Memory Systems in the Multi-Core Era
Lecture 2.2: Emerging Technologies and Hybrid Memories

Prof. Onur Mutlu
http://www.ece.cmu.edu/~omutlu
onur@cmu.edu
Bogazici University
June 14, 2013
What Will You Learn in Mini Course 2?

- **Memory Systems in the Multi-Core Era**
  - June 13, 14, 17 (1-4pm)

- Lecture 1: Main memory basics, DRAM scaling
- Lecture 2: Emerging memory technologies and hybrid memories
- Lecture 3: Main memory interference and QoS

- Major Overview Reading:
Readings and Videos
Memory Lecture Videos

- Memory Hierarchy (and Introduction to Caches)
  - [http://www.youtube.com/watch?v=JBdfZ5i21cs&list=PL5PHm2jkkXmidJOD59REog9jDnPDTG6IJ&index=22](http://www.youtube.com/watch?v=JBdfZ5i21cs&list=PL5PHm2jkkXmidJOD59REog9jDnPDTG6IJ&index=22)

- Main Memory
  - [http://www.youtube.com/watch?v=ZLCy3pG7Rc0&list=PL5PHm2jkkXmidJOD59REog9jDnPDTG6IJ&index=25](http://www.youtube.com/watch?v=ZLCy3pG7Rc0&list=PL5PHm2jkkXmidJOD59REog9jDnPDTG6IJ&index=25)

- Memory Controllers, Memory Scheduling, Memory QoS
  - [http://www.youtube.com/watch?v=ZSotvL3WXmA&list=PL5PHm2jkkXmidJOD59REog9jDnPDTG6IJ&index=26](http://www.youtube.com/watch?v=ZSotvL3WXmA&list=PL5PHm2jkkXmidJOD59REog9jDnPDTG6IJ&index=26)
  - [http://www.youtube.com/watch?v=1xe2w3_NzmI&list=PL5PHm2jkkXmidJOD59REog9jDnPDTG6IJ&index=27](http://www.youtube.com/watch?v=1xe2w3_NzmI&list=PL5PHm2jkkXmidJOD59REog9jDnPDTG6IJ&index=27)

- Emerging Memory Technologies
  - [http://www.youtube.com/watch?v=LzfoGhMKyA0&list=PL5PHm2jkkXmidJOD59REog9jDnPDTG6IJ&index=35](http://www.youtube.com/watch?v=LzfoGhMKyA0&list=PL5PHm2jkkXmidJOD59REog9jDnPDTG6IJ&index=35)

- Multiprocessor Correctness and Cache Coherence
  - [http://www.youtube.com/watch?v=U-VZKMgItDM&list=PL5PHm2jkkXmidJOD59REog9jDnPDTG6IJ&index=32](http://www.youtube.com/watch?v=U-VZKMgItDM&list=PL5PHm2jkkXmidJOD59REog9jDnPDTG6IJ&index=32)
Readings for Lecture 2.1 (DRAM Scaling)

Readings for Lecture 2.2 (Emerging Technologies)

Readings for Lecture 2.3 (Memory QoS)

Readings for Lecture 2.3 (Memory QoS)

- Ebrahimi et al., “Parallel Application Memory Scheduling,” MICRO 2011.

- More to come in next lecture...
Readings in Flash Memory


Online Lectures and More Information

- **Online Computer Architecture Lectures**
  - [http://www.youtube.com/playlist?list=PL5PHm2jkkXmidJ0d59REog9jDnPDTG6IJ](http://www.youtube.com/playlist?list=PL5PHm2jkkXmidJ0d59REog9jDnPDTG6IJ)

- **Online Computer Architecture Courses**
  - Advanced: [http://www.ece.cmu.edu/~ece742/doku.php](http://www.ece.cmu.edu/~ece742/doku.php)

- **Recent Research Papers**
  - [http://users.ece.cmu.edu/~omutlu/projects.htm](http://users.ece.cmu.edu/~omutlu/projects.htm)
  - [http://scholar.google.com/citations?user=7XyGUGkAAAAJ&hl=en](http://scholar.google.com/citations?user=7XyGUGkAAAAJ&hl=en)
Emerging Memory Technologies
Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
- Conclusions
- Discussion
Major Trends Affecting Main Memory (I)

- Need for main memory capacity and bandwidth increasing
- Main memory energy/power is a key system design concern
- DRAM technology scaling is ending
Demand for Memory Capacity

- More cores ➔ More concurrency ➔ Larger working set

- Emerging applications are data-intensive

- Many applications/virtual machines (will) share main memory
  - Cloud computing/servers: Consolidation to improve efficiency
  - GP-GPUs: Many threads from multiple parallel applications
  - Mobile: Interactive + non-interactive consolidation
The Memory Capacity Gap

Core count doubling ~ every 2 years
DRAM DIMM capacity doubling ~ every 3 years

- Memory capacity per core expected to drop by 30% every two years
Major Trends Affecting Main Memory (II)

- Need for main memory capacity and bandwidth increasing
  - Multi-core: increasing number of cores
  - Data-intensive applications: increasing demand/hunger for data
  - Consolidation: Cloud computing, GPUs, mobile

- Main memory energy/power is a key system design concern

- DRAM technology scaling is ending
Major Trends Affecting Main Memory (III)

- Need for main memory capacity and bandwidth increasing

- Main memory energy/power is a key system design concern
  - IBM servers: ~50% energy spent in off-chip memory hierarchy [Lefurgy, IEEE Computer 2003]
  - DRAM consumes power when idle and needs periodic refresh

- DRAM technology scaling is ending
Major Trends Affecting Main Memory (IV)

- Need for main memory capacity and bandwidth increasing

- Main memory energy/power is a key system design concern

- DRAM technology scaling is ending
  - ITRS projects DRAM will not scale easily below 40nm
  - Scaling has provided many benefits:
    - higher capacity, higher density, lower cost, lower energy
The DRAM Scaling Problem

- DRAM stores charge in a capacitor (charge-based memory)
  - Capacitor must be large enough for reliable sensing
  - Access transistor should be large enough for low leakage and high retention time
  - Scaling beyond 40-35nm (2013) is challenging [ITRS, 2009]

- DRAM capacity, cost, and energy/power hard to scale
Trends: Problems with DRAM as Main Memory

- Need for main memory capacity and bandwidth increasing
  - DRAM capacity hard to scale

- Main memory energy/power is a key system design concern
  - DRAM consumes high power due to leakage and refresh

- DRAM technology scaling is ending
  - DRAM capacity, cost, and energy/power hard to scale
Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
- Conclusions
- Discussion
Requirements from an Ideal Memory System

- **Traditional**
  - Enough capacity
  - Low cost
  - High system performance (high bandwidth, low latency)

- **New**
  - Technology scalability: lower cost, higher capacity, lower energy
  - Energy (and power) efficiency
  - QoS support and configurability (for consolidation)
Requirements from an Ideal Memory System

