PRET DRAM Controller: Bank Privatization for Predictability and Temporal Isolation

Sungjun Kim	Columbia University	
Edward A. Lee	UC Berkeley	
Isaac Liu	UC Berkeley	
Hiren D. Patel	University of Waterloo	
Jan Reineke	UC Berkeley	<speaker>

CODES+ISSS as part of ESWEEK 2011

Taipei, Taiwan, October 10th, 2011

Predictability and Temporal Isolation

- Many embedded systems are real-time systems
- Memory hierarchy has a strong influence on their performance:

→ Need for Predictability

Side airbag in car, Reaction in <10 mSec

Crankshaft-synchronous tasks,

Reaction in <45 μSec

- Trend towards integrated architectures:

→ Need for Temporal Isolation

Audio + video playback with latency and bandwidth constraints

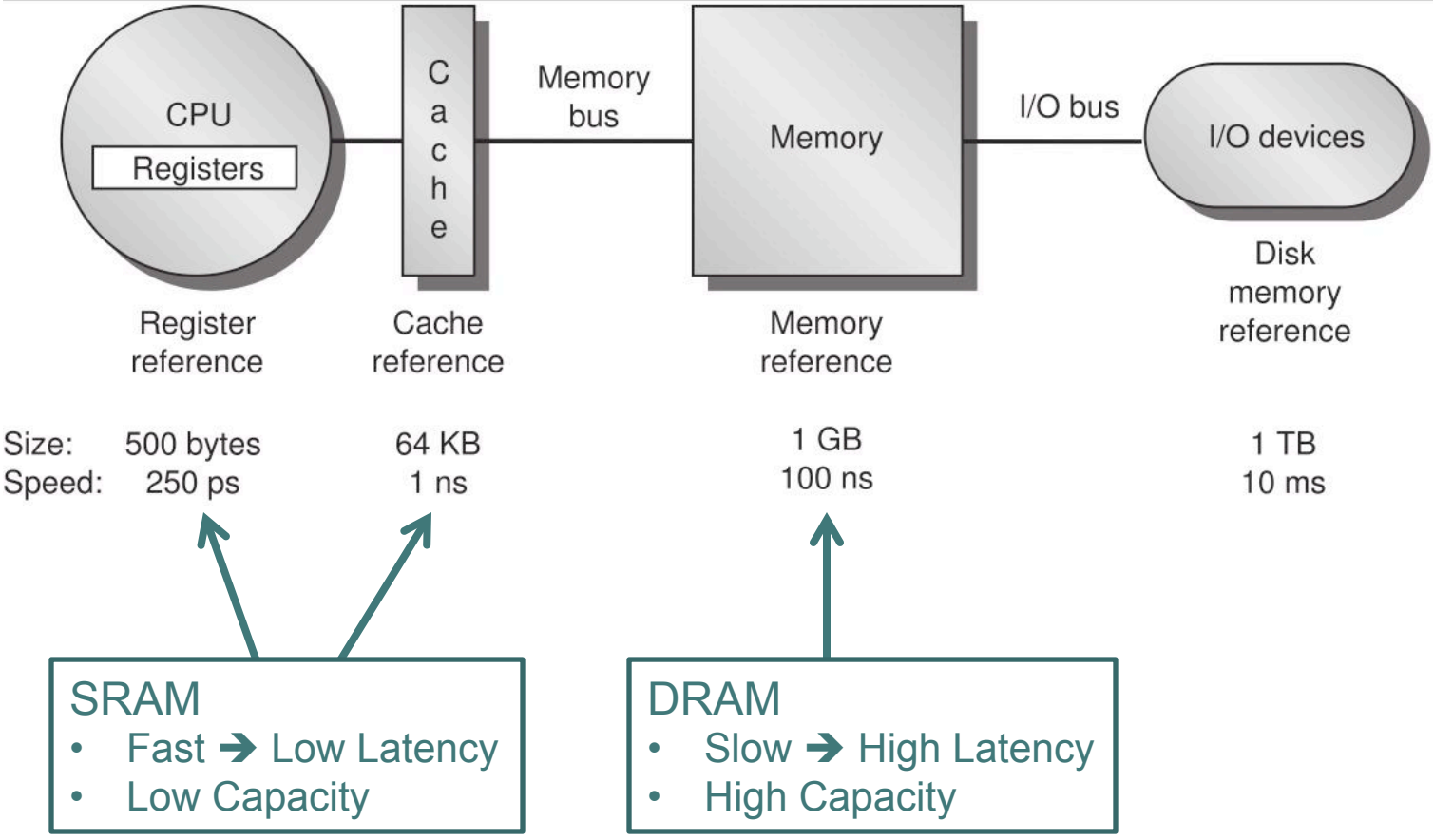
Outline

- Introduction
- DRAM Basics
- Related Work: Predator and AMC
- PRET DRAM Controller: Main Ideas
- Evaluation
- Integration into Precision-Timed ARM

Outline

- Introduction
- DRAM Basics
- Related Work: Predator and AMC
- PRET DRAM Controller: Main Ideas
- Evaluation
- Integration into Precision-Timed ARM

Memory Hierarchy: Dynamic RAM vs Static RAM



from Hennessy and Patterson, Computer Architecture: A Quantitative Approach, 2007.

Dynamic RAM Organization Overview

DRAM Cell

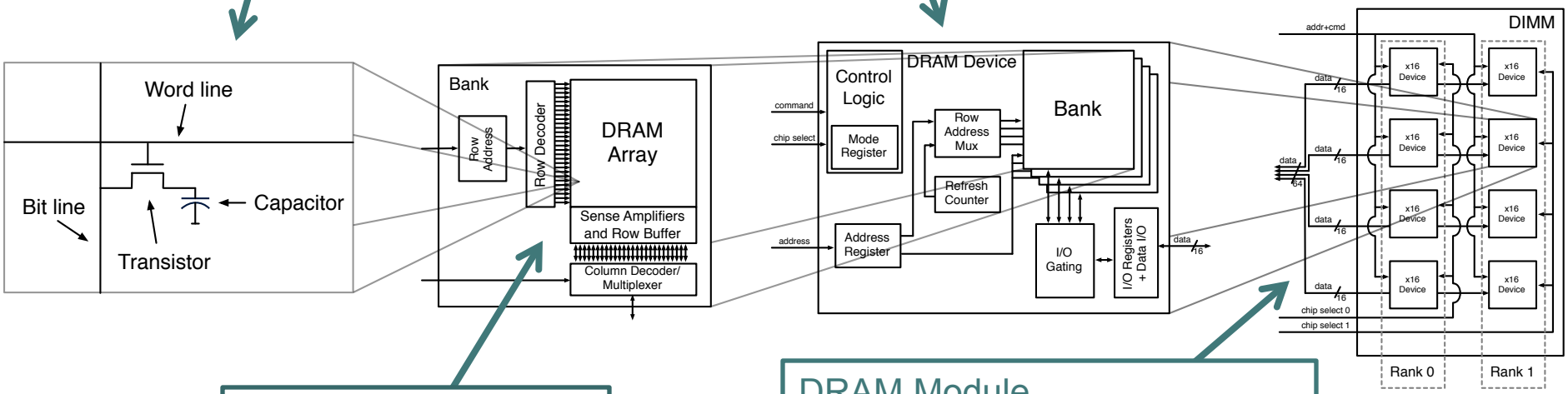
Leaks charge ➔ Needs to be refreshed (every 64ms for DDR2/DDR3) therefore "dynamic"

DRAM Device

Set of DRAM banks +

- *Control logic*
- *I/O gating*

Accesses to banks can be pipelined, however I/O + control logic are shared



DRAM Bank

= Array of DRAM Cells
+ Sense Amplifiers and Row Buffer
Sharing of sense amplifiers and row buffer

