An Overview of Advance Microcontroller Bus Architecture Relate on AHB Bridge

K. Vamsi Krishna1, K. Amarendra Prasad 2

1Research Scholar, Dept of ECE, Shree Institute of Technical Education, Tirupathi, AP, India, Email: krishnavamsi.636@gmail.com.  
2Asst Prof, Dept of ECE, Shree Institute of Technical Education, Tirupathi, AP, India, Email: amarendraprasad464@gmail.com.

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: Advanced Microcontroller Bus Architecture (AMBA), AMBA High-Performance Bus (AHB), System-On-Chip, Field Programmable Gate Array16t (FPGA), 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 off-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.

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 Microcontroller 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. It is developed, tested, and licensed by ARM Limited. Supports multiple memory devices of different types

- SynthesizableVerilog/VHDL RTL source
- Adheres to the AMBA AHB protocol
- The AHB - MC has several features: like SRAM, ROM, and Flash Shares data path between memory devices to reduce
- Provides test bench and verification vectors
- Supports AHB single beat, 4 beat and 8 beat wrapping
- Pincount bursts and split transaction.

II. ARCHITECTURE OF AMBA BASED SIMPLEMICROCONTROLLER

An AMBA-based microcontroller typically consists of a high-performance system backbone bus (AMBA AHB or AMBA ASB), able to sustain the external memory bandwidth, on which the CPU, on-chip memory and other Direct Memory Access (DMA) devices reside. This bus provides a high-bandwidth interface between the elements that are involved in the majority of transfers. Fig1 shows AMBA based Simple Microcontroller. Also located on the high performance bus is a bridge to the lower bandwidth APB, where most of the peripheral devices in the system are
STRUCTURE AS A SECONDARY BUS FROM GENERATION OF AMBA BUS WHICH IS -

The AMBA specification has become a de-facto standard for the semiconductor industry, it has been adopted by more than 95% of ARM’s partners and a number of IP providers. The specification has been successfully implemented in several ASIC designs. Since the AMBA interface is processor and technology independent, it enhances the reusability of peripheral and system components across a wide range of applications. The AMBA specification has been derived to satisfy the following four key requirements:

- To facilitate the right-first-time development of Embedded Microcontroller Products with one or more CPUs or signal processors.
- To be technology-independent and ensure that highly reusable peripheral and system macro cells can be migrated across a diverse range of IC processes and be appropriate for full custom, standard cell and gate array technologies.
- To encourage modular system design to improve processor independence, providing a development roadmap for advanced cached CPU cores and the development of peripheral libraries.
- (iv) To minimize the silicon infrastructure required supporting efficient on-chip and off-chip communication for both operation and manufacturing test.

**Fig. 1. AMBA based Simple Microcontroller.**

**III. DIFFERENT AMBABUSES**

The Advanced Microcontroller Bus Architecture (AMBA) is ARM’s no-cost, open specification, which defines an on-chip communications standard for designing high performance Embedded Microcontrollers. Three distinct buses are defined within the AMBA specification:

A. The Advanced System Bus (ASB)
B. The Advanced Peripheral Bus (APB).
C. The Advanced High-performance Bus (AHB)

**A. The Advanced System Bus (ASB)**

ASB is the first generation of AMBA system bus. A typical AMBA ASB system may contain one or more bus masters. For example, at least the processor and test interface. However, it would also be common for a Direct Memory Access (DMA) or Digital Signal Processor (DSP) to be included as bus masters. The external memory interface, APB bridge and any internal memory are the most common ASB slaves. Any other peripheral in the system could also be included as an ASB slave. However, low-bandwidth peripherals typically reside on the APB.

**B. The Advanced Peripheral Bus (APB)**

The APB is part of the AMBA hierarchy of buses and is optimized for minimal power consumption and reduced interface complexity. The AMBA APB appears as a local secondary bus that is encapsulated as a single AHB or ASB slave device. APB provides a low-power extension to the system bus which builds on AHB or ASB signals directly. The APB bridge appears as a slave module which handles the bus handshake and control signal retiming on behalf of the local peripheral bus. By defining the APB interface from the starting point of the system bus, the benefits of the system diagnostics and test methodology can be exploited. The AMBA APB should be used to interface to any peripherals which are low bandwidth and do not require the high performance of a pipelined bus interface. The latest revision of the APBs is specified so that all signal transitions are only related to the rising edge of the clock. This improvement ensures the APB peripherals can be integrated easily into any design flow. These changes to the APB also make it simpler to interface it to the new AHB. An AMBA APB implementation typically contains a single APB bridge which is required to convert AHB or ASB transfers into a suitable format for the slave devices on the APB. The bridge provides latching of all address, data and control signals, as well as providing a second level of decoding to generate slave select signals for the APB peripherals.

**C. The Advanced High-performance Bus (AHB)**

AHB is a new generation of AMBA bus which is intended to address the requirements of high-performance synthesizable designs. It is a high-performance system bus that supports multiple bus masters and provides high-bandwidth operation. Bridging between this higher level of bus and the current ASB/APB can be done efficiently to ensure that any existing designs can be easily integrated. An AMBA AHB design may contain one or more bus masters; typically a system would contain at least the processor and test interface. However, it would also be common for a Direct Memory Access (DMA) or Digital Signal Processor (DSP) to be included as bus masters. The external memory interface, APB bridge and any internal memory are the most common AHB slaves. Any other peripheral in the system could also be included as an AHB slave.