- Traditional
  - Higher capacity
  - Continuous low cost
  - High system performance \((\text{higher bandwidth, low latency})\)

- New
  - Technology scalability: lower cost, higher capacity, lower energy
  - Energy (and power) efficiency
  - QoS support and configurability (for consolidation)

Emerging, resistive memory technologies (NVM) can help
Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- **Opportunity: Emerging Memory Technologies**
- Conclusions
- Discussion
The Promise of Emerging Technologies

- Likely need to replace/augment DRAM with a technology that is
  - Technology scalable
  - And at least similarly efficient, high performance, and fault-tolerant
    - or can be architected to be so

- Some emerging resistive memory technologies appear promising
  - Phase Change Memory (PCM)?
  - Spin Torque Transfer Magnetic Memory (STT-MRAM)?
  - Memristors?
  - And, maybe there are other ones

- Can they be enabled to replace/augment/surpass DRAM?
Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- **Opportunity: Emerging Memory Technologies**
  - Background
  - PCM (or Technology X) as DRAM Replacement
  - Hybrid Memory Systems
- Conclusions
- Discussion
Charge vs. Resistive Memories

- Charge Memory (e.g., DRAM, Flash)
  - Write data by capturing charge $Q$
  - Read data by detecting voltage $V$

- Resistive Memory (e.g., PCM, STT-MRAM, memristors)
  - Write data by pulsing current $dQ/dt$
  - Read data by detecting resistance $R$
Limits of Charge Memory

- Difficult charge placement and control
  - Flash: floating gate charge
  - DRAM: capacitor charge, transistor leakage

- Reliable sensing becomes difficult as charge storage unit size reduces
Emerging Resistive Memory Technologies

- **PCM**
  - Inject current to change material phase
  - Resistance determined by phase

- **STT-MRAM**
  - Inject current to change magnet polarity
  - Resistance determined by polarity

- **Memristors**
  - Inject current to change atomic structure
  - Resistance determined by atom distance
What is Phase Change Memory?

- Phase change material (chalcogenide glass) exists in two states:
  - Amorphous: Low optical reflexivity and high electrical resistivity
  - Crystalline: High optical reflexivity and low electrical resistivity

PCM is resistive memory: High resistance (0), Low resistance (1)

PCM cell can be switched between states reliably and quickly
How Does PCM Work?

- **Write**: change phase via current injection
  - SET: sustained current to heat cell above $T_{\text{cryst}}$
  - RESET: cell heated above $T_{\text{melt}}$ and quenched

- **Read**: detect phase via material resistance
  - amorphous/crystalline

---

**Large Current**

SET (cryst)
Low resistance

$10^3 - 10^4 \, \Omega$

**Small Current**

RESET (amorph)
High resistance

$10^6 - 10^7 \, \Omega$

Photo Courtesy: Bipin Rajendran, IBM  
Slide Courtesy: Moinuddin Qureshi, IBM
Opportunity: PCM Advantages

- **Scales better than DRAM, Flash**
  - Requires current pulses, which scale linearly with feature size
  - Expected to scale to 9nm (2022 [ITRS])
  - Prototyped at 20nm (Raoux+, IBM JRD 2008)

- **Can be denser than DRAM**
  - Can store multiple bits per cell due to large resistance range
  - Prototypes with 2 bits/cell in ISSCC’08, 4 bits/cell by 2012

- **Non-volatile**
  - Retain data for >10 years at 85°C

- **No refresh needed, low idle power**
Phase Change Memory Properties

- Surveyed prototypes from 2003-2008 (ITRS, IEDM, VLSI, ISSCC)
- Derived PCM parameters for F=90nm

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Horri$^6$</th>
<th>Ahn$^{12}$</th>
<th>Bedeschi$^{13}$</th>
<th>Oh$^{14}$</th>
<th>Pellizer$^{15}$</th>
<th>Chen$^{5}$</th>
<th>Kang$^{16}$</th>
<th>Bedeschi$^9$</th>
<th>Lee$^{10}$</th>
<th>Lee$^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process, F(nm)</td>
<td>**</td>
<td>120</td>
<td>180</td>
<td>120</td>
<td>90</td>
<td>**</td>
<td>100</td>
<td>90</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>Array size (Mbytes)</td>
<td>**</td>
<td>64</td>
<td>8</td>
<td>64</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>256</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td>Material</td>
<td>GST, N-d</td>
<td>GST, N-d</td>
<td>GST</td>
<td>GST</td>
<td>GST</td>
<td>GS, N-d</td>
<td>GST</td>
<td>GST</td>
<td>GST, N-d</td>
<td></td>
</tr>
<tr>
<td>Cell size ($\mu m^2$)</td>
<td>**</td>
<td>0.290</td>
<td>0.290</td>
<td>**</td>
<td>0.097</td>
<td>60 nm$^2$</td>
<td>0.166</td>
<td>0.097</td>
<td>**</td>
<td>0.085 to 0.097</td>
</tr>
<tr>
<td>Cell size, F$^2$</td>
<td>**</td>
<td>20.1</td>
<td>9.0</td>
<td>**</td>
<td>12.0</td>
<td>**</td>
<td>16.6</td>
<td>12.0</td>
<td>5.8</td>
<td>9.0 to 12.0</td>
</tr>
<tr>
<td>Access device</td>
<td>**</td>
<td>**</td>
<td>BJT</td>
<td>**</td>
<td>**</td>
<td>FET</td>
<td>**</td>
<td>FET</td>
<td>BJT</td>
<td>**</td>
</tr>
<tr>
<td>Read time (ns)</td>
<td>**</td>
<td>70</td>
<td>48</td>
<td>68</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>62</td>
<td>**</td>
<td>55</td>
</tr>
<tr>
<td>Read current ($\mu A$)</td>
<td>**</td>
<td>**</td>
<td>40</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>Read voltage (V)</td>
<td>**</td>
<td>3.0</td>
<td>1.0</td>
<td>1.8</td>
<td>1.6</td>
<td>**</td>
<td>**</td>
<td>1.8</td>
<td>**</td>
<td>40</td>
</tr>
<tr>
<td>Read power ($\mu W$)</td>
<td>**</td>
<td>**</td>
<td>40</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>Read energy (pJ)</td>
<td>**</td>
<td>**</td>
<td>2.0</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>Set time (ns)</td>
<td>100</td>
<td>150</td>
<td>150</td>
<td>180</td>
<td>**</td>
<td>80</td>
<td>300</td>
<td>**</td>
<td>400</td>
<td>150</td>
</tr>
<tr>
<td>Set current ($\mu A$)</td>
<td>200</td>
<td>200</td>
<td>300</td>
<td>200</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>Set voltage (V)</td>
<td>**</td>
<td>**</td>
<td>2.0</td>
<td>**</td>
<td>**</td>
<td>1.25</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>Set power ($\mu W$)</td>
<td>**</td>
<td>**</td>
<td>300</td>
<td>**</td>
<td>**</td>
<td>34.4</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>Set energy (pJ)</td>
<td>**</td>
<td>**</td>
<td>45</td>
<td>**</td>
<td>**</td>
<td>2.8</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>Reset time (ns)</td>
<td>50</td>
<td>10</td>
<td>40</td>
<td>10</td>
<td>**</td>
<td>60</td>
<td>50</td>
<td>**</td>
<td>50</td>
<td>40</td>
</tr>
<tr>
<td>Reset current ($\mu A$)</td>
<td>600</td>
<td>600</td>
<td>600</td>
<td>600</td>
<td>**</td>
<td>90</td>
<td>600</td>
<td>300</td>
<td>600</td>
<td>300</td>
</tr>
<tr>
<td>Reset voltage (V)</td>
<td>**</td>
<td>2.7</td>
<td>**</td>
<td>1.8</td>
<td>1.6</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>Reset power ($\mu W$)</td>
<td>**</td>
<td>**</td>
<td>1620</td>
<td>**</td>
<td>**</td>
<td>80.4</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>Reset energy (pJ)</td>
<td>**</td>
<td>**</td>
<td>64.8</td>
<td>**</td>
<td>**</td>
<td>4.8</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>Write endurance (MLC)</td>
<td>$10^7$</td>
<td>$10^8$</td>
<td>**</td>
<td>$10^8$</td>
<td>$10^4$</td>
<td>**</td>
<td>$10^6$</td>
<td>$10^5$</td>
<td>$10^8$</td>
<td>$10^8$</td>
</tr>
</tbody>
</table>