DRAM Module

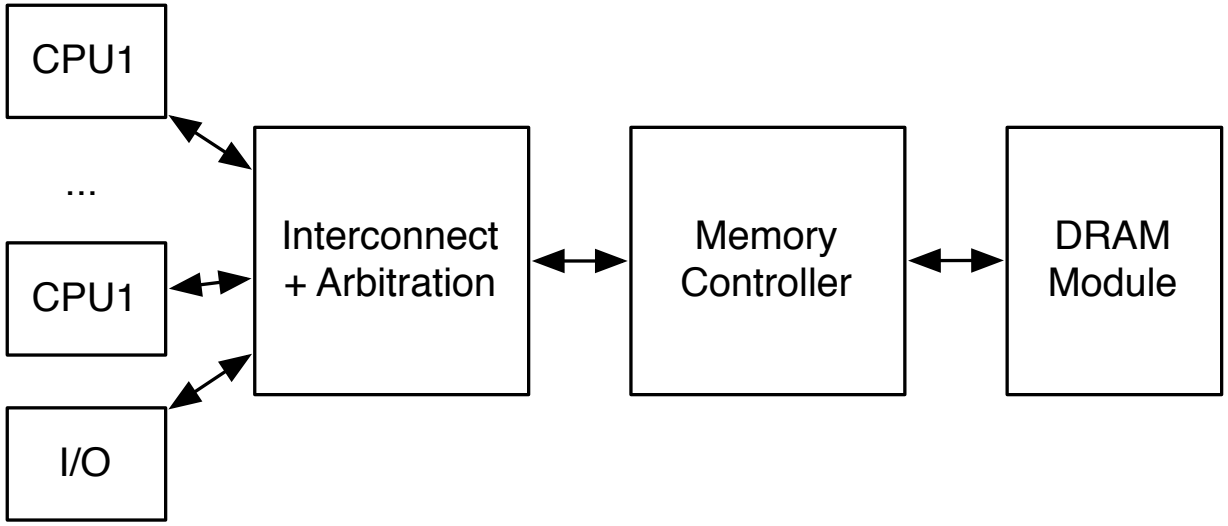
Collection of DRAM Devices

- *Rank = groups of devices that operate in unison*
- *Ranks share data/address/ command bus*

DRAM Memory Controller

Translates sequences of memory accesses by Clients (CPUs and I/O) into **legal** sequences of DRAM commands

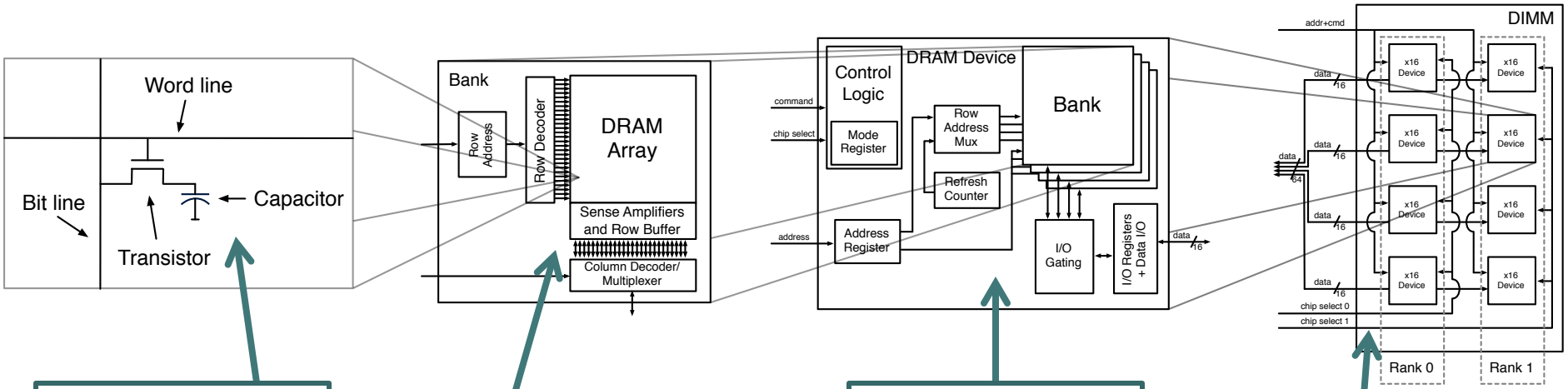
- Needs to obey all timing constraints
- Needs to insert refresh commands sufficiently often
- Needs to translate “physical” memory addresses into row/column/ bank tuples



Dynamic RAM Timing Constraints

DRAM Memory Controllers have to conform to different timing constraints that define minimal distances between consecutive DRAM commands.

Almost all of these constraints are due to the sharing of resources at different levels of the hierarchy:



Needs to insert refresh commands sufficiently often

Rows within a bank share sense amplifiers

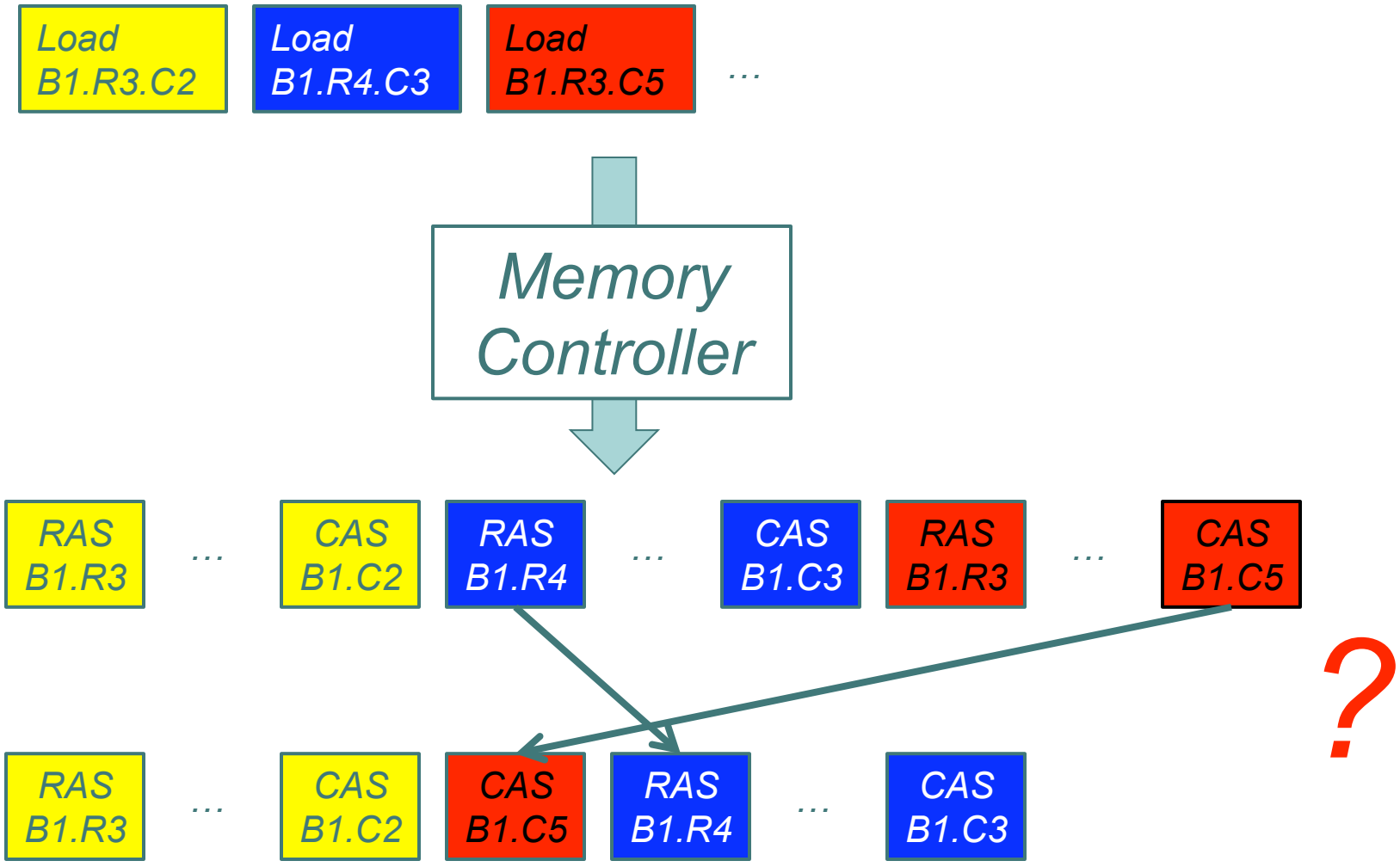
Banks within a DRAM device share I/O gating and control logic

