Circuits design of memory accessing system based on AXI interface

BING LI
School of Integrated Circuits
Southeast University
Sipailou No.2, Nanjing, Jiangsu Province, China
CHINA
Southeast University Chengxian College
Dongda Road No.6, Nanjing, Jiangsu Province, China
CHINA
XIAOLING WANG
School of Integrated Circuits
Southeast University
Sipailou No.2, Nanjing, Jiangsu Province, China
CHINA
YONG LIU
School of Integrated Circuits
Southeast University
Sipailou No.2, Nanjing, Jiangsu Province, China
CHINA
w123xl@126.com

Abstract: - This paper proposes a method for designing the DDR3 SDRAM (Double Data Rate3 Synchronous Dynamic Random Access Memory) memory accessing system based on AXI (Advanced eXtensible Interface) interface, which can achieve data transmission between SoC (System on-Chip) and off-chip SDRAM through AXI interface complying with AXI protocol. The whole design has been accomplished by using Verilog hardware description language, and the functional simulation has been done in Modelsim10.0a software tool. Three different parameters, including bandwidth, delay and the size of buffer FIFO (First In First Out) have been analysed in the proposed system. Through FPGA (Field Programmable Gate Array) on-board verification, this DDR3 memory accessing system can operate at 200MHz well.

Key-Words: - AXI; DDR3 SDRAM; memory accessing system; Packets; memory performance; SoC

1 Introduction

Acting as a bridge between ARM processor and DDR3 SDRAM, the proposed memory accessing system allows on-chip system (SoC) to access off-chip DDR3 SDRAM memories based on AXI interface. As for SoCs who need a lot of off-chip storage space, AXI compliant memory accessing system is one of the most important design parts in system [1]. In this paper, with the study of ARM's latest on-chip advanced extensible interface we generate memory accessing commands through AXI bus on SoC by imitating ARM processor.

Two main components constitute the memory accessing system, including packets sending side and packets receiving side, both of which are implemented by Verilog HDL. Through the proposed system the processor can access off-chip DDR3 SDRAM. The parallel channel between packets sending side and packets receiving side is responsible for transmitting datum. Finally, we finish accessing DDR3 SDRAM by invoking MIG IP supplied by Xilinx to shorten the development cycle of the whole system [2]. The memory accessing system is functionally simulated by Mentor Company's software Modelsim10.0a and analysed on different bandwidth, delay and buffering size. Finally, we validate the DDR3 SDRAM memory accessing system on Xilinx FPGA board to test its function and implementation. The results of simulation and verification demonstrate that the design reaches the expected design objective, and it can operate steadily.

2 AXI Bus

The design scale is increasing faster and faster with the rapid development of SoC in recent years. The current mainstream on-chip AMBA (Advanced micro-controller bus architecture) bus protocol for SoC is released by ARM Company. It's not only just a kind of on-chip bus, but also an interconnection system with interface modules. AMBA bus interface has been widely used in system level chip design based on the ARM processor. AXI (Advanced eXtensible Interface) bus protocol is the most important part of AMBA3.0 protocol, which was first introduced in 1996. The first version of AXI was first included in AMBA3.0, released in 2003. AMBA4.0, released in 2010, includes the second version of AXI, AXI4 [3]. The AXI specifications describe an interface between a single AXI master and a single AXI slave, representing IP cores that exchange information with each other. Data can move in both directions between master and slave simultaneously, and data transfer sizes can vary. The key features of the AXI protocol are [4]: separate address/control and data phases; support for unaligned data transfers, using byte strobes; uses burst-based transactions with only the start address issued; separate read and write data channels, that can provide low-cost Direct Memory Access (DMA); support for issuing multiple outstanding addresses; support for out-of-order transaction completion; permits easy addition of register stages to provide timing closure. The AXI protocol includes the optional extensions that cover signalling for low-power operation. Each of the independent channels consists of a set of information signals and VALID and READY signals that provide a two-way handshake mechanism. Both the read data channel and the write channel also include a LAST signal to indicate the transfer if the final data item in a transaction.

3 Memory Accessing System RTL design

Fig.1 shows the proposed DDR3 memory accessing system architecture. According to the function of the system, the design is divided into two parts. There are packets sending side and packets receiving side, which will be implemented on FPGA chip respectively, and communicating through a point-to-point mode parallel channel. Packets sending side is implemented on Xilinx ZYNQ-7000 series FPGAs [5]. It receives memory accessing commands and datum from ARM processor through AXI interface, and then buffers the information before packing it in accordance with parallel channel timing and sending the packets out. The channel is in charge with packets transmitting. Packets receiving side is implemented on Xilinx Kintex7 series FPGAs [6]. It accepts packets from the channel and buffers them before unpacking. Then it translates the commands and datum into standard memory commands, which are only readable by DDR3 SDRAM. The packets receiving side also packs the read-back datum from DDR3 SDRAM and sends them back to ARM processor eventually. On the whole, the accessing process is finished.