* BJT: bipolar junction transistor; FET: field-effect transistor; GST: Ge$_2$Sb$_2$Te$_5$; MLC: multilevel cells; N-d: nitrogen doped.

** This information is not available in the publication cited.
Phase Change Memory Properties: Latency

- Latency comparable to, but slower than DRAM

- Read Latency
  - 50ns: 4x DRAM, $10^{-3}$x NAND Flash

- Write Latency
  - 150ns: 12x DRAM

- Write Bandwidth
  - 5-10 MB/s: 0.1x DRAM, 1x NAND Flash
Phase Change Memory Properties

- **Dynamic Energy**
  - 40 uA Rd, 150 uA Wr
  - 2-43x DRAM, 1x NAND Flash

- **Endurance**
  - Writes induce phase change at 650C
  - Contacts degrade from thermal expansion/contraction
  - $10^8$ writes per cell
  - 10^{-8}x DRAM, 10^3x NAND Flash

- **Cell Size**
  - 9-12F^2 using BJT, single-level cells
  - 1.5x DRAM, 2-3x NAND (will scale with feature size, MLC)
Phase Change Memory: Pros and Cons

- Pros over DRAM
  - Better technology scaling
  - Non volatility
  - Low idle power (no refresh)

- Cons
  - Higher latencies: ~4-15x DRAM (especially write)
  - Higher active energy: ~2-50x DRAM (especially write)
  - Lower endurance (a cell dies after ~10^8 writes)

- Challenges in enabling PCM as DRAM replacement/helper:
  - Mitigate PCM shortcomings
  - Find the right way to place PCM in the system
  - Ensure secure and fault-tolerant PCM operation
PCM-based Main Memory: Research Challenges

- Where to place PCM in the memory hierarchy?
  - Hybrid OS controlled PCM-DRAM
  - Hybrid OS controlled PCM and hardware-controlled DRAM
  - Pure PCM main memory

- How to mitigate shortcomings of PCM?

- How to minimize amount of DRAM in the system?

- How to take advantage of (byte-addressable and fast) non-volatile main memory?

- Can we design specific-NVM-technology-agnostic techniques?
PCM-based Main Memory (I)

- How should PCM-based (main) memory be organized?

- **Hybrid PCM+DRAM** [Qureshi+ ISCA’09, Dhiman+ DAC’09, Meza+ IEEE CAL’12]:
  - How to partition/migrate data between PCM and DRAM
Hybrid Memory Systems: Challenges

- **Partitioning**
  - Should DRAM be a cache or main memory, or configurable?
  - What fraction? How many controllers?

- **Data allocation/movement (energy, performance, lifetime)**
  - Who manages allocation/movement?
  - What are good control algorithms?
  - How do we prevent degradation of service due to wearout?

- **Design of cache hierarchy, memory controllers, OS**
  - Mitigate PCM shortcomings, exploit PCM advantages

- **Design of PCM/DRAM chips and modules**
  - Rethink the design of PCM/DRAM with new requirements
How should PCM-based (main) memory be organized?

- Pure PCM main memory [Lee et al., ISCA’09, Top Picks’10]:
  - How to redesign entire hierarchy (and cores) to overcome PCM shortcomings
Aside: STT-RAM Basics

- Magnetic Tunnel Junction (MTJ)
  - Reference layer: Fixed
  - Free layer: Parallel or anti-parallel

- Cell
  - Access transistor, bit/sense lines

- Read and Write
  - Read: Apply a small voltage across bitline and senseline; read the current.
  - Write: Push large current through MTJ. Direction of current determines new orientation of the free layer.

Aside: STT MRAM: Pros and Cons

- **Pros over DRAM**
  - Better technology scaling
  - Non volatility
  - Low idle power (no refresh)

- **Cons**
  - Higher write latency
  - Higher write energy
  - Reliability?

- **Another level of freedom**
  - Can trade off non-volatility for lower write latency/energy (by reducing the size of the MTJ)
Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- **Opportunity: Emerging Memory Technologies**
  - Background
  - PCM (or Technology X) as DRAM Replacement
  - Hybrid Memory Systems
- Conclusions
- Discussion
An Initial Study: Replace DRAM with PCM

  - Surveyed prototypes from 2003-2008 (e.g. IEDM, VLSI, ISSCC)
  - Derived “average” PCM parameters for F=90nm

Density
- 9 - 12\(F^2\) using BJT
- 1.5× DRAM

Latency
- 50ns Rd, 150ns Wr
- 4×, 12× DRAM

Endurance
- 1E+08 writes
- 1E-08× DRAM

Energy
- 40\(\mu\)A Rd, 150\(\mu\)A Wr
- 2×, 43× DRAM
Results: Naïve Replacement of DRAM with PCM

- Replace DRAM with PCM in a 4-core, 4MB L2 system
- PCM organized the same as DRAM: row buffers, banks, peripherals
- 1.6x delay, 2.2x energy, 500-hour average lifetime

Architecting PCM to Mitigate Shortcomings

- Idea 1: Use multiple narrow row buffers in each PCM chip
  → Reduces array reads/writes → better endurance, latency, energy

- Idea 2: Write into array at cache block or word granularity
  → Reduces unnecessary wear
Results: Architected PCM as Main Memory

- 1.2x delay, 1.0x energy, 5.6-year average lifetime
- Scaling improves energy, endurance, density

Caveat 1: Worst-case lifetime is much shorter (no guarantees)
Caveat 2: Intensive applications see large performance and energy hits
Caveat 3: Optimistic PCM parameters?
Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
  - Background
  - PCM (or Technology X) as DRAM Replacement
  - Hybrid Memory Systems
- Conclusions
- Discussion
Hybrid Memory Systems

Hardware/software manage data allocation and movement to achieve the best of multiple technologies

One Option: DRAM as a Cache for PCM

- PCM is main memory; DRAM caches memory rows/blocks
  - Benefits: Reduced latency on DRAM cache hit; write filtering
- Memory controller hardware manages the DRAM cache
  - Benefit: Eliminates system software overhead

- Three issues:
  - What data should be placed in DRAM versus kept in PCM?
  - What is the granularity of data movement?
  - How to design a low-cost hardware-managed DRAM cache?

