Test Plan Document

- 1. INTRODUCTION
  - 1.1. PROJECT
  - 1.2. PROJECT DESCRIPTION
  - 1.3. TEST STRATEGY / METHODOLOGY
- 2. INPUT VALIDIFICATION
- 3. QUEUE REQUESTS
- 4. POLICY IMPLEMENTED CORRECTLY
  - 4.1. LEVEL 0
    - 4.1.1. CLOSED PAGE POLICY
    - 4.1.2. IN-ORDER SCHEDULING
  - 4.2. LEVEL 1
    - 4.2.1. OPEN PAGE POLICY
  - 4.3. LEVEL 2
    - 4.3.1. BANK LEVEL PARALLELISM
  - 4.4. LEVEL 3
    - 4.4.1. OUT-OF-ORDER SCHEDULING
- 5. DRAM COMMANDS FUNCTIONALLY CORRECT
  - 5.1. READ0, READ1
  - 5.2. WRITE0, WRITE1
  - 5.3. ACTIVATE0, ACTIVATE1
  - 5.4. PRECHARGE
- 6. DRAM TIMING CORRECT
  - 6.1. Level 0
  - 6.2. Level 1
  - 6.3. Level 2
  - 6.4. Timing Descriptions
    - 6.4.1. tRCD
    - 6.4.2. tRTP
    - 6.4.3. tWR
    - 6.4.4. tCL
    - 6.4.5. tCWL
    - 6.4.6. tRAS
    - 6.4.7. tRP
    - 6.4.8. tRC
    - 6.4.9. tFAW
    - 6.4.10. tRRD_S and tRRD_L
    - 6.4.11. tCCD_S and tCCD_L
    - 6.4.12. tCCD_S_WR and tCCD_L_WR
    - 6.4.13. tCCD_S_RTW and tCCD_L_RTW
    - 6.4.14. tCCD_S_WTR and tCCD_L_WTR
    - 6.4.15. tBURST

1. INTRODUCTION

1.1. PROJECT

Project Name:

> Team 05 ECE485/585 Final Project

Members:

- Abdulaziz Alateeqi
- Meshal Almutairi
- Eduardo Simancas
- Gene Hu

Last Updated:

> 11/27/2023

1.2. PROJECT DESCRIPTION

See final_project_description.pdf under docs for detailed explanation.

Scheduling Aggressiveness

> For extra credit this project team will *try* to add open page policy, bank level parallelism, and out of order scheduling.

1.3. TEST STRATEGY / METHODOLOGY

Create a trace file as an input (ASCII text file) using a test case generator.

1. Test case generator will be in program(Python) or Excel spreadsheet
    - Enter {time, request, row, bank, bank group, col}
    - Macro generates request (combining fields into address)
        - **REMINDER**: The input's address need to be hexadecimal and 8-byte aligned
2. Test cases can be commented for documentation (will be removed by pre-processor)

2. INPUT VALIDIFICATION

> Input should be in the format: time, core, operation, address. Where time is in (absolute) CPU clocks, core is between 0 to 11. operation is 0, 1, or 2, and address is 34-bit address represented in hexadecimal and it is 8-byte aligned.

Test Cases:

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
1	Invalid time	Negative time -1	Detect that CPU time can not be negative	
2	Invalid Core	Core 24	Detect core is not 0-11 and let user know	

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
3	Invalid operation	-1	Detect operation is not 0-2	
4	Address larger than 34 bits	0x7FFFFFFFF	Detect 35>34	
5	Parsing address correctly	4 requests, each making all the bits of one parameter 1's, order of BA,BG,ROW,COLUMN	Req1 = BA 3, Req2 = BG 7, Req3 = ROW 65535, Req4 = COL 1023	

HOW:

- Address can be tested by making all of the one parameter's bits all 1's. For example, to test that the parsing is working for ROW address, make ROW=65535 and everything else 0. If the output shows ROW!=65535 and another parameter!=0, then the input is not being taken in correctly. Repeat for BA,BG,COL. Checkpoint2 shows examples of this.
- Error checking should be made for time, core, and operation.

3. QUEUE REQUESTS

> Only 16 outstanding requests are allowed in the simulation at a time. If there are more than 16 requests in the queue when a new request comes in, that new request will not enter the queue until an item is completed to free up a spot. A request will only be completed and removed from the queue when its last command is executed (ex. read request will leave once the READ command is completed; Total timing: tCL + tBURST -> request is done).