Different ranks share data/address/ command busses

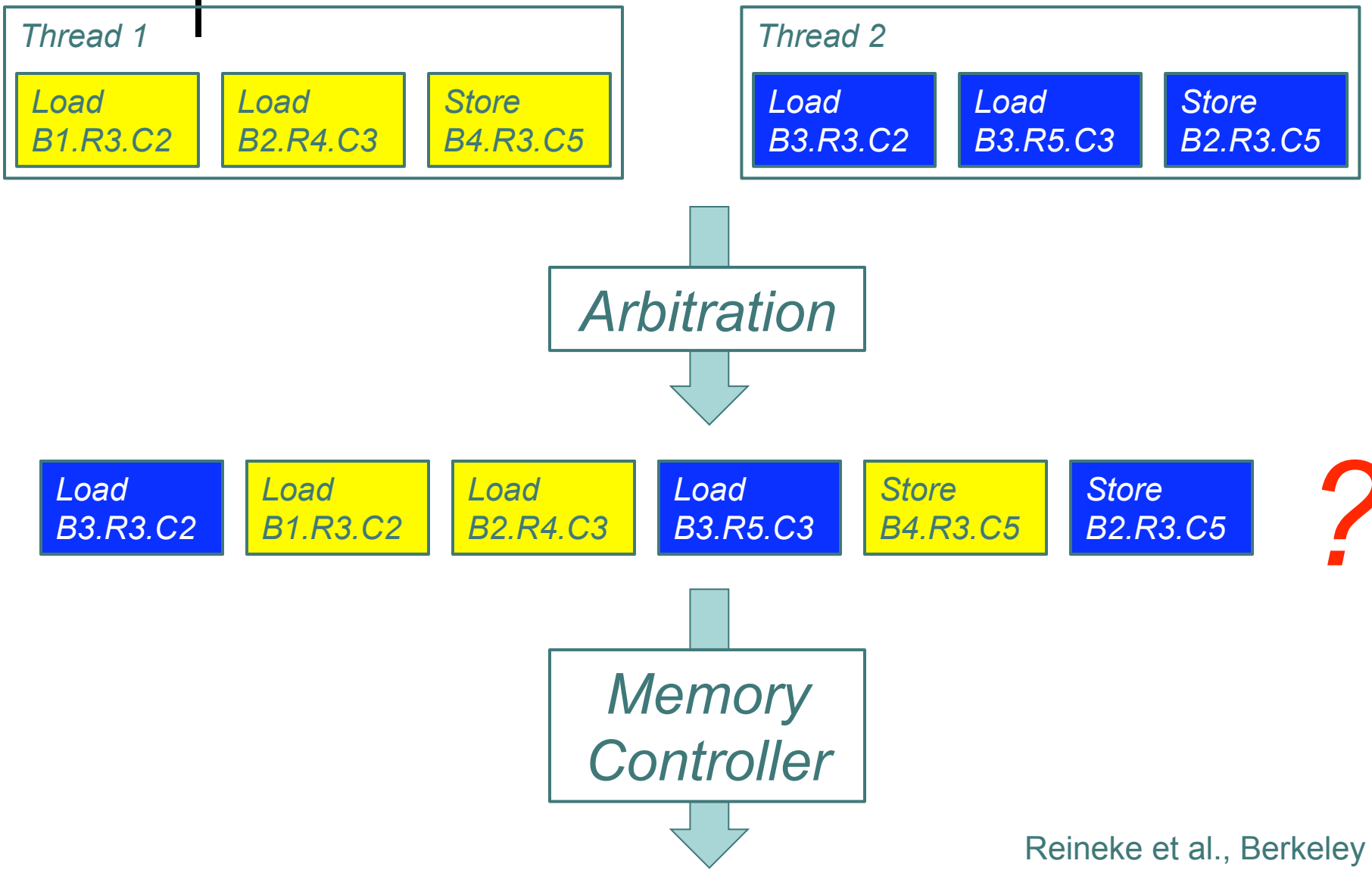
General-Purpose DRAM Controllers

- Schedule DRAM commands dynamically
- Timing hard to predict even for single client:
  - Timing of request depends on past requests:
    - Request to same/different bank?
    - Request to open/closed row within bank?
    - Controller might reorder requests to minimize latency
  - Controllers dynamically schedule refreshes
- Non-composable timing. Timing depends on behavior of other clients:
  - They influence sequence of “past requests”
  - Arbitration may or may not provide guarantees

