International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in

Vol.3., Issue.3, 2015

REVIEW ARTICLE

ISSN: 2321-7758

AXI COMPATIBLE EXTERNAL MEMORY CONTROLLER FOR SOC APPLICATIONS

GOKULKIRAN PRATHIBHA[1], G.M.RAJATHI[2]

[1]ECE-PG,SREC,Coimbatore,Tamilnadu,India

[2]ECE Professor, SREC, Coimbatore, Tamilnadu, India

Article Received:07/05/2015 | Article Revised on:15/05/2015 | Article Accepted on:20/05/2015

ABSTRACT

Memory devices are most commonly used in various embedded application like networking, image/video processing, Laptops etc. Those applications need more and more cheap and fast memory. Especially in the field of signal processing, requires significant amount of memory. In most of the SOC design, memory devices are commonly used. ARM processor is widely used in SOC's; so that we are focusing in implementing AXI compatible External Memory Controller (EMC) that will support for different memory types. This module blocks provides memory controller functionality for SRAM, NOR Flash and PSRAM/Cellular RAM memory devices. This core is Simulated, Synthesized in Xilinx Vivado Design Suite and Implemented in ZYBO ZYNQ Device.

Key words: AXI, External memory controller(EMC), Vivado design suite, ZYBO ZYNC 7000

I.INTRODUCTION

A system on a chip or system on chip (SoC or SOC) is an integrated circuit (IC) that integrates all components of a computer or other electronic system into a single chip. It may contain digital, analog, mixed-signal,and often radio-frequency functions all on a single chip substrate. SoCs are very common in the mobile electronics market because of their low power consumption. A typical application is in the area of embedded systems. Systems-on-Chip (SoC) and in particular embedded real-time systems typically consist of several computational elements. These elements fulfill different tasks for processing an overall solution. Let's take a set-top box for TV sets as an example . A set-top must generate a TV-signal for a particular TV channel from a digital satellite signal. This process takes different tasks. One task is to split the incoming digital signal into data streams, such as video and audio. Another task is to convert the video stream into an actual TV-signal. One more conversion has to be made to turn the audio stream into an audio signal for the TV set. Meanwhile, another task handles the user input such as changing the channel when the remote control is pressed. All these tasks have to be done in parallel and are bound by real-time deadlines. The cost of missing these deadlines is visible as black boxes on the screen or audible as noise. This is unacceptable and therefore it is necessary to always deliver this data within hard real-time deadlines.

A processor needs to interact with other processors, memories or I/O devices to complete a task. Currently busses are used to interconnect these IP blocks. The current research in the field suggests using Networks-on-Chip (NoC) to interconnect IP blocks, because NoCs allow more flexibility than

busses. Dynamic memories provide high data rates and high storage capabilities. Nevertheless, dynamic memories lack flexibility in accessing random memory locations efficiently and require refresh cycles to keep the content. Allowing efficient communication between IP blocks and a shared memory requires a memory controller. IP blocks, which communicate with the memory, are named herein after requestors. The memory controller arbitrates between the requestors and manages the memory accesses. The goal of this thesis is to design a memory controller to provide hard real-time guarantees, and provides an efficient communication between IP blocks and the memory. The memory controller should also be easily/efficiently combined with the interconnection. AXI External Memory Controller (EMC) is designed to communicate with the on chip ddr3 memory The adaptable block provides memory controller functionality for SRAM, NOR Flash and PSRAM/CellularRAM memory devices. The core provides an AXI4 Slave Interface that can be connected to AXI4 Master or Interconnect devices in the AXI4 Systems.

