# *ZORO: Zack's Open-Row Oriented Memory Scheduler*

Jack Davidson and
Zackery Painter

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

# *Introduction*

- The goal was to determine if we can make a better scheduling algorithm using a predictor
- Originally wanted to predict if we should leave the page open or close the page after a read / write
- However, decided it would be more appropriate to predict what we should send from the CPU to the memory controller

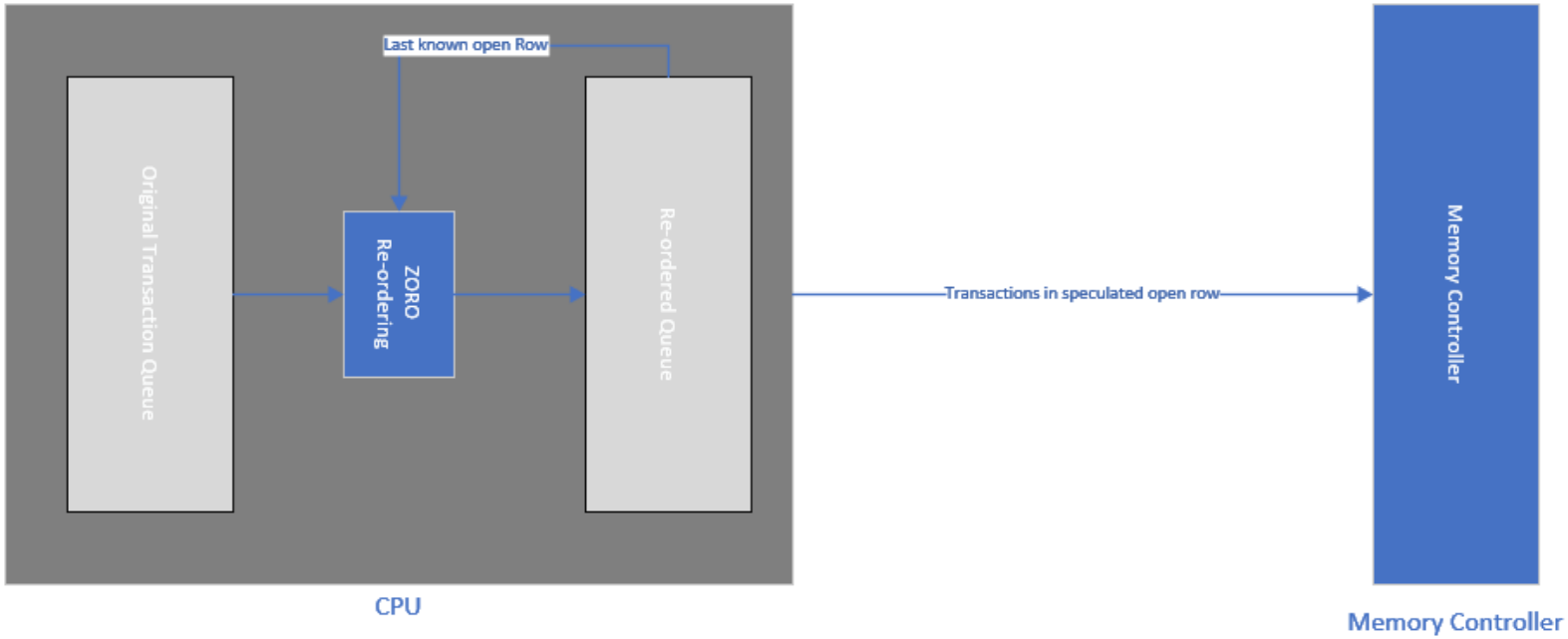
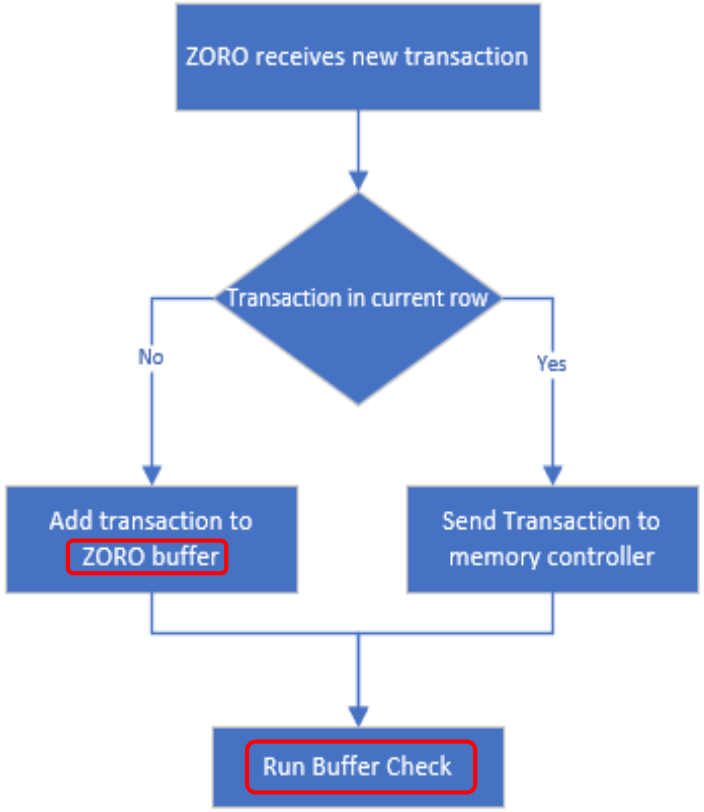