- Two idea directions:
  - Locality-aware data placement [Yoon+, ICCD 2012]
  - Cheap tag stores and dynamic granularity [Meza+, IEEE CAL 2012]
DRAM as a Cache for PCM

- Goal: Achieve the best of both DRAM and PCM/NVM
  - Minimize amount of DRAM w/o sacrificing performance, endurance
  - DRAM as cache to tolerate PCM latency and write bandwidth
  - PCM as main memory to provide large capacity at good cost and power
Write Filtering Techniques

- Lazy Write: Pages from disk installed only in DRAM, not PCM
- Partial Writes: Only dirty lines from DRAM page written back
- Page Bypass: Discard pages with poor reuse on DRAM eviction

Results: DRAM as PCM Cache (I)

- Simulation of 16-core system, 8GB DRAM main-memory at 320 cycles, HDD (2 ms) with Flash (32 us) with Flash hit-rate of 99%
- Assumption: PCM 4x denser, 4x slower than DRAM
- DRAM block size = PCM page size (4kB)
Results: DRAM as PCM Cache (II)

- PCM-DRAM Hybrid performs similarly to similar-size DRAM
- Significant power and energy savings with PCM-DRAM Hybrid
- Average lifetime: 9.7 years (no guarantees)
Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
  - Background
  - PCM (or Technology X) as DRAM Replacement
  - Hybrid Memory Systems
    - Row-Locality Aware Data Placement
    - Efficient DRAM (or Technology X) Caches
- Conclusions
- Discussion
Row Buffer Locality Aware Caching Policies for Hybrid Memories

HanBin Yoon
Justin Meza
Rachata Ausavarungnirun
Rachael Harding
Onur Mutlu

Carnegie Mellon University
Hybrid Memory

• Key question: How to place data between the heterogeneous memory devices?
Outline

• Background: Hybrid Memory Systems
• Motivation: Row Buffers and Implications on Data Placement
• Mechanisms: Row Buffer Locality-Aware Caching Policies
• Evaluation and Results
• Conclusion
Hybrid Memory: A Closer Look

CPU

Memory channel

DRAM
(small capacity cache)

PCM
(large capacity store)

Row buffer
Row Buffers and Latency

Row (buffer) hit: Access data from row buffer → fast

Row (buffer) miss: Access data from cell array → slow
Key Observation

• Row buffers exist in both DRAM and PCM
  – Row hit latency **similar** in DRAM & PCM [Lee+ ISCA’09]
  – Row miss latency **small** in DRAM, **large** in PCM

• Place data in DRAM which
  – is likely to miss in the row buffer (**low row buffer locality**) → miss penalty is smaller in DRAM
    AND
  – is reused **many times** → cache only the data worth the movement cost and DRAM space
Let’s say a processor accesses four rows
RBL-Awareness: An Example

Let’s say a processor accesses four rows with different row buffer localities (RBL)

Case 1: RBL-Unaware Policy (state-of-the-art)
Case 2: RBL-Aware Policy (RBLA)
Case 1: RBL-\textit{Unaware} Policy

A row buffer locality-\textit{unaware} policy could place these rows in the following manner:

- **DRAM** (High RBL):
  - Row C
  - Row D

- **PCM** (Low RBL):
  - Row A
  - Row B
Case 1: RBL-Unaware Policy

Access pattern to main memory:
A (oldest), B, C, C, C, A, B, D, D, D, A, B (youngest)

DRAM (High RBL)

PCM (Low RBL)

RBL-Unaware: Stall time is 6 PCM device accesses
Case 2: RBL-Aware Policy (RBLA)

A row buffer locality-aware policy would place these rows in the opposite manner.

- **DRAM** (Low RBL)
  - Access data at lower row buffer miss latency of DRAM

- **PCM** (High RBL)
  - Access data at low row buffer hit latency of PCM
Case 2: RBL-Aware Policy (RBLA)

Access pattern to main memory:
A (oldest), B, C, C, C, A, B, D, D, D, A, B (youngest)

DRAM (High RBL)

PCM (Low RBL)

RBL-Unaware: Stall time is 6 PCM device accesses

DRAM (Low RBL)

PCM (High RBL)

RBL-Aware: Stall time is 6 DRAM device accesses
Outline

• Background: Hybrid Memory Systems
• Motivation: Row Buffers and Implications on Data Placement
• Mechanisms: Row Buffer Locality-Aware Caching Policies
• Evaluation and Results
• Conclusion
Our Mechanism: RBLA

1. For recently used rows in PCM:
   – Count row buffer misses as indicator of row buffer locality (RBL)

2. Cache to DRAM rows with $\text{misses} \geq \text{threshold}$
   – Row buffer miss counts are periodically reset (only cache rows with high reuse)
Our Mechanism: RBLA-Dyn

1. For recently used rows in PCM:
   – Count row buffer misses as indicator of row buffer locality (RBL)

2. Cache to DRAM rows with $\text{misses} \geq \text{threshold}$
   – Row buffer miss counts are periodically reset (only cache rows with high reuse)

3. Dynamically adjust threshold to adapt to workload/system characteristics
   – Interval-based cost-benefit analysis
Implementation: “Statistics Store”

• Goal: To keep count of row buffer misses to recently used rows in PCM

• Hardware structure in memory controller
  – Operation is similar to a cache
    • Input: row address
    • Output: row buffer miss count
  – 128-set 16-way statistics store (9.25KB) achieves system performance within 0.3% of an unlimited-sized statistics store
Outline

• Background: Hybrid Memory Systems
• Motivation: Row Buffers and Implications on Data Placement
• Mechanisms: Row Buffer Locality-Aware Caching Policies
• Evaluation and Results
• Conclusion
Evaluation Methodology

• Cycle-level x86 CPU-memory simulator
  – **CPU**: 16 out-of-order cores, 32KB private L1 per core, 512KB shared L2 per core
  – **Memory**: 1GB DRAM (8 banks), 16GB PCM (8 banks), 4KB migration granularity