**IV. ARCHITECTURE OF AMBA-AHB**

The AHB is a new generation AMBA bus which is intended to address the requirements of high-performance synthesizable designs. It is a system bus that supports multiple bus masters and provides high bandwidth operation.
The AHB implements features required for high clock frequency systems including: i) burst transfers; ii) split transactions; iii) single-cycle bus master handover; iv) single-clock edge operations; v) non-tristate implementations; and vi) wider data bus configurations (64/128 bits).

A. Bus interconnection

The AHB bus protocol is designed to be used with a central multiplexer (MUX) interconnection scheme. Using this scheme, all bus masters drive out the address and control signals, indicating the transfer they wish to perform, while the arbiter determines which master has its address and control signals routed to all of the slaves. A central decoder is also required to control the read data and response MUX, which selects the appropriate signals from the slave that is involved in the transfer.

B. Basic transfer

The AHB data transfer consists of two distinct phases: (i) address phase, which lasts only one single cycle; (ii) data phase, which may require several cycles. This is realized by using the Hready signal. The simplest transfer, one with no wait states. The top-level implementation of an AHB compliant MC is shown in Figure. The data are initiated by the master and communicated through the slave to the MC. Initially, the master generates the data and control signals; those controls cannot directly communicate with any given generic memory and hence, data are processed through the slave; further, data pass through the slave interface. We have used FIFO for data and control buffering so that even the slave and memory are in different clocks, ensuring that our communication is guaranteed. It reduces the complexity. Further, data are transmitted through the random-access memory(RAM) or read only memory (ROM), depending upon the read or write communication.

C. AMBA high performance bus (AHB) master

An AHB master has the most complex bus interface in an AMBA system. It contains mainly the address generator and controller. Initially, the first address is given by the CPU; then, it passes to the MUX. Depending on the selection, it sends its data to the address generation block. The number of addresses generated is given by a counter which decides how many addresses are required depending on the HBurst signal. There is a controller implemented by FSM which takes care of the entire operation of the different blocks. Figure 8 presents the structural implementation of the address generator block, which is a combination of 2:1 MUX and 8:1 MUX. In this architecture, the initial address is given by the CPU and next addresses are generated by the various blocks such as an increment operator (INCR).

D. AMBA high-performance bus (AHB) slave

An AHB slave responds to transfers initiated by the bus masters within the system. The slave uses a select control signal (HSELx) from the decoder to determine when it should respond to a bus transfer. All other signals required for the transfer, viz. the address and control information will be generated by the bus master. The state diagram of a slave interface. It is an FSM implementation; the initial condition is the reset state that is an idle state when no operation is involved. As the start signal arrives, the FSM triggers. Depending on the HRead and HWrite signals, it decides upon to which state it has to move. If HReady is low, the system will stay in the starting state; if HReady is high, based on HWrite, it goes to read or write state. If HWrite is high, the state passes on to the write state; otherwise, if HRead is high, the system enters the read state. If HReady is low, the system transfers to the wait state; it remains in this state until HReady becomes high again. If any error occurs, the system enters the error state.

E. Memory controller (MC):

The state diagram of the MC. It is also an FSM implementation; the initial condition is reset state which is an idle state when no operation is there. When start signals arrive, the FSM triggers; depending upon the instruction, its operation is decided on by command control state. Based on the instruction, it moves to RAM read, RAM write and ROM read operations. Two important tasks for the MC are to handle all the scheduling among the different commands and to keep track of which rows are presently activated. The activation of rows is time consuming and therefore, the MC has a look ahead functionality where the arbitrator can report which command is going to be executed after the current one has been completed. The arbitrator makes it possible to activate the row in advance if the next command is not retrieving the same bank or chip as the current command.

![Fig.2. Top architecture of AHB memory controller.](image)

F. First-in first-out (FIFO)

The FIFO is a method of procession and retrieving data in a FIFO system, the first items entered are the first ones to be removed. In other words, the items are removed in the order they are entered. A FIFO buffer consisting of two modules (called source and sink) connected to one another. When data are being passed from module to module, the source is the module that is outputting data. The sink is the module that is receiving that data. Three signals are shown between the two modules A and B: data, FIFO full and FIFO empty. The line marked data represents the wire that actually passes the data
from the source to sink. The FIFO full and FIFO empty are known as handshaking signals, which allow the source and sink to communicate when it is time to pass the data. The FIFO full signal indicates that the FIFO is full and puts valid data on the data line. FIFO full is what is called a state signal – it is high only when data are valid. If data are not valid on the data line during a particular cycle, valid should be low during that cycle.

**V. SIMULATION RESULTS**

![Block diagram](image1.png)

**Fig 3.** Top architecture of AHB bus master.

![Address generator](image2.png)

**Fig 4.** Address generator of master.

![First in first out (FIFO)](image3.png)

**Fig 5.** First in first out (FIFO).
VI. CONCLUSION

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.

VII. REFERENCES