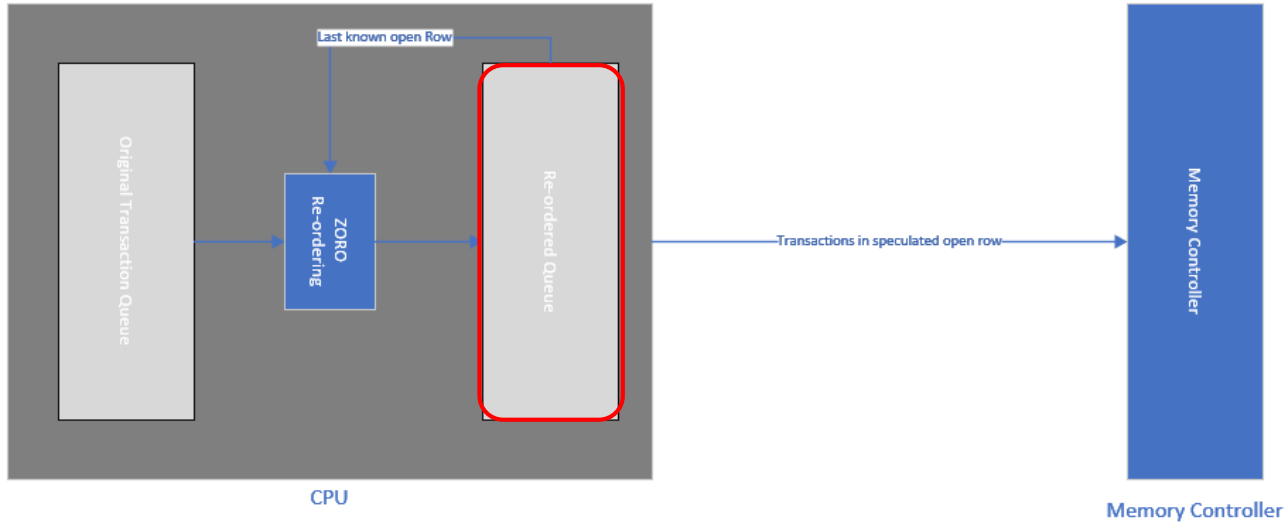

Figure 1 – General idea of ZORO

# ZORO

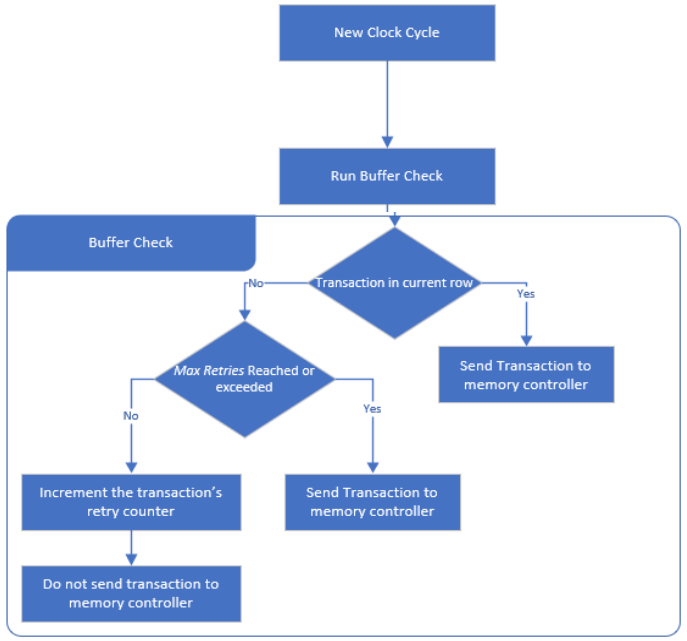
- Important vocabulary
  - *Transaction Groups – A transaction group is group of memory transactions that are in the same row*
  - *cation that transactions go before sent to the memory controller*

| Transaction | Address | Row |
|---|---|---|
| T1 | 0x0A | 1 |
| T2 | 0xAB | 1 |



# *ZORO*




- How does ZORO work?
  - *ZORO aims to take advantage of FR-FCFS by always attempting to send groups of memory transactions that are in the current row*
  - *When a new transaction is received:*
    - Calculate transaction's row
    - Compare to last known opened row
    - If equal, send to memory controller
    - Otherwise, put in ZORO buffer
    - Run a buffer check
  - On each clock cycle
    - Run a buffer check

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

# *Simulations*

- Gem5 with a side of DRAMsim3
- Put it in the oven for 100 iterations
- Gives a good balance of command flavor and runtime spice
- We stuck to the standard Coremark workload, didn't have time to experiment with other benchmark recipes
- Tried a few different cache sizes to increase the number of memory transactions and expand out palates

