Implementation of AMBA compliant Memory Controller on a FPGA

Shilpa Rao and Arati S. Phadke

Abstract: As microprocessor performance has relentlessly improved in recent years, it has become increasingly important to provide a high-bandwidth, low-latency memory subsystem to achieve the full performance potential of these processors. In the past years, improvements in memory latency and bandwidth have not kept pace with reductions in instruction execution time. Caches have been used extensively to patch over this mismatch, but some applications do not use caches effectively. The result is that the memory access time has been a bottleneck which limits the system performance. A Memory Controller is designed to cater to this problem. The Memory Controller is a digital circuit which manages the flow of data going to and from the main memory. It can be a separate chip or can be integrated into the system chipset. This paper revolves around building an Advanced Microcontroller Bus Architecture (AMBA) compliant Memory Controller as an Advanced High-performance Bus (AHB) slave. The whole design is captured using VHDL, simulated with ModelSim and configured to a FPGA target device belonging to the Virtex4 family using Xilinx.

Keywords: **AHB, ARM, AMBA, Memory Controller.**

I. INTRODUCTION

Since 1985, microprocessor performance has improved at a rate of 60% per year. In contrast, latencies have improved by only 7% per year, and bandwidths by only 15-20% per year. The result is that the relative performance impact of memory accesses continues to grow. In addition, the demand for memory bandwidth has increased proportionately (and possibly even super linearly). These trends make it increasingly hard to make effective use of the tremendous processing power of modern microprocessors.

Hence, Memory Controllers are built to attack these disadvantages. It provides a high bandwidth and low latency access to o□-chip memory. A Memory Controller translates the accesses from the requestors into commands understood by the memory. It generates the necessary signals to control the reading and writing of information from and to the memory, and interfaces the memory with the other major parts of the system. The front end of the general memory controller buffers requests and responses and provides an interface to the rest of the system. The back end provides an interface towards the target memory.

Shilpa Rao and Arati S. Phadke are with Department of Electronics Engineering, K.J. Somaiya College of Engineering, Vidyavihar, Mumbai-400077, India, Email: shilparao2000@yahoo.com

The embedded processors, such as ARM, of today's technology operate up to a few hundred mega-hertz. Most of embedded and SoC applications are suited to operate with these frequency requirements. But some high end applications, such as InfiniBand, require much more processing power and should operate at much higher frequency. In such a case, it is desirable to implement embedded and SoC application with multiple processors. AMBA (Advanced Micro-controller Bus Architecture) is one of the widely used system bus architecture which caters to the above requirements.

The memory controller is compatible with Advanced High-performance Bus (AHB) which is the latest generation of AMBA bus, and hence called as "AHB-MC". The AHB-MC is an Advanced Microcontroller Bus Architecture (AMBA) compliant System-on-Chip (SoC) peripheral [6]. It is developed, tested, and licensed by ARM Limited.

The AHB - MC has several features:

- Adheres to the AMBA AHB protocol
- Synthesizable VHDL / Verilog RTL source
- Supports multiple memory devices of different types like SRAM, ROM, and Flash
- Provides test bench and verification vectors
- Shares data path between memory devices to reduce pin count
- Supports AHB single beat, 4 beat and 8 beat wrapping bursts and split transaction.

II. ARCHITECTURE OF AMBA-AHB

The Advanced Microcontroller Bus Architecture (AMBA) specification defines an on-chip communications standard for designing high-performance embedded microcontrollers. AHB is a new generation of AMBA bus which is intended to address the requirements of high-performance synthesizable designs [2].

The AHB-MC mainly consists of five modules: AHB slave interface, configuration interface, external memory interface, memory system, and data buffers [1]. Fig. 1, shows the architecture of AHB-MC.

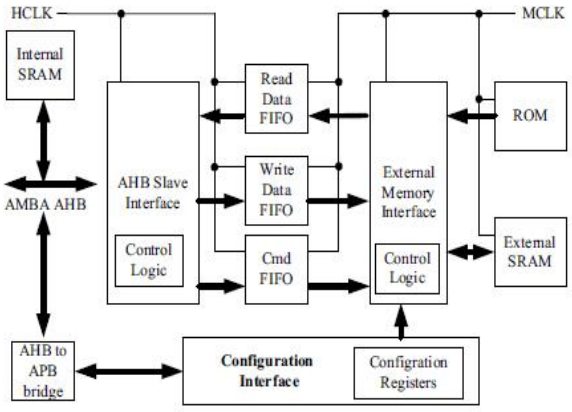

Fig. 1 Architecture of AHB-MC

A. AHB Slave Interface

The AHB slave interface [4] converts the incoming AHB transfers to the protocol used internally by the AHB-MC. Because of the design of the internal inter-connect, some optimizations are made in the interface to improve performance. It is a fully validated component which ensures that it obeys both the AHB protocol and the internal protocol that the interconnect uses.

