MICROPROCESSOR SYSTEM DESIGN

DDR 5 MEMORY CONTROLLER

SCHEDULER PORTION

CLOSED PAGE POLICY

FINAL PROJECT REPORT

SUBMITTED BY:

Alaina Anand Nekuri (PSU ID: 959604917)

alainaa@pdx.edu

Dhruthiee Nelli (PSU ID: 950321302)

dnelli@pdx.edu

Mohammeed Abbas Shaik (PSU ID: 950376592)

mohammee@pdx.edu

Nivedita Boyina (PSU ID: 958615179)

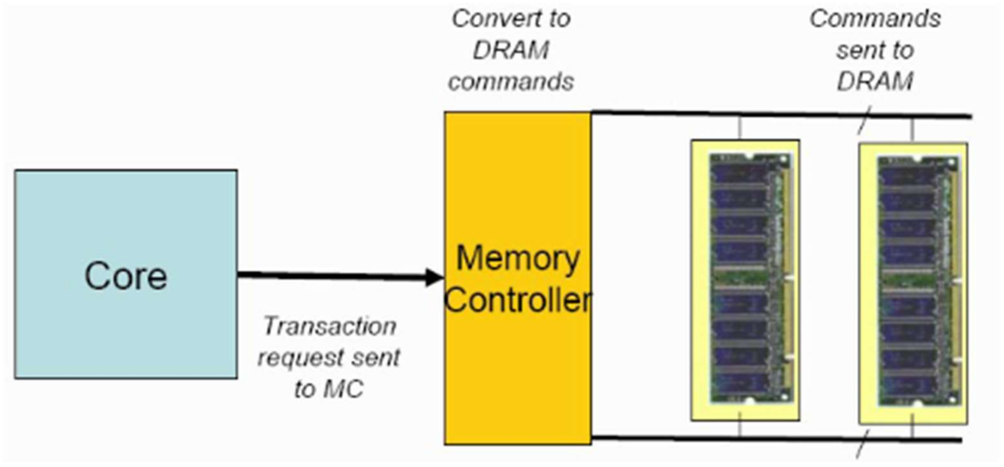
nivedita@pdx.edu

DDR5 SDRAM:

DDR5 SDRAM is a type of synchronous dynamic random access memory. Compared to DDR4, DDR 5 is designed to have reduced power consumption, while doubling bandwidth.

- Unlike DDR4 and DDR3, DDR 5 has 2 channels which improve its ability to provide increased bandwidth, parallelize data transfers and enhance overall system performance.
- In DDR5 SDRAM, ECC is integrated on-die, providing on-chip error correction capabilities that enhance the reliability of memory subsystem by automatically identifying and rectifying errors in stored data.
- DDR5 typically offers higher speeds and operates at a lower voltage than its predecessors, contributing to energy efficiency.
- DDR5 supports higher memory densities, enabling large memory capacities per module.

The DDR5 memory scheduler lies in the memory controller, which plays the vital role of prioritizing the memory access requests which might probably be from different cores, management of banks and channels and synchronization of the command timings. It optimizes data access thereby reducing latency.



Project Description:

This project Is simulation of the scheduler portion of the memory controller which is capable of serving a 12 core 4.8GHz processor employing a single 16 GB PC-5 38400 DIMM. This system uses relaxed consistency model which allows for some reordering or delay of memory operations for performance optimization. Here, the controller uses the two channels of the DIMM independently. We are implementing this project on single channel.

The DIMM is made up of memory chips which are organized as x8 devices with 1 KB page size and 40-39-39-76 timing. All these devices have 8 bank groups with each having 4 banks and there is no ECC here for this DDR5.

It is also said that the DIMM uses 1N mode for commands requiring two cycles. Which means, though the memory commands that naturally require 2 cycles, they are set to operate in more efficient mode by completing in a single clock cycle.

Implementation:

We are implementing the scheduler portion of the memory controller in closed page policy. In closed page policy, we assume that all the banks are precharged initially. Now, after performing the read or write operation, we immediately close the page. As there will be leakage (or) discharge of the charge stored in the capacitors even if it is left unused, we need to Precharge the bank and wait for the Row to Precharge delay and close the page and then issue the Activate command again.

We implemented the memory controller using system Verilog. The given input trace will be accessed only if Debug is enabled. If debug is ON, then the given input trace file will be parsed and the contents are read line by line and pushed into the queue with respect to the CPU clock cycles. The parsed data will be popped out of the queue based on the DIMM clock cycles and then the commands are executed one after the other by obeying the timing constraints.