# *Simulations*

- Why DRAMSim3?
  - Interfaced “nicely” with GEM5
  - Very easy to change and look at source code
  - More accurate then GEM5’s DRAM modules
  - Very short compile time
- Why GEM5?
  - Very robust and easy to create workloads for
  - Very well documented on their website
  - Optimized compiler and well commented code

# *Simulations*

*(Additional details)*

- DRAM: DDR4_8Gb_x8_2400
- CPU: O3CPU (Basic Out-of-Order CPU)
- Default Cache size:
  - *L1d - 64kB*
  - *L1i – 32kB*
  - *L2 – 2MB*
  - *L3 – 16MB*

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

# *We got...*

# *Data*

*(lots of data)*

15 data files x 10 folders = 150 data files!

(Not even including testing data and text files!)

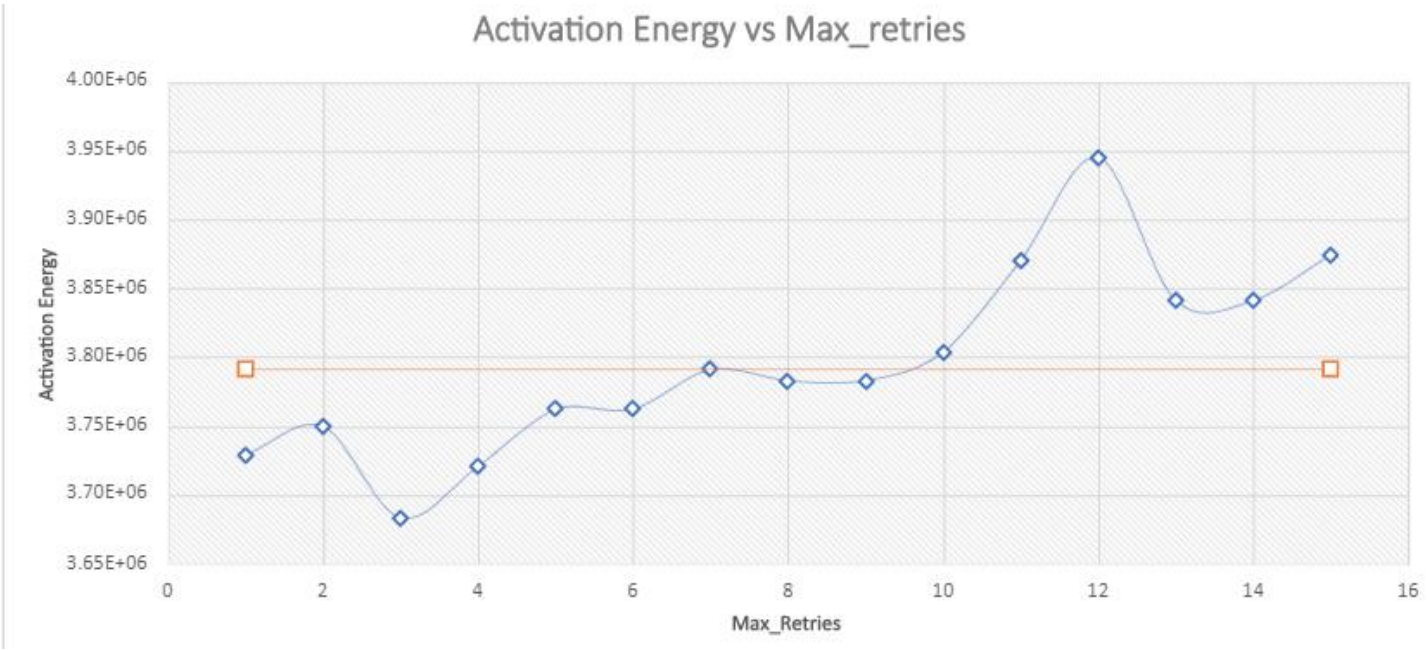
# *Results*

## Overview:

- Row hit rate did what we expected when cache was normal, less so for the smaller configs
- Energy also did what we expected for the initial setup
- The Baseline hit rate did some strange things, crossing over read/write as cache decreases

