Reducing Solid-State Drive Read Latency by Optimizing Read-Retry

Jisung Park¹, Myungsuk Kim², Myoungjun Chun², Lois Orosa¹, Jihong Kim², and Onur Mutlu¹

¹SAFARI
²ETH Zürich

ASPLOS 2021 (Session 17: Solid State Drives)
Executive Summary

- **Problem:** Long read latency in modern SSDs due to read-retry
  - Frequently requires multiple retry steps to read an erroneous page

- **Goal:** Reduce the latency of each read-retry operation

- **Key Ideas:**
  - **Pipelined Read-Retry (PR²):** Concurrently perform consecutive retry steps using the CACHE READ command
  - **Adaptive Read-Retry (AR²):** Reduce read-timing parameters for every retry step by exploiting the reliability margin provided by strong ECC
  - Small implementation overhead and no changes to NAND flash chips

- **Evaluation Results:** Our proposal improves SSD response time by
  - Up to 51% (35% on average) compared to a high-end SSD
  - Up to 32% (17% on average) compared to a state-of-the-art baseline
Talk Outline

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

- NAND flash memory stores data by using cells’ $V_{TH}$ values

![Diagram showing read-reference voltage and cell's threshold voltage](image-url)
NAND Flash Basics

- NAND flash memory stores data by using cells’ $V_{TH}$ values.

![Diagram showing NAND Flash Basics]

- $V_{TH} < V_{REF}$
- $V_{TH} > V_{REF}$
NAND flash memory stores data by using cells’ $V_{TH}$ values.

- $P(x-1)$ state
- $P_x$ state
- $P(x+1)$ state
NAND Flash Basics

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

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

![Diagram showing the distribution of threshold voltages and states in NAND flash memory.](image)

Modern NAND flash memory is highly error-prone due to narrow \( V_{TH} \) margins.
Error-Correcting Codes (ECC)

- Store redundant information (ECC parity) for error correction

![Diagram of NAND Flash Chip and Flash Controller]

Read Page 0

NAND command

Data transfer

Flash Controller
- Request Handler
- ECC Engine

NAND Flash Chip
- Page 0
- Page 1
- Page 2
- Page 3

ECC Parity

...
Error-Correcting Codes (ECC)

- Store redundant information (ECC parity) for error correction

# of raw bit errors > ECC correction capability → Uncorrectable errors in stored data
Read-Retry Operation

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

![Diagram showing cell states and threshold voltages](image)

Erroneous cells  Cell’s Threshold Voltage ($V_{\text{TH}}$)
Read-Retry Operation

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

**Read-retry: Adjusting $V_{\text{REF}}$ values**

![Graph showing the relationship between number of cells, cell's threshold voltage, and the states P(x-1), Px, P(x+1), and Erroneous cells.]

**Erroneous cells**

**Cell's Threshold Voltage (V_{\text{TH}})**

Read using **properly-adjusted $V_{\text{REF}}$ values**

→ Decreases # of raw bit errors
Read-Retry: Performance Overhead

$t_{DMA}$: Data transfer
$t_R$: Page sensing
$t_{ECC}$: ECC decoding

$N_{ERR} = 32 < ECC$ capability $C_{ECC} = 72$
Read-Retry: Performance Overhead

ECC Capability $C_{ECC} = 72$

Read-retry increases the read latency almost linearly with the number of retry steps

$N_{ERR} = 232$

$N_{ERR} = 173$

$N_{ERR} = 118$

$N_{ERR} = 87$

$N_{ERR} = 23$

$\text{tREAD} \rightarrow \text{tRETRY} = N \times (tR + tDMA + tECC)$
Characterization of 160 real 3D TLC NAND flash chips
- ECC correction capability: 72 bits per 1-KiB data

Read-Retry in Modern SSDs: Experimental Data

- Elapsed time since the page was programmed

![Diagram showing read-retry steps and retention age]
Read-Retry in Modern SSDs: Experimental Data

- Characterization of 160 real 3D TLC NAND flash chips
  - ECC correction capability: 72 bits per 1-KiB data

![Graph showing the relationship between P/E cycles, retention age, and retry steps]

High P/E cycles and long retention age → More retry steps per read
Characterization of **160 real** 3D TLC NAND flash chips

