Simultaneous Multi-Layer Access
Improving 3D-Stacked Memory Bandwidth at Low Cost

Donghyuk Lee, Saugata Ghose, Gennady Pekhimenko, Samira Khan, Onur Mutlu
Carnegie Mellon University

HiPEAC 2016
Executive Summary

- In 3D-stacked DRAM, we want to leverage high bandwidth
  - TSVs provide a huge opportunity
  - In-DRAM bandwidth does not match

- Problem: Global structures in layer very expensive — can’t replicate

- Our Solution: Simultaneous Multi-Layer Access (SMLA)
  - Use multiple layers at once to overcome bottleneck
  - Requires smart multiplexing
  - Two alternatives: Dedicated-IO, Cascaded-IO

- Evaluation vs. Wide I/O (16 core):
  55% speedup, 18% DRAM energy reduction, negligible area overhead

**SAFARI**
Outline

Limited Bandwidth in 3D DRAM

Simultaneous Multi-Layer Access

1. Dedicated-IO
2. Cascaded-IO

Performance Evaluation
Connecting Layers in 3D-Stacked DRAM

DRAM layer

cell array

peripheral logic
Connecting Layers in 3D-Stacked DRAM

3D-stacked DRAM: $512 - 4K$ TSVs
Traditional 2D DRAM: 64-bit bus

8x – 64x increase – Can we exploit this?
How Much Can Bandwidth Increase?

If TSVs provide 16x bus width vs. 2D, do we get 16x bandwidth from the DRAM?
How Much Can Bandwidth Increase?

Can each layer deliver 16x the data?

If TSVs provide 16x bus width vs. 2D, do we get 16x bandwidth from the DRAM?
How Much Can Bandwidth Increase?

- cell array
- peripheral logic
How Much Can Bandwidth Increase?
How Much Can Bandwidth Increase?

- Global bitlines ➔ X16?
- Global sense amplifiers ➔ X16?
- Data path ➔ X16?

- TSVs ➔ X16

SAFARI
How Much Can Bandwidth Increase?

Global sense amplifiers and global bitlines are costly
→ Cannot provide 16x in-DRAM BW → Bottleneck
Problem

**Limited in-DRAM bandwidth**, leading to high costs for high-bandwidth 3D-stacked DRAM

Our Goal

Design a new 3D-stacked DRAM that supplies **high DRAM bandwidth** at **low cost**

Our Approach

**Simultaneous Multi-Layer Access (SMLA)**
Outline

Limited Bandwidth in 3D DRAM

Simultaneous Multi-Layer Access

1. Dedicated-IO

2. Cascaded-IO

Performance Evaluation
Simultaneous Multi-Layer Access

Idle

Idle

Accessed layer

Idle
Simultaneous Multi-Layer Access

Exploit in-DRAM bandwidth across **idle layers** by **accessing multiple layers simultaneously**
Simultaneous Multi-Layer Access

CHALLENGE: How to avoid TSV channel conflicts?

→ Space Multiplexing & Time Multiplexing
Outline

Limited Bandwidth in 3D DRAM

Simultaneous Multi-Layer Access

1. Dedicated-IO
2. Cascaded-IO

Performance Evaluation
Dedicated-IO: \textbf{Space Multiplexing}

Dedicate a subset of TSVs for each layer \(\rightarrow\)
Transfer each layer’s data with \textit{narrower & faster IO}
Dedicated-IO: Two Problems

1. Differences in layers → Fabrication difficulties
2. Weaker power network at upper layers
Outline

Limited Bandwidth in 3D DRAM

Simultaneous Multi-Layer Access

1. Dedicated-IO

2. Cascaded-IO

Performance Evaluation
Cascaded-IO: **Time Multiplexing**

Segment TSVs to move data through 3D stack *one layer at a time* at **4F frequency**
Cascaded-IO: **Time Multiplexing**

Segment TSVs to move data through 3D stack *one layer at a time* *at 4F frequency*
Cascaded-IO: **Time Multiplexing**

Segment TSVs to move data through 3D stack *one layer at a time* **at 4F frequency**
Cascaded-IO: Time Multiplexing

Segment TSVs to move data through 3D stack one layer at a time at 4F frequency
Cascaded-IO: Time Multiplexing

Throughput

Layer 3

Layer 2

Layer 1

Layer 0
Cascaded-IO: Time Multiplexing

Different throughput requirements for each layer

→ Reduce IO frequency in upper layers
Cascaded-IO Data Multiplexer

clock → cell array & peripheral logic → mux → latch → data

cell array & peripheral logic

clock
data
Cascaded-IO Clock Propagator

counter\[en.\] cell array & peripheral logic

clock

latch

mux\[ctrl.\]
data
Outline

Limited Bandwidth in 3D DRAM

Simultaneous Multi-Layer Access

1. Dedicated-IO
2. Cascaded-IO

Performance Evaluation
Evaluation Methodology

- **CPU**: 4-16 cores
  - **Simulator**: Instruction-trace-based x86 simulator
  - 3-wide issue, 512kB cache slice per core

