International Journal For Technological Research In Engineering
Volume 2, Issue 4, December-2014 *ISSN (Online): 2347 - 4718*

DESIGN OF SDRAM MEMORY CONTROLLER USING AN OUT OF ORDER SCHEDULER FOR IMPROVING THROUGHPUT

A. Lohitha[1], T. Sudheer Kumar[2]
D.V.R College of Engineering and Technology

ABSTRACT: SDRAMs are the most preferred form of memories for use as a System memory in SOCs due to their lower area compared to SRAMs. Memory controllers are used to perform the DRAM specific functions like Refresh, Activate and Precharge. A typical memory controller consists of a Command generator and a Scheduler. Designing an interface bridge between AXI and DDR3 SDRAM is the main objective of this paper. In this paper an out of order scheduler is proposed, which will give higher priority for the requests which belong to the same row as the one that is currently activated. This will lead to less number of Activates and Precharges as compared to the first come first serve policy. The lesser number of Activates and Precharges will lead to a higher system memory throughput and contributes significantly to the improvement of system performance as memory is the bottleneck for high performance in any SOC. On an average a minimum throughput of 30% is achieved with "out of order scheduler" based memory controller compared to first come first serve based memory controller.
Keywords: SDRAM, Memory Controller, AXI, Scheduler, Precharges

I. INTRODUCTION

In this era of fast processors, there is a requirement for faster and bigger memories. But today the speed of fetching data from memories is not able to match up with speed of processors. Memory devices are almost found in all systems and nowadays high speed and high performance memories are in great demand. For better throughput and speed, the controllers are to be designed with clock frequency in the range of megahertz. As the clock speed of the controller is increasing, the design challenges are also becoming complex. The big difference between DDR and SDRAM is that DDR reads that on both rising and falling edges of the clock signal, while SDRAM only carries information on the rising edge of a signal. Because of that it transfer the data twice as fast as SDRAM, it consumes less power[9]. This paper mainly deals with the implementation of an interface bridge between AXI and DDR3 SDRAM. The controller accepts the Read / Write commands from interfacing bus and converts it into memory access. While doing this it combines AXI burst transactions into single DDR access where ever possible to achieve the best possible performance from DDR3 memory. The AXI DDR3 Controller allows access of DDR3 memory through AXI Bus interface. The controller works as an intelligent bridge between the AXI host and DDR3 memory. It takes care of the DDR initialization and various timing requirements of the DDR3 memory. Further enhancing the AXI compliant DDR3 controller[7] to improve the overall throughput of the DDR3 memory by modifying the algorithm of DDR3 controller by "OUT OF ORDER SCHEDULING ALGORITHM" which gives higher priority to the requests that belongs to same row.

II. BLOCK DIAGRAM

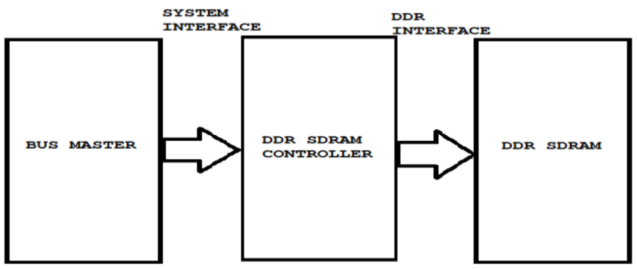

Fig. 1. DDR SDRAM controller system

III. FUNCTIONAL DESCRIPTION OF DDR3

Step	Function	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$
1	Load Mode	L	L	L
2	Auto Refresh	L	L	H
3	Precharge (1)	L	H	L
4	Bank Activate	L	H	H
5	Write	H	L	L
6	Read	H	L	H
7	No Operation/IDLE	H	H	H

Table.1. DDR3 COMMANDS

Read and write accesses to the DDR SDRAM are burst oriented; accesses start at a selected location and continue for a programmed number of locations in a programmed sequence. Accesses begin with the registration of an ACTIVE command, which is then followed by a READ or WRITE command[4].