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

# *Activation Energy Results (Default cache)*



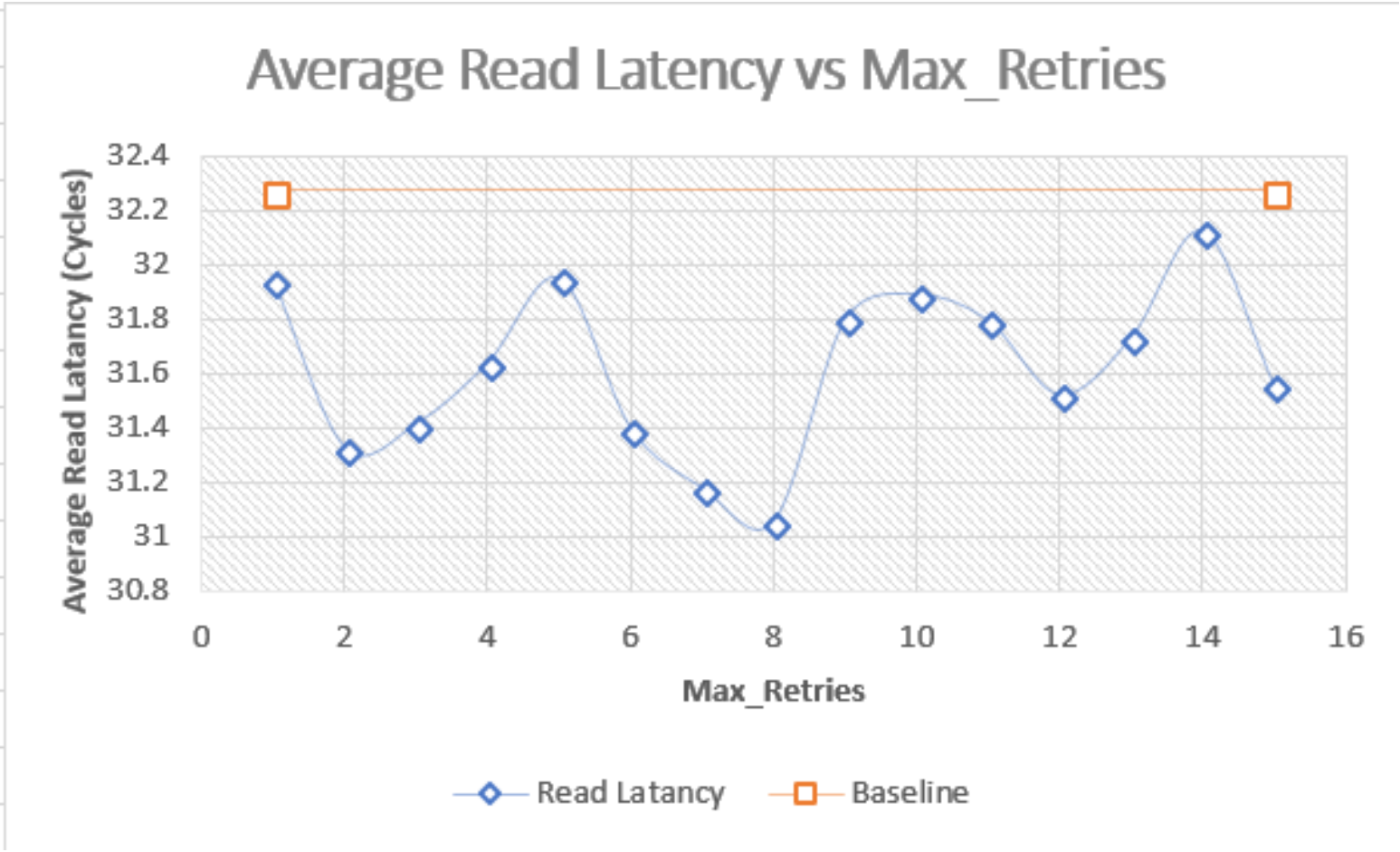

- The left shows our activation energy using ZORO as we change the max retries variable
- Notice an interesting trend!

Our best value at Max Retries = 3 was 2.93%!

Blue line – ZORO results
Orange line – Baseline results (unmodified)

