Portland State
UNIVERSITY

ECE 571 – Introduction to SystemVerilog

Final Project Report

Design Verification of DDR3 Memory Controller

By

Ajna Bindu R
Jyothsna K
Rithvik M B
Vadiraja M N

OBJECTIVE

Design Verification of DDR3 Memory Controller using Mentor Graphics QuestaSim for Simulation and Mentor Graphics Veloce for Emulation.

DESIGN

The designed DDR3 used the **MICRON 1GB automotive DDR3** datasheet as reference.

VERIFICATION STRATEGY

An Object-Oriented Verification Environment was developed to verify the DDR3 Memory.

COMPONENTS OF VERIFICATION ENVIRONMENT:

- Transaction: Declaration of the signals which are required to be generated for DUT.
- Generator: Responsible for generating the stimulus by randomizing the transaction class and sends randomized data to the driver.
- Driver: Drives the generated stimulus to the DUT by assigning transaction values to interface signals.
- Monitor: Samples the interface signals and convert the signal level activity to transaction level to send to the scoreboard via mailbox.
- Scoreboard: Receives the sampled packet from Monitor and logic to store write data and compare read data with stored data is implemented.
- Interface(BFM): Will group the signals, specifies the direction of the signals (Modport).

TESTCASES:

Directed Testcases:

- Reset: Upon reset, the memory is cleared and contains garbage data.
  The Reset signal is asserted while the burst is in progression and the DUT handles it by discarding the current transaction and considering the data in the memory as garbage.

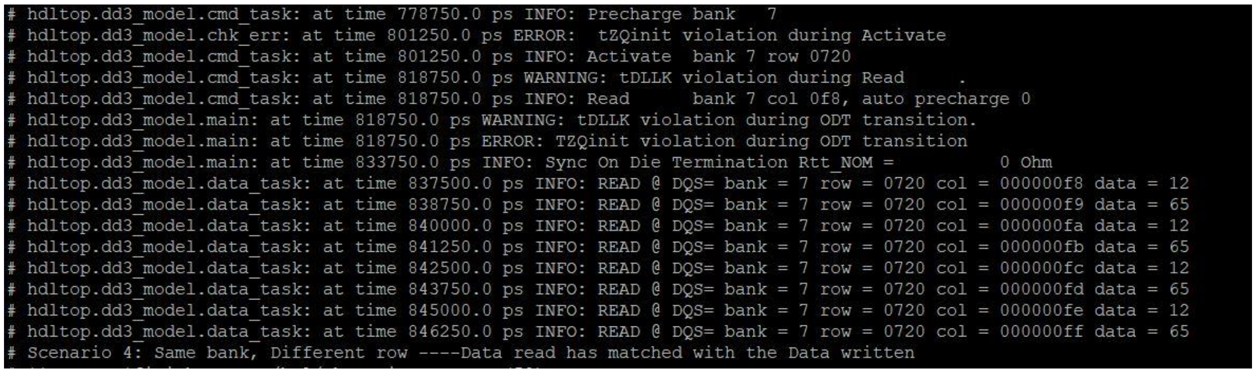
- Write and Read from the same address
  A single write operation to a particular Bank, Row and Column address followed by a read to the same Bank, Row, Column. The Data written is verified with the data read.

- Overwrite data to the same address and read from same location.
  A write operation to a particular Bank, Row and Column address followed by another write to the same Bank, Row, Column with new data. The latest data written is verified with the data read.

- Write data to same row but a different column
  This testcase is to verify that all columns in a row can be written and read without errors.