• 36 multi-programmed server, cloud workloads
  – Server: TPC-C (OLTP), TPC-H (Decision Support)
  – Cloud: Apache (Webserv.), H.264 (Video), TPC-C/H

• Metrics: Weighted speedup (perf.), perf./Watt (energy eff.), Maximum slowdown (fairness)
Comparison Points

• **Conventional LRU Caching**

• **FREQ**: Access-frequency-based caching
  – Places “hot data” in cache [Jiang+ HPCA’10]
  – Cache to DRAM rows with accesses ≥ threshold
  – Row buffer locality-unaware

• **FREQ-Dyn**: Adaptive Freq.-based caching
  – **FREQ** + our dynamic threshold adjustment
  – Row buffer locality-unaware

• **RBLA**: Row buffer locality-aware caching

• **RBLA-Dyn**: Adaptive RBL-aware caching
System Performance

Benefit 1: Increased row buffer locality (RBL) in PCM by moving low RBL data to DRAM

Benefit 2: Reduced memory bandwidth consumption due to stricter caching criteria

Benefit 3: Balanced memory request load between DRAM and PCM
Average Memory Latency

- FREQ
- FREQ-Dyn
- RBLA
- RBLA-Dyn

Normalized Avg Memory Latency

Server: FREQ 14% improvement, FREQ-Dyn 14% improvement, RBLA 9% improvement, RBLA-Dyn 12% improvement.

Cloud: FREQ 14% improvement, FREQ-Dyn 14% improvement, RBLA 9% improvement, RBLA-Dyn 12% improvement.

Avg: FREQ 14% improvement, FREQ-Dyn 14% improvement, RBLA 9% improvement, RBLA-Dyn 12% improvement.
Memory Energy Efficiency

Increased performance & reduced data movement between DRAM and PCM
Thread Fairness

Normalized Maximum Slowdown

- **Server**: 7.6% decrease, 1.17
- **Cloud**: 4.8% decrease, 1.04
- **Avg**: 6.2% decrease, 1.06

**Workload**

- **FREQ**
- **FREQ-Dyn**
- **RBLA**
- **RBLA-Dyn**
Our mechanism achieves 31% better performance than all PCM, within 29% of all DRAM performance.
Other Results in Paper

• RBLA-Dyn increases the portion of PCM row buffer hit by 6.6 times

• RBLA-Dyn has the effect of balancing memory request load between DRAM and PCM
  – PCM channel utilization increases by 60%.
Summary

• Different memory technologies have different strengths
• A hybrid memory system (DRAM-PCM) aims for best of both
• **Problem:** How to place data between these heterogeneous memory devices?
• **Observation:** PCM array access latency is higher than DRAM’s – But peripheral circuit (row buffer) access latencies are similar
• **Key Idea:** Use row buffer locality (RBL) as a key criterion for data placement
• **Solution:** Cache to DRAM rows with low RBL and high reuse
• Improves both performance and energy efficiency over state-of-the-art caching policies
Row Buffer Locality Aware Caching Policies for Hybrid Memories

HanBin Yoon
Justin Meza
Rachata Ausavarungnirun
Rachael Harding
Onur Mutlu
Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
  - Background
  - PCM (or Technology X) as DRAM Replacement
  - Hybrid Memory Systems
    - Row-Locality Aware Data Placement
    - Efficient DRAM (or Technology X) Caches
- Conclusions
- Discussion
The Problem with Large DRAM Caches

- A large DRAM cache requires a large metadata (tag + block-based information) store
- How do we design an efficient DRAM cache?
Idea 1: Tags in Memory

- Store tags in the same row as data in DRAM
  - Store metadata in same row as their data
  - Data and metadata can be accessed together

- Benefit: No on-chip tag storage overhead
- Downsides:
  - Cache hit determined only after a DRAM access
  - Cache hit requires two DRAM accesses
Idea 2: Cache Tags in SRAM

- Recall Idea 1: Store all metadata in DRAM
  - To reduce metadata storage overhead

- Idea 2: Cache in on-chip SRAM frequently-accessed metadata
  - Cache only a small amount to keep SRAM size small
Idea 3: Dynamic Data Transfer Granularity

- Some applications benefit from caching more data
  - They have good spatial locality
- Others do not
  - Large granularity wastes bandwidth and reduces cache utilization

Idea 3: Simple dynamic caching granularity policy

- Cost-benefit analysis to determine best DRAM cache block size
- Group main memory into sets of rows
- Some row sets follow a fixed caching granularity
- The rest of main memory follows the best granularity
  - Cost–benefit analysis: access latency versus number of cachings
  - Performed every quantum
TIMBER Tag Management

- A Tag-In-Memory Buffer (TIMBER)
  - Stores recently-used tags in a small amount of SRAM

- Benefits: If tag is cached:
  - no need to access DRAM twice
  - cache hit determined quickly
Case 1: TIMBER hit

**Our proposal**

TIMBER: \( X \rightarrow \text{DRAM} \)

Access X

SAFARI
TIMBER Tag Management Example (II)

- Case 2: TIMBER miss

1. Access M(Y)
2. Cache M(Y)
3. Access Y (row hit)
Methodology

- System: 8 out-of-order cores at 4 GHz

- Memory: 512 MB direct-mapped DRAM, 8 GB PCM
  - 128B caching granularity
  - DRAM row hit (miss): 200 cycles (400 cycles)
  - PCM row hit (clean / dirty miss): 200 cycles (640 / 1840 cycles)

- Evaluated metadata storage techniques
  - All SRAM system (8MB of SRAM)
  - Region metadata storage
  - TIM metadata storage (same row as data)
  - TIMBER, 64-entry direct-mapped (8KB of SRAM)
Metadata Storage Performance

<table>
<thead>
<tr>
<th>Normalized Weighted Speedup</th>
<th>SRAM (Ideal)</th>
<th>Region</th>
<th>TIM</th>
<th>TIMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.5</td>
<td>0.7</td>
<td>0.8</td>
</tr>
</tbody>
</table>

(SRAM denotes a specific type of memory storage, and TIM and TIMBER are likely abbreviations or names of different technologies or methods being compared.)
Metadata Storage Performance

Performance degrades due to increased metadata lookup access latency

<table>
<thead>
<tr>
<th>Normalized Weighted Speedup</th>
<th>SRAM (Ideal)</th>
<th>Region</th>
<th>TIM</th>
<th>TIMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>-48%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Metadata Storage Performance

- Increased row locality reduces average memory access latency
  - 36%

- SRAM (Ideal)
- Region
- TIM
- TIMBER

Normalized Weighted Speedup

95
Metadata Storage Performance

Data with locality can access metadata at SRAM latencies 23%
Dynamic Granularity Performance

- SRAM: 1
- Region: 50%
- TIM: 70%
- TIMBER: 90%
- TIMBER-Dyn: 90% (10% improvement)

Reduced channel contention and improved spatial locality.
TIMBER Performance

Reduced channel contention and improved spatial locality

-6%

TIMBER Energy Efficiency

Fewer migrations reduce transmitted data and channel contention

Enabling and Exploiting NVM: Issues

- Many issues and ideas from technology layer to algorithms layer

- Enabling NVM and hybrid memory
  - How to tolerate errors?
  - How to enable secure operation?
  - How to tolerate performance and power shortcomings?
  - How to minimize cost?

