

# Implementation of Interface between AXI and DDR3 memory Controller for SoC

<sup>1</sup> Onteru Sreenath, <sup>2</sup> Syed kareemsaheb Associate Prof.

<sup>1</sup> PG Scholar, Dept of ECE, Audisankara Institute of Technology, Gudur, AP, India.

<sup>2</sup> Associate Professor, Dept of ECE, Audisankara Institute of Technology, Gudur, AP, India

## Abstract

This paper describes the implementation of AXI compliant DDR3 memory controller. It discusses the overall architecture of the DDR3 controller along with the detailed design and operation of its individual sub blocks, the pipelining implemented in the design to increase the design throughput. It also discusses the advantage of DDR3 memories over DDR2 memories and the AXI protocol operation. The AXI DDR3 Controller provides access to DDR3 memory. It accepts the Read / Write commands from AXI and converts it into DDR3 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 subsystem. 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. The controller implements multiple schemes to increase the effective memory throughput commands. It operates all the memory banks in parallel for attaining the maximum throughput from the memory and minimizes the effect of precharge /refresh and other DDR internal operations. Design is simulated in Modelsim 10.4a and synthesis on Xilinx ISE tool to report the area ,power, and delay.

**Keywords**—DDR-Double Data Rate, AXI –Advanced Extensible Interface, Command Scheduler , Command Generator

## I. INTRODUCTION

A bus protocol that supports separate address/control and data phases, unaligned data transfers using byte strobes, burst-based transactions with only start address issued, separate read and write data channels to enable low-cost DMA, ability to issue multiple outstanding addresses, out-of-order transaction completion, and easy addition of register stages to provide timing closure. The AXI protocol also includes optional extensions to cover signaling for low-power operation. AXI is targeted at high performance high clock frequency system designs and includes a number of features that make it very suitable for high speed sub-micron interconnects. The Memory is being improved to achieve high speed, low power consumption, cost-effective. DDR3 proves to achieve such goal. AXI compliant DDR3 Controller permits access of DDR3 memory through AXI Bus interface. The DDR3 controller works as an essential bridge between the AXI host processor and DDR3 memory. It takes care of the DDR3 initialization and various timing requirements of the DDR3 memory. Multiple schemes are performed to increase the effective memory throughput. These schemes include combining and reordering the Read/Write commands. For attaining the maximum throughput from the memory, it operates all the memory banks in parallel and minimizes the effect of precharge/refresh and other DDR3 internal operations. The DDR3 controller uses bank management modules to monitor the status of each SDRAM bank. Banks are only opened or closed when necessary, minimizing access delays.

## II. RELATED WORK

In several systems, dynamic RAM memories are important components such as in embedded systems. They have been used in embedded processors such as Black fin and some specific applications including fingerprint recognition system, HDTV SoC and so on. In order to enhance overall performance, SDRAMs offer

features including multiple internal banks, burst mode access, and pipelining of operation executions. Accessing one bank while precharging or refreshing other banks is enabled by the feature of multiple internal banks. By using burst mode access in a memory row, current SDRAM architectures can reduce the overhead due to access latency. The pipelining feature permits the controller to send commands to other banks while data is delivered to or from the currently active bank, so that idle time during access latency can be eliminated. This technique is also called interleaving. Researches on SDRAM controllers have been trying to deploy the features that SDRAMs can offer. The interleaving technique and pipelining feature have been respectively exploited in a memory controller of a commercial HDTV mixer application and in a SDRAM controller of a HDTV video decoder added arbitration mechanism to the SDRAM controller and used full page mode to finish the access requirements. However, due to the complexity and the cost of IC implementation, SDRAM features are hard to be applied to a system all together. The most popular groups researching on SDRAM controllers should be the numerous IP (Intellectual Property) suppliers such as XILINX, ALTERA, Lattice Semiconductor Corporation etc. Using IP cores can significantly shorten the development time. However, due to cost issues, it is not always the best way to buy an IP core from a supplier. To solve this problem describe above in SoC, there are usually multiple modules which need to access memories off chip. we design two way cable networked SoC, that is SDRAM controller connected by AMBA (Advanced Microcontroller Bus Architecture). The AMBA AHB is for high-performance, high clock frequency system modules. It has their own bandwidth requirements and responding speed requirements for SDRAM. By analyzing the multiple accesses from the 4 modules and the SDRAM specifications such as its accessing delay, we take both side 1 and side 2 into consideration respectively. On side 1, we use bank closing control. On side 2, the controller employs two data write buffers to reduce the data access awaiting time, and uses 2 read buffers to decrease the CAS delay time when reading data from SDRAM.