- Write to different row, same bank

  Data is written to a different row of the same bank and read. A data write and read match ensures different rows are functioning well.



```
# hdltop.dd3_model.cmd_task: at time 778750.0 ps INFO: Precharge bank   7
# hdltop.dd3_model.chk_err: at time 801250.0 ps ERROR:  tZQinit violation during Activate
# hdltop.dd3_model.cmd_task: at time 801250.0 ps INFO: Activate  bank 7 row 0720
# hdltop.dd3_model.cmd_task: at time 818750.0 ps WARNING: tDLLK violation during Read     .
# hdltop.dd3_model.cmd_task: at time 818750.0 ps INFO: Read      bank 7 col 0f8, auto precharge 0
# hdltop.dd3_model.main: at time 818750.0 ps WARNING: tDLLK violation during ODT transition.
# hdltop.dd3_model.main: at time 818750.0 ps ERROR: TZQinit violation during ODT transition
# hdltop.dd3_model.main: at time 833750.0 ps INFO: Sync On Die Termination Rtt_NOM =          0 Ohm
# hdltop.dd3_model.data_task: at time 837500.0 ps INFO: READ @ DQS= bank = 7 row = 0720 col = 000000f8 data = 12
# hdltop.dd3_model.data_task: at time 838750.0 ps INFO: READ @ DQS= bank = 7 row = 0720 col = 000000f9 data = 65
# hdltop.dd3_model.data_task: at time 840000.0 ps INFO: READ @ DQS= bank = 7 row = 0720 col = 000000fa data = 12
# hdltop.dd3_model.data_task: at time 841250.0 ps INFO: READ @ DQS= bank = 7 row = 0720 col = 000000fb data = 65
# hdltop.dd3_model.data_task: at time 842500.0 ps INFO: READ @ DQS= bank = 7 row = 0720 col = 000000fc data = 12
# hdltop.dd3_model.data_task: at time 843750.0 ps INFO: READ @ DQS= bank = 7 row = 0720 col = 000000fd data = 65
# hdltop.dd3_model.data_task: at time 845000.0 ps INFO: READ @ DQS= bank = 7 row = 0720 col = 000000fe data = 12
# hdltop.dd3_model.data_task: at time 846250.0 ps INFO: READ @ DQS= bank = 7 row = 0720 col = 000000ff data = 65
# Scenario 4: Same bank, Different row ----Data read has matched with the Data written
```

- Write to consecutive rows

  This testcase covers a scenario when a large packed array has to be written into memory, where it gets stored in consecutive addresses. The packed array is then read and verified with the array written.

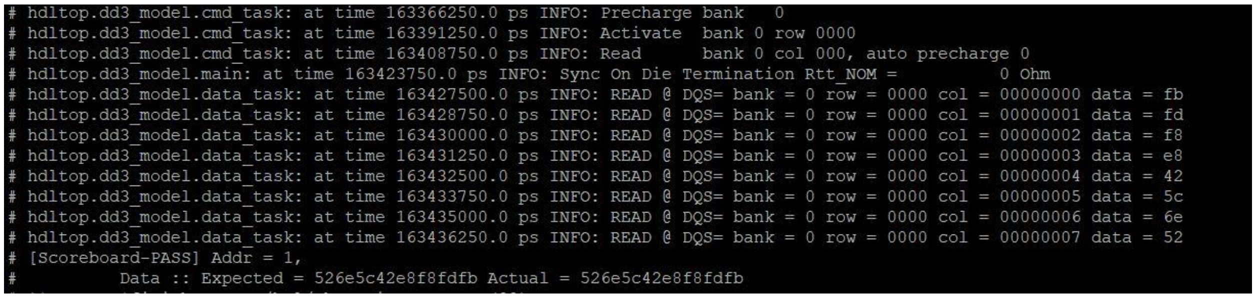
All the above testcases completely verify the working of a bank.

- Single write to every bank

  Writes and Reads to all the 8 banks are performed to verify all the banks.

- Generated random testcases

  Random valid signal, read/write signal, write data, read data and address are sent as packets to the DUT.
  During a write operation, the write data is stored in a local memory in the scoreboard.
  During a read operation, the data read from the DUT is matched with the data present in the local memory in the scoreboard. If the data matches, then a scoreboard pass message is displayed but if the data does not match, then the scoreboard throws an error which depicts that the DUT didn't function as expected.
  There may exist a case when data is being read from an address which has not been written before. The Simulator displays a message saying that "data is being read from an address which has not been written".



```
# hdltop.dd3_model.cmd_task: at time 163366250.0 ps INFO: Precharge bank   0
# hdltop.dd3_model.cmd_task: at time 163391250.0 ps INFO: Activate  bank 0 row 0000
# hdltop.dd3_model.cmd_task: at time 163408750.0 ps INFO: Read      bank 0 col 000, auto precharge 0
# hdltop.dd3_model.main: at time 163423750.0 ps INFO: Sync On Die Termination Rtt_NOM =          0 Ohm
# hdltop.dd3_model.data_task: at time 163427500.0 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000000 data = fb
# hdltop.dd3_model.data_task: at time 163428750.0 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000001 data = fd
# hdltop.dd3_model.data_task: at time 163430000.0 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000002 data = f8
# hdltop.dd3_model.data_task: at time 163431250.0 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000003 data = e8
# hdltop.dd3_model.data_task: at time 163432500.0 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000004 data = 42
# hdltop.dd3_model.data_task: at time 163433750.0 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000005 data = 5c
# hdltop.dd3_model.data_task: at time 163435000.0 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000006 data = 6e
# hdltop.dd3_model.data_task: at time 163436250.0 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000007 data = 52
# [Scoreboard-PASS] Addr = 1,
#         Data :: Expected = 526e5c42e8f8fdfb Actual = 526e5c42e8f8fdfb
```