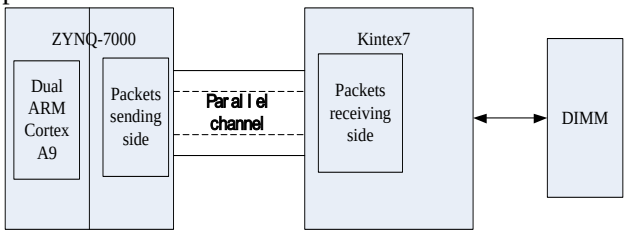


Fig.1 Memory accessing system architecture

3.1 Parallel signals and packets

In the paper, we define a parallel point-to-point mode channel through which the two parts implemented on two FPGA chips can communicate with each other. The channel interface protocol is described as the following aspects:

- Few control signals, commands completely enclosed in the packets.
- 64bit data width, high throughput.
- Full duplex transmission, peer-to-peer communications.

Memory accessing system has nine interconnection interface signals, which are shown in Table 1.

Table 1 Parallel interface signals

Name	***Source***	***description***
CLK	Global	Clock: all signals synchronize with it
RST_N	Global	Reset: when valid, any data transmission will be aborted
DATA[63:0]	src	Data: 64bit, driven by source, unidirectional transmission
SRC_RDY_N	src	Source ready to receive data
DST_RDY_N	dst	Destination ready to receive data
SOF_N	src	Frame head: the first 64bit data
EOF_N	src	Frame tail: the last 64bit data
SRC_DSC_N	src	Source interruption: cancel

Name	*Source*	*description*
		the transmission when detects any abnormal
DST_DSC_N	dst	Destination interruption: refuse data caused by lacking of buffer and etc.

Fig.2 shows the interconnection of memory accessing system architecture. The figure illustrates the topological connection between packets sending side and packets receiving side. We can see from Figure2 that both sides have source and destination. Each transaction from the interconnection interface is initiated by the source and receiving by the destination. In the case when there are no errors or interruptions, source and destination are ready. Then packets can be sent by the source to the destination at any time, and vice versa, the destination can send back read data packets to the source. The sender gives frame hand signal SOF_N and frame tail signal EOF_N to identify start and ending of every transaction.

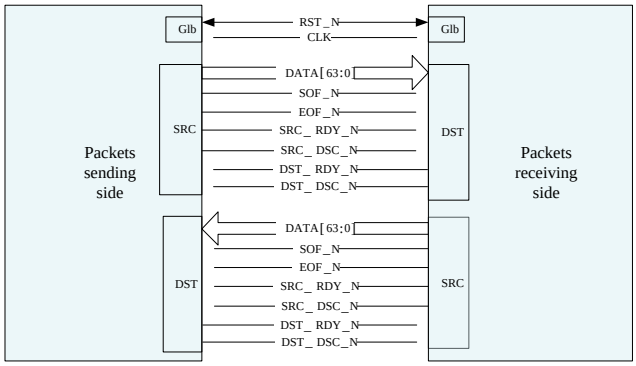


Fig.2 Interconnection topologic of parallel interface

The memory accessing system command format is the foundation of data packets transaction in system. We set a fixed read and write granularity, that's 32 Byte. Read and write request packets format is shown in Fig.3 and Fig.4 The fixed packet width is 64bit, whose frame head is command, including address, ID and read and write identification. The 32bit address is in the high address. The lowest one bit is the identification, "1" stands for read and "0" stands for write. The 4bit ID is placed in front of command identification. Remainder of the low 32bit address is the system reserved bits which is set to "0". There is a total 32 Byte data divided into four consecutive 64bit wide sequence following the command complied with interface protocol. The packets sending side packs consists of read/write commands and write data, while the packets sending side packs consists of read data and read control information.

Address(32bit)	reserved	ID(4bit)	1

Fig.3 Read command format

Address(32bit)	reserved	ID(4bit)	0
Data(64bit)			
Data(64bit)			
Data(64bit)			
Data(64bit)			

Fig.4 Write command format

3.2 Packets sending side RTL design

Fig.5 shows the architecture of the packets sending side which consists of the following modules, including AXI stimulus module, AXI write receiving module, AXI read receiving module, AXI read back module, asynchronous FIFOs, channel sending module and channel receiving module.

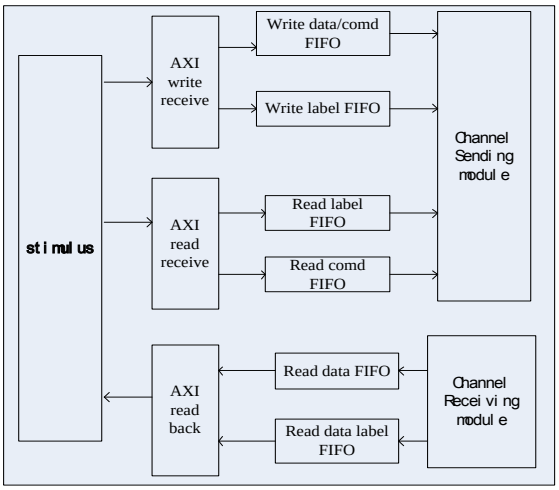


Fig.5 Architecture of packets sending side

3.2.1 AXI stimulus module

The function of AXI stimulus module is to generate memory accessing commands imitating the ARM processor, and send these commands to packets sending side. The stimulus module generates a large amount of read/write commands coping STREAM testing tool [8]. There are five test points in five modules. Copy: read large scales of data from a piece of memory area, and then write back the data to another piece of memory area. ADD: read large scales of data from a piece of memory area, add two of them, and then write the sum back to another piece of memory area. SCALE: read large scales of data from a piece of memory area, and then write back the data multiplied by a multiplier named scalar to another piece of memory area. TRIAD: read large scales of data from a piece of memory area, add two data (one of them is multiplied by a multiplier named scalar), and then write back the sum to another piece of memory area. GUPS:

generate a random address for read operation, and write back the read data from the random address after multiplying it by a random multiplier named scalar.