- **Memory**: 4 DRAM layers
  - **Simulator**: Ramulator [Kim+ CAL 2015]
    [https://github.com/CMU-SAFARI/ramulator](https://github.com/CMU-SAFARI/ramulator)
  - 64-entry read/write queue, FR-FCFS scheduler
  - **Energy Model**: built from other high-frequency DRAMs
  - **Baseline**: Wide I/O 3D-stacked DRAM, 200 MHz

- **Workloads**
  - 16 multiprogrammed workloads for each core count
  - Randomly selected from TPC, STREAM, SPEC CPU2000
SMLA Improves Performance

Over Wide I/O 3D-stacked DRAM

Performance Improvement (%) over Wide I/O 3D-stacked DRAM

- Dedicated-IO (4X)
- Cascaded-IO (4X)

55%
SMLA Reduces Memory Energy

over Wide I/O 3D-stacked DRAM

18%
Conclusion

• Through-silicon vias (TSVs) offer high bandwidth in 3D-stacked DRAM

• Problem
  – In-DRAM bandwidth limits available IO bandwidth in 3D-stacked DRAM

• Our Solution: **Simultaneous Multi-Layer Access (SMLA)**
  – Exploit in-DRAM bandwidth available in other layers by accessing **multiple layers** simultaneously
  – Dedicated-IO
    • Divide TSVs across layers, clock TSVs at higher frequency
  – Cascaded-IO
    • Pipelined data transfer through layers
    • Reduce upper layer frequency for lower energy

• Evaluation
  – Significant **performance improvement** with **lower DRAM energy consumption** (55%/18% for 16-core) at negligible DRAM area cost
Simultaneous Multi-Layer Access
Improving 3D-Stacked Memory Bandwidth at Low Cost

Donghyuk Lee, Saugata Ghose, Gennady Pekhimenko, Samira Khan, Onur Mutlu
Carnegie Mellon University

HiPEAC 2016

We will release the SMLA source code by February.
SMLA vs. HBM

• HBM (High-Bandwidth Memory)
  – Used in GPUs today
  – Much wider bus than other 3D-stacked DRAMs
    (4096 TSVs vs. 1024 for Wide I/O)
  – Double the global bitline count of Wide I/O
    – Simultaneously access two bank groups: double the global sense amplifiers

• HBM adds more bandwidth by scaling up resources (i.e., at higher cost)
  – Each layer now gets two dedicated 128-bit channels
  – Similar to Dedicated-IO, but more TSVs/global bitlines instead of faster TSVs

• SMLA delivers higher bandwidth than HBM w/o extra bitlines/TSVs (at lower cost)
  – Dedicated-IO: performance, energy efficiency similar to HBM; cost is lower
  – Cascaded-IO: higher performance, energy efficiency; lower cost than HBM
Scaling SMLA to Eight Layers

• Dedicated-IO
  1. Double the TSVs
     • Same # of TSVs per layer
     • 2x total DRAM bandwidth of 4-layer SMLA
  2. Pairs of layers share a single set of TSVs
     • Creates a layer group
     • Same total DRAM bandwidth as 4-layer SMLA

• Cascaded-IO
  1. Double the TSVs – 2 groups, 4 layers each
  2. Two groups, but sharing the TSVs
     • Groups necessary to stay within burst length, limit multi-layer traversal time
     • Same total DRAM bandwidth as 4-layer SMLA
Conventional DRAM

IO Bandwidth = Bus Width X Wire Data Rate

Limited package pins  clock frequency
Cascaded-IO Operation

- Cell array & peripheral logic
- Mux
- Counter
-Latch
- Clock
- Data

F: Clock
B: Data
2F: Clock
2B: Data
Cascaded-IO Operation

Diagram showing the flow of data through a cascaded-IO operation with cell array & peripheral logic, counter, latch, mux, and control signals.
Cascaded-IO Operation
Estimated Energy Consumption

---

Standby currents are proportional to data channel frequency → Cascaded-IO reduces standby current
Cascaded-IO provides **standby power reduction** due to **reduced upper layer frequency**

**Standby Power Consumption**

![Bar chart showing standby power consumption across different layers and IO types. The chart highlights a 14% reduction in standby power for Cascaded-IO compared to other IO types.]
Thermal Analysis

<table>
<thead>
<tr>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Layer 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wide I/O (200MHz)</td>
<td>Wide I/O (1600MHz)</td>
<td>Dedicated-IO (1600MHz)</td>
<td>Cascaded-IO (1600MHz)</td>
</tr>
</tbody>
</table>

Cascaded-IO reduces operating temperature due to reduced upper layer frequency