Portland State UNIVERSITY

ASSERTIONS:
To make verification more effective, assertions has been implemented in verification plan.

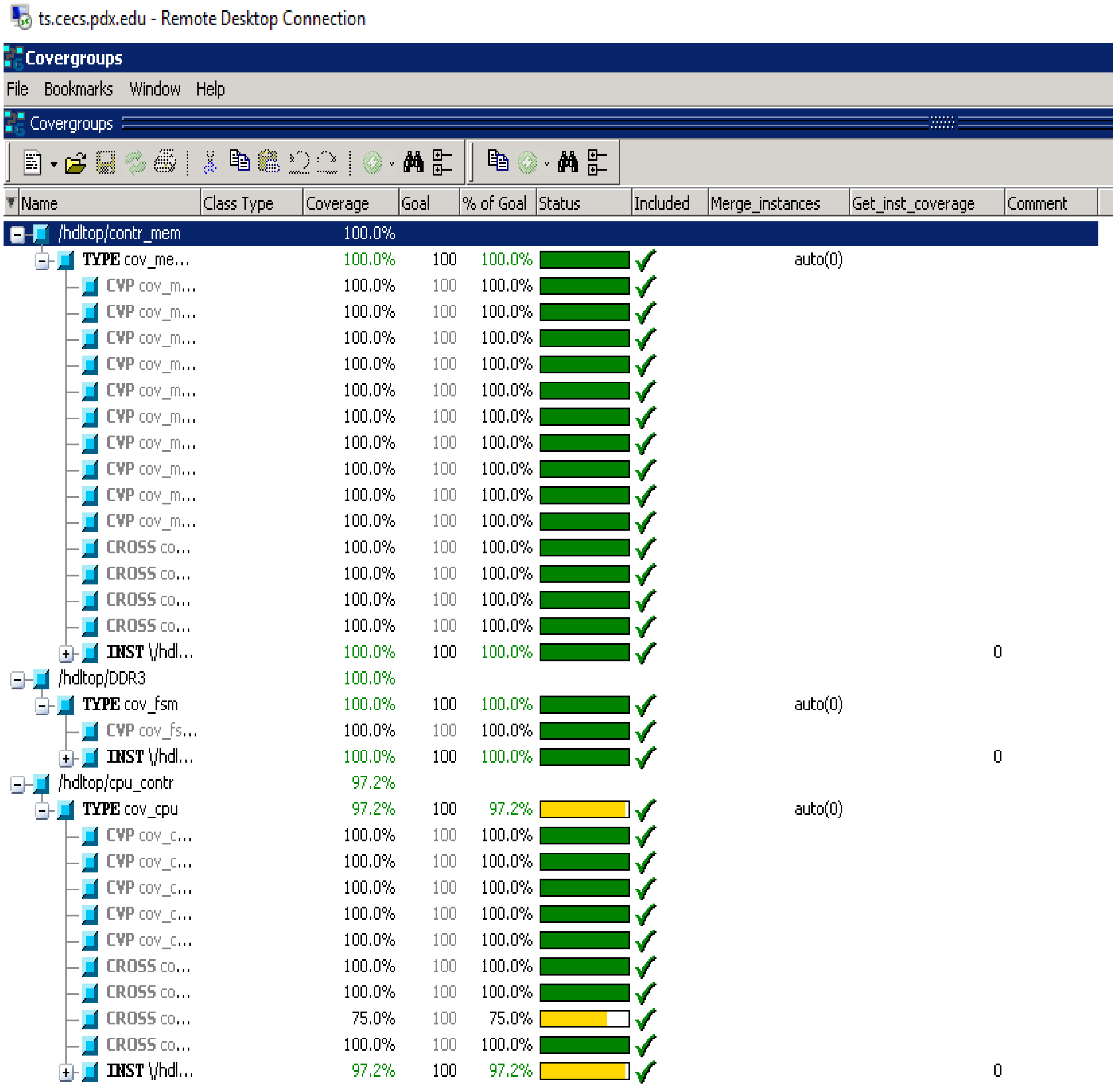
- Concurrent assertions in design unit are used to verify DDR3 timing parameters.
- During write operation, T_RCD (Row-to- Column Delay), T_WL (Write Latency), the data strobe (DQS) available after write latency has been verified.
- During read operation, T_CL (Read latency), T_RP (Pre-charge time), T_RC (Row cycle time) parameters are tested.
- Immediate assertions during activate, read and write operations are implemented to verify whether ras_n and cas_n signals are being asserted low at current simulation time.