General-Purpose DRAM Controllers



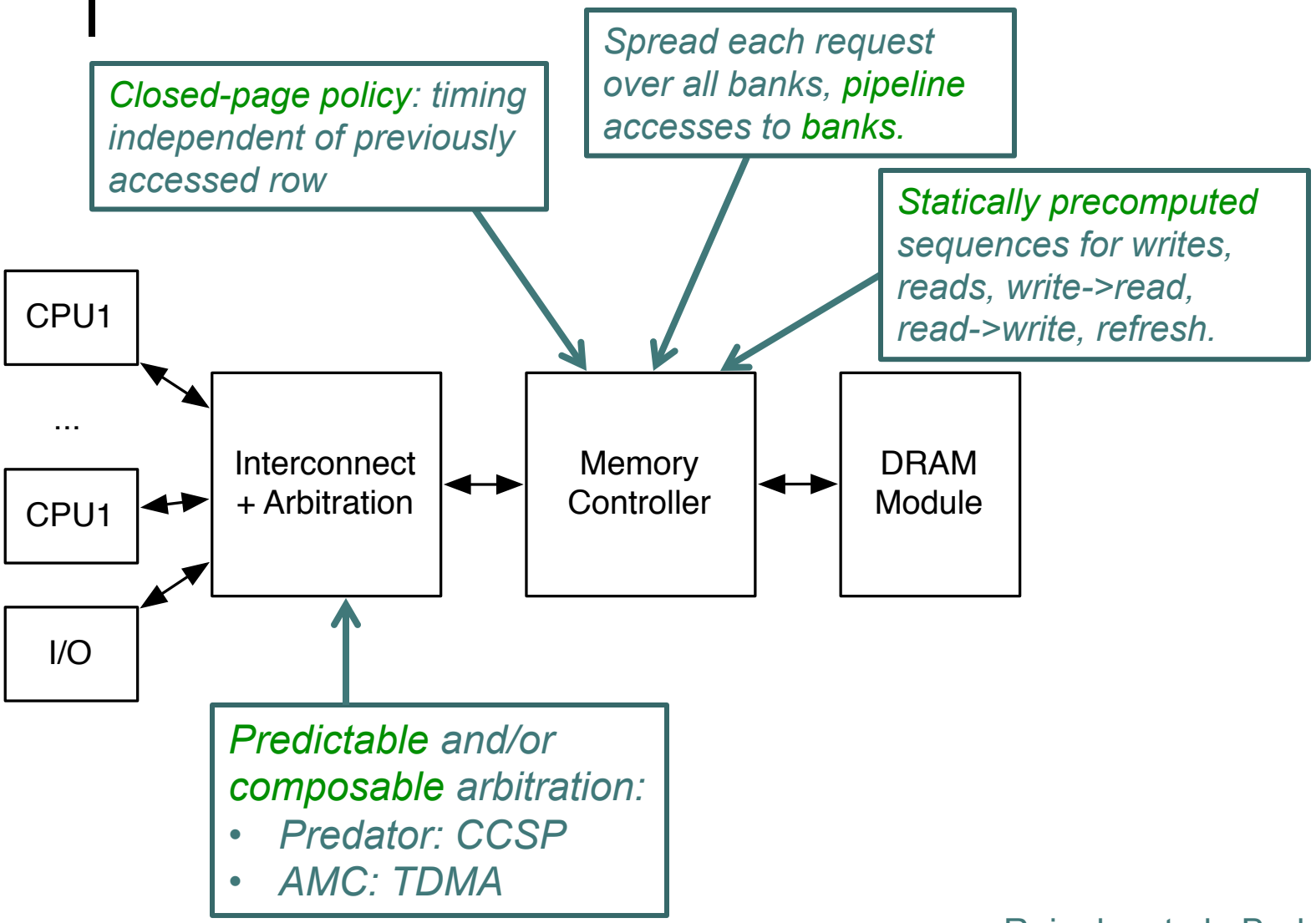
General-Purpose DRAM Controllers



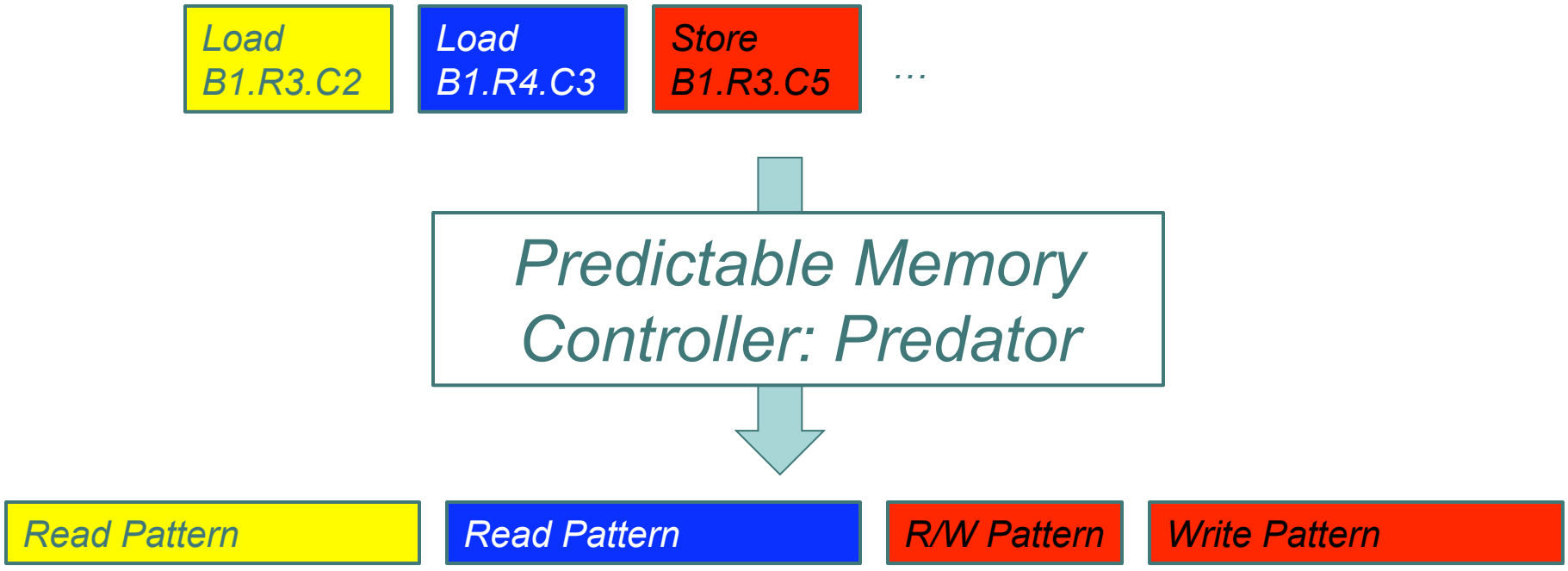
Outline

- Introduction
- DRAM Basics
- Related Work: Predator and AMC
- PRET DRAM Controller: Main Ideas
- Evaluation
- Integration into Precision-Timed ARM

Predictable DRAM Controllers: Predator (Eindhoven) and AMC (Barcelona)



Predictable DRAM Controllers: Predator (Eindhoven)



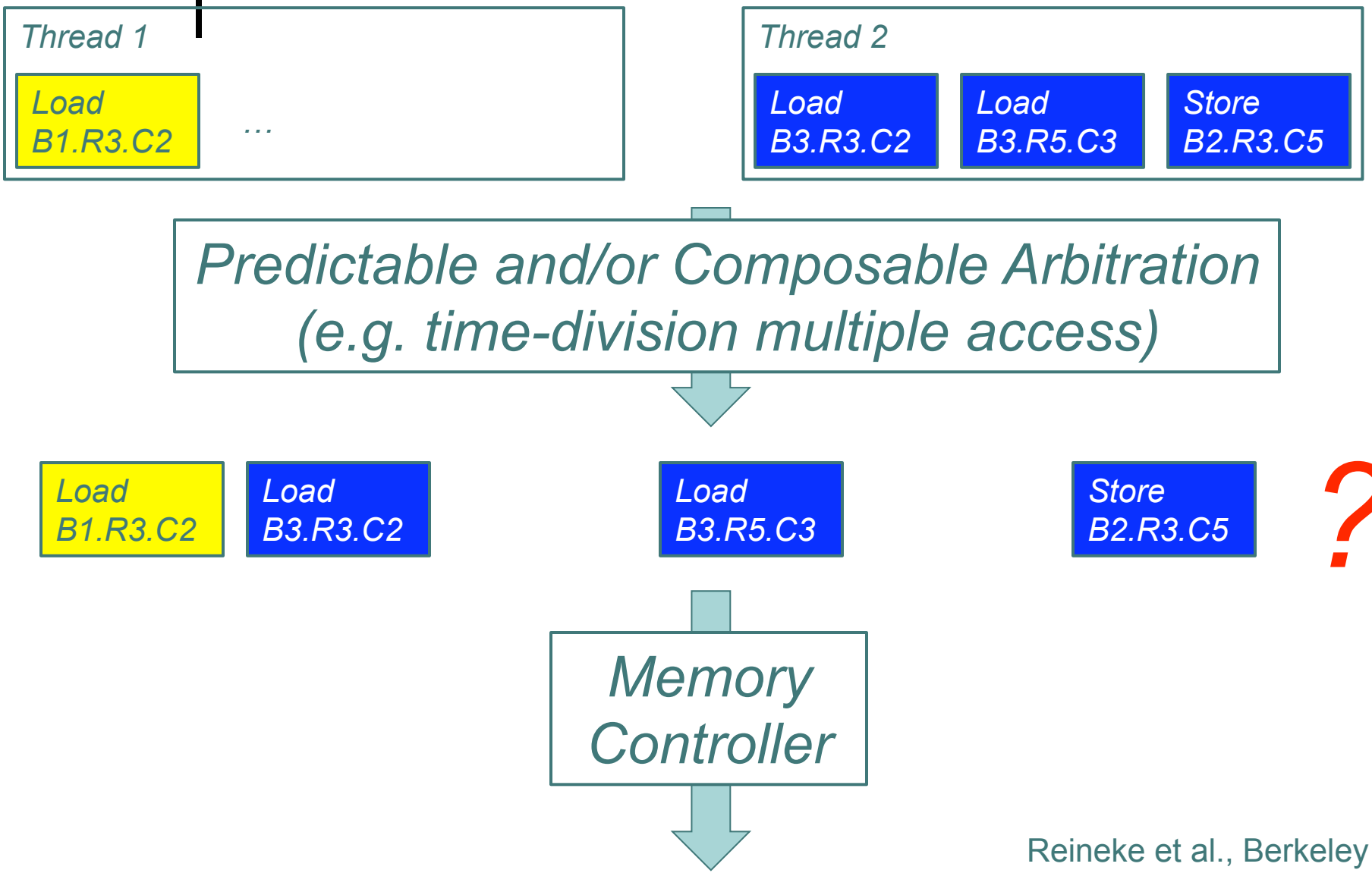
Closed-page policy: timing independent of previously accessed row

Spread each request over all banks, pipeline accesses to banks.

