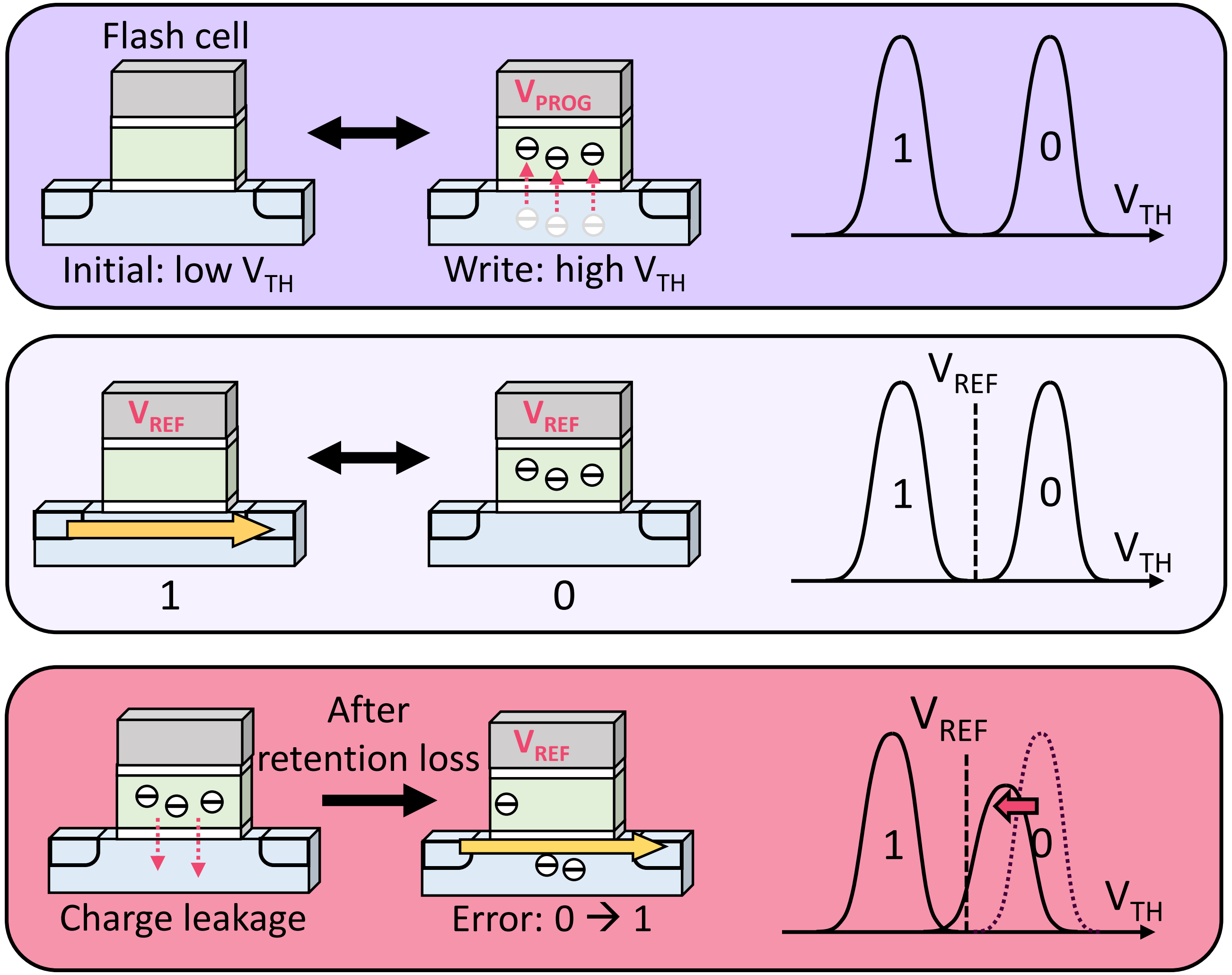
RiF: Improving Read Performance of Modern SSDs Using an On-Die Early-Retry Engine

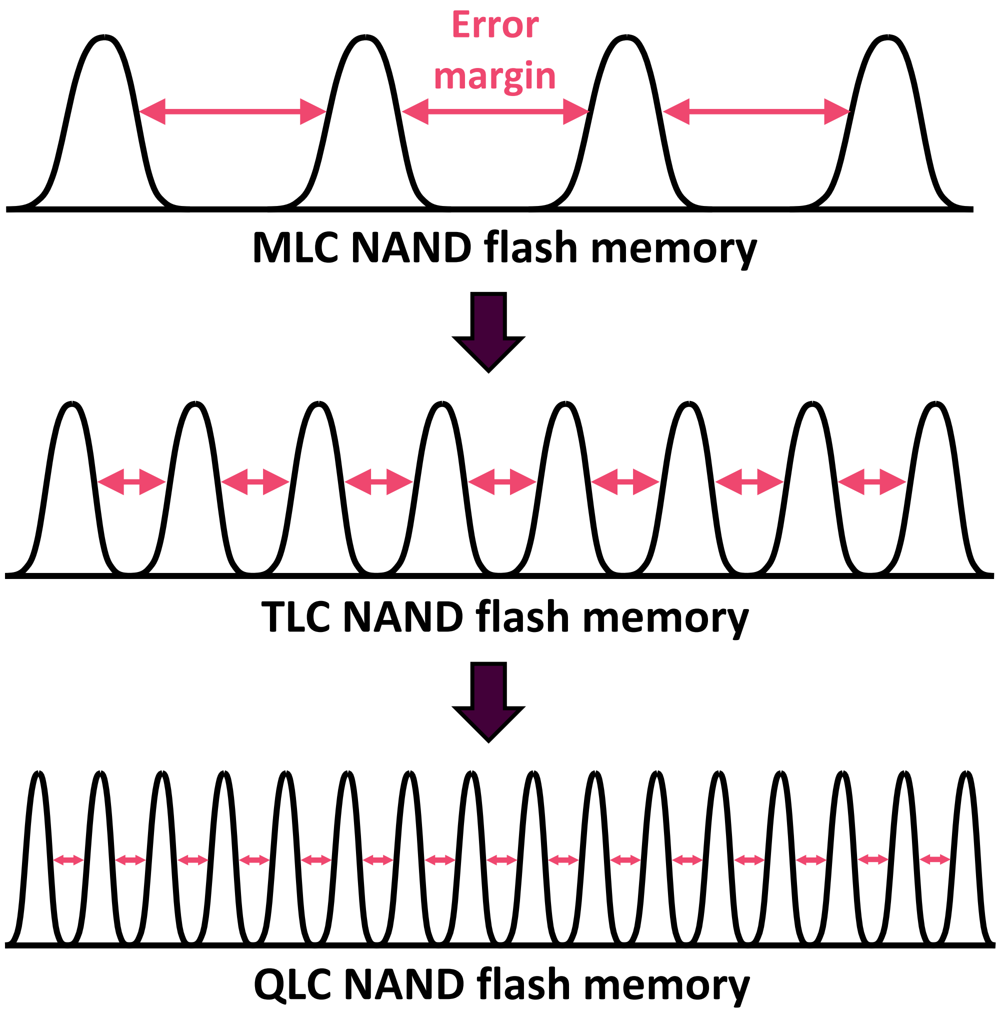
Myoungjun Chun[1], **Jaeyong Lee**[1], Myungsuk Kim[2],
Jisung Park[3], and Jihong Kim[1]

