P&S Modern SSDs
Cutting-Edge Research in SSDs

Dr. Jisung Park
Prof. Onur Mutlu

ETH Zürich
Spring 2023
26 May 2023
Reducing Solid-State Drive Read Latency by Optimizing Read-Retry

Jisung Park¹ Myungsuk Kim²,³ Myoungjun Chun² Lois Orosa¹ Jihong Kim² Onur Mutlu¹
¹ETH Zürich Switzerland ²Seoul National University Republic of Korea ³Kyungpook National University Republic of Korea

The 26th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS’21)
Problem: Long, Non-Deterministic Latency

Read request (4 KiB)

Requested data
Expected latency: 100 µs
Actual latency: > 1 ms

NAND Flash-Based SSD

Degrades the quality of service of read-intensive, latency-sensitive applications
Problem: Long, Non-Deterministic Latency

Read request (4 KiB)

Requested data
Expected latency: 100 µs
Actual latency: > 1 ms

NAND Flash-Based SSD

Flash-Translation Layer (FTL): SSD Firmware

Read Chip#2

Read CMD

NAND Flash
Chip#1

NAND Flash
Chip#2

Internal Read-Retry Operations
Errors in NAND Flash Memory

- NAND flash memory stores data by using cells’ $V_{TH}$ levels.
Errors in NAND Flash Memory

- Various sources **shift and widen** programmed $V_{TH}$ states
  - Retention loss, program interference, read disturbance, etc.

![Diagram showing $V_{TH}$ Level and Reference Voltage $V_{REF}$](image)

- **Interference**
  - 1 Erased State
  - 0 Programmed State

- **Retention loss**

Errors:

$<V_{TH}$ Distribution of an SLC Page>
Error-Correcting Codes (ECC)

- Store **redundant information** (ECC parity)
  - To detect and correct row bit errors
Error-Correcting Codes (ECC)

- Store redundant information (ECC parity)
  - To detect and correct row bit errors
Errors in Modern NAND Flash Memory

- High row bit-error rates (RBER) in MLC NAND flash memory
  - Narrow margin b/w adjacent $V_{TH}$ states

### <$V_{TH}$ Distribution of an SLC Page>

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
<th>CSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Erased (E)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Programmed</td>
</tr>
</tbody>
</table>

### <$V_{TH}$ Distribution of a TLC Page>

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
<th>VTH Margin</th>
<th>VREF0</th>
<th>VREF1</th>
<th>VREF2</th>
<th>VREF3</th>
<th>VREF4</th>
<th>VREF5</th>
<th>VREF6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>VREF0</td>
<td>111</td>
<td>110</td>
<td>100</td>
<td>000</td>
<td>010</td>
<td>011</td>
<td>001</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>VREF1</td>
<td>VREF2</td>
<td>VREF3</td>
<td>VREF4</td>
<td>VREF5</td>
<td>VREF6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VREF: Reference Voltage
Errors in Modern NAND Flash Memory

• High row bit-error rates (RBER) in MLC NAND flash memory
  ◦ Narrow margin b/w adjacent $V_{TH}$ states

Strong ECC: Corrects ~80 bit errors per 1-KiB data
Not Scalable: Area, power, latency, ...

What if RBER > ECC Capability?
Read-Retry Operation

- Reads the page again with adjusted $V_{REF}$ values
Read-Retry Operation

- Reads the page again with adjusted $V_{REF}$ values

Read-retry: Adjusting $V_{REF}$ values

Read using properly-adjusted $V_{REF}$ values

→ Decreases # of raw bit errors to be lower than the ECC capability
Read-Retry: Performance Overhead

$N_{\text{ERR}} = 32 < \text{ECC capability } C_{\text{ECC}} = 72$

$t_{\text{DMA}}: \text{Data transfer}$
$t_{R}: \text{Page sensing}$
$t_{\text{ECC}}: \text{ECC decoding}$