The five synthesizable stimulus modules can be instantiated separately in the packets sending side. They are in responsible for generating commands.

The stimulus module has five channel interfaces linked to AXI write receiving module, AXI read receiving module and AXI read back module. It starts the operation status once the DDR3 initialization is completed. The stimulus module has global clock signal, global reset signal and memory initialization signal. Other signals are described in the following paragraphs.

AXI write address channel: Output axi_wvalid indicates that the channel is signaling valid write address and control information. Input axi_wready indicates that the slave is ready to accept an address and associated control signals. Output axi_wid 4bit write address ID which is the identification tag for the write address group of signals. Output axi_wadder is a 32bit address indicating the first transfer in a write burst transaction. Output axi_wlen is the burst length, which equals 7 according to 32Byte. Output axi_wsize is the burst size which indicates the size of each transfer in the burst, it equals 2 according to 32bit width (2^2=4, 4Byte =32bit). Output axi_wburst is the burst type, which equals 2'b01 which indicates increasing type. The remainder signals which have not been used in this paper will be set to 0. AXI read address channel is similar to AXI write address channel, so we will not give unnecessary details.

AXI write data channel: Output axi_wd_valid indicates that valid write data and strobes are available. Input axi_wd_ready indicates that the slaves can accept the write data. Output axi_wd_wid is the 4bit write ID tag. Output axi_wd_data is the 32bit write data. Output axi_wd_strb is the 4bit write strobe which indicates which byte lanes hold valid data. Output axi_wd_last is write last signal which indicates the last transfer in a write burst. AXI read data channel is similar to AXI write data channel except for input axi_rd_resp which is the read response signal.

AXI write response channel: Input axi_wd_bid is the 4bit response ID tag. Input axi_wd_bresp indicates the status of the write transaction. Input axi_wd_bvalid indicates that the channel is signaling a valid write response. Output axi_wd_bready indicates that the master can accept a write response.

3.2.2 AXI write receiving module

The function of AXI write receiving module is to link to write address channel, write data channel and write response channel of AXI stimulus module. It accepts id and write identify, and then put them together to form a 32bit data, i.e. {{27 {1 'b0}}, AWID, 1' b0}. At the same time, it buffers the newly formed data, 32bit address and write data in the write data/command FIFO. The write command label FIFO will be updated when all above is done. The AXI write receiving module responses to stimulus module once the data and associated information in one burst transaction have been buffered in FIFO. BRESP signal which equals to 2'b00 implies a successful transaction. When response is done, the write label FIFO address pointer will add 1 to indicate that the write data FIFO now has data for read.

3.2.3 AXI read receiving module

The function of AXI read receiving module is to link to read address channel of AXI stimulus module. It accepts id and read identify, and then put them together to form a 32bit data, i.e. {{27 {1 'b0}}, ARID, 1' b1}. At the same time, it buffers the newly formed data and 32bit address in the read command FIFO. The read command label FIFO will be updated when all above is done. When all above is done, the read label FIFO address pointer will add 1 to indicate that the read command FIFO now has data for read.

3.2.4 AXI read back module

The function of AXI read back module is to link to read data channel of AXI stimulus module. It fetches data and address in the read data FIFO, and then sends them to stimulus module. The information includes read data, read address and response. It can get the information from the read data FIFO if the read data label FIFO is not empty which indicates that there are at least a 32Byte burst read data can be read. After reading 32Byte data from the read data FIFO, the address pointer of read data label FIFO will subtract 1 to indicate a 32Byte data has been red.

3.2.5 Channel sending module

The function of channel sending module is link to all kinds of FIFOs on the left side, to receive 32bit data from them. On the other hand, it links to parallel channel on the right side. In take the charge of sending 64bit data to the parallel channel. However, it firstly should get two 32bit data together to form a 64bit data according to parallel channel timing protocol. There are no priorities in

the process of sending operation. We apply a round-robin method to get the read and write commands. If a read transaction is undergoing then next will be a write transaction if the associated FIFO is not empty. This method is fair in the paper.

3.2.6 Channel receiving module

The function of channel receiving module is to accept data from the parallel channel. It gets the 64bit data packets, divides them into two 32bit data and put them into buffer FIFOs. The read data label FIFO address pointer will add 1 to indicate there are 32Byte read data in the read data FIFO.

3.2.7 Asynchronous FIFOs

In order to make it convenient for the designer to adjust clock frequency on both sides of the FIFO, FIFOs in packets sending side are all asynchronous. This will prevent any meta-stabilities [7]. All the FIFO is 32 in width and 4K in depth. FIFOs in packets receiving module has the same functions with 64bit in width and 2K in depth.

