Portland State
UNIVERSITY

ECE571

Project Report
DDR3 SDRAM Controller

Rahul, Sai, Suraj, Tejas
Portland State University
ECE571

Document Info

Project	DDR4 SDRAM Controller				
Document Name	Proposal.pdf				
Rev No	0.1	**Date**	Mar-18-2018	**Maturity**	Release
Owner 1	Sai T Bodanki bodanki@pdx.edu				
Owner 2	Suraj Avinash ssathy34@pdx.edu				
Owner 3	Sri Rahul Challagulla sri1@pdx.edu				
Owner 4	Tejas Chavan techavan@pdx.edu				
Authors	Tejas, Suraj, Rahul, Teja				
Distribution	Roy Kravitz				

Revision History

Version No.	Date	Change Description	Author	Reviewed by
0.1	03-17-2018	Created.	Sai T Bodanki	Tejas Chavan
0.2	03-18-2018	Edited	Tejas Chavan	Sai Bodanki
0.3	03-18.2018	Added File Structure	Rahul	Sai Bodanki

Contents

1. **The Goal** 5
2. **Scope** 5
3. **Assumptions and Resources** 5
4. **Design** 6
4.1. **Architecture** 6
4.2. **FSM's** 8
4.3. **Port List** 12
5. **Verification** 13
5.1. **Plan** 13
5.2. **Framework** 13
5.3. **Gaussian LFSR** 14
5.4. **Coverage** 14
6. **Emulation** 15
6.1. **Changes for Emulation** 15
6.2. **Results** 15
7. **Bottle Necks** 17
8. **Extended Scope** 17
9. **Conclusion** 17
10. **File Structure** 18

1. The Goal

To Design and Verify a DDR3 SDRAM Controller.

2. Scope

Not all the features of JEDS79-4B are the scoped in the project. The features we currently propose are the following.

- Power-up Sequencing
- Configure DDR3 Registers and DLL
- ZQ Caliberation
- Activate
- Write - Closed Page Policy
- Read – Closed Page Policy
- Refresh
- Precharge
- NOP
- Self-Refresh

The features would be however being not limited to mentioned above, but will be extended based on available time.

3. Assumptions and Resources

- The Memory Model which is 1GB DDR3-800 arranged as 128M x 8 has been leveraged from Micron. The file also included a parameter file (1024Mb_ddr3_parameter.sv) that contains timing specifications and delays according to which the memory operators.
- Our design is completely based on the Micron 1GB DDR3 data sheet. The timing specifications, mode of operations exactly match the operation of an ideal DDR3 memory.

4. Design

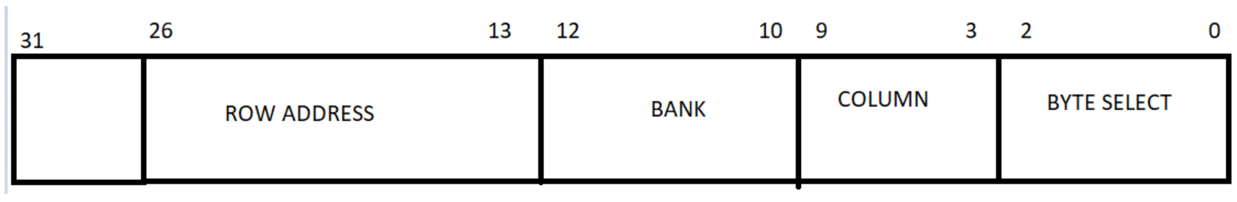
The Memory Controller design is completely based the timing specifications and operation of the memory model. DDR3 SDRAM uses a double data rate architecture to achieve high-speed operation. The double data rate architecture is an 8*n*-prefetch architecture with an interface designed to transfer two data words per clock cycle at the I/O pins.

4.1. Architecture

- The Architecture of the controller is completely based on implementation of Finite state machines to perform various operations. The controller performs various operations starting from Power up sequence, ZQ Calibration, Mode register load, Activate, precharge, Read/ Write Burst.
- As the memory model is DDR3-800, the real clock rate is 400MHz. Thus one clock cycle equals 2.5ns.
- To follow the timing specifications and various delays, an internal down counter has been considered. In the beginning of each state, appropriate max count is fed to the counter depending upon the delay. Once the count reaches 0, the system moves to the next state mentioned.
- Elaborating the timing delays, the memory model follows 6-6-6 timing specifications. That is Tcl = Trcd = Trp = 6.
- One of the major aspect of the operation is the burst mode of data transfer. During each read and write operation, the data transfer takes place in the bursts of 8 bits.

