

ECE 485/585
Spring 2018
Final Project Description

Your team is responsible for the design and simulation of a memory controller capable serving the shared last level cache of a four core 3.2 GHz processor employing a single memory channel. The system uses a relaxed consistency (XC) model. The memory channel is populated by a single-ranked 8GB PC4-25600 DIMM (constructed with memory chips organized as x8 devices with a 1KB page size and 24-24-24 timing). There is no ECC.

Assume the DIMM is already initialized and all banks are in the pre-charged state. Do not model additive latency or CRC.

You can use any programming language (e.g. C, C++, Java) or hardware description language (e.g. SystemVerilog) you like to implement the simulation. Note that you are not doing a synthesizable design, though you must keep track of DRAM cycle counts.

Timing constraints

Your memory controller simulation must correctly schedule DRAM commands to comply with the following DDR-4 timing values. Unless otherwise indicated, all values are given in DRAM clock cycles.

Parameter	Value
tRC	76
tRAS	52
tRRD_L	6
tRRD_S	4
tRP	24
tRFC	350ns
CWL (tCWD)	20
tCAS (CL)	24
tRCD	24
tWR	20
tRTP	12
tCCD_L	8
tCCD_S	4
tBURST	4
tWTR_L	12
tWTR_S	4
REFI	7.8μs

The meaning and use of these parameters can be found in the course lecture slides and in datasheets for DDR4 devices such as this [Micron DDR4 device](#). Do not assume that this device is in any way similar in speed, size, organization, or timing to the device(s) in this assignment. The datasheet is provided only so that you can see how to interpret the parameters given above.

Input format

Your program should accept input trace files with the following format:

<time> <operation> <hex address>

where <time> is the time of the request in (absolute) CPU clock cycles from the beginning of the trace execution, <operation> is either FETCH, READ, or WRITE to indicate the type of memory request being made (instruction fetch, data read, data write, respectively), and <hex address> is the hexadecimal address being referenced by the operation. Fields are separate by one or more spaces or tabs. All addresses will be 8-byte aligned. The last level caches employ critical word first and early restart.

For example:

```
30 IFETCH 02000D5C0
30 IFETCH 01FF97000
31 IFETCH 01FF97080
32 READ    10FFFFFF00
40 WRITE   10FFFFFF80
```

Output format

Your program should generate as output a text file with a trace of all DRAM commands issues using the following format:

<time> <DRAM command>

where <time> is the time in (absolute) elapsed CPU cycles when the DRAM command is issued and <DRAM command> is one of the following:

```
ACT <bank group> <bank> <row>
PRE <bank group> <bank>
RD <bank group> <bank> <column>
RDAP <bank group> <bank> <column>
WR <bank group> <bank> <column>
WRAP <bank group> <bank> <column>
REF
```

and <bank group>, <bank>, <row>, and <column> are non-negative hexadecimal values. Note that you may not use RD or WR with auto precharge. All fields are separated by one or more spaces or tabs.

For example:

```
100  PRE 0 0
200  ACT 0 0 03FF
300  RD 0 0 EF
```

Grading

The project is worth 100 points. Your grade will be based upon:

- External specification
- Completeness of the solution (adherence to the requirements above)
- Correctness of the solution
- Quality and readability of the project report (e.g. specifications, design decisions)
- Validity of design decisions
- Quality of implementation
- Structure and clarity of design
- Readability
- Maintainability
- Testing
- Presentation of results

There are a variety of policies a memory controller can employ in an attempt to improve performance (usually bandwidth, though sometimes worst-case latency or power).

The highest possible score you can receive depends upon the complexity of the policies that you implement. The table below summarizes them.

In-order, no access scheduling, closed page policy	80
In-order, no access scheduling, open page policy	90
Access scheduling, possibly out of order	100

Note that your score can be lower (possibly significantly) if you have bugs: incorrect results, DRAM timing violations, etc. or your code is poorly written (difficult to understand, unreliable, difficult to maintain, hard to re-use).

You can receive extra credit for implementing refresh operations, further exploration of the design space (e.g. implementing and measuring performance of different out-of-order scheduling policies, request buffer size).

Your simulation does not need to model or support mode register programming, low power modes, or additive latency.

Scheduling policies

The in-order scheduling, closed page policy issues DRAM commands for memory requests in the order they are received and explicitly closes each page after every column access. It must maintain a buffer of pending memory requests because they may arrive faster than DRAM commands can be issued. When the buffer is full, no further memory requests can be acknowledged until a pending memory operation completes and the request is removed from the queue.

The in-order scheduling, open page policy also processes requests in order. However, it doesn't explicitly close a page unless the queue is non-empty and the next in-order reference is to a different row in the same bank.

First ready, first access scheduling still processes requests in the order received. However, it can schedule individual DRAM commands from more than one in-flight request so that, for example, after issuing an activate command for one memory reference it can issue the activate command to another bank for another request before issuing the read command for the earlier reference.

Out-of-order policies can process memory requests out of order to optimize the DRAM command schedule. There are many possibilities. One example might be to schedule a memory read ahead of one that arrived earlier if it's to the currently open row in the same bank.

Academic honesty

You are free to consult the literature for ideas provided you cite all sources in your final report/presentation. You cannot make use of existing code or have anyone outside your team write code for you. Doing so will result in a code of conduct violation complaint and a score of 0 on the assignment.

Competition

All teams doing more than the minimum in-order, no access scheduling (closed or open page policy) can participate in a competition to achieve the best bandwidth on a benchmark that I will provide.

Teams with top three performance will be recognized and earn extra credit. The team with the top-performing controller will win a small prize and be able to smugly mock their unworthy competitors.

Presentation and demonstration

Each team will demo their final project in a 30 minute time slot during finals week. You will provide a brief written report of your project as well as all documentation, testcases, and source code along with either a makefile (with default target) or README file with instructions for building and running your project. These should be packaged in a single zip file and uploaded to D2L 24 hours prior to your demo slot.

Verification (testing) is important. Be sure to include a description of your test strategy and testcases in your report.

Checkpoints

Because of the complexity of the project it's important that you get started early and make steady progress throughout the quarter. I've devised a series of early and intermediate checkpoints to help you with this. It has the added advantage that any major misunderstandings can be caught early when you can still correct them.

The dates associated with each checkpoint will be shown on the course web site Syllabus page. By the end of the date, your team will need to demonstrate to the course TA or instructor the capability indicated.

1. Have working code to read and parse (and show via debugging statements) the input trace file format. Demonstrate that debugging information can be turned on/off at either runtime or compile time.
2. Diagram showing how address bits are mapped to required fields (e.g. row, column, bank, bank group).
3. Have working code to insert items into the queue, stub to remove items from queue, and maintain correct (simulated) time. What information needs to be maintained in the queue?
4. Demonstrate code generates correct output format for DRAM commands.