II. RELATED WORK

In a SoC, it houses many components and electronic modules; to interconnect these, a bus is necessary. There are many buses introduced in the due course some of them being AMBA [2] developed by ARM, CORE CONNECT [4] developed by IBM, WISHBONE [5] developed by Silicore Corporation, etc. Different buses have their own properties the designer selects the bus best suited for his application. The AMBA bus was introduced by ARM Ltd in 1996 which is a registered trademark of ARM Ltd. Later advanced system bus (ASB) and advanced peripheral bus (APB) were released in 1995, AHB in 1999, and AXI in 2003[6]. AMBA bus finds application in wide area. AMBA AXI bus is used to reduce the precharge time using dynamic SDRAM access scheduler (DSAS) [7]. Here the memory controller is capable of predicting future operations thus throughput is improved. Efficient Bus Interface (EBI) [8] is designed for mobile systems to reduce the required memory to be transferred to the IP, through AMBA3 AXI. The advantages of introducing Network-on-chip (NoC) within SoC such as quality of signal, dynamic routing, and communication links was discussed in [8]. To verify on-chip communication properties rule based synthesizable AMBA AXI protocol checker is used. The block diagram of the AXI Generic Mater Controller has the following blocks and is shown in the fig 1 : 1) Master 2) AMBA AXI4 Interconnect 3) Slave

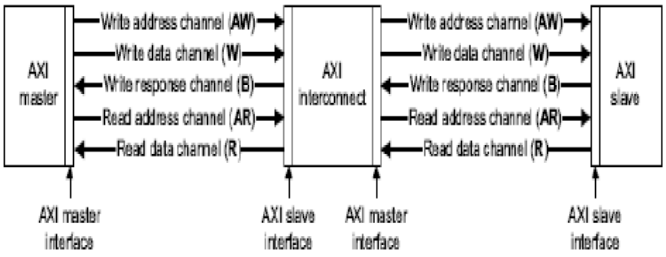

Fig 1: The Block Diagram of AXI Generic Master Controller

The master is connected to the interconnect using a slave interface and the slave is connected to the interconnect using a master interface as shown in fig 2. The AXI4 master gets connected to the AXI4 slave interface port of the interconnect and the AXI slave gets connected to the AXI4 Master Interface port of the interconnect. The parallel capability of this interconnects enables master M1 to access one slave at the same as master M0 is accessing the other.

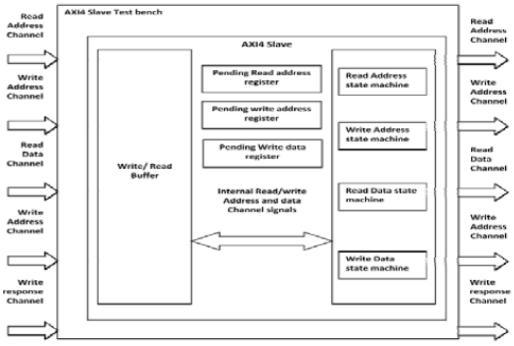

Fig 2: AMBA AXI slave Read/Write block Diagram

III. BLOCK DIAGRAM

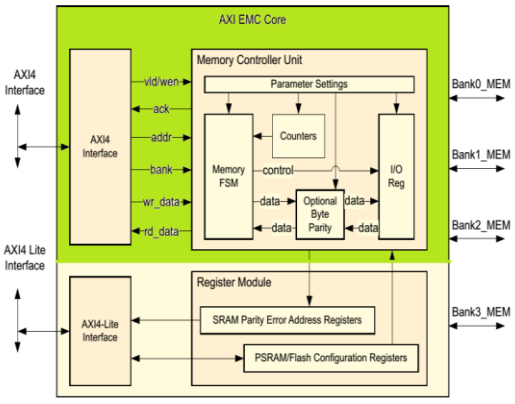

Fig 3.Block Diagram of the AXI EMC Core

IV. MODULE DESCRIPTIONS

The AXI EMC operations can be categorized into five modules:

•AXI4SlaveInterface
• AXI4-Lite Interface
• Memory Controller Unit
• Byte Parity Logic
• EMC Register
• I/O Registers

AXI4 Slave Interface

AXI EMC core provides AXI4 slave interface to map to AXI4 master or AXI4 interconnect devices in the FPGA logic. The AXI4 slave interface of the core complies with AMBA® AXI4 protocol specifications for 32-bit and 64 bit data widths. The core supports single beat or burst AXI4 transactions. The burst transactions for incremental bursts [INCR] can be from 1 beat to 256 beats and the burst transactions for wrapping burst [WRAP] can be 2, 4, 8, or 16 beats. The core also supports AXI4 Narrow and AXI4 Unaligned transfers.

