**ECE 585**

**Final Project Report**

GROUP-24

Seggari Koushik

Sigicherla Srikanth

R. Jitendra Prasad

C. Venkat Sahith Reddy

**Introduction:**

We used closed page policy, no bank parallelism, and implemented the code in System Verilog to design the scheduler part of memory controller. In the closed-page policy, the DRAM page is quickly closed following a read or write operation, with the issuance of a pre-charge command. Unlike the open-page policy, the closed-page policy assumes that the row/page is deactivated immediately after data access. If the next incoming instruction maps to the same row/page that was previously accessed, the closed-page policy requires reactivation of the row/page through the issuance of an Activate (ACT) command followed by the appropriate Read or Write command.

**Specifications of the design:**

The scheduler portion of memory controller is capable of handling 12-core 4.8 GHz processor employing a single 16GB PC5-38400 DIMM. Two channels of the DIMM are used. Chips are organized as x8 devices with a 1KB page. These devices have 8 bank groups of 4 banks each.

**Implementation of the design:**

Mapping of bits:

The capacity of DIMM is 16 GB = 234 => we get 34 total address bits.

Page size = 1 KB = 210 => 10 column bits.

Bank groups = 8 = 23 => 3 bits are assigned to bank groups.

Banks = 4 = 22 => 2 bits are assigned to bank, 2 bits to byte select, 1 bit to channel.

Number of row bits = 34 – 10 – 3 – 2 – 2 – 1 = 16 bits.

In address mapping, 2 bits are assigned to byte select, 4 bits are assigned to lower column, 1 bit to channel, 3 bits to bank groups, 2 bits to banks, 6 bits to high column, 16 bits are assigned to row. There are a total of 34 bits. To extract information from addresses, the length of each entry is as follows:

Byte-select: [0:1], low column: [5:2], channel: [6:6], bank group: [9:7], bank: [11:10], high column: [17:12], row: [33:18] all these are extracting information from request part of address. We give the request at every Pos-edge clk of cpu cycle and according to that request we issue the DRAM command.

We created an array of structure and every line is read from the file and stored in temp variable, if the input time is less than cpu time then it is being pushed inside the array, else input time is being subtracted form the cpu time and the cpu clock is repeated that many times, after that cpu time is assigned input time, and once the array size is reached 16, then the cpu clock will repeat till the size again ,decreases by one, once it decreases it again pushes a new element into the array. Then parallelly DIMM clock runs, it reads the first element from the array, and time parameters are being calculated and DIMM clock is being synchronized with the timing parameters. Once everything is done, the array is left shifted, hence the first element is being deleted and rest elements are being left pushed. And the array size is decreased by one. Hence the above cpu loop is again repeated, since the array size is being decreased, this is performed till every line is being read form the file.

**Timing parameters:**

TRP (Row Precharge delay) : Time to pre-charge DRAM array in preparation for another row access.

TCL (CAS latency) : Time between read command and first valid data.

TBurst (Burst length) : The number of data chunks that can be accessed in a single, uninterrupted sequence following the activation of a row in the memory.

TRCD (Row column delay) : Minimum time between ACT command and READ command.

**Test steps:**

1.1 If the processed request and the next request is having the same bank group and the same bank respectively, then the ACT command for the next request will be issued after Trp clock cycles since the row needs to be pre-charged.

ACT0 | ACT1 (TRCD)

RD0 | RD1 (TCL + T Brust)

+PRE (TRP)

1.2 If the processed request and the next request is not having the same bank group and the different bank respectively, then the ACT command for the next request will be issued in next clock cycle after We need NOT to wait till TRP to issue next ACT Command.

ACT0 | ACT1 (TRCD)

RD0 | RD1 (TCL + T Brust)

PRE (TRP)

**Expected output for the given input:**

INPUT:

Time Core Operation Address

1. 2 0 0x01FFFF009

(Row: 7FF, Columnh:3F ,Bank: 0,BG:0 ,Channel:0 , ColumnL:2 ,Byte-Select: 1)

1. 10 1 0x10FFFF052

(Row: 43FF, Columnh:3F ,Bank: 0,BG:0 ,Channel:1 , ColumnL:4 ,Byte-Select: 2)

1. 12 2 0x10FDFF433

(Row: 43F7, Columnh:3F ,Bank: 1,BG:0 ,Channel:0 , ColumnL:C ,Byte-Select: 1)

If the processed instruction and next instruction have same bank group and same bank, and the ACT command for the next instruction will be issued after waiting Trp clock cycle:

2 0 ACT0 0 0 7FF

4 0 ACT1 0 0 7FF

80 0 Read0 0 0 3F (TRCD) (Where, TRCD = TRP x 2 + ACT (Here, TRP = 39 clock cycles, ACT = 2 clock cycles))

82 0 Read1 0 0 3F (TRCD)

178 0 PRE 0 0 ((TCL + TBurst )x 2 +80) (TCL = 40, TBurst = 8)

(TRP x 2 + 176) = (39 x 2) + 180 = 258 clock cycles

Hence the ACT command for the next instruction is given after 254 clock cycle i.e 256.

258 1 ACT0 0 0 43FF

260 1 ACT1 0 0 43FF

336 1 Read0 0 0 3F (TRCD) ( TRCD = TRP x 2 + ACT ( TRP = 39, ACT = 262)

338 1 Read1 0 0 3F (TRCD)

434 1 PRE 0 0 ((TCL + TBurst)x 2 +342) (TCL = 40, TBurst = 8)

If the processed instruction and next instruction have same bank group and different banks, the process need not wait for TRP ACT command for the next instruction will be issued at the next DIMM clock cycle

436 0 ACT0 0 1 43F7

438 0 ACT1 0 1 43F7

514 0 Read0 0 1 3F (TRCD) ( TRCD = TRP x 2 + ACT ( TRP = 39, ACT = 442)

516 0 Read1 0 1 3F (TRCD)

612 0 PRE 0 0 ((TCL + TBurst)x 2 +522) (TCL = 40, TBurst = 8)

**Same bank and bankgroup for the below address respectively:**

0 1 0 2575ADCFC

1 2 1 2B76EDCA9

2 3 2 3B6EFFCE5

4 41 3B6DE8CC3

A screenshot of a computer

Description automatically generated

**Different bank and bankgroup for the below address respectively:**

0 1 0 344EB56A8

1 2 1 2BB14A1F9

2 3 2 2D1DEB816

4 4 1 3BCA51E6F

A screenshot of a computer

Description automatically generated