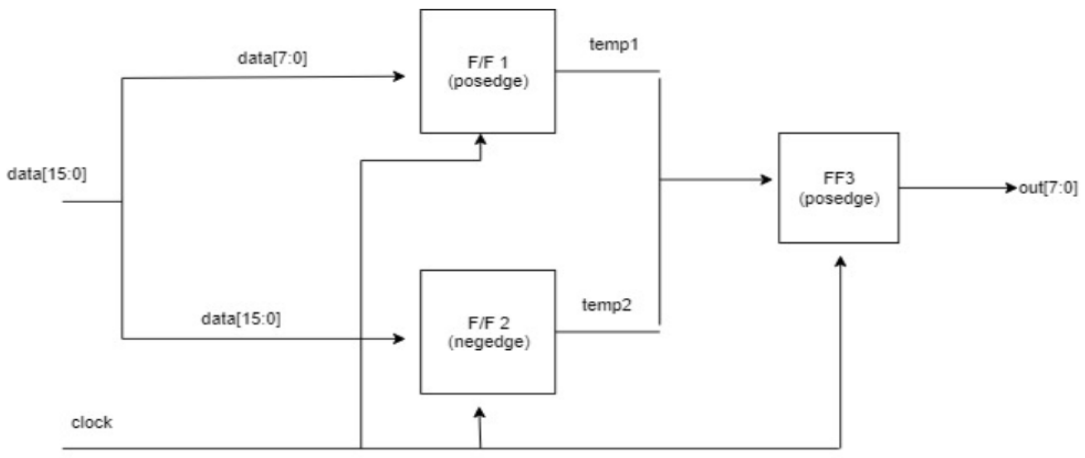
ADDRESS DECODING

- The memory organization is 128M x 8. Thus there are **8 DQ bits.** Therefore 3 LSBs of 32 bit address are allotted for byte select.
- The memory requires 7 bits for column. Thus after activation, the memory selects 1 column from the 128 columns / bank. This is how it receives 1 bit per bank. Thus 8 bits per clock edge from 8 banks. Address[9:3] are allotted for columns.
- Address[12:10] are allotted for bank selection. As there are 8 banks, we need 3 bits for selection.
- The rest 14 bits Address[26:13] are allotted for row selection.



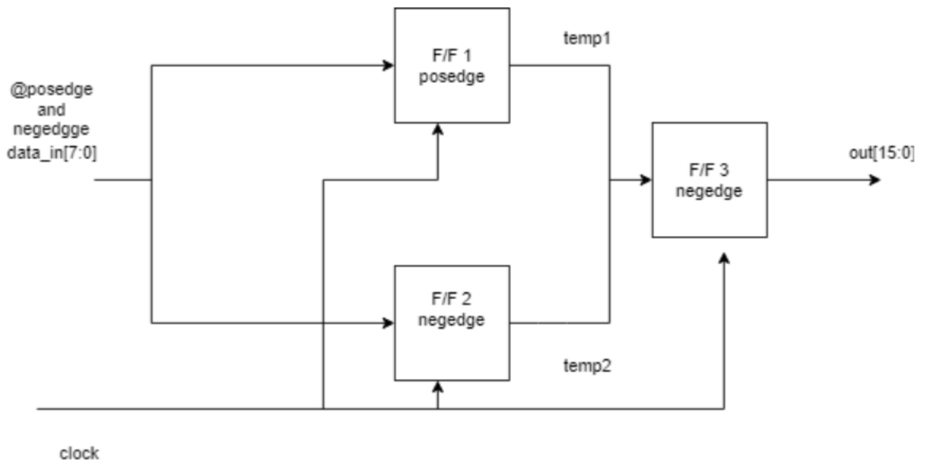
WRITE BUFFER:

- Whenever the CPU wants to perform a read operation, it sends 16 bits of data per clock cycle to the memory controller. These 16 bits are stored in the a WRITE buffer. The controller upon receiving the data, splits it into 2, 8-bit chunks and sends it to the memory as 8 bits per clock edge. Thus it takes in 4 clock cycles to send all the 64 bits of data to the memory. This operation is completely taken care in the WRITEBURST state of the FSM.
- To implement this operation, the hardware required are 3 flipflops. First flipflop samples the 8 LSBs on the posedge and sends it ahead and the second flipflop samples the 8 MSBs. Once the 16 bits are split into 2 8 bit chunks, they are sent to the memory using the 3rd Flipflop.



READ BUFFER:

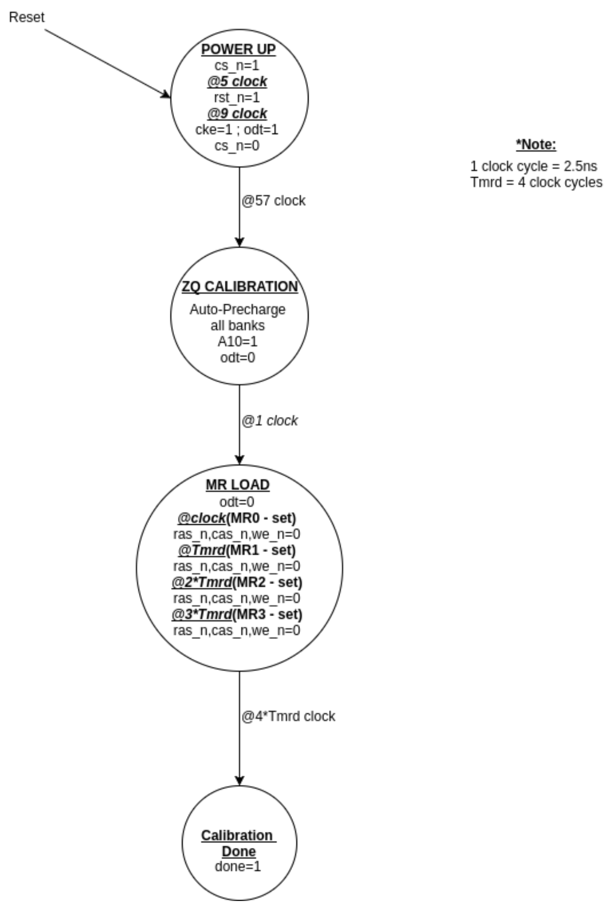
- Whenever the CPU wants to perform a read operation, upon receiving the complete address, it starts sending 16 bits into the read buffer. As the memory is of x8 configuration, it sends 8 bits per clock edge as an output. These 8 bits are captured by two different flipflops and are sent to the CPU as 16 bits. This the CPU receives 64 bits of data.



4.2. FSM's

- The design consists of 2 FSMs : 1. Powerup and Calibration 2. Read/Write

A. PowerUp and Calibration FSM.



- **POWER_UP state:**

On receiving the reset signal from the CPU, the controller moves to powerup state where chip select is disabled. After 5 clock cycles reset is disabled followed by clock enable = 1 and enabling of chip select 4 clock cycles later. After 57 clock cycles the system moves to ZQ calibration state.

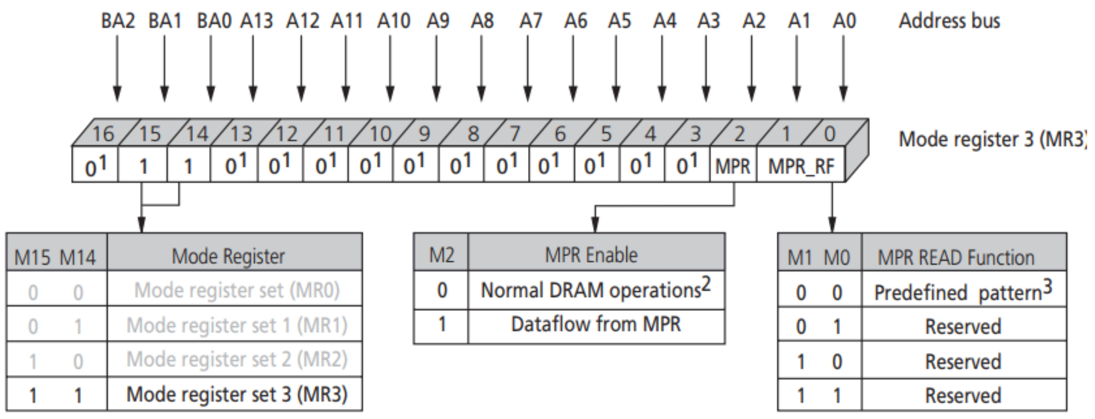
- **ZQ_CALIBRATION:**

In this state all banks are auto precharged by setting the A10 bit high. On die termination is disabled.

- **MR_LOAD**

  Mode registers (MR0–MR3) are used to define various modes of programmable operations of the DDR3 SDRAM. A mode register is programmed via the mode register set (MRS) comma nd during initialization, and it retains the stored information (except for MR0[8], which is self-clearing) until it is reprogrammed, RESET# goes LOW, the device loses power.

- The base register, MR0, is used to define various DDR3 SDRAM modes of operation. These definitions include the selection of a burst length, burst type, CAS latency, operating mode, DLL RESET, write recovery, and precharge power-down mode
- The mode register 1 (MR1) controls additional functions and features not available in the other mode registers: Q OFF (OUTPUT DISABLE), TDQS (for the x8 configuration only), DLL ENABLE/DLL DISABLE, RTT,nom value (ODT), WRITE LEVELING, POSTED CAS ADDITIVE latency, and OUTPUT DRIVE STRENGTH.
- The mode register 1 (MR1) controls additional functions and features not available in the other mode registers: Q OFF (OUTPUT DISABLE), TDQS (for the x8 configuration only), DLL ENABLE/DLL DISABLE, RTT,nom value (ODT), WRITE LEVELING, POSTED CAS ADDITIVE latency, and OUTPUT DRIVE STRENGTH.


