CSC 2224: Parallel Computer Architecture and Programming
Advanced Memory

Prof. Gennady Pekhimenko
University of Toronto
Fall 2019

The content of this lecture is adapted from the slides of Vivek Seshadri, Yoongu Kim, and lectures of Onur Mutlu @ ETH and CMU
Review #5

Flipping Bits in Memory Without Accessing Them
Yoongu Kim et al., ISCA 2014
Review: Memory Latency Lags Behind

Memory latency remains almost constant
We Need A Paradigm Shift To …

- Enable computation with *minimal data movement*

- Compute where it makes sense *(where data resides)*

- Make computing architectures more *data-centric*
Many questions ... How do we design the:
- compute-capable memory & controllers?
- processor chip?
- software and hardware interfaces?
- system software and languages?
- algorithms?
Why In-Memory Computation Today?

Pull from Systems and Applications

- Data access is a major system and application bottleneck
- Systems are energy limited
- Data movement much more energy-hungry than computation
Two Approaches to In-Memory Processing

1. Minimally change DRAM to enable simple yet powerful computation primitives
   - **RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data** (Seshadri et al., MICRO 2013)
   - **Fast Bulk Bitwise AND and OR in DRAM** (Seshadri et al., IEEE CAL 2015)
   - **Gather-Scatter DRAM: In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses** (Seshadri et al., MICRO 2015)

2. Exploit the control logic in 3D-stacked memory to enable more comprehensive computation near memory
   - **PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture** (Ahn et al., ISCA 2015)
   - **A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing** (Ahn et al., ISCA 2015)
   - **Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation** (Hsieh et al., ICCD 2016)
Approach 1: Minimally Changing DRAM

- DRAM has great capability to perform **bulk data movement and computation** internally with small changes
  - Can exploit internal bandwidth to move data
  - Can exploit analog computation capability
  - ...

Examples: RowClone, In-DRAM AND/OR, Gather/Scatter DRAM

- **RowClone**: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data *(Seshadri et al., MICRO 2013)*
- **Fast Bulk Bitwise AND and OR in DRAM** *(Seshadri et al., IEEE CAL 2015)*
- **Gather-Scatter DRAM**: In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses *(Seshadri et al., MICRO 2015)*
Starting Simple: Data Copy and Initialization

**Bulk Data Copy**

**Bulk Data Initialization**
Bulk Data Copy and Initialization

The Impact of Architectural Trends on Operating System Performance
Mendel Rosenblum, Edouard Bugnion, Stephen Alan Herrod, Emmett Witchel, and Anoop Gupta

Hardware Support for Bulk Data Movement in Server Platforms
Li Zhao¹, Ravi Iyer² Srihari Makineni³, Laxmi Bhuyan⁴ and Don Newell⁵
¹Department of Computer Science and Engineering, University of California, Riverside, CA 92521
Email: {zhaol, bhuyan}@cs.ucr.edu
²Communications Technology Lab, Intel Corporation

Architecture Support for Improving Bulk Memory Copying and Initialization Performance
Xiaowei Jiang, Yan Solihin
Dept. of Electrical and Computer Engineering
North Carolina State University
Raleigh, USA

Li Zhao, Ravishankar Iyer
Intel Labs
Intel Corporation
Hillsboro, USA
Forking

Zero initialization (e.g., security)

Checkpointing

Many more

Memmove & memcpy: 5% cycles in Google’s datacenter [Kanev+ ISCA’15]
Today’s Systems: Bulk Data Copy

1) High latency
2) High bandwidth utilization
3) Cache pollution
4) Unwanted data movement

1046ns, 3.6uJ (for 4KB page copy via DMA)
Future Systems: In-Memory Copy

1) Low latency
2) Low bandwidth utilization
3) No cache pollution
4) No unwanted data movement

1046ns, 3.6uJ
90ns, 0.04uJ
RowClone: In-DRAM Row Copy

Idea: Two consecutive ACTivates

Negligible HW cost

Step 1: Activate row A
Step 2: Activate row B

DRAM subarray

Row Buffer (4 Kbytes)

Transfer row

4 Kbytes

Data Bus

8 bits
RowClone: Intra-Subarray

Data gets copied

Data gets copied

Sense Amplifier (Row Buffer)

Amplify the difference

\[ V_{DD}/2 \]

\[ V_{DD}/2 + \]

\[ V_{DD}/2 \]

\[ \delta \]

\[ V_{DD}/2 + \]

\[ \delta \]

\[ V_{DD}/2 \]

\[ V_{DD}/2 \]

\[ V_{DD}/2 \]

\[ V_{DD}/2 \]
RowClone: Intra-Subarray (II)

1. **Activate** src row (copy data from src to row buffer)

