Available Online at www.ijecse.org **ISSN- 2277-1956**

Design of DDR3 SDRAM controller

Vineeta Singh, Ajinkya Raut, Apurva Lonkar, Kanchan Salvekar
Department of Electronics and CommunicationEngineering,
PES MCOE, PUNE
24vineeta@gmail.com, aji_kya@yahoo.co.in, apslonkar@gmail.com, kksalvekar@gmail.com

Abstract- DDR3 SDRAM Memory controller is the interface between DDR3 memory and the user. The design consists of front end and back end modules. The memory controller manages the flow of data going to and from the main memory. The front end provides interface to the user side and it consists of read data, write data and attributes FIFO. It is followed by request breaker. The arbiter decides the order in which the requests should be executed. The back end provides interface to the memory side and consists of calibration logic, data path, application address FIFO, application read and write FIFO.

Software used for designing the RTL Schematic, state diagrams and flow diagrams is DIA, can be done using Orcad as well. The code has been implemented in VHDL. GVIM editor has been used to write the code. The code has been tested on ModelSim simulator as per the industry norms.

Keywords – RTL, DIA, DDR3, SDRAM.

I. INTRODUCTION

'Design of DDR3 Memory Controller' involves mainly the design of the front-end and is made compatible with the Xilinx back-end. The main aim is to hide the complexities of the DDR protocol from the user following the bottom-up approach. With two transfers per cycle of quadrupled clock, a 64-bit wide DDR3 module may achieve a transfer rate of up to 64 times the memory clock speed in megabytes per second (MB/s). As 64 bits data can be transferred at a time per memory module, DDR3 SDRAM gives transfer rate of (memory clock rate) * 4(for bus clock multiplier)*2(for data rate)*64(number of bits transferred) /8(number of bits/byte).

The DDR3 memory operates using a differential clock provided by the controller. Commands are registered at every positive edge of the clock. A bidirectional data strobe (DQS) is transmitted by the DDR3 SDRAM device during Reads and by the controller during Writes.