[1]Seoul National University
[2]Kyungpook National University
[3]POSTECH

Talk Outline

Errors in NAND Flash Memory

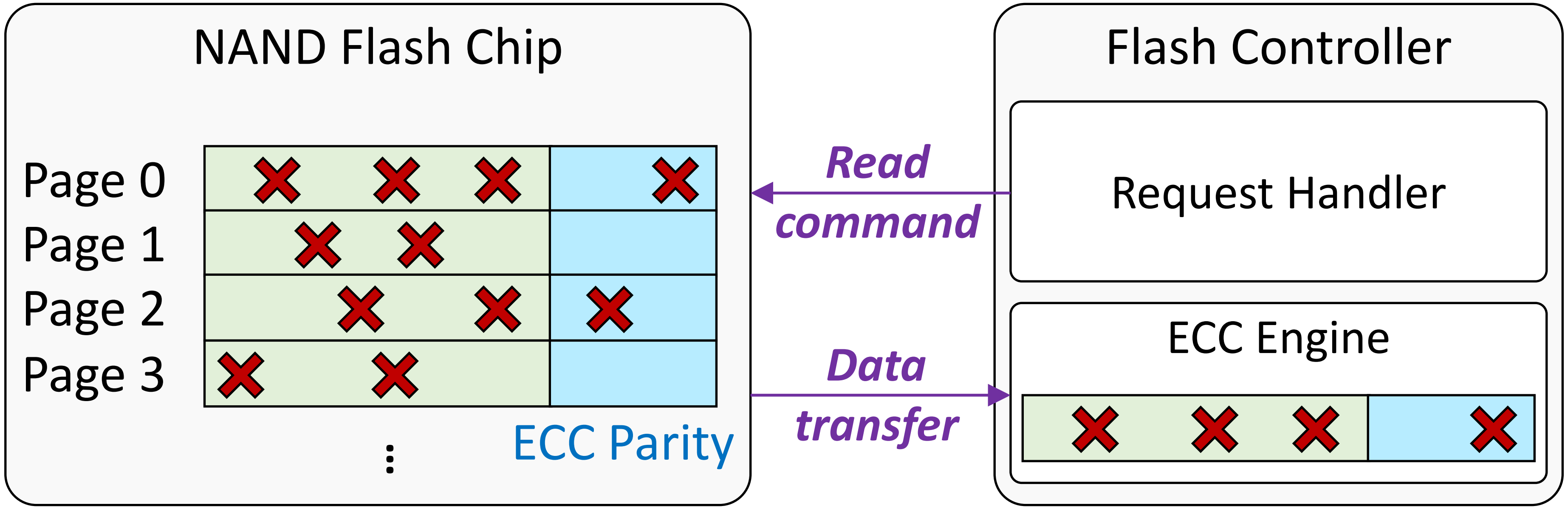




High-density NAND flash memory is ***highly error-prone***

Error-Correcting Codes (ECC)

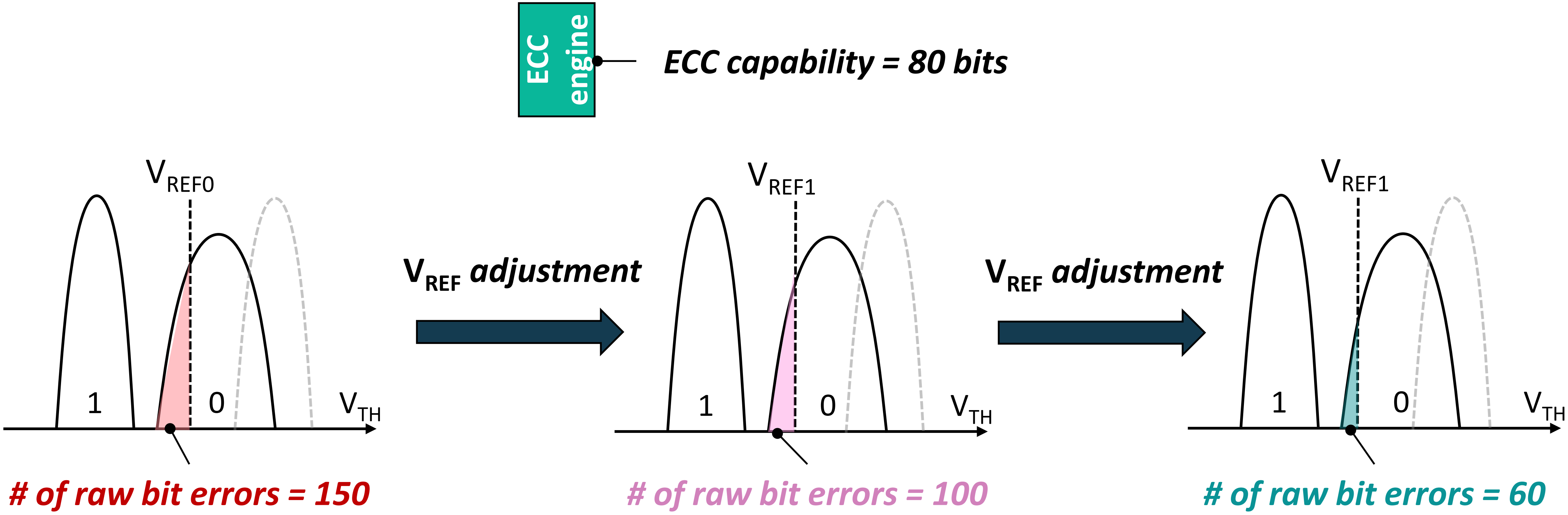
- Store **redundant information (ECC parity)** for error correction