- Exploiting emerging technologies
  - How to exploit non-volatility?
  - How to minimize energy consumption?
  - How to exploit NVM on chip?
Security Challenges of Emerging Technologies

1. Limited endurance → Wearout attacks

2. Non-volatility → Data persists in memory after powerdown
   → Easy retrieval of privileged or private information

3. Multiple bits per cell → Information leakage (via side channel)
Securing Emerging Memory Technologies

1. Limited endurance → Wearout attacks
   - Better architecting of memory chips to absorb writes
   - Hybrid memory system management
   - Online wearout attack detection

2. Non-volatility → Data persists in memory after powerdown
   - Easy retrieval of privileged or private information
   - Efficient encryption/decryption of whole main memory
   - Hybrid memory system management

3. Multiple bits per cell → Information leakage (via side channel)
   - System design to hide side channel information
Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
  - Background
  - PCM (or Technology X) as DRAM Replacement
  - Hybrid Memory Systems
- Conclusions
- Discussion
Main memory scaling problems are a critical bottleneck for system performance, efficiency, and usability

Solution 1: Tolerate DRAM (yesterday)

Solution 2: Enable emerging memory technologies
- Replace DRAM with NVM by architecting NVM chips well
- Hybrid memory systems with automatic data management

We are examining many other solution directions and ideas
- Hardware/software/device cooperation essential
- Memory, storage, controller, software/app co-design needed
- Coordinated management of persistent memory and storage
- Application and hardware cooperative management of NVM
Flash Memory Scaling
Readings in Flash Memory


Flash memory widening its range of applications
- Portable consumer devices, laptop PCs and enterprise servers
Decreasing Endurance with Flash Scaling

- Endurance of flash memory decreasing with scaling and multi-level cells
- Error correction capability required to guarantee storage-class reliability (UBER < $10^{-15}$) is increasing exponentially to reach less endurance

UBER: Uncorrectable bit error rate. Fraction of erroneous bits after error correction.
Future NAND Flash Storage Architecture

- Memory Signal Processing
  - Read voltage adjusting
  - Data scrambler
  - Data recovery
  - Soft-information estimation

- Error Correction
  - Hamming codes
  - BCH codes
  - Reed-Solomon codes
  - LDPC codes
  - Other Flash friendly codes

BER < $10^{-15}$

Need to understand NAND flash error patterns
Test System Infrastructure

Software Platform
- USB Driver
- Host USB PHY

USB Daughter Board
- USB PHY Chip
- Control Firmware
- FPGA USB controller

Mother Board
- NAND Controller

Flash Board
- Flash Memories

Host Computer

1. Reset
2. Erase block
3. Program page
4. Read page

Algorithms
- Wear Leveling
- Address Mapping
- Garbage Collection
- ECC (BCH, RS, LDPC)

Signal Processing

SAFARI
NAND Flash Testing Platform

- USB Daughter Board
- USB Jack
- HAPS-52 Mother Board
- Virtex-V FPGA (NAND Controller)
- Virtex-II Pro (USB controller)
- 3x-nm NAND Flash
- NAND Daughter Board
NAND Flash Usage and Error Model

Start

P/E cycle 0

...

P/E cycle i

...

P/E cycle n

End of life

Erase Errors

Program Errors

Erase Block

Program Page

Retention1 (t₁ days)

Read Page

Retention Errors

Retention j (tⱼ days)

Read Page

Retention Errors

Retention Errors
Error Types and Testing Methodology

- **Erase errors**
  - Count the number of cells that fail to be erased to “11” state

- **Program interference errors**
  - Compare the data immediately after page programming and the data after the whole block being programmed

- **Read errors**
  - Continuously read a given block and compare the data between consecutive read read sequences

- **Retention errors**
  - Compare the data read after an amount of time to data written
    - Characterize short term retention errors under room temperature
    - Characterize long term retention errors by baking in the oven under 125°C
Observations: Flash Error Analysis

- Raw bit error rate increases exponentially with P/E cycles
- Retention errors are dominant (>99% for 1-year ret. time)
- Retention errors increase with retention time requirement
Electron loss from the floating gate causes retention errors
- Cells with more programmed electrons suffer more from retention errors
- Threshold voltage is more likely to shift by one window than by multiple
Cells with more programmed electrons tend to suffer more from retention noise (i.e. 00 and 01)
More Details on Flash Error Analysis

- Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai, "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis"
Threshold Voltage Distribution Shifts

As P/E cycles increase ...

- Distribution shifts to the right
- Distribution becomes wider
Flash Correct-and-Refresh

Retention-Aware Error Management for Increased Flash Memory Lifetime

Yu Cai\textsuperscript{1} Gulay Yalcin\textsuperscript{2} Onur Mutlu\textsuperscript{1} Erich F. Haratsch\textsuperscript{3}
Adrian Cristal\textsuperscript{2} Osman S. Unsal\textsuperscript{2} Ken Mai\textsuperscript{1}

\textsuperscript{1} Carnegie Mellon University
\textsuperscript{2} Barcelona Supercomputing Center
\textsuperscript{3} LSI Corporation
Executive Summary

- NAND flash memory has low endurance: a flash cell dies after 3k P/E cycles vs. 50k desired → Major scaling challenge for flash memory
- Flash error rate increases exponentially over flash lifetime
- **Problem:** Stronger error correction codes (ECC) are ineffective and undesirable for improving flash lifetime due to
  - diminishing returns on lifetime with increased correction strength
  - prohibitively high power, area, latency overheads
- **Our Goal:** Develop techniques to tolerate high error rates w/o strong ECC
- **Observation:** Retention errors are the dominant errors in MLC NAND flash
  - flash cell loses charge over time; retention errors increase as cell gets worn out
- **Solution:** Flash Correct-and-Refresh (FCR)
  - Periodically read, correct, and reprogram (in place) or remap each flash page before it accumulates more errors than can be corrected by simple ECC
  - Adapt “refresh” rate to the severity of retention errors (i.e., # of P/E cycles)
- **Results:** FCR improves flash memory lifetime by 46X with no hardware changes and low energy overhead; outperforms strong ECCs
Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
- Evaluation
- Conclusions
Problem: Limited Endurance of Flash Memory