2. **Activate** dst row (disconnect src from row buffer, connect dst – copy data from row buffer to dst)
RowClone: Inter-Bank

Overlap the latency of the read and the write

1.9X latency reduction, 3.2X energy reduction
Generalized RowClone

0.01% area cost

Inter Subarray Copy
(Use Inter-Bank Copy Twice)

Inter Bank Copy
(Pipelined Internal RD/WR)

Intra Subarray Copy
(2 ACTs)
RowClone: Fast Row Initialization

Fix a row at Zero
(0.5% loss in capacity)
RowClone: Bulk Initialization

- Initialization with arbitrary data
  - Initialize one row
  - Copy the data to other rows

- Zero initialization (most common)
  - Reserve a row in each subarray (always zero)
  - Copy data from reserved row (FPM mode)
  - 6.0x lower latency, 41.5x lower DRAM energy
  - 0.2% loss in capacity
RowClone: Latency & Energy Benefits

Very low cost: 0.01% increase in die area
Copy and Initialization in Workloads

![Graph showing fractions of memory traffic for various workloads: bootup, compile, forkbench, mcached, mysql, shell. The graph indicates the fractions of read, write, copy, and zero operations.]
RowClone: Application Performance

![Bar Chart: IPC Improvement and Energy Reduction](chart.png)
End-to-End System Design

- Application
- Operating System
- ISA
- Microarchitecture
- DRAM (RowClone)

How to communicate occurrences of bulk copy_INITIALIZATION across layers?

How to ensure cache coherence?

How to maximize latency and energy savings?

How to handle data reuse?
Ambit
In-Memory Accelerator for Bulk Bitwise Operations
Using Commodity DRAM Technology

Vivek Seshadri

Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, Todd C. Mowry

SAFARI Carnegie Mellon Intel
Microsoft ETH Zürich
Executive Summary

• **Problem: Bulk bitwise operations**
  – present in many applications, e.g., databases, search filters
  – existing systems are memory bandwidth limited

• **Our Proposal: Ambit**
  – perform bulk bitwise operations *completely inside DRAM*
  – **bulk bitwise AND/OR**: simultaneous activation of three rows
  – **bulk bitwise NOT**: inverters already in sense amplifiers
  – less than 1% area overhead over existing DRAM chips

• **Results compared to state-of-the-art baseline**
  – average across seven bulk bitwise operations
    • 32X performance improvement, 35X energy reduction
  – 3X-7X performance for real-world data-intensive applications
Bitmap indices (database indexing)

Set operations

Encryption algorithms

BitWeaving (database queries)

BitFunnel (web search)

DNA sequence mapping

Bulk Bitwise Operations
Today, DRAM is just a storage device!

Throughput of bulk bitwise operations limited by available memory bandwidth
Our Approach

Use analog operation of DRAM to perform bitwise operations completely inside memory!
Inside a DRAM Chip

2D Array of DRAM Cells

Sense amplifiers

8KB
DRAM Cell Operation

- **Wordline**
- **Capacitor**
- **Access Transistor**
- **Sense Amp**
- **Enable**
- **Bitline**
DRAM Cell Operation

1. Raise wordline
2. Capacitor gains charge
3. Cell loses charge to bitline
4. Cell regains charge
5. Sense Amp deviation in bitline voltage
6. Connects cell to bitline
7. Enable sense amp
8. Enable wordline
9. Bitline
Triple-Row Activation: Majority Function

activate all three rows

enable sense amp
Bitwise AND/OR Using Triple-Row Activation
Bitwise AND/OR Using Triple-Row Activation

Output = AB + BC + CA

= C (A OR B) + ~C (A AND B)

Control the value of C to perform bitwise OR or bitwise AND of A and B

38X improvement in raw throughput
44X reduction in energy consumption for bulk bitwise AND/OR operations
Bulk Bitwise AND/OR in DRAM

Statically reserve three designated rows t1, t2, and t3

Result = row A

1. Copy data of row A to row t1
2. Copy data of row B to row t2
3. Initialize data of row t3 to 0/1
4. Activate rows t1/t2/t3 simultaneously
5. Copy data of row t1/t2/t3 to Result row

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

Vivek Seshadri vseshadri@cs.cmu.edu  Yoongu Kim yoongukim@cmu.edu  Chris Fallin* cfallin@c1f.net  Donghyuk Lee donghyuk1@cmu.edu
Rachata Ausavarungnirun rachata@cmu.edu  Gennady Pekhimenko gpekhime@cs.cmu.edu  Yixin Luo yixinluo@andrew.cmu.edu
Onur Mutlu onur@cmu.edu  Phillip B. Gibbons† philip.b.gibbons@intel.com  Michael A. Kozuch† michael.a.kozuch@intel.com  Todd C. Mowry tcm@cs.cmu.edu
Carnegie Mellon University  †Intel Pittsburgh
Bulk Bitwise AND/OR in DRAM