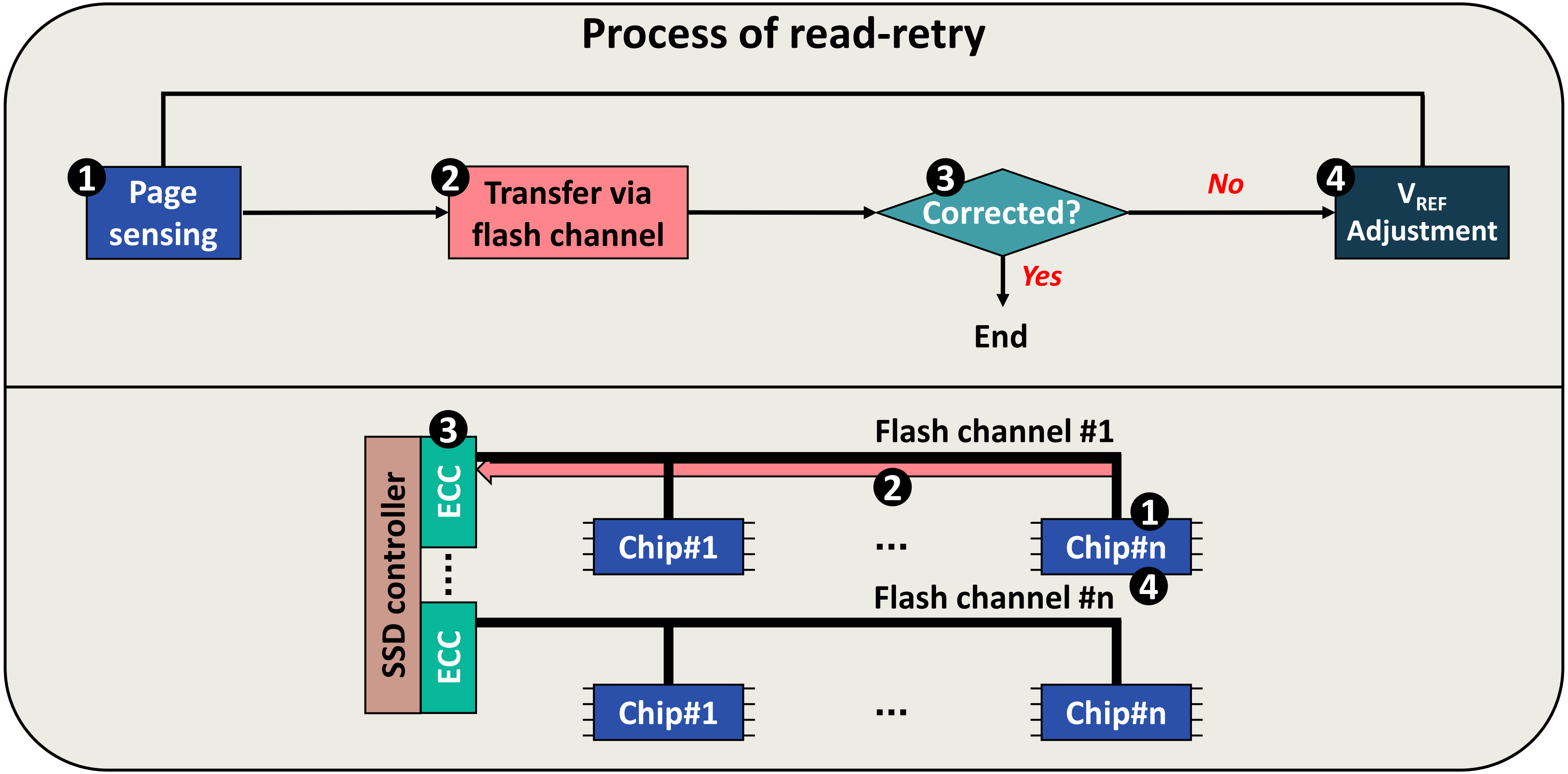
When # of raw bit errors (RBER) exceeds ECC correction capability, **uncorrectable errors** occur

Read-Retry in NAND Flash Memory

- Read-retry **re-reads** the target page with adjusted V_{REF} values

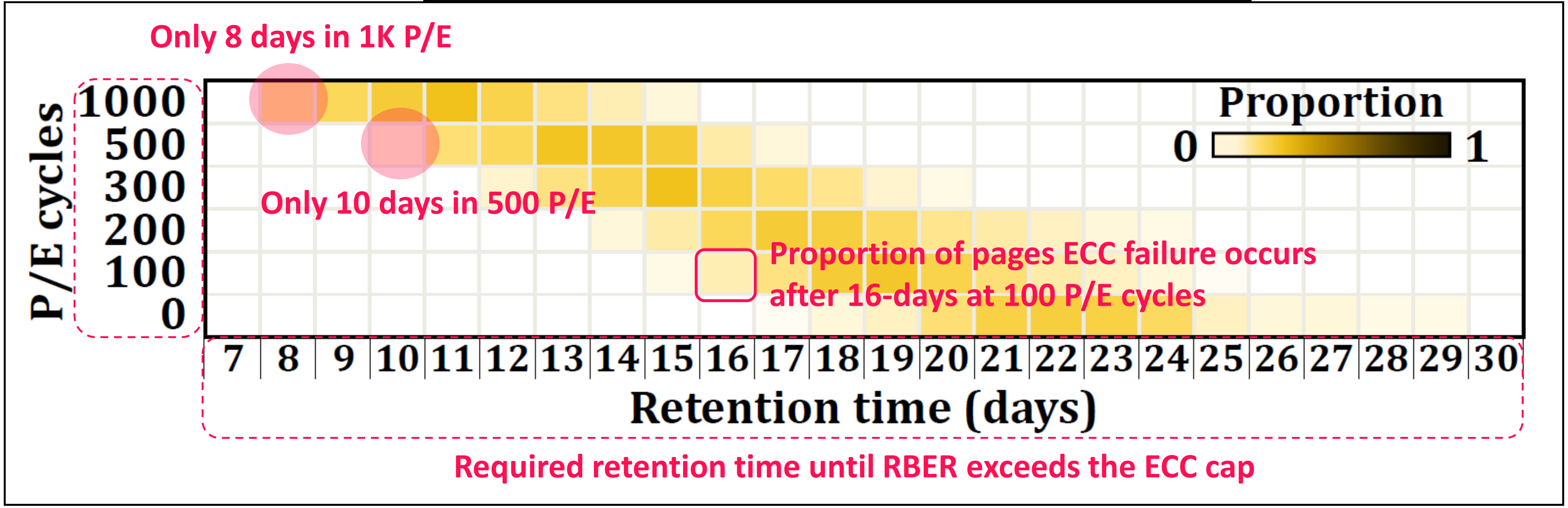


Read-Retry in NAND Flash Memory (cont'd)