Test Cases:

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
1	Requests coming in with a full queue	18 consecutive requests starting at CPU 0	Request 17 should be queued after a request is finished, then request 18 after another request is finished.	Use debug mode, should be valid in all levels

4. POLICY IMPLEMENTED CORRECTLY

4.1. LEVEL 0

4.1.1. CLOSED PAGE POLICY

> Page is closed after a read/write command. Thus a precharge must always follow those two commands. Every page reference will page empty.

Test Cases:

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
1	Test page empty on successive references.	Back to back references to same BG,BA, same row, different column.	ACT -> READ -> PRE -> ACT -> READ -> PRE	Read or write should not matter
2	Test page empty on two different references.	Back to back references to different BG,B,ROW.	ACT -> READ -> PRE -> ACT -> READ -> PRE	
3	Read followed by write followed by read	Three requests going to same row	ACT -> READ -> PRE -> ACT -> WRITE -> PRE -> ACT -> READ -> PRE	

4.1.2. IN-ORDER SCHEDULING

> Complete the first item in the queue before moving onto the next item. For closed page, output will always be in order: ACT,RD/WR,PRE. For open page, there is more variation depending on page hit or page miss, but it should still be clear if a request is being process once the previous one is completed.

4.2. LEVEL 1

4.2.1. OPEN PAGE POLICY

> Page is left open for a moment after a read/write command. It will stay open until a read/write is referencing its bank, bank group, but different row (page miss). This means page misses will incur a precharge before activate, and page hits do not need activate (but will need to be more wary of tCCD).

Test Cases:

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
1	Test page hit.	Back to back references to same BG,BA, same row, different column.	ACT -> READ -> READ	

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
2	Test page empty followed by page miss followed by page hit.	Back to back references to same BG,BA, different rows, and then second row gets referenced again.	ACT -> READ -> PRE -> ACT -> READ -> READ	
3	Test open page tracking by intersecting a successive request with another request, while trying to trick it by making the BA number be the same.	Back to back references that are intersected by a page access from a different BG with different ROW.	ACT -> READ -> ACT -> READ -> READ	
4	Test open page tracking by intersecting a successive request with another request, while trying to trick it by making the BG number be the same.	Back to back references that are intersected by a page access from a different BA with different ROW.	ACT -> READ -> ACT -> READ -> READ	Same result as 3
5	Test open page tracking by intersecting a successive request with another request, while trying to trick it by making the ROW be the same.	Back to back references that are intersected by a page access from a different BG,BA, with same ROW.	ACT -> READ -> ACT -> READ -> READ	Same result as 3
6	Test open page tracking by intersecting a successive request with another request, while trying to trick it by making the BA,ROW the same.	Back to back references that are intersected by a page access from a different BG, with same BA,ROW.	ACT -> READ -> ACT -> READ -> READ	Same result as 3
7	Page hit with read and write combo.	Read followed by write request to same BG,BA,ROW	ACT -> READ -> WRITE	
8	Page miss with read and write, then a page hit with read.	Read followed by write to same BG,BA, different ROW, then a read to the same BG,BA,ROW as write.	ACT -> READ -> PRE -> ACT -> WRITE -> READ	
9	Back to back page hits, alternating read, write, read	Read, write, read requests all going to same BG,BA,ROW.	ACT -> READ -> WRITE -> READ	

4.3. LEVEL 2

4.3.1. BANK LEVEL PARALLELISM

> Scheduler is able to interleave commands to different BG,BA. For example, after it issues an activate command to BA0,BG0, it can issue another activate to a different BX,BGX combination before coming back and issuing the READ command.