READ A

$t_{\text{READ}}$
Read-Retry: Performance Overhead

- tR
- tECC
- tDMA

**READ A**

- **tREAD**
- **tRETRY**

- **N_{ERR} = 232 > ECC capability C_{ECC} = 72**
- **N_{ERR} = 173**
- **N_{ERR} = 118**
- **N_{ERR} = 87**
- **N_{ERR} = 23**

\[ N_{ERR} = 118 = N \times (tR + tDMA + tECC) \]

---

Read-retry increases the read latency almost linearly with the number of retry steps.
P&AR$^2$: Outline

- Read-Retry in Modern NAND Flash-Based SSDs
- PR$^2$: Pipelined Read-Retry
- AR$^2$: Adaptive Read-Retry
- Evaluation Results
Pipelined Read-Retry (PR²): Key Idea

In common cases, multiple (up to 25) retry steps occur.
Pipelined Read-Retry (PR²): Key Idea

\[ N_{\text{ERR}} = 232 > \text{ECC capability } C_{\text{ECC}} = 72 \]

\[ N_{\text{ERR}} = 173 \]

\[ N_{\text{ERR}} = 118 \]

\[ N_{\text{ERR}} = 87 \]

\[ N_{\text{ERR}} = 23 \]

In common cases, multiple (up to 25) retry steps occur → Speculatively starts the next retry step
Pipelined Read-Retry (PR²): Key Idea

Removes $t_{DMA}$ & $t_{ECC}$ (~30% of each retry step) from the critical path
P&AR\textsuperscript{2}: Outline

- Read-Retry in Modern NAND Flash-Based SSDs
- PR\textsuperscript{2}: Pipelined Read-Retry
- AR\textsuperscript{2}: Adaptive Read-Retry
- Evaluation Results
Adaptive Read-Retry (AR$^2$): Key Idea

- $N_{ERR} = 232 >$ ECC capability $C_{ECC} = 72$
- $N_{ERR} = 173$
- $N_{ERR} = 118$
- $N_{ERR} = 87$
- $N_{ERR} = 23$

$t_{READ} - t_{RETRY} = N \times t_{R} + t_{DMA} + t_{ECC}$
Adaptive Read-Retry (AR²): Key Idea

A large ECC margin in the final retry step when read-retry succeeds

→ Can we leverage this ECC (reliability) margin?
Adaptive Read-Retry (AR²): Key Idea

- Trading the large ECC margin to reduce $t_R$

$$t_R' = \alpha \times t_R$$

$$N_{ERR} = 232 > \text{ECC capability } C_{ECC} = 72$$

$$N_{ERR} = 173 + e_1$$

$$N_{ERR} = 118 + e_2$$

$$N_{ERR} = 87 + e_{N-1}$$

$$N_{ERR} = 23 + e_N$$

$$t_{READ} \cdots t_{RETRY} = N \times t_R + t_{DMA} + t_{ECC}$$
Adaptive Read-Retry (AR²): Key Idea

Trading the large ECC margin to reduce \( t_R \)

\( \rightarrow \) Further reduction in the read-retry latency
Adaptive Read-Retry (AR²): Key Idea

**Latency Reduction**

\[ N_{ERR} = 232 > \text{ECC capability } C_{ECC} = 72 \]

\[ N_{ERR} = 173 + e_1 \]

\[ N_{ERR} = 118 + e_2 \]

\[ N_{ERR} = 87 + e_{N-1} \]

\[ N_{ERR} = 23 + e_N \]

**AR² reduces tR in every retry step**
Adaptive Read-Retry (AR²): Key Idea

AR² reduces $t_R$ in every retry step ensuring $N_{ERR} < C_{ECC}$ in the final retry step.
Real-Device Characterization

• 160 real 48-layer TLC NAND flash chips

• Observation 1: A large ECC margin in the final retry step even under worst-case operating conditions
  ◦ At most 40 errors per KiB under 1-year retention time @ 2K program and erase (P/E) cycles
  ◦ Use of near-optimal $V_{REF}$ in the final retry step