Frequency of Read-Retry in Modern NAND Flash

Characterization results using 160 real chips

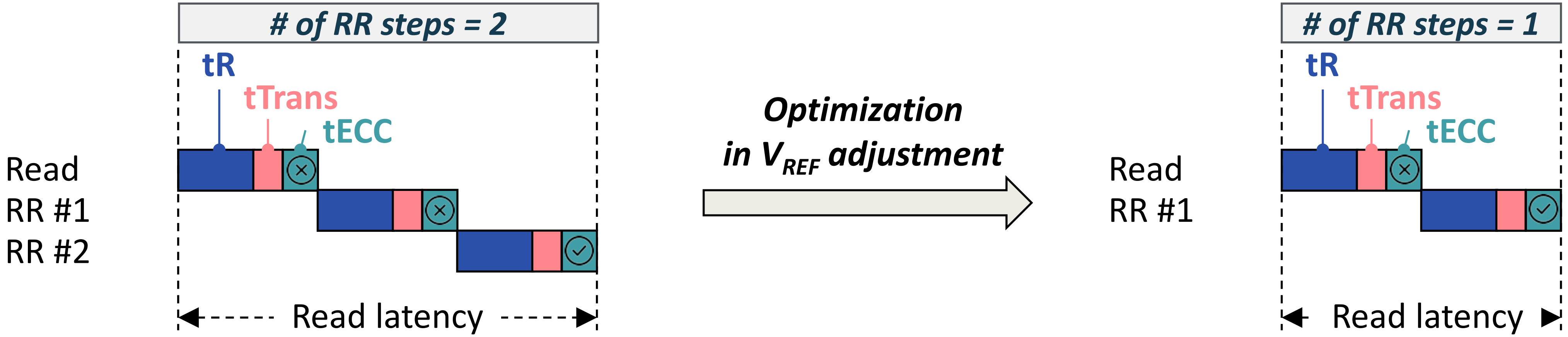


Read-retry (i.e., ECC failure) ***occurs very frequently***
in modern NAND flash

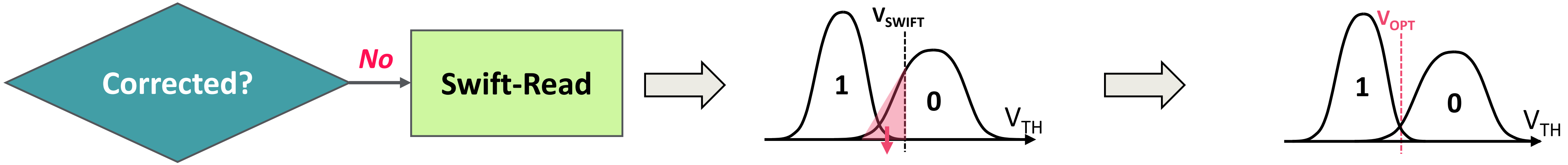
Talk Outline

Read-Retry Optimization

- To mitigate the performance overhead of frequent read-retry, many prior works focused on ***reducing # of read-retry steps***



- State-of-the-art: Swift-Read (ISSCC22) ***reduces # of read-retry steps to almost 1***

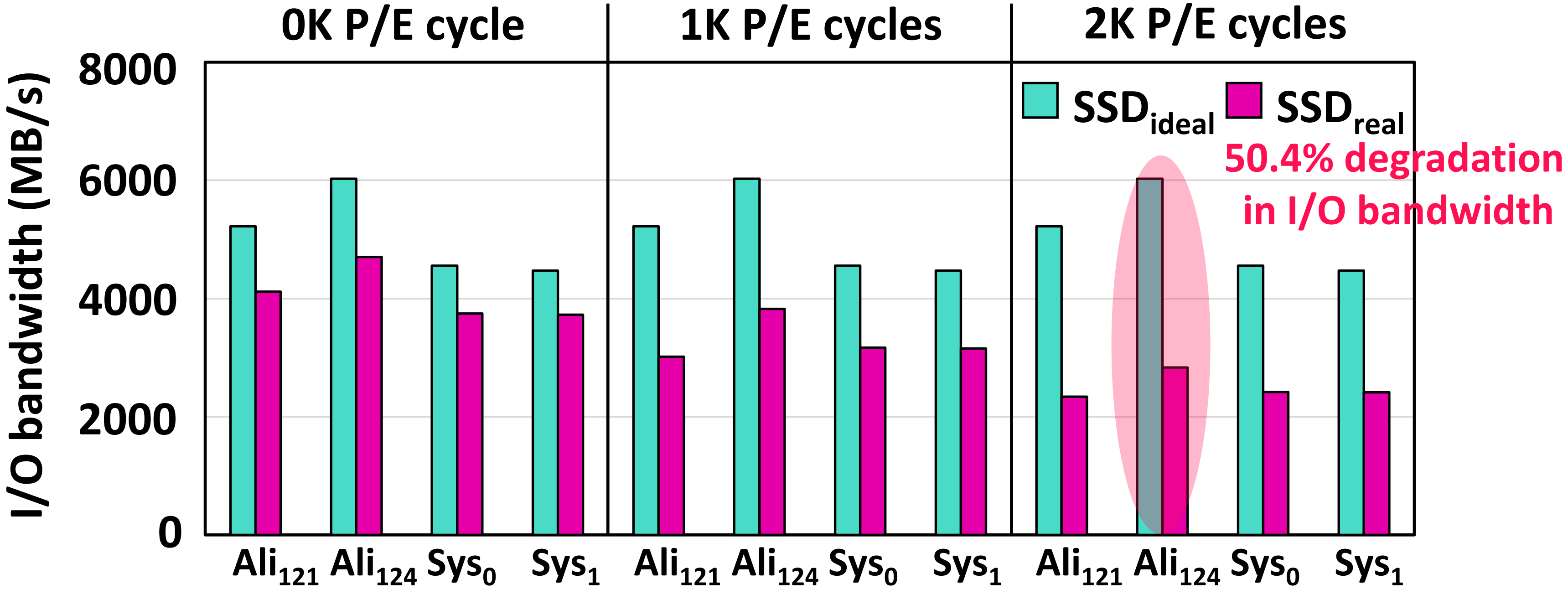


1) Count # of left-shifted cells

2) Estimate V_{OPT} and re-reads the page

Impact of Read-Retry on SSD's Performance

- **Compare I/O bandwidth of two SSDs using MQSim-E**
  - ***SSD$_{real}$***: An SSD w/ ***optimal*** read-retry solution ***(# of RR steps = 1)***
  - ***SSD$_{ideal}$***: An ideal SSD w/ no read-retry