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

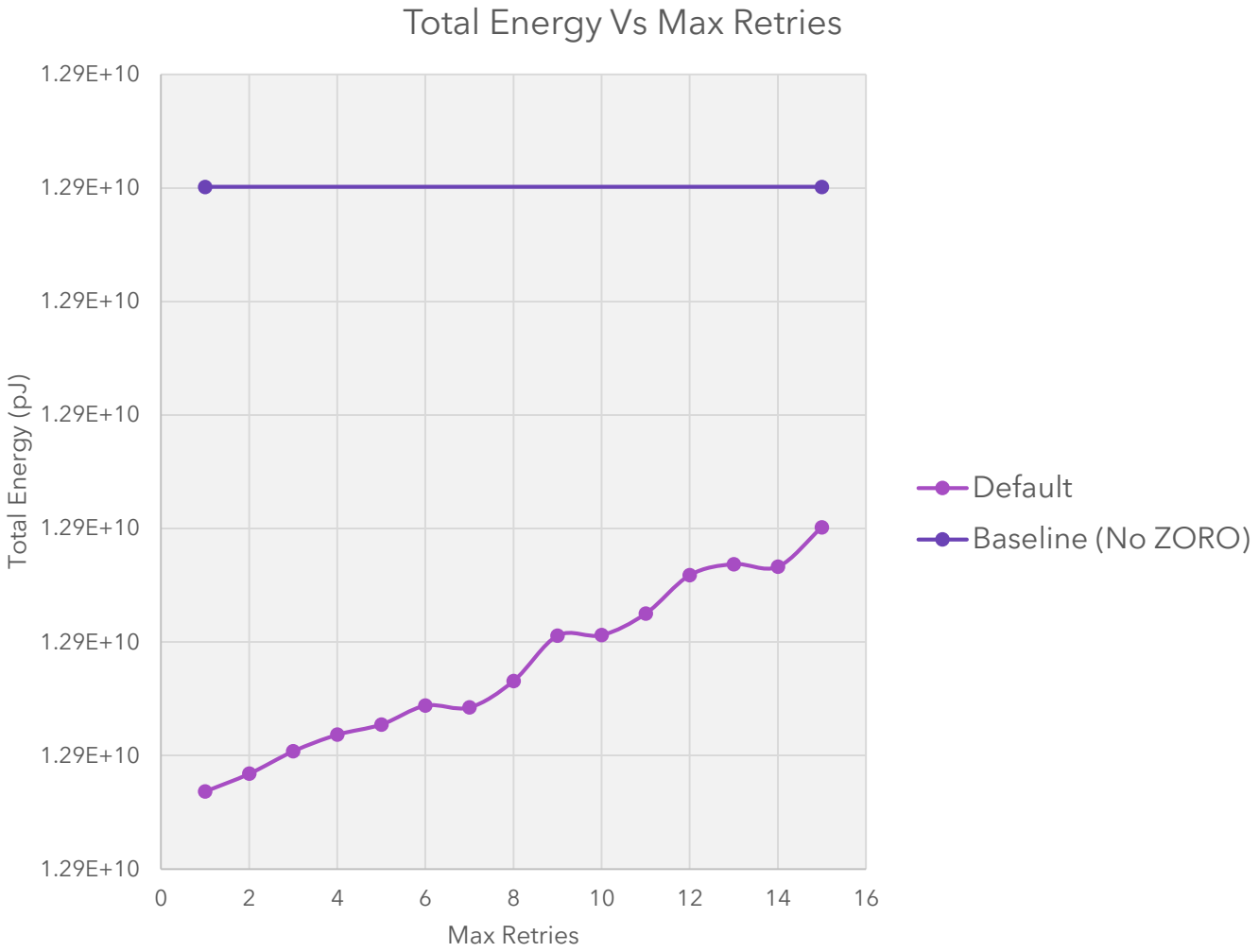
# *Average Read Latency Results (Default Cache)*




- Shows the overall read latency plotted against the ZORO max retries parameter
- Overall, ZORO seems to perform better for every value of max retries than the baseline