BA2 BA1 BA0 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
Address bus
16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Mode register 3 (MR3)
0[1] 1 1 0[1] 0[1] 0[1] 0[1] 0[1] 0[1] 0[1] 0[1] 0[1] 0[1] 0[1] MPR MPR_RF

M15	M14	Mode Register
0	0	Mode register set (MR0)
0	1	Mode register set 1 (MR1)
1	0	Mode register set 2 (MR2)
1	1	Mode register set 3 (MR3)

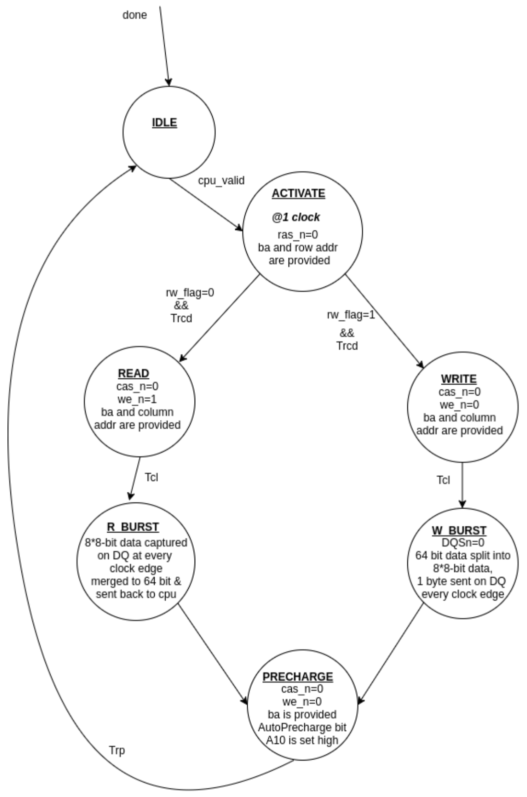
M2	MPR Enable
0	Normal DRAM operations[2]
1	Dataflow from MPR

M1	M0	MPR READ Function
0	0	Predefined pattern[3]
0	1	Reserved
1	0	Reserved
1	1	Reserved

Notes:
1. MR3[16 and 13:3] are reserved for future use and must all be programmed to 0.
2. When MPR control is set for normal DRAM operation, MR3[1, 0] will be ignored.
3. Intended to be used for READ synchronization.

- **Calibration__Done**:
  This is basically a dummy state after which the FSM switches to the main read write FSM

B. READ/WRITE FSM:


done
IDLE
cpu_valid
ACTIVATE
@1 clock
ras_n=0
ba and row addr are provided
rw_flag=0 && Trcd
rw_flag=1 && Trcd
READ
cas_n=0
we_n=1
ba and column addr are provided
WRITE
cas_n=0
we_n=0
ba and column addr are provided
Tcl
Tcl
R_BURST
8*8-bit data captured on DQ at every clock edge merged to 64 bit & sent back to cpu
W_BURST
DQSn=0
64 bit data split into 8*8-bit data, 1 byte sent on DQ every clock edge
PRECHARGE
cas_n=0
we_n=0
ba is provided
AutoPrecharge bit A10 is set high
Trp

- **IDLE :**
  As the design moves from the power_up FSM to the Read/Write FSM, it enters a dummy state IDLE. It remains in idle state unless it receives any transaction request from CPU.

- **ACTIVATE**:
  The ACTIVATE command is used to open (or activate) a row in a particular bank for a subsequent access. The value on the BA[2:0] inputs selects the bank, and the address provided on inputs A[*n*:0] selects the row. This row remains open (or active) for accesses until a PRECHARGE command is issued to that bank.
  A PRECHARGE command must be issued before opening a different row in the same bank.

- **READ**:
  - The READ command is used to initiate a burst read access to an active row. The address provided on inputs A[2:0] selects the starting column address, depending on the burst length and burst type selected (see Burst Order table for additional information). The value on input A10 determines whether auto precharge is used.
  - In read mode CAS is strobed low and column and bank address are fed to the controller.

- **WRITE:**
  - The WRITE command is used to initiate a burst write access to an active row. The value on the BA[2:0] inputs selects the bank. The value on input A10 determines whether auto-precharge is used. The value on input A12 (if enabled in the MR) when the WRITE command is issued determines whether BC4 (chop) or BL8 is used.
  - Write command is given by strobing the RAS low and providing the bank and column address.

- **R_BURST:**
  - Operation in this state is basically performed using the read buffers.
  - 8 Bits are captured on DQ of each clock edge upon entering the R_Burst state. These 8 bits are merged to form 16 bits and eventually 64 bits throughout the 4 clock cycles and sent to the CPU one clock cycle later.

- **W_BURST:**
  - Similar to R_BURST, the write burst is performed with the write buffer.
  - When DQSn is set to 0, 64 bit data is split into 8, 8-bit chunks along the 8 clock cycles and 8 bits are sent to the CPU on each clock edge of the subsequent 4 clock cycles.