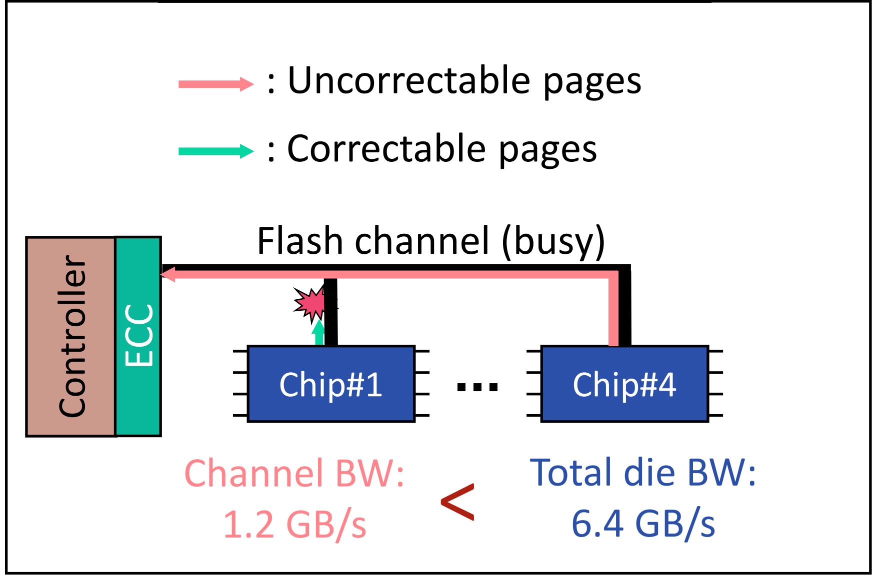


Read-Retry degrades SSD's performance significantly even ***with the optimal solution***

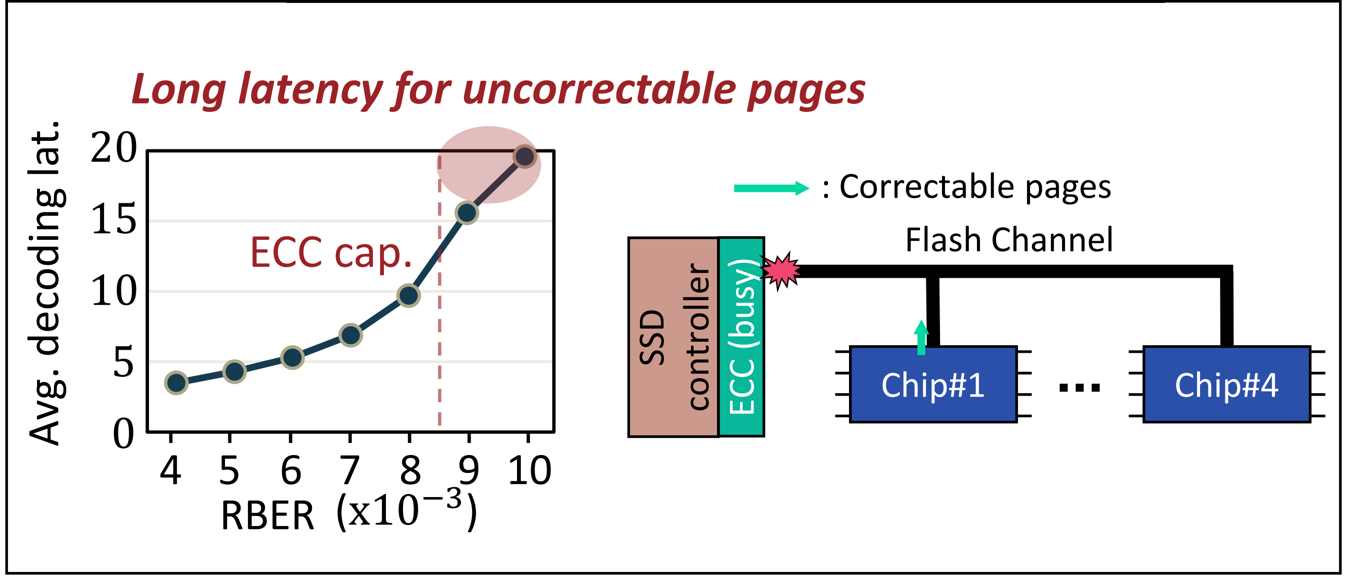
Root Cause Analysis

- ***Root Cause***: wasted effort for ***uncorrectable pages***

Waste 1: page transfer

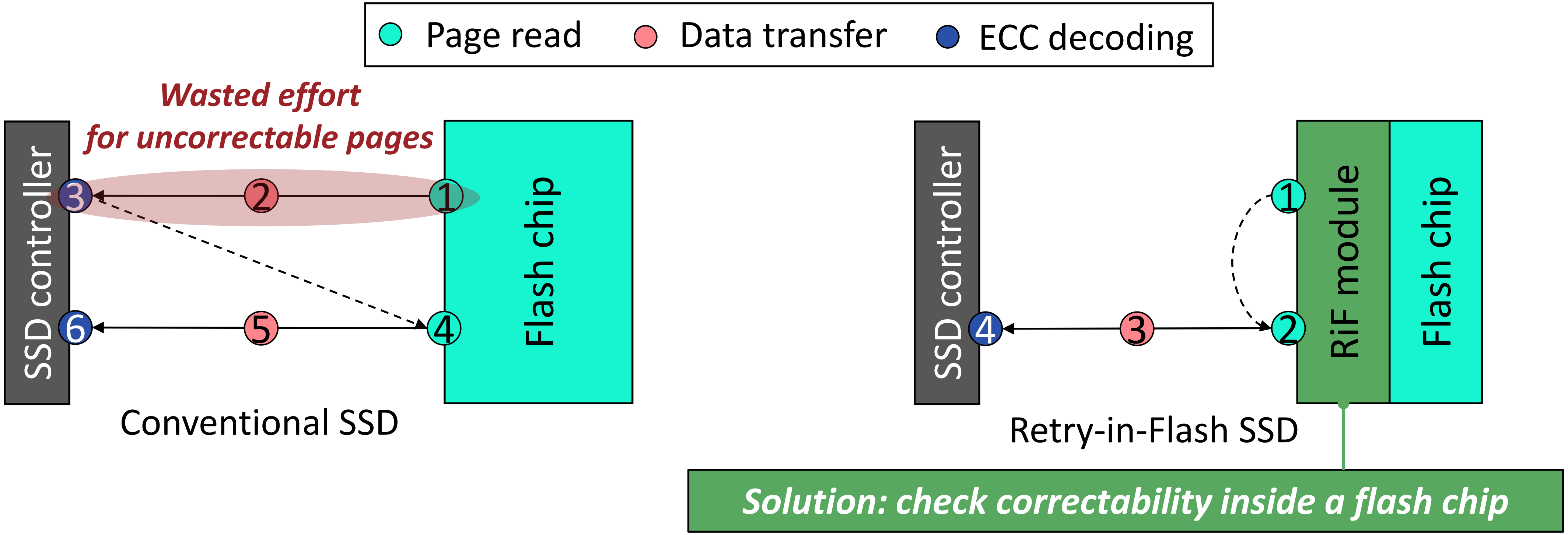


Waste 2: long ECC decoding time



Talk Outline

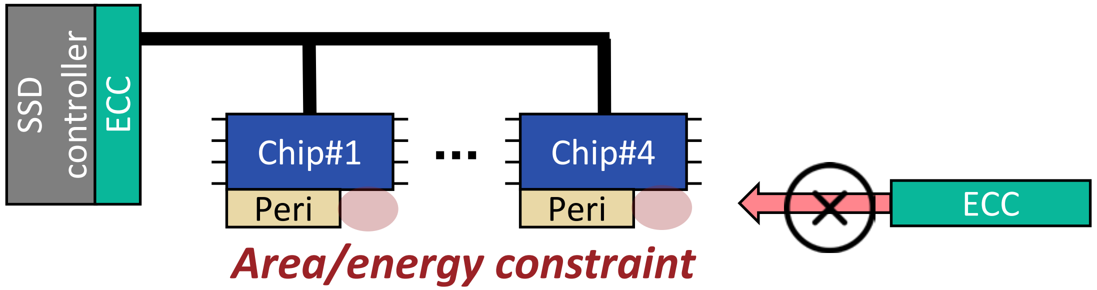
Key Idea: Retry-in-Flash (RiF)