A. ACTIVE

The ACTIVE command is used to open (or activate) a row in a particular bank for a subsequent access. This row remains active (or open) for accesses until a Precharge (or READ or WRITE with AUTOPRECHARGE) is issued to that bank. A PRECHARGE (or READ or WRITE with AUTOPRECHARGE) command must be issued before opening a different row in the same bank.

B. READ

The READ command is used to initiate a burst read access to an active row. If auto precharge is selected, the row being accessed will be precharged at the end of the read burst; if auto precharge is not selected, the row will remain open for subsequent accesses.

C. WRITE

The WRITE command is used to initiate a burst write access to an active row. If auto precharge is selected, the row being accessed will be precharged at the end of the write burst; if auto precharge is not selected, the row will remain open for subsequent accesses.

D. PRECHARGE

The PRECHARGE command is used to deactivate the open row in a particular bank or the open row in all banks. The bank(s) will be available for a subsequent row access a specified time (tRP) after the precharge command is issued. Once a bank has been precharged, it is in the idle state and must be activated prior to any READ or WRITE commands being issued to that bank. A PRECHARGE command will be treated as a NOP if there is no open row in that bank, or if the previously open row is already in the process of precharging.

IV. THE AXI PROTOCOL

The AXI protocol is burst-based and defines the following independent transaction channels:

- read address channel
- read data channel
- write address channel
- write data channel
- Write response channel

An address channel carries control information that describes the nature of the data to be transferred. The data is transferred between master and slave using either:
A write data channel transfer data from the master to the slave. In a write transaction, the slave uses the write response channel to signal the completion of the transfer to the master.
A read data channel transfer data from the slave to the master.

V. AXI COMPLIANT DDR3 CONTROLLLER

The AXI COMPLIANT DDR3 Memory controller design consists of following blocks.

- AXI interface
- DDR interface

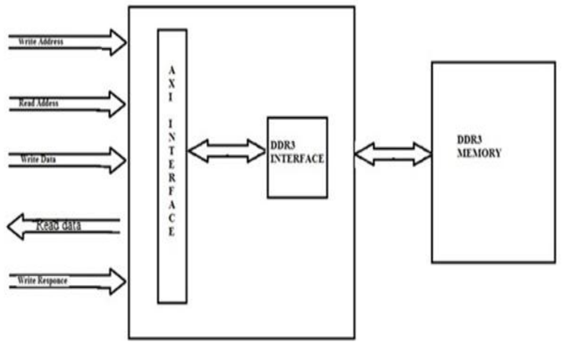

Figure .2. AXI compliant DDR3 Controller.

DDR3 memory interacts with the DDR3 interface and AXI channels are interact with the AXI interface.

