ECE585 FINAL PROJECT REPORT
GROUP 15

SIMULATION of a MEMORY CONTROLLER

1) **RITVIK TIWARI**
PSU ID: 915097622
Email ID: ritviktiwari@pdx.edu

2) **MEHUL SHAH**
PSU ID: 964893951
Email ID: mehul@pdx.edu

3) **RAMAA POTNIS**
PSU ID: 937862463
Email ID: rgp2@pdex.edu

4) **SHIVANI PALKAR**
PSU ID: 920177877
Email ID: spalkar@pdx.edu

Given Specifications:

- Four core 3.2GHZ processor with a single memory channel
- memory channel is populated by a single-ranked 8GB PC4-25600 DIMM.
  (Constructed with memory chips organized as x8 devices with a 2KB page size and 24-24-24 timing)
- All banks are initially in the PRECHARGED state.

Design Considerations:

- The given bandwidth of the DIMM is 25600Mbps
  Single clock cycle = 1/(25600/16) = 0.625ns
- The controller is implemented taking into consideration the CPU clock cycles where:
  1 DRAM cycle = 2 CPU clock cycles.
- Programming of the memory controller is done using C++.

Bit Mapping:

The following diagram represents what and how the CPU addresses are mapped to

32 18	17 10	9 8	7 6	5 4 3	2 1 0
Row	High column	Bank	Bank group	Low column	Byte Order

- Lower 3 LSB bits used to select byte order.
- To access burst of 8, low column bits are used which are 3 bits.
- Next to utilize the bank parallelism, we have to make sure that the next reference goes to a different open bank in a parallel bank group; so considering 4 bank groups, we use the next 2 bits for bank groups.
- Then 2 bits to select 1 out of 4 banks from the above selected bank group.
- Assuming 11 total columns and using lower column bits for column selection, we use the remaining 8 bits.
- The 15 bits are for accessing 256 rows/banks next.

Brief Description:

- The memory controller is servicing memory requests from input trace files which contain one of the following parameters:
  - a READ
  - a WRITE
  - a FETCH

- This trace file is fed to a parser and the output of this is to be added to a queue which can hold up to 16 requests at a time.
- The commands enter the queue serially and are serviced in a First Come First Served manner.
- Once the queue is full, the next incoming request will be stalled until another request is evicted from the queue.
- We are implementing in-order scheduling without access.
- We have designed a Bank Status Register to monitor the following:
  - -whether the bank is open or close
  - -whether the row activated in the respective banks
  - -which one of the following commands is to be performed
    - PRECHARGE -To close an open row so as to access a new row.
    - ACTIVATE - To activate a new row
    - READ
    - WRITE
- The BSR gets updated as and when a new request occurs and sets the status accordingly.
- In order to keep track of time elapsed since the issuing of the last command , there are counters kept.

Testing Strategies:

To ensure that the design is stable and can withstand all the various cases, there are certain test cases which were carried out. These tests being performed comprise a few of the possible cases that can happen.

- Following could be the possibilities of where the next memory reference would occur:
  - o To the same bank group, same bank, same row and a *different* column
  - o To the same bank group, same bank, *different* row and a column in that row
  - o To the same bank group, *different* bank and a row and a column in that bank
  - o To a *different* bank group, and a bank and a row and a column in that bank group

- For each access, certain timing constraints are obeyed to maintain a proper controller function.

Consider the following cases. The memory requests that are mentioned below come sequentially.
1st request**:** time = 30 and for a READ command, the address being accessed is
0X0000000000

- Row being accessed = 0 , (R_0)
- Column being accessed = Low column 0 ,High column 0
- Bank being accessed = 0 ,(B_0)
- Bank group being accessed = 0, (BG_0)

The time needed to satisfy timing constraints = 30 + time needed to read
= 30 + t_{RCD}+ t_{CAS} + t_{Burst}

2nd reference: time = 36, and the command being issued is a READ, thus the address is:
0X000000400

- Row being accessed = 0 , (R_0)
- Column being accessed = Low column 0, high column 1
- Bank being accessed = 0 ,(B_0)
- Bank group being accessed = 0, (BG_0)

The time needed to satisfy timing constraints = 36+ t_{CAS}+ t_{BURST}

3rd reference**:** time = 42, and command is a WRITE, address being addressed is
0X000100400H

- Row = 4, (R_4)
- Column = Low column 0, high column 1
- Bank = 0
- Bank group = 0

The time needed to satisfy timing constraints = 42+ t_{RTP}+t_{RP}+t_{RCD}+t_{CWD}+t_{BURST}

4th reference : time = 58, and the command is a WRITE, address is 0X000100400

- Row = 4, (R_4)
- Column = Low column 0, high column 1
- Bank = 0
- Bank group = 0

The time needed to satisfy timing constraints = 58+ t_{CWD}+t_{BURST}

5th reference: time = 60, for a READ command to address 0X0000C0500

- Row = 3
- Column = Low column 0, high column 1
- Bank = 1
- Bank group = 0

The time needed to satisfy timing constraints = 60+t_{RCD}+t_{CAS}+t_{BURST}

6th reference: time = 100, for a FETCH command to address 0X000281600H

- Row = 10
- Column = Low column 0, high column 1
- Bank = 2
- Bank group = 0

The time needed to satisfy timing constraints = 100+ t_{RCD}+t_{CAS}++t_{BURST}