B. Configuration Interface

The internal memory controller has an APB configuration port. The AHB configuration port is mapped to it using an AHB to APB bridge. The main function of the configuration interface is to change the certain configuration registers according to the commands from AHB to APB bridge which converts AHB transfers from the configuration port to the APB transfers that the configuration interface [3] require. It also provides the read and write enable signals to the data buffers.

C. External Memory Interface

The external memory [4] issues commands to the memory from the command FIFO, and controls the cycle timings of these commands. It generates the 1) Memory bank select and 2) Memory write control signals.

1) Memory Bank Select

AHB-MC has four memory banks, which are selected by XCSN signal. The XCSN chip select signal is controlled by the address of a valid transfer, and the system memory map mode. Since, the system will change the memory map after the system boot, the AHB-MC is designed to support a remap signal which is used to provide a different memory map [7]. So before the system memory is remapped, the boot ROM at 0x3000 0000 is also mapped to the base address of 0x0000 0000. The relationship between the inputs and the generated value of XCSN [4] is shown in Table I.

hselect	remap	haddr[29:28]	XCSN
0	X	XX	1111
1	0	00	1110
1	0	01	1101
1	0	10	1011
1	0	11	0111
1	1	00	1110
1	1	01	1101
1	1	10	1011
1	1	11	0111

Table I. XCSN Coding

2) Memory write control

The external memory controls XWEN for writes in word (32-bit), half-word (16-bit), and byte (8-bit) quantities. The external memory uses hsize[1:0] and haddr[1:0] to select the width and order of each write to memory. Table II shows the relationship between XWEN and the inputs from AHB bus.

Table II. XWEN coding

hsize	haddr[1:0]	XWEN
10(word)	XX	0000
01(half)	0X	1100
01(half)	1X	0011
00(byte)	00	1110
00(byte)	01	1101
00(byte)	10	1011
00(byte)	11	0111

D. Memory System

In the ARM architecture, instructions are all 32-bits, while instructions are 8-bits in the external ROM and SRAM. Therefore the lowest two addresses of ROM and SRAM are not connected to the external address bus. Additionally, to support byte writing, SRAM needs to be separated as four independent banks or has a byte-write enable signal. The basic memory system architecture is shown in Fig. 2.

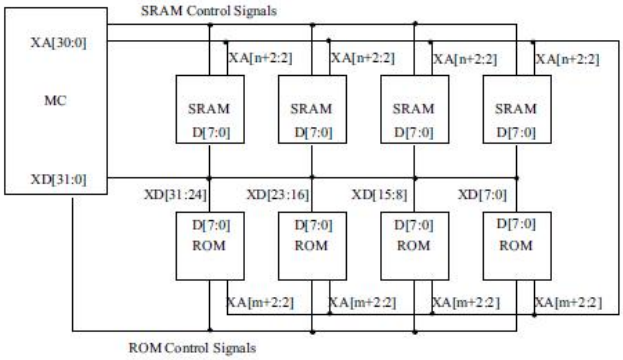

Fig. 2 Memory system architecture

The AHB-MC has two clock domains [3]: AHB clock domain (hclk) and external memory clock domain (mclk). Asynchronous FIFO is used between two clock domains as a data buffer. The main benefit of asynchronous clocking is that the system performance can be maximized, while running the memory interface at a fixed system frequency. Additionally, in sleep-mode situations when the system is not required to do much work, you can lower the frequency to reduce power consumption.

E. Data Buffers

One of the most popular methods of passing data between clock domains is to use a FIFO [5]. A dual port memory is used for the FIFO storage. One port is controlled by the sender which puts data into the memory as fast as one data word (or one data bit for serial applications) per write clock. The other port is controlled by the receiver, which pulls data out of memory, one data word per read clock. Two control signals are used to indicate if the FIFO is empty, full or partially full.

