A NOVEL DESIGN APPROACH TO AN AMBA AHB COMPLIANT MEMORY CONTROLLER

Nithin Joe and Anjali Brite
Department of Electronics and Communication Engineering, Nehru College of Engineering and Research Centre, Pampady, Thrissur, Kerala, India.

ABSTRACT
Microprocessor performance has improved rapidly these years. In contrast, memory latencies and bandwidths have improved little. The result is that the memory access time has been a bottleneck 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 Micro controller Bus Architecture (AMBA) compliant MC as an Advanced High-performance Bus (AHB) slave. The Advanced Microcontroller Bus Architecture (AMBA) is used as the on-chip bus in system-on-a-chip designs The MC is designed for system memory control with the main memory consisting of SRAM and ROM. The design perspective presented over here is very simple and helps to understand on development of a logical design from available theoretical details. This paper aims to reveal the fact that the efficiency of a memory controller can be increased to a large extend with the help of AMBA AHB.

KEYWORDS: AMBA, memory controller, AHB bus.

I. INTRODUCTION
The AHB MC is an Advanced Microcontroller Bus Architecture (AMBA) compliant System-on-Chip (SoC) peripheral. It is developed, tested, and licensed by ARM Limited [1]. The memory controller is the part of the system that, well, controls the memory. The memory controller is normally integrated into the system chipset. The MC is designed for system memory control and to improve the system performance.

The memory controller is compatible with Advanced High-performance Bus (AHB) which is a new generation of AMBA bus, so it is called as “AHB-MC”.

The rate of improvement in microprocessor speed exceeds the rate of improvement in DRAM memory speed. There are a number of reasons to account for this growing disparity. The division of the semiconductor industry into microprocessor and memory camps is the prime reason having the industry split into two camps memory manufacturers and processor manufacturers, also has its inherent disadvantages. While microprocessor performance has been improving at a rate of 60 percent per year, the access time to DRAM has been improving at less than 10 percent per year [6]. Thus though each is improving exponentially, the exponent for the microprocessor is substantially larger than that for the DRAMs. The difference between diverging exponentials also grows exponentially, so although the disparity between processor and memory speed is already an issue, downstream someplace it will be a much bigger one. Hence computer designers are faced with an increasing Processor – Memory Performance Gap, which now is the primary obstacle to improved computer system performance [15]. Because of the growing memory access latencies (measured in processor cycles), any request that misses in the caches may eventually take hundreds of cycles to satisfy. Thus system speed will now be dominated by memory performance. Burst transfer (alternatively burst-
Two common goals in computing system design are increasing performance and decreasing power consumption. Memory controllers employ strategies to efficiently schedule DRAM operations to reduce latency and to utilize DRAM low power modes when possible. One of the most important of these is the page policy, which determines when to close pages in DRAM. An effective DRAM memory controller page policy is important to minimizing power consumption and increasing system performance [8]. According to H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte and O. Mutlu (2011), the impact a memory controller page policy has on performance as measured by the number of page-hits minus page-misses and estimated average memory access latency. Real-time memory trace capture was chosen as the method of collecting data due to the completeness of traces that it provides and the ability to capture memory traffic that has already been filtered by CPU caches. Impulse is a new memory system architecture that adds two important features to a traditional memory controller. First, Impulse supports application-specific optimizations through configurable physical address remapping. By remapping physical addresses, applications control how their data is accessed and cached, improving their cache and bus utilization. Second, Impulse supports prefetching at the memory controller, which can hide much of the latency of DRAM accesses. John Carter and Wilson Hsieh (2010) have described the design of the Impulse architecture, and show how an Impulse memory system can be used to improve the performance of memory-bound programs [4]. Impulse improves performance by 67%. Because it requires no modification to processor, cache, or bus designs, Impulse can be adopted in conventional systems. In addition to scientific applications, we expect that Impulse will benefit regularly stride, memory bound applications of commercial importance, such as database and multimedia programs.
As deep submicron techniques are increasingly developed, it is possible to design and manufacture a System-on-a-Chip (SoC) comprised of various Intellectual Property (IP) cores meeting short time-to-market requirements. Although the design time can be reduced by utilizing reusable IPs, the testing time is significantly increased because of the high complexity of the SoC. Testing cost is mainly affected by the memory size and application time of the Automatic Test Equipment (ATE), the structure of the core test wrapper, Test Access Mechanism (TAM), and test methodology. It becomes crucial to improve the test quality while maintaining the testing cost as low as possible to survive in the emerging silicon market. Advanced Microcontroller Bus Architecture (AMBA) is an on-chip-bus protocol developed by ARM to strengthen the reusability of IP cores, and Test Interface Controller (TIC), External Bus Interface (EBI) and test wrappers are extensively adopted as test IP cores embedded in an SoC. Since in general the functional test requires lengthy test application time without guaranteeing sufficient fault coverage, structural testing techniques such as Built-In-Self-Test (BIST) and scan design are predominantly used. Jaehoon Song and Piljae Min (2007) proposed an efficient design technique, named the AMBA based Test Access Mechanism (ATAM) utilizing the TIC, EBI, and Test Harness with minor area overhead, is introduced to minimize test application time by supporting simultaneous scan shifting-in and out while preserving complete compatibility with AMBA protocols [10]. A conventional AMBA system is comprised of Advanced High-performance Bus (AHB) and Advanced Peripheral Bus (APB). AHB interfaces high-speed cores such as a microprocessor, and slow devices are linked to the APB. A bridge must be adopted to interface AHB and APB with different speed and bus widths.