Due to the complexity of implementing the interleaving technique, we haven't introduced that technique to our design yet. However our design is proved to be functionally correct and high-

performance. According to the data sheet of general SDRAMs, a SDRAM must be initialized before starting to access it. In the last part of section we also give a universal and configurable timing analysis scheme considering that the timing process might have tiny differences between different SDRAMs from different corporations. Different SDRAMs from different corporations need different periods of maintaining time and different auto-refresh times. For example, a kind of SDRAM from MICRON needs 100us after power up, while a SDRAM from HYNIX requires 200us. After precharging all banks in the progress of initialization, auto-refresh command needs to be applied 2 times for a MICRON SDRAM MT48L8M16A2 and 8 times for HY57V561620C. FPGA (Field Programmable Gate Array) is using extensively and playing more and more important roles in the designing of digital circuit. Its programmable characteristics make circuit design much more flexible and shorten the time to market. Using FPGAs can also improve the system's integration, reliability and reduce power consumptions. FPGAs are always used to implement simple interface circuit or complex state machines to satisfy different system requirements. After implementation of the whole SoC for a Xilinx Virtex2P FPGA, the SDRAM software test programs are executed to verify the accuracy of the SDRAM controller. We run a program that fully writes the off-chip SDRAM, and then reads all the data out from it.

### III. AXI PROTOCOL SPECIFICATION

AXI is part of ARM AMBA, a family of micro controller buses first introduced in 1996. The first version of AXI was first included in AMBA 3.0, released in 2003. AMBA 4.0, released in 2010, includes the second version of AXI, AXI4.

There are three types of AXI4 interfaces:

- AXI4: for high-performance memory-mapped requirements.
- AXI4-Lite: for simple, low-throughput memory-mapped communication (for example, to and from control and status registers).
- AXI4-Stream: for high-speed streaming data.

The AMBA AXI protocol is targeted at high-performance, high-frequency system designs and includes a number of features that make it suitable for a high-speed submicron interconnects.

The objectives of the latest generation AMBA interface are to:

- Be suitable for high-bandwidth and low-latency designs.
- Enable high-frequency operation without using complex bridges.
- Meet the interface requirements of a wide range of components.
- Be suitable for memory controllers with high initial access latency.
- Provide flexibility in the implementation of interconnect architectures.
- Be backward-compatible with existing AHB and APB interfaces.
- The key features of the AXI protocol are:
  - Separate address/control and data phases.
  - Support for unaligned data transfers using byte strobes.
  - Burst-based transactions with only start address issued.
  - separate read and write data channels to enable low-cost Direct Memory Access (DMA)
  - Ability to issue multiple outstanding addresses
  - Out-of-order transaction completion

### IV. DDR MEMORY

Its current is current technology basically pumped version of SDRAM. 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. DDR memory controllers are used to drive DDR SDRAM, where data is transferred on the rising and falling access of the memory clock of the system. DDR memory controllers are significantly more complicated than Single Data Rate controllers, but allow for twice the data to be transferred without increasing the clock rate or increasing the bus width to the memory cell. Data transfer comparison between SDRAM and DDR SDRAM is

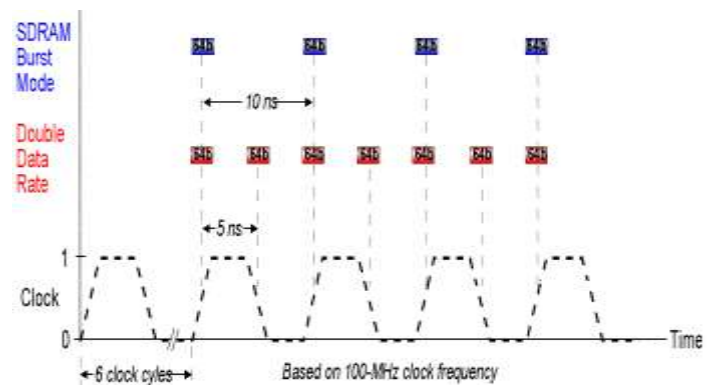


Fig. 1. Data transfer rate comparison between SDRAM (with burst mode access) and DDR SDRAM