III. SYNTHESIS AND SIMULATION RESULTS

The architecture of AHB-MC shown in Fig. 1, can be represented in form of a block diagram as shown in Fig. 3. The overall inputs and outputs of the system are also indicated.

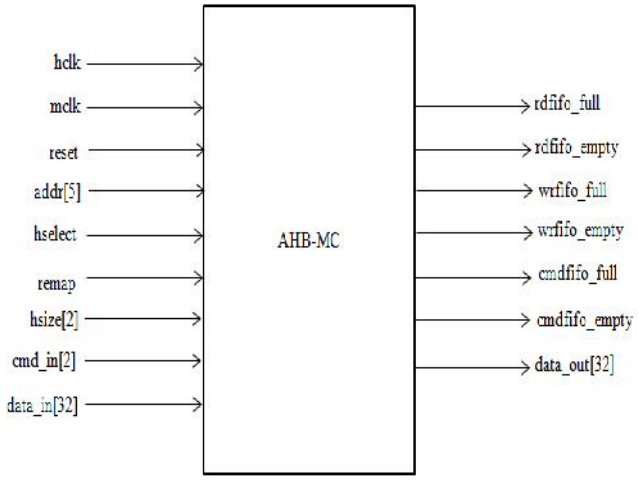

Fig. 3 Overall inputs and outputs of AHB - MC

The functionality of the modules of AHB-MC is tested using Xilinx ISE 10.1 tool in VHDL and simulated using ModelSim 6.2. The following steps are followed during the implementation of this design: Code generation, Simulation, Synthesis, Power analysis, RTL schematic, Translate, Map, Place and Route, and finally Configured to a target device.

The RTL Viewer of AHB - MC is shown in Fig. 4. From this figure, it can be observed that asynchronous FIFOs are used between two clock domains as data buffers.

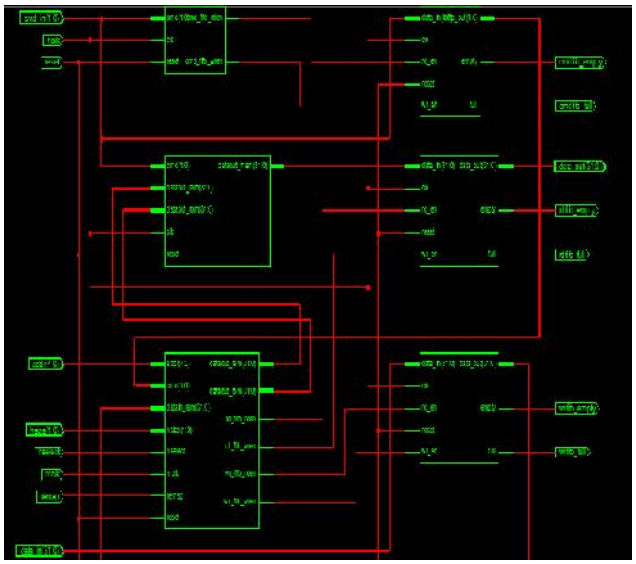

Fig. 4 RTL Viewer of AHB-MC

For a target device xc4vlx15-12sf363 belonging to the Virtex4 family the Device Utilization Summary is shown in Table III.

Table III. Device Utilization Summary

Logic Utilization	Used	Available	Utilization
Number of Slices	2141	6144	34%
Number of Slice FFs	2377	12288	19%
Number of 4 input LUTs	2409	12288	19%
Number of bonded IOBs	84	240	35%
Number of GCLKs	16	32	50%

Power analysis of AHB-MC was carried out by using Xilinx's XPower Analyzer. Table IV which shows the Power report indicates that the total power dissipated by the AHB - MC is 412mW.

Table IV. Power Report of AHB-MC

Power summary	I(mA)	P(mW)
Total estimated power consumption		412
Total Vccint 1.20V	182	219
Total Vccaux 2.50V	77	193
Total Vcco25 2.50V	0	0

Fig. 5, shows read with zero wait states form the ROM. The cmd_in (10 which indicates the ROM read operation) is registered at falling edge of hclk (AHB bus clock), after which rd_fifo_rden (Read FIFO read enable) signal goes high. The read data then reaches data_out (AHB data bus) at falling edge of hclk. Write with zero states to the external RAM is also

shown in Fig. 5. A write operation to RAM is initiated when cmd_is set to 00. Then the address is send to AHB address bus and wr_fifo_rden (Write FIFO read enable) signal goes high to enable the data from data_in (AHB input data bus) to be stored in the RAM.