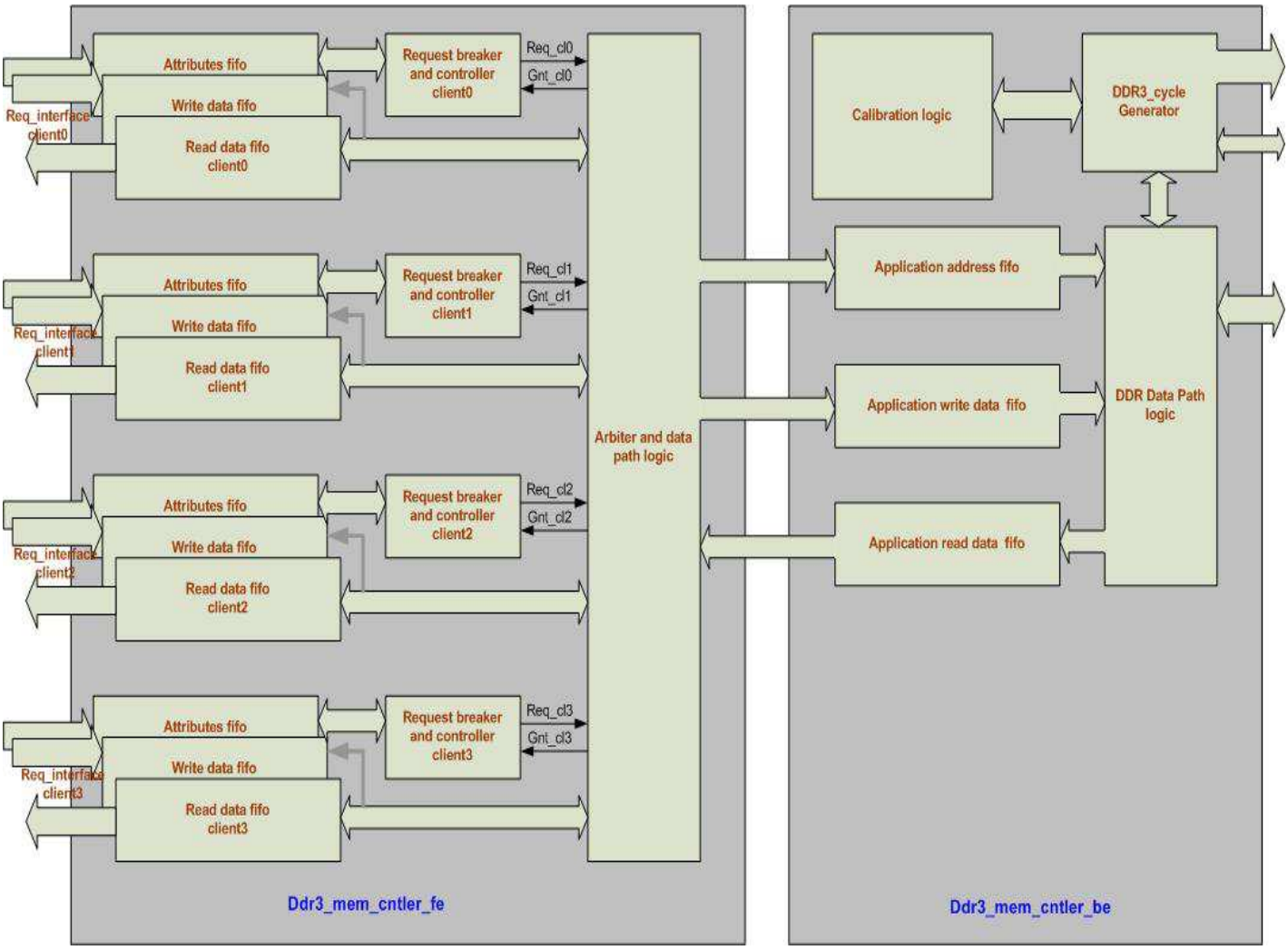
It uses a user backend interface to generate the Write address, Write data, and Read addresses. This information is stored in three backend FIFOs for address and data synchronization between the backend and other controller modules.

II. SYSTEM DESIGN

Block diagram and description

1) Attributes FIFO-

The Attributes FIFO is responsible for storing the data related to the request. When client sends a request, he sends the information regarding the requests which are called the attributes. They are burst length, memory address, request length, etc. Attributes FIFO is a part of per channel logic, hence there are four such attributes FIFOs, each belonging to the respective client. This FIFO then interacts with the request breaker and sends all the required data to the request breaker for analysis.