A. ARCHITECTURE OF MEMORY CONTROLLER

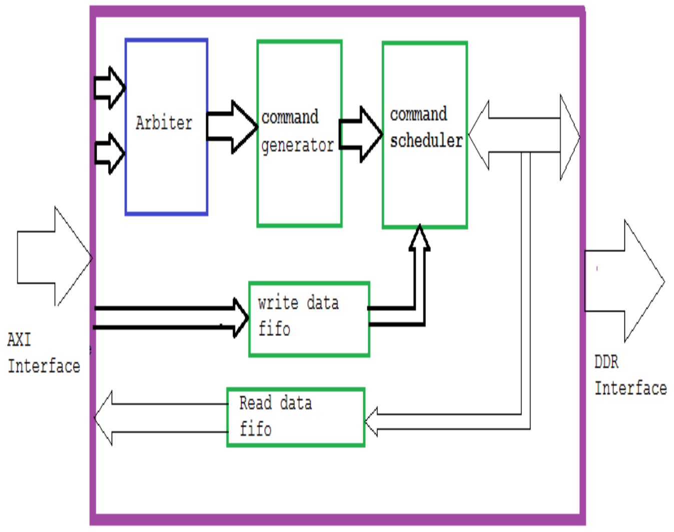

Figure 3. Architecture of AXI compliant DDR3 controller

B. SIGNAL FLOW OF AXI COMPLIANT DDR3 CONTROLLER

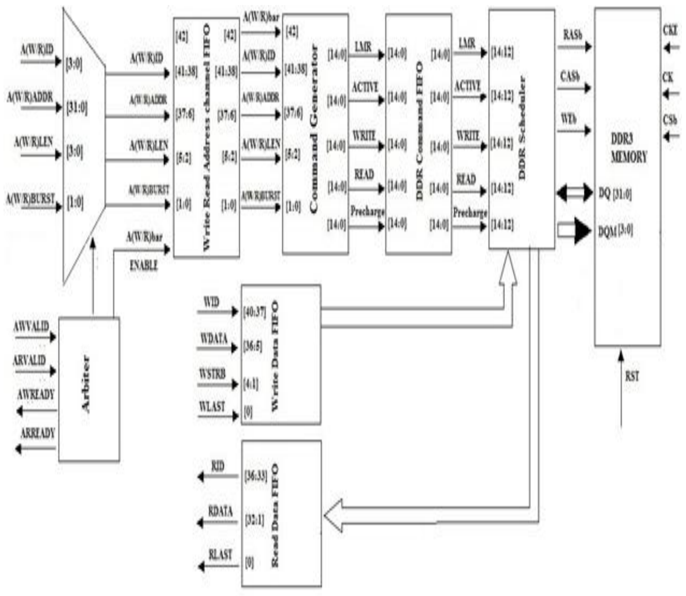

Figure.4. SIGNAL FLOW of AXI compliant DDR3 controller

C. ARBITER

The arbitration mechanism is used to ensure that only one Request (read/write) enters the FIFO. The arbiter performs this function by observing a number of different requests to use the bus and decide which is currently the highest priority requesting the bus. The arbiter block selects the one request either read request or write request when two requests are valid. The selection of request based on the previously completed write or read request.

Inputs channels of the arbiter

- Write Address channel
- Read Address channel

Internal logic for write operation

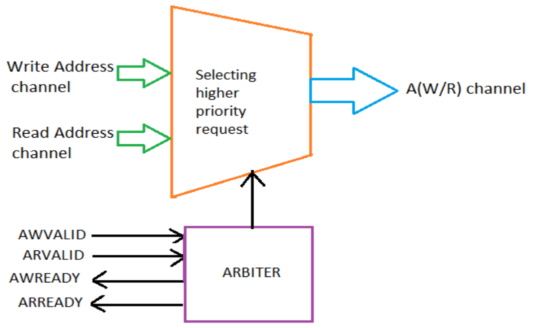

Figure .5 Arbiter

D. WRITE/READ ADDRESS FIFO

First in first out (FIFO)

FIFOs are often used to safely pass data from one clock domain to another asynchronous clock domain. Using a FIFO, pass data from one clock domain to another clock domain requires multi-asynchronous clock design techniques. We can write the data into the FIFO up to the FIFO is FULL and we can read the data from the FIFO until the FIFO become EMPTY. FIFO status cannot be corrupted by invalid requests. Requesting a read operation while the EMPTY flag is active will not cause any change in the current state of the FIFO. Similarly, requesting a write operation while the FULL flag is active will not cause any change in the current state of the FIFO.

E. COMMAND GENERATOR

The main purpose of the command generator is to generate the command for the memory. Depending on address lines command generator performs the operation. The address lines which command generator is received is predefined The inputs of the command generator are taken from the output of Write/Read address channel FIFO rd_ptr. Based on the given address the command generator generate the required commands (active, precharge, LMR, write, read) for the DDR3 memory.