3.3 Packets receiving side RTL design

Fig.6 shows the architecture of the packets receiving module which has the following modules, including parallel channel interface module, MIG interface module, AXI_MIG and asynchronous FIFOs.

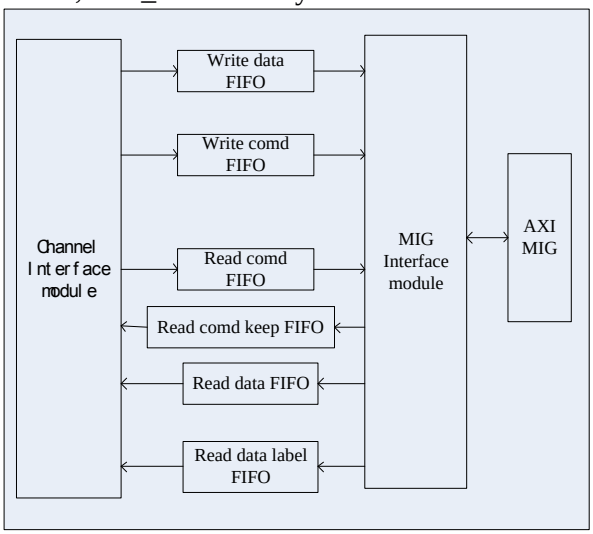


Fig.6 Architecture of packets receiving side

3.3.1 Channel interface module

The function of channel interface module is to accept 64bit width packets from parallel, and then unpacks them before buffering. It divides the packets into write commands and read commands according to the identification. Write data is sent to write data FIFO while commands are sent to command FIFO. All label FIFOs are been updated after above actions. What's more, channel interface module also accepts data red from DDR3 SDRAM and read address form read command keep FIFO, and then it packs them into 64bit width packets before sending them to the parallel channel.

3.3.2 MIG interface module

The function of MIG interface module is to link to AXI_MIG through five AXI channels. It communicates with AXI_MIG in accordance with required timing. It also applies the round-robin method to read write and read commands.

3.3.3 AXI_MIG

We generate the free IP core named AXI_MIG from Xilinx ISE software tool with a license. It works with DDR3 SDRAM directly. In this paper, we use ISE14.2 core generator to generate a DDR3 physical layer controller with AXI4 interface, which is responsible for changing the AXI commands into standard memory commands.

4 Test Results

In the paper, we choose Modelsim10.0a simulation software from Mentor Company during the functional simulation. This software supports PC, hybrid platform of UNIX and LINUX. It supports testbench in Verilog HDL language.

The process of simulation is described as follows: packets sending module and packets receiving module are instanced in the top module, which also instances DDR3 SDRAM. The top module generate global clock and reset signals from FPGA board. Five stimulus modules are been tested to cover most of the accessing cases.

As shown in Fig.7 in the end of this article, after the DDR3 initialization, stimulus module writes a piece of DDR3 area. The following accessing commands will only limited to this area address from 32'h0000_0000 to 32'h0000_4000.

As shown in Fig.8 in the end of this article, when the write address reaches 32'h0000_4000, the fill_done signal will be pull valid. The memory accessing system starts sending large scales of commands.

As shown in Fig.9 in the end of this article, once there are read data back to stimulus module, the data will be writing back to another part of DDR3 after COPY or other operations. From the simulation waveform we can see that the data read from DDR3 match what we write before.

Fig.10 in the end of this article shows the read and write process of DDR3 SDRAM. The smallest unit that can be process by DDR3 is 64Byte. We use 32Byte while another 32Byte is masked by strobe

signals. For write command: cs_n=0, RAS_n=1, CAS_n=0, WE_n=0. And for read command: cs_n=0, RAS_n=1, CAS_n=0, WE_n=1 [9].

In order to analyze the performance of the memory accessing system, we vary three parameters to recorder the simulation time. We use different buffer size, channel frequency and channel delay to test the design. The buffer size is varied by changing the depth of FIFOs in packets receiving module. The channel delay is varied by adding a delay-chain in the channel, in the delay-chain one register has a clock cycle delay. After analyzing the results we can balance the three parameters to get a memory accessing system with best performance.

Fig.11 shows the histogram of different parallel channel frequency. From the histogram we can find that with the decreasing of channel frequency, the smaller buffer size is the longer simulation time.

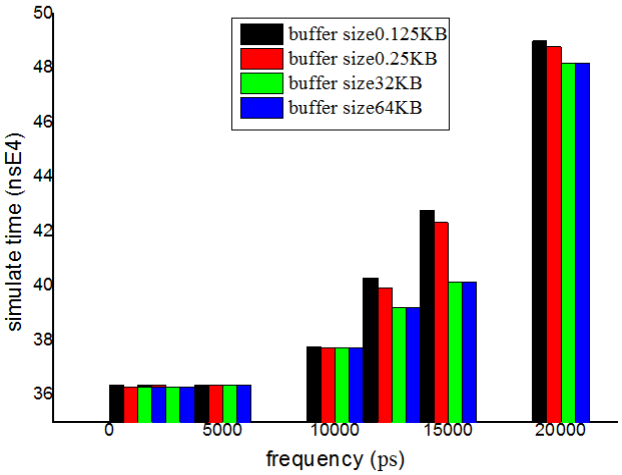


Fig.11 Frequency vs. Simulate time

Fig.12 shows the histogram of different buffer size. With fixed channel delay (set to zero), the greater the channel delay, the longer simulation time, the less influence on simulation time by buffer size.