2) Write data FIFO-

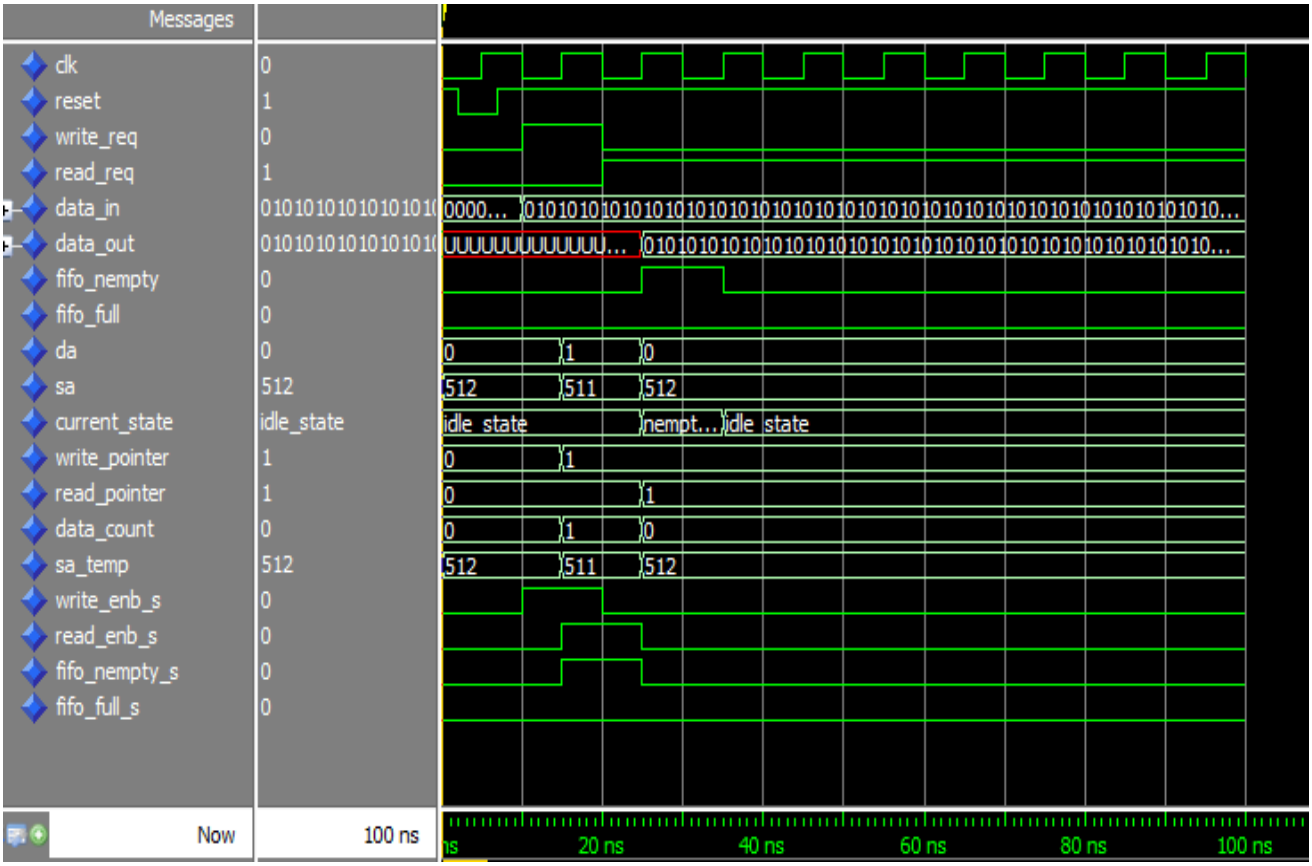
The write data FIFO receives all the data that has to be written on to the memory from the client. It stores the data and sends it by the first in first out mechanism. When it has the data from the client, it notifies the arbiter about the data available and that it needs to be sent to the back end. When the write data FIFO goes full, it cannot receive any further data. The write data is directly sent to the arbiter and then to the back end. As it comes under per channel logic, there are four such write data FIFOs.This is accessed by the user back end modules.

3) Read data FIFO-

The read data FIFO receives all the data that client needs to read from the memory. When the read data FIFO receives data from the memory, it notifies the client that it has the data and that he can read it. When read data FIFO goes full, it cannot receive and further data from the memory. The arbiter interacts with the read data FIFO when it needs to send the data. As this FIFO comes under per channel logic, there are four such FIFOs, each belonging to one client.

