RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization

Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, Todd C. Mowry

Carnegie Mellon University and Intel (Pittsburgh)
MICRO - 2013
Outline

Motivation
- Bottleneck
- Goal
- Observation
- Proposed Idea

DRAM
- Structure
- Operation

RowClone
- Fast Parallel Mode
  - Limitation
- Pipelined Serial Mode

Evaluation Results
Bottleneck: Memory Channel

Limited Bandwidth

High Energy

Core

Cache

MC

Channel

Memory
Goal: Reduce Memory Bandwidth Demand

Reduce unnecessary data movement
Motivation

Observation: Bulk Data Copy and Initialization

Bulk Data Copy

Bulk Data Initialization

Neither requires any computation!
Many Applications involve Bulk Copy and Initialization

- Forking
- Zero initialization (e.g., security)
- Checkpointing
- VM Cloning
- Deduplication
- Page Migration

Many more
Weakness of Current Process

High Energy
(3600 nJ to copy 4 KB)

High latency
(1046 ns to copy 4 KB)
Proposed Idea: Row Cloning $\rightarrow$ In-DRAM Copy
Inside DRAM
Digging In Little Deeper

- DRAM Chip
  - Shared Internal Bus
  - Bank
    - Subarray
      - Bank I/O
    - Bank I/O
      - Row of DRAM cells
    - Row buffer
      - Memory Channel

- Bank
  - Row Decoder
    - row
      - row buffer
- Subarray
  - wordline
    - cell
      - capacitor
      - access transistor
    - bitline
      - sense amplifier
DRAM Operation

This is where RowClone comes into the picture
Row Clone: Fast Parallel Mode (FPM)

- DRAM can transfer an entire row of data from the DRAM cells to the corresponding row buffer.
- FPM first copies the data from the source row to the row buffer and then copies the data from the row buffer to the destination row.
**Fast Parallel Mode - Limitations**

It only works if the source and destination rows to be within the same sub-array

1. What about transfer across different banks?
2. What about transfer across different sub-arrays within a bank?
Row Clone: *Pipelined Serial Mode (PSM)*

- Shared internal bus across all banks
- Provides an alternate mechanism to copy data across banks
- PSM allows to transfer data without crossing the Chip I/O interface
Row Clone: Pipelined Serial Mode (PSM)

Inter-Bank Transfer:
1) Activates both source and destination rows in the corresponding banks
2) Puts the source bank in Read mode and destination bank in Write mode to transfer data

Intra-Bank Transfer:
1) Use PSM to transfer data from source to a temporary row in a different bank
2) Again use PSM to transfer data from temporary location to the destination
Miscellaneous

ISA support:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operands</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>memcpy</td>
<td>src, dst, size</td>
<td>Copy size bytes from src to dst</td>
</tr>
<tr>
<td>meminit</td>
<td>dst, size, val</td>
<td>Set size bytes to val at dst</td>
</tr>
</tbody>
</table>

Row Clone - ZI:

- RowClone-Zero-Insert (RowClone-ZI) inserts a zero cache line into the processor cache corresponding to the page that is zeroed out.
- By doing this, RowClone-ZI avoids the cache misses during both zeroing operation and when the application accesses the cache lines of the zeroed page.
Raw Latency Improvement

![Diagram showing latency improvement comparison between baseline, FPM, and PSM (Inter-bank) scenarios. The diagram illustrates the sequence of operations (ACT s, ACT d, R, W, Tr, PRE) and the corresponding latencies (90ns, 540ns) for each scenario.]

ACT s — ACTIVATE source, ACT d — ACTIVATE destination
R — READ, W — WRITE, Tr — TRANSFER, PRE — PRECHARGE
## Raw Energy Improvement

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Copy</th>
<th>Absolute</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Latency (ns)</td>
<td>Memory Energy (µJ)</td>
<td>Latency</td>
</tr>
<tr>
<td>Baseline</td>
<td>1046</td>
<td>3.6</td>
<td>1.00x</td>
</tr>
<tr>
<td>FPM</td>
<td>90</td>
<td>0.04</td>
<td><strong>11.62x</strong></td>
</tr>
<tr>
<td>Inter-Bank - PSM</td>
<td>540</td>
<td>1.1</td>
<td>1.93x</td>
</tr>
<tr>
<td>Intra-Bank - PSM</td>
<td>1050</td>
<td>2.5</td>
<td>0.99x</td>
</tr>
<tr>
<td>Baseline</td>
<td>546</td>
<td>2.0</td>
<td>1.00x</td>
</tr>
<tr>
<td>FPM</td>
<td>90</td>
<td>0.05</td>
<td><strong>6.06x</strong></td>
</tr>
</tbody>
</table>
Traffic Distribution – Copy and Initialization

![Bar chart showing traffic distribution for different applications.]

- **bootup**: Read > Copy > Write > Initialization
- **compile**: Read > Write > Copy > Initialization
- **forkbench**: Read > Copy > Write > Initialization
- **mcached**: Read > Copy > Write > Initialization
- **mysql**: Read > Write > Copy > Initialization
- **shell**: Read > Copy > Write > Initialization

- ✔️ Copy is dominant in bootup, forkbench, and shell.
- ❌ Initialization is dominant in compile and mcached.
- ✔️ Copy and Initialization are nearly equal in mysql.
Performance Comparison – Single-core Environment

The diagram shows a comparison of instructions per cycle across different workloads: bootup, compile, forkbench, mcached, mysql, and shell. The bars represent Baseline, RowClone, and RowClone-ZI. The metrics are in percentage differences compared to the Baseline.

- Bootup: Baseline 15%, RowClone 9%, RowClone-ZI 4%
- Compile: Baseline 66%, RowClone 40%, RowClone-ZI 40%
- Forkbench, Mcached, Mysql, Shell: Baseline 4%, RowClone 40%, RowClone-ZI 40%

The green checks indicate positive performance improvements, while the red Xs indicate negative performance impacts.
## Energy and Bandwidth Improvement

<table>
<thead>
<tr>
<th>Application</th>
<th>Energy Reduction</th>
<th>Bandwidth Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RowClone</td>
<td>+ZI</td>
</tr>
<tr>
<td>bootup</td>
<td>39%</td>
<td>40%</td>
</tr>
<tr>
<td>compile</td>
<td>-2%</td>
<td>32%</td>
</tr>
<tr>
<td>forkbench</td>
<td>69%</td>
<td>69%</td>
</tr>
<tr>
<td>mcached</td>
<td>0%</td>
<td>15%</td>
</tr>
<tr>
<td>mysql</td>
<td>-1%</td>
<td>17%</td>
</tr>
<tr>
<td>shell</td>
<td>68%</td>
<td>67%</td>
</tr>
</tbody>
</table>
Multi-core Performance

- RowClone improves performance for most workloads
- RowClone-ZI improves workload for all workloads
Thank you