Address Mapping:

16GB DIMM using x8 devices

- Each channel needs 4 devices/rank to provide 32 bits (4 bytes) at a time
- Each channel has 16GB/2 = 8 GB
- 8 GB/4 bytes = 2G addresses x 4 bytes
- Each device is x8, so each device is 2G x8 = 16Gb
- There are 32 banks, so each bank provides 16 Gb/32 = $2^{34}/2^{5} = 2^{29}$ bits
- The page size is given as 1 KB = 8 Kb/ row = 2^{13} bits
- So, there are $2^{29}/2^{13} = 2^{16}$ = 64k rows
- Each page is 8 Kb/8 bits/column = 2^{10} = 1024 columns
- Each burst provides 16 chunks x 8 bits= $2^{4} \times 2^{3} = 2^{7}$ = 128 bits
- 8 Kb/128 = $2^{13}/ 2^{7} = 2^{6}$, so a bank is internally 64Kx64x128

Address bits

- 1 channel bit
- 2 Bank address bits because 4 banks/bank group
- 3 Bank group bits because 8 bank groups
- 16 row bits because 64k rows/bank
- 10 column bits because 1K columns
- Total of 32 bits

But each chunk from DIMM is comprised of 4 bytes, So,

- 2 byte select bits (which are not sent to DIMM)
- Total of 34 bits of (byte) address
- Check work : 2^{34} = 16 GB

Mapping

33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row																High Column						Bank		Bank Group			CHANNEL	Low Column				Byte Select	

Timing Parameters:

Parameter	Description	Clock cycles
tRC	Delay between the ACT commands	115
tRAS	Time between ACT command and end of restoration of data	76
tRRD_L	ACT to ACT between different banks on same bank group	12
tRRD_S	ACT to ACT between different banks on different bank groups	8
tRP	PRE to ACT	39
tRFC	Refresh cycle time i.e., time for a single refresh command	295 ns
tCWD	WR to Data Out	38
tCAS(CL)	RD to Data Out	40
tRCD	ACT to RD	39
tWR	Write recovery time i.e., delay between WR and precharging it (tRAS – tRCD)	30
tRTP	RD to PRE delay	18
tCCD_L	RD to RD for same bank, same bank group	12
tCCD_S	RD to RD for different bank group	8
tCCD_L_WR	WR to WR for same bank, same bank group	48
tCCD_S_WR	WR to WR for different bank group	8
tBURST	Time to read the BURST data completely	8
tCCD_L_RTW	RD to WR for same bank groups	16
tCCD_S_RTW	RD to WR for different bank groups	16
tCCD_L_WTR	WR to RD delay for same bank, same bank group	70
tCCD_S_WTR	WR to RD delay for different bank group	52

Test plan:

Case:1 Consecutive Read Operations in same Bank, same Bank group, same Row, different Column

Input:

```
00000000 1 0 0x009FF6917

00000001 3 2 0x009FC6917

00000002 5 0 0x009FF1917
```

Output:

```
0002  0 ACT0 2 2 027F

0004  0 ACT1 2  2 027F

0082  0 RD0   2  2 365

0084  0 RD1   2  2365

0180  0 PRE    2  2

0258  0 ACT0  2  2  027F

0260  0 ACT1  2  2  027F

0338  0 RD0  2  2  065

0340  0 RD1  2  2  065

0436  0 PRE  2  2

0514  0 ACT0  2  2  027F

0516  0 ACT1  2  2  027F

0594  0 RD0  2  2  315

0596  0 RD1  2  2  315

0692  0 PRE  2  2
```

<u>Case:2 Consecutive Read operations in Same Bank group, same bank, different row, different column</u>

<u>Input:</u>

000011   0 0 0x0002C2D95

000012 11 0 0x000AD2D95

<u>Output:</u>

12 0 ACT0 3 3 000B

14 0 ACT1 3 3 000B

92 0 RD0 3 3 025

94 0 RD1 3 3 025

190 0 PRE 3 3

268 0 ACT0 3 3 002B

270 0 ACT1 3 3 002B

348 0 RD0 3 3 125

350 0 RD1 3 3 125

446 0 PRE 3 3

<u>Case:3 Consecutive Read operations in Same Bank group, Different bank, Different Row, Different Column</u>

