ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 1, Issue 7, September 2012

Improving Memory Access time by Building an AMBA AHB compliant Memory Controller

Arun G
M.Tech(Student) ,VLSI
SJBIT, Bangalore-60

Vijaykumar T
Associate Lecturer, Dept. of ECE
SJBIT, Bangalore-60

***Abstract*—Memory access time has been a bottleneck in many microprocessor applications which limits the system performance. Memory controller (MC) is designed and built to attacking this problem. The memory controller is the part of the system that, well, controls the memory. The memory controller is normally integrated into the system chipset. This paper shows how to build an Advanced Microcontroller Bus Architecture (AMBA) compliant MC as an Advanced High-performance Bus (AHB) slave. The MC is designed for system memory control with the main memory consisting of SRAM and ROM. Additionally, the problems met in the design process are discussed and the solutions are given in the paper.**

Keywords-.ARM; AMBA; Memory Controller; AHB bus

I. INTRODUCTION

Memory access time has been a bottleneck in many microprocessor applications which limits the system performance. Memory controller (MC) is designed and built to attacking this problem. Memory controller (MC) is designed and built to attacking this problem. The memory controller is the part of the system that, well, controls 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 memory controller is normally integrated into the system chipset. In this paper, an Advanced Microcontroller Bus Architecture (AMBA) compliant memory controller is designed for system memory control with the main memory consisting of SRAM and ROM. The memory controller is compatible with Advanced High-performance Bus (AHB) which is a new generation of AMBA bus, so we call it "AHB-MC".

The AHB-MC has several features which are shown as flows:

- ➢ Designed with synthesizable HDL for Application Specific Integrated Circuit (ASIC) synthesis
- ➢ Supports multiple memory devices including static random access memory (SRAM), read-only memory (ROM)
- ➢ Complies with AMBA AHB protocol
- ➢ Supports one to four memory banks for SRAM and ROM
- ➢ Programmable memory timing register and configuration registers
- ➢ Shared data path between memory devices to reduce pin count

This paper describes how to build the AHB-MC. And combining the problem met in the process of designing, the corresponding solutions are presented. Finally, the simulation results are presented.

II. ARCHITECTURE OF AHB-MC.

The AHB-MC mainly consists of three modules: AHB slave interface, configuration interface, and external memory interface. Figure 1 shows the architecture of AHB-MC.

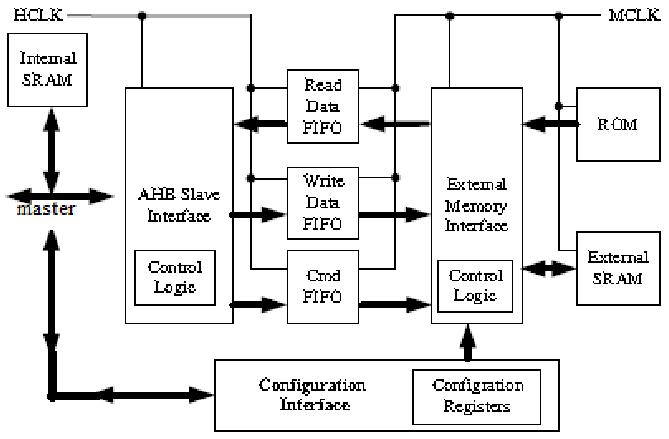


Figure 1. Architecture of AHB-MC

A. AHB slave interface

The AHB slave interface converts the incoming AHB transfers to the protocol used internally by the AHB-MC. The state machine is shown in Figure 2.

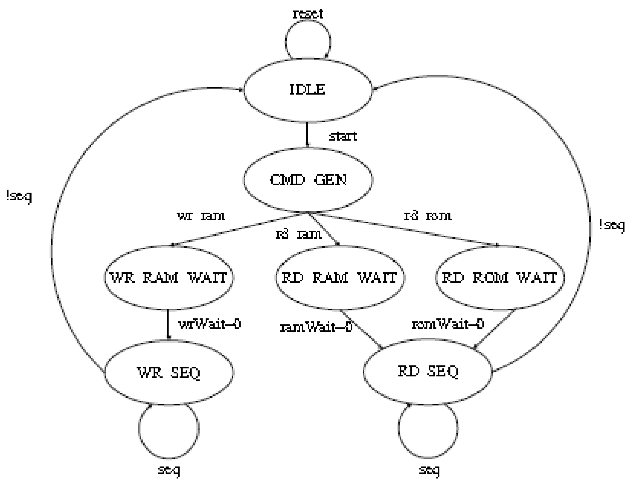


Figure 2. AHB slave interface state machine

B. External memory interface

The external memory issues commands to the memory from the command FIFO, and controls the cycle timings of these commands. The state machine is shown in Figure3.