- ECC correction capability: **72 bits** per 1-KiB data

Many reads require multiple retry steps even under **modest operating conditions**
Existing Read-Retry Mitigation Schemes

- Try to reduce $N_{RR}$ by predicting near-optimal $V_{REF}$ values

Read retry: Adjusting $V_{REF}$

After retention loss

Right after programming

Gradually adjusting
Existing Read-Retry Mitigation Schemes

- Try to reduce $N_{RR}$ by predicting near-optimal $V_{REF}$ values

**Read retry: Adjusting $V_{REF}$**

- **After retention loss**
  - $V_{REF(x-1)}$
  - $V_{REFx}$

- **Right after programming**
  - $V_{OPT}$
  - $V_{REFx}$

$V_{TH}$ changes are **fast and large** in modern SSDs → **Hard to eliminate** read-retry
Talk Outline

- Read-Retry in Modern NAND Flash-Based SSDs
- PR²: Pipelined Read-Retry
- AR²: Adaptive Read-Retry
- Evaluation Results
**PR²: Pipelined Read-Retry**

- **Key idea:** Concurrently perform consecutive retry steps

### ECC Capability

\[ C_{ECC} = 72 \]

### Read A

- **tREAD**
- **tRETRY**
- **tR**
- **tDMA**
- **tECC**

#### EXAMPLE

- **RR1**: 
  - \( N_{ERR} = 232 \)
  - **tR**
  - **tDMA**
  - **tECC**

- **RR2**: 
  - \( N_{ERR} = 173 \)
  - **tDMA**
  - **tECC**

- **RR(N − 1)**
  - \( N_{ERR} = 118 \)
  - **tDMA**
  - **tECC**

- **RRN**: 
  - \( N_{ERR} = 87 \)
  - **tDMA**
  - **tECC**

- **RR(N − 1)**
  - \( N_{ERR} = 23 \)
  - **tDMA**
  - **tECC**

\[ = N \times (tR + tDMA + tECC) \]
**PR²: Pipelined Read-Retry**

- **Key idea:** Concurrently perform consecutive retry steps

![Diagram of READ A with tREAD, tRETRY, tDAM, tECC, N_ERR values]

- **ECC Capability** $C_{ECC} = 72$

- Formulas:
  - $N_{ERR} = 232$
  - $N_{ERR} = 173$
  - $N_{ERR} = 118$
  - $N_{ERR} = 87$
  - $N_{ERR} = 23$

- Equation:
  
  \[ t_{READ} = N \times (t_R + t_{DMA} + t_{ECC}) \]
**PR²: Pipelined Read-Retry**

- **Key idea:** Concurrently perform consecutive retry steps

---

**ECC Capability** $C_{ECC} = 72$

---

**PR²: Removes** $t_{DMA}$ & $t_{ECC}$

($\sim$30% of each retry step) from the critical path
**PR²: Pipelined Read-Retry**

- **Key idea:** Concurrently perform consecutive retry steps

**ECC Capability \( C_{ECC} = 72 \)**

![Diagram](image)

- \( N_{ERR} = 232 \)
- \( N_{ERR} = 173 \)
- \( N_{ERR} = 118 \)
- \( N_{ERR} = 87 \)
- \( N_{ERR} = 23 \)

\[ t_{READ} = t_{RETRY}' = N \times t_R + t_{DMA} + t_{ECC} \]

**Latency reduction**
**PR²: Pipelined Read-Retry**

- **Key idea:** Concurrently perform consecutive retry steps

---

**READ A**

<table>
<thead>
<tr>
<th>RR1</th>
<th>RR2</th>
<th>:</th>
<th>RR(N - 1)</th>
<th>RRN</th>
<th>RR(N+1)</th>
</tr>
</thead>
</table>