- **PRECHARGE:**
  - The PRECHARGE command is used to de-activate the open row in a particular bank or in all banks. The bank(s) are available for a subsequent row access a specified time (tRP) after the PRECHARGE command is issued, except in the case of concurrent auto precharge.
  - A READ or WRITE command to a different bank is allowed during a concurrent auto precharge as long as it does not interrupt the data transfer in the current bank and does not violate any other timing parameters. Input A10 determines whether one or all banks are precharged. In the case where only one bank is precharged, inputs BA[2:0] select the bank; otherwise, BA[2:0] are treated as “Don’t Care.”
  - After a bank is precharged, it is in the idle state and must be activated prior to any READ or WRITE commands being issued to that bank.

4.3. Port List

- The Port list is basically specified in the interface. There are 2 interfaces :
  - mem_if : Interface between Controller and memory
  - mem_intf : Interface between CPU and Controller

A. CPU Controller Interface Ports

Port Names	Size	I/O Types for Controller	I/O Type for CPU	Description
i_cpu_reset;	1 bit	Input	Output	Reset signal from CPU
i_cpu_addr;	32 bits	Input	Output	Address signal from CPU
i_cpu_cmd;	1 bit	Input	Output	Read or Write Command
i_cpu_wr_data;	64 bits	Input	Output	Data to be written during write operation
i_cpu_valid;	1 bit	Input	Output	Valid signal from CPU
i_cpu_enable;	1 bit	Input	Output	Enable signal from CPU
i_cpu_dm;	1 bit	Input	Output	Data Mask
i_cpu_burst;	4 bits	Input	Output	Burst length configuration
o_cpu_rd_data;	64 bits	Output		Data to CPU on read request
o_cpu_data_rdy;	1 bit	Output	Input	Data ready signal
o_cpu_rd_data_valid;	1 bit	Output	Input	Cpu read data valid signal

B. Memory-Controller interface Ports.

Port Names	Size	I/O Types for memory	I/O Types for controller	Description
rst_n;	1 bit	Input	Output	Reset signal from Controller
ck;	1 bit	Input	Output	Clock
ck_n;	1 bit	Input	Output	clock
cke;	1 bit	Input	Output	Clock enable
cs_n;	1 bit	Input	Output	Chip select from controller
ras_n;	1 bit	Input	Output	Row address strobe from memory
cas_n;	1 bit	Input	Output	Column address strobe from memory
we_n	4 bits	Input	Output	Write enable signal from controller
ba	3 bits	Input	Output	Bank address
addr	27 bits	Input	Output	Total address
dq	8 bit	Inout	Inout	Data from memory
dqs	1 bit	Inout	Inout	Data strobe signal
dqs_n	1 bit	Inout	Inout	Data strobe neg
tdqs_n	1 bit	Input	Output	Terminating data strobe signal
odt	1 bit	Input	Output	On die termination
dm_tdqs	1 bit	Inout	Inout	

5. Verification

Verification is a method in which a design is verified compared to a given design specification before tape-out. This occurs parallel with the development of the design and can start from the time the design architecture/micro architecture definition happens. The main goal of verification is to ensure functional correctness of the design before the tape out.

5.1. Plan

- First, we generate the 64-bits of data required for write Operation, and address in our case 27 bits required for DRAM Architecture along with control signals from the specification of Micron Data Sheet and send the Data and Address to DDR3 Controller.
- Second, The DDR3 Controller, decodes the address in to Row, Column, bank, Chip Select depending on the DRAM Architecture, and send the write request to DRAM Memory Model with required timing Specifications.
- Third, When the write operation is done, to verify we send the same address to DDR3 controller and wait for the data.
- Fourth, we compare the data sent is equal to data received. If the data received is same as data sent then we go back to write operation with different memory address and repeat the process. the data sent is not equal to data received then we raise the flag and wait for some time and the issues still remains i we stop the simulation and debug the code.

5.2. Framework

The Below figure shows the framework we used to verify the DUT (DDR3 Controller).