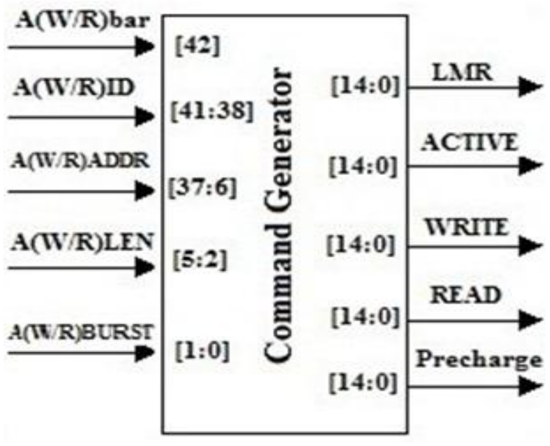

Figure 6. Command generator

Inputs of the command generator

A(W/R)ID[3:0], A(W/R)ADDR[31:0], A(W/R)LEN[3:0], A(W/R)BURST[1:0], A(W/R)VALID, A(W/R)READY

Outputs of the command generator

LMR, ACTIVE, WRITE, READ, PRECHARGE commands

Based on 3 conditions, Command generator is generate the commands

(i) No row is open

(ii) Currently open row = requested open row

(iii) Currently open row ≠ requested open row

Before generating the commands, the command generator checks the write/ read address length with same ID.

(i) No row is open

When No row is open condition the LMR, ACTIVE, WRITE, READ commands are generated

(ii) Currently open row = requested open row

When No row is open condition the LMR, WRITE, READ commands are generated

(iii) Currently open row ≠ requested open row

When No row is open condition the, PRECHARGE, LMR, ACTIVE commands are generated

Command generation table

Function	Addresses
Load Mode Register(LMR)	000, burst
Precharge(PRE)	001, row address
Active	010, row address
Write	011, column address, AWID
Read	100, column address, ARID, burst length.
No operation(NOP)	101

Table2. Command generation

F. DDR COMMAND FIFO

The command generator is generate all required commands at the same time. These commands are stored in the command FIFO

G. DDR SCHEDULER

The scheduler will take the commands from the command FIFO and generate the required signals for the DDR memory The command scheduler will not process all the commands at the same time. Scheduler works on command address, burst length, different ID's. Scheduler takes the input from the command generator to release the appropriate command for the next operation. If the state in reset mode, then data, DQM, address, FIFO read and writes pointers are assigned as '0'. Otherwise the command scheduler issues command for LMR. The LMR gives different modes of operation. It also defines the length of the burst. We have two different lengths of burst's 4 and 8. The command address for the LMR is '000'. After the LMR operation the Scheduler has two choices for scheduling commands, one is go for precharge other one is active mode. Precharge command issued when there is a change in row address. The active command is issued when LMR in same mode and it need go for read or writer operation. Precharge command changes the row address and it went pass to the new row. Activate command responsible two transfer to the two ways read operation or write. Which it needs to prefer is depend on the command issued by the scheduler. If scheduler issues write command then write operation is perform in FIFO. Write operation is in progress when FIFO has the free space in it. Write operation start at the FIFO of write pointer. If scheduler issues read command then read operation is perform in FIFO. Read operation start at the FIFO of read pointer.

Inputs of the DDR Scheduler
LMR, ACTIVE, WRITE, READ, PRECHARGE commands
OUTPUTS of the DDR Scheduler
RASb, CASb, WEb, DQ, DQM