CASES

1. When we are waiting for timing contraints on a command being executed in a certain BG,B, and there is a request to a different BG,B in the queue, start the new request (if it does not violate other timings). Repeat.
    - Allowed to skip over requests that are waiting for a bank to be available (technically out of order)

Test Cases:

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
1	BLP when each request is going to a different BA in the same BG.	4 requests going to same BG, each with different BA.	ACT -> ACT -> ACT -> ACT -> RD -> RD -> RD -> RD	Testing BG 0, 3, 6 to check 0, even, and odd checks.
2	BLP is working correctly when there are two requests vying for same BG,BA.	3 requests, first two will go to same BG,BA, and the third will go to different BG,BA.	ACT -> ACT -> RD -> RD -> PRE -> ACT -> RD	First two ACT are for request 1 and 3 respectively. Second request page miss.
3	Test open page tracking when interlearving BG,BA, while trying to trick it by making the BA number be the same.	Back to back references that are intersected by a page access from a different BG with different ROW.	ACT -> ACT -> READ -> READ -> READ	Output order is important. It should be BGX->BGY->BGX
4	Test open page tracking when interlearving BG,BA, while trying to trick it by making the BG number be the same.	Back to back references that are intersected by a page access from a different BA with different ROW.	ACT -> ACT -> READ -> READ -> READ	Same output as 3
5	Test open page tracking when intervleaving BG,BA, while trying to trick it by making the ROW be the same.	Back to back references that are intersected by a page access from a different BG,BA, with same ROW.	ACT -> ACT -> READ -> READ -> READ	Same output as 3
6	Test open page tracking when intervleaving BG,BA, while trying to trick it by making the BA,ROW the same.	Back to back references that are intersected by a page access from a different BG, with same BA,ROW.	ACT -> ACT -> READ -> READ -> READ	Same result as 3
7	Page hit with read and write combo.	Read followed by write request to same BG,BA,ROW	ACT -> READ -> WRITE	Result should be same as open page policy

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
8	Page miss with read and write, then a page hit with read.	Read followed by write to same BG,BA, different ROW, then a read to the same BG,BA,ROW as write.	ACT -> READ -> PRE -> ACT -> WRITE -> READ	Result should be same as open page policy
9	Back to back page hits, alternating read, write, read	Read, write, read requests all going to same BG,BA,ROW.	ACT -> READ -> WRITE -> READ	

4.4. LEVEL 3

4.4.1. OUT-OF-ORDER SCHEDULING

> Scheduler is able to process requests out-of-order. Must incorporate "aging"/timeout so a request isn't sitting in queue forever.

CASES

1. READ over WRITES
    - Only when the addresses are different. If this is done when the addresses are the same, it will lead to reading stale data.
2. Prioritize page hits over page misses

Test Cases:

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
1	Read over write when valid	3 requests all going to the same BG,BA. R1 will be read, R2 will be write, and R3 will be read. All are going to different ROW,COL.	RD -> RD -> WR	Activates and precharge are omitted
2	No read over write when not valid (SAME addresses).	3 requests all going to the same BG,BA. R1 will be read, R2 will be write, and R3 will be read. All are going to the SAME ROW,COL.	RD -> WR -> RD	Also tests how simulator handles when a page hit over page miss is possible, but invalid due to not being able to put read over write.
3	Prioritize page hits over page misses.	3 requests all going to the same BG,BA. R1 will be read, R2 will be read (different row from R1), R3 will be read(same row as R1).	ACT -> RD -> RD -> PRE -> ACT -> RD	

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
4	Test our ageing process (should be 920)	12 requests all going to same BG,BA, different rows. Only req2 is a write. First req comes in at 197. All other requests start coming in at time 200.	(ACT -> RD -> PRE)x8, ACT -> WR -> PRE	

note: (R# = request number)

5. DRAM COMMANDS FUNCTIONALLY CORRECT

Note: In level 2+, the scheduler will always pick READ1, WRITE1, and ACT1 if they are available. This project is using 1n mode, so there is a 1 DIMM cycle delay between 0 -> 1.

5.1. READ0, READ1

> READ0 selects the command being issued (due to MUX) and READ1 will get column address. Followed by tCL = 40. Output should show the bank, bank group, and column being accessed (same as the input column address for this project).

5.2. WRITE0, WRITE1

> WRITE0 selects the command being issued (due to MUX) and WRITE1 will get the column address. Followed by tCL = 40. Output should show the bank, bank group, and column being accessed (same as the input column address for this project).

5.3. ACTIVATE0, ACTIVATE1

> ACTIVATE0 selects the command being issued (due to MUX) and ACTIVATE1 will get the row address. Followed by tRCD = 39. Output should show the bank, bank group, and row being accessed.