→ *increases access granularity*

Statically precomputed sequences for writes, reads, write->read, read->write, refresh.

Predictable DRAM Controllers: Predator (Eindhoven) and AMC (Barcelona)

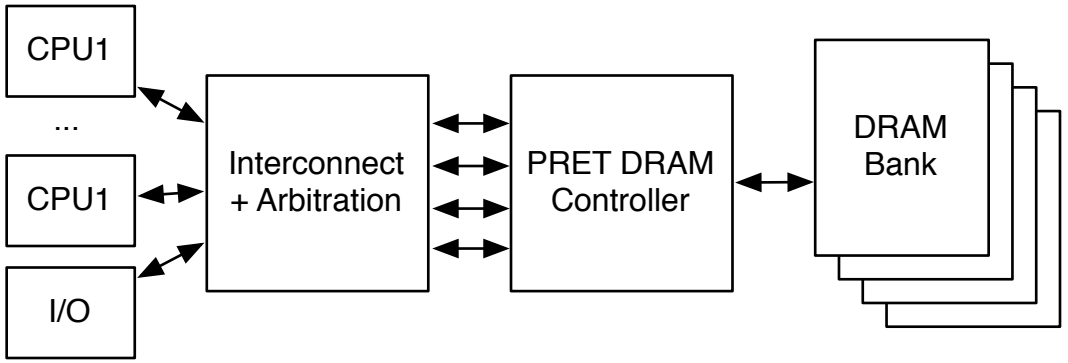


Outline

- Introduction
- DRAM Basics
- Related Work: Predator and AMC
- PRET DRAM Controller: Main Ideas
- Evaluation
- Integration into Precision-Timed ARM

PRET DRAM Controller: Three Innovations

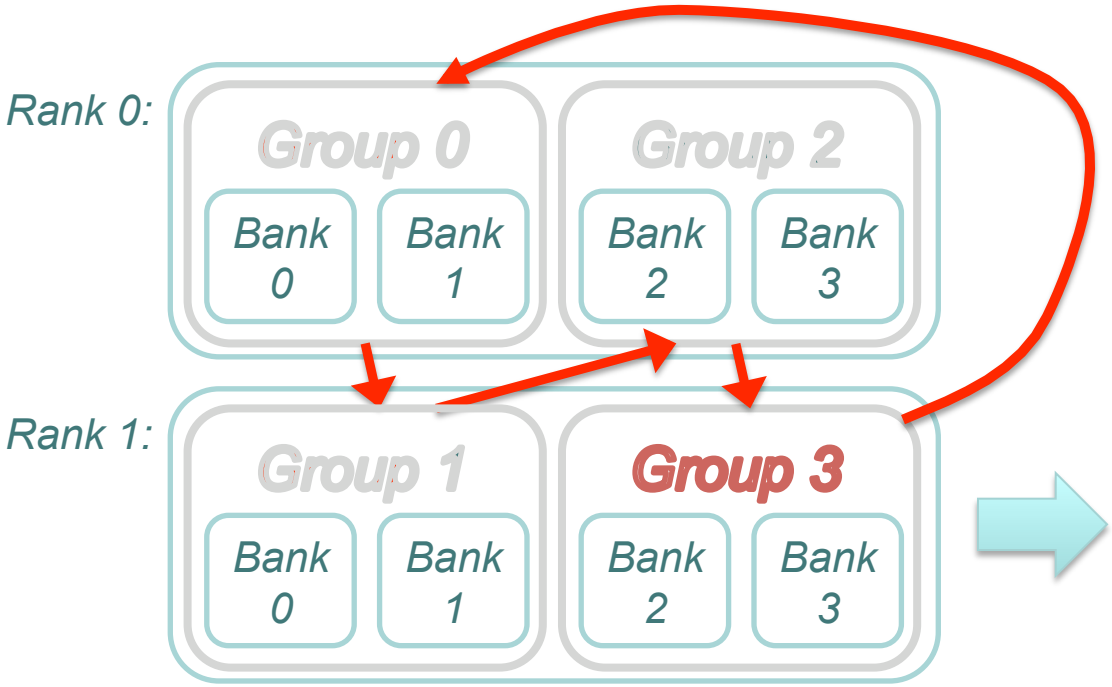
- Expose internal structure of DRAM devices:
  - Expose individual banks within DRAM device as multiple independent resources



- Defer refreshes to the end of transactions
  - Allows to hide refresh latency
- Perform refreshes “manually”:
  - Replace standard refresh command with multiple reads

PRET DRAM Controller: Exploiting Internal Structure of DRAM Module

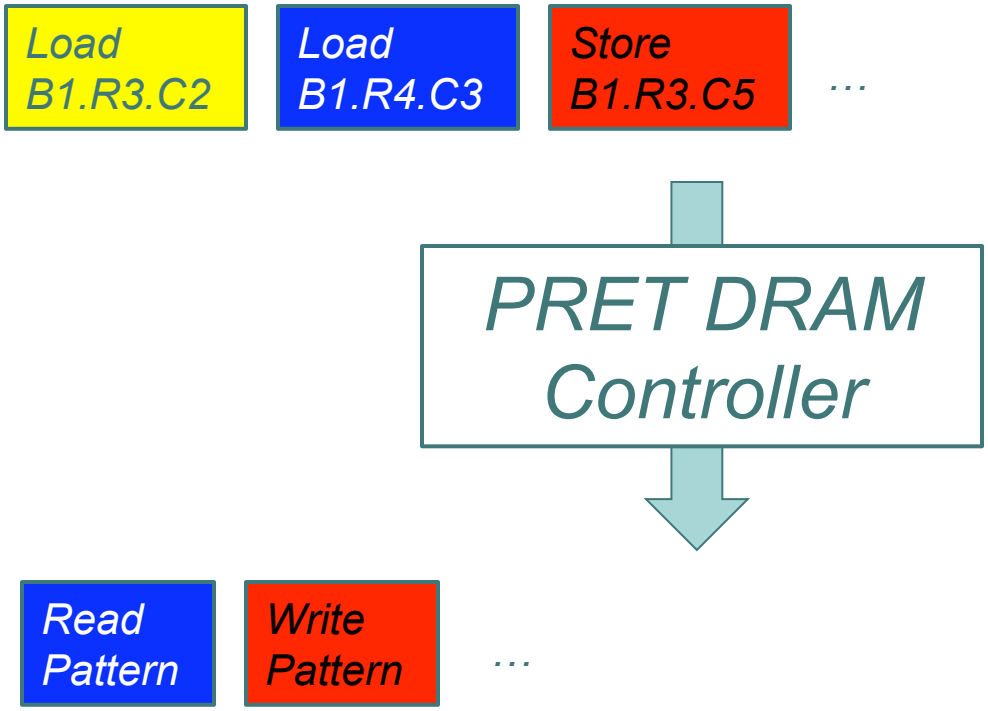
- Consists of 4-8 banks in 1-2 ranks
  - Share only command and data bus, otherwise independent
- Partition into four groups of banks in alternating ranks
- Cycle through groups in a time-triggered fashion



