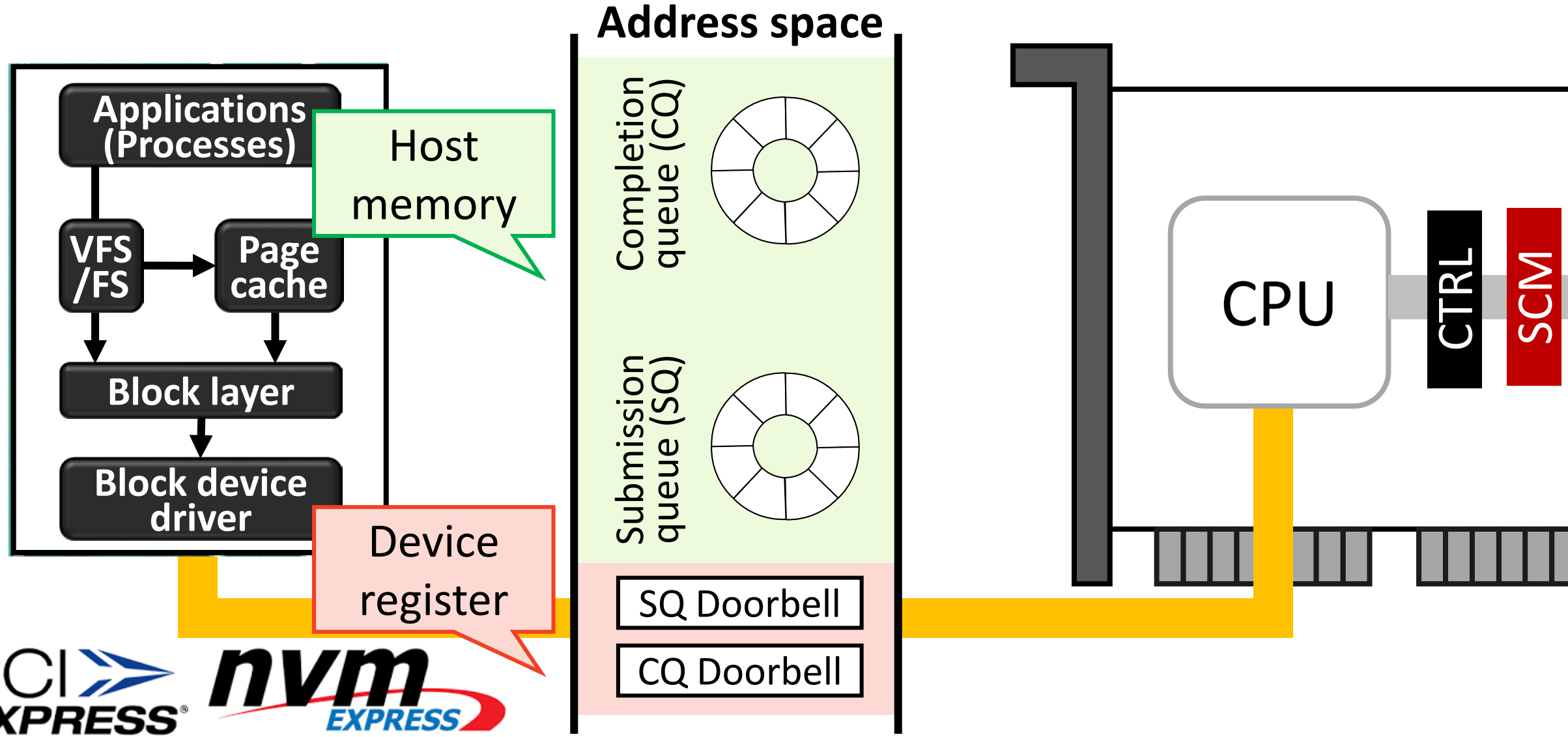
SCM-based NVMe Storage Card



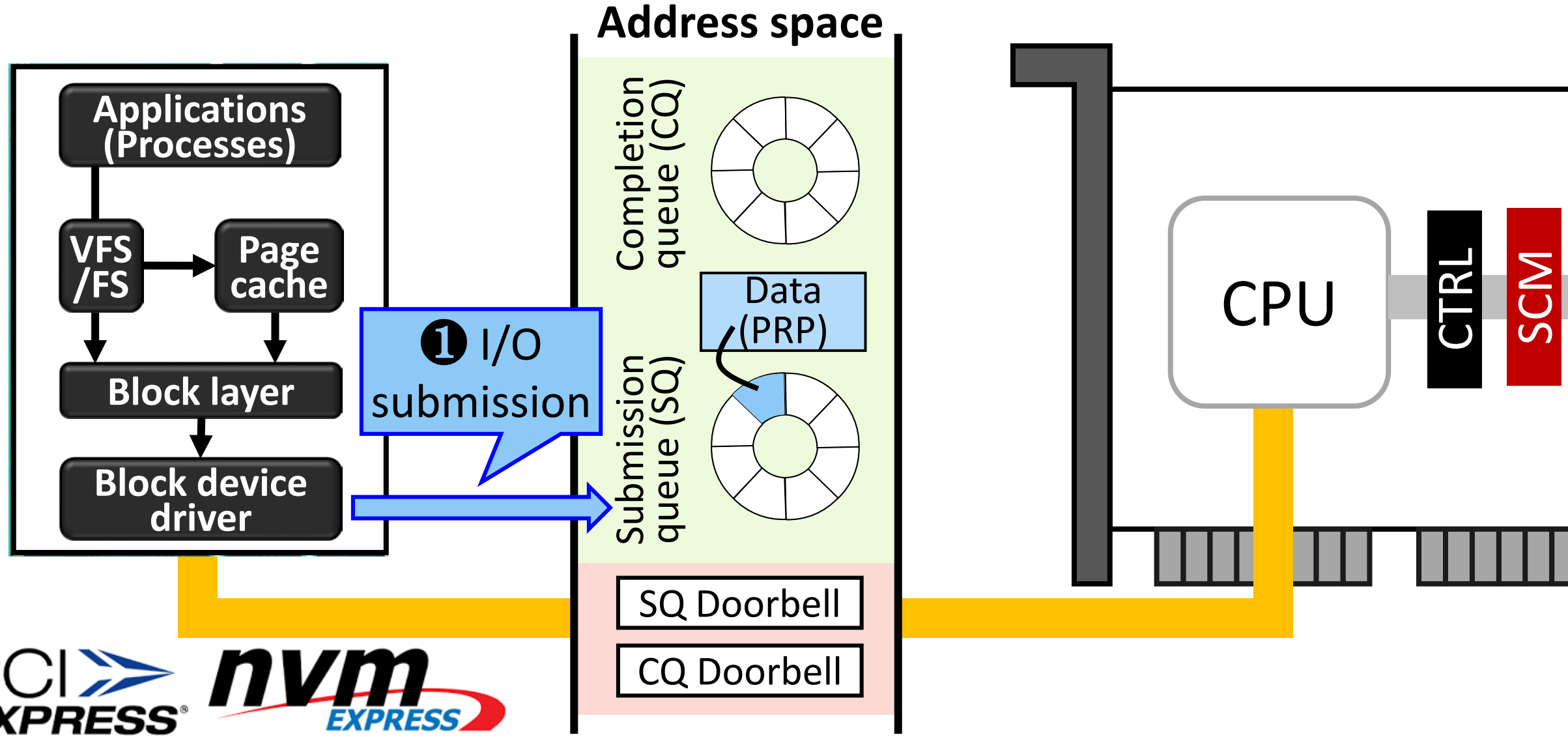
What Does SSD's CPU Do?



What Does SSD's CPU Do?

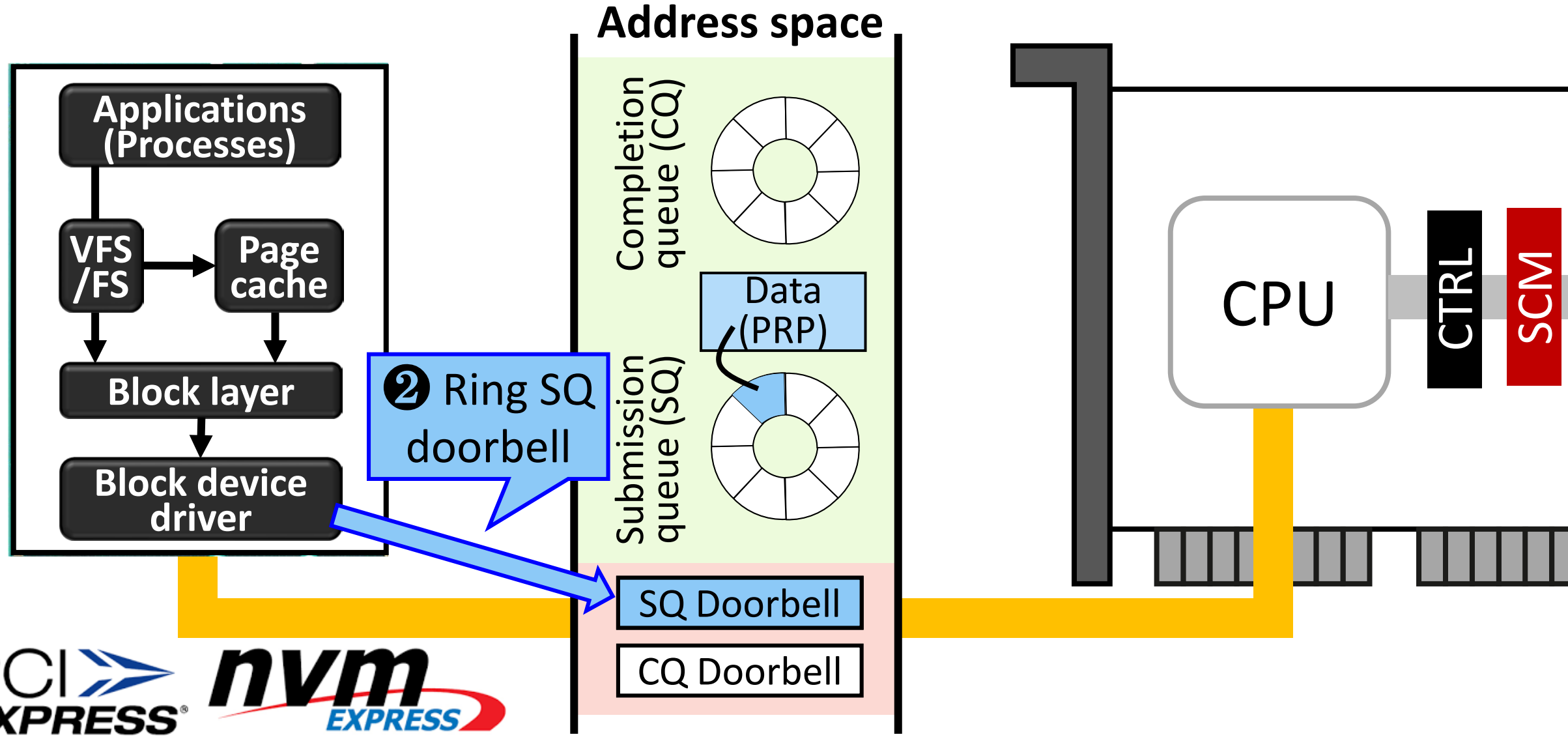


What Does SSD's CPU Do?



PCI EXPRESS® nvm EXPRESS

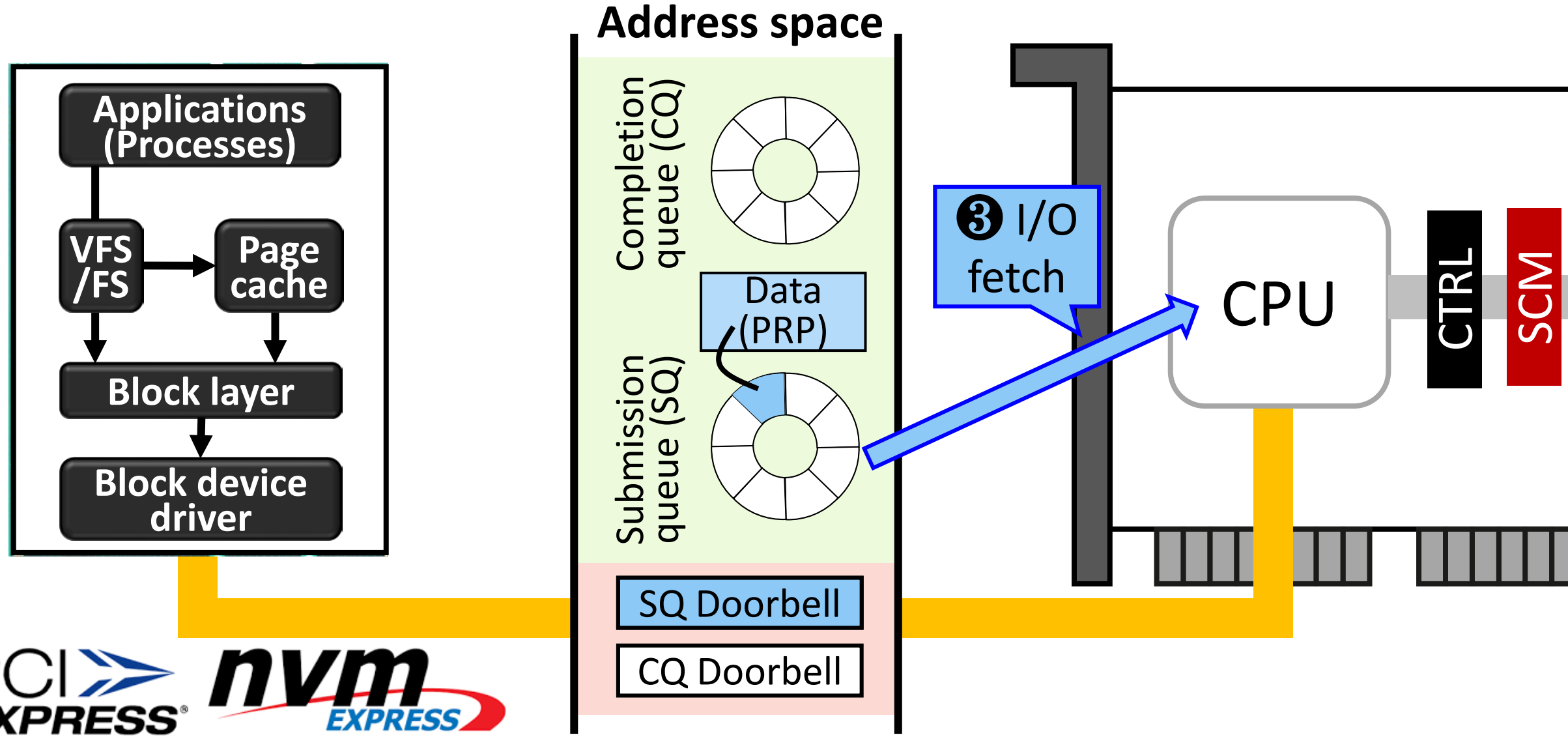
What Does SSD's CPU Do?



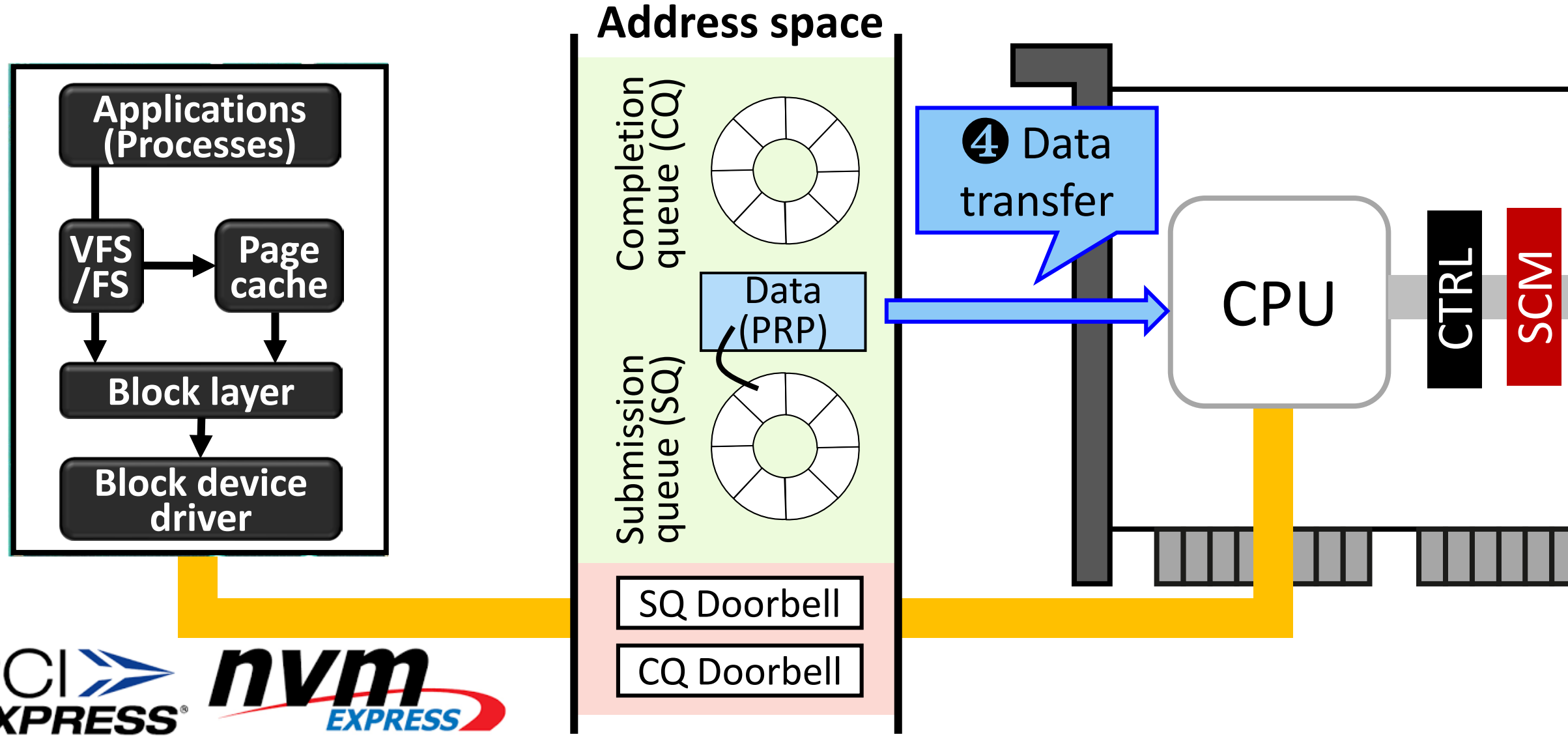
PCI EXPRESS®

nvm EXPRESS

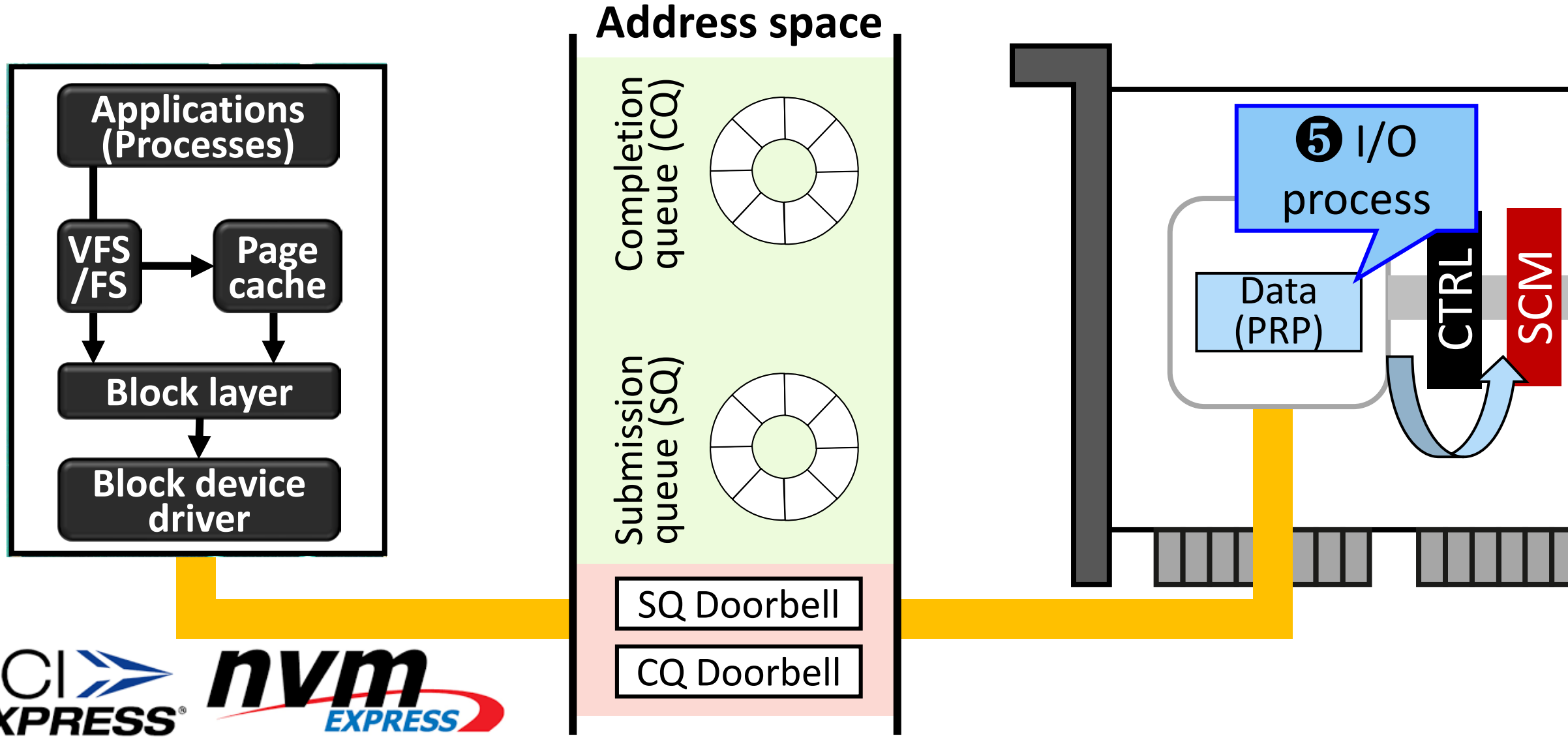
What Does SSD's CPU Do?