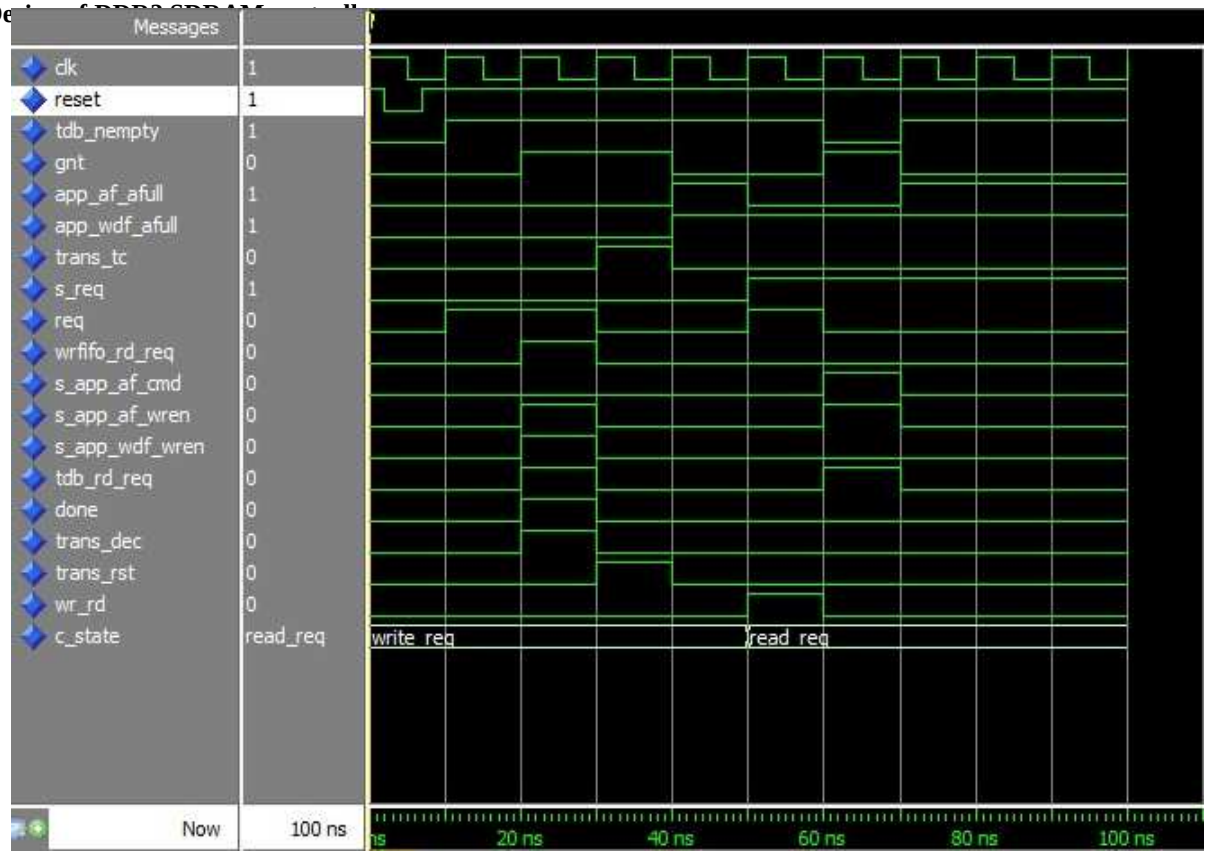
The read data FIFO receives all the data that client needs to read from the memory. When the read data FIFO receives data from the memory, it notifies the client that it has the data and that he can read it. When read data FIFO goes full, it cannot receive and further data from the memory. The arbiter interacts with the read data FIFO when it

needs to send the data. As this FIFO comes under per channel logic, there are four such FIFOs, each belonging to one client.



4) Request Breaker and Controller-

It is the heart of the front end of the memory controller. Request breaker as the name suggests looks after analyzing the requests that it receives from the client. It basically interacts with the attributes FIFO and the arbiter. It decodes the address located in the FIFO. It receives the all the information about a client request from the attributes FIFO. It then decides the request length from the data. Then the manner in which the data needs to be sent to the arbiter and further to the back end is decided by the request breaker. It puts up a request to the arbiter and when a grant is received, it sends the required information ahead. It manages issuing the commands in correct sequencing order while determining the timing requirements of the memory. The commands are pipelined to synchronize with the address signals before being issued to the DDR 3 memory. As the request breaker comes under per channel logic, there are four such request breakers, each belonging to one client.



5) Arbiter and Data path logic-

It determines the order in which requests are passed to the memory device. In our design the arbiter interacts with the back end module as well as all the other blocks of the front end i.e the request breaker and the FIFOs. When the arbiter receives a single request, that request is passed immediately; however when multiple requests are received, the arbiter uses arbiter rules to determine the order in which to pass requests to the memory device.

