ECE585_Team3

Specification Document for

Team 3's final project for ECE 485/585 Portland State University, Fall 2021

Braden Harwood, Stephen Short, Michael Weston, Drew Seidel

**Information sourced from Dr. Faust's Final Project Description and Group Discussion*

CPU Information

4 Core, 3.2 GHz Quad Core CPU Clock = 3200 MHz, CPU Period = 0.3125 ns Single Memory Channel

Memory DIMM Information

8 GB PC4-25600, 24-24-24 Timing - 64 Gb Capacity - 1 rank of 8 Chips of 1 Gb x 8 - No ECC - 2 KB Page Size - 24-24-24 Timing - DIMM Clock = 1600 MHz - DIMM Period = 0.625 ns (2x the CPU Period)

Timing Constraint Definitions

Parameter	Description	Formula	Default Value	Values (DIMM Cycles)	Values (CPU Cycles)
tRC	Time between successive row accesses to different rows	tRC = tRAS + tRP	76	76	152
tRAS	Time between ACT command and end of restoration of data in DRAM array	Row Address Strobe	52	52	104
tRRD_L	Row-to-row delay, for Activates to the SAME bank group		6	6	12
tRRD_S	Row-to-row delay, for Activates to the DIFFERENT bank group		4	4	8
tRP	Time to pre-charge DRAM array in preparation for another row access	Row Precharge Delay	24	24	48
tRFC	Refresh cycle time/Recovery Delay. Amount of time to complete a refresh command		350ns	560	1120
CWL (tCWD)	CAS Write Latency is the delay between WRITE and the availability of the first bit of input data		20	20	40
tCAS (CL)	Time between READ command and first data valid	Column Access Time	24	24	48
tRCD	Minimum time between an ACT command and READ/WRITE command	Row Column Delay (RAS/CAS Delay)	24	24	48
tWR	Write recovery time, minimum interval between the end of write data burst and the start of a precharge command		20	20	40
tRTP	Read to Precharge Time		12	12	24
tCCD_L	Column-to-column delay, for accesses to the SAME bank group		8	8	16
tCCD_S	Column-to-column delay, for accesses to DIFFERENT bank group		4	4	8
tBURST	Data burst duration		4	4	8
tWTR_L	Write to Read turn around time, specifies how much time must elapse from the last bit of the write data burst prior to a read command being issued on the same bank group		12	12	24

Parameter	Description	Formula	Default Value	Values (DIMM Cycles)	Values (CPU Cycles)
tWTR_S	Write to Read turn around time, specifies how much time must elapse from the last bit of the write data burst prior to a read command being issued on a different bank group		4	4	8
REFI	The required average interval of REFRESH commands needed		7.8us	12480	24960

Memory Controller Requirements

- Exploit Bank Parallelism
- Open Page Policy
- Assume DIMM is already initialized and all banks are in pre-charged state
- Queue of 16 Outstanding Memory Requests
- Last Level Caches employ critical word first
- Last Level Caches employ early restart
- We may not use RD or WR with auto pre-charge

Input Format

Input filename will be provided by the user at runtime

Format:

- \<time> \<operation> \
    - Split with spaces and/or tabs
    - \<time> represents the time of the request in absolute CPU clock cycles from the beginning of the trace execution
    - \<operation> represents the type of memory request being made
    - 0 = Data Read
    - 1 = Data Write
    - 2 = Instruction Fetch
    - \ represents the hexadecimal address being referenced the the operation
    - All addresses are 8-byte aligned (3 least significant bits are 0's)

Example Input:

- 30 2 0x01FF97000
- 31 2 0x01FF97080
- 32 0 0x10FFFFF00
- 40 1 0x10FFFFF80

Output Format

Program will generate a text file as output of all DRAM commands Format: - \<time> \ - \<time> represents the time in absolute elapsed CPU cycles when the DRAM command is issued. - \ is one of the following: - ACT \ \<bank> \<row> - PRE \ \<bank> - RD \ \<bank> \<column> - WR \ \<bank> \<column> - REF - \, \<bank>, \<row>, and \<column> are hexadecimal values - All fields are separated by one or more spaces or tabs - Time should be displayed in a long fixed-width field - Commands should be displayed in fixed-width fields as well

Example Output:

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

Scheduling Policies

- Basic implementation must use in-order scheduling
- Advanced implementations must include a run-time switch to enable/disable them
- In-order scheduling processes requests in order
- First ready, first access scheduling still processes requests in order, but does allow the scheduling of individual DRAM commands from more than one request.
    - Example: After issuing an activate command for one memory reference, it can issue the activate command to another bank/bank group for another request before issuing the read command for the earlier reference.
- Out-of-order scheduling can process requests out of order to optimize the DRAM command schedule.
    - Example: Putting a later request first because it is accessing the same row that is already open before another request which would require the row to be closed.
    - Additional ideas available.

Memory Mapping (Draft)

Values needed: # Rows, # Columns

# Columns

- Page Size known: 2 KB
- Page Size = 2^ColBits * (Internal Access Size/8)
    - 2 KB = 16 Kb = 2^ColBits * (64/8)
    - 16 Kb = 2^ColBits * 8
    - 2 Kb = 2^ColBits
    - ColBits = 11 bits
    - Columns = 2^11 = 2K Columns

# Rows

- Number of Banks, Number of Bank Groups, and Number of Columns known
- 1 Gb x 8
    - 2 Bits for Bank Groups
    - 2 Bits for Banks
    - Rows determined by Capacity and Columns
    - 1 Gb x 8 Organization for each chip (8 chips total)
    - 2^30 bits to represent 1 Gb
    - Capacity / (# Columns * # Banks * # Bank Groups) = 2^30 * (2^-11) * (2^-2) * (2^-2) = 2^15
    - Rows = 2^15 = 32 K Rows

# Bytes from Burst

- 8 Bytes from Burst
- 3 bits needed to index which byte is desired, but is not passed on to the DIMM WORK IN PROGRESS

Total Bits

- Rows = 2^15
- Columns = 2^11
- Bank Select = 2^2
- Bank Group Select = 2^2
- Byte Select = 2^3
- Total number of bits = 33

Proposed Organization

MSB: Rows > Banks > Columns[10:3] > Bank Groups > Columns[2:0] > Byte Select -Consider Row/Bank/UpperColumns order with open page policy idea

Improved Organization

MSB: Rows[14:0] > Columns[10:3] > Banks[1:0] > Bank Groups[1:0] > Columns[2:0] > Byte Select[2:0]

Reasoning

The reasons for this choice are: - Spatial locality - sequential code and data is likely to be fetched and accessed. - Bank parallelism allows access to open pages while performing a precharge or activation on another. It is most efficient to spread these commands across bank groups, rather than banks within the same bank group when possible.

Runtime Instructions

Download this Github repository and unzip it in an empty directory of your choice on your system or clone it.

Access the directory in your terminal or compiler of your choice. Run 'make help' to view all options.