**ECE 571**

**Design and Verification**

**DDR4 Memory Controller Interface**

**by**

**Jeff Nguyen, Jon Vlas, Benjamin Huntsman**

1. **Scope**

This document provides details of planning, design, and verification of DDR4 Controller Interface for ECE517 Final Project.

* 1. **Functional Overview**

The main purpose of this project is to design and test a DDR4 Controller and interface using System Verilog constructs. The project is developed using the lecture on TLM as an example. The project is broken into three major blocks: the synthesizable interface, which provides the methods to translate the commands, data into pin signal levels used by the DDR Controller and Memory DDR4 DIMM. The second block is DDR Controller, which is built as a behavior model and iteratively converts to synthesizable block as the example in TLM lecture. The last block is the DIMM model and other parts, which are used in the test bench to verify the DDR Interface.

1. **Definitions**

tCCD CAS to CAS latency

tRRD Activate to Activate latency

tWTR Write to Read delay

tRTP Read to PreCharge.

tWR Write to PreCharge

tRCD Activate to CAS latency

tRP PreCharge to ACT latency

tCL CAS to DQ read

tCWL CAS to DQ write

tZQ Initialize process to ACT

AL Additive Latency

Burst Length Number of Bytes Read/Write

Preamble 1 or 2nCK

REF Refresh period

**3 Design and Verification Specification**

* The project shall implement the interface between the DDR Controller and DIMM using the interface construct.
* The interfaces shall declare signals and provide methods to share by both devices. The methods are to translates the commands such as PreCharge, ACT, CAS, etc. to pin levels, write data to DQ bus, and generate strobe data pins for DQ pin.
* The strobe pin data shall be center-aligned for write data while edge-aligned for read data.
* The project shall implement the DDR Controller as a behavior model to verify the interface. If time permitted, the DDR Controller will be developed to synthesize block.
* The DDR Controller shall provide true burst mode operation. That means the Controller will generate all the commands to DIMM as interleaving mode between the banks in the way that ACT, CAS commands continues assert without waiting for the previous command completed unless there is a hit to a same bank, different row.
* All the controller commands shall be satisfied the minimum timing specified in the standard.
* The DDR Controller shall implement the initialization sequence as well as refresh sequence. The initialization sequence to setup all the timing for DIMM operation. Other parts of the operation will not be implemented due to time constraint and/or cannot be verified by the simulation.
* The setup timing values shall be parameterized instead of variables.
* The DDR Controller shall work with both 4 and 8 bytes burst length.
* The DIMM mode shall be implemented to be part of verification of the Interface. The model can be able to store and read data and call the method to transmit the read data on the pin level.
* The stimulus for the verification shall be used the randomization in the last chapter.
* The module will be implemented for monitor read/write transactions via the interface and capture in the log file as the automated checker.
* The System Verilog Current Assertion shall be implemented for checking the protocols and timing.

**4 Design/Verification Implementation**

**4.1 Top Level Block Diagram**

The following block diagram is top-level test bench of the project

**DIMM Model**

**DDR Interface**

**DDR Controller**

**Test Bench Monitor**

**Test Bench**

**Stimulus**

**Concurrent**

**Assertion**

**Clock**

**4.2 DDR Interface (ddr\_interface.sv)**

The module implements the System Verilog (SV) interface construct for all signals connect between DIMM and DDR Controller as well as methods to translate the DDR Controller commands into pin signal transactions, strobe data pins DQS, write data pins and read data pins. There needs separate methods for write and read data because DIMM requires write data as center-aligned to strobe signal while read data as edge-aligned to strobe signals. But both DIMM and DDR Controller share the strobe data pins method. The interfaces does not completed implements all DIMM commands but adequate for setup, read/write operation, refresh such as MRS, ZQCL, ACT, CAS Read, CAS Write, PreCharge, Refresh, DES, and NOP. The SV constructs of packed array, structure, and user typedef enable the design to implement only one method for all the commands.

**4.2 DDR Controller Top Module (ddr\_top.sv)**

The following block diagram is for DDR Controller Top level Module.

**DDR Controller**

**DDR Config**

**DDR Burst**

**ACT**

**DDR Burst**

**R/W**

**DDR Burst**

**Data**

**Controller Interface**

**DDR Burst**

**CAS**

In this module, there are instances of the following modules:

* DDR Controller Module (ddr\_controller.sv)
* DDR Configuration Module (ddr\_config.sv)
* DDR Burst ACT Module (ddr\_act.sv)
* DDR Burst CAS Module (ddr\_cas.sv)
* DDR Burst RW Module (ddr\_rw.sv)
* DDR Burst Data Module (ddr\_data.sv)
* Controller Interface (ctrl\_intf.sv)

**4.2.1 Controller Interface (ctrl\_intf.sv)**

The interface only declared the share signals between DDR Controller sub-blocks and used as module port. There is contains no method in this interface.