- NAND flash has limited endurance
  - A cell can tolerate a small number of Program/Erase (P/E) cycles
  - 3x-nm flash with 2 bits/cell $\rightarrow$ 3K P/E cycles

- Enterprise data storage requirements demand very high endurance
  - $>$50K P/E cycles (10 full disk writes per day for 3-5 years)

- Continued process scaling and more bits per cell will reduce flash endurance

- One potential solution: stronger error correction codes (ECC)
  - Stronger ECC not effective enough and inefficient
Decreasing Endurance with Flash Scaling

- Endurance of flash memory decreasing with scaling and multi-level cells
- Error correction capability required to guarantee storage-class reliability (UBER < 10^{-15}) is increasing exponentially to reach less endurance

UBER: Uncorrectable bit error rate. Fraction of erroneous bits after error correction.
The Problem with Stronger Error Correction

- Stronger ECC detects and corrects more raw bit errors → increases P/E cycles endured

- Two shortcomings of stronger ECC:
  1. High implementation complexity
     → Power and area overheads increase super-linearly, but correction capability increases sub-linearly with ECC strength
  2. Diminishing returns on flash lifetime improvement
     → Raw bit error rate increases exponentially with P/E cycles, but correction capability increases sub-linearly with ECC strength
Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
- Evaluation
- Conclusions
Methodology: Error and ECC Analysis

- Characterized errors and error rates of 3x-nm MLC NAND flash using an experimental FPGA-based flash platform

- Quantified Raw Bit Error Rate (RBER) at a given P/E cycle
  - Raw Bit Error Rate: Fraction of erroneous bits without any correction

- Quantified error correction capability (and area and power consumption) of various BCH-code implementations
  - Identified how much RBER each code can tolerate
    → how many P/E cycles (flash lifetime) each code can sustain
NAND Flash Error Types

- Four types of errors [Cai+, DATE 2012]

- Caused by common flash operations
  - Read errors
  - Erase errors
  - Program (interference) errors

- Caused by flash cell losing charge over time
  - Retention errors
    - Whether an error happens depends on required retention time
    - Especially problematic in MLC flash because voltage threshold window to determine stored value is smaller
Observations: Flash Error Analysis

- Raw bit error rate increases exponentially with P/E cycles
- Retention errors are dominant (>99% for 1-year ret. time)
- Retention errors increase with retention time requirement

![Graph showing retention errors vs. P/E cycles]
Methodology: Error and ECC Analysis

- **Characterized errors and error rates** of 3x-nm MLC NAND flash using an experimental FPGA-based flash platform

- **Quantified Raw Bit Error Rate (RBER) at a given P/E cycle**
  - Raw Bit Error Rate: Fraction of erroneous bits without any correction

- **Quantified error correction capability** (and area and power consumption) of various BCH-code implementations
  - Identified how much RBER each code can tolerate
    → how many P/E cycles (flash lifetime) each code can sustain
ECC Strength Analysis

Error correction capability increases sub-linearly

Power and area overheads increase super-linearly

<table>
<thead>
<tr>
<th>Code length (n)</th>
<th>Correctable Errors (t)</th>
<th>Acceptable Raw BER</th>
<th>Norm. Power</th>
<th>Norm. Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>7</td>
<td>1.0x10^{-4} (1x)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1024</td>
<td>12</td>
<td>4.0x10^{-4} (4x)</td>
<td>2</td>
<td>2.1</td>
</tr>
<tr>
<td>2048</td>
<td>22</td>
<td>1.0x10^{-3} (10x)</td>
<td>4.1</td>
<td>3.9</td>
</tr>
<tr>
<td>4096</td>
<td>40</td>
<td>1.7x10^{-3} (17x)</td>
<td>8.6</td>
<td>10.3</td>
</tr>
<tr>
<td>8192</td>
<td>74</td>
<td>2.2x10^{-3} (22x)</td>
<td>17.8</td>
<td>21.3</td>
</tr>
<tr>
<td>32768</td>
<td>259</td>
<td>2.6x10^{-3} (26x)</td>
<td>71</td>
<td>85</td>
</tr>
</tbody>
</table>
Resulting Flash Lifetime with Strong ECC

- Lifetime improvement comparison of various BCH codes

Graph showing P/E Cycle Endurance for 512b-BCH, 1k-BCH, 2k-BCH, 4k-BCH, 8k-BCH, and 32k-BCH. The x-axis represents different BCH codes, and the y-axis represents P/E Cycle Endurance. The graph indicates a 4X lifetime improvement for 32k-BCH. Additionally, it shows 71X power consumption and 85X area consumption improvement.

Strong ECC is very inefficient at improving lifetime.
Our Goal

Develop new techniques to improve flash lifetime without relying on stronger ECC
Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
- Evaluation
- Conclusions
Flash Correct-and-Refresh (FCR)

- **Key Observations:**
  - Retention errors are the dominant source of errors in flash memory [Cai+ DATE 2012][Tanakamaru+ ISSCC 2011]
    → limit flash lifetime as they increase over time
  - Retention errors can be corrected by “refreshing” each flash page periodically

- **Key Idea:**
  - Periodically read each flash page,
  - Correct its errors using “weak” ECC, and
  - Either remap it to a new physical page or reprogram it in-place,
  - Before the page accumulates more errors than ECC-correctable
  - Optimization: Adapt refresh rate to endured P/E cycles
**FCR Intuition**

### Errors with No refresh

- **Program Page**: 
  - Time T: 
  - Time 2T: 
  - Time 3T:

### Errors with Periodic refresh

- **Program Page**: 
  - Time T: 
  - Time 2T: 
  - Time 3T:

**Symbols:**
- × Retention Error
- × Program Error
FCR: Two Key Questions

- How to refresh?
  - Remap a page to another one
  - Reprogram a page (in-place)
  - Hybrid of remap and reprogram

- When to refresh?
  - Fixed period
  - Adapt the period to retention error severity
Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- **Flash Correct and Refresh Techniques (FCR)**
  1. Remapping based FCR
  2. Hybrid Reprogramming and Remapping based FCR
  3. Adaptive-Rate FCR
- Evaluation
- Conclusions
Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
  1. Remapping based FCR
  2. Hybrid Reprogramming and Remapping based FCR
  3. Adaptive-Rate FCR
- Evaluation
- Conclusions
Remapping Based FCR

- Idea: Periodically remap each page to a different physical page (after correcting errors)
  - Also [Pan et al., HPCA 2012]
  - FTL already has support for changing logical → physical flash block/page mappings
  - Deallocated block is erased by garbage collector

- Problem: Causes additional erase operations → more wearout
  - Bad for read-intensive workloads (few erases really needed)
  - Lifetime degrades for such workloads (see paper)
Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
  1. Remapping based FCR
  2. Hybrid Reprogramming and Remapping based FCR
  3. Adaptive-Rate FCR
- Evaluation
- Conclusions
In-Place Reprogramming Based FCR

- **Idea:** Periodically reprogram (in-place) each physical page (after correcting errors)

  - Flash programming techniques (ISPP) can correct retention errors in-place by recharging flash cells

- **Problem:** Program errors accumulate on the same page → may not be correctable by ECC after some time

![Diagram of reprogramming process]

Select Block → Read Page Data → Error Correction → Reprogram corrected data → Select next Block

Page Num ++

Last Page? → Yes/No
In-Place Reprogramming of Flash Cells

Floating Gate Voltage Distribution for each Stored Value:

- **Pro:** No remapping needed → no additional erase operations
- **Con:** Increases the occurrence of program errors

Retention errors are caused by cell voltage shifting to the left.