TABLE I DDR FEATURE COMPARISON

Feature	DDR	DDR2	DDR3
Data Rate	200-400Mbps	400-800Mbps	800-1600Mbps
Burst Length	BL=2,4,8	BL=4,8	BL=4,8
No. of Bank	4banks	512Mb Banks : 4 1 Gb Banks : 8	512Mb/1Gb Banks:8
Pre fetch	2bits	4bits	8bits
CL/Trcd/Trp	15/15/15ns	15/15/15ns	15/15/15ns
Source synchronous	Bi-directional DQS (single ended default)	Bi-directional DQS (single/Diff. default)	Bi-directional DQS (Differential default)
Vdd/Vddq	2.5+/-0.2V	1.8+/-0.1V	1.5+/-0.075V
Reset	No	No	Yes
ODT	No	No	No

### V. IMPLEMENTATION OF AXI COMPLIANT DDR3 CONTROLLER

The AXI DDR3 Controller provides access to DDR3 memory. It accepts the Read / Write commands from AXI and Converts it into DDR3 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 subsystem. DDR3 memory interact with the DDR3 interface and AXI channels are interact with the AXI interface. The architecture of the design is shown in the figure. The design consists of following blocks.

- AXI interface
- DDR interface

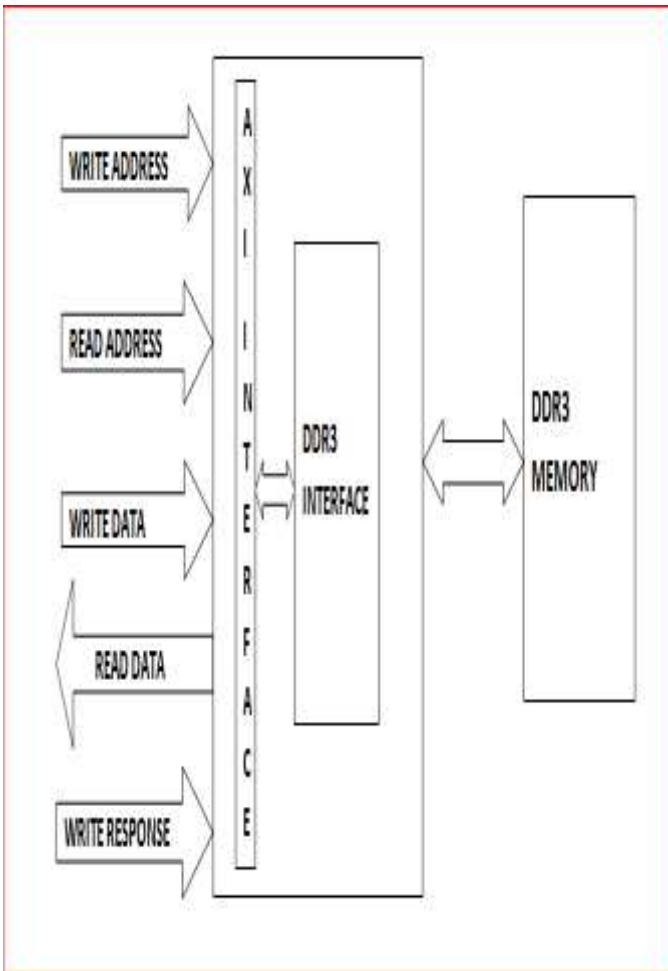


Fig. 2. Block diagram of AXI DDR Interface

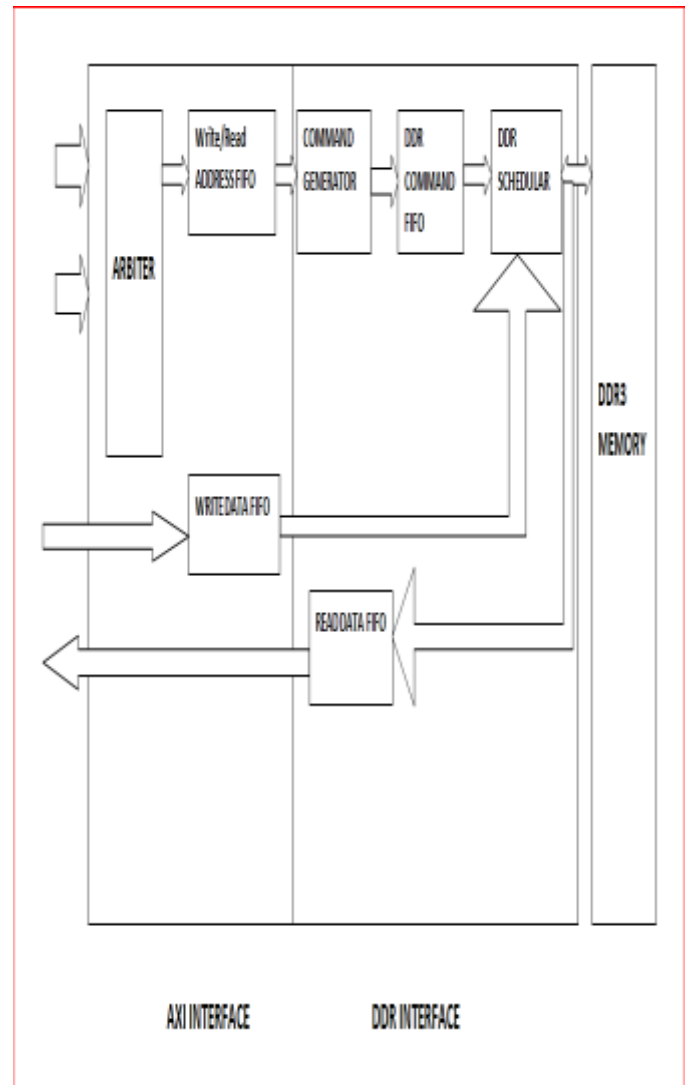


Fig. 3. Block Diagram of AXI compliant DDR3 controller has the following 8 blocks.

The Block Diagram of AXI compliant DDR3 controller has the following 8 blocks.

1. Arbiter
2. Write/Read Address FIFO
3. Command Generator
4. DDR Command FIFO
5. DDR Scheduler
6. Write Data FIFO
7. Read Data FIFO
8. DDR memory

#### A. Arbiter

The arbiter block is select 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. 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 is select 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.

#### B. Write/Read Address FIFO

Write/Read address channel FIFO Width = 43Bit

Write/Read address channel FIFO Depth = 8