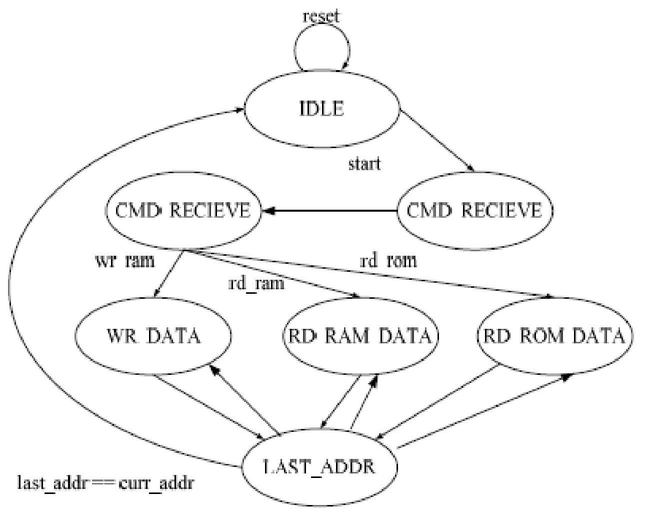


Figure 3. External memory interface state machine

Figure 4 and Figure 5 show the timing of a read from memory and a write to memory with two wait states

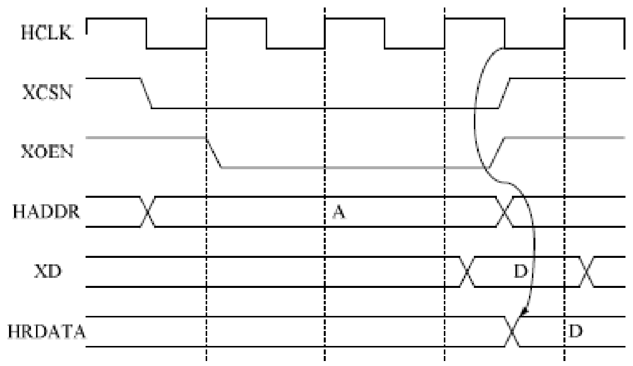


Figure 4. Memory read with two wait states

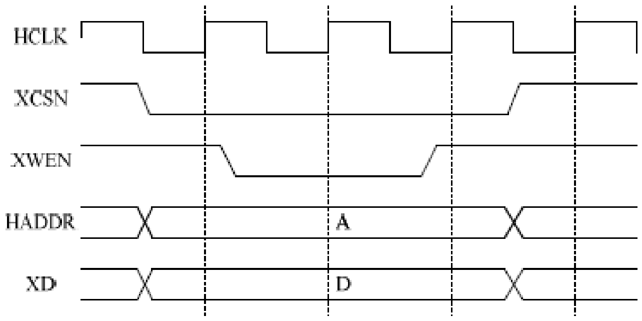
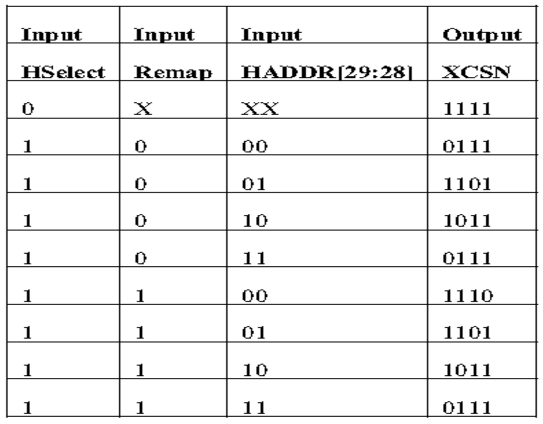


Figure 5. Memory write with two wait states

1) Memory bank select

Because system will change the memory map after system boot, AHB-MC is designed to support a remap signal which is used to provide a different memory map. AHB-MC has four memory banks, which are selected by XCSN signal. The XCSN signal is controlled by the address of a valid transfer, and the system memory map mode. So before the system memory is remapped, the boot ROM at 0x3000 0000 is also mapped to the base address of 0x0000 0000 as shown in Table 1.