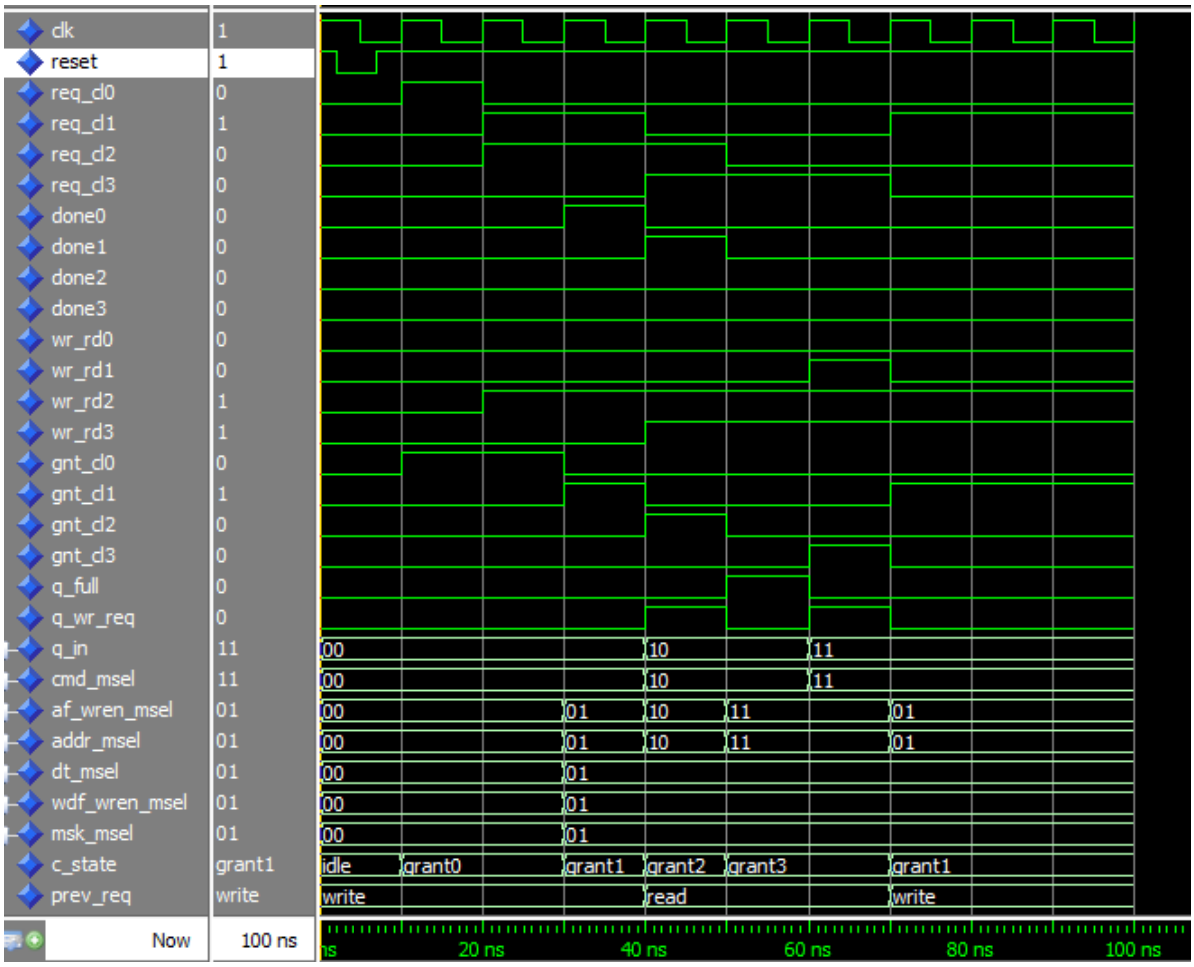
Following rules are followed-

- If only one master is issuing a request, grant that request immediately.
- If there are requests from 2 or more masters, tests are applied

a) Is there a read request? If so, arbiter grants read requests ahead of write requests.

b) If neither, then it grants the oldest request first.

c) It can assign priority to the requests like sequential, round robin or weighted. The arbiter designed by us uses the round robin arbitration scheme with decided priority.