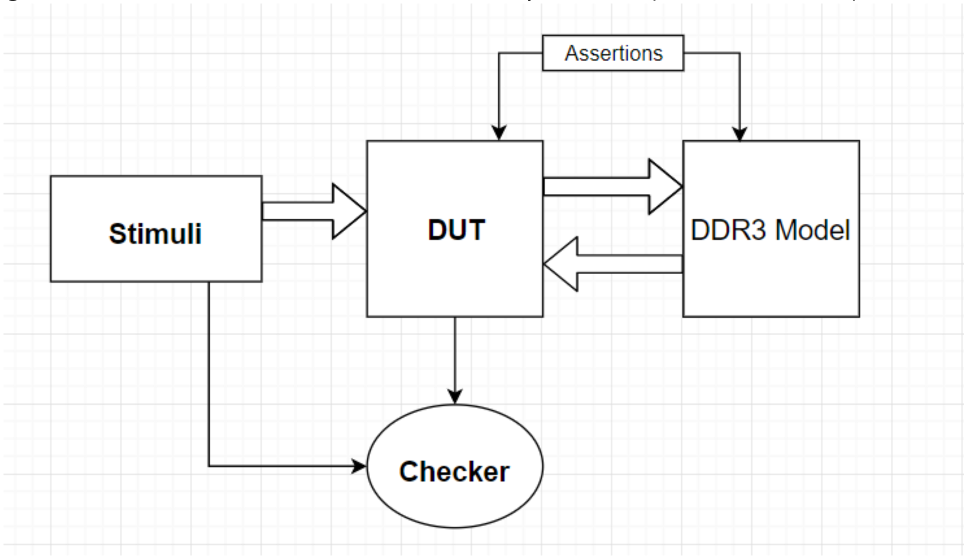


Fig-5.2.1-Block Diagram for verification Framework

5.3. Gaussian LFSR

- **linear-feedback shift register (LFSR)** is a shift register whose input bit is a linear function of its previous state
- Gaussian LFSR is a Pseudo Random Generator, where we have generated address 27 bits in our case , to specific values we want in short we can say we constrained the random generator to follow a certain pattern in order to test our DUT. **Fig-5.3.1** shows the normal distribution of a random patterns generated by the Gaussian LFSR

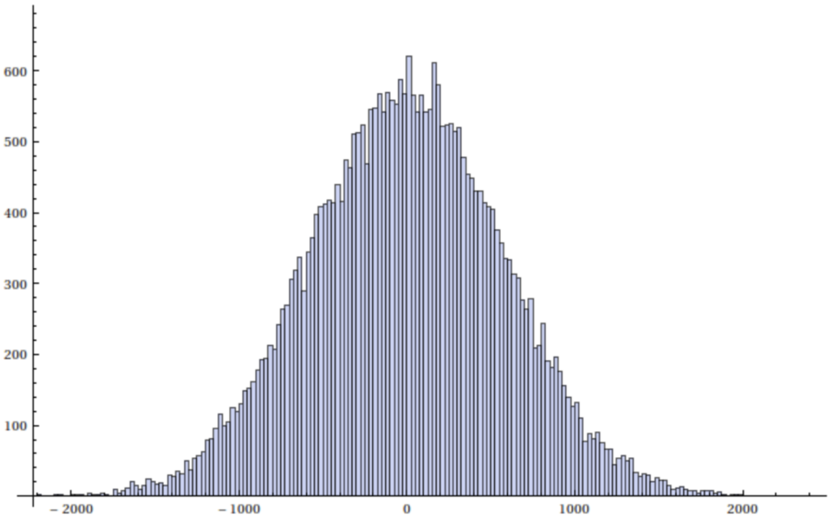


Fig-5.3.1- Normal Distribution of Generated Random Numbers using Gaussian LFSR

5.4. Coverage

The Coverage We have attained with Verification Plan using Gaussian Pseudo Random Generator for address (LFSR) are as below

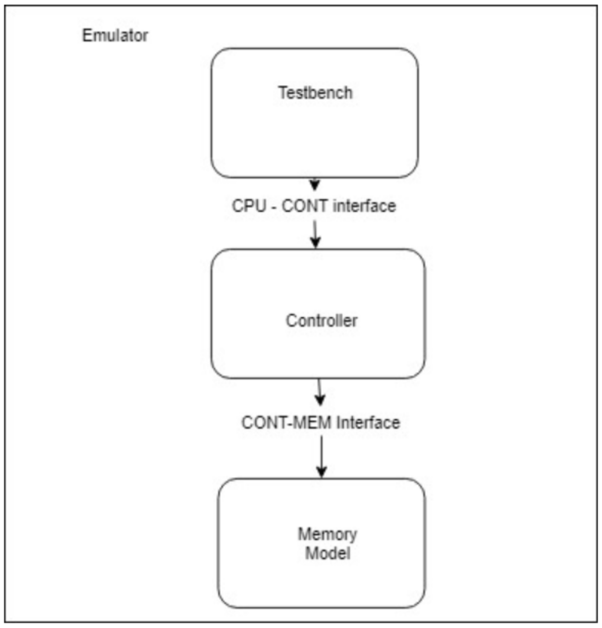
- **Latency-30 Clock Cycles**

- **Coverage DUT -100%**
  1) **Memory Model -60%(Micron)**
  2) **Memory Model -80% (Simpler BFM for Emulation)**

6. Emulation

6.1. Changes for Emulation

- As mentioned earlier, the memory model leveraged from micron was not synthesizable. Due to which we faced a few issues emulating the design.
- We came up with a solution to design a simpler DDR3 controller based on a synthesizable memory model. Thus we were able to move completely towards standalone implementation as our CPU, Controller and Memory model, all were synthesizable.

