Tiered-Latency DRAM: A Low Latency and A Low Cost DRAM Architecture

Donghyuk Lee, Yoongu Kim, Vivek Seshadri, Jamie Liu, Lavanya Subramanian, Onur Mutlu

Carnegie Mellon University

HPCA - 2013
Executive Summary

- **Problem**: DRAM latency is a critical performance bottleneck
- **Goal**: Reduce DRAM latency with low area cost
- **Observation**: Long bitlines in DRAM are the dominant source of DRAM latency
- **Key Idea**: Divide long bitlines into two shorter segments
  – Fast and slow segments
- **Tiered-latency DRAM**: Enables latency heterogeneity in DRAM to improve performance and reduce power consumption
- **Results**: Significant performance improvement (>12%) and power reduction (>23%) at low area cost (3%)
Historical DRAM Trend

DRAM latency continues to be a critical bottleneck
What Causes the Long Latency?

DRAM Latency = Subarray Latency + I/O Latency

Dominant
Why is the Subarray So Slow?

**Long Bitline:**
- Amortize sense amplifier → Small area
- Large bitline cap. → High latency

**Cell:**
- Wordline
- Access transistor
- Capacitor
- Bitline

**Subarray:**
- Bitline: 512 cells
- Extremely large sense amplifier
  (≈100X the cell size)
Trade-Off: Area (Die Size) vs. Latency

Long Bitline

Short Bitline

Faster

Smaller

Trade-Off: Area vs. Latency
Trade-Off: Area (Die Size) vs. Latency

- Cheaper
- Normalized DRAM Area
- Latency (ns)
- GOAL

- Fancy DRAM Short Bitline
- Commodity DRAM Long Bitline
- 512 cells/bitline
Approximating the Best of Both Worlds

- **Long Bitline**: Small Area, High Latency
- **Proposal**: Add Isolation Transistors
- **Short Bitline**: Large Area, Low Latency

Need Isolation

Add Isolation Transistors
Tiered-Latency DRAM

- Divide a bitline into two segments with an isolation transistor
Near Segment Access

- Turn **off** the isolation transistor

- Reduced bitline length
- Reduced bitline capacitance
  - **→** Low latency & low power

Isolation Transistor (off)

Near Segment

Sense Amplifier
Far Segment Access

- Turn *on* the isolation transistor

Long bitline length
Large bitline capacitance
Additional resistance of isolation transistor

⇒ High latency & high power

![Diagram](image-url)
Latency, Power, and Area Evaluation

- DRAM Latency (tRC)
  - Commodity DRAM
  - Near TL-DRAM
  - Far TL-DRAM
  - (52.5 ns)
  - Latency: 
    - Commodity: 100%
    - Near: 56%
    - Far: 23%

- DRAM Power
  - Commodity DRAM
  - Near TL-DRAM
  - Far TL-DRAM
  - Power:
    - Commodity: 100%
    - Near: 51%
    - Far: 49%

Area Overhead: ~3% (mainly due to the isolation transistors)
Latency, Power, and Area Evaluation

- Longer near segment length leads to higher near segment latency
- Far segment latency is higher than commodity DRAM latency
Part 02: Leveraging Tiered-Latency DRAM
Challenge 1: How to efficiently migrate a row between segments?

Challenge 2: How to efficiently manage the cache?
How to efficiently migrate a row between segments?

- **Goal**: Migrate source row into destination row
- **Naïve way**: Memory controller reads the source row byte by byte and writes to destination row byte by byte → High latency
Proposed way of Inter-Segment Migration

- Source and destination cells *share bitlines*
- Transfer data from source to destination across *shared bitlines* concurrently

**Migration is overlapped with source row access**

**Additional ~4ns over row access latency**

**Step 1:** Activate source row

**Step 2:** Activate destination row to connect cell and bitline
How to efficiently manage the cache?

1. **SC** (Simple Caching)
   - Classic LRU cache

2. **WMC** (Wait-Minimized Caching)
   - Identify and cache only *wait-inducing rows*
   - Benefit: Reduced wait

3. **BBC** (Benefit-Based Caching)
   - Benefit: Reduced reuse latency & reduced wait

Another benefit of caching:

- **SC** (Simple Caching)
  - Classic LRU cache

- **WMC** (Wait-Minimized Caching)
  - Identify and cache only *wait-inducing rows*
  - Benefit: Reduced wait

- **BBC** (Benefit-Based Caching)
  - BBC ≈ SC + WMC
  - Benefit: Reduced reuse latency & reduced wait
Evaluation Setup

• **System configuration**
  – CPU: 5.3GHz (??)
  – LLC: 512kB private per core
  – **Memory: DDR3-1066**
    • 1-2 channel, 1 rank/channel
    • 8 banks, 32 subarrays/bank, **512 cells/bitline**
    • Row-interleaved mapping & closed-row policy

• **TL-DRAM configuration**
  – Total bitline length: **512 cells/bitline**
  – Near segment length: 1-256 cells
Using near segment as a cache improves performance and reduces power consumption.
By adjusting the near segment length, we can trade off cache capacity for cache latency.
Thank you