In the System-on-a-Chip (SoC) era, it is a challenge to verify and debug system chip efficiently and rapidly. For design verification and debugging at system level and chip level, not only external I/O signals observation, but also internal signals tracing can help designer to efficiently analyze and verify the design such as the software program, hardware protocol, and system performance. The bus tracing is used to catch related signals for further investigation and analysis [3]. However, the trace size of cycle accurate tracing is large and the trace cycle is shallow unless using a proper compression mechanism. SoC bus signal tracing can be accomplished with either software or hardware approaches. The software approach relies on using an instruction set simulator running on a host machine to simulate the software program behavior in the target processor. Although being simple and easy to adopt, the software approach has the disadvantage that simulation, running on a host machine instead of in the actual physical environment, often fails to generate and capture the real time behaviors. Due to the relatively slow simulation speed, it is impractical to collect larger execution traces with the simulator-based approach. Furthermore, most of the software approaches trace only program address, and the cost of hardware implementation of software approaches would be high. To address the bus tracing issue, an embedded multi-resolution signal tracing for AMBA AHB was proposed in [3]. The bus tracer consists of two major tracing approaches: (1) signal monitor/tracing approach, and (2) trace reduction approach. In the first approach, it provides the trade-off between the trace accuracy and the trace depth. In the second approach, the bus tracer compresses the trace data according to AHB signal characteristics such as address, data, and control signals [3]. Bus signal tracing can help designer to debug and analysis system design during implementation stage and final chip testing. Unlike the traditional cycle-based trace, the multi resolution trace approach is proposed which supports both cycle and transaction-based trace mechanisms, and provides the levels of abstraction for easy debugging. Furthermore, the different types of trace result can be stored in a single trace file. For trace reduction issue, AMBA address, data, and control signals trace can be compressed according to their own characteristics. It shows that the proposed approach can reach a good compression ratio and the trace depth is more than two thousand cycles at the higher abstraction level.

### III. MEMORY CONTROLLERS

#### 3.1. Microprocessor Systems

A microprocessor incorporates the functions of a computer's central processing unit (CPU) on a single integrated circuit (IC), or at most a few integrated circuits. It is a multipurpose, programmable device that accepts digital data as input, processes it according to instructions stored in its memory, and provides results as output. It is an example of sequential digital
logic, as it has internal memory [9]. Microprocessors operate on numbers and symbols represented in the binary numeral system. The advent of low-cost computers on integrated circuits has transformed modern society. General-purpose microprocessors in personal computers are used for computation, text editing, multimedia display, and communication over the Internet. Many more microprocessors are part of embedded systems, providing digital control over myriad objects from appliances to automobiles to cellular phones and industrial process control. A microprocessor control program (embedded software) can be easily tailored to different needs of a product line, allowing upgrades in performance with minimal redesign of the product. Different features can be implemented in different models of a product line at negligible production cost. Microprocessor control of a system can provide control strategies that would be impractical to implement using electromechanical controls or purpose-built electronic controls [12]. Microprocessors are the devices in a computer which make things happen. Microprocessors are capable of performing basic arithmetic operations, moving data from place to place, and making basic decisions based on the quantity of certain values. Most of the microprocessors provide a latency time of 33 ns for the operations.

