# Memory Access Scheduler

Matthew Cohen, Alvin Lin

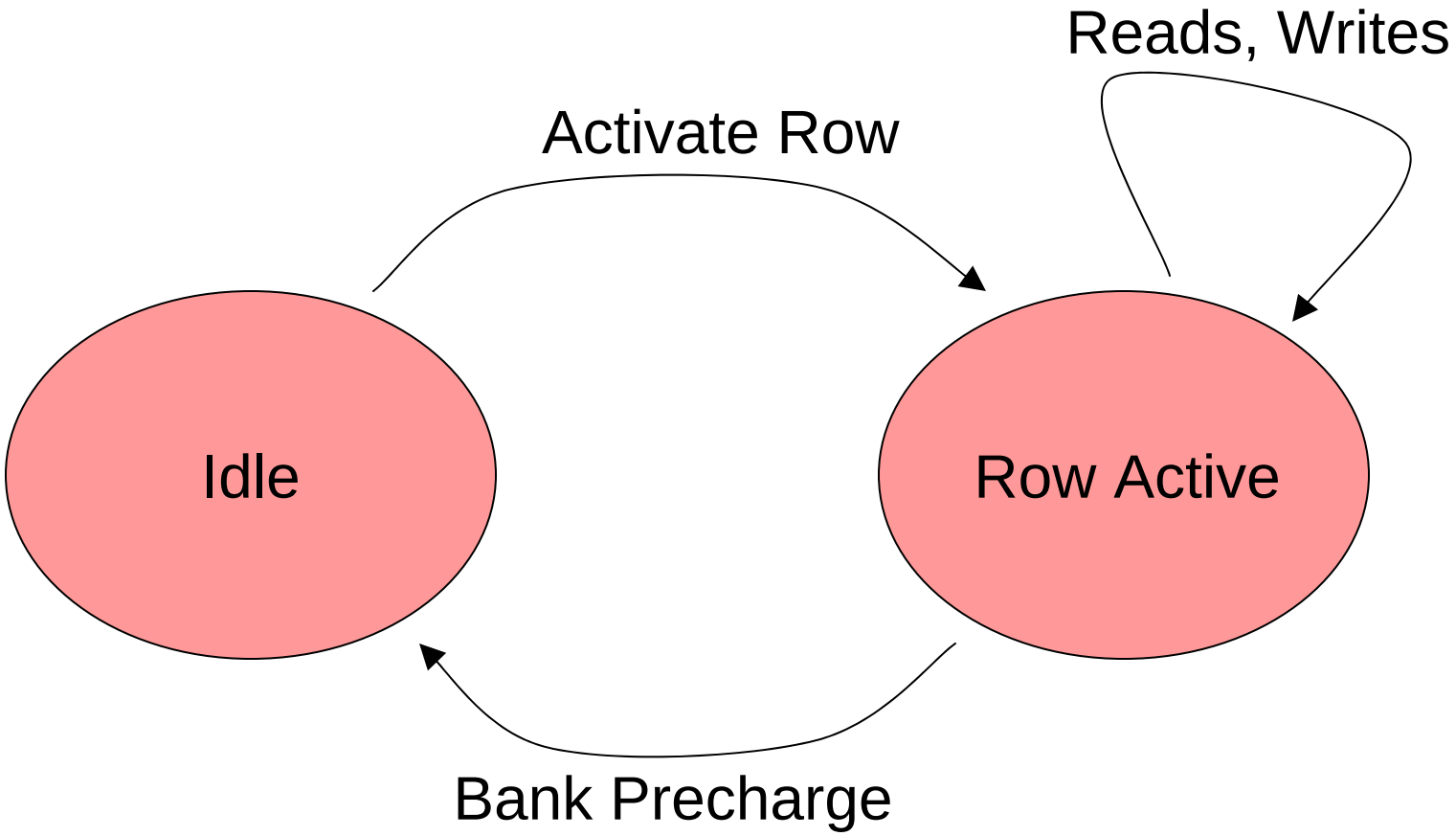
6.884 – Complex Digital Systems

May 6th, 2005

# Why Use Scheduling?

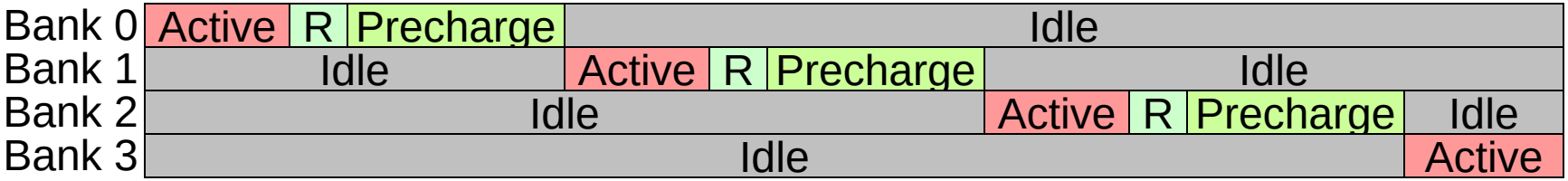
- Sequential accesses to DRAM are wasteful
- Improve latency and bandwidth of memory requests
- Order requests to take advantage of DRAM characteristics

# DRAM Bank FSM

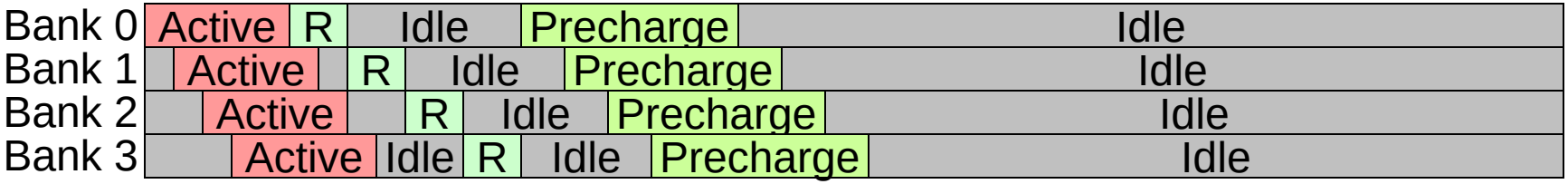


# Memory Access Scheduling

Traditional Scheduling:

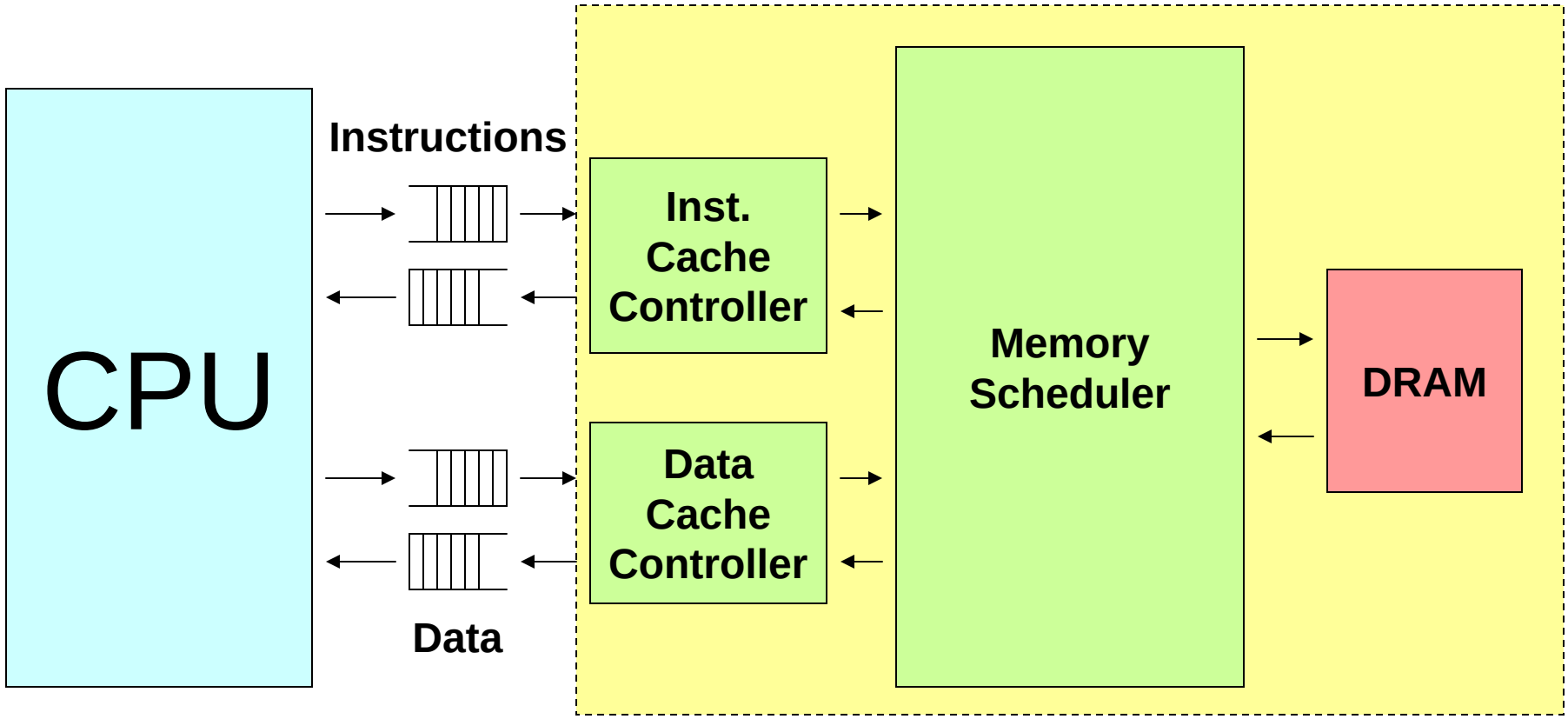



Memory Access Scheduling:




- Avoid data line conflicts (read/write)
- Avoid control line conflicts

# High-Level Architecture



# Instruction and Data Cache

- Separate I- and D-caches
- Fully parameterizable sizes
- Direct mapped caches
- Write-through, no-write-allocate
- Four words per cache line

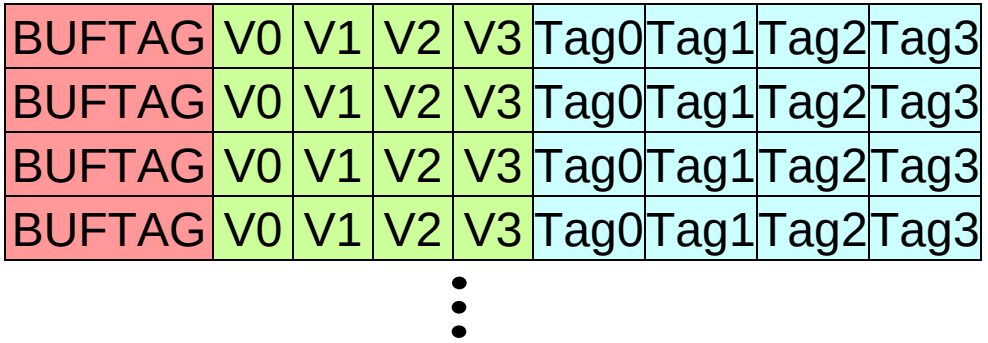
| V | Tag | Word 0 | Word 1 | Word 2 | Word 3 |