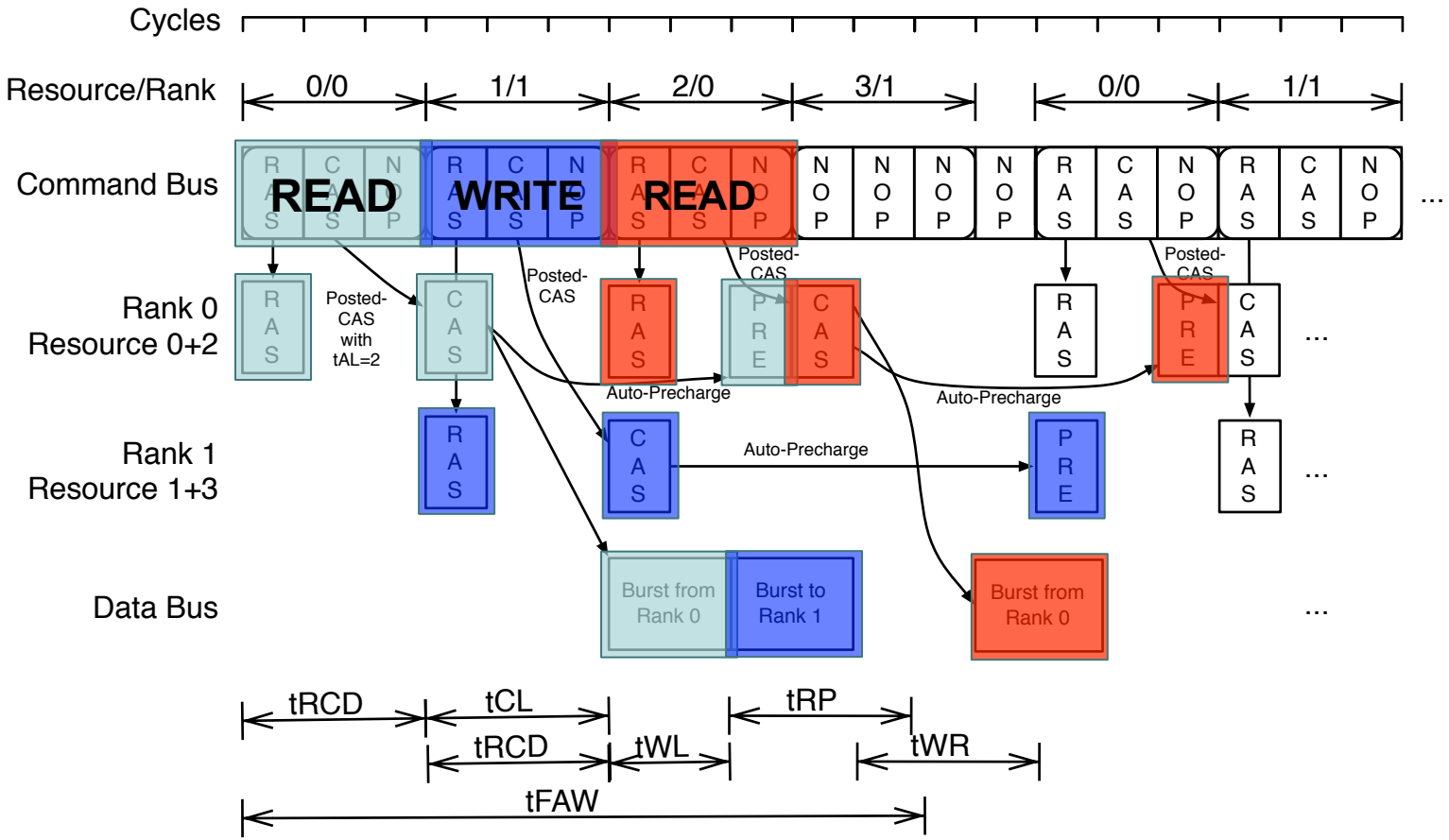
- *Successive accesses to same group obey timing constraints*
- *Reads/writes to different groups do not interfere*

Provides four independent and predictable resources

PRET DRAM Controller: Exploiting Internal Structure of DRAM Module

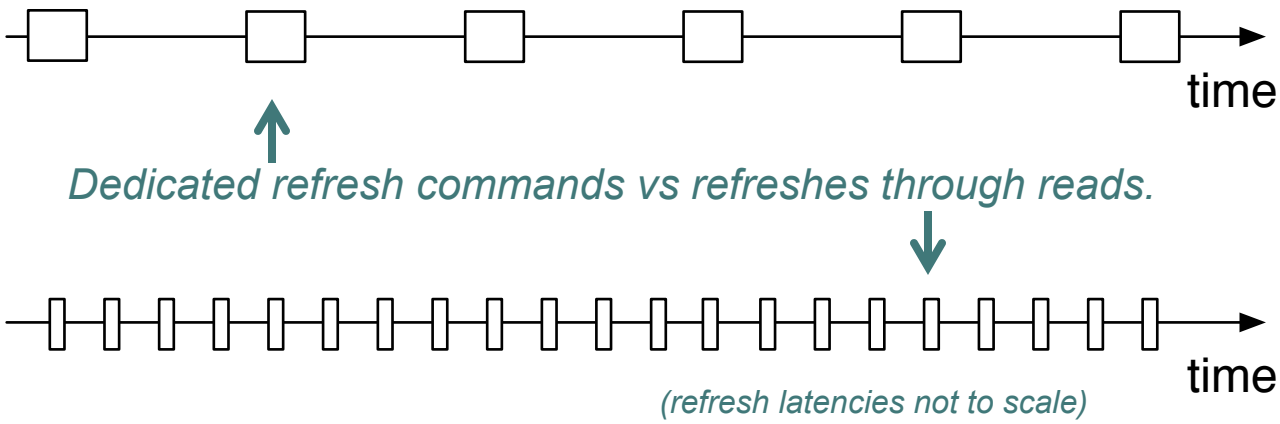


Pipelined Bank Access Scheme



PRET DRAM Controller: "Manual" Refreshes

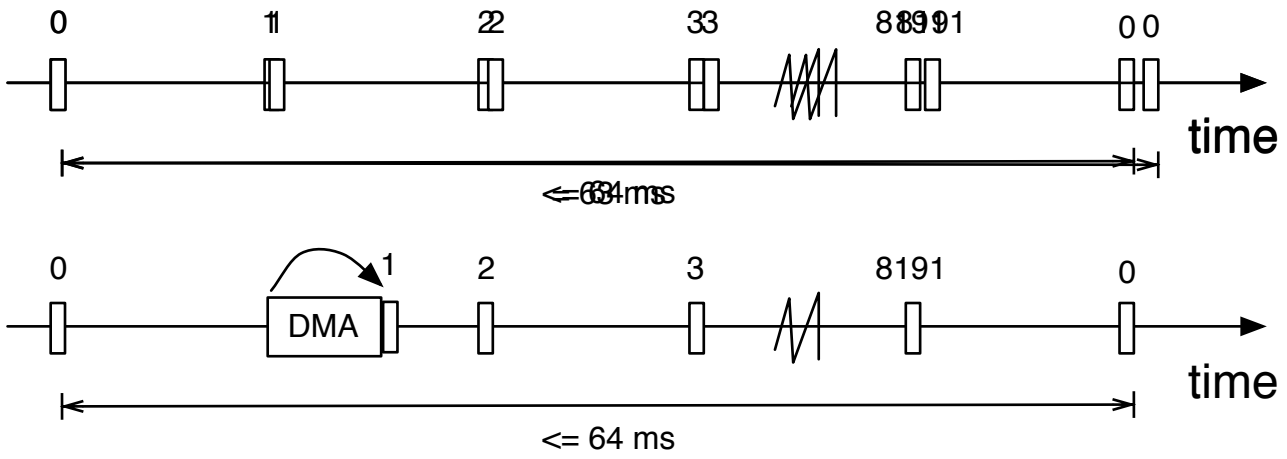
- Every row needs to be refreshed every 64ms
- Dedicated refresh commands refresh one row in each bank at once
- We replace these with "manual" refreshes through reads
  - Improves worst-case latency of short requests



Dedicated refresh commands vs refreshes through reads.