Table 1. XCSN coding



Input	Input	Input	Output
HSelect	Remap	HADDR[29:28]	XCSN
0	X	XX	1111
1	0	00	0111
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

2) Memory write control

To support for writing in word (32-bits), half-word (16-bits) and byte (8-bits), the XWEN signal is used in the AHB-MC. Table 2 shows the relationship between XCSN and the inputs from AHB bus.

Table 2. XWEN coding

HSIZE[1:0]	HADDR[1:0]	XWEN[3:0]
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

III. Burst transfer support

With the increasing system frequency, it's hard to accomplish the address decoding and memory access operations in one clock cycle. Therefore, wait states are inserted into the data cycle to ensure there is enough time for address decoding and memory accessing. But the method that inserting wait state will cause system performance drop dramatically.

Therefore, a sequential-access (burst) method is presented to resolve this problem in this paper. In this method, all AHB fixed length burst types are directly translated to fixed length bursts, and all undefined length INCR bursts are converted to INCR4 bursts. Burst operation has performance benefits because when the first beat of a burst is accepted, it contains data about the remaining beats. For example, when AHB-MC got 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.

To further improve the system performance, a RETRY response is used that AHB-MC can release the bus when it is preparing the data. This mechanism allows the transfer to finish on the bus and therefore allows a higher-priority master to get access to the bus.

The state machine is showed in Figure 6.

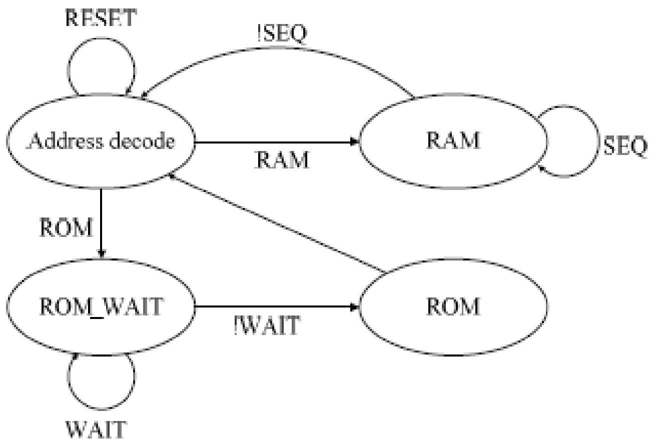


Figure 6. Sequential-access state machine

IV. 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 Figure 7.

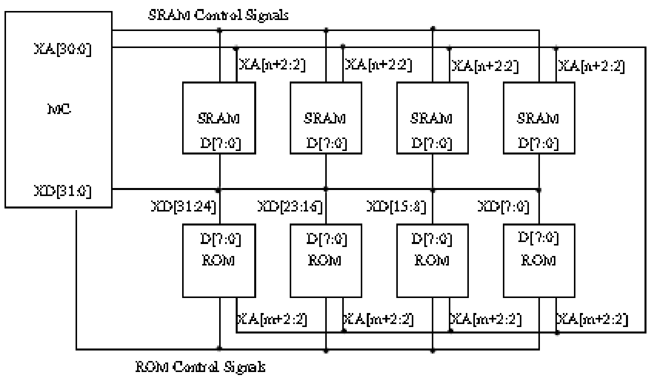


Figure 7. Memory system architecture

V. Asynchronous clock

The AHB-MC has two clock domains: AHB clock domain and external memory clock domain as shown in Figure 8. Asynchronous FIFO is used between two clock domains as a data buffer.