3.2. Memory Controller Systems

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 integrated into another chip, such as on the die of a microprocessor. This is also called a Memory Chip Controller (MCC). A memory controller is integrated on the microprocessor in order to reduce memory latency. While this has the potential to increase the system's performance, it locks the microprocessor to a specific type (or types) of memory, forcing a redesign in order to support newer memory technologies. Memory controllers contain the logic necessary to read and write to DRAM, and to "refresh" the DRAM by sending current through the entire device. Without constant refreshes, DRAM will lose the data written to it as the capacitors leak their charge within a fraction of a second [7]. Reading and writing to DRAM is performed by selecting the row and column data addresses of the DRAM as the inputs to the demultiplexer circuit, where the demultiplexer on the DRAM uses the converted inputs to select the correct memory location and return the data, which is then passed back through a multiplexer to consolidate the data in order to reduce the required bus width for the operation. A general memory controller consists of front end and back end parts. The front end part includes buffers requests and responses [16]. It provides an interface to the rest of the system and is independent of the memory type. The back end part provides an interface towards the target memory and is dependent on the memory type. Double Data Rate 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. DDR SDRAM memory controllers have a delay of 12 clock cycles for the operations. SDR SDRAM memory controllers provide high performance access logic with read and write queuing. This also supports SDRAM self refresh option and power down modes [17]. SDR SDRAM memory controllers ensure a latency time of 14 ns. Both these types of memory controllers provide high system performance than microprocessors. But still these memory controllers have got some disadvantages like the microprocessors which can hinder the system efficiency and performance. Main disadvantage is the lack of burst transfer support. Burst transfer (alternatively burst-mode) is a generic computing term referring to any situation in which a device is transmitting data repeatedly without waiting for input from another device or waiting for an internal process to terminate before continuing the transfer of data [2]. 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. No such asynchronous clocking in the present systems. There is no multiple memory support. Also there is no access to external memory devices in the present microprocessor system and there is lack of multiple memory support. So there is can considerably reduce the system performance. The system performance can be increased further by building memory controllers using advanced microcontroller bus architecture (AMBA) by removing all these disadvantages.
IV. AHB MC

An Advanced Microcontroller Bus Architecture (AMBA) compliant memory controller is designed here 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 it is called as “AHB-MC”. 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. The AHB-MC mainly consists of three modules [1]. They are AHB slave interface, configuration interface, and external memory interface. The AHB-MC has several features. It is designed with synthesizable HDL for Application Specific Integrated Circuit (ASIC) synthesis. Another feature is that it supports multiple memory devices including static random access memory (SRAM), read-only memory (ROM). AHB-MC complies with AMBA AHB protocol. Another main feature is that this memory controller supports one to four memory banks for SRAM and ROM.AHB-MC consists of programmable memory timing register and configuration registers. AHB-MC provides shared data path between memory devices to reduce pin count.

![Figure 1. Architecture of AHB-MC](image)

AHB slave interface converts the incoming AHB transfers to the form used internally by the AHB-MC. The command FIFO issues commands to external memory interface. The write data FIFO and read data FIFO are controlling the read and write operations. The external memory interface controls the cycle timings of the commands. The control logic will decode the information and given to the memories. The configuration interface controls the command FIFO. The configuration interface consists of configuration registers. These registers contain all parameters that are required for the controller to set the correct access timings. Finally the read operations occur in ROM and read or write operations occur in RAM with required access timings.

4. 1. AHB Slave Interface

The AHB slave interface converts the incoming AHB transfers to the protocol used internally by the AHB-MC [2]. The logic for this interface is developed. The state machine is shown in figure 2. With this logic the code can be developed.
4. 2. External Memory Interface

The external memory issues commands to the memory from the command FIFO, and controls the cycle timings of these commands. The wait states are avoided in this interface and the memory access time can be reduced considerably. Following figures show the timing of a read from memory and a write to memory with two wait states [13].

There are two wait states in case of memory read and also in memory write. The external memory interface avoids this in order to reduce memory access time and to increase the system performance.
4. 2. 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

<table>
<thead>
<tr>
<th>Select</th>
<th>Remap</th>
<th>HADDR</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>XX</td>
<td>1111</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>00</td>
<td>0111</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>01</td>
<td>1101</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>10</td>
<td>1011</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>11</td>
<td>0111</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>00</td>
<td>1110</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>01</td>
<td>1101</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>1011</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0111</td>
</tr>
</tbody>
</table>

4. 2. 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.
The main function of the configuration interface is to change the configuration registers (SETCYCLE and SETOPMODE register) 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 require [2]. Each memory chip supported by AHB-MC has two registers (CYCLE register and OPMODE register), which contain all the timing parameters that are required for the controller to set correct access timings. Which cycle and operation mode registers are updated is determined by the configuration registers like SETCYCLE and SETOPMODE. The configuration interface converts its inputs to FIFO order to be used by the external memory interface. The configuration registers will change the inputs to the FIFO order to be used by the memory controller.