The AXI4 interface of the core also performs the channel arbitration and address decoding functionalities. The channel arbitration implements a round robin algorithm and is applicable only when both AXI4 write channels and AXI4 read channels are active simultaneously. The address decoding logic utilizes the AXI4 address to determine the bank address and read/write address in the selected memory bank. For more details on Memory Address Generation.

The performance of AXI EMC core gets determined by Memory Width, Memory access latencies and AXI burst size. In general larger AXI4 burst sizes can increase the overall performance of the core.

Note: The AXI4 FIXED transactions are not supported by the core and when issued can result in non-deterministic core behavior.

AXI4-Lite Interface

AXI EMC core provides AXI4-Lite slave interface to allow access to Internal Control and Status registers of the core. The AXI4 slave interface of the core complies with AMBA AXI4-Lite protocol specifications for 32-bit data width. The AXI4-Lite interface of the core is optional and is enabled only when the Enable Internal registers option is set.

Memory Controller Unit

The Memory Controller Unit performs Reads/Writes to the external memories in compliance with access mechanism defined for the selected memory. It generates the memory chip select (or bank select) and the control signals associated with the memory in the selected bank. The Read/Write Address, Write Data and the Bank Address generated for the memory is as provided by the AXI4 interface module to the Memory Controller Unit.

The Memory Controller Unit in addition aligns the address and data received from the AXI4 interface module to match the address and data width configured for the bank. Different data widths for various banks are now only supported when selected memory is Synchronous/Asynchronous SRAM. For other memory types, data width shave to be same across banks. The width conversion in this case is performed dynamically as per the width defined for each memory bank. The Memory interface timing for the asynchronous memory interfaces are parameterized and can be set independently for each bank. The Memory Controller Unit maintains internal counters and registers to match the interface timing as defined by these parameters.

Byte Parity Logic

The Byte Parity Logic is applicable only for the SRAM type and is used to calculate the parity bit for each data byte written to or read from the memory. This parity bit is attached to the data byte and is then written to or read from the memory. The parity calculation logic is valid only when Parity option is set to either Odd Parity or Even Parity. The Parity type has to be the same across banks. The Parity configuration option allows No Parity, Even Parity and Odd Parity options. For a memory read, the parity bit is calculated on a read byte and then compared with the parity bit read from memory. When the parity error occurs, the AXI4 logic responds with an AXI SLVERR response. The error address is updated in the Parity Error Address register. For more than one SRAM bank, the number of Parity Error Address registers is defined by the Number of Memory Banks configuration option.

EMC Register

There are two register sets in the EMC Register Module: the Parity Error Address register (PERR_ADDR_REG_x) and the PSRAM/Numonyx Flash Configuration register (PSRAM_FLASH_CONFIG_REG_x). The

PERR_ADDR_REG_x registers contain the address for the parity error that occurred.

The PSRAM_FLASH_CONFIG_REG_x register is provided to configure the PSRAM or flash memory access mechanisms such as operation mode for the memory.

I/O Registers

The I/O register module provides additional timing control for synchronous SRAM memories. All input/output signal on the memory interface is driven on the rising edge of the EMC clock

