Moving from AMBA AHB to AXI Bus in SoC Designs: A Comparative Study

Priyanka Gandhani[#1], Charu Patel*[2]

[#] *Electronics & Communication Department,Nirma University,AHMEDABAD, INDIA*
[*] *Electronics & Communication Department, CHARUSAT, CHANGA, INDIA*
1 priyanka.3787@gmail.com
2 patelcharu1988@yahoo.co.in

***Abstract*—The ever increasing amount of logic that can be placed onto a single silicon die is driving the development of highly integrated SoC designs. Such high computing power must be matched by interconnect fabrics with adequate bandwidth and efficiency. Traditional SoC interconnects, as exemplified by AMBA AHB, are based upon low-complexity shared buses, in an attempt to minimize area overhead. Such architectures, however, are not adequate to support the trend for SoC integration, motivating the need for more scalable designs. This paper describes the most important AMBA bus architectures and how they evolved to accommodate to the ever increasing complexity of SoC technology.**

***Index Terms*— AMBA, AHB, AXI**

1. Introduction

Embedded system designers have a choice of using a share or point-to-point bus in their designs. Typically, an embedded design will have a general purpose processor, cache, SDRAM, DMA port, and Bridge port to a slower I/O bus, such as the Advanced Micro controller Bus Architecture (AMBA) Advanced Peripheral Bus (APB). In addition, there might be a port to a DSP processor, or hardware accelerator, common with the increased use of video in many applications. As chip-level device geometries become smaller and smaller, more and more functionality can be added without the concomitant increase in power and cost per die as seen in prior generations.

The Advanced Microcontroller Bus Architecture (AMBA) was introduced by ARM Ltd 1996 and is widely used as the on-chip bus in system on chip (SoC) designs. AMBA is a registered trademark of ARM Ltd. The first AMBA buses were Advanced System Bus (ASB) and Advanced Peripheral Bus (APB). In its 2nd version, AMBA 2, ARM added AMBA High-performance Bus (AHB) that is a single clock-edge protocol. In 2003, ARM introduced the 3rd generation, AMBA 3, including AXI to reach even higher performance interconnect and the Advanced Trace Bus (ATB) as part of the Core Sight on-chip debug and trace solution. These protocols are today the de-facto standard for 32-bit embedded processors because they are well documented and can be used without royalties.

The paper has been organized as follows. The first section contain the description of AHB protocol both single layer and multilayer bus. Second section describes the 3rd generation of AMBA bus that is AXI. Third section shows the migration from AHB to AXI based on different parameters.

2. Advanced High Performance Bus (AHB)

2.1. Single layer AHB

AHB supports single data access and various types of burst accesses. Each transfer is defined by an address and a data phase where the address phase of one transfer occurs during the data phase of the previous transfer. Underlying AHB is traditional bus architecture with arbitration between multiple masters. The protocol supports advanced features such as SPLIT and RETRY signaling in cases where a slave is not able to respond immediately. The master that had been granted the bus will back off and other masters will get a turn.

2.2. Multilayer AHB

Although traditional multiplexed multi-master systems are still quite common, little over a decade ago the ARM SoC world started shifting towards crossbar switched interconnects, in the form of multi-layer busses. This was a rather important initial step which lead over time to some critical improvements. Each layer of the bus is an independent single master AHB system. Instead of a rather complex monolithic multiplexing scheme, a multi-layer AHB bus architecture with M masters and S slaves is structured as M X 1:S multiplexers plus S X M:1 slave multiplexers all connected to separate arbitration and decoding logic.

Multiple masters can talk to multiple slaves concurrently, as long as no two masters don't try to access the same slave at the same time. Think of a DMA controller moving data from a receiver into a memory region, while the processor continues to execute code in a different memory region. All arbitration and protocol complexity moves into the fabric. The interface implementation becomes simpler as a number of unneeded signals, most notably HGRANT and HBUSREQ, can be removed along with their associated protocol. Although not a necessary

consequence of the multi-layer architecture, getting rid of the unpopular SPLIT and RETRY handshaking mechanism was another advantage.