Write/Read address channel take the inputs from the output of the arbiter block. The width is decided based on the input signals size taken from the arbiter output. After select the R/W request, the request stored in the write/ read address channel FIFO. These signals are stored in the FIFO memory location, from wrt\_ptr and based on the space available in the FIFO memory location Each request is stored in FIFO one location. 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.

The main features of Asynchronous FIFO are:

- Supports data widths up to 256 bits
- Supports memory depths of up to 65,535 locations
- Supports FULL and EMPTY status flags
- Optional ALMOST\_FULL and ALMOST\_EMPTY status flags
- Invalid read or write requests are rejected without affecting the FIFO state

### C. Command Generator

Main Purpose of the Command Generator Generates command for the memory. Depending on address lines command generator performs the operation. The inputs of the command generator taken from 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

### D. DDR Command FIFO

FIFOs are often used to safely pass data from one clock domain to another asynchronous clock domain. Using a FIFO to pass data from one clock domain to another clock domain requires multi-asynchronous clock design techniques. The Asynchronous FIFO is a First-In-First-Out memory queue with control logic that performs management of the read and write pointers, generation of status flags, and optional handshake signals for interfacing with the user logic. The command generator is generate all required commands at the same time. These commands are stored in the command FIFO

### E. 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 work 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 length 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 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.