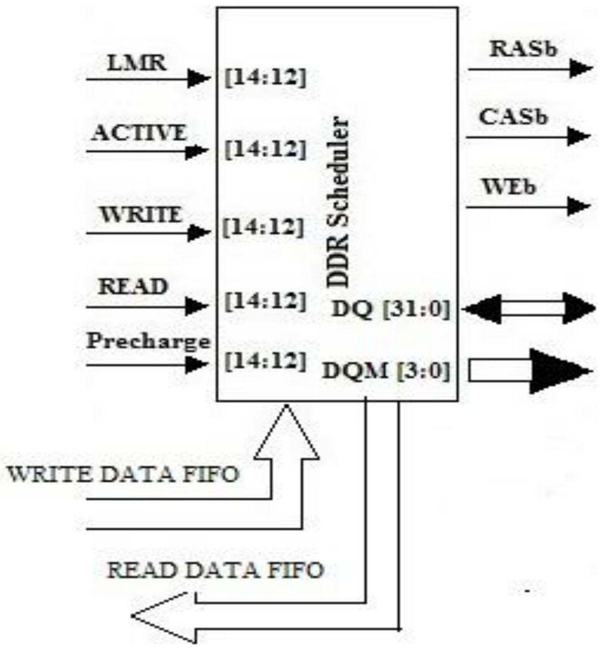

Figure 7 DDR Scheduler

VI. SIMPLIFIED STATE DIAGRAM OF THE DDR SCHEDULER

State Diagram provides a quick reference of available commands. Two additional Truth Tables provide current state/next state information. DESELECT and NOP are functionally interchangeable. The commands to the scheduler are generated by the command generator. Command generator checking for the burst length, write and read ID's. Command generator works on different addresses like row address, column address, write and read ID address, burst lengths. It gives commands to do a particular operation in mean time. Command scheduler issues the commands for LMR, precharge (PRE), active (ACT), read (RD), write (WR). Command scheduler work with different modes and states. It has the ability to control the all states in the diagram. Applies only to READ bursts with auto precharge disabled; this command is undefined (and should not be used) for READ bursts with auto precharge enabled and for WRITE bursts. This command is AUTO REFRESH if CKE is HIGH; SELF REFRESH if CKE is LOW. CKE is HIGH for all commands shown except SELF REFRESH; all states and sequences not shown are illegal or reserved.

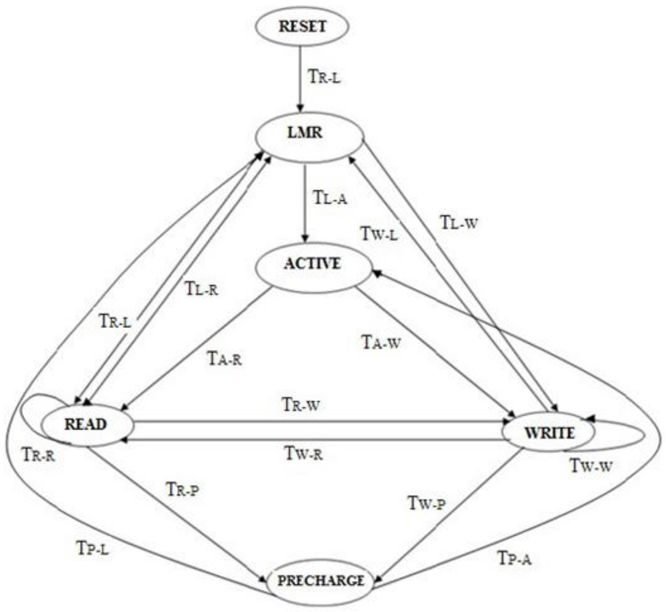

Figure 8 Simplified State Diagram of the DDR Scheduler

VII. OUT OF ORDER SCHEDULER

For improving throughput of the memory controller, an out of order scheduler is used in place of the arbiter. Rest of the command generator and command scheduler remains the same.

Architecture of Out of Order based Memory Controller

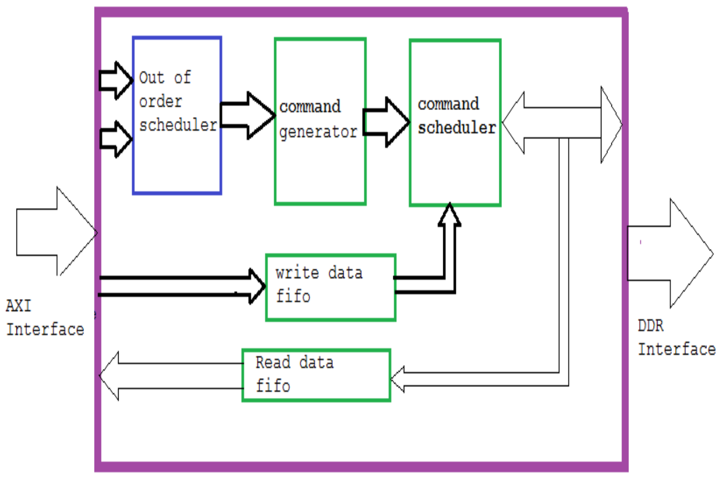

Figure 9. Architecture of Out of Order based Memory Controller.