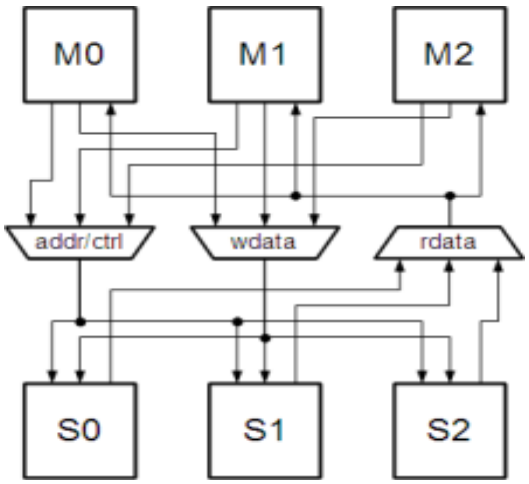

Figure 1. Multiplexed Bus

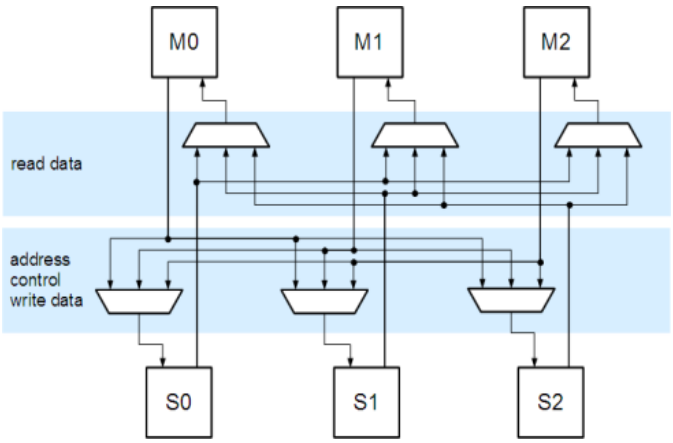

Figure 2 Multilayer Bus

3. Advanced Extensible Bus (AXI)

3.1. AXI3

The AMBA 3 AXI protocol is targeted at high-performance, high-frequency system designs and includes a number of features that make it suitable for a high-speed, submicron interconnect. The AMBA 3 AXI protocol objectives: The AMBA 3 AXI specification was created with the following objectives in mind to ensure its suitability for the next generation of designs.

- Suitability for high-bandwidth and low-latency designs
- To enable high-frequency operation without using complex bridges
- Meet the interface requirements of a wide range of components
- Suitability for memory controllers with high initial access latency
- Provide flexibility in the implementation of interconnect architectures
- Easily interface with existing AMBA technology

Features of the AMBA 3 AXI protocol include:

- Separate address/control and data phases
- Support for unaligned data transfers using byte strobes
- Burst-based transactions with only start address issued
- Separate read and write data channels to enable low-cost direct memory access (DMA)
- Ability to issue multiple outstanding addresses
- Out-of-order transaction completion
- Easy addition of register stages to provide timing closure
- Protocol includes optional extensions that cover signaling for low-power operation

Advantages of the AMBA 3 AXI protocol include:

- Independently acknowledged address and data channels
- Out-of-order completion of bursts
- Exclusive access (atomic transaction)
- System level cache support
- Access security support
- Unaligned address & byte strobe
- Static burst, which allows bursts to FIFO memory
- Low power mode

The AMBA 3 AXI architecture differs significantly from previous AMBA protocols with the introduction of channels. Each of the five independent channels consists of a set of information signals and uses a mechanism. The information source uses the VALID signal to show when valid data or control information is available on the channel. The destination uses the READY signal to show when it can accept the data. Both the read data channel and the write data channel also include a LAST signal to indicate when the transfer of the final data item within a transaction takes place. Read and write transactions each have their own address channel. The appropriate address channel carries all of the required address and control information for a transaction.

The Read data channel conveys both the read data and any read response information from the slave back to the master. The Read data channel includes the data bus, which can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide and a read response indicating the completion status of the read transaction. The Write data channel conveys the write data from the master to the slave. The Write data channel includes the data bus, which can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide, and one byte lane strobe for every eight data bits, which indicates which bytes of the data bus are valid. The unaligned transfer support makes for a more efficient use of the bus yielding higher performance, lower latency and increased bandwidth operation.

3.2. AXI4 and AXI4-Lite

AXI4 is the latest revision of the AXI protocol described above. Functionality has been added and several known issues in AXI3 have been addressed to ensure that AMBA busses remain the dominant standard in SoC connectivity. Some key points: The maximum burst length has been increased from 16 to 256 transfers for certain types of bursts (INCR, non-

exclusive). Additional Quality-of-Service signaling has been added, where the finer details of the interpretation are implementation defined.

AXI4 defines address regions for slaves, which allows implementations of memory perspectives on the bus level. No doubt this will be used at some point in the future to break the 4GB address boundary. Some ordering requirements and transfer dependencies have been refined, as have the meanings of the cache policy signals AxCACHE. Abstract memory types as defined by ARMv6/v7 architectures and multicore architectures are much better represented by these changes.

