Project Report

Team 17 :

1) Venkat Reddy Lakkireddy

   PSUID – 989142860

2) Manoj Murthy Bontalakoti

   PSUID – 955778409

3) Kartheek Pakala

   PSUID - 988379356

4) Bhavana Marpadaga

   PSUID - 985955815

Scheduling Algorithm for a DRAM Controller

Description:

We are simulating a Truly Open Page scheduling algorithm for a DRAM controller capable of serving a 12-core 4.8 GHz processor employing a single 16GB PC5- 38400 DIMM. The open page policy serves the purpose of spatial locality when there are sequential memory references to the page. This corresponds to multiple page hits and avoiding the issuance of PRE command for every request as opposed to the closed page policy. This eventually increases the effective bandwidth of the DIMM when there are multiple page hits.

The controller has a request queue which can hold up to 16 outstanding requests from the cores. Each DIMM cycle consists of 2 CPU cycles. The tests to check for proper functioning of the queue, closed page policy implementation and open page implementation are as follows.

Test Environment: The input files are generated using Microsoft Excel spreadsheets.

Testcase-1: Proper working of the queue

1.1 – Queue Enqueuing:

Scan the input trace file and keep incrementing the CPU clock to check for the time<=cpu_clock. At every DIMM clock cycle keep checking if there is a memory request to be handled. If there is a request at the current CPU cycle enqueue the request into the queue. Keep checking this until the end of the input trace file. If time<=cpu_clock does not satisfy then just increment the cpu_clock and keep checking for the request time to satisfy. If the queue is full the request should not be enqueued until a request is dequeued.

1.2 – Dequeue:

When a request is completely satisfied, and the last chunk of data is delivered at the end of tBURST the request is then dequeued from the queue.

Testcase-2: Closed Page Policy

Request handling:

At every DIMM cycle (even CPU clock cycle) check if the queue has any requests. If the queue is not empty take a request from the head of the queue to handle it. For a memory controller implementing a closed-page policy, the controller needs to issue ACT0 and ACT1. In a closed-page policy, the controller closes the page after it is done accessing the row and column requested by the CPU core. The CPU clock must be constantly incremented, and the queue needs to be enqueued and dequeued accordingly while handling the request.

2.1 – Check tRCD:

After issuing ACT0 and ACT1 the controller needs to wait for a delay of tRCD to issue RD0,RD1/WR0,WR1.

2.2 – Check the operation:

Depending on the operation from the CPU core the controller issues a RD or WR. If the operation is 0(Memory Read) or 2 (Instruction Fetch) it is basically a read operation. If it is 1(Memory Write), it is a memory write into the DRAM.

2.3- Check Read to Precharge timing:

The controller needs to wait for a minimum of tRTP from RD0 and RD1 commands to issue a PRE to the current bank. It also needs to satisfy tRAS from ACT0 and ACT1 to issue the PRE command. The PRE command can be issued only when these two-timing parameters are satisfied.

2.4 – Check tCL+tBURST:

The controller needs to wait for tCL +tBURST to delete the request from the queue and move on to the next request.

2.5 – Check Write to Precharge timing:

To issue PRE after WR0, WR1 the controller needs to wait for write recovery time (tWR) after the last burst cycle. To issue PRE after WR0, WR1 the controller needs to wait for a delay of CWL+tBURST+tWR.

2.6 – A request arrives while a request is being handled:

If there is a reference from a core while a request is currently being handled and the queue is not full, then the request needs to be enqueued.

2.7 – The next request is to the different bank (Different Bank Group/Bank/ both):

If the next request in the queue is to a different bank than that of the current request, then the controller does not need to wait for a delay of tRP to issue the ACT to the different bank. But it needs to wait until it satisfies tRAS and tRRD for it to issue another ACT.

2.8 – The next request is to the same bank and bank group:

If the next request in the queue is to the same bank and bank group as the current request, then the controller needs to wait for a delay of tRP to issue the ACT again to the same bank.

Example inputs:

1) Input to test the case 1.1, 2.1, 2.2, 2.3, 2.4, 1.2 - " 5 0 0 000000000 "

This request is pushed into the queue at time 5 but it will be handled at the next DIMM clock which is CPU clock 6.

Expected output for the test case –

```
6     0  ACT0  0  0  0
8     0  ACT1  0  0  0
86    0  RD0   0  0  0
88    0  RD1   0  0  0
160   0  PRE   0  0
```

2) Next input to test the case 2.2, 2.5, 2.6, 2.7 - " 10 1 1 00007F4BF "

This request is enqueued into the queue at CPU clock 10 but is handled right after the previous request is dequeued from the queue. This request is a write operation. It must follow the timing as said in case 2.5. The request is to a different bank hence the controller need not wait for tRP to issue ACT0 and ACT1 but can be issued after tCl+tBURST.