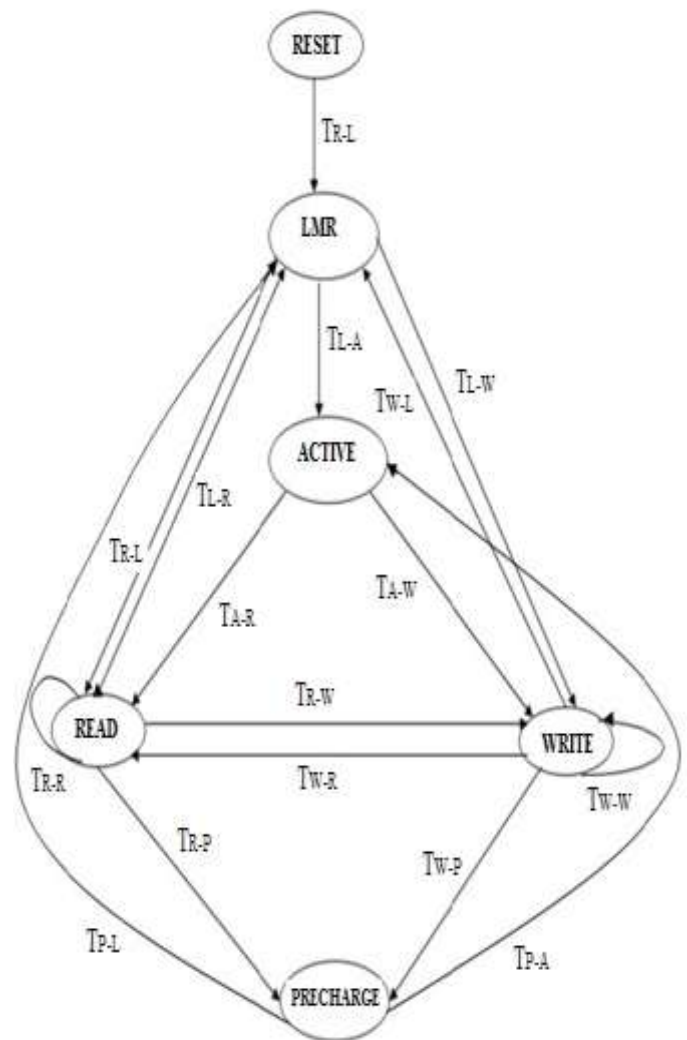


Fig. 4. FSM of DDR scheduler

### F. Write Data FIFO

FIFOs are often used to safely pass data from one clock domain to another asynchronous clock domain. Using a FIFO to pass data from one clock domain to another clock domain requires multi-asynchronous clock design techniques. The Asynchronous FIFO is a First-In-First-Out memory queue with control logic that performs management of the read and write pointers, generation of status flags, and optional handshake signals for interfacing with the user logic. The FIFO width is decided based on the input data size taken from the memory output. After decide the FIFO depth the write data is stored in the FIFO. These signals are stored in the FIFO memory location, from wrt\_ptr and based on the space available in the FIFO memory location

### G. Read Data FIFO

FIFOs are often used to safely pass data from one clock domain to another asynchronous clock domain. Using a FIFO to pass data from one clock domain to another clock domain requires multi-asynchronous clock design techniques. The Asynchronous FIFO is a First-In-First-Out memory queue with control logic that performs management of the read and write pointers, generation of status flags, and optional handshake signals for interfacing with the user logic. The FIFO width is decided based on the input data size taken from the memory output. After decide the FIFO depth the write data is stored in the FIFO. These signals are stored in the FIFO memory location,

from wrt\_ptr and based on the space available in the FIFO memory location

### H. DDR Memory

DDR3 SDRAM employs the 8-bit prefetch architecture for high-speed operation though DDR2 SDRAM employs 4-bit prefetch architecture. The bus width of the DRAM core has been made eight times wider than the I/O bus width, which enables the operating frequency of the DRAM core to be 1/8 of the data rate of the I/O interface section. READ operation: Converts 8-bit data read in parallel from the DRAM core to serial data, and outputs it from the I/O pin in synchronization with the clock (at double data rate). WRITE operation: Converts serial data that is input from the I/O pin in synchronization with the clock (at double data rate) to parallel data, and writes it to the DRAM core as 8-bit data