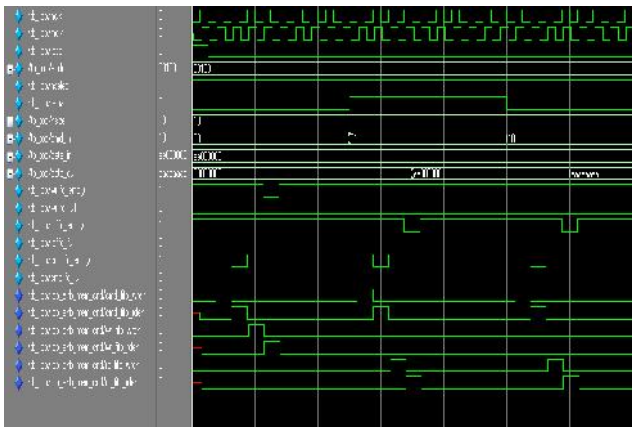

Fig. 5 Read and write with zero wait state from ROM and to RAM respectively

IV. CONCLUSIONS

The read write operation is accomplished with zero wait states from the external ROM and the write operation with zero states to the external RAM. By proposing the parallel communication in AMBA-AHB, the data transfer operation will be fast as compared to serial communication. It also provides the opportunity to use masters and slaves up to 16 numbers and the data of every master is read and written simultaneously.

With the increase in system frequency, it is hard to achieve address decoding and memory access operations in one clock cycle due to which wait states are inserted. But the method of inserting wait states will result in dramatic drop in system performance. Therefore, the burst method is presented in this paper to resolve the problem. When the first beat of a burst is accepted, it contains data about the remaining beats. For example, when AHB-MC gets the first beat of a read burst, all the data required to complete the transfer can be read from memory and restored in the read data FIFO. So this first transfer has some delay before data is returned. But subsequent beats of the burst can have less delay because the data they require might have already been prepared in the FIFO.

Power dissipation is becoming a limiting factor for high performance microprocessor design due to ever increasing device counts and clock rates. The proposed architecture of AHB-MC also aims to optimize power. It is found that it consumes a power of 412mW with a maximum frequency of 355.77MHz.

REFERENCES

[1] Hu Yueli,Yang Ben, "Building an AMBA AHB compliant Memory Controller", IEEE, 2011.

[2] "AMBA Specification (Rev2.0)", ARM Inc, 1999.

[3] "PrimeCell AHB SRAM/NOR Memory Controller", Technical Reference Manual, ARM Inc, 2006.

[4] "AHB Example AMBA System", Technical Reference Manual, ARM Inc, 1999.

[5] Clifford E. Cummings, Sunburst Design Inc., "Synthesis and scripting Techniques for Designing Multi-Asynchronous Clock Designs", 2001.

[6] Sudeep Pasricha, "On-Chip Communication Architecture Synthesis for Multi-Processor Systems-on-Chip", University of California, 2008.

[7] "PrimeCell Synchronous Static Memory Controller", Technical Reference Manual, ARM Inc, 2001-2005.

Shilpa Rao has obtained Bachelor of Engineering in Instrumentation Technology at Sri Jayachamarejendra College of Engineering, Mysore. She is currently pursuing Master of Engineering in Electronics Engineering from K. J. Somaiya College of Engineering, Mumbai. She has worked as a software programmer for two years and also as a lecturer in the Department of Electronics for three years. Her area of interests include Low Power VLSI Design, Hardware Descriptive Languages, and Digital System Design.

Prof. Arati S. Phadke is currently working as an Associate Professor in the Department of Electronics Engineering in K. J. Somaiya College of Engineering, Mumbai with 22 years of experience in the field of teaching. She completed her Bachelor of Engineering in Electrical Engineering from College of Engineering, Pune and Master of Technology in Power Electronics from Indian Institute of Technology, Mumbai. She is an active member of IEEE since 1993 and has been a Student Branch Counselor from 2003-2009. She has been the recipient of the Best Teacher Award, KJSCE, Mumbai in 2000 for her excellence in academic performance. She has delivered a lot of guest lectures on Hardware Descriptive Languages and Embedded Systems for faculty and students of various engineering colleges. Her research interests include Digital System Design, Embedded Systems, Microprocessors and Microcontrollers.