**4.2.2 DDR Controller Module (ddr\_controller.sv)**

The module implements the FSM below to control between initialization, read/write, refresh, and update. The state machine starts to INIT state during the reset. The INIT state will initiate a sequence of Mode Register Set to setup the DDR operation. Then it stays in RW for burst read/write operation. The module implements a timer to keep track tREF. When the timer is close to expire, the FSM will asserts dev\_busy to Memory Manager Unit (MMU) or in this case, the stimulus generator, to stop send request and completed all read or write operation. The controller asserts the Refresh command to DIMM and return to RW state after tRC delay.

The Controller also provides a way for MMU to switch Burst Length for 8 bytes to 4 bytes and vice versa. The Controller signals to stop receiving any requests and finishes all current operation before start to set DIMM to new Burst Length via command MR0



**4.2.3 DDR Configure Module (ddr\_conf.sv)**

The modules implements the sequences of initialized commands send to DIMM to setup the operation after reset as well as CKE pin. This sequence is implemented using SV task and setup timing is parameterized in this module and top level module. There setup timing can be change as its instance without modified the code.

**4.2.4 DDR Burst ACT Module (ddr\_act.sv)**

The module implements the simplified FSM that control the timing of ACT command. The FSM enters to WAIT STATE if there is a new DIMM access request. This Wait state is to satisfy the tRRD constrain. The requested address is check against the table that stores opened row for each bank. If the accessing bank memory has not been activated, the FSM enters the CMD state to send an ACT command. If accessing to same opened row, the FSM enters Skip ACT command state (Note 1). If accessing to same bank and different row, the FSM enters PreCharge state (Note 2). In PreCharge state, it waits for the current CAS finish and adds another latency for write recover (tWR) or read to Precharge (tRTP) before sent out PreCharge command (Note 3). The FSM returns to ACT CMD state after tRP.

**4.2.4 DDR Burst CAS Module (ddr\_cas.sv)**

The module is to implement the simplified FSM below that controls the timing of CAS command. The FSM enters to Wait State after receives new ACT command has been issued. The Wait State is for tRCD, ACT to CAS latency, as well as tCCD, CAS to CAS latency. The Extra Wait also needed if there is Read to Write or Write to Read operation (tWTR). The state machine returns to Wait State, if there still has ACT command in queue. The queues are used in this module to keep track the new ACT command while executing the previous one. Each time the ACT command is retrieved from the queue, the remaining ACT commands are updated for waited time in queue. The wait time will check for tRCD and tCCD.



**4.2.5 DDR Burst RW Module (burst\_rw.sv)**

The module implements the simplified state machine below that control the timing to send out write or read data. The FSM enters to Wait State if there is new CAS command. The time in Wait State depends on whether read or write operation to satisfy the constraints of CL or CWL. Similar to Burst CAS Module, the module also uses a queue to capture CAS commands while executing the previous one.



**4.2.6 DDR Burst data Module (ddr\_data.sv)**

The module is a central to setup the addr/data/command for set\_cmd\_pins() method, and set\_strobe\_pins() and set\_wdata\_pin. The module captures the stimulus input and queues in two separate queues, one for Cas command and one for Write data. When the commands ready, the data/addr/setup are retrieved from the queue sand and packed into the DIMM command.

**4.3 DIMM Model (dimm\_model.sv)**

The module is implemented as part of test bench simulation to provide a mean to store and retrieve data. The modules samples the signal pins to capture address sent in ACT and CAS commands and the data in DQ. The data is written into an associate array using the captured address in ACT and CAS as an index. For the read operation, the data is retrieved for the array and call methods from DDR Interface to translated into pin transactions.

**4.4 Memory Check Module (memory\_check.sv)**

The module is to monitor Read/Write transaction between DDR Controller and DIMM Model via the DDR Interface. The module captures the Read/Write request from stimulus and stored the write data in an associate array. The read data is captured from DQ and DQS pins are check against the data in the array. The result will be logged into a text file.

**4.5 Assertion Checker (assert\_check.sv)**

The module contains the SVA concurrent assertions to check DDR timing and protocols. Included: Initialization, CKE, CL (Read Latency), CWL (Write Latency, tCCD , tWTP, tRTP, tRCD, tZQ, tREF, tRAS , tRP.

**4.6 Stimulus Generator (Rand\_stimulus.sv)**

The module uses SV randomization method and class construct to generate stimulus.

**4.7 DDR Clock Module (ddr\_clock.sv)**

The module is to generate a differential clock for DIMM and double frequency clock to align data to strobe pins.

**4.8 TB Interface (tb\_inteface.sv)**

The interface only declared the share signals between test bench modules and used as module port. There is contains no method in this interface.

**4.9 DDR Package (ddr\_package.pkg)**

The file contains all parameters, user-defines types. No function/task

**5. Test Results**

The test bench is simulated under QuestaSim with 500 test cases. All the test cases are PASS.

All SVAs are PASS.