Implementation-defined per-channel sideband signals are now officially supported as AxUSER. Legacy (AHB) locked transfers are no longer supported. The entire concept that a master can request exclusive access to the entire bus doesn't fit within the idea of a switched interconnect. The one-and-only ARM instruction causing this signal to be asserted is no longer supported in the v7 architectures.

A rather significant change seems to be the banning of write interleaving, which could help improve the system throughput. In practice, removing write interleaving from this part of the AMBA standard makes certain aspects of the AXI protocol easier to handle. Write interleaving is hardly used by regular masters but can be used by fabrics that gather streams from different sources. With the new AXI4-Stream protocol (see below), write interleaving is still available for fabrics. As described so far the focus of AXI has been on high-performance data transfer, but what about the low-end - hardware registers, configuration, etc? With good old APB there is an established, robust interface, which received an upgrade in AMBA3 extending it with slave response signaling (PERROR, PREADY), a feature that was missed dearly by designers.

The issue with APB is the bridge. In a traditional system, including AMBA3, one or more of the slaves are bridges between the main system protocol (AHB, AXI) and APB. The intention was that with many small peripherals on a "real" bus including the multi-layer variant, the fan-out of multidrop signals (HWDATA, HADDR in AHB) would be too high. A typical bridge supports up to sixteen slaves, which are assigned fragments of the address region occupied by the bridge itself so that all APB peripherals connected to this bridge are in one contiguous address region.

In modern interconnects, you may find built-in 1:1 bridges which connect between system bus and a single APB slave, enabling higher flexibility. Still a bridge though. AXI4-Lite addresses this last issue by defining certain restrictions that would allow a slave to be connected directly to an AXI fabric. In AXI4-Lite, you might say that AXI gets "dumbed" down to a few basic transaction types. The burst length is fixed to one data transfer, transfers are non-cacheable and non-bufferable, exclusive access is not allowed and access width must always be the same as data bus width. This is supposed to make the interface design simple enough to be implemented quickly in custom IP.

3.3. AX14 Stream

The new AXI4-Stream protocol was designed for streaming data to destinations that are not memory mapped internally. Display controllers, transmitters, but also routing fabrics are among the target applications for this new protocol. Building upon the proven simple AXI channel handshake AXI4-Stream is essentially an AXI write data channel with additional control signals and a slightly modified protocol. The burst (packet) length is not restricted and the number of bytes of the data signals TDATA can be an arbitrary integer including zero.

4. Migration From AHB To AXI

With modern Systems on Chip including multi-core clusters, additional DSP, graphics controllers and other sophisticated peripherals, the system fabric poses a critical performance bottleneck. The AHB protocol, even in its multi-layer configuration cannot keep up with the demands of today's SoC. The reasons for this include:

1. AHB is transfer-oriented. With each transfer, an address will be submitted and a single data item will be written to or read from the selected slave. All transfers will be initiated by the master. If the slave cannot respond immediately to a transfer request the master will be stalled. Each master can have only one outstanding transaction.

2. Sequential accesses (bursts) consist of consecutive transfers which indicate their relationship by asserting HTRANS/HBURST accordingly.

3. Although AHB systems are multiplexed and thus have independent read and write data buses, they cannot operate in full-duplex mode.

An AXI interface consists of up to five channels which can operate largely independently of each other. Each channel uses the same trivial handshaking between source and destination (master or slave, depending on channel direction), which simplifies the interface design.

Unlike AHB concept is not an afterthought but is the central focus of the protocol design. In AXI3 all transactions are bursts of lengths between 1 and 16. The addition of byte enable signals for the data bus supports unaligned memory accesses and store merging.

The communication between master and slave is transaction-oriented, where each transaction consists of address, data, and response transfers on their corresponding channels. Apart from rather liberal ordering rules there is no strict protocol-enforced timing relation between individual phases of a transaction. Instead every transfer identifies itself as part of a specific transaction by its transaction ID tag. Transactions may complete out-of-order and transfers belonging to different transactions may be interleaved. Thanks to the ID that every transfer carries, out-of-order transactions can be sorted out at the destination.