Statically reserve three designated rows $t_1$, $t_2$, and $t_3$

Result = row A

1. Copy data of row A to row $t_1$
2. Copy RowClone data of row B to row $t_2$
3. Initialize RowClone data of row $t_3$ to 0/1
4. Activate rows $t_1$, $t_2$, and $t_3$ simultaneously
5. Copy data of row $t_1$, $t_2$, and $t_3$ to Result row

Use RowClone to perform copy and initialization operations completely in DRAM!
Can we copy the negated value from bitline to a DRAM cell?
Negation Using the Sense Amplifier

Regular wordline

Negation wordline

enabl...
Negation Using the Sense Amplifier

activate source

sorce

activate negation wordline

enable sense amp

Sens e Amp

-bitline

0/2 VDD
Ambit vs. DDR3: Performance and Energy

- Performance Improvement
- Energy Reduction

<table>
<thead>
<tr>
<th>Operation</th>
<th>Performance Improvement</th>
<th>Energy Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>60</td>
<td>32</td>
</tr>
<tr>
<td>and/or</td>
<td>45</td>
<td>X</td>
</tr>
<tr>
<td>nand/nor</td>
<td>30</td>
<td>X</td>
</tr>
<tr>
<td>xor/xnor</td>
<td>20</td>
<td>X</td>
</tr>
<tr>
<td>mean</td>
<td>35</td>
<td></td>
</tr>
</tbody>
</table>

32 35 X X
Integrating Ambit with the System

1. PCIe device
   - Similar to other accelerators (e.g., GPU)

2. System memory bus
   - Ambit uses the same DRAM command/address interface

Pros and cons discussed in paper (Section 5.4)
Real-world Applications

• **Methodology** (Gem5 simulator)
  – Processor: x86, 4 GHz, out-of-order, 64-entry instruction queue
  – L1 cache: 32 KB D-cache and 32 KB I-cache, LRU policy
  – L2 cache: 2 MB, LRU policy
  – Memory controller: FR-FCFS, 8 KB row size
  – Main memory: DDR4-2400, 1 channel, 1 rank, 8 bank

• **Workloads**
  – Database bitmap indices
  – BitWeaving – column scans using bulk bitwise operations
  – Set operations – comparing bitvectors with red-black trees
Bitmap Indices: Performance

Consistent reduction in execution time. 6X on average.
Speedup offered by Ambit for BitWeaving

```
select count(*) where c1 < field < c2
```

Number of rows in the database table

![Graph showing speedup for different number of bits for each column value. The y-axis represents the speedup offered by Ambit, and the x-axis shows the number of bits for each column value. The graph includes data points for 1m, 4m, and 8m rows, with 12 as a notable value.]
ROW HAMMER
DRAM CHIP VICTIM AGGRESS VICTIM HIGH VOLTAGE WORDLINE READ DATA FROM HERE, GET ERRORS OVER THERE
GOOGLE’S EXPLOIT

Project Zero

News and updates from the Project Zero team at Google

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

“We learned about rowhammer from Yoongu Kim et al.”

http://googleprojectzero.blogspot.com
GOOGLE’S EXPLOIT

OUR PROOF-OF-CONCEPT

PROPOSED SOLUTIONS

EMPIRICAL ANALYSIS
REAL SYSTEM

MANY READS TO SAME ADDRESS ≠ OPEN/CLOSE SAME ROW

1. CACHE HITS
2. ROW HITS
x86 CPU

LOOP:

mov (X), %reg
mov (Y), %reg
clflush (X)
clflush (Y)
jmp LOOP

DRAM

http://www.github.com/CMU-SAFARI/rowhammer

MANY ERRORS!
WHY DO THE ERRORS OCCUR?
DRAM CELLS ARE LEAKY

CHARG

0ms  TIME  64ms

NORMAL CELL

‘0’

‘1’

REFRE  SH
DRAM CELLS ARE LEAKY

AGGRESS

VICTIM CELL

CHARG

0ms TIME 64ms

OR

REFRESH

‘1’

‘0’
ROOT CAUSE?

COUPLING
- Electromagnetic
- Tunneling

ACCELERATES CHARGE LOSS
AS DRAM SCALES

• CELLS BECOME SMALLER
  Less tolerance to coupling effects

• CELLS BECOME PLACED CLOSER
  Stronger coupling effects
1. **ERRORS ARE RECENT**
   Not found in pre-2010 chips

2. **ERRORS ARE WIDESPREAD**
   >80% of chips have errors
   Up to one error per ~1K cells
MOST MODULES AT RISK

A VENDOR

86% (37/43)

B VENDOR

83% (45/54)

C VENDOR

88% (28/32)
ERRORS PER $10^9$ CELLS

MANUFACTURE DATE

MODULES: ● A ■ B ◆ C
DISTURBING FACTS

• AFFECTS ALL VENDORS
  Not an isolated incident
  Deeper issue in DRAM scaling

• UNADDRESSED FOR YEARS
HOW TO PREVENT COUPLING ERRORS?