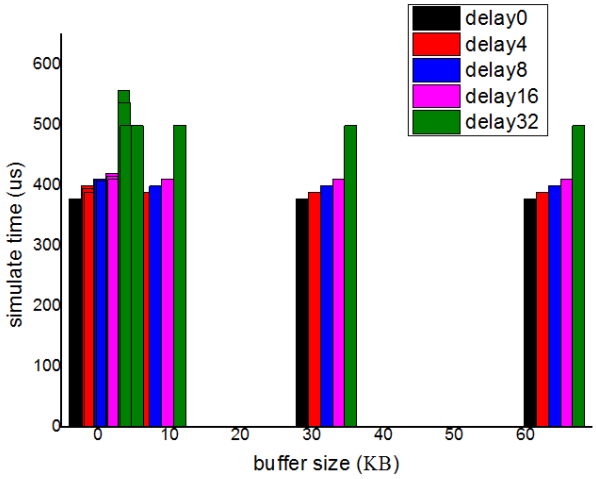


Fig.12 Buffer size vs. Simulate time

Fig.13 shows the histogram of different channel delay. From the histogram we can find that when the channel frequency is higher, the simulation time is shorter. However, AXI_MIG will not afford the throughput when the frequency is too high. So if we add delay in the channel, it will ease data for AXI_MIG.

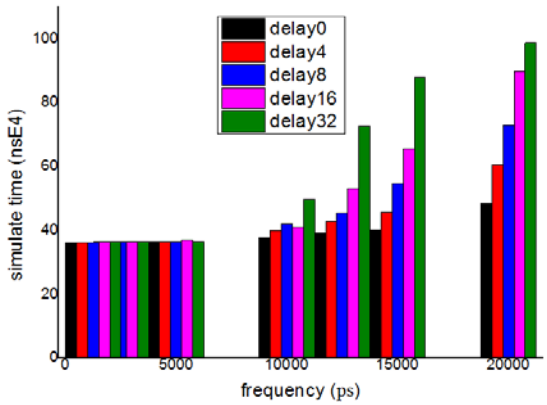


Fig.13 Delay vs. Simulate time

In the end, the function of the memory accessing system is implemented on FPGA board. The results demonstrate that the design can work at 200MHz with low rates of resource utilization listed in Table 2 and Table 3.

Table 2 Resource utilization sending side

Name	***Used number***	***Total number***	***ratio***
register	2841	437200	1%
Look-up-table	23394	218600	10%
RAM/FIFO	29	545	5%
IOBs	145	362	40%
BUFGs	3	32	9%
BSCANs	1	4	25%
PLLE2_ADVs	1	8	12%

Table 3 Resource utilization receiving side

Name	***Used number***	***Total number***	***ratio***
register	11080	508400	2%
Look-up-table	41928	254200	16%
RAM/FIFO	45	795	5%
IOBs	261	500	52%
BUFGs	5	32	15%
IDELAYE2	64	500	12%

Name	*Used number*	*Total number*	*ratio*
IDELAYCTRLs	2	10	20%
ISERDESE2s	64	500	12%
OSERDESE2s	114	500	22%
PHASER_IN_PHYs	8	40	20%
PHASER_OUT_PHYs	12	40	30%
BSCANs	1	4	25%
MMCME2_ADVs	1	10	10%
PHASER_REFs	3	10	30%
PHY_CONTROLs	3	10	30%
PLLE2_ADVs	2	10	20%

Fig.14 (a) and (b) in the end of this article show the logical circuits after synthesis in ISE14.6 respectively.

5 Conclusion

A novel architecture of DDR3 memory accessing system with AXI interface is proposed in this paper, which consists of two main components, there are packets sending side and packets receiving side. The whole design is implemented in Verilog HDL. Functional simulation and FPGA validation shown from Fig.7 to Fig.10 demonstrate that the system can work as expected. Three parameters are been varied to analyse the performance of the memory accessing system as shown from Fig.11 to Fig13.

The design can be improved in the future. For example, more AXI_MIG can be instanced in the design to extend the off-chip DDR3 capability. And commands from ARM processor is limited in fact, but in this paper the stimulus sends consecutive commands without an upper limit. What's more, parallel channel can be replaced by series channel to further improve the performance of the memory accessing system.

References:

[1] J H Ahn, J Leverich, R Schreiber, N P Jouppi. Multicore DIMM: An energy efficient memory module with independently controlled DRAMs[J]. IEEE Computer Architecture Letters, January 2009, 8(1): 5-8.

[2] 7 Series FPGAs Memory Interface Solutions User Guide[Z]. UG586 July 25, 2012.

[3] AXI Reference Guide[Z]. UG761 (v13.1) March 7, 2011.

[4] AMBA AXI and ACE Protocol Specification[Z]. ARM IHI0022D(ID102711), 2011.

[5] Zynq-7000 All Programmable SoC Overview[Z]. DS190 (v1.2)August 21, 2012.

[6] Przybus B. Xilinx Redefines Power, Performance, and Design Productivity with Three New 28 nm FPGA Families: Virtex-7, Kintex-7, and Artix-7 Devices[J]. Xilinx White Paper, 2010.