What Does SSD's CPU Do?



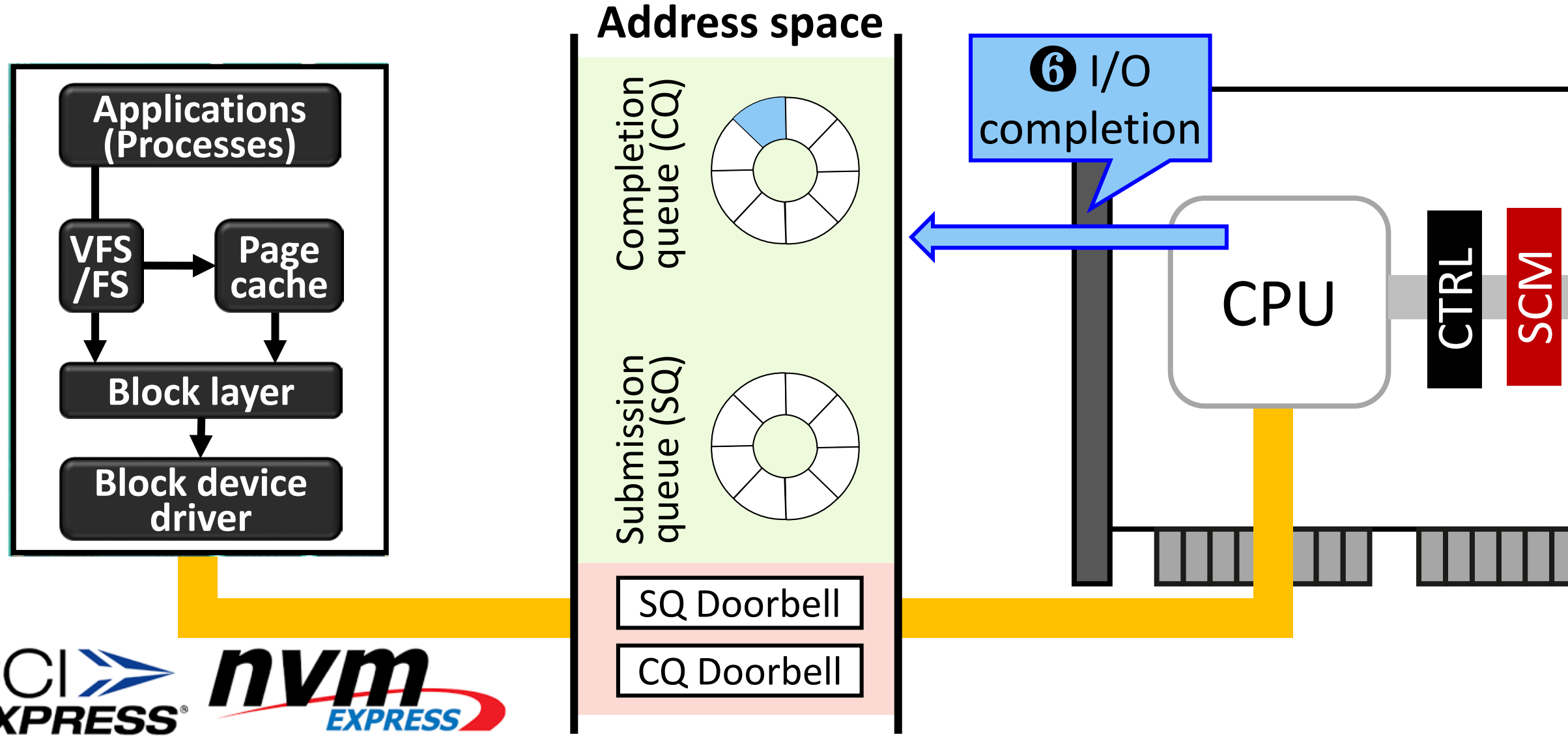
What Does SSD's CPU Do?



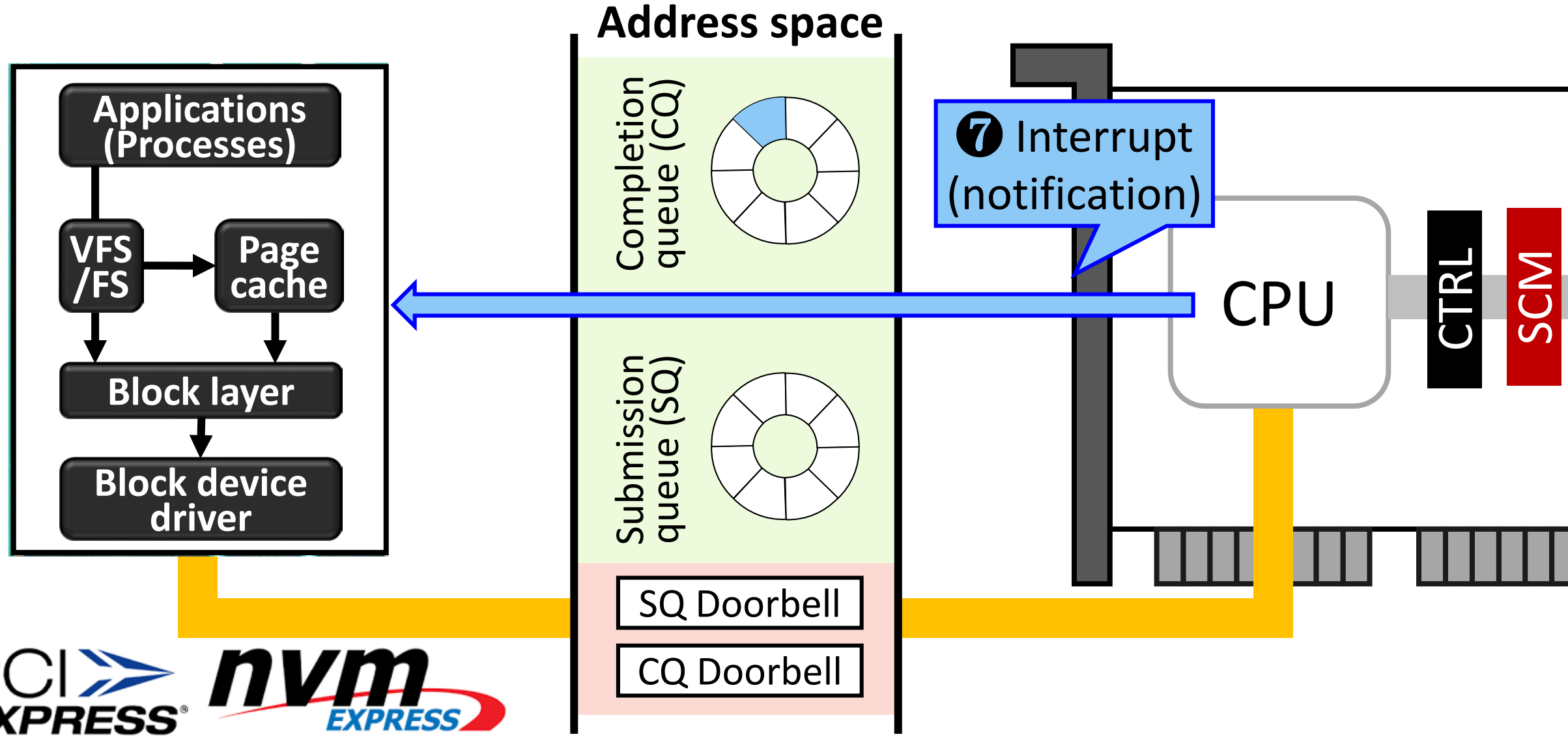
PCI EXPRESS

nvm EXPRESS

What Does SSD's CPU Do?

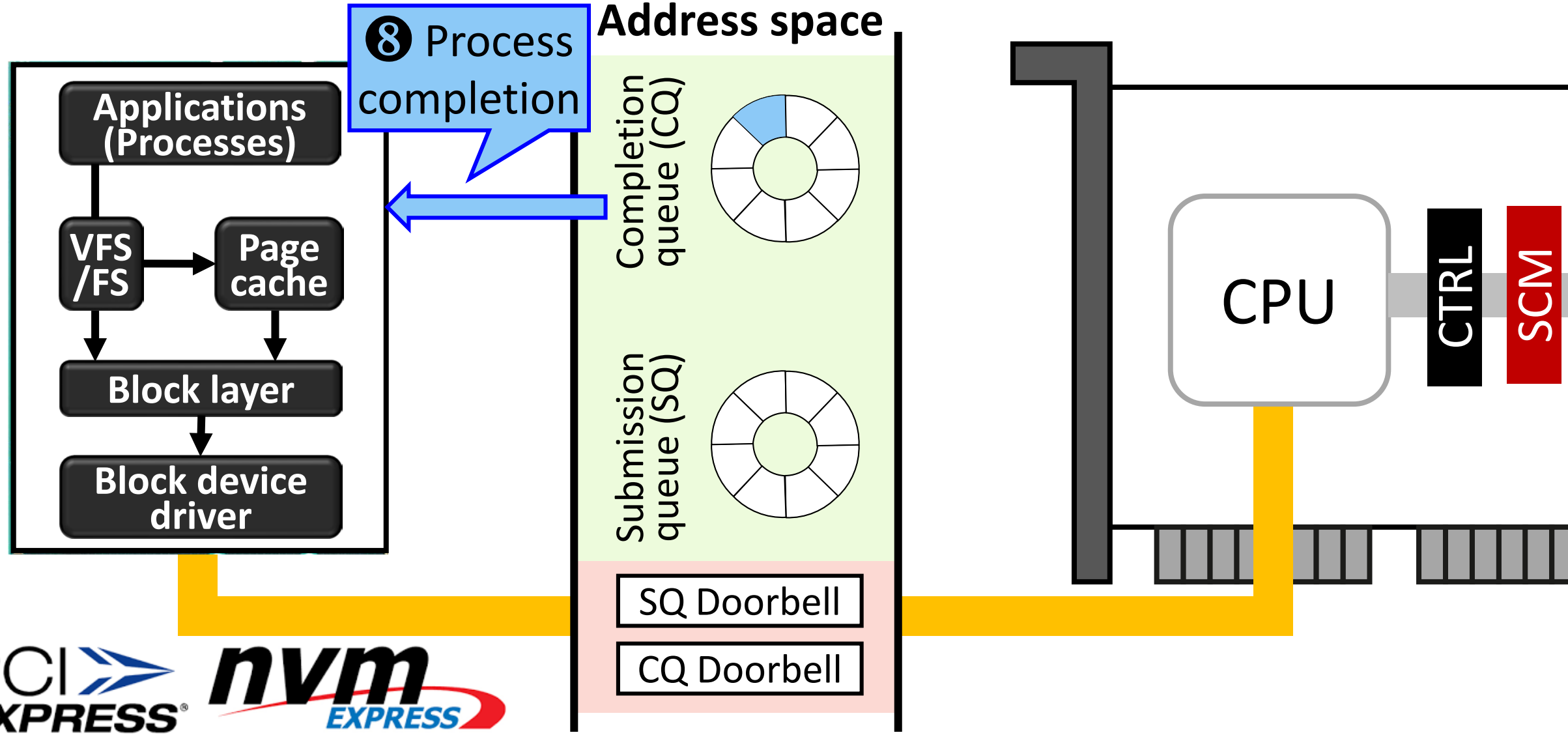


What Does SSD's CPU Do?



PCI EXPRESS® nvm EXPRESS

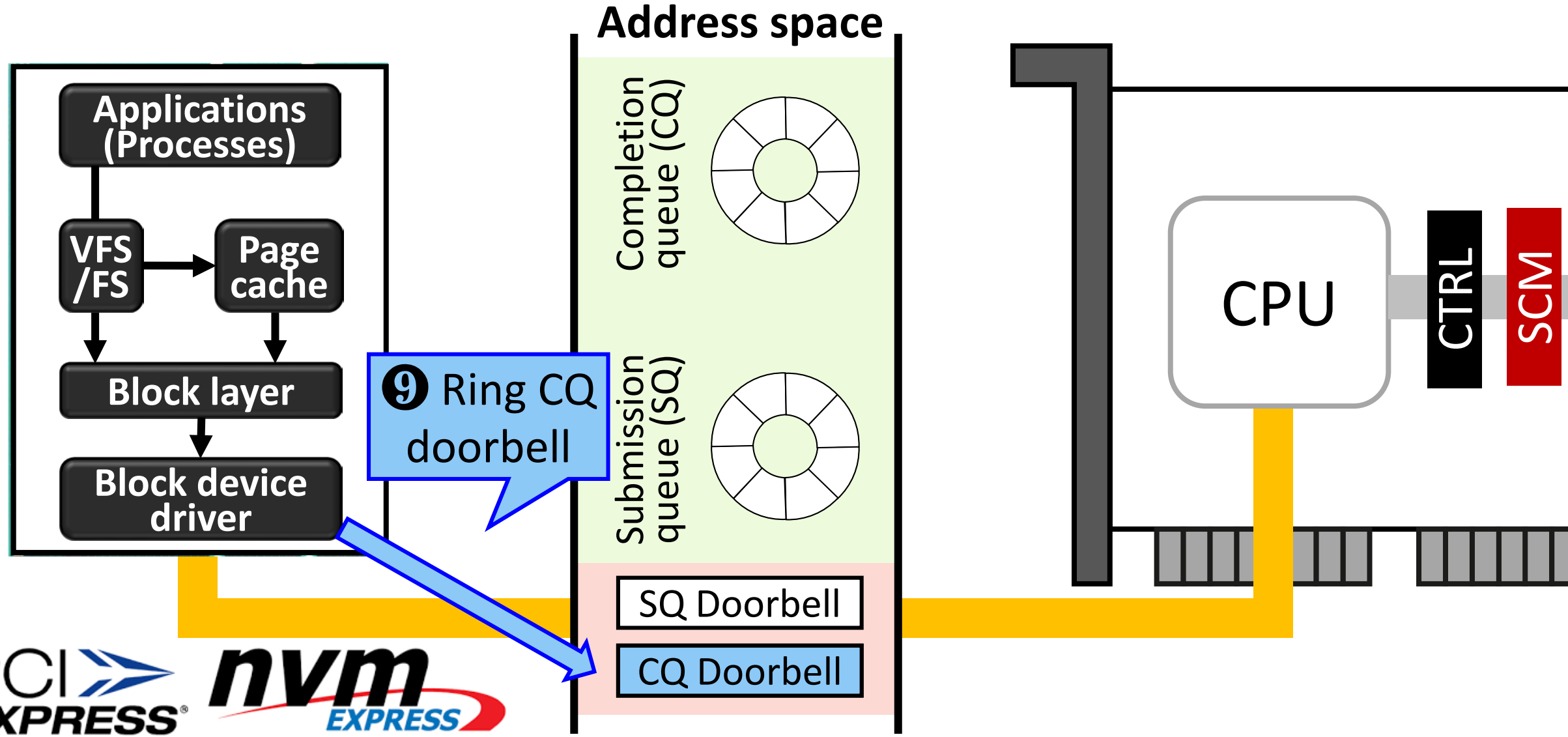
What Does SSD's CPU Do?