- $tR$, $tDMA$, $tECC$
- $N_{ERR} = 232$
- $N_{ERR} = 173$
- $N_{ERR} = 118$
- $N_{ERR} = 87$
- $N_{ERR} = 23$

**ECC Capability** $C_{ECC} = 72$

**Unnecessarily started**

$tRETRY' = N \times tR + tDMA + tECC$

**Latency reduction**
PR²: Pipelined Read-Retry

- **Key idea**: Concurrently perform consecutive retry steps

ECC Capability $C_{\text{ECC}} = 72$

Unnecessarily started

$N_{\text{ERR}} = 232$

RESET

$t\text{READ}$

$t\text{RETRY}'$

$= N \times tR + t\text{DMA} + t\text{ECC}$

Latency reduction
**PR²: Pipelined Read-Retry**

- **Key idea:** Concurrently perform consecutive retry steps

---

**ECC Capability** $C_{ECC} = 72$

---

**PR²: Large latency reduction (~30%)**

w/ negligible performance penalty
Talk Outline

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

ECC Capability $C_{ECC} = 72$

\[
N_{ERR} = 232
\]

\[
N_{ERR} = 173
\]

\[
N_{ERR} = 118
\]

\[
N_{ERR} = 87
\]

\[
N_{ERR} = 23
\]

\[
tREAD = tRETRY' = N \times tR + tDMA + tECC
\]
**AR²: Adaptive Read-Retry**

ECC Capability $C_{ECC} = 72$

Observation: A positive ECC margin in the **final retry step** when read-retry succeeds.
AR²: Adaptive Read-Retry

- **Key idea:** Reduce read-timing parameters for every retry step

ECC Capability $C_{ECC} = 72$

- $t_{READ} = \sum_{i=1}^{N} t_{R} + t_{DMA} + t_{ECC}$
**AR\(^2\): Adaptive Read-Retry**

- **Key idea:** Reduce read-timing parameters for every retry step

---

- \( t_{R} \)
- \( t_{ECC} \)
- \( t_{DMA} \)

---

**ECC Capability** \( C_{ECC} = 72 \)

---

\[ t_{RETRY}' = N \times \rho \times t_{R} + t_{DMA} + t_{ECC} \]

---

**Latency reduction**
**AR²: Adaptive Read-Retry**

- **Key idea:** Reduce read-timing parameters for every retry step

---

**ECC Capability** \( C_{ECC} = 72 \)

- \( \text{READ A} \)
- \( \text{RR1} \)
- \( \text{RR2} \)
- \( \vdots \)
- \( \text{RR}(N-1) \)
- \( \text{RRN} \)

- \( N_{ERR} = 232 + \alpha \) (additional errors)
- \( N_{ERR} = 173 + \alpha \)
- \( N_{ERR} = 118 + \alpha \)
- \( N_{ERR} = 87 + \alpha \)
- \( N_{ERR} = 23 + \alpha \)

- Shorter \( \text{tR} \)

---

**Needs to ensure that**

\# of additional errors < ECC margin
AR²: Necessary Conditions

- **Condition 1: Large ECC margin in the final retry step**
  - **Strong ECC:** 72 bits correctable per 1-KiB data
  - **Use of near-optimal** $V_{REF}$ **in the final retry step**
    - # of raw bit errors **drastically increases** if $V_{REF} \gg V_{OPT}$
    - $\therefore$ In the final retry step, $V_{RRN} \sim V_{OPT}$

- **Condition 2: Sufficient reliability margin in read-timing parameters**
  - Manufacturers **pessimistically** set read-timing parameters
  - To cover for **worst-case process variation and operating conditions**

- We **experimentally analyze** if these conditions hold
AR$^2$: Real-Device Characterization

- Goals: Rigorously characterize
  - The ECC margin in the final retry step
  - Reliability impact of reducing read-timing parameters
  - Under different operating conditions

- Methodology
  - 160 real chips (48WL-layer 3D TLC NAND memory)
  - Randomly selected 11,059,200 pages
  - FPGA-based custom flash controller
    - Basic commands + test-mode commands (e.g., changing $V_{REF}$ values and read-timing parameters)
AR²: ECC Margin in the Final Retry Step

Large ECC margin in the final retry step even under worst-case operating conditions

Smaller ECC margin at higher P/E cycles and longer retention age
Large reliability margin in read-timing parameters → 25% $t_R$ reduction under worst-case conditions

Considerable variation depending on operating conditions
AR²: Effect of Reducing Timing Parameters