## VI. SIMULATION AND SYNTHESIS RESULTS

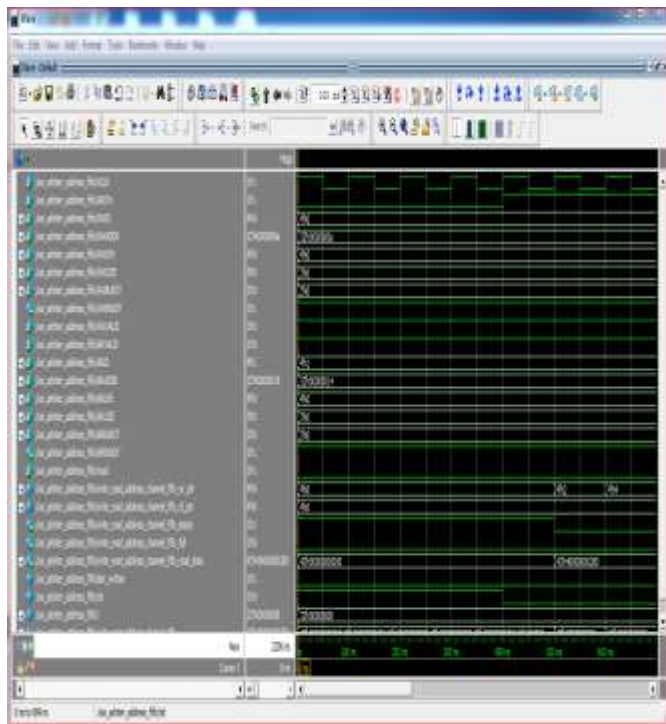


Fig. 5. Simulation result of Axi arbiter address FIFO

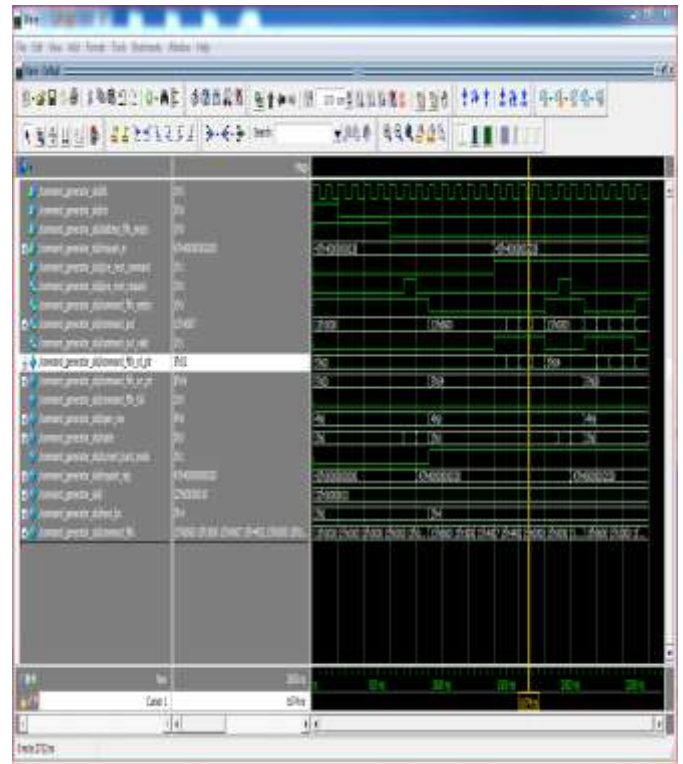


Fig. 6. Simulation Result of Command Generator

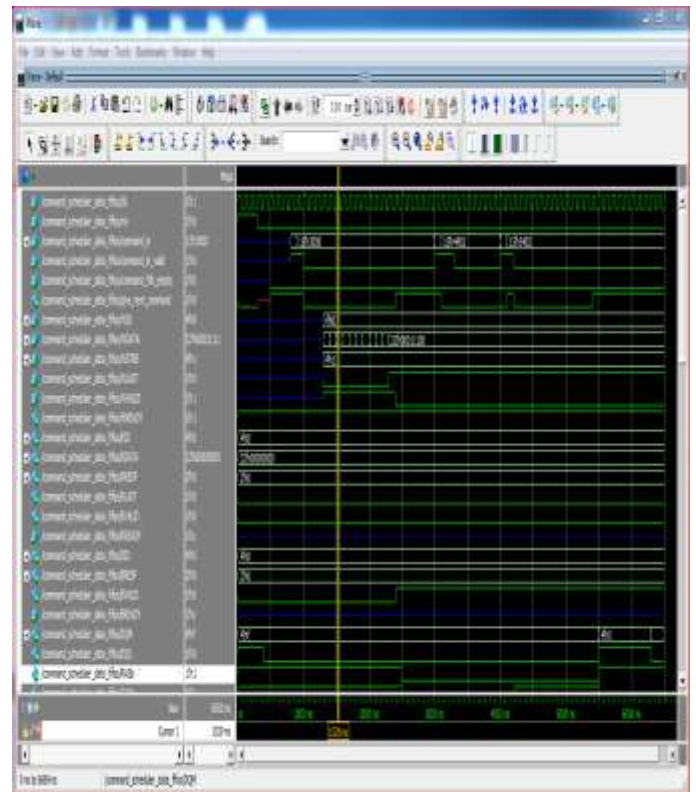


Fig. 7. Simulation Result of Command Scheduler

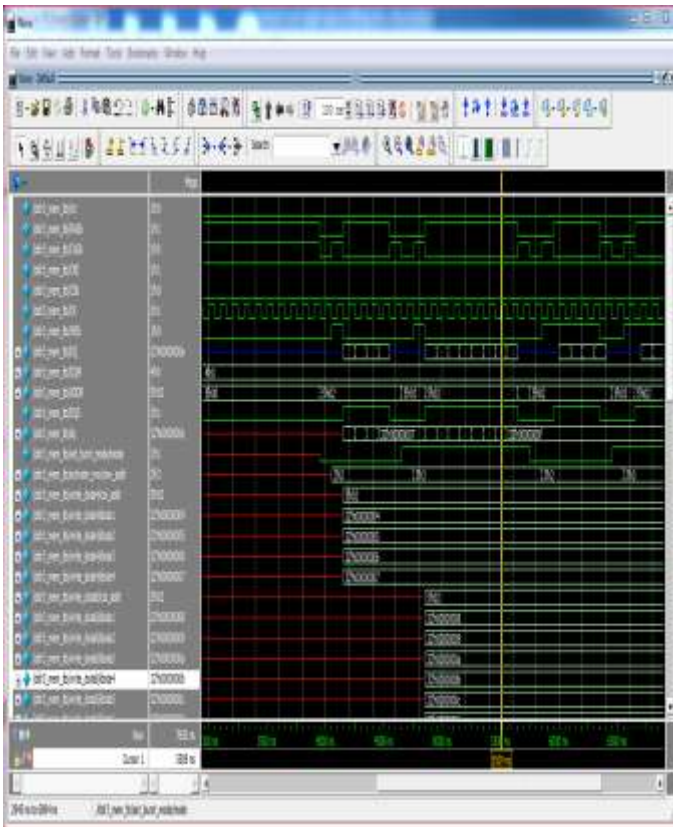


Fig. 8. Simulation Result of DDR3 Memo

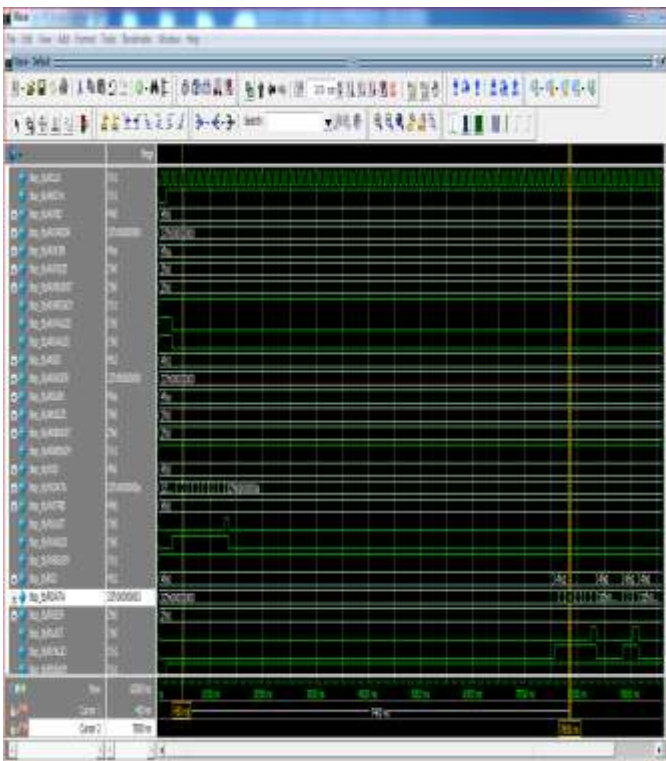


Fig. 9. Simulation Result of AXI-DDR3 Memory Top Level

TABLE II SYNTHESIS REPORT

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	11039	93296	11%
Number of Slice LUTs	28488	46648	61%
Number of fully used LUT-FF pairs	10887	28640	38%
Number of bonded IOBs	129	348	37%
Number of BUFG/BUFGCTRLs	1	16	6%
Total Delay	32.456ns	-	-

## VII. CONCLUSION

User transactions are transferred repeatedly, without any delay in between and its maximum operation frequency is 250 MHz. 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. We examined the performance of the design 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 is close to the theoretical max. The latency of the design is between 10 to 35 clocks based on the command generated and the internal DDR state.