Effect 1: No wasted page transfer (2)
Effect 2: No wasted decoding time (3)

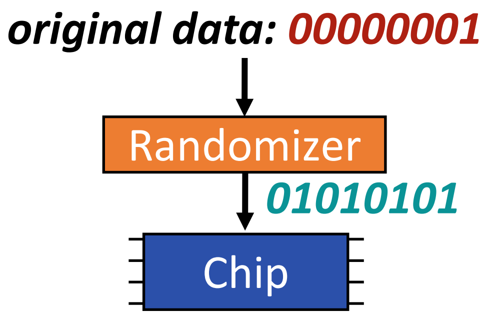
Implementation Challenge

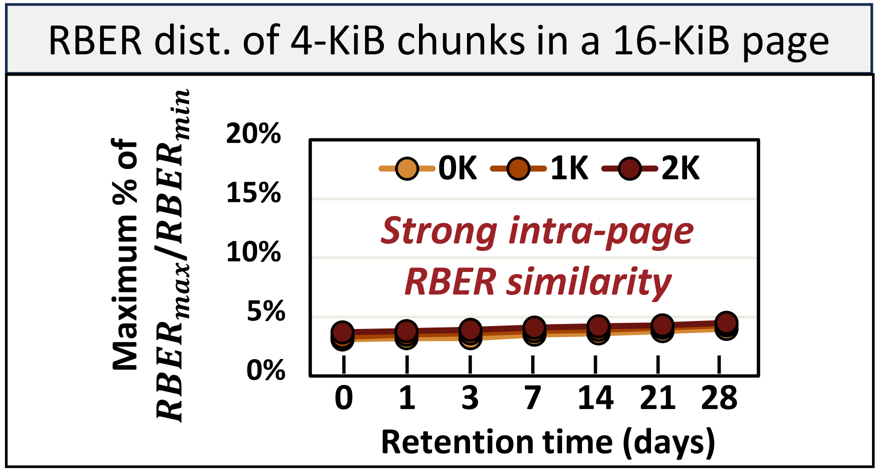
- Hard constraints on ***in-flash RiF module***

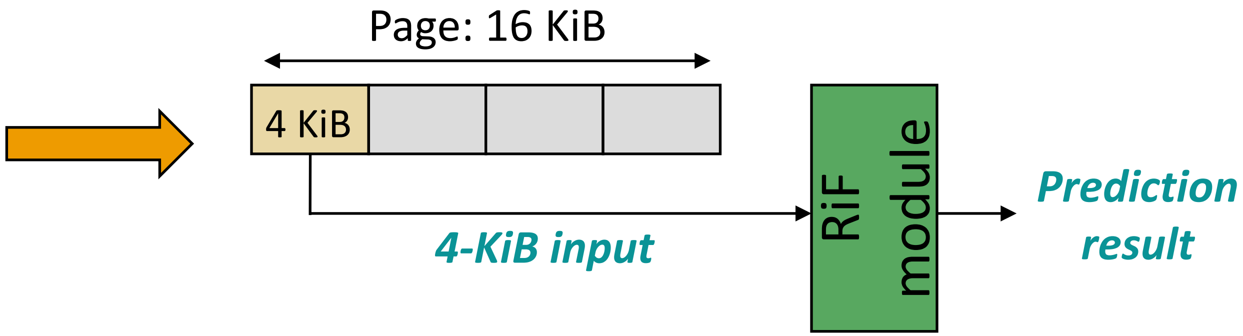


- ***Optimization 1:*** Subpage-based prediction

All data in pages must be ***randomized*** in modern NAND flash memory







Implementation Challenge (cont'd)

- ***Optimization 2:*** Prediction-centric module

Uncorrectable pages do not require ***entire ECC decoding***
Instead, we simply want to know ***if a page is correctable or not***

Syndrome weight-based prediction

H = a parity check matrix, ***c*** = a codeword = $(c_0, c_1 \ldots c_{N-1})$, ***S*** = a syndrome vector = $(s_0, s_1, s_2, \ldots s_{M-1})$

$$\boldsymbol{H} = \begin{array}{c} \\ M = 4 \end{array} \underbrace{\begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}}_{N = 8}$$

1's represent the bit position of a codeword

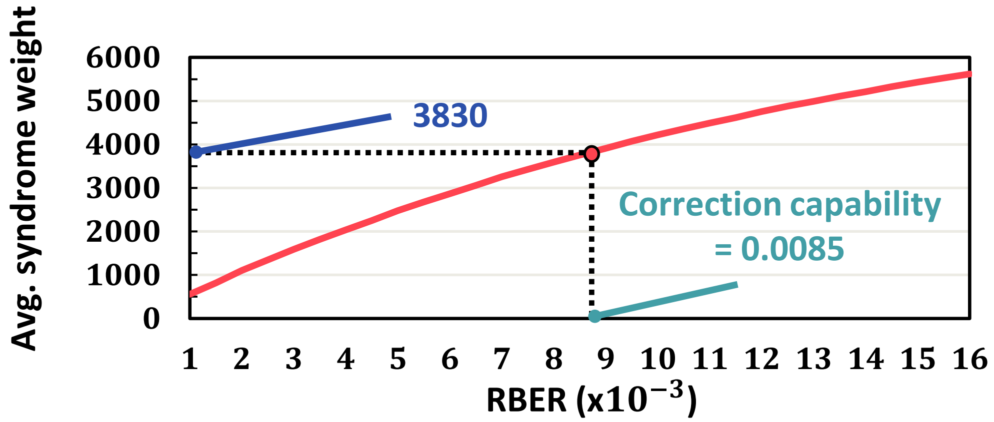
$$\boldsymbol{S}^T = \boldsymbol{H}\boldsymbol{c}^T = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} c_1 \oplus c_3 \oplus c_4 \oplus c_7 \\ c_0 \oplus c_1 \oplus c_2 \oplus c_5 \\ c_2 \oplus c_5 \oplus c_6 \oplus c_7 \\ c_0 \oplus c_3 \oplus c_4 \oplus c_6 \end{bmatrix}$$