PRET DRAM Controller: Defer Refreshes

- Refreshes do not have to happen periodically
- Refresh every row *at least* every 64 ms
- Schedule refreshes slightly more often than necessary → Enables to defer refreshes



General-Purpose DRAM Controller vs PRET DRAM Controller

General-Purpose Controller

- Abstracts DRAM as a single shared resource
- Schedules refreshes dynamically
- Schedules commands dynamically
- “Open page” policy speculates on locality

PRET DRAM Controller

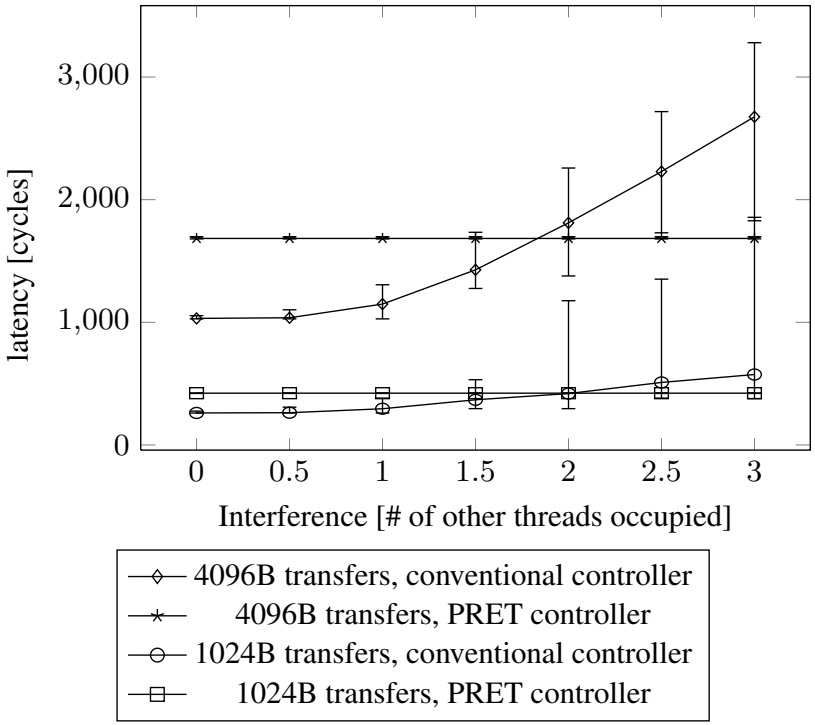
- Abstracts DRAM as multiple independent resources
- Refreshes as reads: shorter interruptions
- Defer refreshes: improves perceived latency
- Follows periodic, time-triggered schedule
- “Closed page” policy: access-history independence

Outline

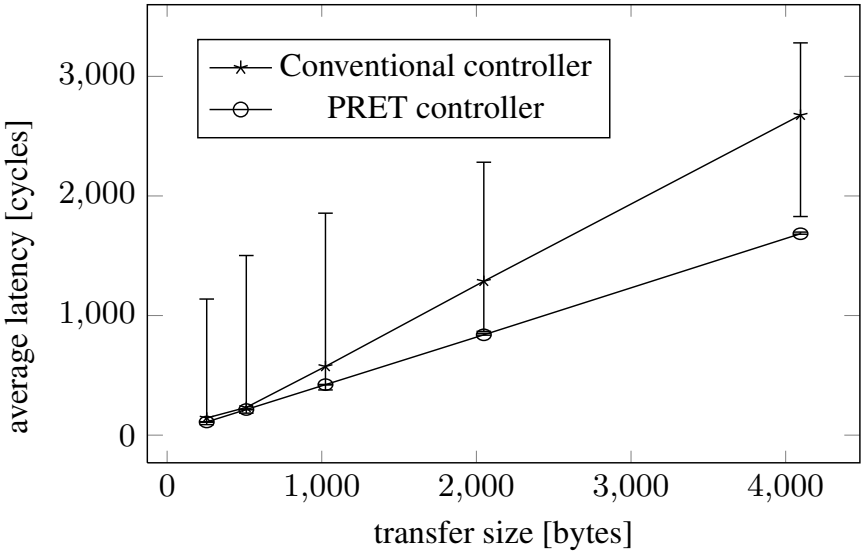
- Introduction
- DRAM Basics
- Related Work: Predator and AMC
- PRET DRAM Controller: Main Ideas
- Evaluation
- Integration into Precision-Timed ARM

Conventional DRAM Controller (DRAMSim2) vs PRET DRAM Controller: Latency Evaluation

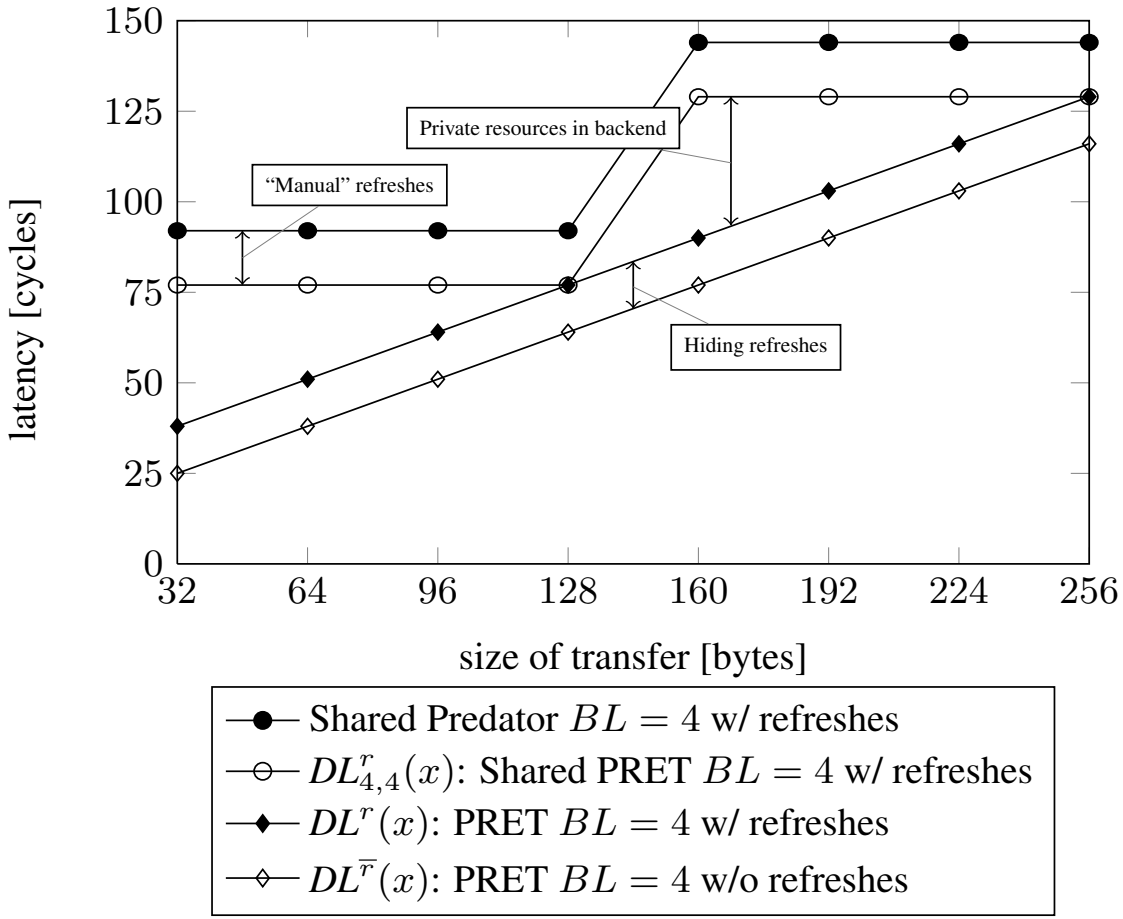
Varying Interference:



Varying Transfer Size:



PRET DRAM Controller vs Predator: Analytical Evaluation

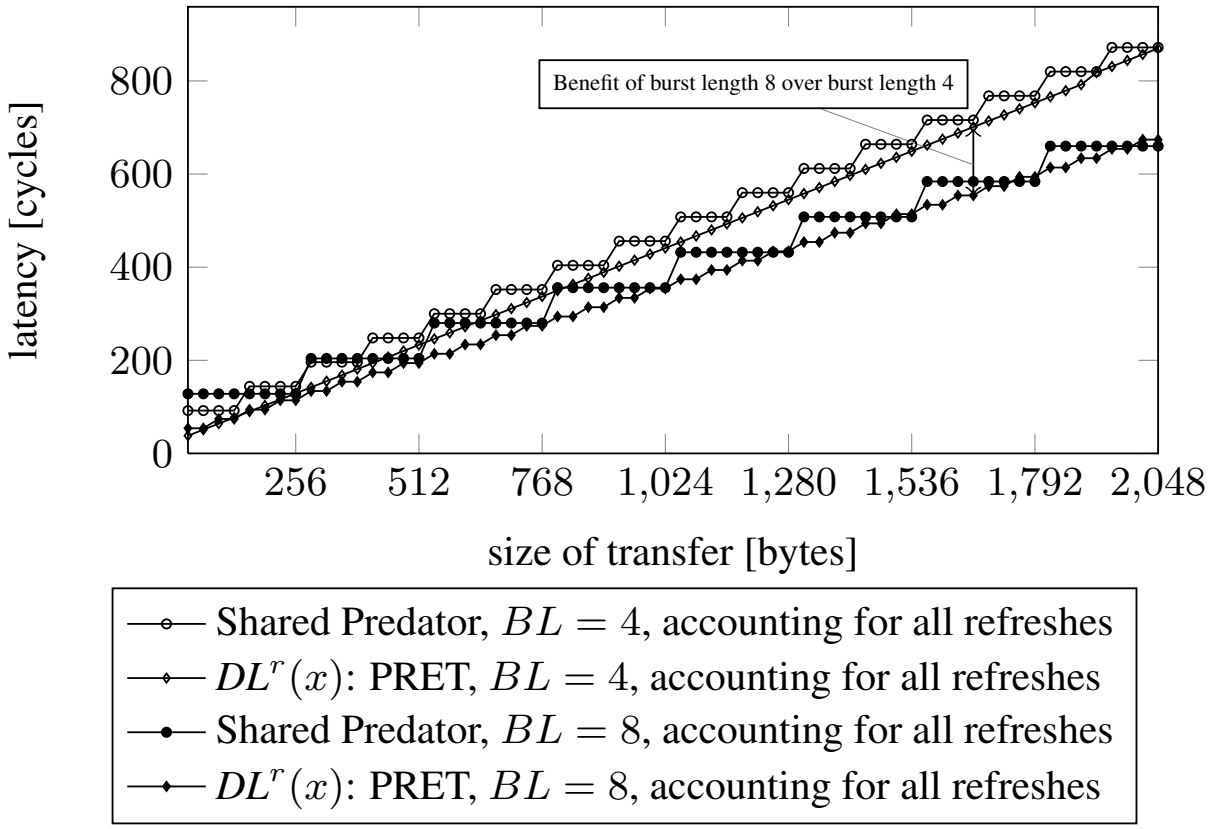


Predator:

- *abstracts DRAM as single resource*
- *uses standard refresh mechanism*

➔ *PRET controller improves worst-case access latency of small transfers*

PRET DRAM Controller vs Predator: Analytical Evaluation

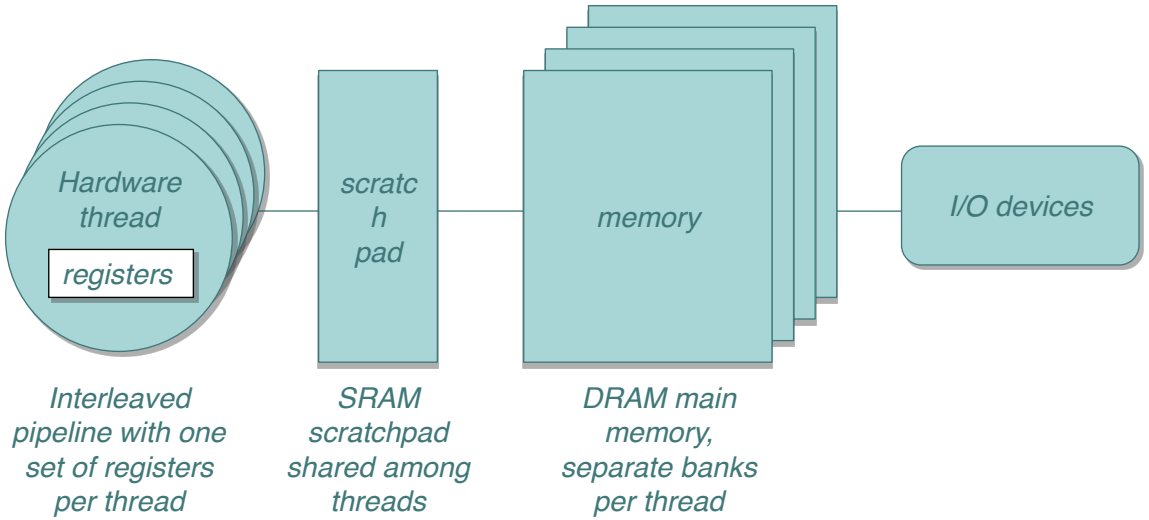


- *Less of a difference for larger transfers*
- *Predator provides slightly higher bandwidth due to more efficient refresh mechanism*

Outline

- Introduction
- DRAM Basics
- Related Work: Predator and AMC
- PRET DRAM Controller: Main Ideas
- Evaluation
- Integration into Precision-Timed ARM

Precision-Timed ARM (PTARM) Architecture Overview *http://chess.eecs.berkeley.edu/pret/*



- **Thread-Interleaved Pipeline** for predictable timing *without* sacrificing high throughput
- One private DRAM Resource + DMA Unit per Hardware Thread
- Shared **Scratchpad** Instruction and Data Memories for low latency access

Conclusions and Future Work

- Temporal isolation and improved worst-case latency by bank privatization
- How to program the inverted memory hierarchy?

Raffaello Sanzio da Urbino – The Athens School

References

Related Work on Memory Controllers:

- M. Paolieri, E. Quiñones, F. Cazorla, and M. Valero, “An analyzable memory controller for hard real-time CMPs,” IEEE Embedded Systems Letters, vol. 1, no. 4, pp. 86–90, 2010.
- B. Akesson, K. Goossens, and M. Ringhofer, “Predator: a predictable SDRAM memory controller,” in CODES+ISSS. ACM, 2007, pp. 251–256.

Work within the PRET project:

- [CODES ’11] Jan Reineke, Isaac Liu, Hiren D. Patel, Sungjun Kim, Edward A. Lee, PRET DRAM Controller: Bank Privatization for Predictability and Temporal Isolation, *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, October, 2011.
- [DAC ‘11] Dai Nguyen Bui, Edward A. Lee, Isaac Liu, Hiren D. Patel, Jan Reineke, Temporal Isolation on Multiprocessing Architectures, *Design Automation Conference (DAC)*, June, 2011.
- [Asilomar ‘10] Isaac Liu, Jan Reineke, and Edward A. Lee, PRET Architecture Supporting Concurrent Programs with Composable Timing Properties, in *Signals, Systems, and Computers (ASILOMAR)*, Conference Record of the Forty Fourth Asilomar Conference, November 2010, Pacific Grove, California.
- [CASES ’08] Ben Lickly, Isaac Liu, Sungjun Kim, Hiren D. Patel, Stephen A. Edwards and Edward A. Lee, " Predictable Programming on a Precision Timed Architecture," in Proceedings of International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES), Piscataway, NJ, pp. 137-146, IEEE Press, October, 2008.