## VIII. FUTURE SCOPE

Future improvements in AXI interface block is to add more features like fixed address mode, address wrapping mode and write response signal generation other than OKEY response. In fixed burst, the address remain all the same for every transfer in the burst. This burst type is for every repeated accesses to the same location such as when loading or emptying a peripheral FIFO and wrapping burst is similar to an incrementing burst, in that the address for each transfer in the burst is an increment of the previous transfer address. However in wrapping burst the address wraps around to a lower address when a wrap boundary is reached. The write response signal other than OKEY are EXOKAY SLVERR and DECERR. Future Improvement in DDR3 Controller is to add Reorder block in between AXI Access Manager and DDR3 Controller block. The Reorder block will enhance the performance of complete DDR3 Controller because it sends same row address command first then sends other row address commands. When we switch the transaction from Row address X to other Row address Y first we have to close Bank corresponding to that Row address X means precharge that Bank and it take tRP time to precharge the particular bank. So to achieve high performance we have to order the same row address command with coming data from AXI Interface block. But at this time we are not implementing this block because this is applicable when we are firing random row address command means that depend on open the customer requirements because this reorder block increase cost and size of the chip.

## References

- [1] Churoo (Chul-Woo) Park, HoeJu Chung, Yun-Sang Lee, Jun-Ho Shin, Jin-Hyung Cho, Seunghoon Lee, Ki-Whan Song, Kyu-Hyoun Kim, Jung-Bae Lee, Changhyun Kim, *Senior Member, IEEE*, and Soo-In Cho. "A 512-Mb DDR3 SDRAM Prototype and Self-Calibration Techniques"

Proc. IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 41,NO.4, APRIL 2006

- [2] K. Kim et al, " A 1.4 Gb/s DLL using 2nd order chargepump scheme with low phase/duty error for high-speed DRAM application ," in IEEE Int. Solid-state Circuits Conf. (ISSCC) Dig. Tech. Papers, 2004, pp. 212-523.
- [3] S.Lee et al, " A. 1.6 Gbs/pin double data rate SDRAM with wavepiplined CAS latency control," in IEEE Int. Solid-State Circuits conf. (ISSCC) Dig. Tech. Papers, 2004, pp.210-213.
- [4] H. Song et al, " A 1.2 Gbs/pin double data rate SDRAM with on die-termination," in IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers, 2003, pp. 314-496.
- [5] AMBA AXI specification (v1.0)
- [6] Samsung 1Gb DDR3 Specs.
- [7] Micron 1Gb DDR3 Specs



**1. Onteru Sreenath received his BTECH degree from Narayana Engineering College, SPSR Nellore, Affiliated to JNTU, Anantapur. He is currently Pursuing MTECH VLSI in Audisankara Institute of Technology, Gudur.**



**2. Syed kareemsaheb MTECH, Associate Professor in Audissankara Institute of Technology. Research Scholar, Department of Electronics & Communication Engineering, Nagarjuna University, Guntur,Andhra pradesh.**