PCI EXPRESS®

nvm EXPRESS

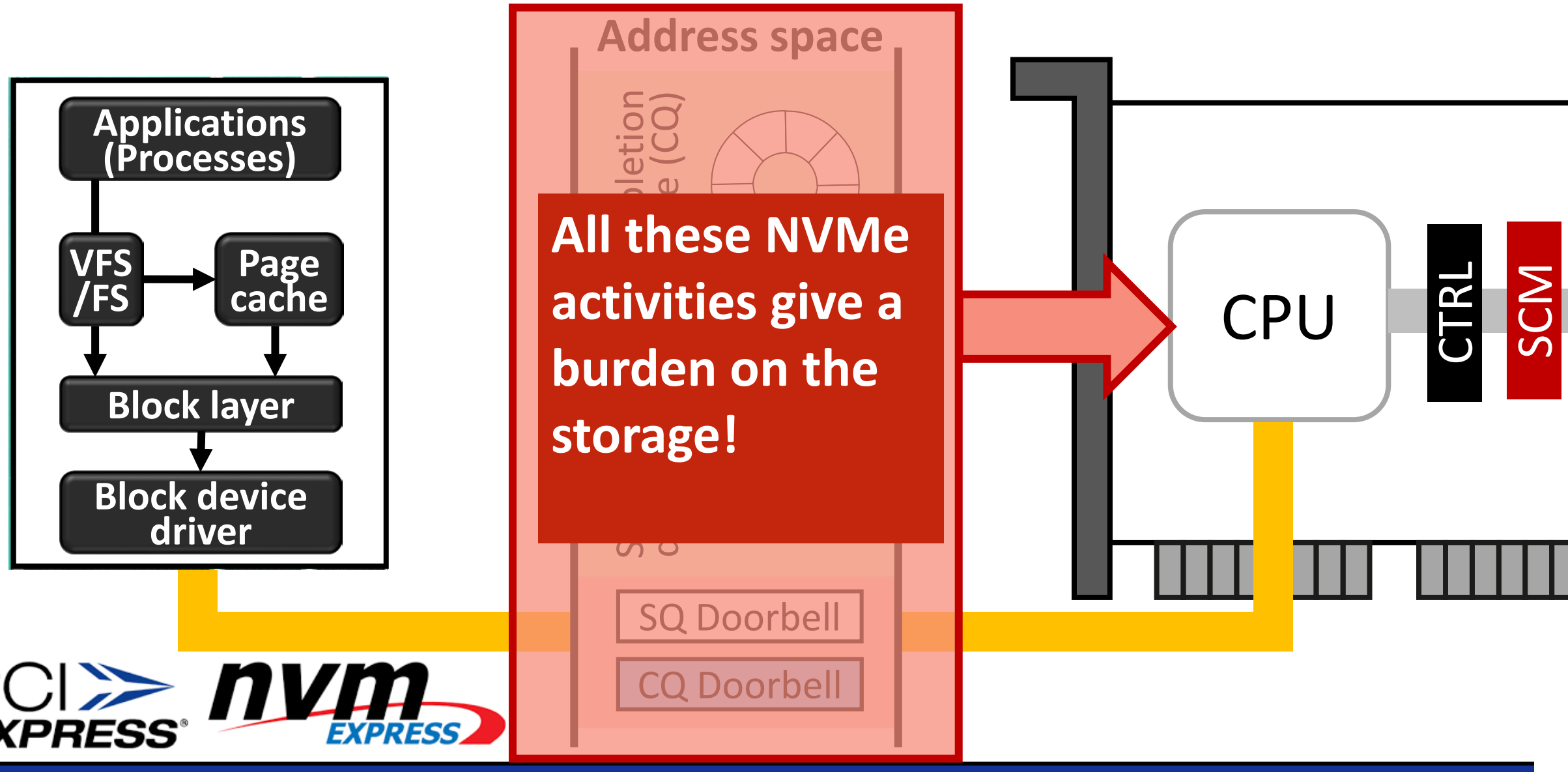
What Does SSD's CPU Do?



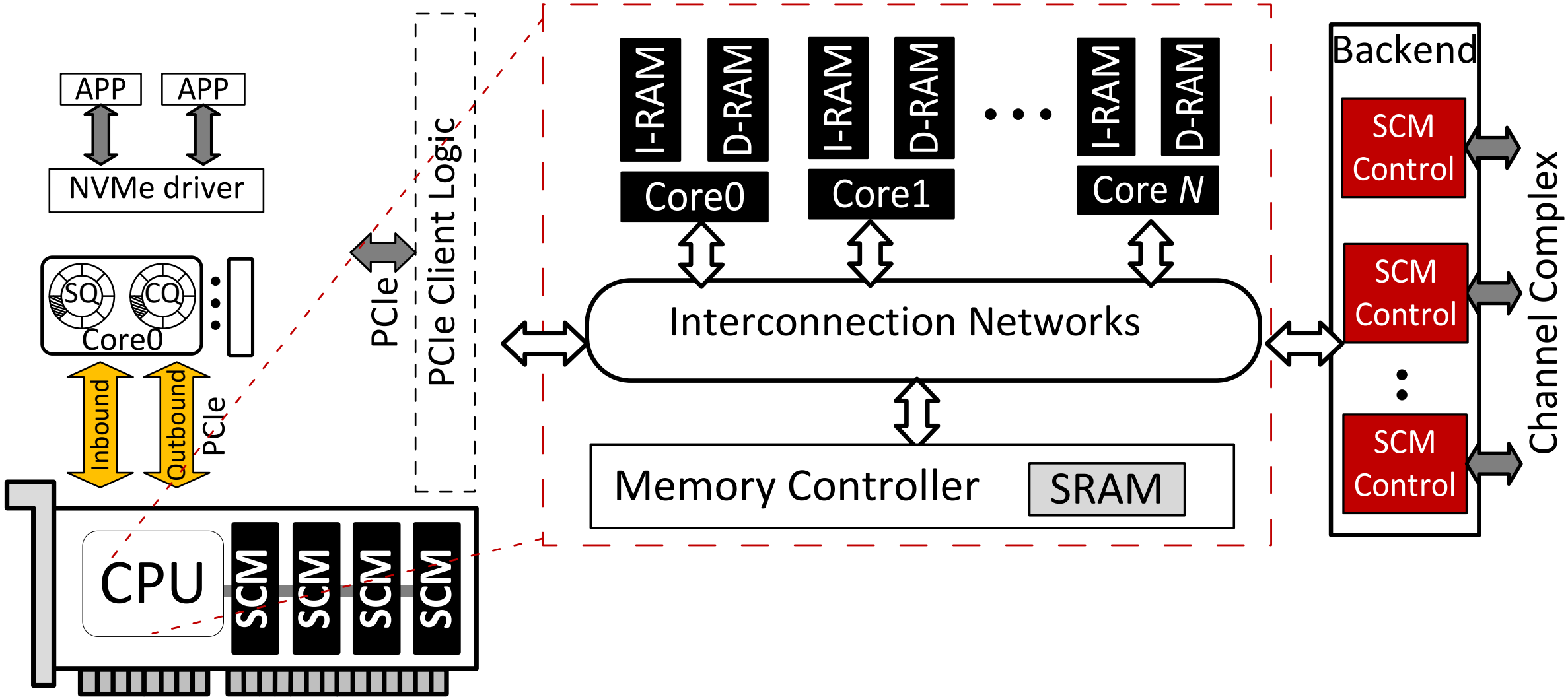
PCI EXPRESS

nvm EXPRESS

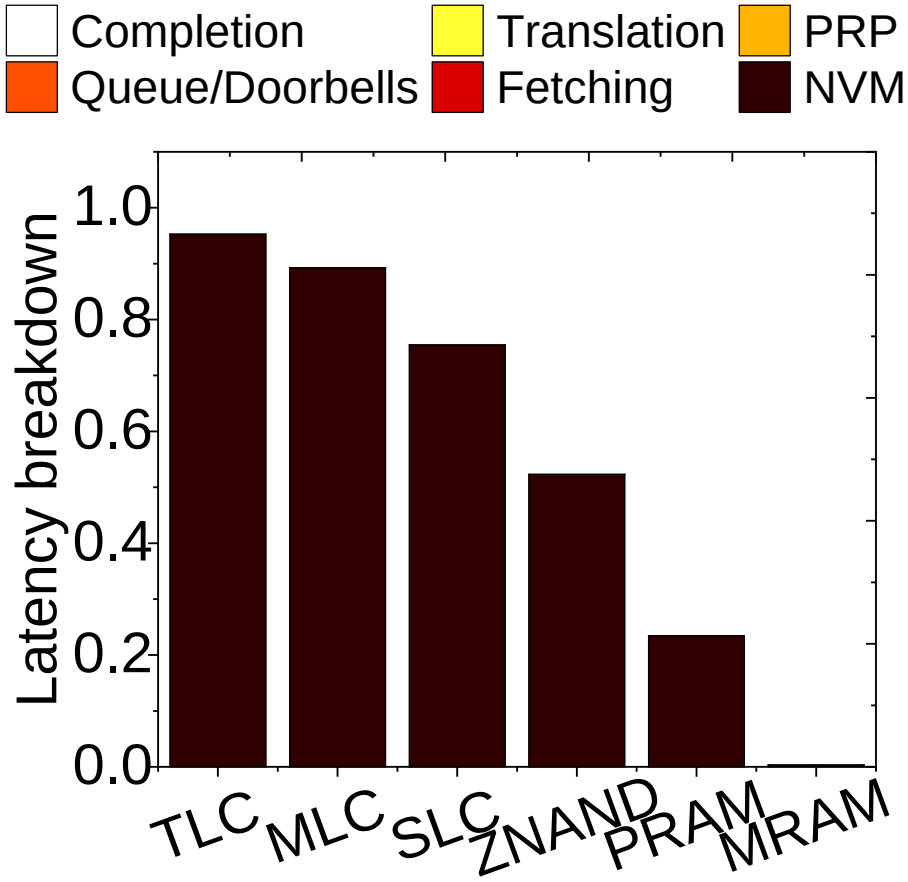
What Does SSD's CPU Do?