[7] JEDEC Standard. JESD79-3E. DDR3 SDRAM Specification[S]. Arlington USA: JEDEC Solid State Technology Association, July 2010.

[8] Kim L S, Dutton R W. Metastability of CMOS latch/flip-flop[J]. Solid-State Circuits, IEEE Journal, 1990, 25(4): 942-951.

[9] McCalpin, John D. STREAM benchmark[CP/OL]. 2007. http: //www.cs. virginia.edu/stream //FTP/Code/stream.c.

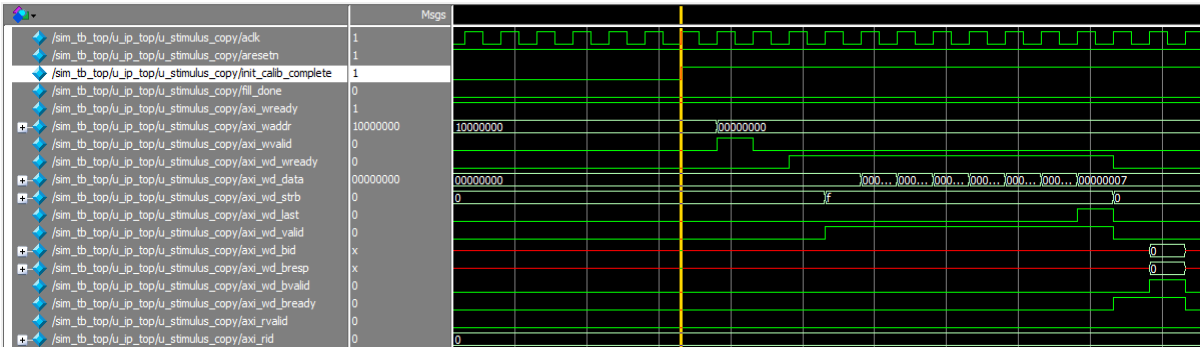


Fig.7 Write a piece of DDR3 area

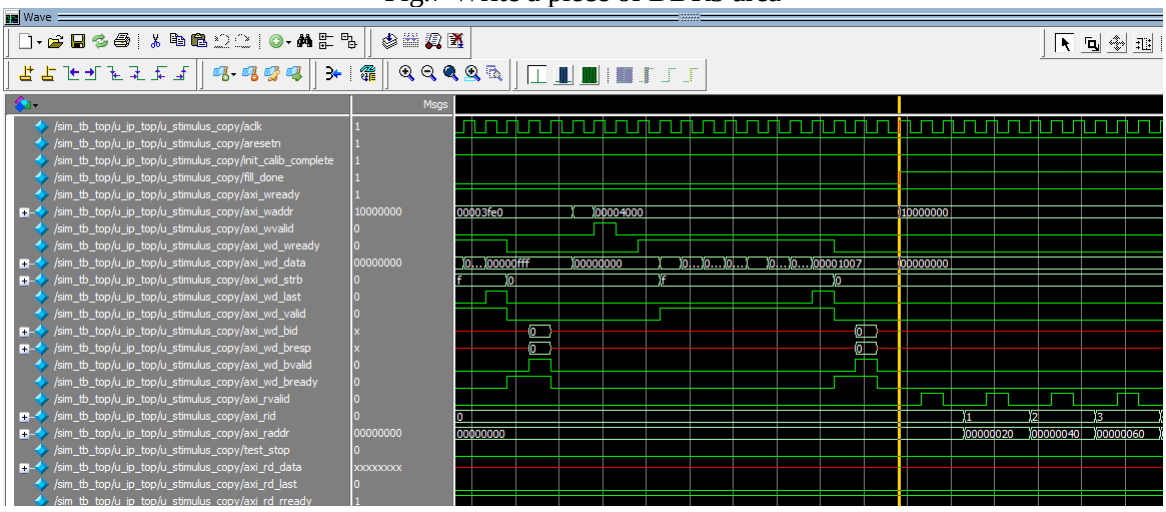


Fig.8 Read commands start

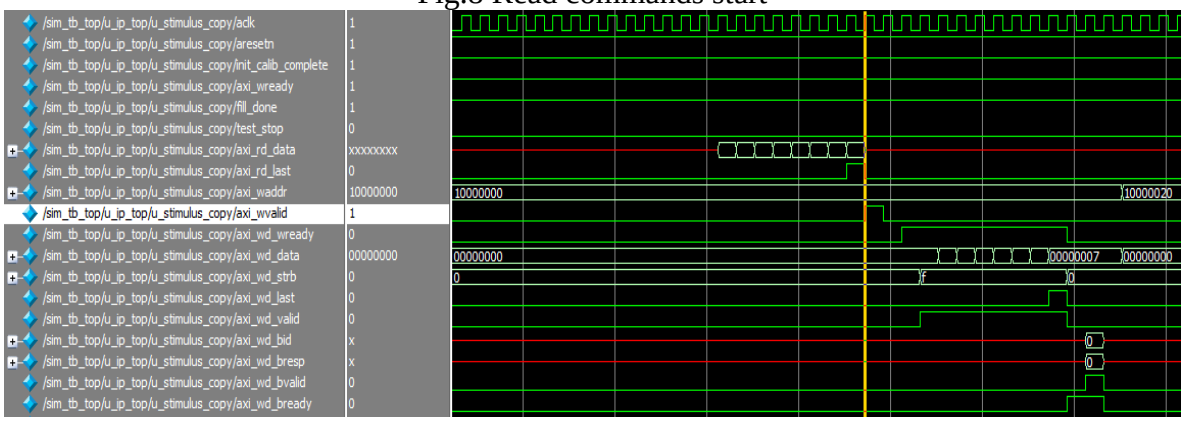
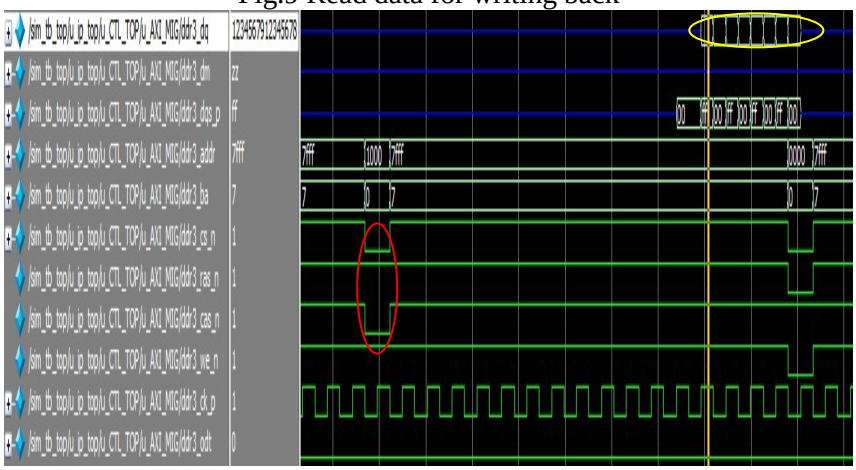
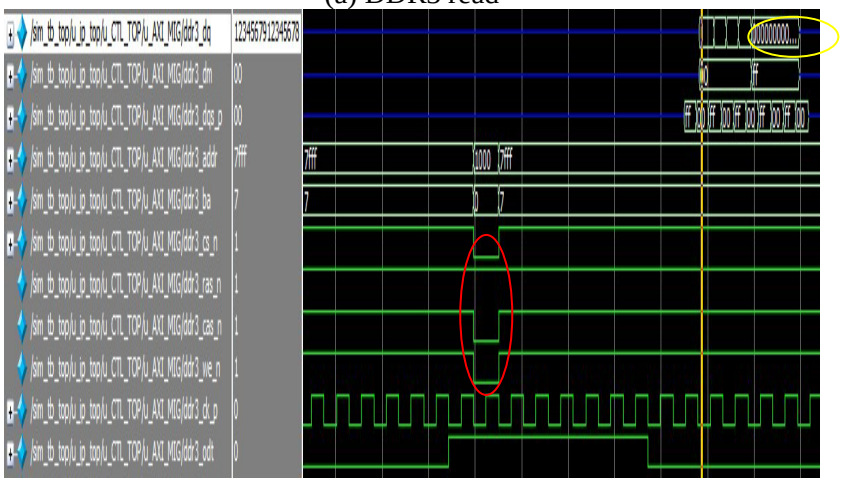