A. Out of Order scheduler

The arbitration mechanism is the primary function of this scheduler which is used to ensure that only one Request (read/write) enters the FIFO. Similar to arbiter, out of order scheduler also performs this function by observing a number of different requests to use the bus and decide which is currently the highest priority requesting the bus. This scheduler block selects the one request either read request or write request when two requests are valid. The selection of request based on the previously completed write or read request. Apart from selecting one request at a time, out of order scheduler also priorities the request based upon the currently activated row. Requests are stored in a FIFO, which are then processed based upon the row address. The request which belong to same row are first processed than the request whose row address is different than the currently opened row address. With this we can reduce lesser number of Precharge and Activate states, there by improve the overall throughput. This can be achieved by using a register which can store a row address of the row which is currently opened and then comparing the next requests in sequence against the current open row address and then processing that request first.

B. Improvement in throughput

Consider a series of requests which are to processed, through a memory controller:

Series of Memory Operations
WRITE DATA => R1 , C1
WRITE DATA => R2 , C2
WRITE DATA => R1 , C3
WRITE DATA => R3 , C4
WRITE DATA => R2 , C3

Notation: WRITE DATA=> R1,C1→ writing data to column C1 of Row with row address R1.

When these requests are processed by a FIFO based Memory Controller then the operations performed by the memory are in the following order:

Operations performed by Memory
LMR , ACTIVATE R1, WRITE C1
PRECHARGE R1, LMR , ACTIVATE R2, WRITE C2
PRECHARGE R2,LMR, ACTIVATE R1, WRITE C3
PRECHARGE R1, LMR, ACTIVATE R3, WRITE C4
PRECHARE R3,LMR, ACTIVATE R2, WRITE C3

TOTAL OPERATIONS PERFORMED BY MEMORY: 19

Using an Out of Order scheduler in place of Arbiter and if we reorder the memory request to

Series of Memory Operations
WRITE DATA => R1 , C1
WRITE DATA => R1 , C3
WRITE DATA => R2 , C2
WRITE DATA => R2 , C3
WRITE DATA => R3 , C4

Operations performed by Memory
LMR , ACTIVATE R1, WRITE C1
LMR , WRITE C2
PRECHARGE R1,LMR, ACTIVATE R2, WRITE C2
LMR, WRITE C3
PRECHARE R2, LMR, ACTIVATE R3, WRITE C4

TOTAL OPERATIONS PERFORMED BY MEMORY: 15

Thus using an out of order scheduler based memory controller for the four write operations specified above there is a reduction of two Precharge and two activate operations by memory when compared to FIFO based memory controller. Thereby throughput of Memory using an Out of order scheduler is improved.

VIII. SIMULATION RESULTS

We are performing following 4 operations
1. Writing data 32’d0 to 32’d10 to the memory at 32’d0 address
2. Writing data 32’d11 to 32’d21 to the memory at 32’d200 address
3. Reading data from memory at 32’d0 address
4. Reading data from memory at 32’d200 address

Simulation results with FIFO based Memory Controller:

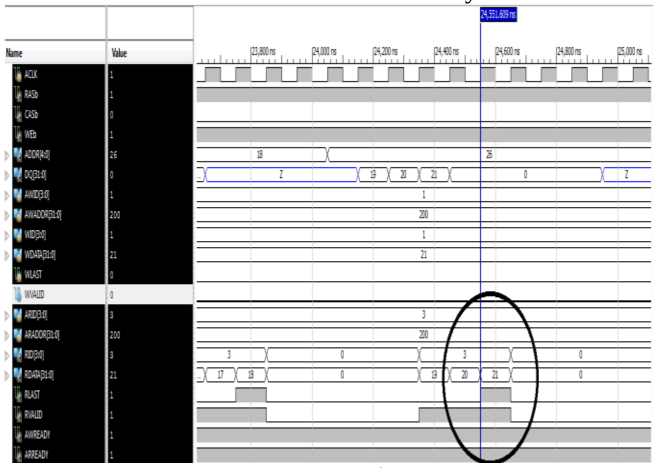

Figure 10. Simulation results of FIFO based Memory controller

The above specified four operations in time period of 24600ns .