FUNCTIONAL COVERAGE:
To verify how effectively the testcases are covering the design specifications, we have implemented coverage in verification environment.

- Coverage models are defined for interface and top-level modules to verify input signals.
- These models are defined using cover group construct.
- Covergroup consists of coverpoints (input signals) which are sampled at every posedge of the clock. During simulation, coverage graph of the signals of the module are generated.
- Transition coverage for FSM (DUT) is calculated to verify the transitions between states.

Portland State
UNIVERSITY

COVERAGE STATISTICS:



ts.cecs.pdx.edu - Remote Desktop Connection

Covergroups

File Bookmarks Window Help

Covergroups

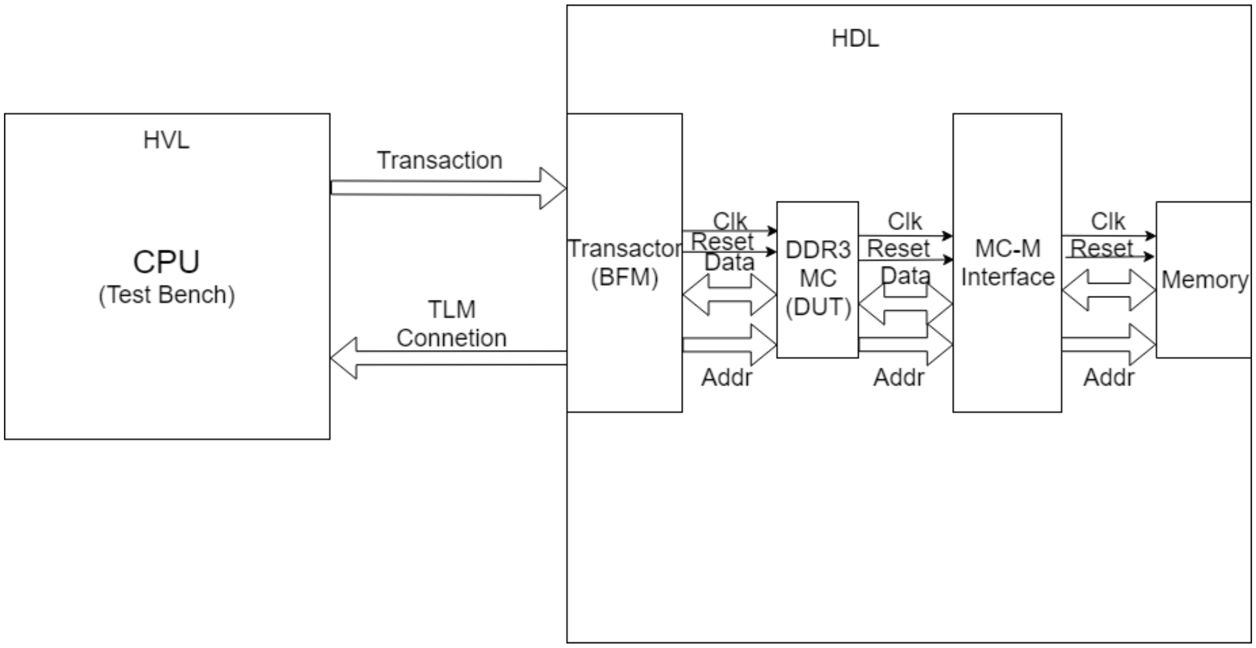
Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/hdltop/contr_mem		100.0%							
TYPE cov_me...		100.0%	100	100.0%	🟩	✓	auto(0)		
CVP cov_m...		100.0%	100	100.0%	🟩	✓			
CVP cov_m...		100.0%	100	100.0%	🟩	✓			
CVP cov_m...		100.0%	100	100.0%	🟩	✓			
CVP cov_m...		100.0%	100	100.0%	🟩	✓			
CVP cov_m...		100.0%	100	100.0%	🟩	✓			
CVP cov_m...		100.0%	100	100.0%	🟩	✓			
CVP cov_m...		100.0%	100	100.0%	🟩	✓			
CVP cov_m...		100.0%	100	100.0%	🟩	✓			
CVP cov_m...		100.0%	100	100.0%	🟩	✓			
CVP cov_m...		100.0%	100	100.0%	🟩	✓			