5.4. PRECHARGE

> If a page is open, the data needs to return from the sense amplifiers so another activate can occur. In a closed page policy, every READ/WRITE command should be followed by PRE. In an open page policy, PRE will occur when the currently open page is not the request page. Followed by tRP = 39. tRAS and tRTP must be satisfied before it can be issued. Output should show the bank, bank group being precharged.

6. DRAM TIMING CORRECT

6.1. Level 0

Test Cases:

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
1	tRCD, tCL, tRAS, tRP: ACT -> READ -> PRE -> ACT	Read request at CPU 197 and 198.	ACT0 at DIMM 99, ACT1 at DIMM 100, RD0 at DIMM 138, RD1 at DIMM 139, PRE at DIMM 176, ACT0 at DIMM 214, ACT1 at DIMM 215	BURST at DIMM 179, Dequeue at DIMM 187, (check debug if you want). If PRE goes early, maybe bug in tRTP.
2	tRCD,tCWL,tBURST,tWR,tRP: ACT -> WRITE -> PRE -> ACT	Write request at CPU 197 and 198.	ACT0 at DIMM 99, ACT1 at DIMM 100, WR0 at DIMM 138, WR1 at DIMM 139, PRE at DIMM 215, ACT0 at DIMM 254	BURST at DIMM 177, Dequeue at DIMM 185.

6.2. Level 1

Bolded is what we are looking for.

Test Cases:

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
1	tCCD_L, tRAS, tRP: ACT -> **READ -> READ -> PRE -> ACT**	Two reads to same BG,BA,ROW at times 197 and 198. Read to same BG,BA, different ROW at time 199.	RD1 at DIMM 139, RD1 at DIMM 151 PRE at DIMM 176, ACT1 at DIMM 215	

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
2	tCCD_L_WTR, tRTP: ACT -> **WRITE -> READ -> PRE** -> ACT	Read and write to same BG,BA,ROW at times 197 and 198. Read to same BG,BA, different row at time 199.	WR1 at DIMM 139, RD1 at DIMM 209, PRE at DIMM 227	tRTP should encompass tWR
3	tCCD_L_RTW, tCWL+tBURST+tWR: ACT -> **READ -> WRITE -> PRE** -> ACT	Write and read to same BG,BA,ROW at times 197 and 198. Read to same BG,BA, different row at time 199.	RD1 at DIMM 139, WR1 at DIMM 155, PRE at DIMM 231	
4	tCCD_L_WR, tCWL+tBURST+tWR: ACT -> **WRITE -> WRITE -> PRE** -> ACT	Two writes to same BG,BA,ROW at times 197 and 198. Write to same BG,BA, different row at time 199.	WR1 at DIMM 139, WR1 at DIMM 187, PRE at DIMM 263	
1	tCCD_L, tRTP: ACT -> **READ -> READ -> PRE** -> ACT	Two reads to same BG,BA,ROW at times 197 and 359. Read to same BG,BA, different ROW at time 360.	RD1 at DIMM 139, RD1 at DIMM 180 PRE at DIMM 198	

6.3. Level 2

Bolded is what we are looking for.

Test Cases:

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
1	tRRD_S, tCCD_S: ACT -> ACT -> READ -> READ	Two read requests to different BG at times 197 and 198.	ACT1 at DIMM 100, ACT1 at DIMM 108, RD1 at DIMM 139, RD1 at DIMM 147	

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
2	tRRD_L, tCCD_L: ACT -> ACT -> READ -> READ	Two read requests to same BG at times 197 and 198.	ACT1 at DIMM 100, ACT1 at DIMM 112, RD1 at DIMM 139, RD1 at DIMM 151	
3	tRRD_L, tCCD_L, tCCD_L: ACT -> ACT -> READ -> READ -> READ	Requests X,Y to same BG at times 197 and 198, Request Z gets a page hit to same BA as X.	ACT1 at DIMM 100, ACT1 at DIMM 112, X_RD1 at DIMM 139, Y_RD1 at DIMM 151, Y_RD1 at DIMM 163	
4	tCCD_S_WTR: ACT -> ACT -> **WRITE -> READ**	Write and read requests to different BG at times 197 and 198.	WR1 at DIMM 139, RD1 at DIMM 191	
5	tCCD_L_WTR: ACT -> ACT -> **WRITE -> READ**	Write and read requests to same BG and different BA at times 197 and 198.	WR1 at DIMM 139, RD1 at DIMM 209	
6	tCCD_S_RTW: ACT -> ACT -> **READ -> WRITE**	Read and write requests to different BG at times 197 and 198.	RD1 at DIMM 139, WR1 at DIMM 155	
7	tCCD_L_RTW: ACT -> ACT -> **READ -> WRITE**	Read and write requests to same BG and different BA at times 197 and 198.	RD1 at DIMM 139, WR1 at DIMM 155	
8	tCCD_S_WR: ACT -> ACT -> **WRITE -> WRITE**	Two write requests to different BG at times 197 and 198.	WR1 at DIMM 139, WR1 at DIMM 147	