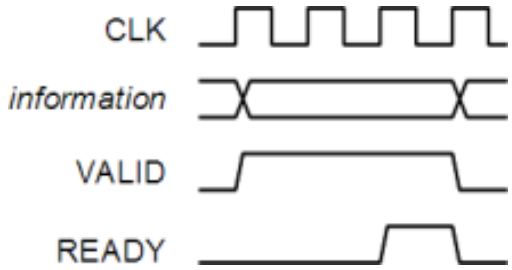

Figure 3.AXI channel handshake

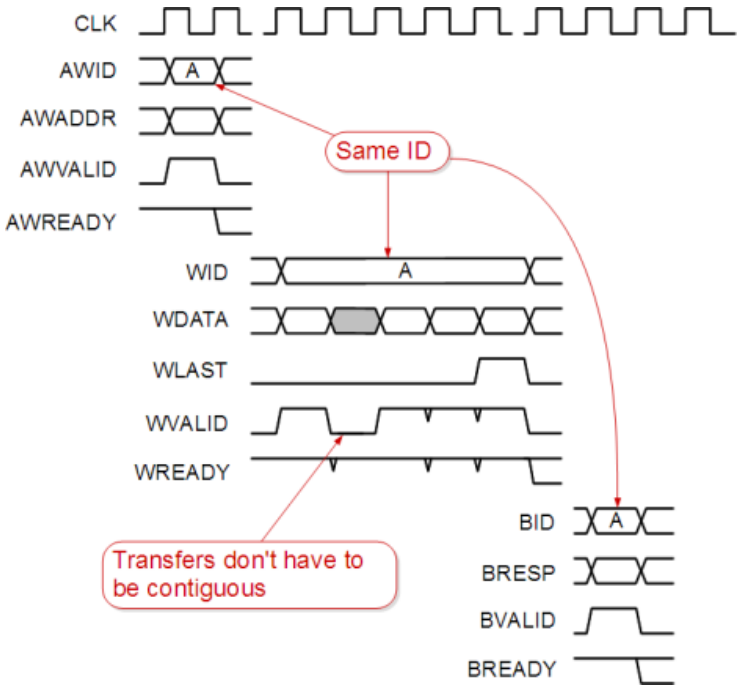

Figure 4. AXI write burst

This flexibility requires all components in an AXI system to agree on certain parameters, such as write acceptance capability, read data reordering depth and many others. Due to the vast number of signals that make up a read/write AXI connection, routing a large AXI fabric could be thought of as rather challenging. However, the independent channels in an AXI fabric make it possible to choose a different routing structure depending on the expected data volume on that channel. Given a situation where the majority of transactions will transfer more than one data item, data channels should be routed via crossbar so that different streams can be processed at the same time. Address and response channels experience rather lower traffic and could perhaps be multiplexed. Some experts consider it an advantage to provide AXI only at the interface level, while a special packetized routing protocol is used inside the fabric, a so called Network-on-Chip.

The AHB is a single-channel, shared bus. The AXI is a multi-channel, read/write optimized bus. Each bus master, or requesting bus port, connects to the single-channel shared bus in the AHB, while each AXI bus master connects to a Read address channel, Read data channel, Write address channel, Write data channel, and Write response channel. The primary throughput channels for the AXI are the Read/Write data channels, while the address, response channels are to improve pipelining of multiple requests. Assume there are four masters on each bus going to three slaves. The four master ports might include microprocessor, Direct Memory Access (DMA), DSP, USB. The three slaves might include on-chip RAM, off-chip SDRAM, and an APB bus bridge.

To approximate the bandwidth of the two busses, one must count the number of read/write channels of the AXI Bus – six for three bus slaves. This suggests that the AHB Bus should support some multiple of bus width and/or speed to match the data throughput. The System Model can vary these combinations with simple parameter changes, however, the AHB bus speed was assumed to be double the AXI Bus, and two times the width. This will make the comparison of the two busses more realistic.

To evaluate the efficiency of both busses, different burst sizes were selected; small, medium, and large. Small equates to the width of the AHB Bus, medium equates to two AHB Bus transfers, and large equates to four AHB bus transfers.

If the AXI is a 64 bit bus running at 200 MHz, then the AHB will be a 128 bit bus running at 400 MHz. The burst sizes will be: small (16 Bytes), medium (32 Bytes), and large (64 Bytes).

5. Conclusion

Over the years AMBA has continued to provide state-of-the-art solutions for SoC interconnects. With the relatively recent addition of the AXI4 protocol family ARM maintains a competitive advantage in the field of high-performance SoC, while at the same time AHB-Lite is still available for less demanding architectures.

References

[1] AMBA specification ,version 2.0
[2] AMBA AXI protocol specification ,version 2.0
[3] Deepak Shankar. "*Comparing AMBA AHB To AXI Bus Using System Modeling*", February 2010
[4] Marcus Harnisch "*Migrating From AHB To AXI Based SOC Design*", 2010