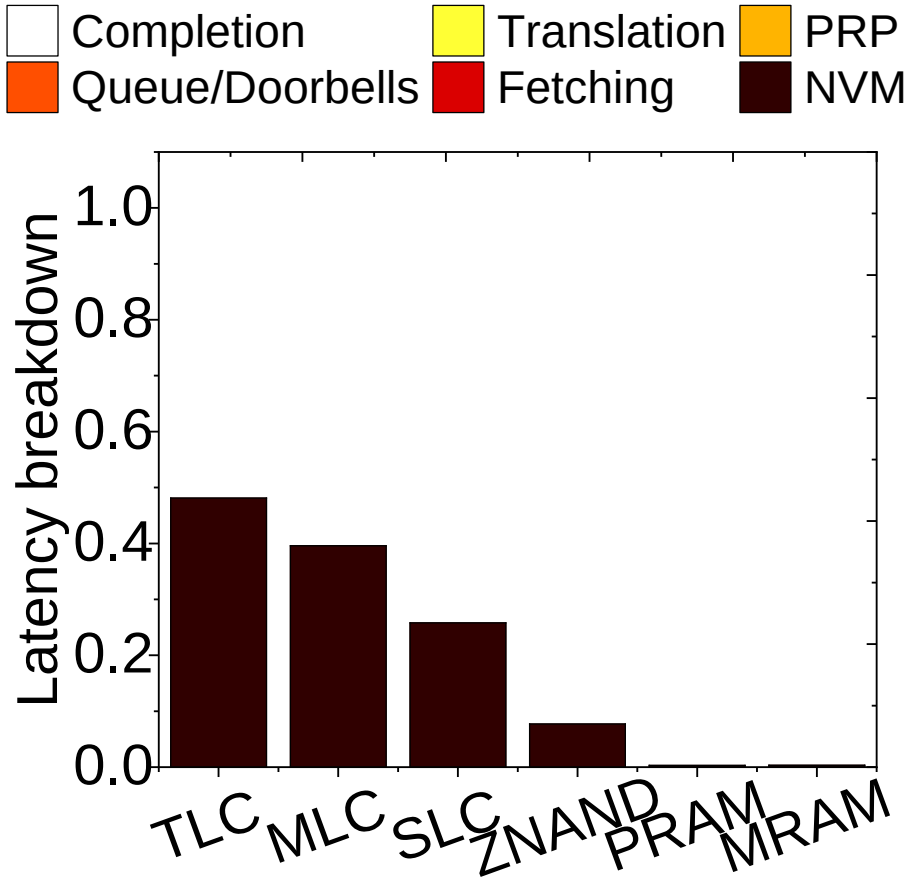


Multi-core IP for High-Performance SSD

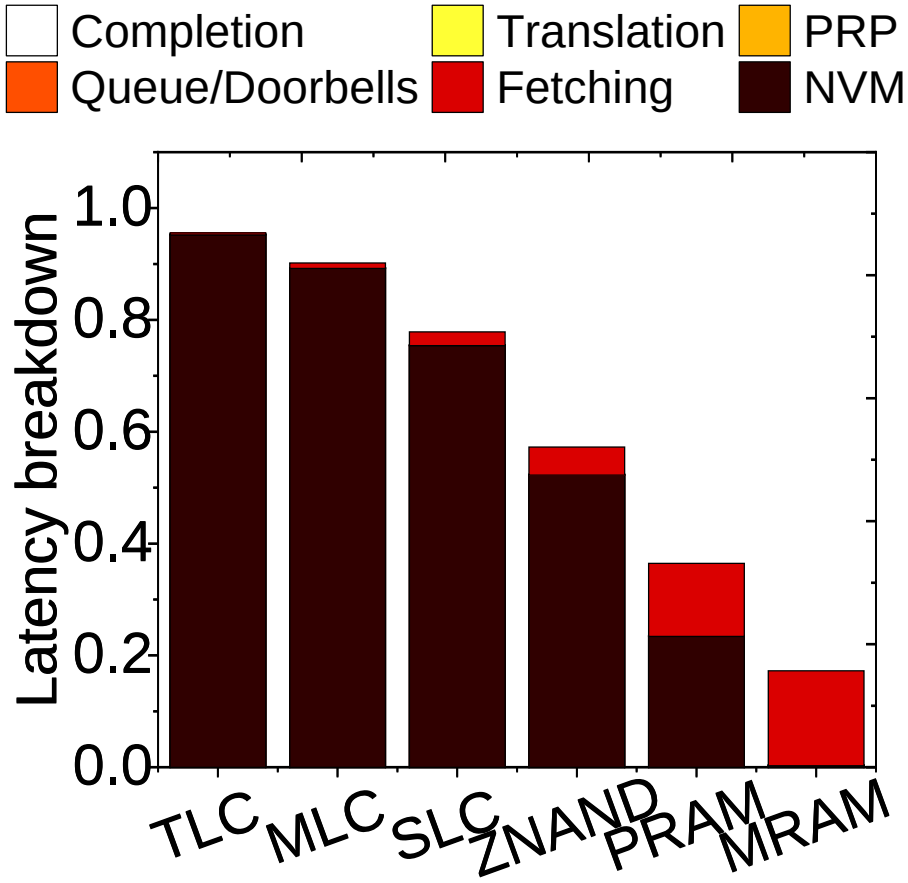


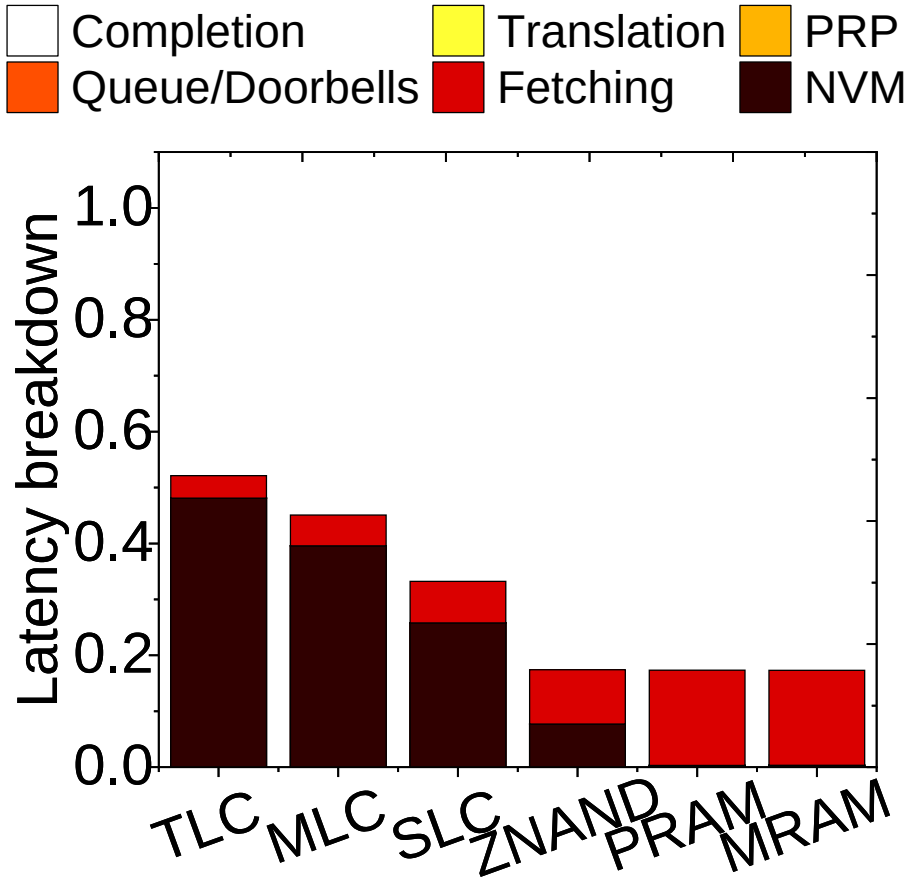
Component Latency Decomposition



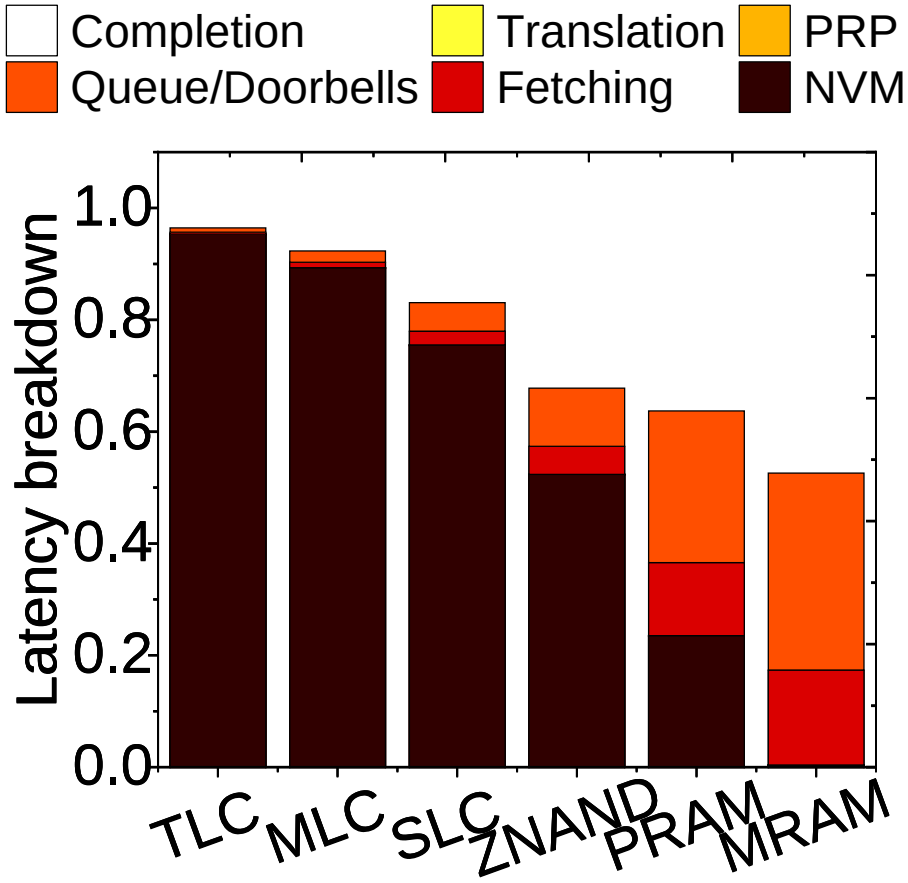


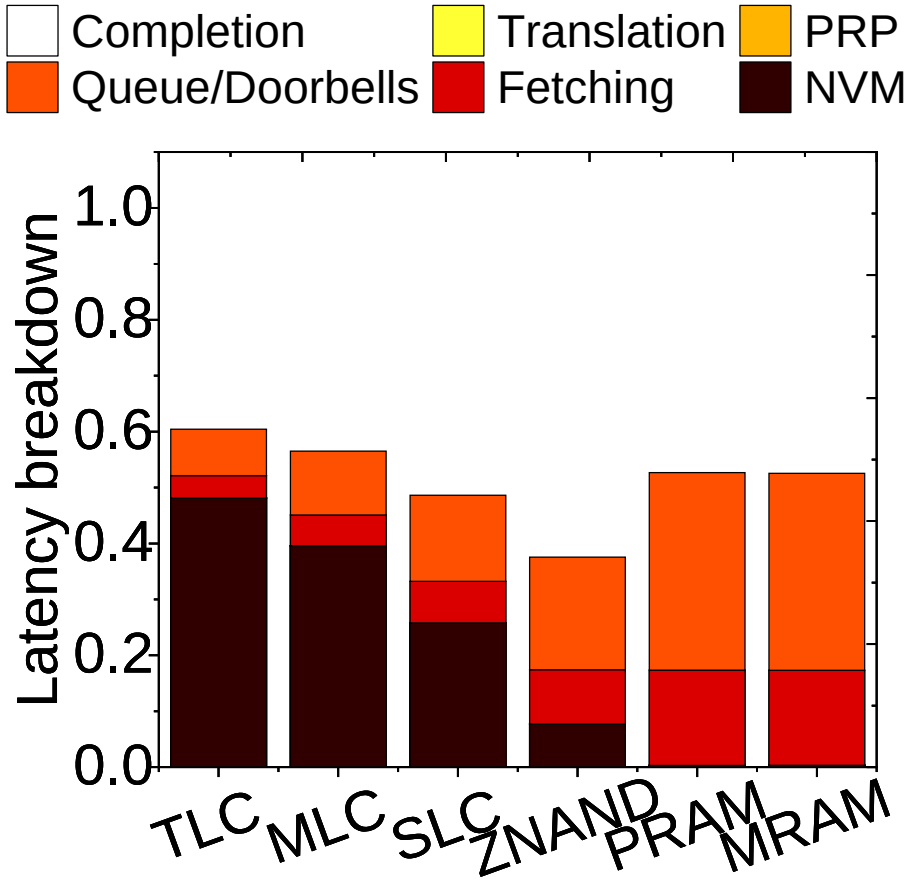
Component Latency Decomposition



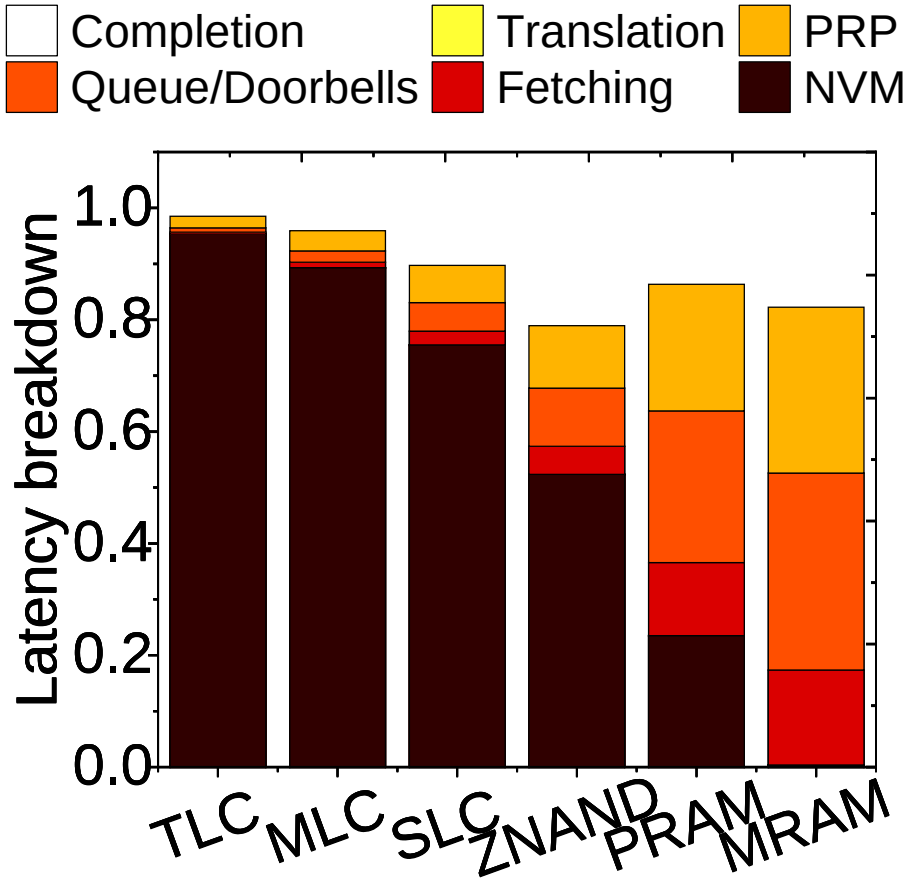


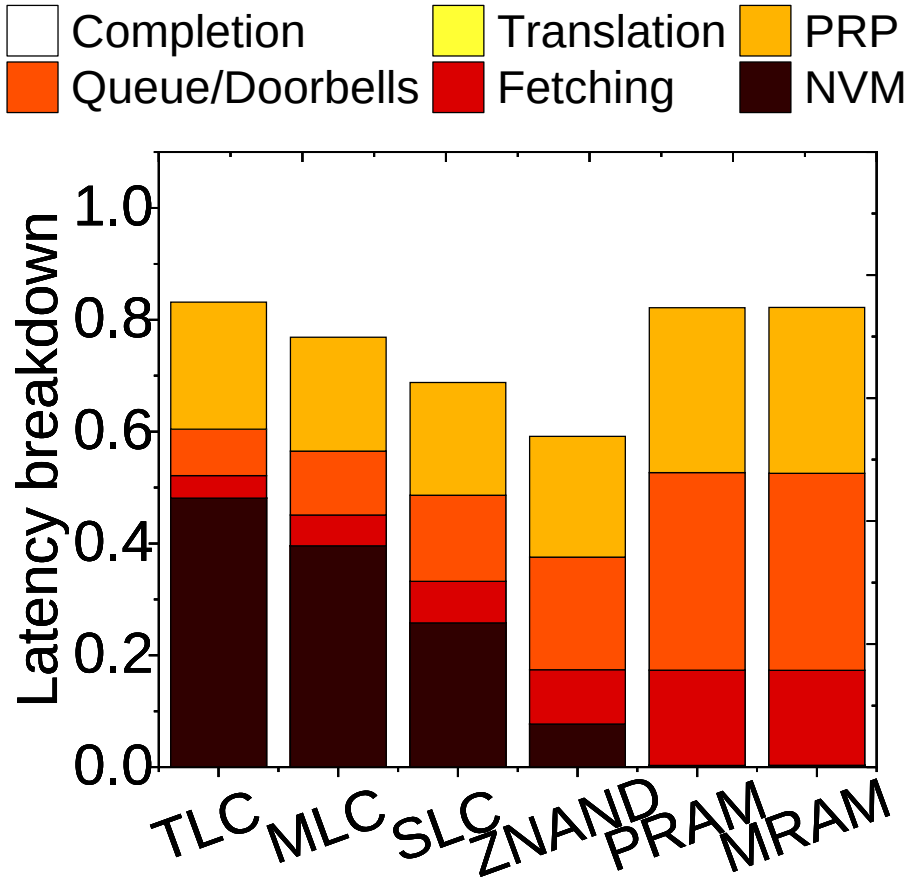
Component Latency Decomposition



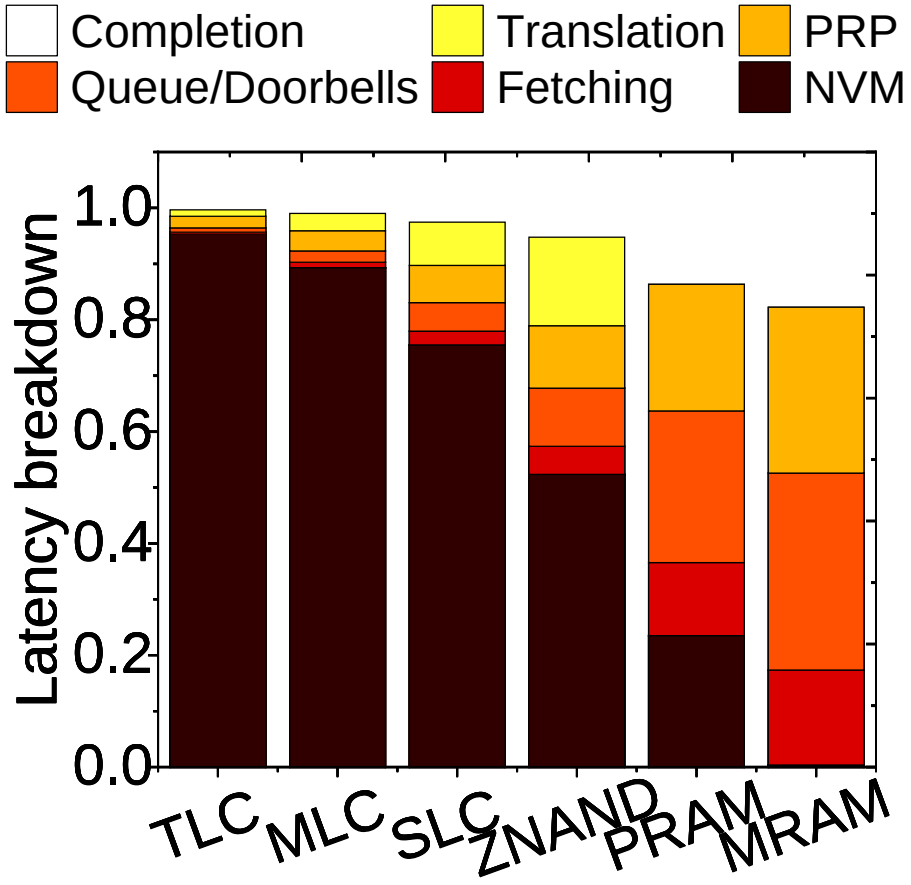


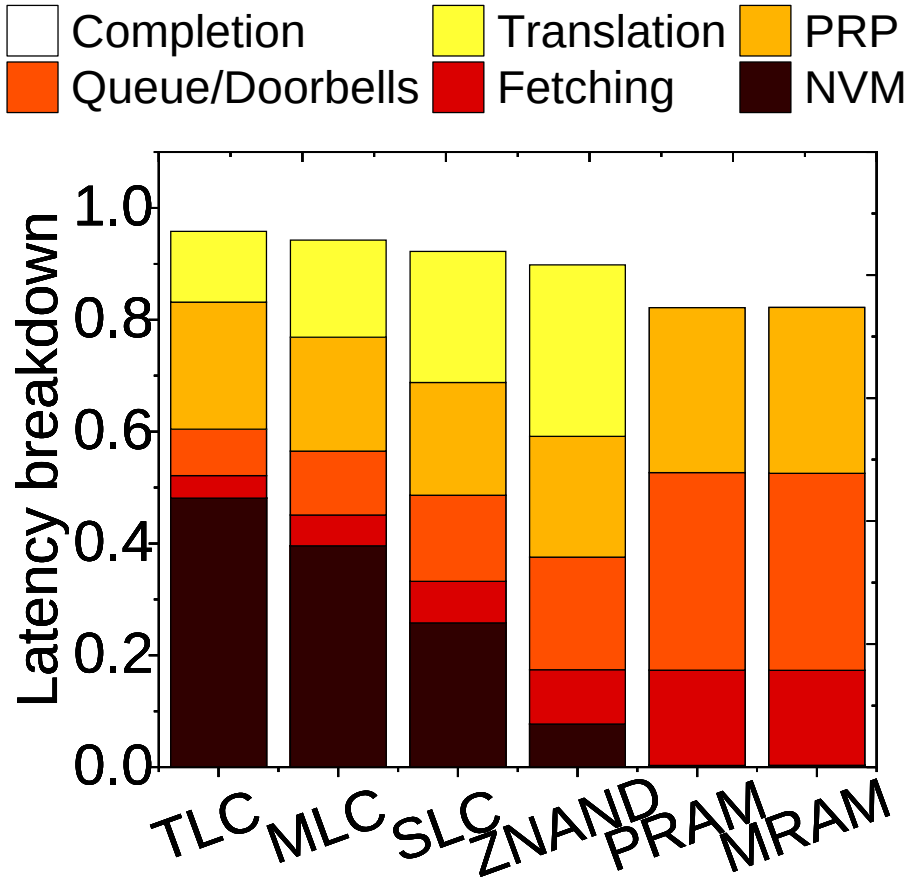
Component Latency Decomposition





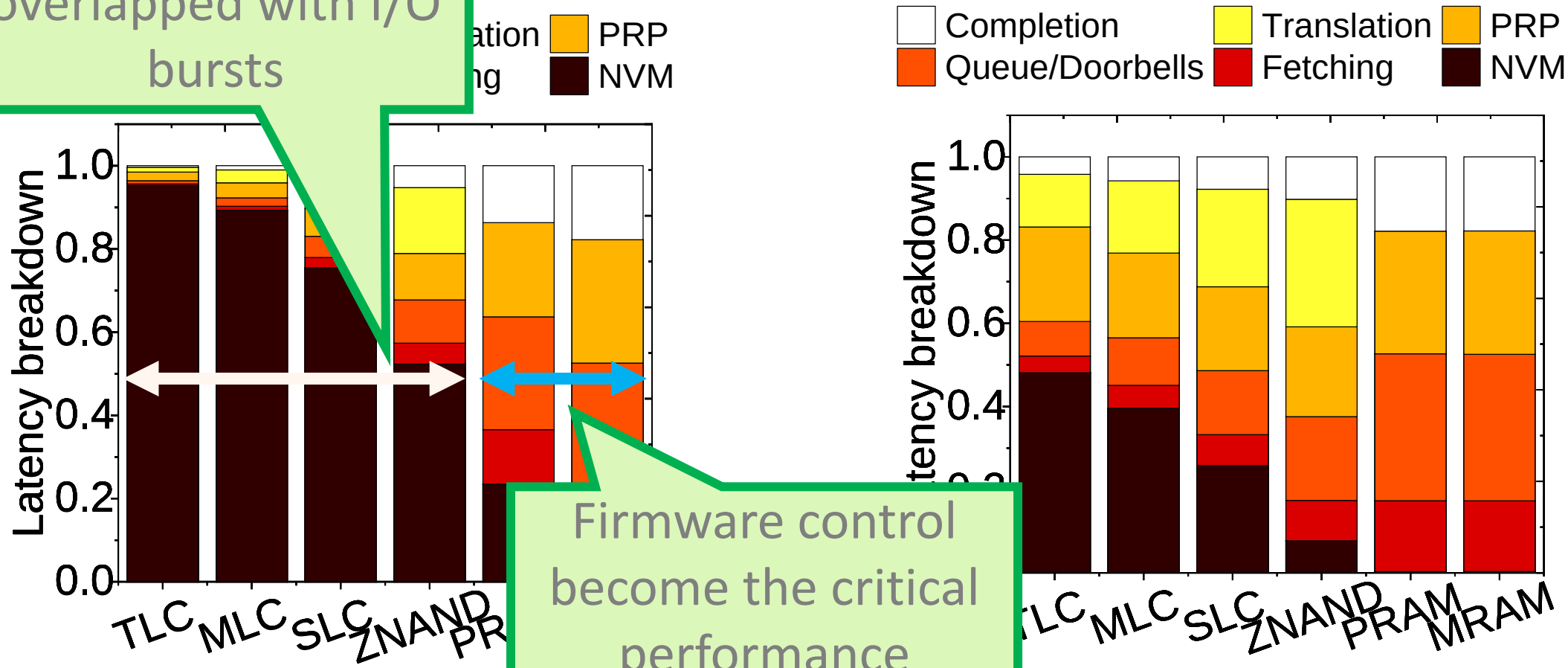
Component Latency Decomposition





Component Latency Decomposition

CPU bursts can be overlapped with I/O bursts

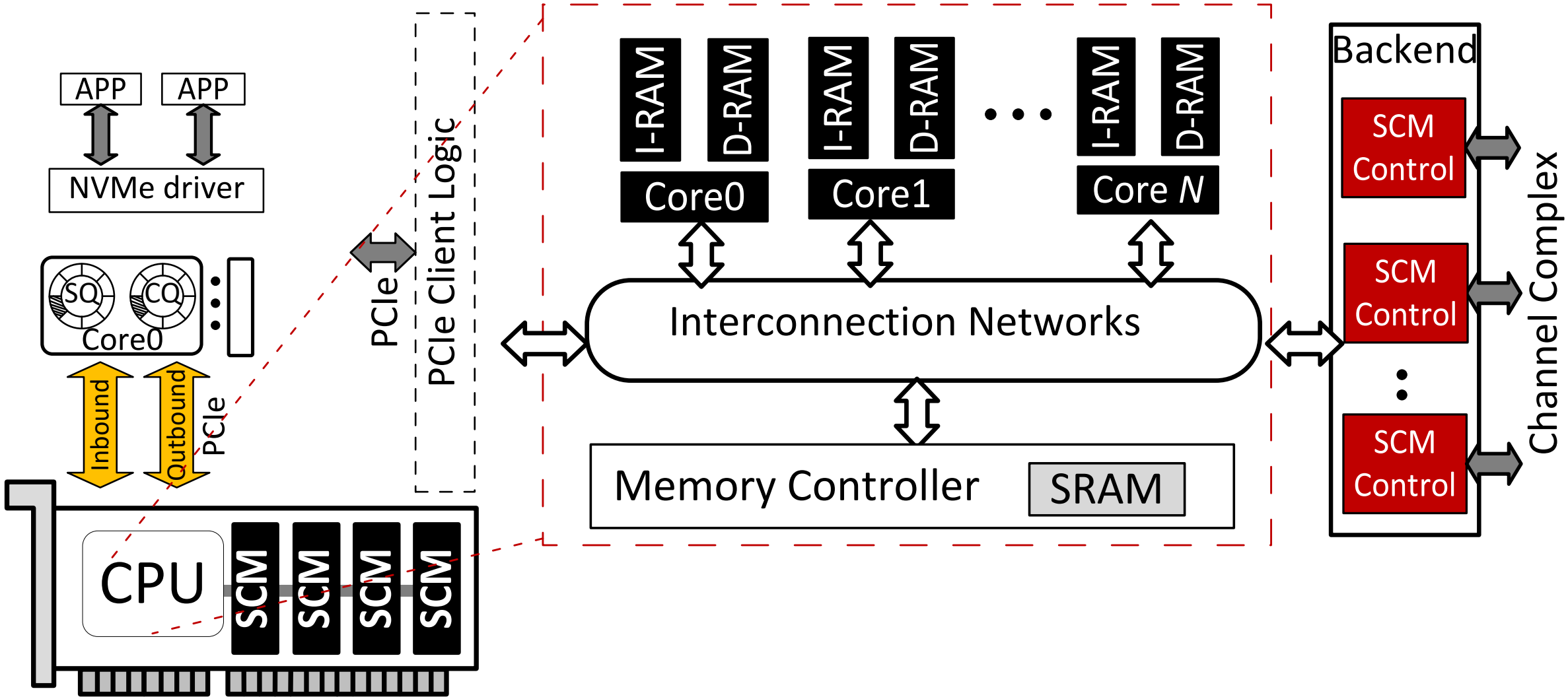


Firmware control become the critical performance bottleneck