|---|---|---|---|---|---|
| V | Tag | Word 0 | Word 1 | Word 2 | Word 3 |
| V | Tag | Word 0 | Word 1 | Word 2 | Word 3 |

# Incremental Design

- Fully blocking, single word per line
- Fully blocking, four words per line
- Hit under miss
- Miss under miss
  - Necessary for full benefits of scheduling

# Non-Blocking Cache Architecture

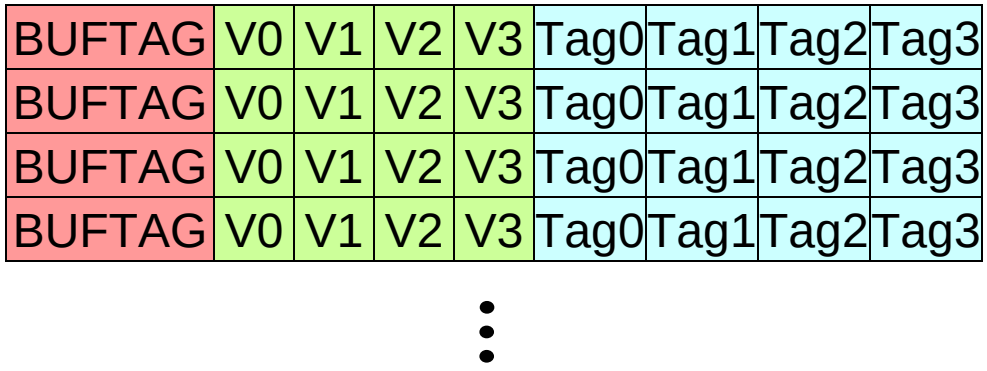
- On cache load miss, add request to Pending Request Buffer (PRB)



