Fine-grained Power Budgeting
to Improve Write Throughput
of MLC PCM

1Lei Jiang, 2Youtao Zhang,
2Bruce R. Childers and 1Jun Yang

1Electrical and Computer Engineering Department
2Computer Science Department
University of Pittsburgh, Pittsburgh
Phase Change Memory (PCM)

ARM CortexA15
4 cores

Intel Xeon
8 cores

AMD Bulldozer
16 cores

# of Cores (C#) ↑

Memory Capacity ↑ = C# x WSST

small

large

DRAM

PCM

Working Set of Single Thread (WSST) ↑

Figures are from ARM, Intel, AMD, VoltDB, Memcached, MySQL and Samsung website
Multi-Level Cell and PCM write

Voltage vs. Time

- Glass Transition Temperature (~300℃)
- Melting Point (~600℃)

Large Resistance Difference

Capacity ↑ Cost-per-bit ↓

Higher than \( V_{dd} \) write voltage

Nondeterministic write

\( V_{reset} \)
\( V_{set,0} \)
\( V_{set,1} \)
\( V_{set,2} \)
\( V_{verify} \)

Glass Transition Temperature (~300℃)
Melting Point (~600℃)
Multi-Level Cell and PCM write

More write power and energy

Write is non-deterministic
**PCM DIMM and Chip Architecture**

1. **Bridge Chip** [FANG_PACT2011]: handles non-deterministic write
   - **Iteration Manager (IM)**: iterative programming algorithm

2. **Local Charge Pump (LCP)**: boosts voltage and current for writes
Power Constraint and Solution for SLC

- **DIMM level power constraint (DLPC)** [HAY_MICRO’11]
  - One DIMM only supports 560 concurrent RESETs (*power token*)
  - ~one 512-bit (64B) write – poor write throughput

- **SLC power management (SPM)** [HAY_MICRO’11]
  - Approximately estimate # of written cells in cache by MC
  - Allocate power tokens based on estimated number
  - Reclaim after a fixed write latency
  - Can write ~ 8 64B lines (assuming 15% cell changing rate)

![Graph showing speedup comparison between Ideal and SPM]

- Ideal
- SPM
- ~Full write throughput
A Different Story on MLC

- Higher power demand, but DLPC does not increase
  - MLC has larger write power
  - MLC needs larger memory line size and LLC
  - More cell changes, lower write throughput

- Nondeterministic write on MLC
  - Reclaim power tokens after a fixed latency?
  
  **Worst case write latency must be used → Power tokens wasted**

<table>
<thead>
<tr>
<th>Speedup</th>
<th>Ideal</th>
<th>SPM</th>
<th>SPM on MLC (DIMM only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SPM does NOT work on MLC!
In Addition: Chip Level Power Constraint

- Total # of cells written per chip is limited too
  - Introduced by local charge pump (LCP) [LEE_JSSCC’09]
  - LCP power supply ability $\propto$ LCP area

15%-20% area overhead
### DIMM and Chip Power Constraints Example

<table>
<thead>
<tr>
<th>Bank 0</th>
<th>Chip 0 budget</th>
<th>Chip 1 budget</th>
<th>Chip 2 budget</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00 00 00 00 00</td>
<td>11 11 11 11 11</td>
<td>00 00 00 00 00</td>
</tr>
<tr>
<td>Bank 1</td>
<td>00 00 00 00 00</td>
<td>00 00 00 00 00</td>
<td>00 00 00 00 00</td>
</tr>
</tbody>
</table>

- **WR-A (bank 0)**: 00 00 00 00 11 00 00 11 00 00 00 00
- **WR-B (bank 1)**: 00 00 00 00 00 00 00 00 00 00

**Chip power constraint is violated!**

1. **Write-A** obeys both **DIMM** and **chip** power constraints. It can go to bank 0.

2. **Write-B** violates **chip** power constraint. It has to be stopped.
Performance with Both Power Constraints

[dim diagram showing speedup with different power constraints:)

- Ideal
- SPM
- DIMM only
- DIMM+chip

DIMM and chip power constraints hurt write throughput / performance a lot!
Simple Solutions?

- **Intra-line wear leveling** [ZHOU_ISCA’09]
  - Periodically shift N bytes for one line

- **Scheduling for power constraints**
  - Reorder writes

<table>
<thead>
<tr>
<th>WR-D</th>
<th>WR-C</th>
<th>WR-B</th>
<th>WR-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR-D</td>
<td>WR-C</td>
<td>WR-B</td>
<td>WR-A</td>
</tr>
</tbody>
</table>

Conflict

<table>
<thead>
<tr>
<th>WR-D</th>
<th>WR-C</th>
<th>WR-B</th>
<th>WR-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR-D</td>
<td>WR-C</td>
<td>WR-B</td>
<td>WR-A</td>
</tr>
</tbody>
</table>

No Conflict

<table>
<thead>
<tr>
<th>WR-D</th>
<th>WR-C</th>
<th>WR-B</th>
<th>WR-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR-D</td>
<td>WR-C</td>
<td>WR-B</td>
<td>WR-A</td>
</tr>
</tbody>
</table>

4x throughput

<table>
<thead>
<tr>
<th>WR-D</th>
<th>WR-C</th>
<th>WR-B</th>
<th>WR-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR-D</td>
<td>WR-C</td>
<td>WR-B</td>
<td>WR-A</td>
</tr>
</tbody>
</table>