CROSS co...		100.0%	100	100.0%	🟩	✓			
CROSS co...		100.0%	100	100.0%	🟩	✓			
CROSS co...		100.0%	100	100.0%	🟩	✓			
CROSS co...		100.0%	100	100.0%	🟩	✓			
INST \/hdl...		100.0%	100	100.0%	🟩	✓		0	
/hdltop/DDR3		100.0%							
TYPE cov_fsm		100.0%	100	100.0%	🟩	✓	auto(0)		
CVP cov_fs...		100.0%	100	100.0%	🟩	✓			
INST \/hdl...		100.0%	100	100.0%	🟩	✓		0	
/hdltop/cpu_contr		97.2%							
TYPE cov_cpu		97.2%	100	97.2%	🟨	✓	auto(0)		
CVP cov_c...		100.0%	100	100.0%	🟩	✓			
CVP cov_c...		100.0%	100	100.0%	🟩	✓			
CVP cov_c...		100.0%	100	100.0%	🟩	✓			
CVP cov_c...		100.0%	100	100.0%	🟩	✓			
CVP cov_c...		100.0%	100	100.0%	🟩	✓			
CROSS co...		100.0%	100	100.0%	🟩	✓			
CROSS co...		100.0%	100	100.0%	🟩	✓			
CROSS co...		75.0%	100	75.0%	🟨	✓			
CROSS co...		100.0%	100	100.0%	🟩	✓			
INST \/hdl...		97.2%	100	97.2%	🟨	✓		0	

EMULATION

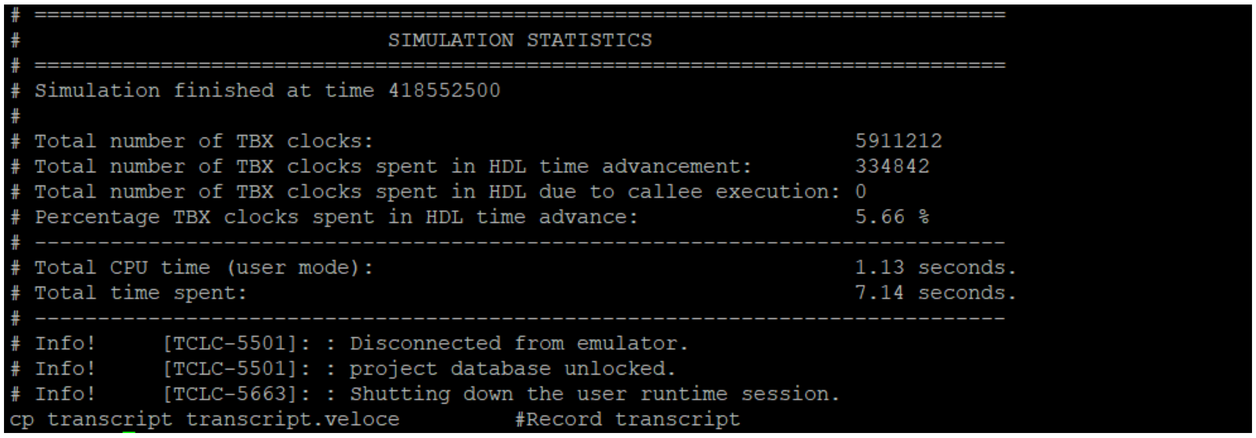
- The Emulation was implemented on the TestBench eXpress(TBX) mode.
- Followed co-modelling technique, where-in the Hardware Verification Language (HVL) consisting of CPU(test bench) is on the server side and the Hardware Description Language(HDL) consisting of Bus Functional Model(BFM), Design Under Test(DUT) and Memory model is on Veloce emulator.
- The communication between HVL and HDL is done through the task and function written in BFM.