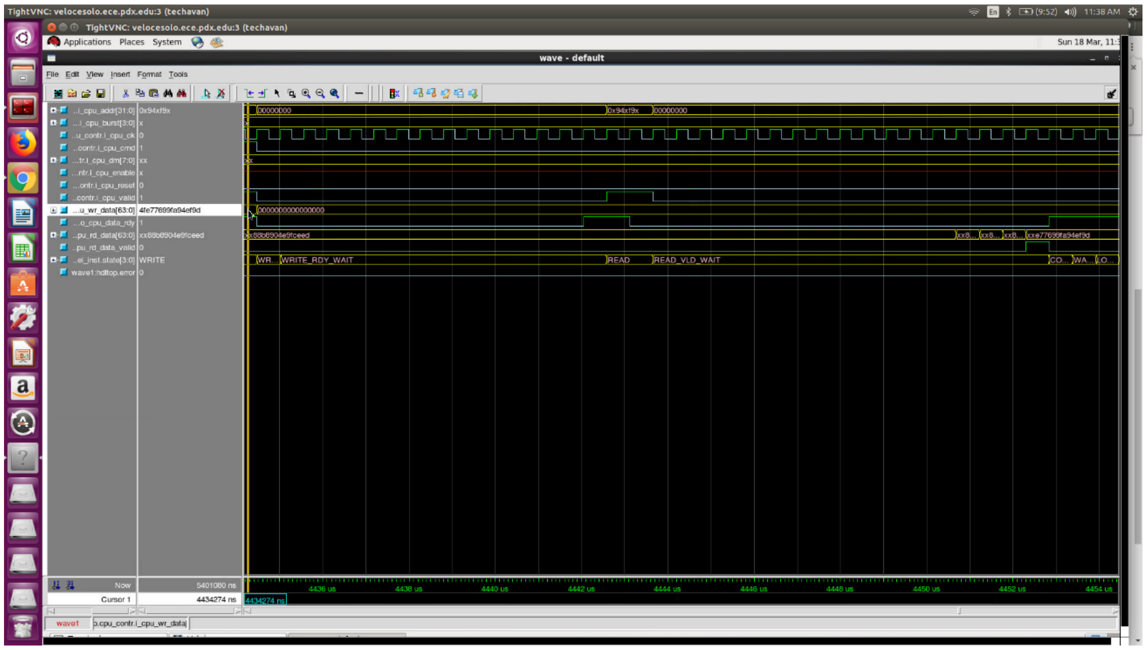


6.2. Results

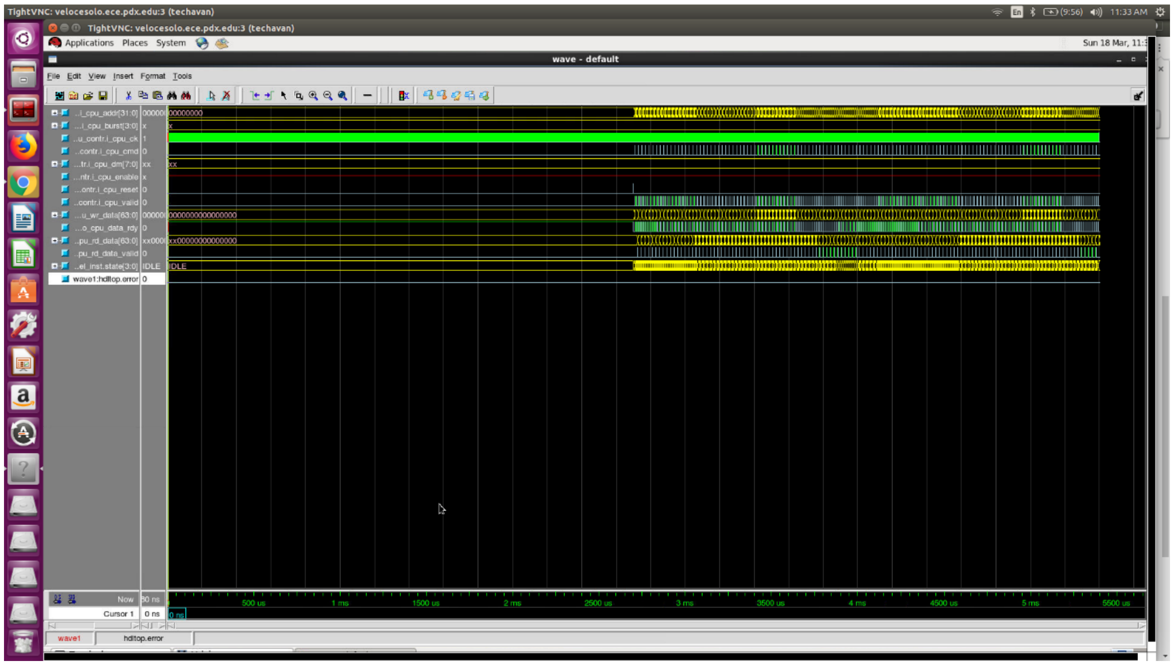
- The speedup that we were able to achieve using the VeloceSolo emulator was drastic comparing the time it took to execute the combinations generated by the LFSR on the simulator.
- It took around 10 mins to implement 5000 transactions generated by the LFSR on the simulator. But it only cost us a minute or so with the emulator.