AR² Device-Characterization Takeaways

1. AR² can easily work in state-of-the-art NAND flash chips

2. Must properly reduce $t_{\text{R}}$ depending on the current operating conditions
Pipelined & Adaptive Read-Retry

Current conditions

<table>
<thead>
<tr>
<th>$PEC$</th>
<th>$t_{RET}$ [days]</th>
<th>$tR$ [$\mu s$]</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>&lt; 360</td>
<td></td>
<td>70</td>
</tr>
<tr>
<td>&lt; 1.5K</td>
<td>&lt; 60</td>
<td>70</td>
</tr>
<tr>
<td>&lt; 360</td>
<td></td>
<td>75</td>
</tr>
</tbody>
</table>

Read-timing Parameter Table

No change to chips and no impact on $V_{TH}$ states → Easy to combine with other techniques
Talk Outline

- Read-Retry in Modern NAND Flash-Based SSDs
- PR²: Pipelined Read-Retry
- AR²: Adaptive Read-Retry
- Evaluation Results
Evaluation Methodology

- **Simulator:** MQSim [Tavakkol, FAST18]
  - Extend NAND flash models w/ real-device characterization results

- **Workload:** 12 real-world I/O workloads
  - 6 from Microsoft Research Cambridge (MSRC) traces
    - 2 write-dominant: stg-0, hm-0
    - 4 read-dominant: prn-1, proj-1, mds-1, usr-1
  - 6 from Yahoo! Cloud Service Benchmark (YCSB)

- **Baselines**
  - **IDEAL:** An ideal SSD where no read-retry occurs
  - **BASE:** A high-end SSD w/o read-retry mitigation
  - **SOTA:** A state-of-the-art read-retry mitigation scheme [Shim, MICRO19]
    - Reduces the average number of retry steps by 70%
    - By predicting $V_{\text{REF}}$ values close to the optimal values
Results: PR²+AR² Performance

SSD response time (Normalized to BASE)

<table>
<thead>
<tr>
<th>MSR-W</th>
<th>MSR-R</th>
<th>YCSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-month retention</td>
<td>-9.5%</td>
<td>-28%</td>
</tr>
</tbody>
</table>

Large response time improvement: Up to 42% (26% on average)

Considerable improvement in write-dominant workloads due to garbage-collection reads
Results: SOTA & Optimal Performance

SSD response time (Normalized to BASE)

- MSR-W
- MSR-R
- YCSB

<table>
<thead>
<tr>
<th></th>
<th>PnAR2</th>
<th>SOTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSR-W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSR-R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>YCSB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3-month retention @ 0 P/E cycles
3-month retention @ 1K P/E cycles
6-month retention @ 2K P/E cycles
Results: SOTA & Optimal Performance

SOTA has large gap from ideal SSD
Evaluation Results: SSD Response Time

Up to 29% performance improvement when combined with SOTA
Other Analyses in the Paper

- Thorough analysis of read mechanism in modern SSDs

- More detailed results from real-device characterization
  - Effect of reducing individual read-timing parameters
  - Effect of reducing multiple read-timing parameters
  - Effect of operating temperature
  - How to choose the best read-timing parameters

- Detailed evaluation of PR² and AR² when applied individually

- Discussion of future directions to reduce SSD read latency
Executive Summary

- **Problem:** Long read latency in modern SSDs due to read-retry
  - Frequently requires multiple retry steps to read an erroneous page

- **Key Ideas:**
  - **Pipelined Read-Retry (PR²):** Concurrently perform consecutive retry steps using the CACHE READ command
  - **Adaptive Read-Retry (AR²):** Reduce read-timing parameters for every retry step by exploiting the reliability margin provided by strong ECC

- **Evaluation Results:** Our proposal improves SSD response time by
  - Up to 51% (35% on average) compared to a high-end SSD
  - Up to 32% (17% on average) compared to a state-of-the-art baseline

We hope that our key idea and characterization results inspire many valuable studies going forward
Reducing Solid-State Drive Read Latency by Optimizing Read-Retry

Jisung Park¹, Myungsuk Kim², Myoungjun Chun², Lois Orosa¹, Jihong Kim², and Onur Mutlu¹

¹ SAFARI
² ETH zürich

ASPLOS 2021 (Session 17: Solid State Drives)