# ECEC-621

### Project 5-B: Memory Controller

Instructor: Anup Das

## 1 Introduction to Memory Architecture

Every last-level cache (LLC) miss or eviction needs to be served by the DRAM. Figure 1 shows a sample architecture of a DRAM. In the beginning, all the DRAM requests either reads or writes are inserted in the transaction queue of a memory controller; the memory controller then schedules a request and sends it to DRAM via a memory channel that contains one rank and eight banks per rank. Each bank within a rank can operate individually, which referred to as *bank-level parallelism*. For example, suppose there are two read requests, R1, and R2 in the transaction queue, and it takes 50 cycles to serve a read request. If R1 and R2 both target to Bank 0, R2 has to wait for R1 to complete to get scheduled, so in total, it takes 100 cycles to serve R1 and R2, we call this situation (two consecutive requests targeting to the same bank) as a *bank-conflict*. However, if R1 aims Bank 0 while R2 targets Bank 1, both requests can be served simultaneously, which improves system performance significantly.

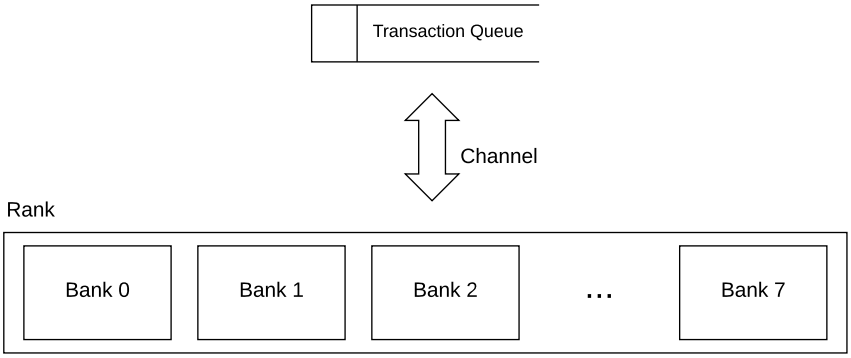


Figure 1: DRAM Architecture

## 2 Experiment

1. To get and run the framework for the project

```
$ git pull
$ cd Memory_Controller
$ make
$ ./Main 510.parset.trace.processed
```

2. The existing memory controller equips with an FCFS scheduler, which serves the request in the same order of insertion. Suppose there are three requests, R1 (target to Bank 0), R2 (target to Bank 0) and R3 (target to Bank 1) in the transaction queue, what is the total serving time with the FCFS scheduler (R1 $\rightarrow$ R2 $\rightarrow$ R3)?

3. What if there is an out-of-order memory controller that can prioritize request that targets to a free bank. Consider the same example; when R1 is issued, Bank 0 stays busy till R1's completion. In the next clock cycle, the memory controller is not able to schedule R2 because Bank 0 is busy; however, all other banks stay free, which means R3 can be scheduled out-of-order. What is the total serving time with the OoO scheduler (R1 $\rightarrow$ R3 $\rightarrow$ R2)?

4. Extra Credits: With an OoO memory controller, do you still need to consider data hazards?

# 3 Submission

1. Pick at least five traces for your evaluations.
`https://www.dropbox.com/sh/o9l12v0fksbdt9e/AACTBAv67FLIjSV9Cma_0_9Ua?dl=0`.

2. Implement a function that counts the number of bank-conflicts for an FCFS memory controller.

3. Implement the OoO memory controller and compare its performance against the FCFS memory controller.

4. Report your experiments, zip it with the source codes, and submit on Bblearn.