Reducing SSD Read Latency via NAND Flash Program and Erase Suspension

USENIX FAST ’12
Introduction

• In flash memories, program/Erase (P/E) operations are much slower than read
  – Read (us)
  – Program (ms), erase (s)
• Due to the long P/E latency, subsequent read should wait!
  – With FIFO, RD2 should wait for WT1’s completion
• With RPS scheduling, read operations can get service first
Motivation

• Once P/E operations are committed (is being processed in the flash), read operations must wait for a long time!
• With RPS scheduling,
  – RD3 is requested after WT1 is committed
  – There is no way to suspend/preempt on-going P/E operations
  – RD3 should wait for WT1’s completion (cannot preempt!)
Motivation

- Disksim+SSD simulation (SLC/MLC) with six traces
- Read latency (normalized to FIFO) is measured
  - PER (P/E latency is set to read latency)
  - PEO (P/E latency is set to zero)
  - PER or PEO setting means that when there’s a read, it can get service right away (PEO) or with a small delay (PER)
- **Goal:** reducing read latency more as PER/PE0 results
Solution

- **Idea:** making P/E operation suspendable/resumable
- Read requests can preempt on-going (committed) program requests
- In the viewpoint of read requests, PER/PE0 effects can be expected (very small/no wait time)
Background: Erase

- (1) Using long pulse of erase voltage for $T_{\text{erase}}$ (3ms), electrons in the target cells are removed
- (2) Checking whether the erase is successful or not (8us)
- Voltage reset (4us)
  - Wires need to be reset for different operations
  - Flash works at different voltage bias for different operations
Background: Program

- Bus: data is transferred into chip-inside register for program
- Incremental Step Pulse Programming (ISPP)
  - Each “program” applies the required program voltage
  - A pair of “program” and following “verify” is repeated
  - Each “program” and “verify” also include “voltage reset”
Design: Suspend Erase

- **Case1**: read arrives “during voltage reset”
- After completing “voltage reset”, erase is suspended

- **Case2**: read arrives “in the middle of erase pulse or verify”
- “Erase pulse” is suspended, followed by “voltage reset”
- After completing “voltage reset”, erase is suspended
Design: Resume Erase

- **Case 1:** Read arrives “during voltage reset”
  - Since “erase pulse” is completed before suspension, **only “verify” is required to redo**

- **Case 2:** Read arrives “in the middle of erase pulse or verify”
  - The remaining time for “erase pulse” should be completed first
  - The status before suspension should be kept by control logic
Design: Suspend Program

- Option1: Inter-Phase Suspension (IPS)
  - After completing “current phase (Program / Verify)”, suspend ISPP
  - **Read should wait for the completion of current phase**

- Option2: Intra-Phase Cancellation (IPC)
  - After canceling the current phase, suspend ISPP
  - Read can get service a little earlier than IPS (option1)
Design: Resume Program

• Due to read, data in register is polluted, need to fill it again

• Inter-Phase Suspension (IPS): resume the remaining phases

• Intra-Phase Cancellation (IPC)
  – Since “the last program phase” might be correctly finished, “verify” first for “the last program phase”
  – If needed, redo “the last program phase” and resume ISPP
Evaluation: Read Latency

- PES_IPC shows near optimal read latency (near PER/PE0)
- PES_IPC shows better read latency than PES_IPS
  - Read can get service with negligible wait time under PES_IPC
Evaluation: Program Overhead

• Since we sacrifice “program” for faster “read service”, there is program overhead
• Overhead caused by IPC & IPS is on average under 5%
Conclusions

- “P/E” latency is much longer than “read” in flash
  - “Read” suffers from the long “P/E”
  - For already committed (on-going) P/E, “read” should wait

- “P/E” suspension/resumption is proposed
  - During P/E service, read operations can preempt
  - Near-optimal read latency (no wait time) can be achieved
  - Reasonable “program” overhead is charged