ECE 571 Final Project

A. One Read Write Cycle.



B. All Transactions.


TightVNC: velocesolo.ece.pdx.edu:3 (techavan)
Applications Places System
wave - default
File Edit View Insert Format Tools
..i_cpu_addr[31:0] 00000
..u_contr.i_cpu_ck 1
..contr.i_cpu_cmd 0
..el_inst.state[3:0] IDLE
wave1:hdltop.error 0
Now 50 ns
Cursor 1 0 ns

7. Bottle Necks

- Using the micron memory model, we had to make sure we met T-setup time for all the data transactions to make the transfer work. Dummy clock was created and phase shifted to 90deg to pass the data and address between the posedge and the negedge of the clock cycle.
- During the power up mode, using the counter to explicitly calculate the clock cycles required for the memory model to be powered up and calibrated based on the timing specifications of the model with clock speed provided.
- For emulation - as we planned to make a standalone design, we had to make sure to keep the design of the cpu and the memory controller not to have any latches. As we encountered issues like such latches and other non synthesizable constructs in the design, we fixed them one by one in the design.

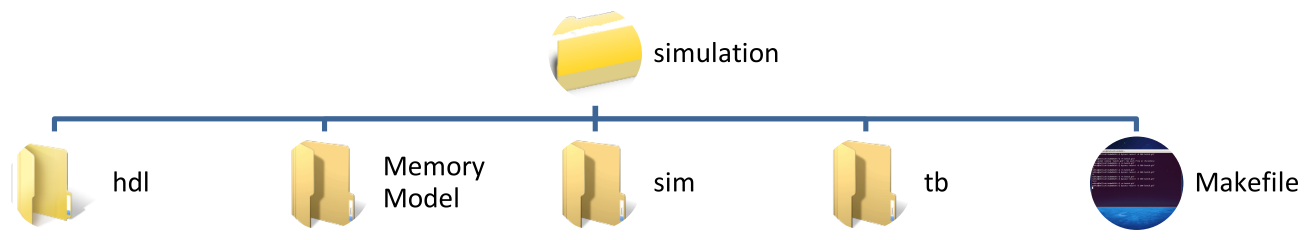
8. Extended Scope

- Perform Verification using the Testbench environment.
  As our implementation was more design oriented than verification, we would like to extend this project by performing extensive testing and verification using major verification concepts.
- Perform Self-Refresh operation.

9. Conclusion

Our initial plan was to support most of the features of DDR3 as we proposed. Unfortunately, because of time constraints, bottlenecks faced and non-synthesizable memory model of micron, we shifted to a simpler design of memory controller and cpu design with a simple handshaking model to emulate the top design. It was a good that we understood the design well and making it to work as per the timing specs provided by Micron. Although we added ZQ calibration which wasn't proposed, given some more time we would have implemented Refresh and Out of sequence to improve the robustness of the design. Overall, we hope that we brought out the best out of the things that we learnt in the class through this project and it was a good experience being novice engineers.

10. File Structure



10.1. HDL File

- DUT.sv- it's Module for Memory Controller DDR3 , where it decodes the address in to row, column ,bank address and it follows timing specification of DDR3 model Micron Data Sheet
- DUT_pkgs.sv-it contains the parameters for time Delay like RAS, CAS, TRP etc and also contain the parameters like data width, Burst length, Row, Column address length etc.
- IDDR.sv- it is write burst state where the Memory Controllers send the data to DRAM memory Model in the 8 burst in every posedge and negedge of clock.
- ODDR.sv- it is Read burst state where the Memory Controllers gets the data from DRAM memory Model in the 8 burst in every posedge and negedge of clock.
- Timer.sv-where the counts gets incremented in every clock edge and responsible for state transition.
- If_top.sv it gives cpu clock to all the above modules

10.2. Memory Model File

- 1024Mbdr3_parameters.sv-it contain the parameters required for MICRON DDR3 model memory to function and set it parameters
- ddr3.sv-it contains the memory model 1GB from Micron.

10.3. TB file

- cpu.sv-it contain the test-bench which generates the data and address and gives it to memory controllers and checks the data coming from the Memory controller and edifies it with it's memory (Test-bench)
- LFSR.sv- it contains Gaussian pseudo random generator which generates 27 bits of data and 64 bits of data for the CPU.
- Top.sv-CPU Clock

Deliverables

- Complete Source Codes
- Micron Packages
- Design Document
- Scripts to run the design
- PPT used for Presentation.
- Demo.