ECE-585

MICROPROCESSOR SYSTEM DESIGN

TEST PLAN – GROUP 11

Simulating the scheduler component of a DDR5 Memory Controller

Avadhoot Modak
PSU ID: 924895764

Satyam Sharma
PSU ID: 956073158

Smit Patel
PSU ID: 967255722

Aaditya Shah
PSU ID: 958728607

ABSTRACT

This project conducts a simulation of the scheduler component of a memory controller designed to support a 12-core 4.8 GHz processor utilizing a single 6GB PC5-38400 DIMM. The system operates under a relaxed consistency (XC) Model, with the memory controller utilizing a single channel of the DIMM. The DIMM is configured with memory chips organized as x8 devices, featuring a 1KB page size and 40-39-76 timing. These devices are equipped with 8 bank groups, each containing banks. The DIMM employs 1N mode for commands requiring two cycles and does not incorporate system level ECC. Adopting a closed-page policy with in-order execution, this design bind to the specified timing constraints which were derived from the datasheet provided.

Implementation

The DDR scheduling module of the Memory Controller was implemented in System-Verilog as per below given features:

- Single Channel Usage: Design uses only channel zero.
- In-Order Execution: Requests are processed in order they are received, which makes it sequential, and execution becomes quite predictable.
- Closed-Page Policy: Memory controller adopts a closed-page policy.
- Timing Constraints: Implementation conforms to the timing constraints specified in datasheet.
- Bank-Level Considerations: Requests which target the same bank or different banks are managed appropriately to resolve potential conflicts. The status of each request, which impacts the subsequent requests, is tracked in an array.

Execution and Tools Used

- The tool which has been used for the execution of this program is QuestaSim, ensuring compatibility with the System Verilog implementation. Makefile has been utilized for testbench execution, providing a streamlined approach for running the testcases.

GROUP 11 – MSD PROJECT TEST CASES PLAN

Note: ACTIVATE and PRECHARGE commands would be required for all of the 16 test cases scenarios discussed below, since we are implementing CLOSED PAGE POLICY with No Bank Parallelism.

Test Case Scenarios	Description	Bank group	Bank	Row	Column	Instruction	Time
1	Accesses to same row in the same bank group and bank different column consecutive reads	1	0	122	122	Read	0
		1	0	122	221	Read	1
		1	0	122	210	Read	2
		1	0	122	110	Read	3

2	Accesses same row, bank group and but different banks	1	0	122	122	Read	4
		1	1	122	221	Read	5
		1	2	122	210	Write	6
		1	3	122	110	Write	7
3	Accesses to different row in the same bank and bank group	1	0	121	122	Read	8
		1	0	122	221	Read	9
		1	0	123	210	Write	10
		1	0	111	101	Write	11
4	Accesses to different bank group	0	0	122	122	Read	12
		1	0	122	221	Write	13
		2	0	122	210	Read	14
		3	0	122	110	Write	15
5	Accesses to the same column	1	0	122	111	Read	16
		1	0	122	111	Read	17
		1	0	122	111	Write	18
		1	0	122	111	Write	19
6	Accesses to the different row, bank and bank group	0	3	123	111	Read	20
		1	2	122	213	Read	21
		2	1	212	101	Read	22
		3	0	101	110	Write	23
7	Accesses to same row in the same bank group and bank with consecutive writes	1	0	122	122	Write	24
		1	0	122	221	Write	25
		1	0	122	210	Write	26
		1	0	122	110	Write	27
8	Accesses to same row in the same bank group and bank with consecutive reads	1	0	122	122	Read	28
		1	0	122	221	Read	29
		1	0	122	210	Read	30
		1	0	122	110	Read	31

Other Test Case Scenarios:

1) Check for the trace file giving an incorrect address and check the output whether it's showing error messages or not.

2) Check for the output file whether it's showing the correct output or not. If not, then proper error messages are shown.

3) Validating the timing constraints for read instructions and ensuring that the timing constraints such as tRP, tRCD, tRAS, tCAS, etc. are as fulfilled as per the requirements. We will check all the timing given in the timing constraints and compare it with the required output.

4) Validating the timing constraints for write instructions and ensuring that the timing constraints such as tRP, tRCD, tRAS, tCAS, etc. are as expected as per the requirements. We will check all the timing given in the timing constraints and compare it with the required output.

5) Checking the CPU clock with a huge time difference: testing that whether there is a new CPU instruction after a very long gap. Suppose the first instruction is given at time #10 and the other is given at time #1000. We need to validate that the correct instruction is displaying or not after a very long-time gap for such an instruction.

6) Checking the CPU clock with multiple consecutive instructions: testing the functionality of the memory controller when there are instructions given on consecutive CPU cycles.

7) Out of Range Scenario: If value of cores or operations exceed beyond the predefined range. The controller should ideally drop any such requests which violate predefined boundary conditions and display an error message.

8) Queue Full: If the queue is filled to its maximum of 16 requests, it should stall the CPU i.e. other requests should not be pushed until popping of the previous instructions in the queue doesn't take place. Here, we assume that the controller handles all the requests by simultaneously enqueueing and dequeueing requests.

9) Queue Empty: Controller should wait / idle from when the first set of instructions is done processing and until the next set of requests arrives from the CPU.

10) Debugging Instructions: Check that when the code is run in the debug mode it shows debugging information as required when mode=" DEBUG" is given.

Conclusion

The DDR5 Memory Controller's scheduling module has been successfully designed to meet specified requirements. Adhering to timing constraints, and effectively handle various request type. A combination of manual and automated testbenches ensures comprehensive validation of the design's functionality and performance across diverse scenarios.