S3 is 0 with no errors in ***c0, c3, c4, c6***

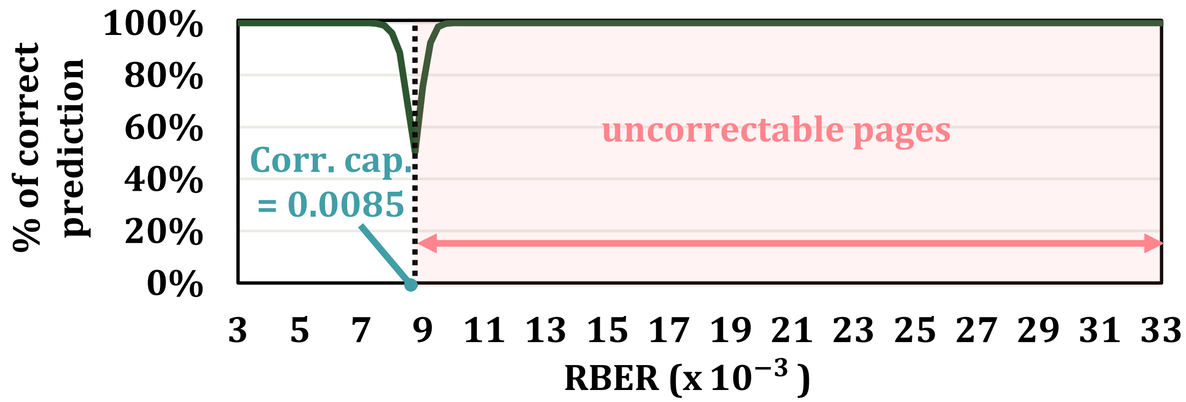
S3 is 1 with any bit error

Syndrome weight *SW*
(i.e., ***s0 + s1 + s2 + s4***) → ***SW* > thr?**
- *no* → *Dout*
- *yes* → ***In-flash read-retry***

Validation Results



✓ ***Strong correlation***



✓ ***High prediction accuracy***

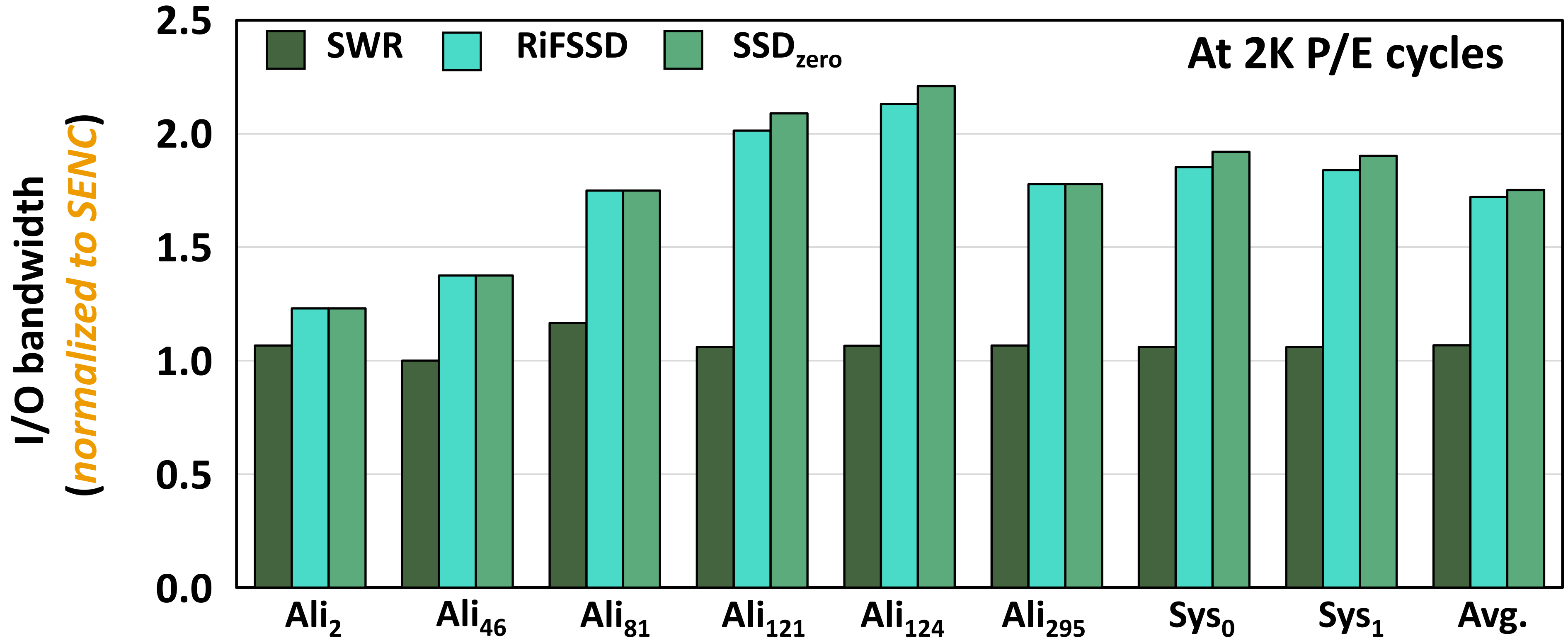
RiF module achieves ***98.7% prediction accuracy*** based on the ***syndrome weight-based prediction***