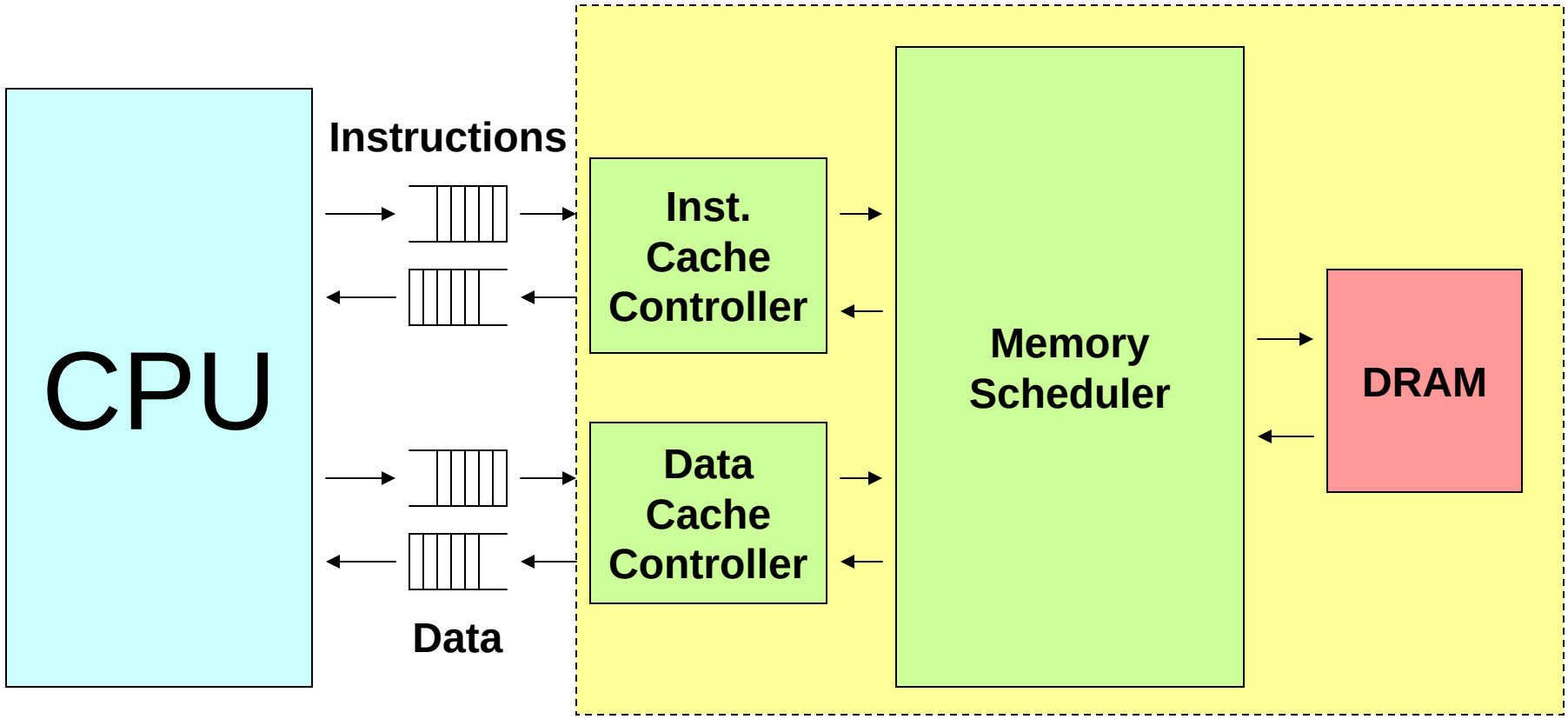

- Place µP tag in Tag location, set Valid, issue read request to scheduler with tag = PRB index
- If another read to same line, set tag and valid but no new read request
- On return of data, match tag to PRB line, retrieve µP tag of valid entries, return data to µP

# Non-Blocking Cache Architecture

- On cache store request, search PRB
- If already issued read to this line, stall