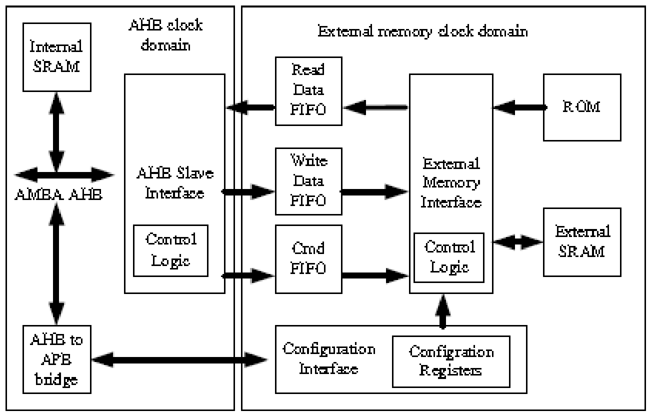


Figure 8. Clock domains

The main benefit of asynchronous clocking is that you can maximize the system performance, 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.

However, asynchronous clock will cause the flip-flop going metastable state and not converging to a legal stable state by the time the output must be sampled again as shown in Figure 9. To resolve this problem, the most common way is inserting a two-flip-flop synchronizer as shown in Figure 10.

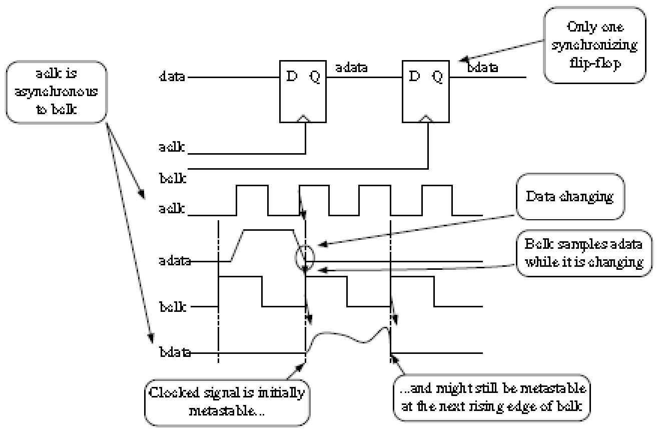

aclk is asynchronous to bclk
data
D Q
adata
D Q
bdata
Only one synchronizing flip-flop
aclk
bclk
aclk
Data changing
adata
Bclk samples adata while it is changing
bclk
bdata
Clocked signal is initially metastable...
...and might still be metastable at the next rising edge of bclk

Figure 9. Asynchronous clocks and synchronization failure

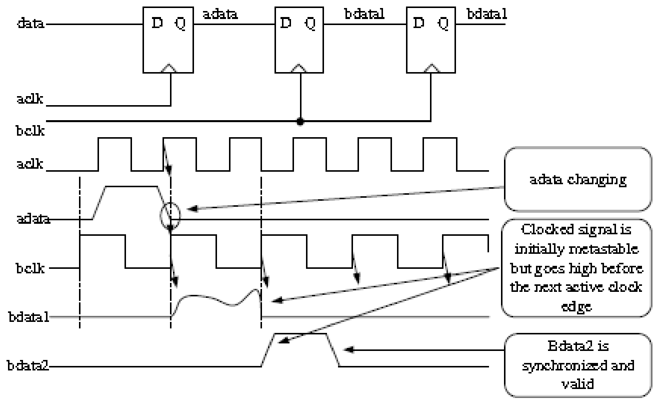

data
D Q
adata
D Q
bdata1
D Q
bdata1
aclk
bclk
aclk
adata changing
adata
Clocked signal is initially metastable but goes high before the next active clock edge
bclk
bdata1
bdata2
Bdata2 is synchronized and valid

Figure 10. Two-flip-flop synchronizer

VI. Verification and Simulation Results

The verification method used in this paper, is to put the AHB-MC into a minimum system which consists of ARM core, AHB bus, and AHB-MC as shown in Figure 11.