### 4. 3. Configuration Interface

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 5.

---

**Table 2. XWEN Coding**

<table>
<thead>
<tr>
<th>HSIZE[1:0]</th>
<th>HADDR[1:0]</th>
<th>XWEN[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 (word)</td>
<td>XX</td>
<td>0000</td>
</tr>
<tr>
<td>01 (half)</td>
<td>0X</td>
<td>1100</td>
</tr>
<tr>
<td>01 (half)</td>
<td>1X</td>
<td>0011</td>
</tr>
<tr>
<td>00 (byte)</td>
<td>00</td>
<td>1110</td>
</tr>
<tr>
<td>00 (byte)</td>
<td>01</td>
<td>1101</td>
</tr>
<tr>
<td>00 (byte)</td>
<td>10</td>
<td>1011</td>
</tr>
<tr>
<td>00 (byte)</td>
<td>11</td>
<td>0111</td>
</tr>
</tbody>
</table>

---

**Figure 5. System Memory Architecture**
4. 5. Asynchronous Clock

The AHB-MC has two clock domains: AHB clock domain and external memory clock domain as shown in figure 6. Asynchronous FIFO is used between two clock domains as a data buffer.

Figure 6. 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 [5]. Additionally, in sleep-mode situations when the system is not required to do much work, you can lower the frequency to reduce power consumption.

V. RESULTS AND DISCUSSION

The simulation is done using ModelSim. The input data is given and three operations can take place in here. One is writing to RAM, second is reading from RAM and third is reading from ROM. When the ‘cmd’ signal is 00 then the operation taking place is writing to RAM. When cmd is 01, the operation is read from RAM and when cmd is 10, the operation is read from ROM. Here the operations are taking place with a 4 clock cycles delay which is the required output. In other memory controllers without AMBA architecture have a delay of 12 clock cycles. The synthesis is done using Xilinx ISE 9.1. According to the synthesis report the minimum input arrival time before clock is 10.274ns. Maximum output required time after clock is 9.572 ns. Latency time for AMBA-MC is 6.23 ns. Latency time of the microprocessor systems is 33 ns and the latency time for memory controllers without AMBA architecture is 14 ns. Compared to the microprocessor systems the efficiency of AMBA-MC is increased by 81% and compared to other memory controllers the efficiency is increased by 58%.
VI. CONCLUSION

The development of an Advanced Microcontroller Bus Architecture (AMBA) compliant MC as an Advanced High-performance Bus (AHB) slave was presented. The Memory Controller, designed for system memory control, was built to increase the system performance by reducing memory access time which generated 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 simulation results show that the delay for the read and write operations is 4 clock cycles which is less than microprocessors and other memory controllers. Also the output time is obtained as 9.527 ns and the latency time is 6.23 ns. With the reduced delay, the system speed is increased. So the system efficiency is increased by 81%. Thus the system performance is improved to a greater rate.

VII. FUTURE SCOPE

Burst transfer support can be introduced in the AHB-MC to accomplish the address decoding and memory access operations in one clock cycle without reducing the system performance. In this method, all AHB fixed length burst types can be 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. Four, eight and sixteen-beat bursts are defined in the AMBA AHB protocol, as well as undefined-length bursts and single transfers. Both incrementing and wrapping bursts are supported in the protocol. Incrementing bursts access sequential locations and the address of each transfer in the burst is just an increment of the previous address. For wrapping bursts, if the start address of the transfer is not aligned to the total number of bytes in the burst (size x beats) then the address of the transfers in the burst will wrap when the boundary is reached.

REFERENCES

[1]. “AMBA Specification (Rev2.0)”, ARM Inc.


AUTHORS

Nithin Joe working as Assistant Professor in the Department of Electronics and Communication Engineering at Nehru College of Engineering and Research Centre, Pampady, Thriviravilawamala, Kerala, India, affiliated to University of Calicut. He obtained the Bachelor of Science Degree in Electronics with Computer Hardware from Presentation College of Applied Sciences affiliated to Mahatma Gandhi University, Kottayam. He received Master of Science degree in Applied Electronics, Nehru Arts and Science College affiliated to Bharathiar University Coimbatore, India and Master of Technology in Sensor Systems Technology from VIT University, Vellore, Tamil Nadu.

Anjali Brite is doing M.Tech in VLSI Design at Nehru College of Engineering & Research Centre, Thrissur, Kerala under the University of Calicut. She received B.Tech degree from University of Calicut in 2011.