V.SIMULATION

The simulation is done using Vivado Design Suite and the language used is VHDL

Here are some of the results of the simulation

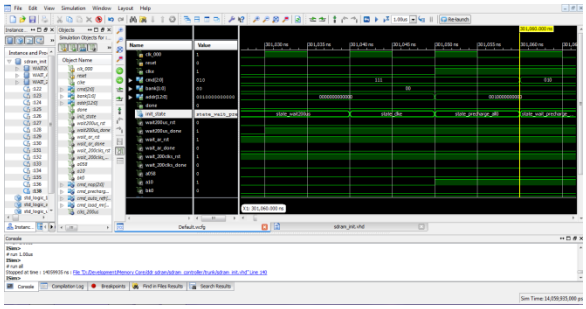

Fig. 4 Simulation for SD RAM Initialization

The above result displays the initialization of the SD RAM

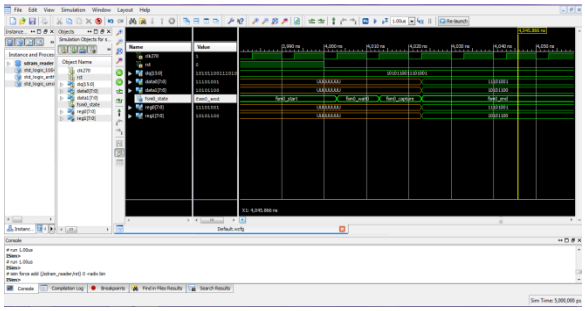

Fig. 5 Simulation of SD RAM Reader

This is the simulation result of the proposed SD RAM reader

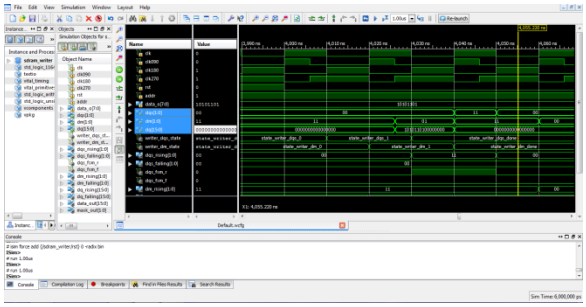

Fig. 6 Simulation for SD RAM Writer

The above waveform depicts the SD RAM writer functionality.

VI HARDWARE IMPLEMENTATION

The Following reports are obtained by implementing the above simulated components in the ZYNQ 7000 Fpga. The advantages of ZyncFpga is that it has both the Programmable logic and the programmable System in a single SoC.

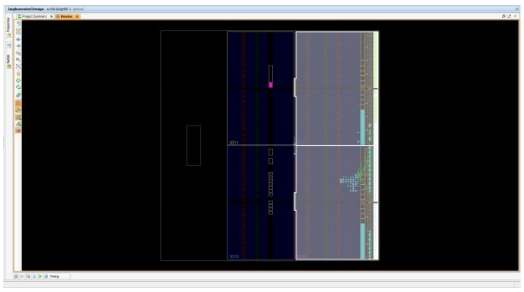

Fig. 7 Implementation Design of EMC

The above figure displays the implementation design of the external memory controller. The green lines refer to the floor planning structure whereas the the blue blocks represent the placement of the synthesized logical blocks to be oocupied in the FPGA.

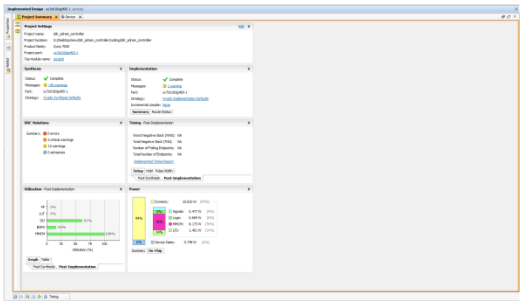

Fig.8 Snapshot of Implementation

The above snapshot shows the post implementation details including the static and dynamic power calculations.

VIICONCLUSION

The DDR3 RAM is initiated using the designed external memory controller and the read and write operations are simulated and the External memory controller is synthesized using the vivado design suite and implemented on the ZYNC 7000 device. The snapshots of various simulation and implementation details are displayed.

REFERENCE

[1] Shaila S Math, Manjula R B, "*Survey of system on chip buses based on industry standards*", Conference on Evolutionary Trends in Information Technology(CETIT), Bekgaum,Karnataka, India, pp. 52, May 2011

[2] ARM, AMBA Specifications ev2.0).[Online]. Available at http://www.arm.com,

[3] ARM, AMBA AXI Protocol Specification (Rev 2.0).

[4] IBM, Core connect bus architecture. IBM Microelectronics.[Online].Available: http://www.ibm.com/chips/products/corec on nect

[5] Silicore Corporation, Wishbone system-on-chip (soc) interconnection architecture for portable ip cores,

[6] ARM, AMBA AXI protocol specifications, Available at, http://www.arm.com, 2003

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

[8] Na Ra Yang, Gilsang Yoon, Jeonghwan Lee, Intae Hwang, Cheol Hong Kim, Sung Woo Chung and Jong Myon Kim, "*Improving the System-on-a-Chip Performance for Mobile Systems by Using Efficient Bus Interface*", IEEE transl, International Conference on Communications and Mobile Computing, Vol 4, pp. 606-608,