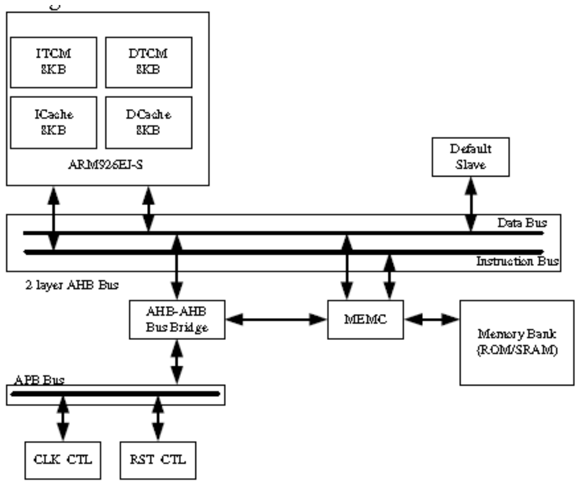


Figure 11. Minimum system for test

The code used for testing is put in ROM, if the system can work correctly, then we know the testcase past. The simulation waveforms of a simple test code are shown. Figure 12 shows read with zero wait states form the external ROM. The address is registered at rising edge of hclk (AHB bus clock), after which ex_oen (external memory read enable) signal goes high, then read data reach hrdata (AHB read data bus) at falling edge of hclk.

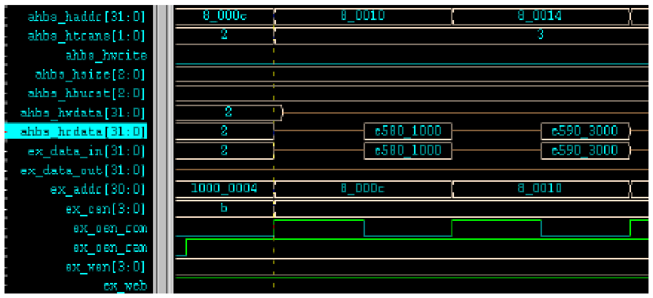


Figure 12. Read with zero wait state from ROM

Write with zero states to the external RAM is shown in Figure 13. A write operation is initiated by hwrite going high. Then the address is send to external memory address bus and ex_wen (external memory write enalbe) signal goes low to enable the data from hwdata (AHB write data bus) stored in the RAM

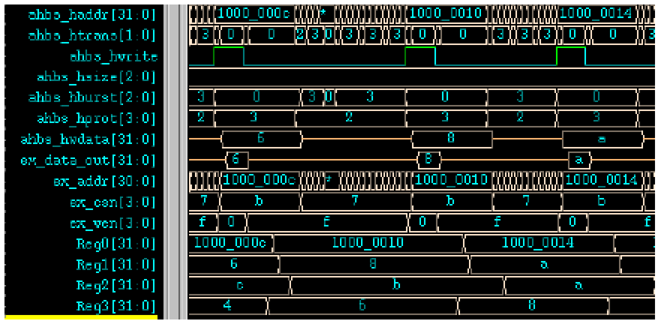


Figure 13. Write with zero wait state to RAM

REFERENCES

[1]http://en.wikipedia.org/wiki/Programmable_Interrupt_Controller
[2] http://en.wikipedia.org/wiki/Intel_8259
[3]http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0024b/index.html
[4] Nazeih M.Batros, " HDL Programming VHDL and Verilog", Dreamtech Press 2008 Reprint.
[5] http://porta/.acm.org/citation.cfm
[6] http://www.thesatya.com/8259.html
[7]http://bochs.sourceforge.net/techspec/intel-8259A pic.pdf.gz
[8] http://infocenter.arm.com/help/topic/com.arm.doc
[9] "AMBA Specification (Rev2.0)", ARM Inc.
[10] "PrimeCell AHB SRAM/NOR Memory Controller", Technical Reference Manual, ARM Inc.
[11] "AHB Example AMBA System", Technical Reference Manual, ARM Inc.
[12] Carter, J.; Hsieh,W., "Impulse: building a smarter memory controller", High-Performance Computer Architecture, Dept. of Comput. Sci., Utah Univ., Salt Lake City, UT.
[5] Clifford E. Cummings, "Synthesis and scripting Techniques for Designing Multi-Asynchronous Clock Designs", Sunburst Design, Inc

Author: Arun G
Mobile: 7204386009