6) Back-end-

Read and Write accesses to the DDR3 SDRAM device are burst oriented. Accesses begin with the registration of an Active command, followed by the Read or Write command. The address bits registered with the Active command are used to select the bank and row to be accessed. The address bits registered with the Read and Write command are used to select the bank and starting column location for the burst access. The back-end interface is used to generate the Write address, Write data, and Read addresses. This information is stored in three backend FIFOs for address and data synchronization between the backend and controller modules. Based on the availability of addresses in the address FIFO, controller issues the correct commands to the memory taking into account the timing requirements of the memory.

DDR Data Path logic-

a) Write data path – The memory specification requires strobe (DQS) and data (DQ) signals to be transmitted center aligned with DQ. The strobe forwarded to the memory is 180 degrees out of phase with CLK0.
b) Read data path - It compromises various register stages to capture the read data from the memory and transfer it to the internal FPGA clock domain. This is accomplished by using a combination of Chip Sync elements available in each I/O and flip-flops located in FPGA fabric.

First Stage- the DQ is captured by the input DDR (IDDR) of each DQ I/O. The differential DQS strobe is placed on a clock I/O pair, drives an IDELAY element and BUFIO local clock network, and clocks each DQ IDDR. The input

of each DQ IDDR is a delayed version using the built-in IDELAY element which is adjusted to provide sufficient timing between the delayed DQ and DQS inputs to the IDDR. The IDELAY setting is determined by a timing calibration routine executed one after system reset.

Second Stage- The outputs of the IDDR (for rising and falling data) are routed to flip-flops located in the FPGA fabric, close to DQ I/O. The fabric flops are clocked with the core (FPGA) clock. Synchronization is achieved by using DQ and DQS IDELAY elements to adjust the output of the IDDR relative to the core block. The output of the flip-flops is now synchronous with the clock used for the rest of the DDR3 interface logic.

Controller Implementation

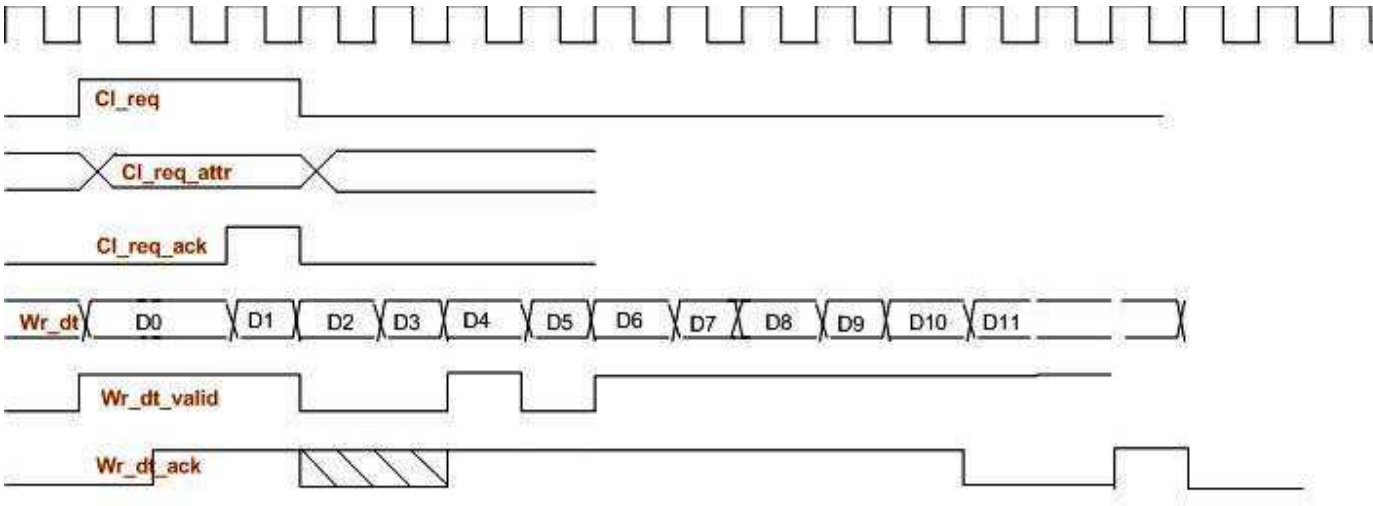
The controller has the ability to keep four banks open at a time. The banks are opened in the order of the commands that are presented to the controller. In the event that four banks are already opened and an access arrives to the fifth bank, the least recently activated bank will be closed and the new bank will be opened. All the banks are closed during auto refresh and will be opened as commands in the correct sequencing order while determining the timing requirements of the memory.