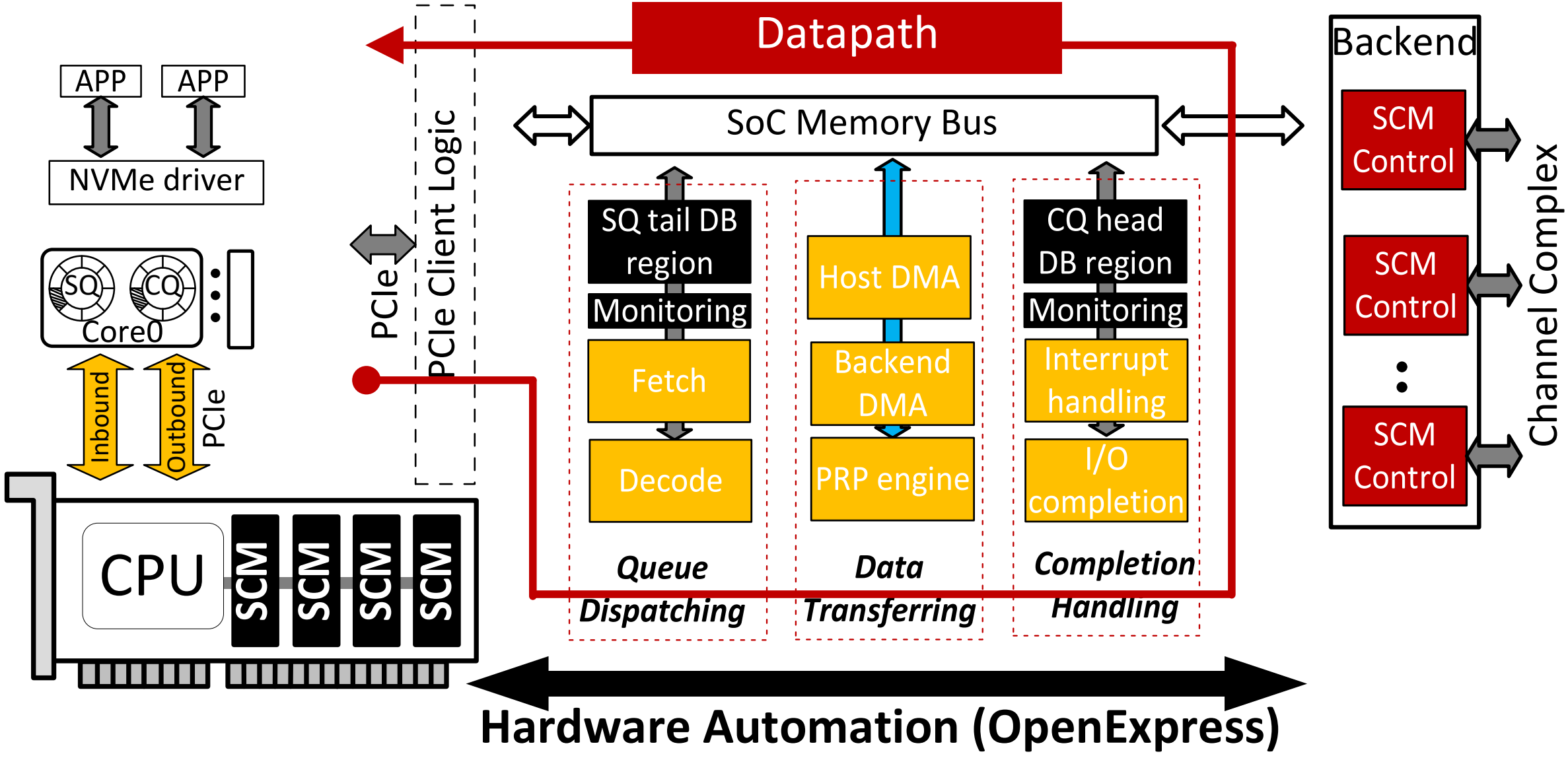
Overview

- A framework that fully automates NVMe control logic in hardware to make it possible to build customizable devices
  - OpenExpress is an NVMe host accelerator IP for the integration with an easy-to-access FPGA design.
- OpenExpress doesn’t require any software intervention to process concurrent read and write NVMe requests
  - It supports scalable data submission, rich outstanding NVMe commands and submissio n/completion queue management.
- We prototype OpenExpress on a commercially available Xilinx FPGA board and optimize all the logic modules to operate a high frequency.

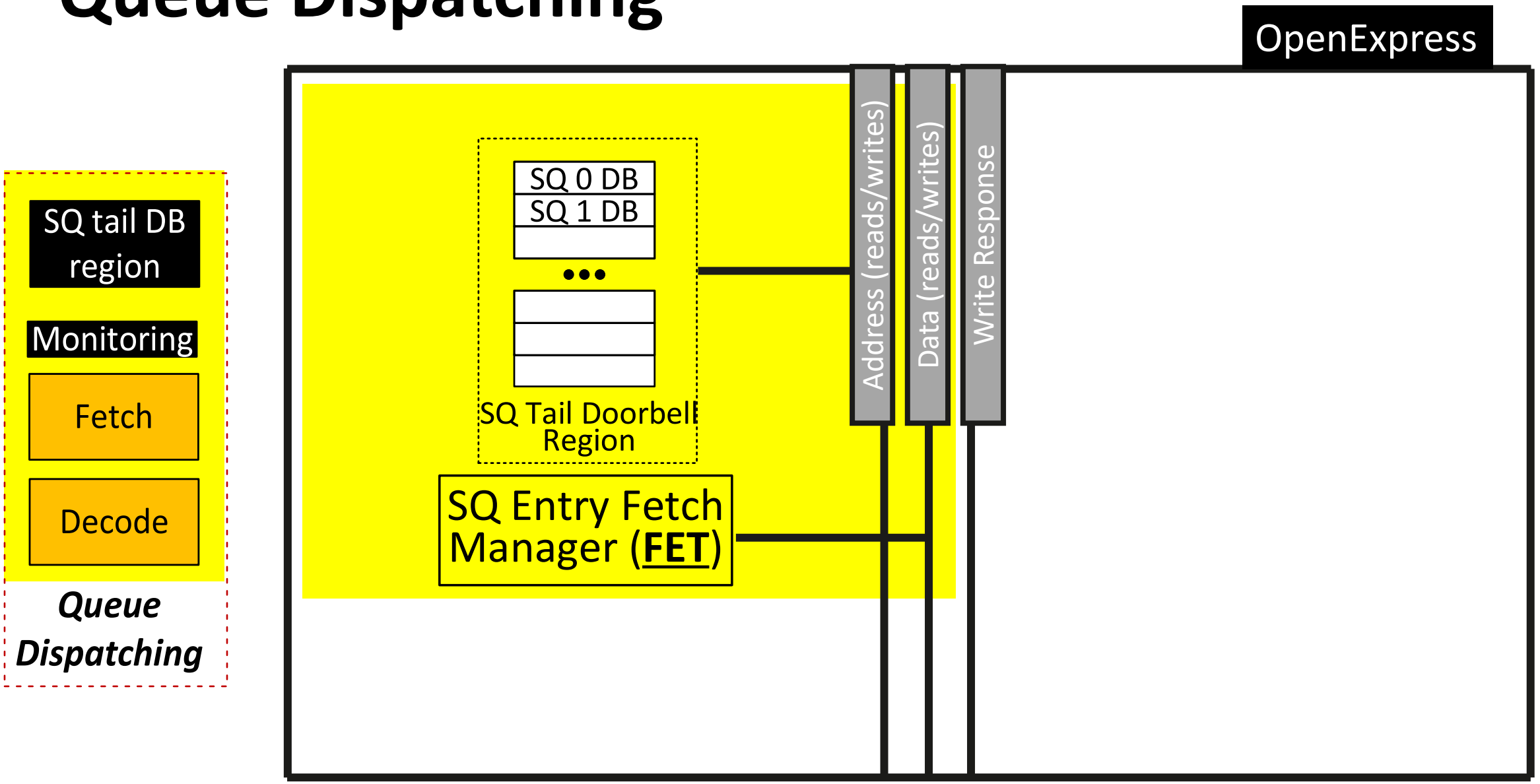
Full Hardware Automation for Critical I/O Path



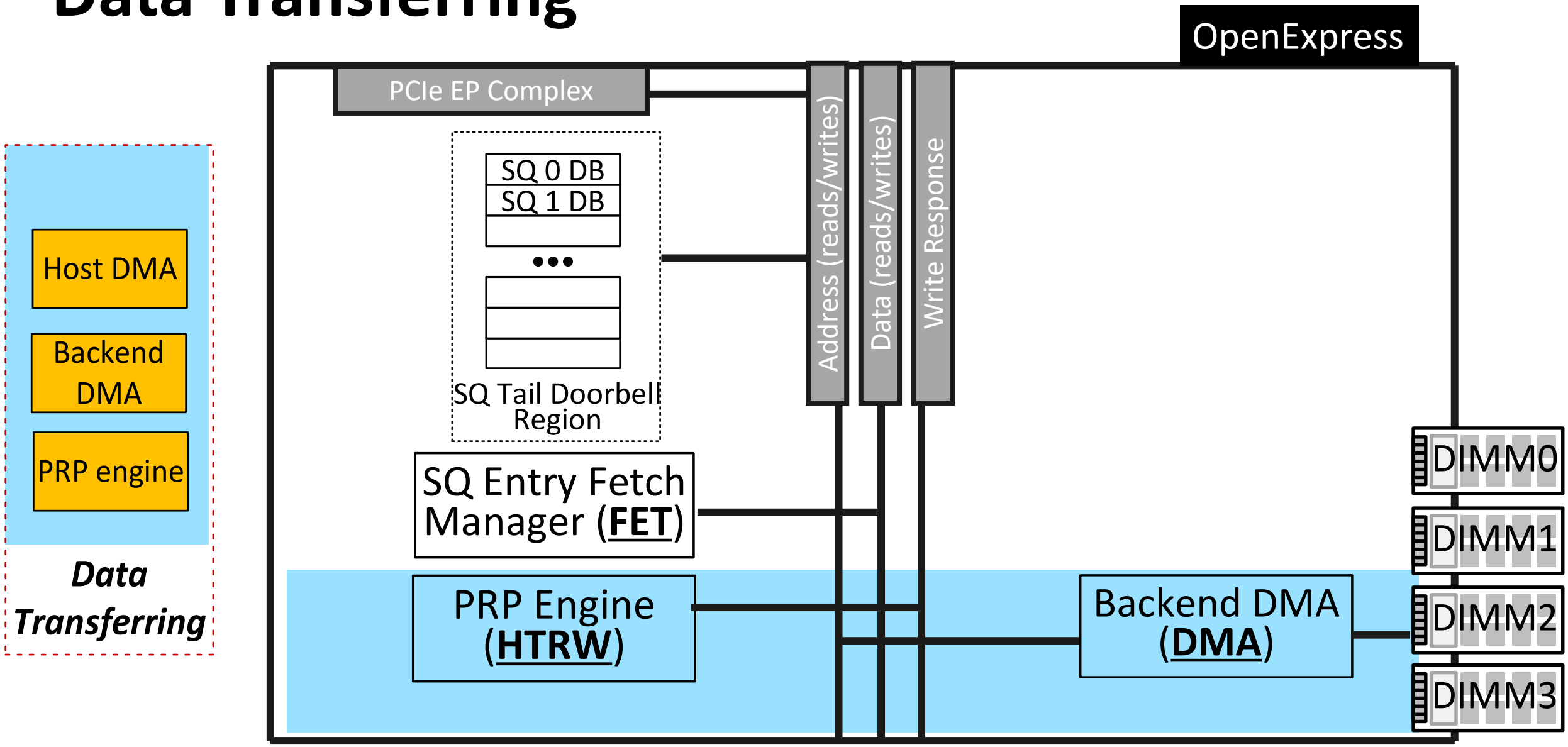
Full Hardware Automation for Critical I/O Path