Simulation results with out of order based Memory Controller:

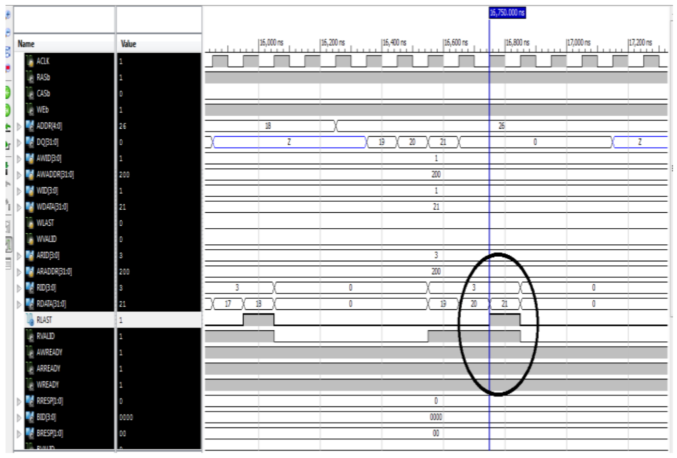

Figure 11.Simulation results of Out of order based memory controller

Time at which above specified operations completed is 16800ns. Thus using out of order scheduler there is improvement in throughput in time period of 7800ns. However a minimum or maximum improvement in throughput cannot be guaranteed as it entirely depends on the requests for the memory. In the above specified case, we are able to produce an average of 30% gain in throughput.

XI. CONCLUSION

User transactions are transferred repeatedly, without any delay in between its maximum operating frequency. We performed the AXI interface in scenario with one master and one slave. This design supports AXI protocol (32 or 64 bit) data width, remapping, run time configurable timing parameters & memory setting, delayed writes, multiple outstanding transactions and also supports the automatic generation refresh sequences. Out of Order scheduler based memory controller provides an improved throughput, however at the cost of additional circuitry. Depending upon timing requirement of the application, memory controller can be implemented with or without out of order scheduler. We examined the performance of the both the designs by generating different type of AXI commands and noting down the time taken by the DDR3 controller in finishing them. In most of the scenario the throughput of the design with “out of order scheduler” is better compared to the design without “out of order scheduler”.

REFERENCES

[1] Shaila S Math, Manjula R B, “Design of AMBA AXI4 protocol for System-on-Chip communication”, International Journal of Communication Network and Security (IJCNS), Vol-1, Issue-3.

[2] Darshana Dongre, Prof.Anil Kumar Sahu.” Implementation of AXI Design core with DDR3 memory controller for SoC”, International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 1 , Issue 3

[3] ARM, AMBA AXI protocol specifications, Available at, http://www.arm.com

[4] Sreehari,S.”AHB DDR SDRAM enhanced memory controller” Advanced Computing and Communication Systems (ICACCS), 2013 International Conference.

[5] Pan Guoteng, Luo Li , Ou Guodong , Dou Qiang, Xie Lunguo. ”Design and implementation of a DDR3-Based memory controller”,INTELLIGENT SYSTEM DESIGN AND ENGINEERING APPLICATIONS (ISDEA),2013.

[6] Shaila S Math, Manjula R B, Manvi.”Data transactions on system-on-chip bus using AXI4 protocol” Recent Advancements in Electrical, Electronics and control Engineering (ICONRAEeCE),2011.

[7] Lakhmani,V;Ali,N;Tripathi.V.S;”AXI Compliant DDR3 Controller”, Computer Modelling and simulation,2010.ICCMS’10. Second International Conference. Volume:4

[8] Samsung 1Gb DDR3 Specifications, Available at “http://www.samsung.com”

[9] Micron 1Gb DDR3 Specifiactions, Available at “http://www.micron.com”

[10] Anurag Srivastav, G.S.Tomar, Ashtosh Kumar singh.”Performance comparision of AMBA Bus Based System-on-chip communication protocol”, IEEE International Conference on communication systems and Network Technologies, 2011.

[11] Jun Zheng, Kang Sun,Xuezeng Pan, Lingdi Ping.”Design of a Dynamic Memory Access Scheduler” IEEE transl,Vol 7,pp 20-23,2007.