Fig.9 Read data for writing back

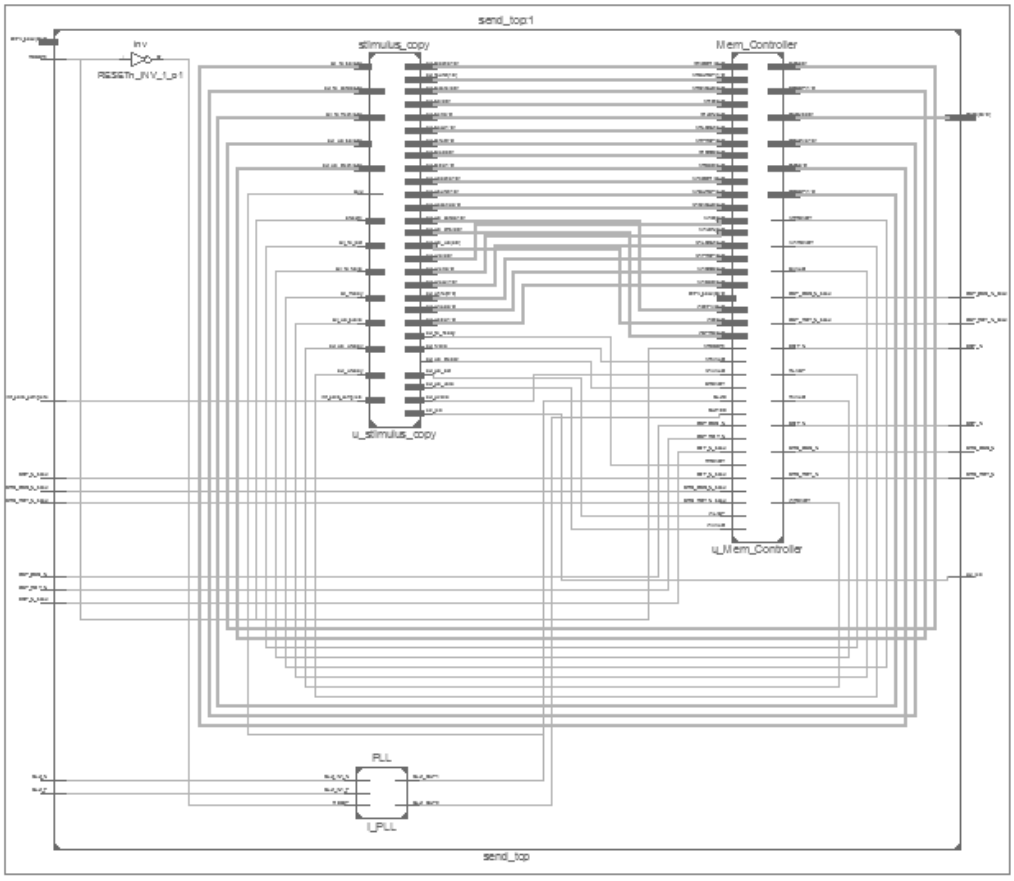


(a) DDR3 read

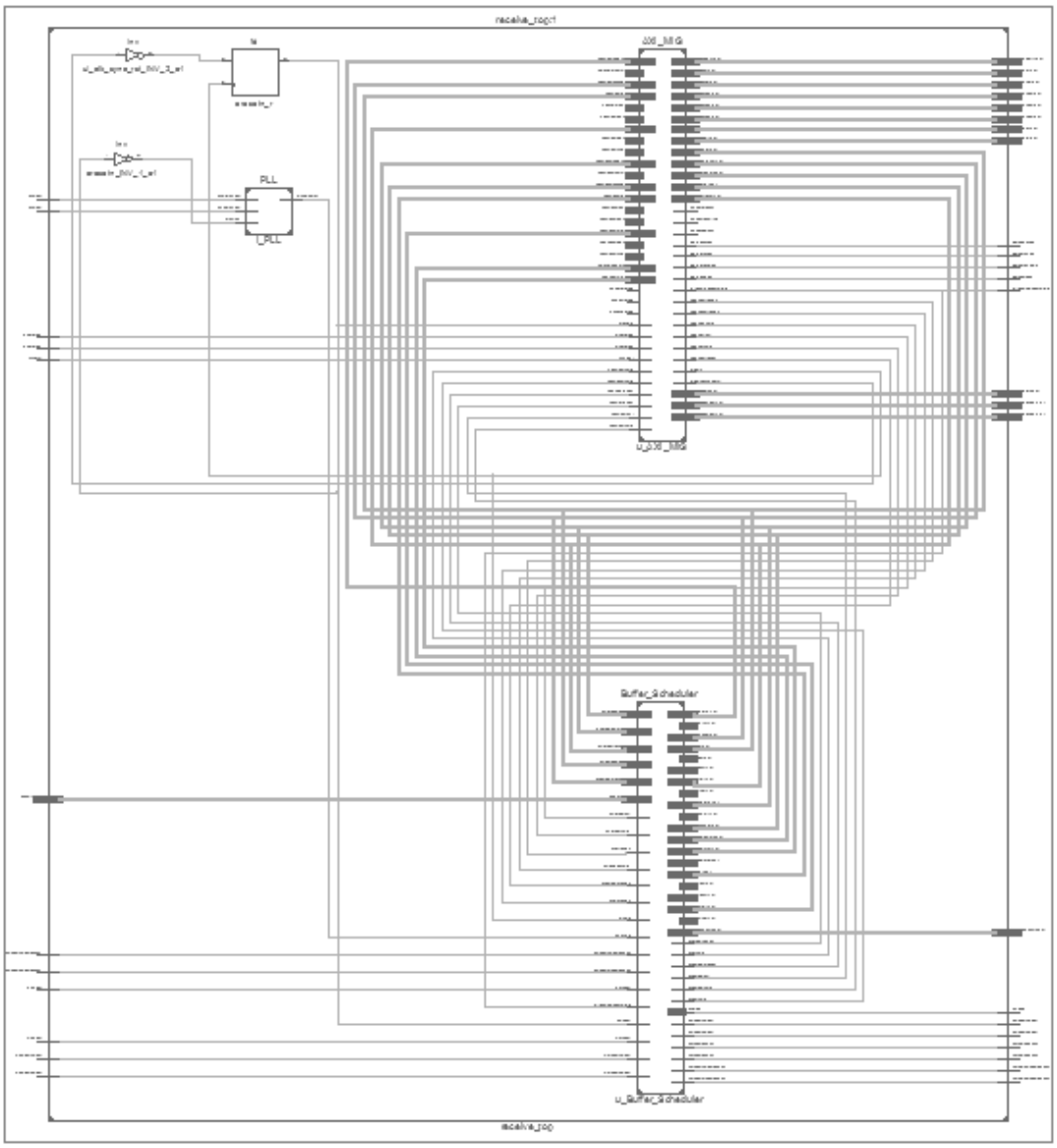


(b)DDR3 write

Fig.10. DDR3 read and write operations



(a) Packets sending side



(b) Packets receiving side

Fig.14 Logic circuits of memory accessing system