# High-Level Architecture



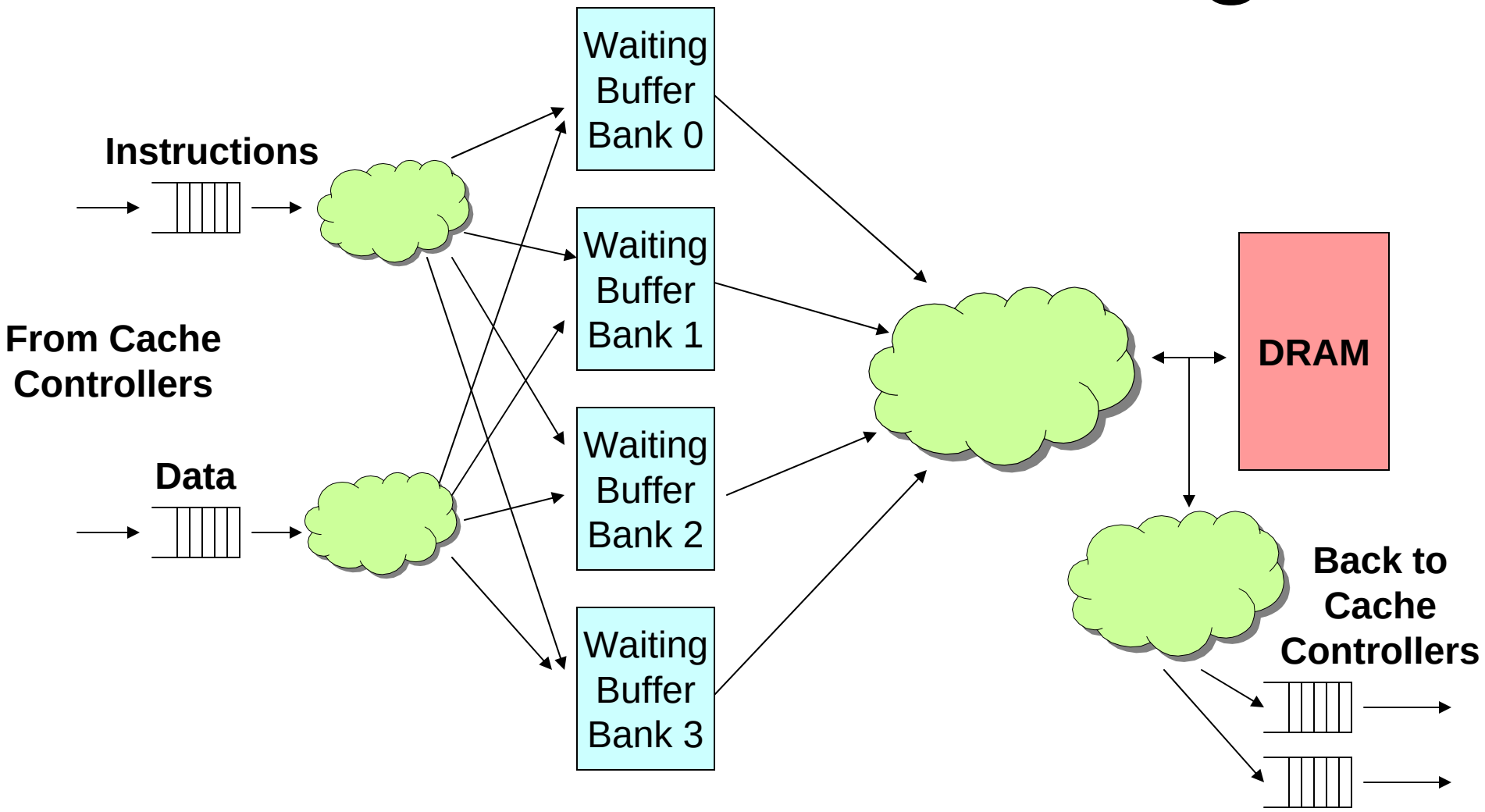
# Scheduler Overview

- Cache misses are sent to the scheduler
- Scheduler is responsible for interfacing with the DRAM
- Requests may be honored out of order

# Scheduler Tasks

- Keep waiting buffers of pending memory requests
- Prioritize accesses in waiting buffer
- Respect timing of the DRAM
- Capture data coming back from DRAM
- Keep the DRAM busy!