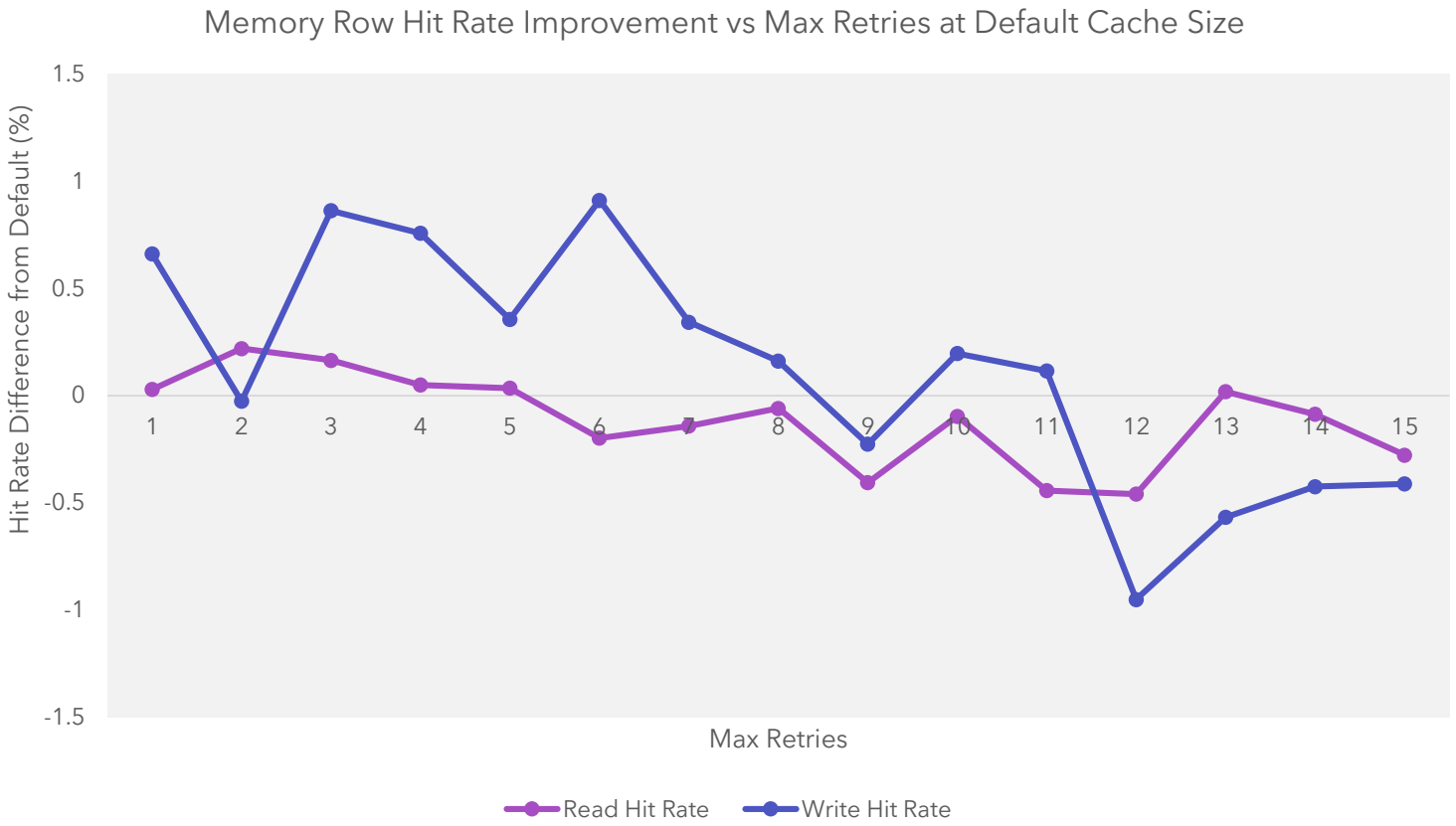
# *Total Energy*
## *(Default Cache)*

- Overall, less total energy than Baseline!



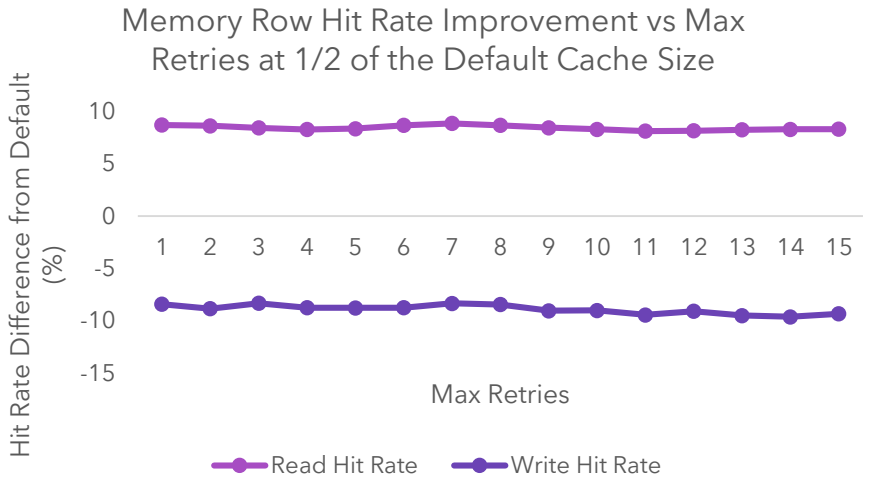
ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

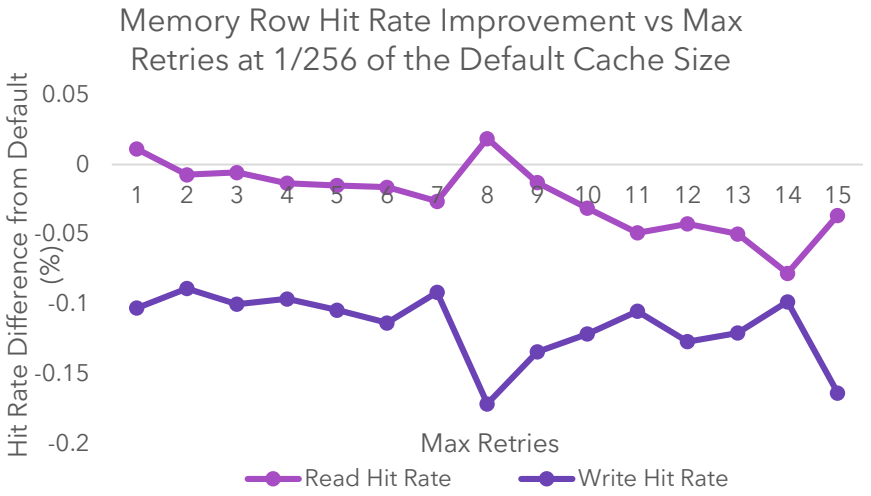
# *Memory Row Hit Rate Improvement Results (Default Cache)*