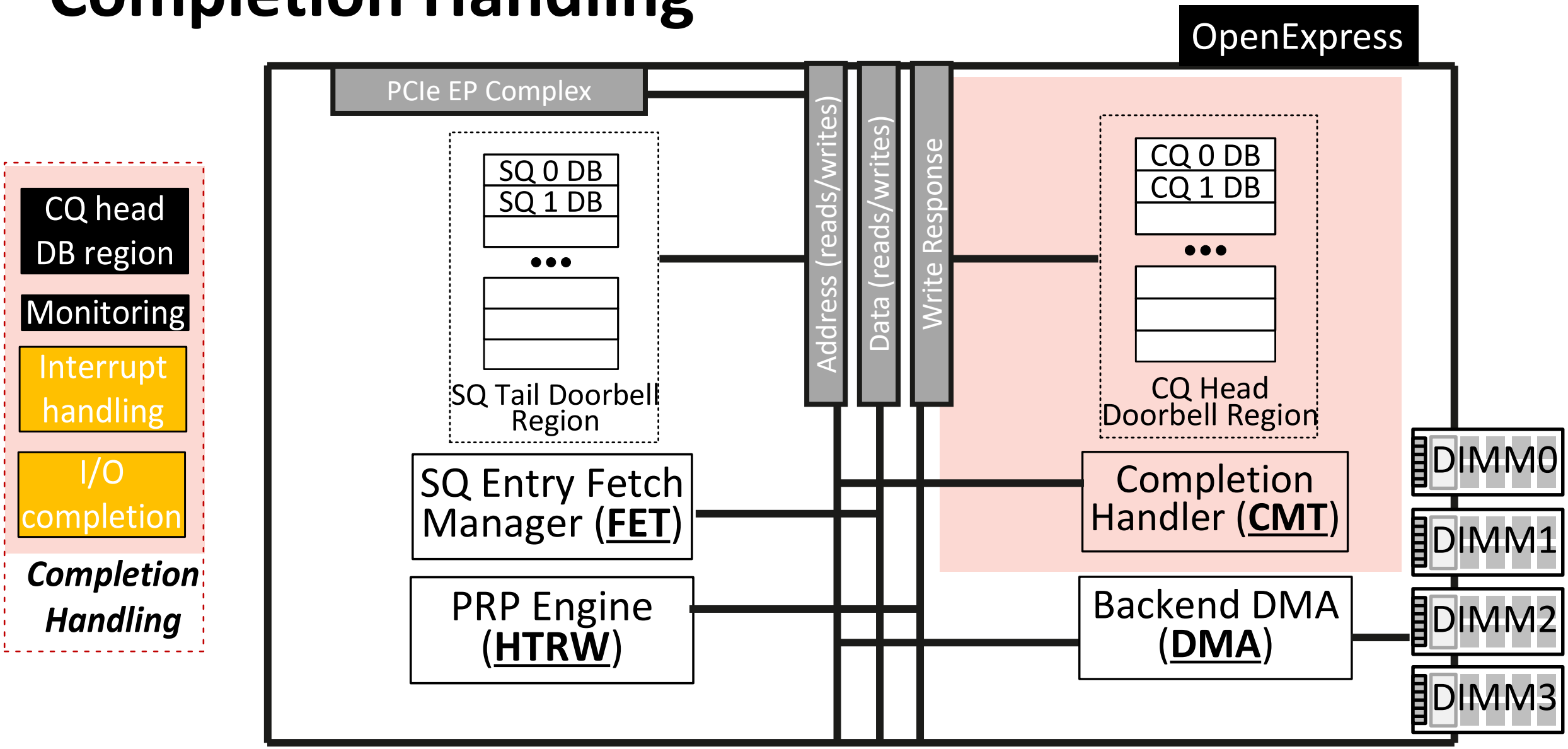
Queue Dispatching



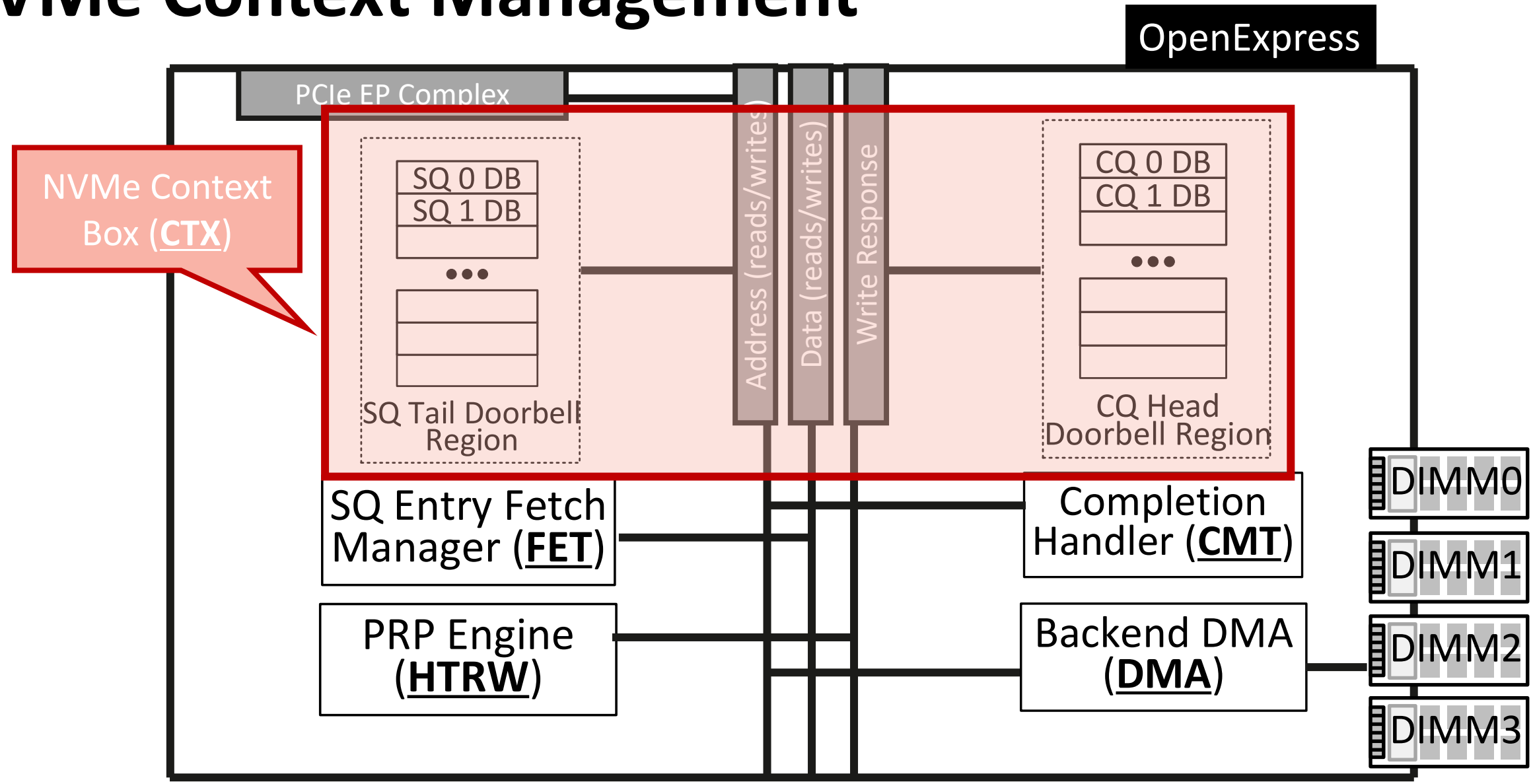
Data Transferring



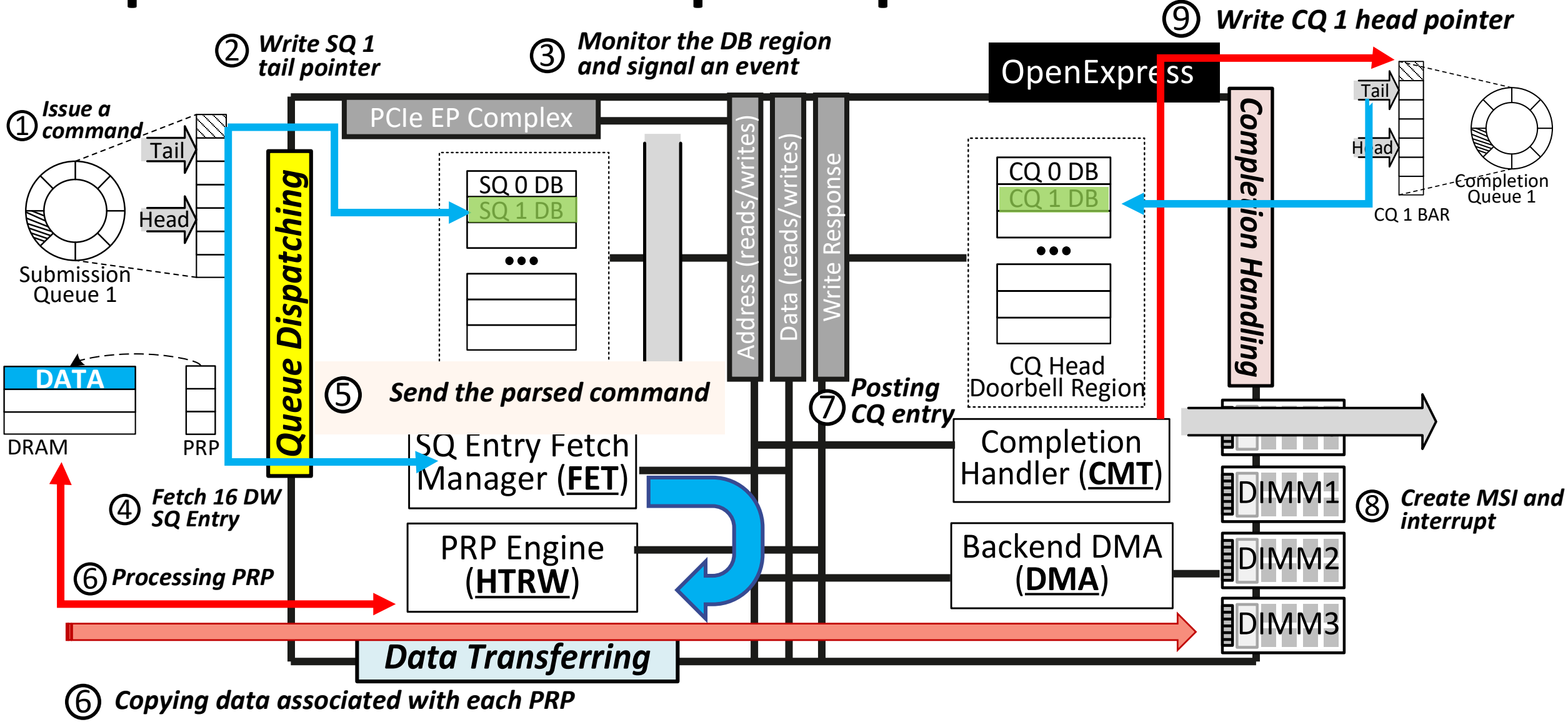
Completion Handling



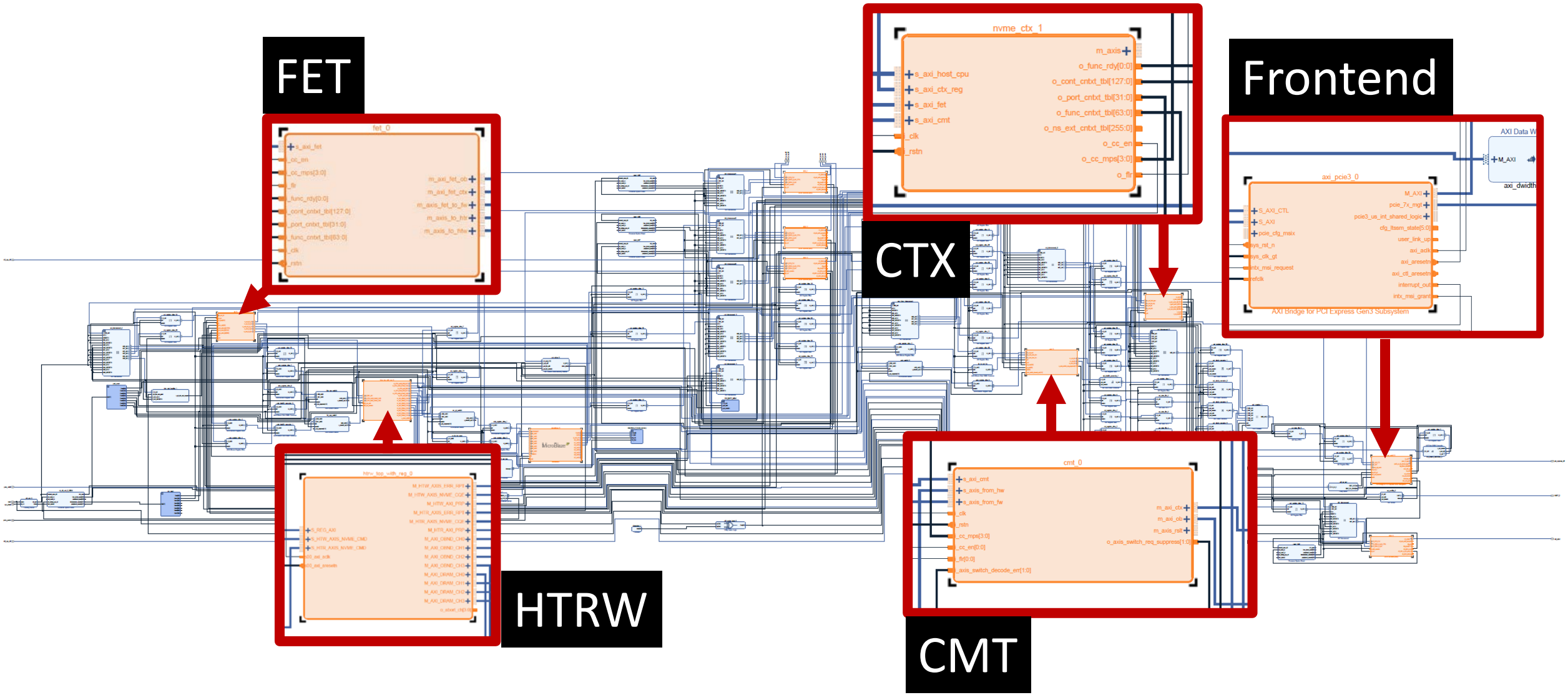
NVMe Context Management



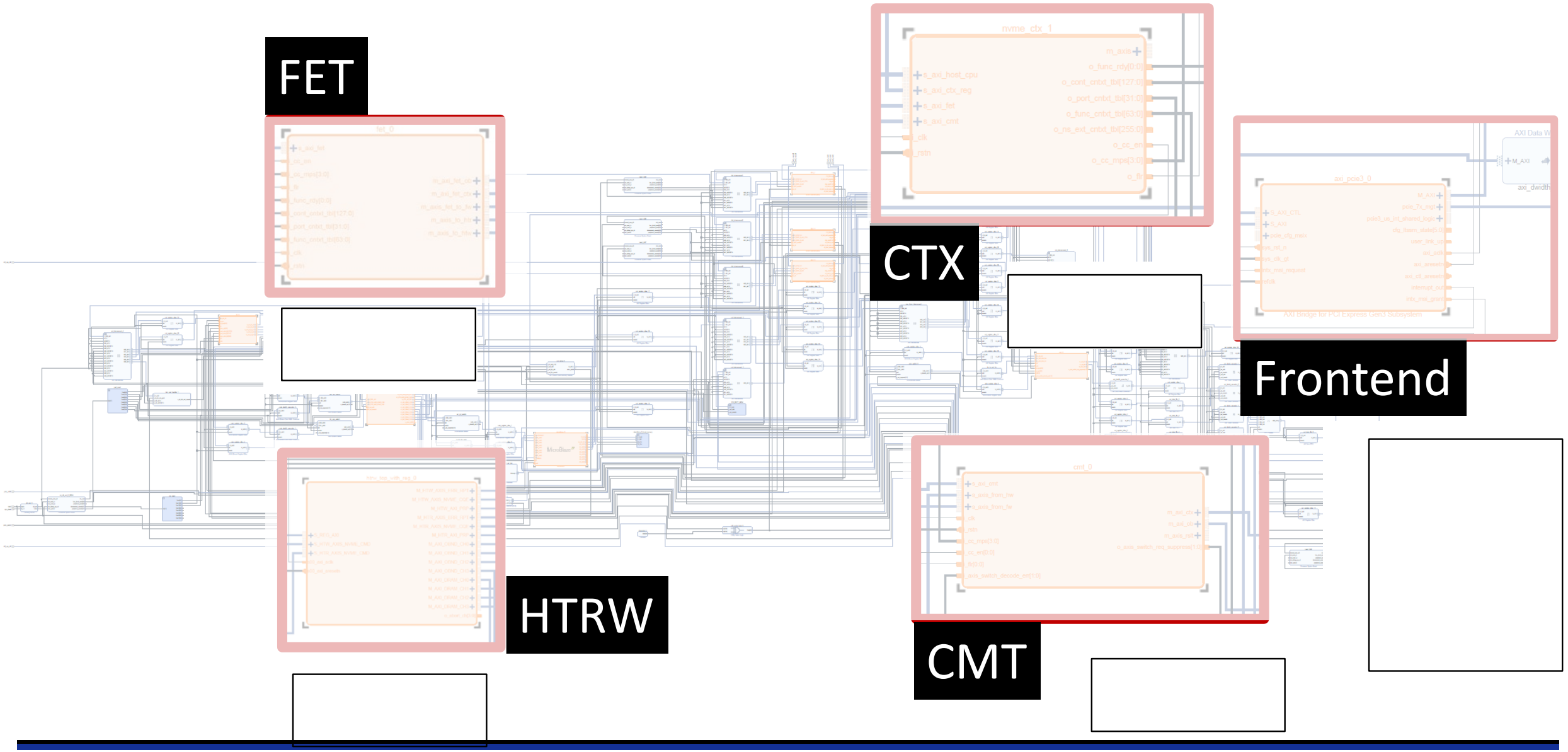
Operation Flow of OpenExpress



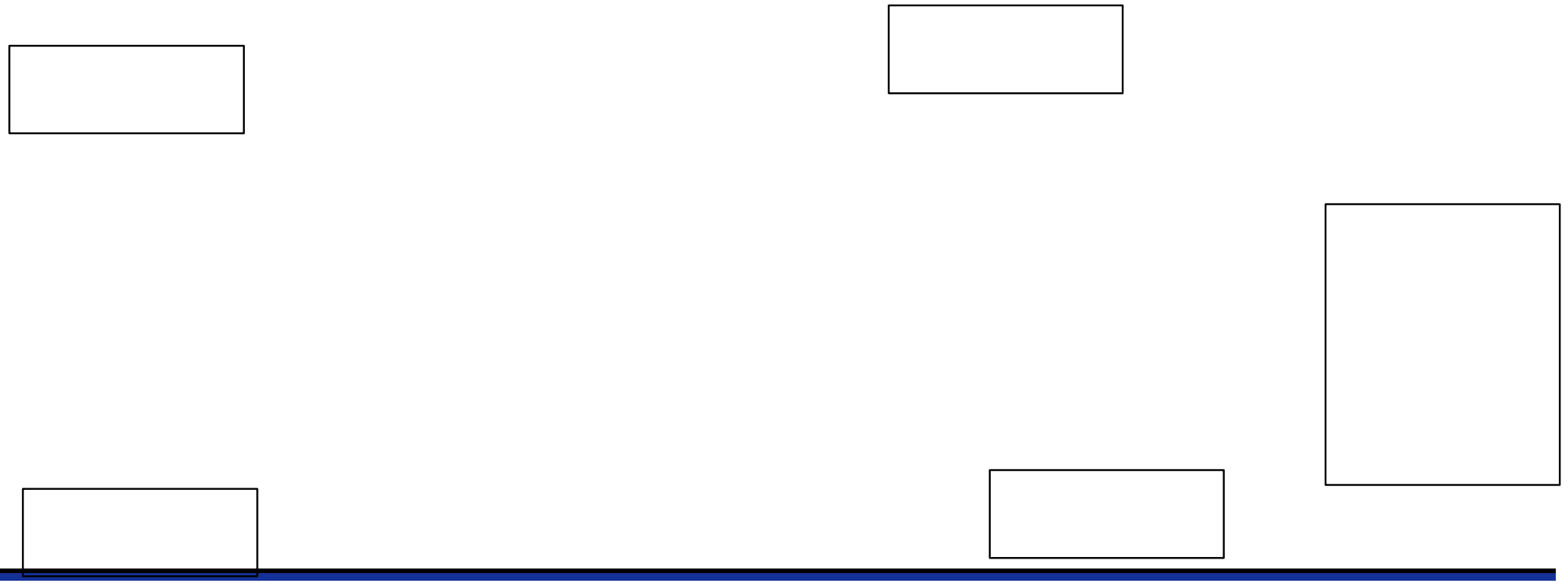
Major IP Cores



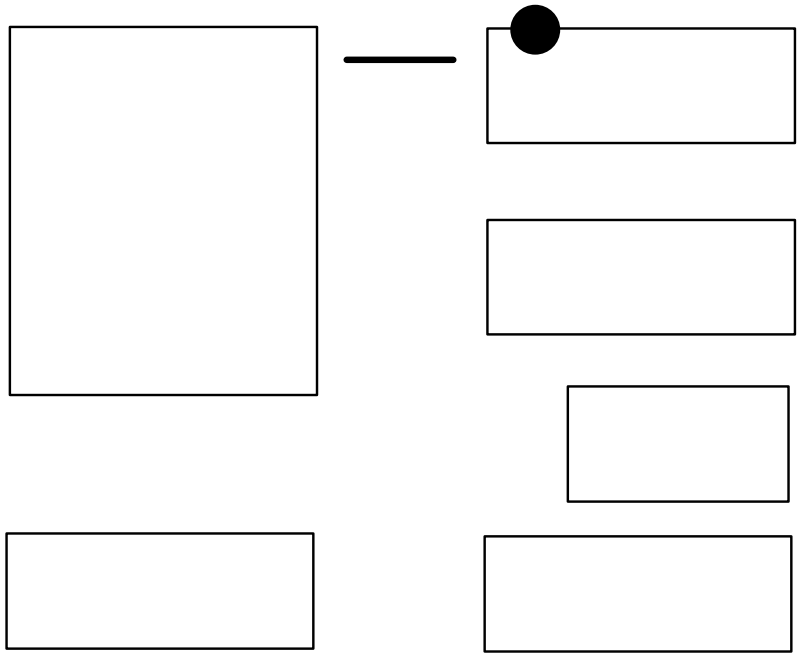
Major IP Cores



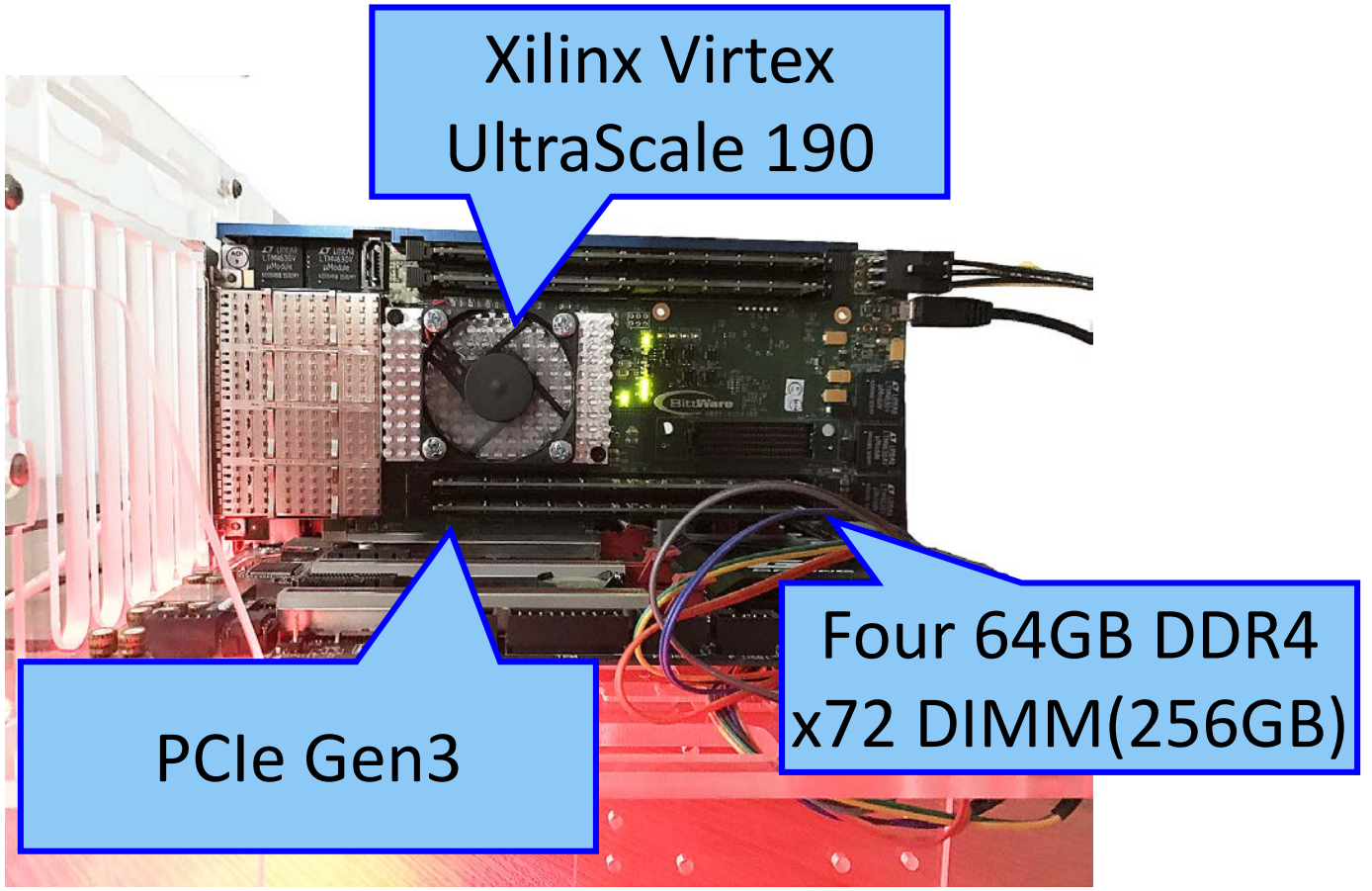
Major IP Cores



Synthesis and Implementation



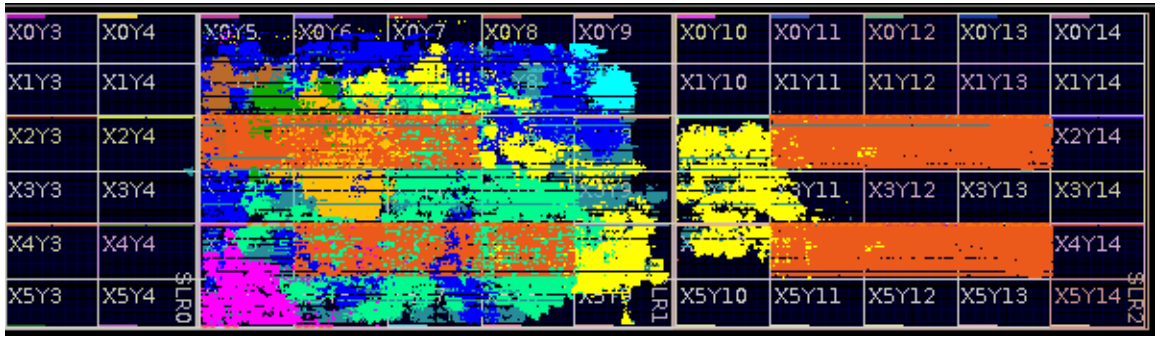
	Descriptions
CPU	i5-9400 2.9GHz
Design Suite	Vivado 2017.3.1
OS	Ubuntu 18.04 LTS
Building Time	More than 7 hours



Frequency Tuning (50MHz ~ 250 MHz)