No Conflict

<table>
<thead>
<tr>
<th>WR-D</th>
<th>WR-C</th>
<th>WR-B</th>
<th>WR-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR-D</td>
<td>WR-C</td>
<td>WR-B</td>
<td>WR-A</td>
</tr>
</tbody>
</table>

1.5x throughput

<table>
<thead>
<tr>
<th>WR-D</th>
<th>WR-C</th>
<th>WR-B</th>
<th>WR-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR-D</td>
<td>WR-C</td>
<td>WR-B</td>
<td>WR-A</td>
</tr>
</tbody>
</table>

Shift bytes
But They do NOT Help

- PWL: intra-level wear leveling without overhead — No effect
- Scheduling: Scheduling writes under both power constraints — No effect
- N x local: Enlarging local charge pump — 1.5xlocal, No effect

2 x local ≈ DIMM only case, but 100% overhead!
How to tackle *chip level power constraint*?
1. GCP balances power supply among chips
2. Power of GCP + LCPs ≤ DIMM level power constraint
3. Each sub-array is powered by either GCP or LCP, not both

4. Long wire → large resistance on wire[OH_JSSC’06] → low efficiency
5. Tradeoff between power utilization and efficiency
Global Charge Pump

- Ideal
- DIMM only
- GCP-NE
- DIMM+chip
- GCP-NE-0.7
- GCP-NE-0.5

GCP+100% eff. can relieve chip level P constraint!

GCP+50% eff. cancels the benefit of GCP!
Cell Mapping

Naïve Mapping (NE)  64B line = 256 cells

Vertical Interleaving (VIM)

Chip# = Cell# mod 8
Can We Do Even Better?

Braided Interleaving (BIM)

Chip# = (Cell# – Cell# / 16) mod 8

Chip

7  6  5  4  3  2  1  0
**Effectiveness of Cell Mapping**

- **Ideal**
- **GCP-NE**
- **GCP-VIM-0.7**
- **GCP-BIM-0.5**

- **DIMM only**
  - **GCP-NE-0.7**
  - **GCP-VIM-0.5**
- **DIMM+chip**
  - **GCP-NE-0.5**
  - **GCP-BIM-0.7**

**Speedup**

- GCP + V/BIM + 70% eff. ≈ GCP + 100% eff. !
- GCP + V/BIM + 50% eff. > GCP + 70% eff.
Can we utilize *DIMM level power budget* much *better*?
**Iteration Power Management**

Total 🍀: 80

A: 50 cell changes

B: 60 cell changes

<table>
<thead>
<tr>
<th> </th>
<th>Reset</th>
<th>Set</th>
<th>Set</th>
<th>Set</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>40</td>
<td>26</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>36</td>
<td>20</td>
<td>12</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

**power latency**

<table>
<thead>
<tr>
<th> </th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th> </th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

Ideally

Complete in 9 units of time

SPM on MLC

Complete in 16 units of time
Iteration Power Management

Total: 80

A: 50 cell changes
B: 60 cell changes

Proposed IPM

50

25

20

13

Complete in 12 units of time

Multi RESET (MR)

50

25

20

13

Complete in 10 units of time

Complete in 12 units of time

<table>
<thead>
<tr>
<th>Set</th>
<th>Rset</th>
<th>Power Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Complete in 12 units of time
Experimental Methodology

- **In-order 8-core 4GHz CMP processor**
  - L1: private i-32KB/d-32KB
  - L2: private 2MB, 64B line
  - L3: DRAM off-chip, **private** 32MB, 256B line

- **4GB 2-bit MLC PCM main memory**
  - One DIMM, single-rank, 8 banks
  - R/W queue 24 entries [HAY_MICRO’11]
  - Read first; schedule writes when NO read
  - Queue is full → **write burst** issuing all write until queue is empty
  - RESET: **500 cycles**, 300μA, **480μW**
  - SET: **1000 cycles**, 150μA, **90μW**
  - MLC **non-deterministic** write model [QURESHI_HPCA’10]

- **Benchmarks**
  - SPEC2006, BioBench, MiBench and STREAM
Effectiveness of IPM

- Ideal
- GCP
- DIMM only
- DIMM+chip
- GCP+IPM+MR

Normalized Write Throughput

- GCP
- GCP+IPM
- GCP+IPM+MR
- Ideal

Speedup

- 86%
- 76%

x2.4

Normalized Write Throughput

- ast_m
- bwa_m
- ibm_m
- les_m
- mcf_m
- xal_m
- mum_m
- tig_m
- qso_m
- cop_m
- mix_1
- mix_2
- mix_3
- gmean
Conclusions

- Increasing # of cores & Enlarging working set
  - Large & scalable main memory: MLC PCM

- Two power restrictions on MLC PCM
  - Limited DIMM level power constraint
  - Small chip level power constraint

- Global charge pump
  - Overcome *chip level power constraint*

- Iteration power management
  - Better utilize *DIMM level power budget*

- Our techniques achieve
  - Write throughput ↑ by *2.4*; Performance ↑ by *76%*
Fine-grained Power Budgeting to Improve Write Throughput of MLC PCM

1Lei Jiang, 2Youtao Zhang, 2Bruce R. Childers and 1Jun Yang

1Electrical and Computer Engineering Department
2Computer Science Department
University of Pittsburgh, Pittsburgh