# Scheduler RTL Design



# Incremental Design

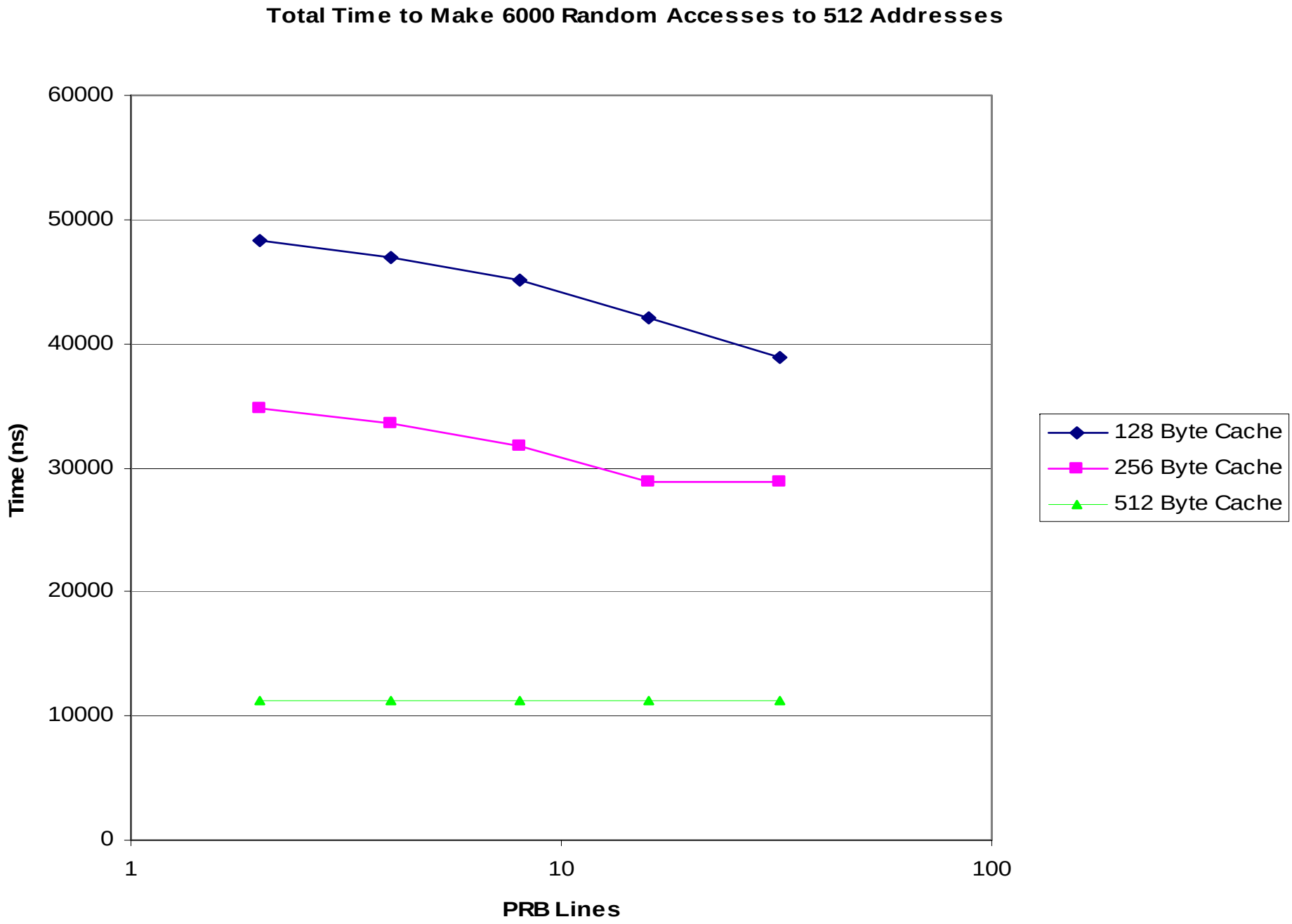
- Blocking In-Order Scheduler
- FIFOs as Waiting Buffers and In-Order Scheduling
- Real Waiting Buffers and Interleaved Scheduling