Talk Outline

Experiment Settings

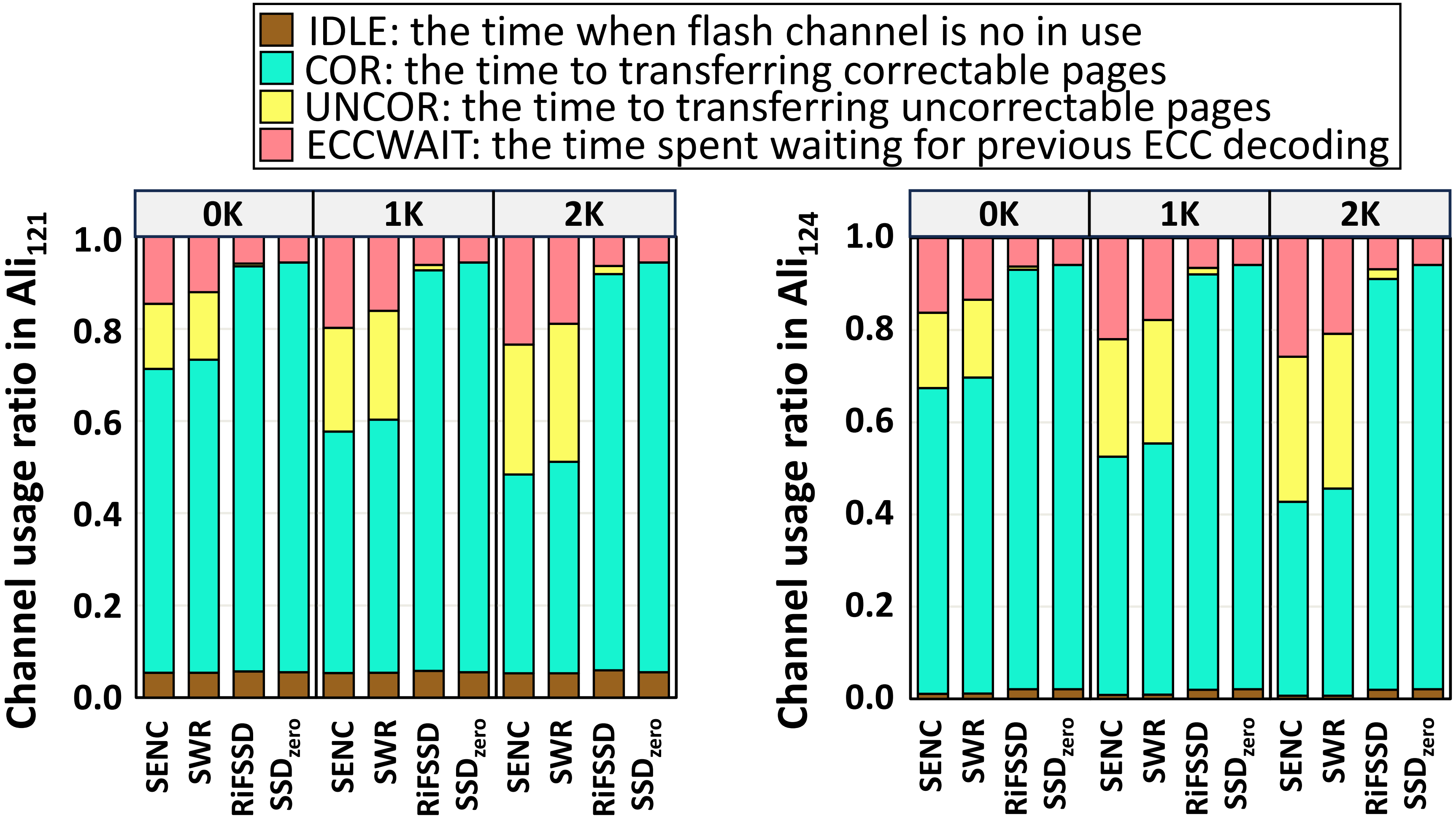
- ***Simulator:*** MQSim-E
  - Extend NAND flash models w/ real-device characterization results
- ***Workloads:*** 8 real-world traces
  - 6 from AliCloud traces
  - 2 from Systor traces
- ***Comparision***
  - $\mathbf{SSD_{zero}}$: An ideal SSD where no read-retry occurs
  - **SENC**: A state-of-the-art read-retry mitigation scheme (Sentinel, MICRO21)
  - **SWR**: A state-of-the-art read-retry mitigation scheme (Swift-Read, ISSCC22)

Experiment Results



RiF improves the I/O bandwidth ***71.2%***, ***61.2%*** over SENC, SWR

Experiment Results (cont'd)



RiF efficiently reduces ***wasted channel bandwidth*** from ***UNCOR, ECCWAIT***

Conclusion

- Observed that SOTA RR optimizations cannot prevent *the degradation of effective channel bandwidth*

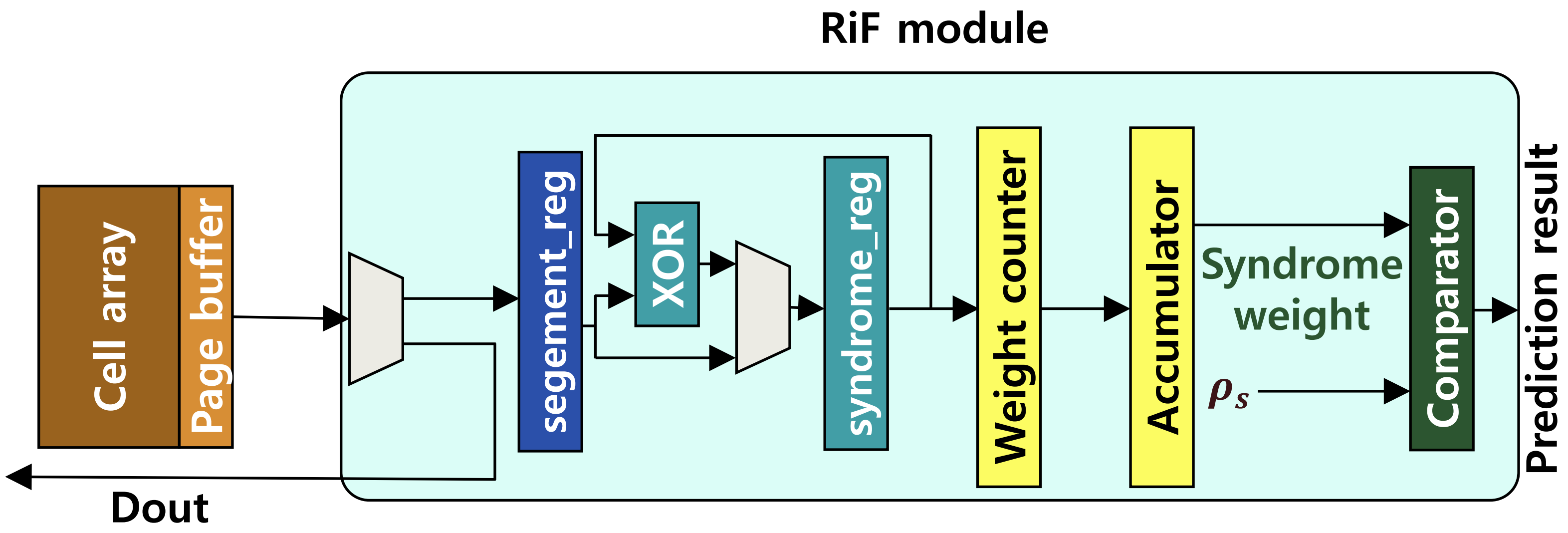
- Presented the RiF scheme, which *determines early on whether a read-retry is required at the flash-chip level*

- Presented a highly optimized RiF module that uses *syndrome weight-based RR prediction*

- Showed the RiF scheme *improved SSD bandwidth by 72.1%* on average at 2K P/E cycles

Thanks for Listening!
Any Questions? ☺

Overhead Analysis



✓ ***2.5 μs*** time overhead
✓ ***0.012 mm²*** area consumption
✓ ***1.28 mW*** power consumption

Need for Randomization