ISPP moves cell voltage to the right; fixes retention errors.

<table>
<thead>
<tr>
<th>VT</th>
<th>REF1</th>
<th>REF2</th>
<th>REF3</th>
</tr>
</thead>
<tbody>
<tr>
<td>VT</td>
<td>11</td>
<td>10</td>
<td>01</td>
</tr>
</tbody>
</table>
Program Errors in Flash Memory

- When a cell is being programmed, voltage level of a neighboring cell changes (unintentionally) due to parasitic capacitance coupling
  - can change the data value stored

- Also called program interference error

- Program interference causes neighboring cell voltage to shift to the right
Problem with In-Place Reprogramming

Floating Gate Voltage Distribution

Original data to be programmed

Program errors after initial programming

Retention errors after some time

Errors after in-place reprogramming

Problem: Program errors can accumulate over time
Hybrid Reprogramming/Remapping Based FCR

**Idea:**
- Monitor the count of right-shift errors (after error correction)
- If count < threshold, in-place reprogram the page
- Else, remap the page to a new page

**Observation:**
- Program errors much less frequent than retention errors → Remapping happens only infrequently

**Benefit:**
- Hybrid FCR greatly reduces erase operations due to remapping
Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- **Flash Correct and Refresh Techniques (FCR)**
  1. Remapping based FCR
  2. Hybrid Reprogramming and Remapping based FCR
  3. Adaptive-Rate FCR
- Evaluation
- Conclusions
Adaptive-Rate FCR

- **Observation:**
  - Retention error rate strongly depends on the P/E cycles a flash page endured so far
  - No need to refresh frequently (at all) early in flash lifetime

- **Idea:**
  - Adapt the refresh rate to the P/E cycles endured by each page
  - Increase refresh rate gradually with increasing P/E cycles

- **Benefits:**
  - Reduces overhead of refresh operations
  - Can use existing FTL mechanisms that keep track of P/E cycles
Adaptive-Rate FCR (Example)

Select refresh frequency such that error rate is below acceptable rate.

Acceptable raw BER for 512b-BCH

3-year FCR

3-month FCR

3-week FCR

3-day FCR

P/E Cycles

Raw Bit Error Rate

3-year Errors

3-month Errors

3-week Errors

3-day Errors
Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
  1. Remapping based FCR
  2. Hybrid Reprogramming and Remapping based FCR
  3. Adaptive-Rate FCR
- Evaluation
- Conclusions
FCR: Other Considerations

- Implementation cost
  - No hardware changes
  - FTL software/firmware needs modification

- Response time impact
  - FCR not as frequent as DRAM refresh; low impact

- Adaptation to variations in retention error rate
  - Adapt refresh rate based on, e.g., temperature [Liu+ ISCA 2012]

- FCR requires power
  - Enterprise storage systems typically powered on
Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
- Evaluation
- Conclusions
Evaluation Methodology

- **Experimental flash platform** to obtain error rates at different P/E cycles [Cai+ DATE 2012]

- **Simulation framework** to obtain P/E cycles of real workloads: DiskSim with SSD extensions

- **Simulated system**: 256GB flash, 4 channels, 8 chips/channel, 8K blocks/chip, 128 pages/block, 8KB pages

- **Workloads**
  - File system applications, databases, web search
  - Categories: Write-heavy, read-heavy, balanced

- **Evaluation metrics**
  - Lifetime (extrapolated)
  - Energy overhead, P/E cycle overhead
Extrapolated Lifetime

- Maximum full disk P/E Cycles for a Technique
- Total full disk P/E Cycles for a Workload
- # of Days of Given Application
- Real length (in time) of each workload trace
Normalized Flash Memory Lifetime

- **Base (No-Refresh)**
- **Remapping-Based FCR**
- **Hybrid FCR**
- **Adaptive FCR**

**Lifetime of FCR much higher than lifetime of stronger ECC**

- Adaptive FCR provides the highest lifetime of FCR much higher than lifetime of stronger ECC.
Lifetime Evaluation Takeaways

- Significant average lifetime improvement over no refresh
  - Adaptive-rate FCR: 46X
  - Hybrid reprogramming/remapping based FCR: 31X
  - Remapping based FCR: 9X

- FCR lifetime improvement larger than that of stronger ECC
  - 46X vs. 4X with 32-kbit ECC (over 512-bit ECC)
  - FCR is less complex and less costly than stronger ECC

- Lifetime on all workloads improves with Hybrid FCR
  - Remapping based FCR can degrade lifetime on read-heavy WL
  - Lifetime improvement highest in write-heavy workloads
Energy Overhead

- **Adaptive-rate refresh**: <1.8% energy increase until daily refresh is triggered
Overhead of Additional Erases

- Additional erases happen due to remapping of pages
- Low (2%-20%) for write intensive workloads
- High (up to 10X) for read-intensive workloads

Improved P/E cycle lifetime of all workloads largely outweighs the additional P/E cycles due to remapping
More Results in the Paper

- Detailed workload analysis
- Effect of refresh rate
Outline

- Executive Summary
- The Problem: Limited Flash Memory Endurance/Lifetime
- Error and ECC Analysis for Flash Memory
- Flash Correct and Refresh Techniques (FCR)
- Evaluation
- Conclusions
Conclusion

- NAND flash memory lifetime is limited due to uncorrectable errors, which increase over lifetime (P/E cycles)

- **Observation**: Dominant source of errors in flash memory is retention errors $\rightarrow$ retention error rate limits lifetime

- **Flash Correct-and-Refresh (FCR) techniques** reduce retention error rate to improve flash lifetime
  - Periodically read, correct, and remap or reprogram each page before it accumulates more errors than can be corrected
  - Adapt refresh period to the severity of errors

- **FCR improves flash lifetime by 46X at no hardware cost**
  - More effective and efficient than stronger ECC
  - Can enable better flash memory scaling
Flash Correct-and-Refresh
Retention-Aware Error Management for Increased Flash Memory Lifetime

Yu Cai¹  Gulay Yalcin²  Onur Mutlu¹  Erich F. Haratsch³
Adrian Cristal²  Osman S. Unsal²  Ken Mai¹

¹ Carnegie Mellon University
² Barcelona Supercomputing Center
³ LSI Corporation