# Infinite Compile Time

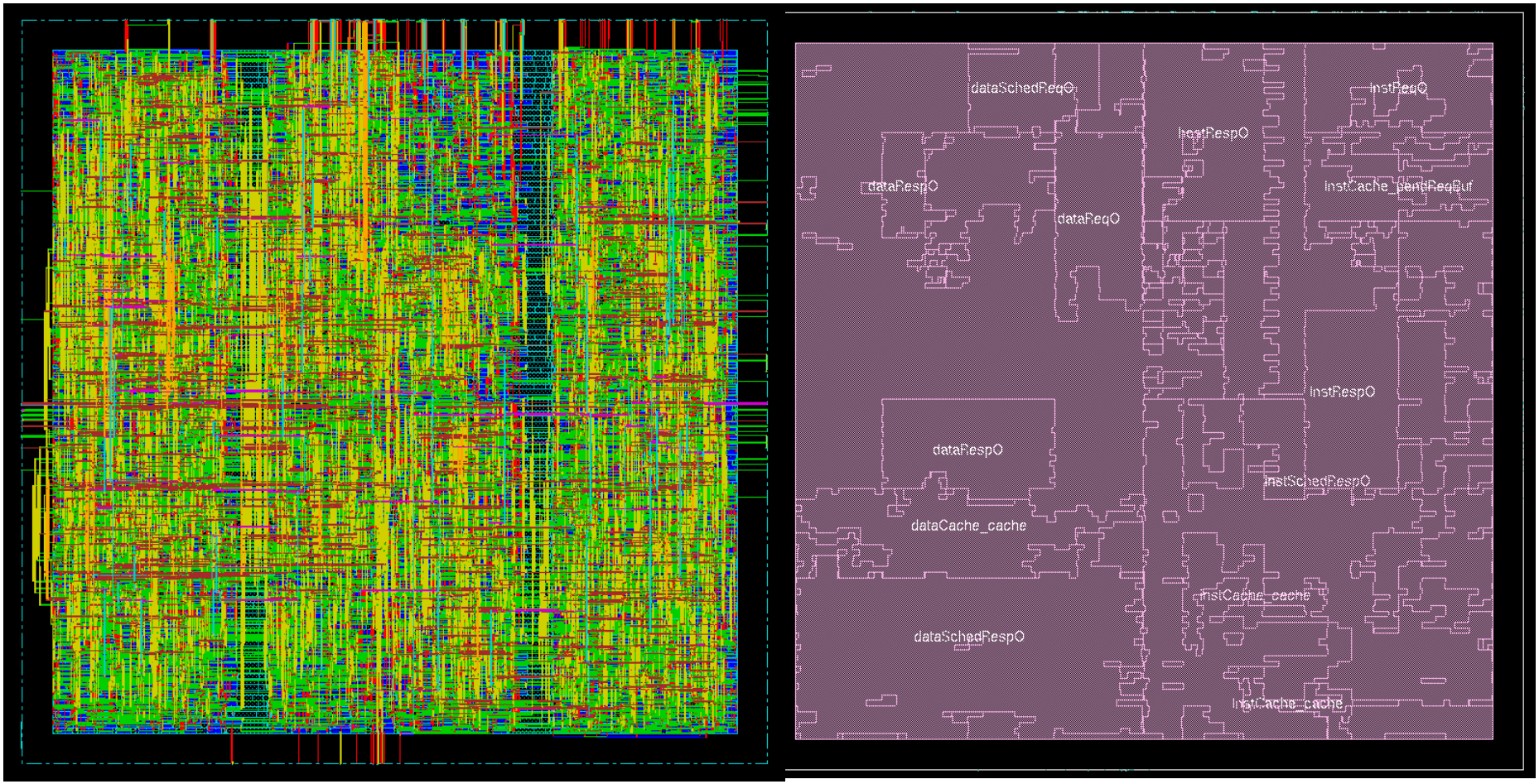
- Scheduler exploded in complexity
- Huge amount of combinational logic
- Memory access scheduling is a difficult problem
- DRAM is not designed to work easily with scheduling

# Architectural Exploration

- Change cache size to adjust cache miss percentage
- Change PRB size to allow for scheduling optimization
- Larger sizes should yield better results but higher cost


Total Time to Make 6000 Random Accesses to 512 Addresses
60000
50000
40000
30000
20000
10000
0
Time (ns)
1
10
100
PRB Lines
128 Byte Cache
256 Byte Cache
512 Byte Cache

# Synthesis Results (Area = 196,117.6 μm²)



# Conclusion

- Memory becoming bottleneck for computer systems
- In-order memory access is simple in logic but wasteful in performance
- Memory access scheduling is much more efficient in theory, but complex in implementation

# Acknowledgements


- 6884-bluespec
- 6884-staff
- group1, for teaching us how to use Vector, even if you didn’t realize it…