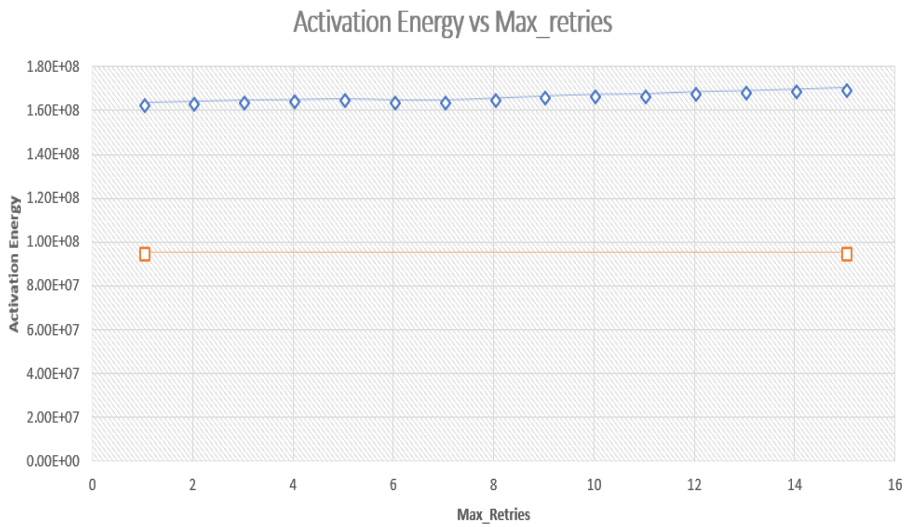


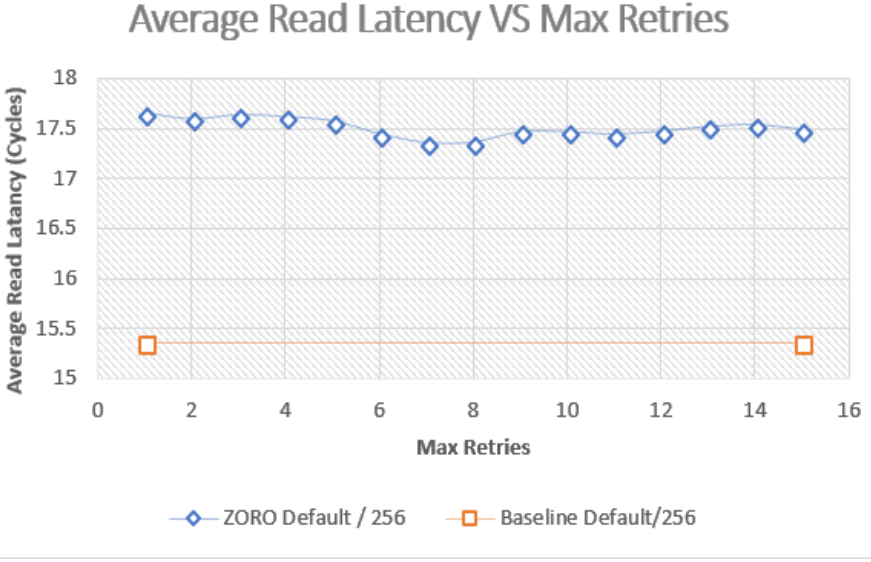
ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

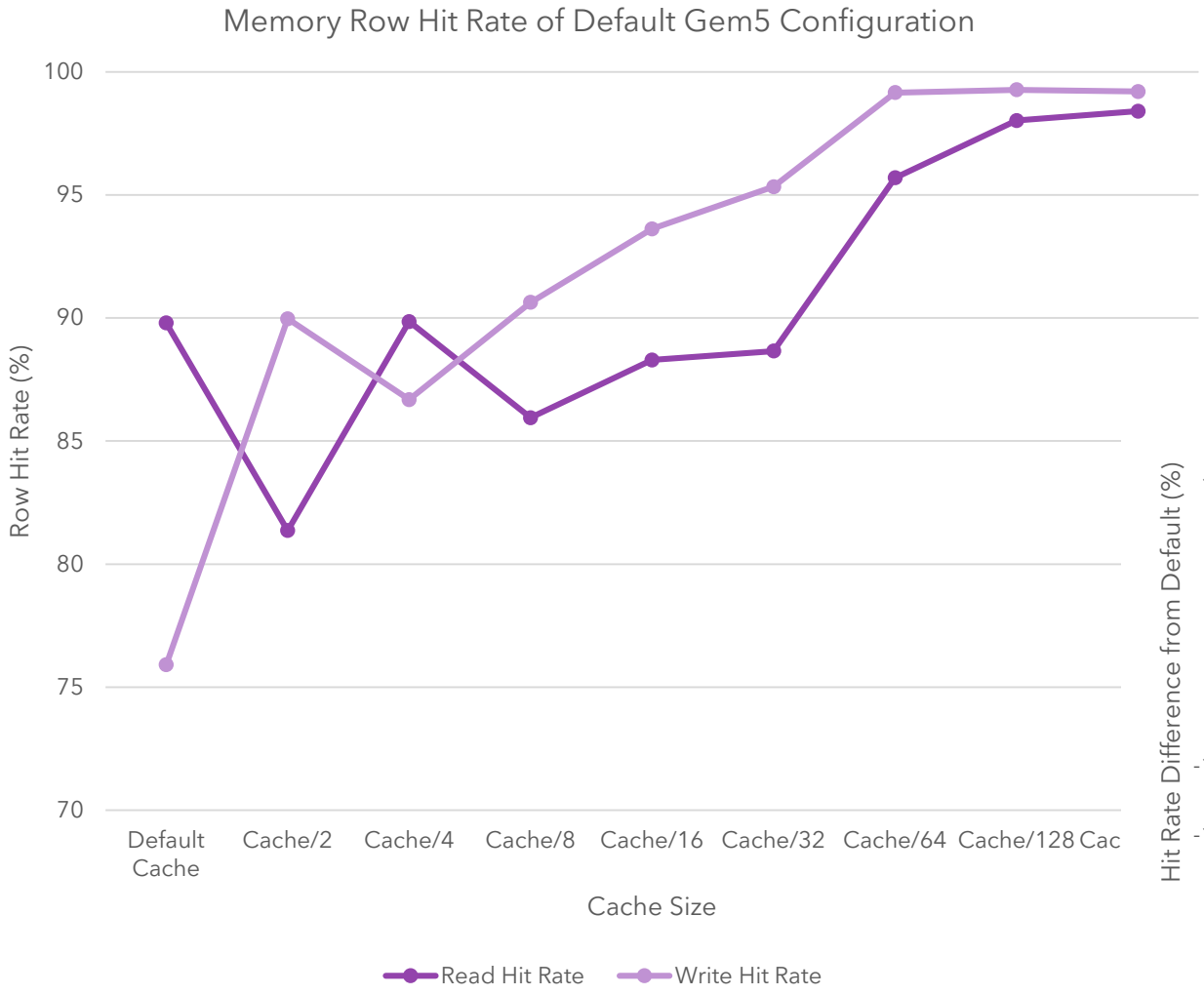
# What about smaller cache sizes?