<u>Input:</u>

000001 10 0 0x0002C2D95

000004 4 0 0x0003C3995

<u>Output:</u>

2 0 ACT0 3 3 000B

4 0 ACT1 3 3 000B

82 0 RD0 3 3 025

84 0 RD1 3 3 025

180 0 PRE 3 3

258 0 ACT0 3 2 000F

260 0 ACT1 3 2 000F

338 0 RD0 3 2 035

340 0 RD1 3 2 035

436 0 PRE 3 2

<u>Case:4 First Write Next Read Operations in Different Bank Group, Different Bank, Different Row, Different Column</u>

<u>Input:</u>

000000001 1 1 001383F8A

000000002 2 2 00438920A

<u>Output:</u>

   2 0 ACT0 7 3 004E

   4 0 ACT1 7 3 004E

 82 0 WR0 7 3 032

 84 0 WR1 7 3 032

176 0 PRE 7 3

254 0 ACT0 4 0 010E

256 0 ACT1 4 0 010E

334 0 RD0 4 0 092

336 0 RD1 4 0 092

432 0 PRE 4 0

<u>Case:5 First Read Next Write operations in Same Bank, Same Bank Group, Same Row, Different Column</u>

<u>Input:</u>

000000018 0 0 0001E0FA5

000000023 1 1 0001E8FA5

Output:

```
20 0 ACT0 7 3 0007

22 0 ACT1 7 3 0007

100 0 RD0 7 3 209

102 0 RD1 7 3 209

198 0 PRE 7 3

276 0 ACT0 7 3 0007

278 0 ACT1 7 3 0007

356 0 WR0 7 3 289

358 0 WR1 7 3 289

450 0 PRE 7 3
```

Case:6 Consecutive Write operations on Same Bank Group, Different bank, Different row, Different columns

Input:

```
000000041 8 1 0004805B5

000000049 3 1 0004409B5
```

Output:

```
42 0 ACT0 3 1 0012

44 0 ACT1 3 1 0012

122 0 WR0  3 1 00D

124 0 WR1  3 1 00D

216 0 PRE  3 1

294 0 ACT0 3 2 0011

296 0 ACT1 3 2 0011

374 0 WR0  3 2 00C

376 0 WR1  3 2 00C

468 0 PRE  3 2
```

Case:7 Consecutive Write Operations on Same Bank Group, Same Bank, Different Row, Different Column

Input:

```
000000048 7 1 3FFFE1485

000000052 4 1 1FFFF1485
```

Output:

```
50 0 ACT0 1 1 FFFF

52 0 ACT1 1 1 FFFF

130 0 WR0  1 1 211

132 0 WR1  1 1 211

224 0 PRE  1 1

302 0 ACT0 1 1 7FFF

304 0 ACT1 1 1 7FFF

382 0 WR0  1 1 311

384 0 WR1  1 1 311

476 0 PRE  1 1
```

Case:8 Consecutive Writes in Different Bank, Different Bank Groups, Different Rows, Different Columns

Input:

```
000000000 0 1 00074868F

000000001 1 1 000728C8F
```

Output:

```
00000002 0 ACT0 1 5 001D

00000004 0 ACT1 1 5 001D

00000082 0 WR0 1 5 083

00000084 0 WR1 1 5 083

00000176 0 PRE 1 5

00000254 0 ACT0 3 1 001C
```

00000256 0 ACT1 3 1 001C

00000334 0 WR0 3 1 283

00000336 0 WR1 3 1 283

00000428 0 PRE 3 1

Case:9 First Write, Next Read in Same Bank, Same bank Group, Same Row, Different Column

Input:

000000221 1 1 0FFF98501

000000228 4 0 0FFF9C501

Output:

222 0 ACT0 1 2 3FFD

224 0 ACT1 1 2 3FFD

302 0 WR0 1 2 180

304 0 WR1 1 2 180

396 0 PRE 1 2

474 0 ACT0 1 2 3FFD

476 0 ACT1 1 2 3FFD

554 0 RD0 1 2 1C0

556 0 RD1 1 2 1C0

652 0 PRE 1 2

Case:10 First Read, Next Write in Different Bank Group, Different bank, Different Row, Different Column

Input:

00000032 3 0 000198501

00000038 9 1 0001DCD81

Output:

34 0 ACT0 1 2 0006

36 0 ACT1 1 2 0006

```
114 0 RD0 1 2 180

116 0 RD1 1 2 180

212 0 PRE 1 2

290 0 ACT0 3 3 0007

292 0 ACT1 3 3 0007

370 0 WR0 3 3 1C0

372 0 WR1 3 3 1C0

464 0 PRE 3 3
```