Before the controller issues the commands to the memory:

1. The controller decodes the address located in the FIFO.
2. The controller opens a row in a bank if that bank and row are not already opened. In that case of an access to a different row in an already opened bank, the controller closes the row in that bank and the new row. The controller moves to the Read/Write states after opening the banks if the banks are already opened.
3. After arriving in the Write state, if the controller gets a Read command, the controller waits for the write_to_read time before issuing the Read command. Similarly, in the Read state, when the controller sees a Write command from the command logic block, the controller waits for the read_to_write time before issuing the Write command. In the Read or Write state, the controller also asserts the read enable to the address FIFO to get the next address.
4. The commands are pipelined to synchronize with the Address signals before being issued to the DDR3 memory.

Timing Diagram and Explanation-

The first signal represents the clock signal of desired frequency and time period. On the rising edge of the second clock pulse a request is sent by the client port. It lasts for two clock pulses and then falls down. A differential clock pulse of the client's request with attributes lasts from the rising edge of client request to the falling edge of the same. This gives all the information of the attributes. An acknowledge signal is then sent by the controller so that the user understands the request has been accepted by the controller. Then the differential signal consisting of the write data is sent to the client port. The write data valid signal defines the data that has to be written and that it is a valid data. The write acknowledge signal makes it clear that the data has been received with bits D2, D3 being don't care conditions.



APPLICATIONS:
1. In National Stock Exchange (calculation done in FPGA)
2. for high-performance computing
3. Mobile application
4. Data Processing
5. Can be used in Network Controllers, high end memory systems using cache, Bus protocol controllers
6. Can be made compatible with different buses like AXI, AHB etc.

FUTURE MODIFICATIONS:
1. Number of client ports can be changed according to the need.
2. Request length can be varied according to the requirements.
3. Different arbitration schemes can be used to enhance the rate of data transfer

III. RESULTS AND CONCLUSION

- The controller hides the complexities of DDR protocol.
- Maximum request length of 8k is achieved that enhances signal data flow to and from memory.
- The multi port architecture enhances data transfer
- The order of transaction is preserved.
- Pipelining increases the speed of execution.

IV. REFERENCES

[1]. DDR3 SDRAM Specification (JESD79-3B) JEDEC Standard, JEDEC Solid State Technology Association, April 2008.

[2].Application Note: Adrian Cosoroaba, 'High-Performance DDR3 SDRAM Interface in Virtex-5 Devices', Xilinx, XAPP721 (v2.2) July 2009.

[3]. 'Implementing High-speed DDR3 Memory controllers in a Mid-Range FPGA,' a Lattice Semiconductor White paper, March 2010.

[4]. VHDL Compiler Reference Manuals: 1-8

[5].'Design of Low Power Double Data Rate 3 Memory Controller with AXI compliant' International journal of engineering and advanced technology (IJEAT) Vol.1, Issue 5, June 2012.