Previous Approaches
1. Make Better Chips: Expensive
2. Rigorous Testing: Takes Too Long
FASTER ACCESS

FREQUENT REFRESH
ONE MODULE: □ A □ B ◇ C

TOTAL ERRORS

10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
0

5ns

Faster → Slower

ACCESS INTERVAL (ns)

500ns

64ms
500ns = 128K

S

55ns

100
200
300
400
500

ACCESS INTERVAL (ns)
### ONE MODULE:

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

#### TOTAL ERROR S

<table>
<thead>
<tr>
<th>REFRESH INTERVAL (ms)</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>11ms</td>
<td>55ns</td>
</tr>
<tr>
<td>64ms</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\frac{11\text{ms}}{55\text{ns}} = 200K
\]

**Often** ← **Seldom**
TWO NAIVE SOLUTIONS

1. LIMIT ACCESSES TO ROW
   Access Interval > 500ns

2. REFRESH ALL ROWS OFTEN
   Refresh Interval < 11ms

LARGE OVERHEAD:
PERF, ENERGY, COMPLEXITY
OUR SOLUTION: PARR

After closing any row

99.9%

Do nothing

0.1%

Refresh (=Open)
adjacent rows

Probabilistic Adjacent Row Refresh

Do nothing

Refresh (=Open) adjacent rows

68
PARR: CHANCE OF ERROR

• NO REFRESHES IN $N$ TRIALS
  Probability: $0.999^N$

• $N=128K$ FOR ERROR (64ms)
  Probability: $0.999^{128K} = 10^{-56}$
STRONG RELIABILITY
LOW PERFORMANCE
NO STORAGE OVERHEAD
9.4×10^{-14}
Errors/Year
0.20%
Slowdown
0 Bytes
RELATED WORK

• Security Exploit (Seaborn@Google 2015)

• Industry Analysis (Kang@SK Hynix 2014)
  “... will be [more] severe as technology shrinks down.”

• Targeted Row Refresh (JEDEC 2014)

• DRAM Testing (e.g., Van de Goor+ 1999)
Emerging Memory Technologies
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
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 $R$
Promising 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/RRAM/ReRAM**
  - 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

![Diagram showing the concepts of SET (cryst) and RESET (amorph) with resistance values: 10^3-10^4 \( \Omega \) for SET and 10^6-10^7 \( \Omega \) for RESET.]

**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 85C

- **No refresh needed, low idle power**
PCM Resistance $\rightarrow$ Value

Cell value:

1 0

Cell resistance
Multi-Level Cell PCM

- Multi-level cell: more than 1 bit per cell
  - Further increases density by 2 to 4x [Lee+, ISCA'09]

- But MLC-PCM also has drawbacks
  - Higher latency and energy than single-level cell PCM
MLC-PCM Resistance $\rightarrow$ Value

Cell value:

- Bit 1
- Bit 0

Cell resistance
MLC-PCM Resistance → Value

*Less margin between values*

→ need more precise sensing/modification of cell contents
→ higher latency/energy (~2x for reads and 4x for writes)
Phase Change Memory Properties

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

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 650°C
  - Contacts degrade from thermal expansion/contraction
  - $10^8$ writes per cell
  - $10^{-8}$x DRAM, $10^3$x NAND Flash

- **Cell Size**
  - 9-12F² 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 (capacity and cost)
  - Non volatile Persistent
  - 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)
  - Reliability issues (resistance drift)

- Challenges in enabling PCM as DRAM replacement/helper:
  - Mitigate PCM shortcomings
  - Find the right way to place PCM in the system
PCM-based Main Memory (I)

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

  - **Hybrid PCM+DRAM** [Qureshi+ ISCA’09, Dhiman+ DAC’09]:
    - How to partition/migrate data between PCM and DRAM
PCM-based Main Memory (II)

- 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
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
- $9 - 12F^2$ using BJT
- $1.5 \times$ DRAM

### Latency
- 50ns Rd, 150ns Wr
- $4 \times$, $12 \times$ DRAM

### Endurance
- $1E+08$ writes
- $1E-08 \times$ DRAM

### Energy
- $40\mu A$ Rd, $150\mu A$ Wr
- $2 \times$, $43 \times$ 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

![Diagram of DRAM and PCM data arrays with sense amplifiers and latches](image-url)
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?
PCM As Main Memory

- Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative"
Review #5

**Flipping Bits in Memory Without Accessing Them**

Yoongu Kim et al., *ISCA 2014*