• Observation 2: A large reliability margin incorporated in read-timing parameters
  ◦ 25% $t_R$ reduction $\rightarrow$ At most 23 additional errors
  ◦ Worst-case-based design due to process variations

AR$^2$ can easily work in commodity NAND flash chips w/ at least 25% $t_R$ reduction
P&AR\textsuperscript{2}: Design

### SSD Firmware (FTL)

<table>
<thead>
<tr>
<th>P/E</th>
<th>$t_{RET}$ [days]</th>
<th>$t_R$ [(\mu)s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 250</td>
<td>&lt; 60</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>&lt; 360</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>&lt; 360</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>&lt; 360</td>
<td>75</td>
</tr>
<tr>
<td>&lt; 1.5K</td>
<td>&lt; 60</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>&lt; 360</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>&lt; 360</td>
<td>75</td>
</tr>
</tbody>
</table>

#### Read-timing Parameter Table (Pre-profiled)

**Flash Controller**

**ECC Engine**

**NAND Flash Chip**

**READ** \(A\)  \(\rightarrow\)  **Fail**

**READ** \(A\)  \(\rightarrow\)  **Data** \(A\)

**ECC fail**
### P&AR²: Design

#### SSD Firmware (FTL)

<table>
<thead>
<tr>
<th>P/E</th>
<th>$t_{\text{RET}}$ [days]</th>
<th>$t_R$ [μs]</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 250</td>
<td>&lt; 60</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; 360</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; 360</td>
<td>75</td>
</tr>
<tr>
<td>&lt; 1.5K</td>
<td>&lt; 60</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; 360</td>
<td>75</td>
</tr>
</tbody>
</table>

---

**Best $t_R$ for the current operating conditions**

---

**Best $t_R$**

**Fail**

---

**Flash Controller**

**ECC Engine**

---

**NAND Flash Chip**

---

**Read-timing Parameter Table (Pre-profiled)**

---

**ECC fail**

---

<table>
<thead>
<tr>
<th></th>
<th>tR</th>
<th>tDMA</th>
<th>tECC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tR</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
P&AR²: Design

SSD Firmware (FTL)

<table>
<thead>
<tr>
<th>P/E</th>
<th>( t_{RET} ) [days]</th>
<th>( t_R ) [μs]</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 250</td>
<td>&lt; 60</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>⋮</td>
<td>⋮</td>
</tr>
<tr>
<td></td>
<td>&lt; 360</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>⋮</td>
<td>⋮</td>
</tr>
<tr>
<td>&lt; 1.5K</td>
<td>&lt; 60</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>⋮</td>
<td>⋮</td>
</tr>
<tr>
<td></td>
<td>&lt; 360</td>
<td>75</td>
</tr>
</tbody>
</table>

Read-timing Parameter Table (Pre-profiled)

1. **Best \( t_R \)** for the current operating conditions

2. Read-retry w/ best \( t_R \) + speculative start

Flash Controller

ECC Engine

NAND Flash Chip

- \( t_R \)
- \( t_{DMA} \)
- \( t_{ECC} \)
- **Best \( t_R \)**
Key Takeaway

**Strong ECC:** to avoid read-retry as much as possible

→ Can provide high reliability margin when read-retry occurs

→ Can be used to reduce the read-retry latency
P&AR^2: Outline

- Read-Retry in Modern NAND Flash-Based SSDs
- PR^2: Pipelined Read-Retry
- AR^2: Adaptive Read-Retry
- Evaluation Results
Evaluation Results

• Simulation using MQSim [Tavakkol+, FAST18] and 12 real-world workloads

• Our proposal improves SSD response time by
  • Up to 51% (35% on average) compared to a high-end SSD w/o read-retry mitigation
  • Up to 32% (17% on average) compared to a state-of-the-art read-retry mitigation technique [Shim+, MICRO19]
P&S Modern SSDs
Cutting-Edge Research in SSDs

Dr. Jisung Park
Prof. Onur Mutlu
ETH Zürich
Spring 2023
26 May 2023