Expected Output for this test case-

```
184     0  ACT0  1  1  1
186     0  ACT1  1  1  1
264     0  WR0   1  1  3f
266     0  WR1   1  1  3f
418     0  PRE   1  1
```

3) Next input to test the case 2.8 - " 12 2 2 00008148F "

This request is a instruction fetch to the same bankgroup and same bank, hence the controller needs to wait for tRP after the previously issued PRE command to satisfy the case 2.8 and issue an ACT command-

Expected Output for this testcase-

```
496     0  ACT0  1  1  2
498     0  ACT1  1  1  2
576     0  RD0   1  1  1
578     0  RD1   1  1  1
650     0  PRE    1  1
```

Testcase-3: Open Page Policy:

In an open page policy, we do not explicitly close a page after its column is accessed. We leave the page open believing there might be a hit to it later. If there is a page hit, we directly give out RD/WR commands without giving any Act or PRE charge. But if there is a page miss, we might have to close the page and open again, ending up in giving PRE,ACT and RD/WR commands.

3.1 – Page Empty:

If the page is empty we issue commands ACT and RD/WR as the page is already precharged. The test input should be a first reference of RD/WR to a bank and then it becomes a case for page empty.

Example: - "0 8 2 26E12C010" – Ba=0 , Bg=0 , Row=9B84

3.2 – Page Hit:

If there is a page hit we just have to issue RD/WR commands to the DRAM as the row is already opened previously. The test input should be a reference of RD/WR to a bank which has the row address same as this request and then it becomes a case for page hit.

Example: - "60 9 0 26E102010" – Ba=0 , Bg=0 , Row=9B84

3.3 – Page Miss:

If there is a page miss we have Precharge the bank again and issue ACT and RD/WR commands for the corresponding request. The test input should be a reference of RD/WR to a bank which has the row address different as this request and then it becomes a case for page miss.

Example: - "60 9 0 000000010" – Ba=0 , Bg=0 , Row=0000

The open page policy tests contain all the timing parameters in the closed page as stated above. These are the other timing constraints to be satisfied.

3.4 – Test for tccd_s:

There should be a delay between 2 consecutive reads to 2 different bank groups of a minimum of tccd_s. The reference should be a page hit of read followed by a read to a different bank group to test for tccd_s.

Example: -

80 11 2 3EC235830 - Ba=2 , Bg=0

90 0 0 00011C090 - Ba=0 , Bg=1 (It's a Hit and must wait the tccd_s.)

3.5 – Test for tccd_l:

There should be a delay between 2 consecutive reads to 2 same bank group of a minimum of tccd_l. The reference should be a page hit of read followed by a read to the same bank group to test for tccd_l.

Example:-

50 8 2 26E12C010 – Ba=0 , Bg=0

60 9 0 26E102010 – Ba=0 , Bg=0 (It's a Hit and must wait the tccd_l.)

3.6 – Test for tccd_l_wr:

There should be a delay between 2 consecutive writes to 2 same bank group of a minimum of tccd_l_wr. The reference should be a page hit of write followed by a write to the same bank group to test for tccd_l_wr.

Example:-

17500 0 1 000024004 – Ba=0 , Bg=0

17523 1 1 0000C5C10 – Ba=3 , Bg=0 (It's a Hit and must wait the tccd_l_wr.)

3.7 – Test for tccd_s_wr:

There should be a delay between 2 consecutive writes to 2 different bank groups of a minimum of tccd_s_wr. The reference should be a page hit of write followed by a write to a different bank group to test for tccd_s_wr.

Example:-

17523 1 1 0000C5C10 – Ba=3 , Bg=0

17600 2 1 0004A4A2C – Ba=2 , Bg=4 (It's a Hit and must wait the tccd_s_wr.)

3.8 – Test for tccd_l_rtw:

There should be a delay between read and write to the same bank group of a minimum of tccd_l_rtw. The reference should be a page hit of write followed by a read to the same bank group to test for tccd_l_rtw.

Example:-

2000 6 0 324A74524 – Ba=1 , Bg=2

2100 7 1 0002FAD30 – Ba=3 , Bg=2 (It's a Hit and must wait the tccd_l_rtw.)

3.9 – Test for tccd_s_rtw:

There should be a delay between read and write to the different bank groups of a minimum of tccd_s_rtw. The reference should be a page hit of write followed by a read to a different bank group to test for tccd_s_rtw.

Example:-

17000 11 2 00039C99C – Ba=2 , Bg=3

17500 0 1 00000900C – Ba=0 , Bg=0 (It's a Hit and must wait the tccd_s_rtw.)

3.10 – Test for tccd_l_wtr:

There should be a delay between write and read to the same bank group of a minimum of tccd_l_wtr. The reference should be a page hit of read followed by a write to the same bank group to test for tccd_l_wtr.

Example:-

2100 7 1 0002E4D24 – Ba=3 , Bg=2

2500 8 2 0004B092C – Ba=2 , Bg=2 (It's a Hit and must wait the tccd_l_wtr.)

3.11 – Test for tccd_s_wtr:

There should be a delay between write and read to the different bank groups of a minimum of tccd_s_wtr. The reference should be a page hit of read followed by a write to the same bank group to test for tccd_s_wtr.

Example:-

15000 10 1 13950C6BC – Ba=1 , Bg=5

17000 11 2 0003A39A4 – Ba=2 , Bg=3 (It's a Hit and must wait the tccd_s_wtr.)