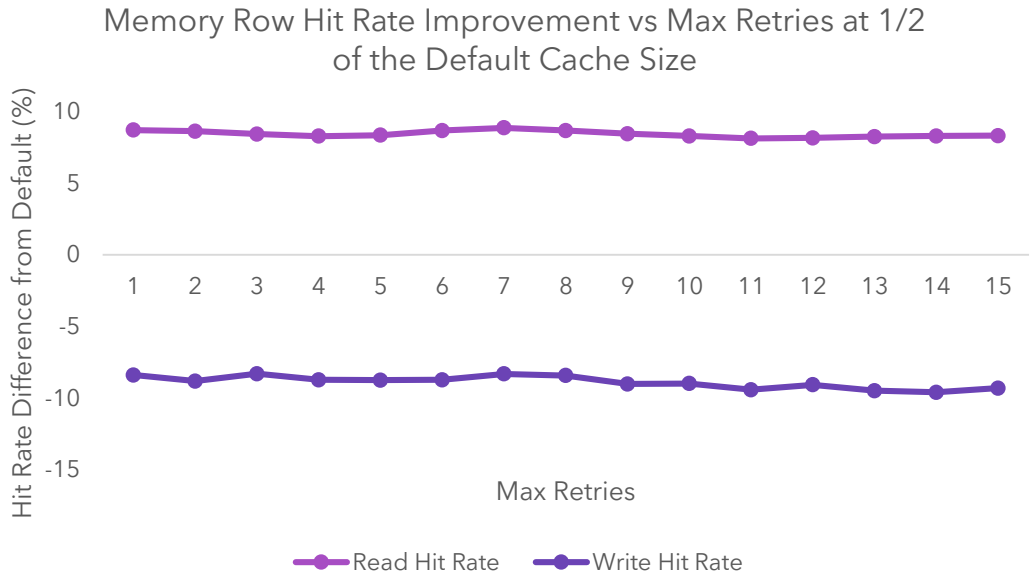






- Baseline Gem5 configuration behaved strangely as cache size decreased.
- Provides some indication as to why ZORO didn't perform as well




- About 7% improvement overall

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

# *Future Work*

- Modify ZORO to potentially improve performance with more complex row tracking
- Would have liked to have done more than a single benchmark
- Try different DRAM configurations, not just the single one we did
- Should also go up in cache size to determine if there are any changes
- We need to take into account CPU metrics instead of just the DRAM metrics
- Also would have liked to have a physical implementation of ZORO

# *Conclusions*

- ZORO takes advantage of FR-FCFS by quickly scheduling transactions that are already in an open row
- ZORO could possibly save energy while having similar performance or even providing a small increase
- ZORO does not work well for lower cache sizes because it ends up being less efficient than simple scheduling

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

# *Related Work*

- Hurr and C. Lin did some work trying to implement a history-based predictor as a scheduling mechanism for the memory controller

  *This differed from what we did because we aimed to change how the CPU schedules memory transactions*

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

# *References*


[1] Hur and C. Lin, "Adaptive History-Based Memory Schedulers for Modern Processors," in IEEE Micro, vol. 26, no. 1, pp. 22-29, Jan.-Feb. 2006, doi: 10.1109/MM.2006.1.

[2] J. Lowe-Power *et al.*, “The gem5 Simulator: Version 20.0+”, *arXiv [cs.AR]*. 2020.

[3] S. Li, Z. Yang, D. Reddy, A. Srivastava and B. Jacob, "DRAMsim3: a Cycle-accurate, Thermal-Capable DRAM Simulator," in IEEE Computer Architecture Letter

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

# *Questions?*

# *Where'd we get the data?*

- Hit rate:  # row hits / # commands
- Energy: Output from DRAMSim3
- Read latency: Output from DRAMSim3

## Number of  Read Commands vs Read Latency Category and Max Retries