#	OBJECTIVE	INPUT	EXPECTED RESULTS	Notes
9	tCCD_L_WR: ACT -> ACT -> **WRITE -> WRITE**	Two write requests to same BG and different BA at times 197 and 198.	WR1 at DIMM 139, WR1 at DIMM 187	
10	tFAW: ACT -> ACT -> ACT -> ACT -> ACT			not completed

6.4. Timing Descriptions

Note:

1. All the RRD and CCD timings are level 1+ or level 2+ only because the cases only occur when a page hit happens, or because our scheduler picks the required back to back commands during bank level parallelism.
2. Delay starts counting from the second 1/2 of the first command to the second 1/2 of the second command. This means the timing from activate to getting data = 1+tRCD+tCL, there is only a single "+1" because the second is overlapped by tRCD.

6.4.1. tRCD

> tRCD = 39. Cycles to open a row. Occurs between ACT -> READ/WRITE.

6.4.2. tRTP

> tRTP = 18. Read to precharge delay. Occurs between READ -> PRECHARGE. This means a PRE can be done before we start (or during) reading out data, but we still need to wait for tRAS to be satisfied if necessary. This is possible because the data is moved to a buffer when the RD command is issued, so we can close the page prematurely. Level 1+ (page hits).

6.4.3. tWR

> tWR = 30. Write recovery time. Occurs between WRITE DATA -> PRECHARGE. This means the time it takes from issuing a WRITE to PRE is tCWL + tBURST + tWR = 76 from WR1 -> PRE.

6.4.4. tCL

> tCL = 40. Cycles to begin receiving data (from read). Occurs between READ -> DATA.

6.4.5. tCWL

> tCWL = 38. Column write delay (WR replacement for tCL). Occurs between WRITE -> DATA.

6.4.6. tRAS

> tRAS = 76. Amount of cycles needed between ACT -> PRE. By doing the math from the previously mentioned timings, in level 0 we are able to issue the precharge command before tCL is satisfied and while tRTP is satisfied. For example if ACT1 finished at DIMM clock 1, tRAS will finish at DIMM clock 77, two cycles before tCL.

6.4.7. tRP

> tRP = 39. Row precharge time. Occurs between PRE -> ACT.

6.4.8. tRC

> tRC = tRAS + tRP = 115. Minimum time between activates in a bank. As long as tRP and tRAS are tested correctly, this should be satisfied as well.

6.4.9. tFAW

> tFAW = 32. Amount of time needed between every 4 ACT commands. Level 2+.

6.4.10. tRRD_S and tRRD_L

> tRRD_S = 8, tRRD_L = 12. Activate to activate command delay. When switching between bank groups use _S, switching inside a bank group use _L. Level 2+.

6.4.11. tCCD_S and tCCD_L

> tCCD_S = 8, tCCD_L = 12. Read to read command delay. When switching between bank groups use _S, switching inside a bank group use _L. Level 1+.

6.4.12. tCCD_S_WR and tCCD_L_WR

> tCCD_S_WR = 8, tCCD_L_WR = 48. Write to write command delay. _S for different bank group, _L for same bank group. Level 1+.

6.4.13. tCCD_S_RTW and tCCD_L_RTW

> tCCD_S_RTW = 16, tCCD_L_RTW = 16. Read to write command delay. _S for different bank group, _L for same bank group. Level 1+.

6.4.14. tCCD_S_WTR and tCCD_L_WTR

> tCCD_S_WTR = 52, tCCD_L_WTR = 70. Write to read command delay. _S for different bank gropu, _L for same bank in same bank group. Level 1+.

6.4.15. tBURST

> tBURST = 8. Burst length 16 with half cycle for each data = 8 cycles total.