- The HVL cannot have any clock , # delays and wait statement with a fixed delay.
- During simulation in puresim mode, we need to access the tasks and functions in the BFM. Hence we declared a virtual interface of the BFM to access the tasks and functions in it.
- The clock is generated in the HDL.
- Pragmas are written to the interface, tasks and functions. This will create a data structure in the emulator, so that we can access the task and function in the HVL during emulation.

BLOCK DIAGRAM:



STATISTICS:



```
# ==============================================================================
#                              SIMULATION STATISTICS
# ==============================================================================
# Simulation finished at time 418552500
#
# Total number of TBX clocks:                                      5911212
# Total number of TBX clocks spent in HDL time advancement:        334842
# Total number of TBX clocks spent in HDL due to callee execution: 0
# Percentage TBX clocks spent in HDL time advance:                 5.66 %
# ------------------------------------------------------------------------------
# Total CPU time (user mode):                                      1.13 seconds.
# Total time spent:                                                7.14 seconds.
# ------------------------------------------------------------------------------
# Info!     [TCLC-5501]: : Disconnected from emulator.
# Info!     [TCLC-5501]: : project database unlocked.
# Info!     [TCLC-5663]: : Shutting down the user runtime session.
cp transcript transcript.veloce         #Record transcript
```

According to the statistics, we have achieved a HDL advancement of 5.66%. This means that we only spend 5.66% of our time in hardware.

We have achieved a speed up of 2%.
The percentage of the HDL advancement is less because we are doing serval transaction in HVL before going to the HDL.

BUGS

- The design and verification were being carried out parallelly.
- In the initial stages of the project, data written into an address, when read was obtained in the reverse order. This was due to an improper burst. The design team fixed this soon.
- An unusual bug which the simulation displays as a warning. Very rarely, during a data read, out of the 64 bits of data read, 8 bits of data are z's. This is because the data on the DQ signal is synchronized with the clock rather than the DQ strobe signal. The simulation does not wait for the data to be stable. But the same scenario when synthesized does not happen as the data is latched only when it is stable.

WORK DISTRIBUTION

Understanding the Design: Ajna Bindu, Jyothsna, Rithvik, Vadiraja
Assertions: Jyothsna and Ajna Bindu
Functional Coverage: Jyothsna
Testbench Environment: Vadiraja
Emulation on VELOCE: Rithvik

CHALLENGES

- We have faced a challenge of using tasks during Emulation and observed that tasks should be executed on edge clock and a wait statement must be preceded by an edge clock.
- Also, to maintain the uniformity of edge clock within a task, we must consider same active edge for all tasks.
- To maintain uniformity of edge clock among tasks, edge of an input signals cannot be used as event signal.
- Referring hierarchical modules at top level.

CONCLUSION

- Effective implementation of testcases to verify the DDR3 memory using classes, functional coverage, assertions.
- Efficiently utilized the SystemVerilog constructs such as virtual interfaces, tasks, randomization, cover groups, assertions to enhance verification strategy.
- From the challenges faced, we have learned how to make use of verification constructs more effectively so as to perform better verification.

Portland State UNIVERSITY

FURTHER SCOPE

If extra time had been allocated, we would have implemented:

- Verification to verify ECC bits in data at chip level.
- Testcases to check if any of the random address in the memory received the data while performing a write to a particular address (COUPLING FAULT).

REFERENCES

- 1GB Automotive DDR3L SDRAM Micron datasheet
- http://www.verificationguide.com/p/systemverilog-tutorial.html
- http://www.testbench.in/
- Emulation: PSU Veloce Examples