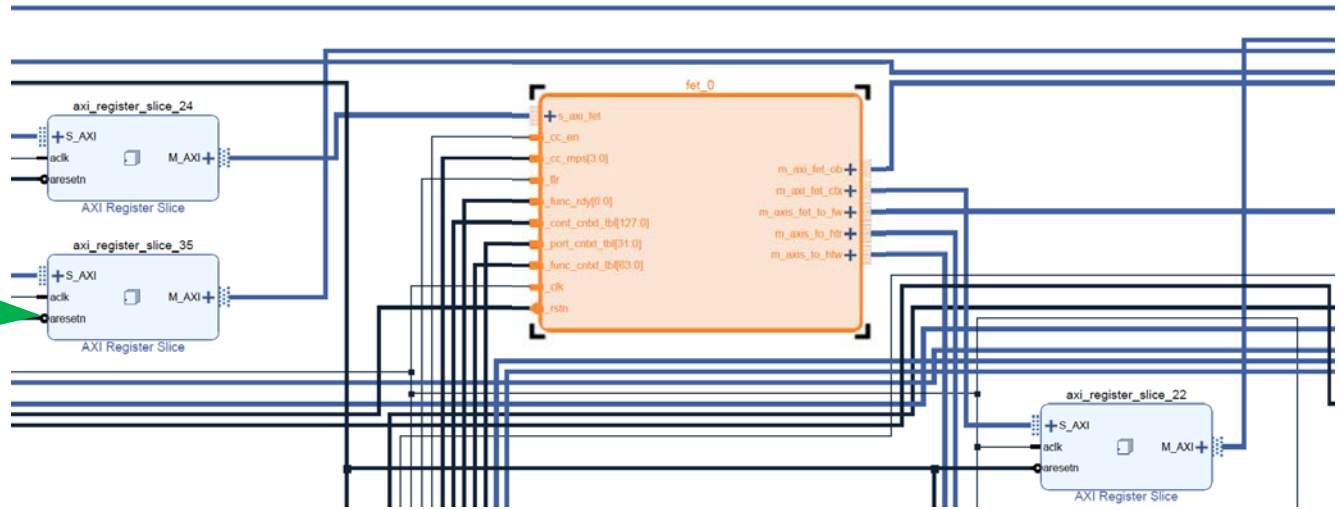
DMA	
PCIe EP	
CMT	
FET	
CTX	
HTRW	

Memory Controller	
DDR interconnect	
AXI	

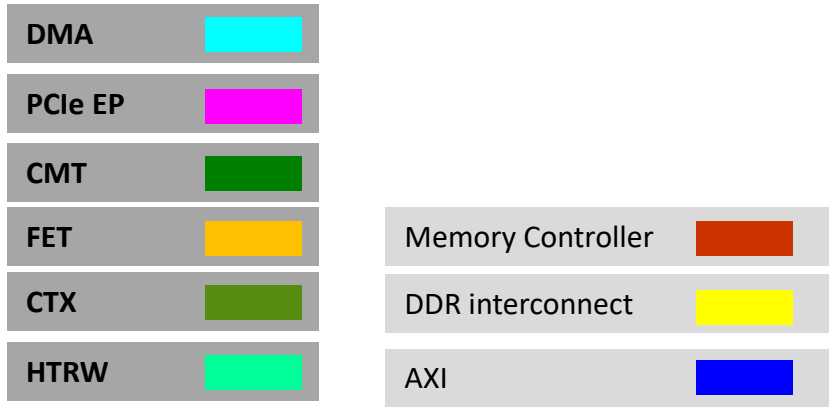


- Detect the long routing delay and place register slice IPs between hardware modules
  - It can increase the frequency while minimizing a negative slack
  - Gradual trial-and error to reduce the amount of route delay

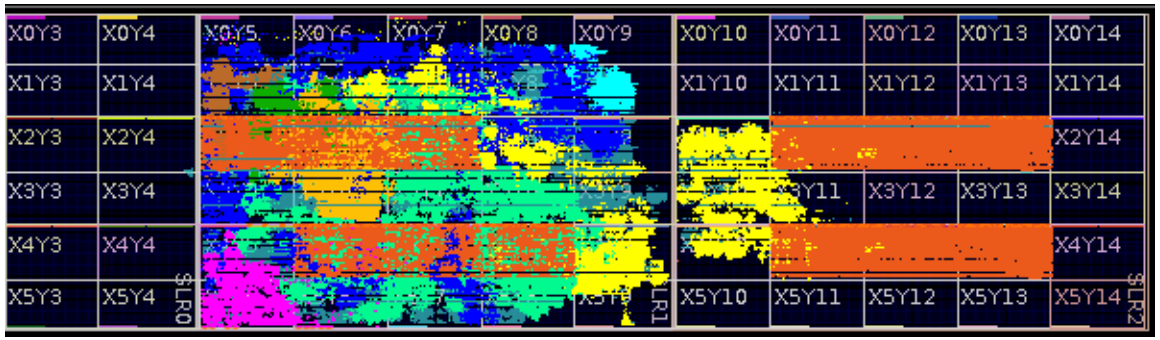
Putting register slides between different IP cores



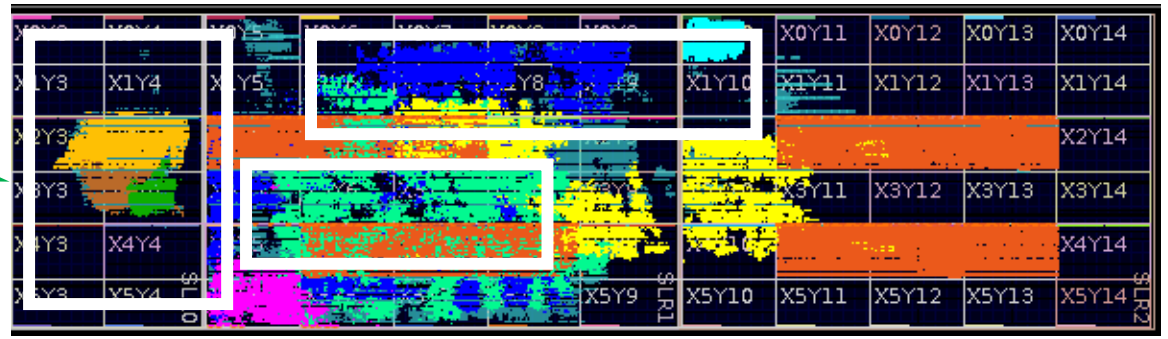
Frequency Tuning (50MHz ~ 250 MHz)



- Group hardware modules in the floor-planning
  - After synthesis, create FPGA pblock (a part of grid cells), allocate hardware IPs to the pblock, and do PNR (place and route)
  - Check the implementation report and do gradual trial-and-error again

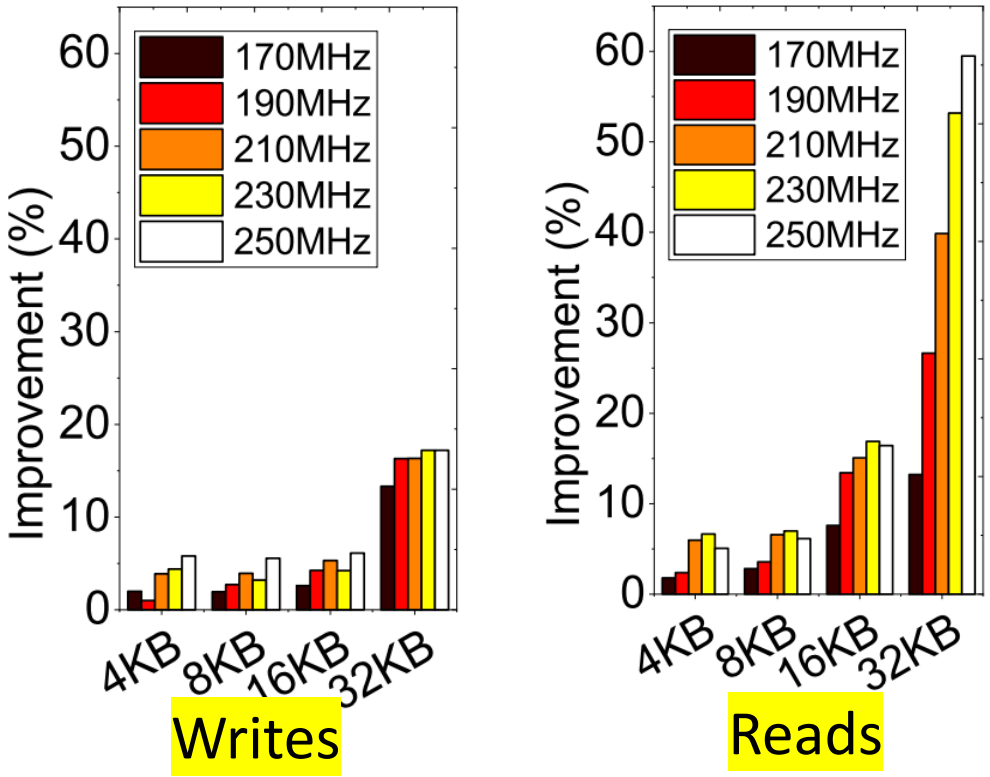


Reducing route distances by grouping IP logic

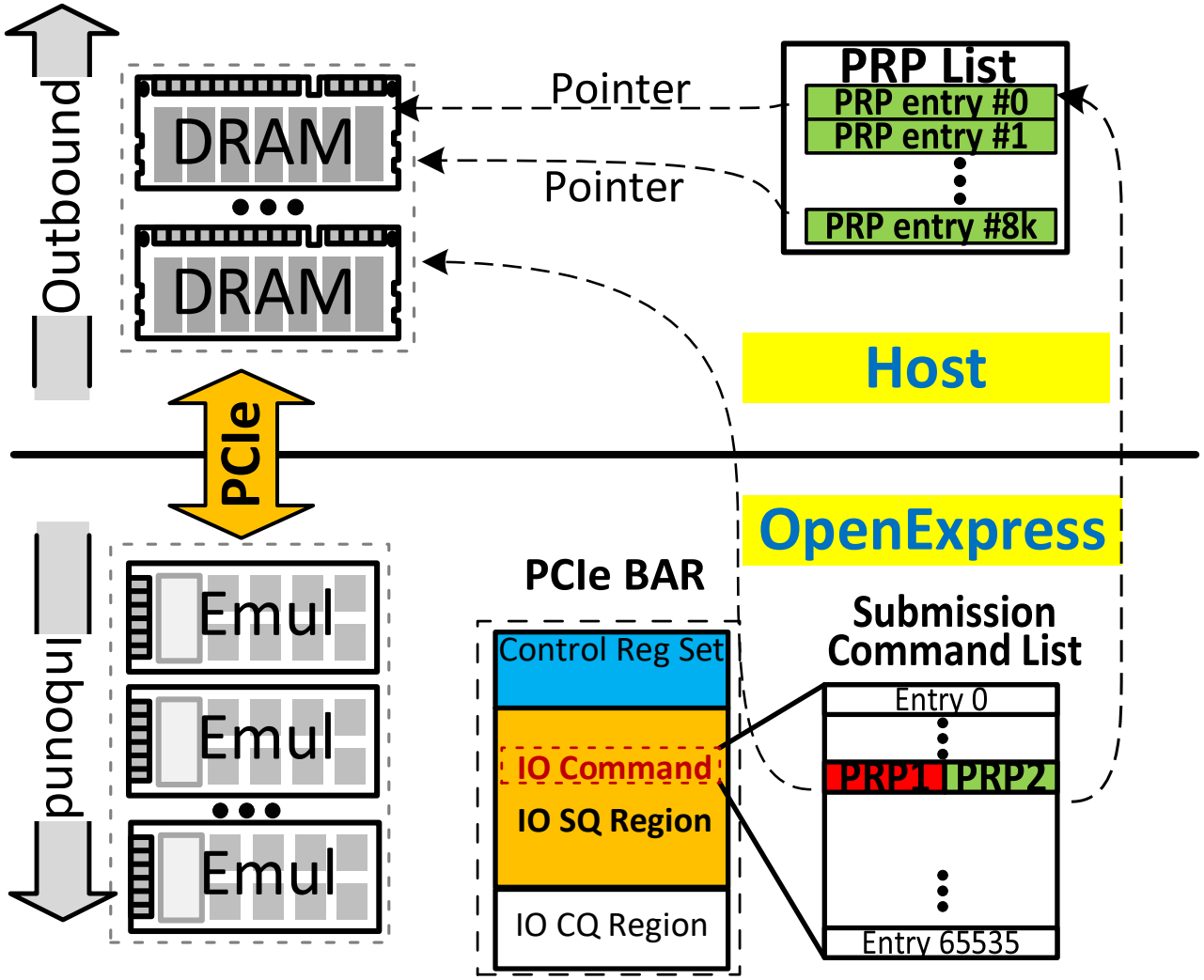


Perf. Improvement of Frequency Tuning

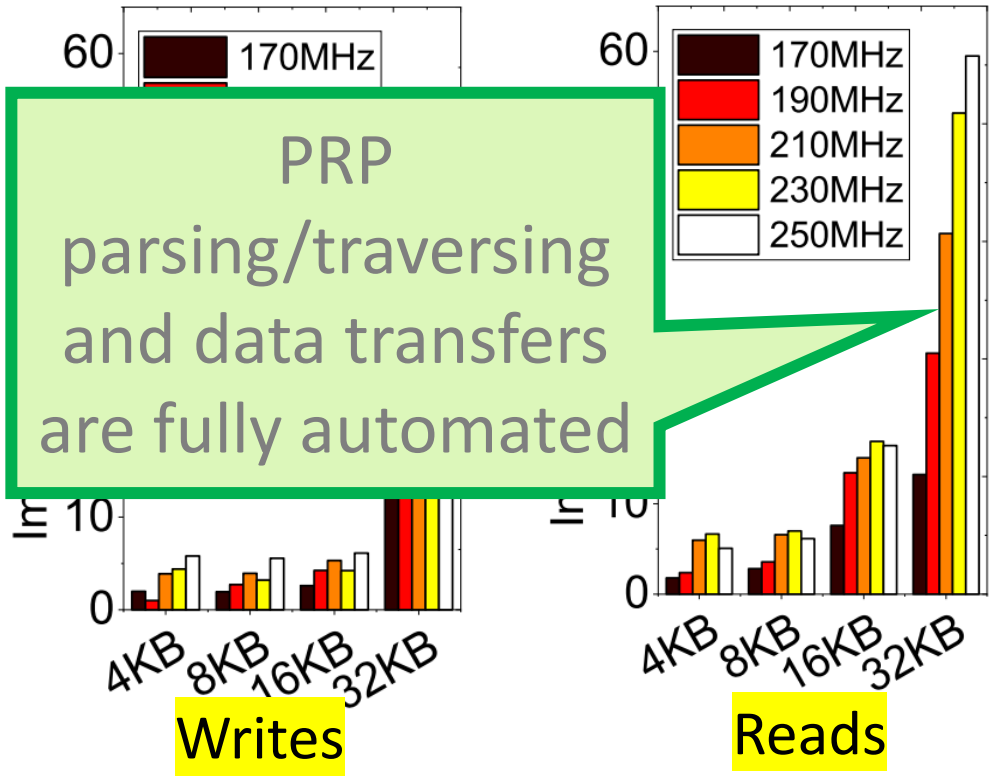
- The performance improvement for reads and writes is as high as 20% and 60%, respectively
- Reads exhibit more benefits as its nature of data movement on PCIe packets (explained shortly)
- The large size block processing has more benefits because of automated PRP data processing



Perf. Improvement of Frequency Tuning



PRP parsing/traversing and data transfers are fully automated

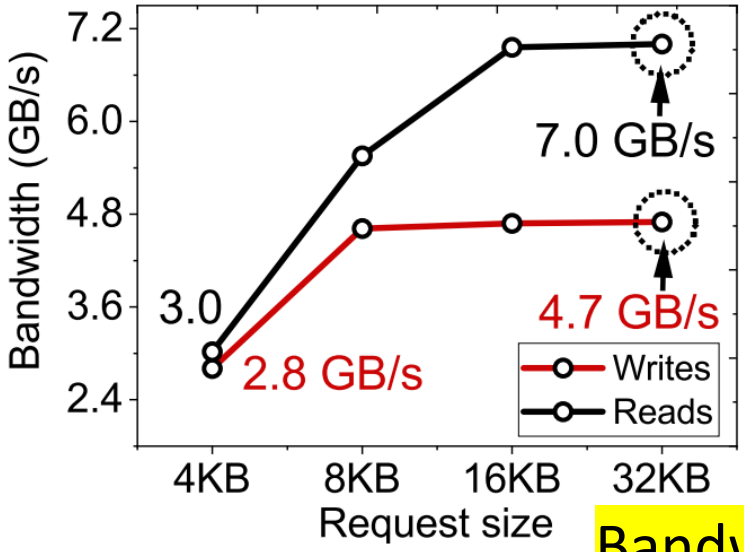


Evaluation

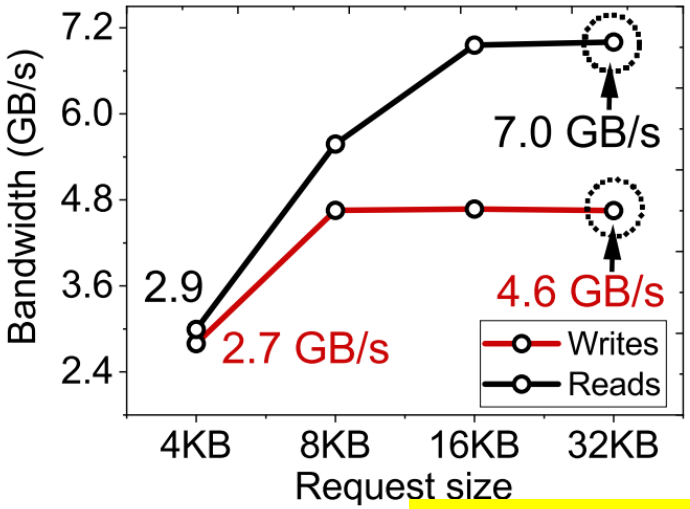
- For all executions, a single I/O worker cannot extract full bandwidth (CPU bottleneck)
  - We execute 10 threads for both benchmark execution and real workload
- Evaluations demonstrated in the paper are all performed with a queue-depth “8”
  - The queue depth exhibiting the best bandwidth
- Note that we don’t claim that OpenExpress can be faster than other fast NVMe devices.
  - Instead, the evaluation shows that an FPGA-based design and implementation for NVMe IP cores can offer good performance to make it a viable candidate for use in storage research.

	Descriptions
CPU	8-core 3.3GHz Intel Skylake-X Server Microarchitecture
DRAM	32GB DDR4 (2666)
Benchmark	FIO
Real workload	CAMEL’s Open Storage Trace (SNIA) https://trace.camelab.org/
I/O Workers	10 threads
Block Device	Optane SSD P4800X

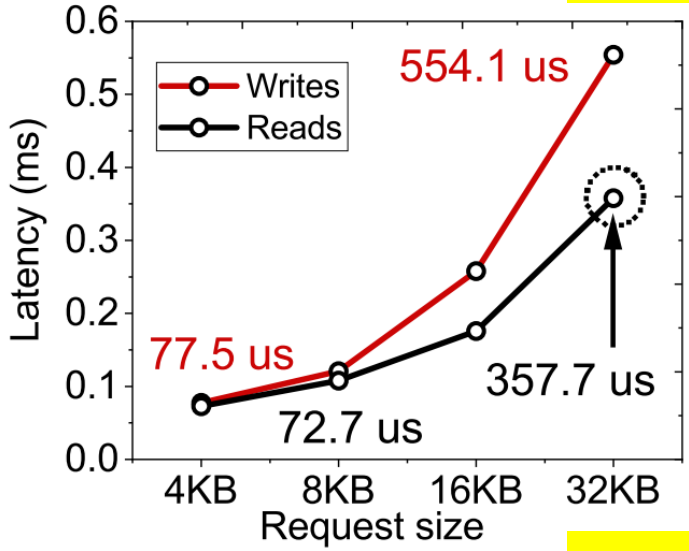
Performance w/ Microbenchmarks



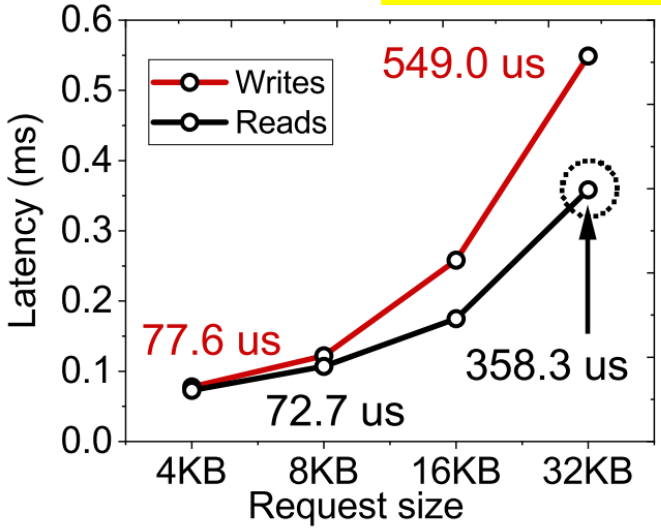
Bandwidth (seq)



Bandwidth (rand)

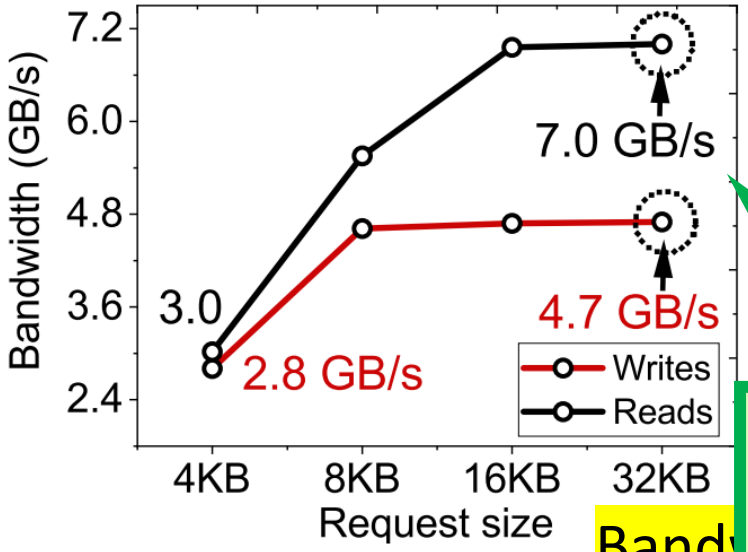


Latency (seq)

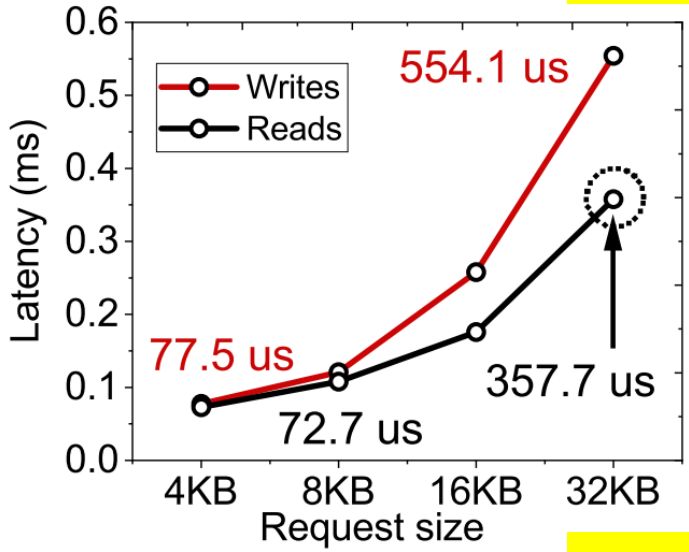


Latency (rand)

Performance w/ Microbenchmarks



Band



Latency (seq)

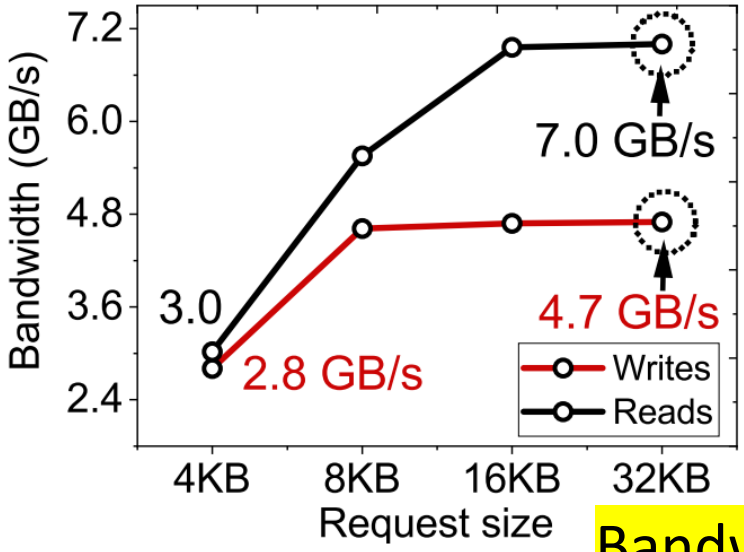
Why are writes slower than reads?

- 4KB-sized requests
  - There is no much bandwidth/latency difference between reads and writes (3 GB/sec vs 2.8/sec)
  - OpenExpress latency is 72~77 us at the e-depth eight
    (P4800X offers 120~150 us)

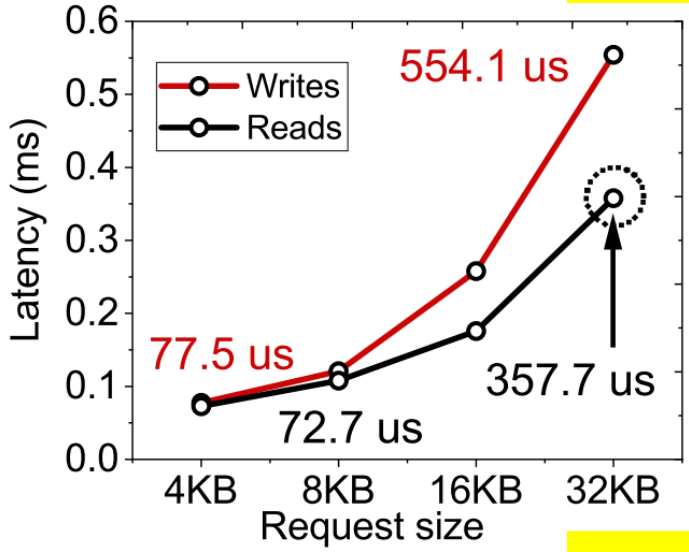
hes the maximum bandwidth
6KB-sized requests

  - 4.7GB/sec for random writes (258 us)
  - 7GB/sec for random reads (175 us)
    (vs. P4800X offers 532 us~600us)

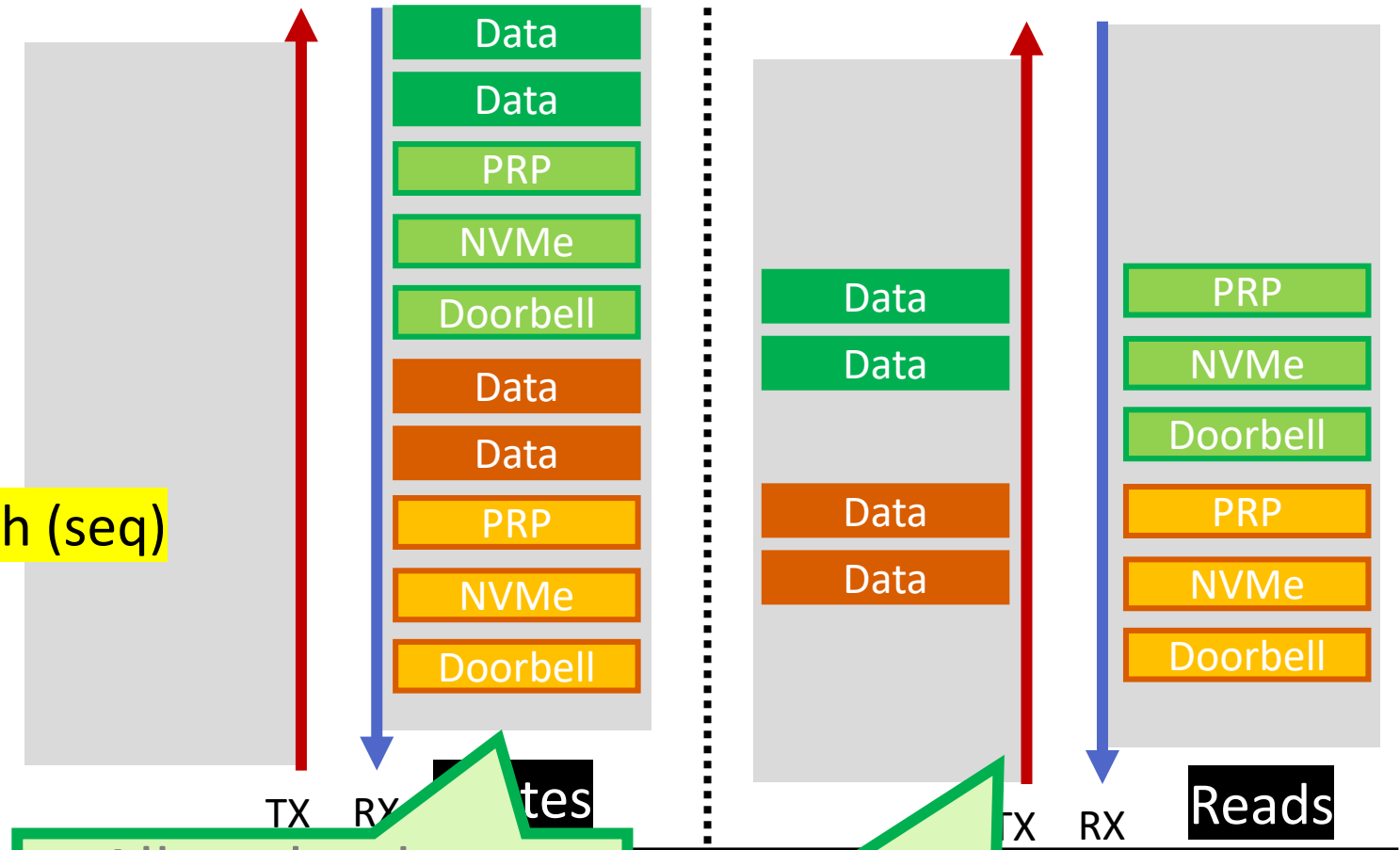
Performance w/ Microbenchmarks



Bandwidth (seq)



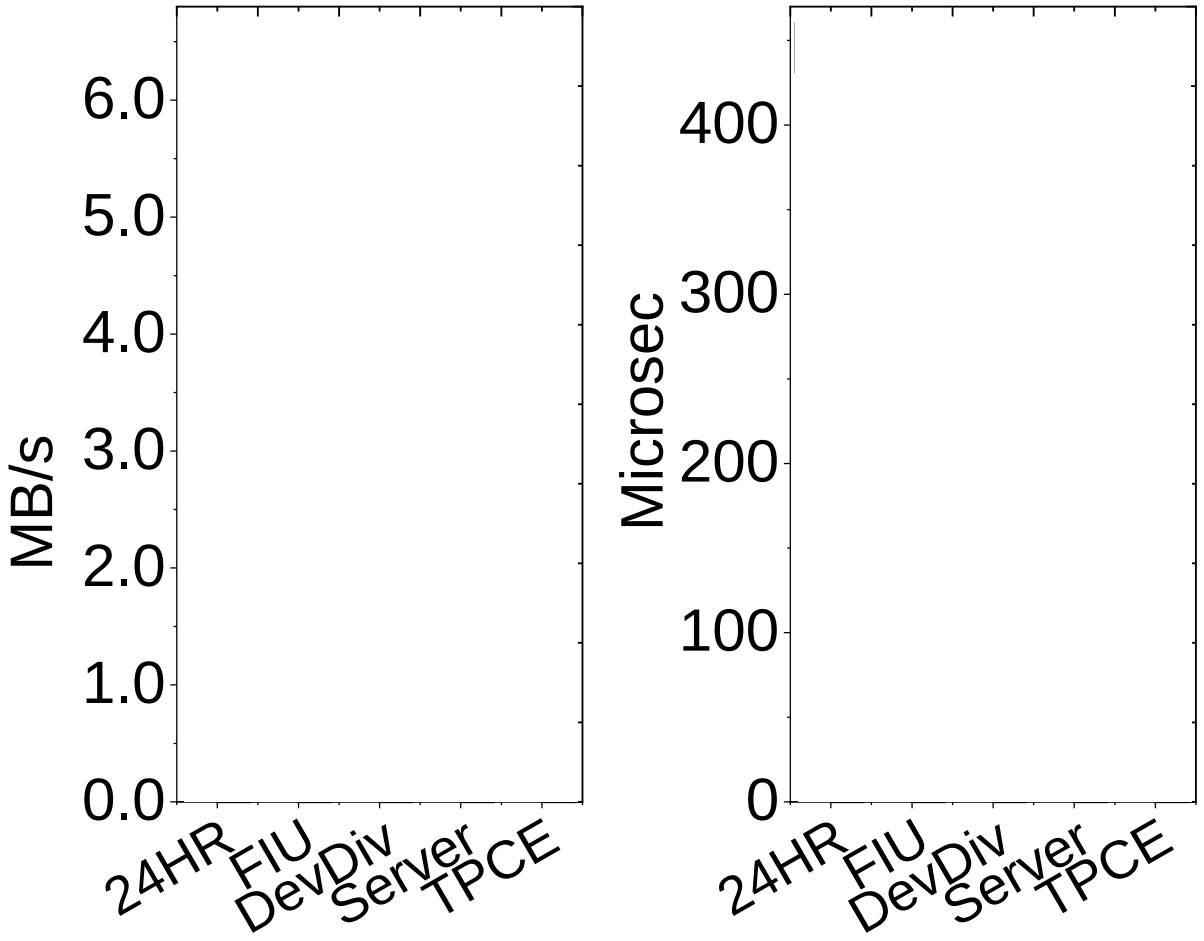
Latency (seq)



All payloads are serialized

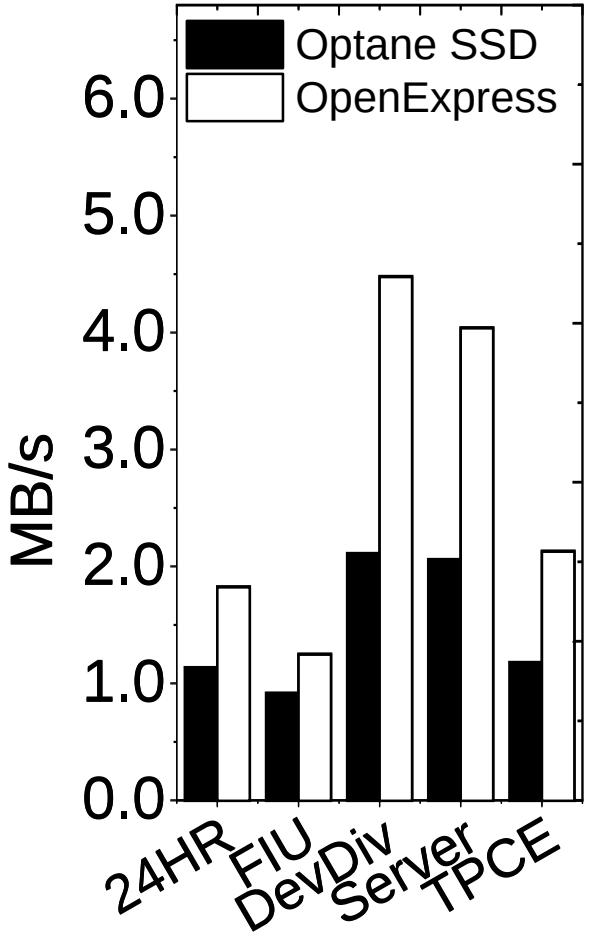
Data and NVMe payloads are served in parallel

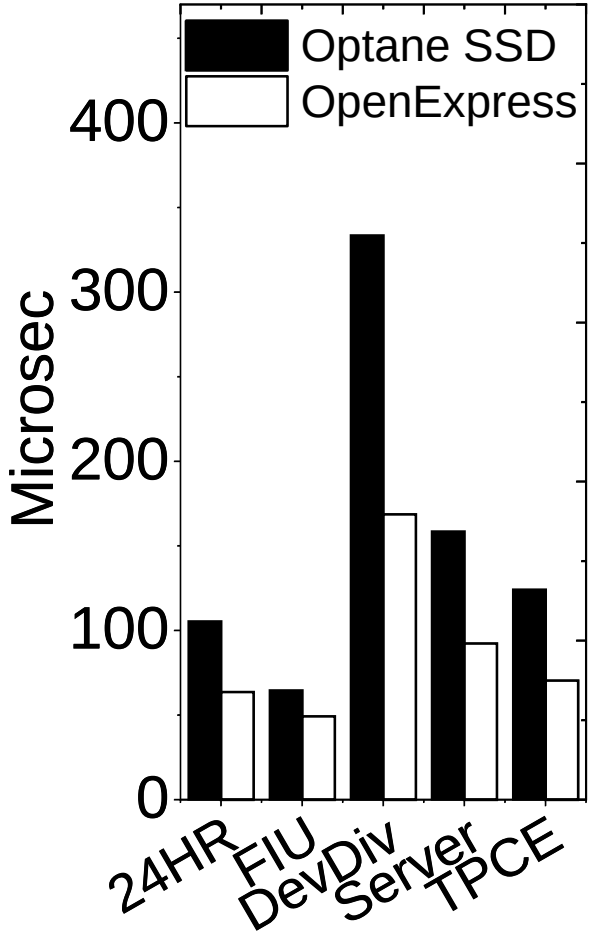
Real Workload Evaluation



- The performance of real workloads are different with microbenchmark results because of unaligned request offsets, sector length variations, etc.
  - Most storage cards cannot reach their best performance with real workload executions
- Read-intensive workloads (**DevDiv** and **Server**)
  - OpenExpress offers 4GB/sec ~ 4.5GB/sec (100~200 us)
- Write-intensive workloads (**24HR, FIU** and **TPCE**)
  - 1.25GB/sec~2.1GB/sec (all under 100 us)

Real Workload Evaluation





- **Bandwidth**
  - OpenExpress shows 76.3% better bandwidth than Optane SSD, on average
- **Latency**
  - It exhibits 68.6% shorter latency than those of OptaneSSD
- **DevDiv**
  - 111.5% better performance compared to Optane SSD (2.1GB/sec)

FPGA is yet much slower, but the FPGA design and implementation for NVMe IP cores are not on the critical path and can be used for system-level studies as a research vehicle

Conclusion: Related Work and Download

Per-month unitary prices for 3rd party NVMe IP Core

The price of a **single-use** source code is around **$ 100K**

- For **academic/non commercial purpose**, OpenExpress will be freely downloadable:
  - Hardware automation IP cores (HTRW, FET, CMT, CTX..)
  - Firmware for MicroBlaze (to control administrator command management, device initialization, etc.)
- Download information

  https://openexpress.camelab.org

OpenExpress

usenix ASSOCIATION
ATC 2020

Fully Hardware Automated Open Research Framework for Future Fast NVMe Device

Myoungsoo Jung

Computer Architecture and Memory systems Laboratory

KAIST EE

CAMELab

CAMEL

Sponsored by

MEMRAY