Ijesm
Consulting, help, relaxation
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & MANAGEMENT

TRENDS AND TRADE-OFFS IN DESIGNING AND PERFORMANCE EVALUATION OF DIFFERENT ON-CHIP AMBA BUS

Anurag Shrivastava[1]*, Amit Kant Pandit[1], Vipan Kakkar[1] and GS Tomar[2]
1, Department of SECE, SMVDU, Katra, (J&K) - India
2, MIR Lab, Gwalior, (M.P.) - India

Abstract
Moore's law predicts that soon it will be possible to integrate billions of transistors on a single chip. Currently on-chip communication for multiprocessor system-on-chip (MPSoC) is realized using buses such as AMBA protocol family provides metric-driven verification of protocol compliance, enabling comprehensive testing of interface intellectual property (IP) blocks and system-on-chip (SoC**)** designs**.** Modern computer system rely more and more on –chip communication protocol to exchange data. System-on-chip (SoC) designs use bus protocols for high performance data transfer among the Intellectual Property (IP) cores. These protocols incorporate advanced features such as pipelining, burst and split transfers. In this paper, we describe a case study for different AMBA SOC bus protocol and their performance comparison: the Advanced Micro-controller Bus Architecture (AMBA) protocol from ARM. It starts with a brief introduction AMBA 2.0 protocol, AMBA 3.0 protocol and AMBA 4.0 protocol ,and concludes with a discussion related to a comparison performance analysis of all three AMBA bus protocol.

Keywords: SOC, On-chip Communication, AMBA bus Protocol

Introduction

On-chip communication architectures can have a great influence on the speed and area of System-on-Chip Designs. Modern computer system[14] rely more and more on highly complex on–chip communication protocol to exchange data. The enormous complexity of these protocol results from tackling high-performance requirements. Protocol control can be distributed, and there may be non-atomicity or speculation. The electronics industry has entered the era of multi-million-gate chips, and thereXs no turning back. This technology promises new levels of integration on a single chip, called the System-on-a-Chip (SoC) design, but also presents significant challenges to the chip designer. Processing cores on a single chip, may number well into the high tens within the next decade, given the current rate of advancements, [1]. The important aspect of a SOC is not only which components or blocks it houses, but also how they are interconnected. AMBA is a solution for the blocks to interface with each other.

*** Corresponding Author**
E-Mail: shrivastavaanurag@rediffmail.com

The objective of the AMBA specification [15] is to:

- facilitate *right-first-time* development of embedded microcontroller products with one or more CPUs, GPUs or signal processors,
- be technology independent, to allow reuse of IP cores, peripheral and system macro cells across diverse IC processes,
- encourage modular system design to improve processor independence, and the development of reusable peripheral and system IP libraries
- Minimize silicon infrastructure while supporting high performance and low power on-chip communication.

The **Advanced Microcontroller Bus Architecture (AMBA)** was introduced by ARM in 1996 and is widely used as the on-chip bus in SoC designs. AMBA is a registered trademark of ARM. The first AMBA buses were Advanced System Bus (ASB) and Advanced Peripheral Bus (APB). In its 2nd version, AMBA 2, ARM [2] added AMBA High-performance Bus (AHB) that is a single clock-edge protocol. In 2003, ARM introduced the 3rd generation, AMBA 3[3], including AXI to reach even higher performance interconnects and the Advanced Trace Bus (ATB) as part of the Core Sight on-chip debugs and trace solution. In 2010, ARM introduced the 4th generation, AMBA 4,[1] including *AMBA 4 AXI4, AXI4-Lite, and AXI4-Stream Protocol,* The AMBA 4.0 protocol defines five buses/interfaces:

- Advanced extensible Interface (AXI)
- Advanced High-performance Bus (AHB)
- Advanced System Bus (ASB)
- Advanced Peripheral Bus (APB)
- Advanced Trace Bus (ATB)

The third generation of AMBA defined in the AMBA 3 protocol specification [7], is targeted at high performance, high clock frequency system designs which make it very suitable for high speed sub-micrometer interconnect.

A simple transaction on the AHB consists of an address phase and a subsequent data phase (without wait states: only two bus-cycles). Access to the target device is controlled through a Mux (non-tristate), thereby admitting bus-access to one bus-master at a time. AMBA 4.0 protocol [12] includes definition of an expanded family interconnect protocols including AXI4, AXI4-Lite and AXI4-Stream.

2. AMBA 2.0 Protocol

The Advanced Microcontroller Bus Architecture (AMBA) specification, developed by ARM [2], defines an on-chip communications [2] standard for designing high performance embedded microcontrollers.[15] Three distinct buses are defined within the AMBA specification:

Advanced High-performance Bus (AHB) - The AMBA AHB is for high-performance, high-clock-frequency system modules.

Advanced System Bus (ASB) - The AMBA ASB is for high-performance system modules. Where the high-performance features of AHB are not required.

Advanced Peripheral Bus (APB)-AMBA APB is optimized for minimal power consumption and reduced interface complexity to support peripheral functions. APB can be used in conjunction with either version of the system bus.

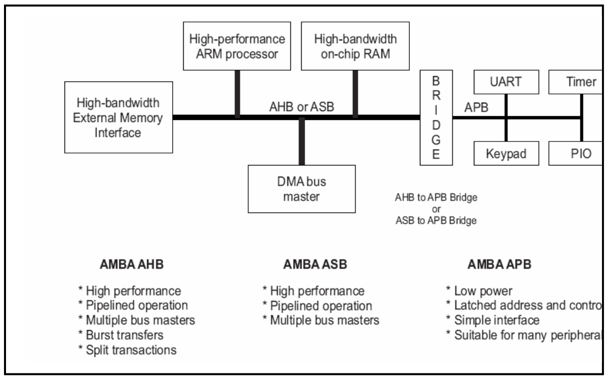

Figure1: AMBA 2.0 Protocol

3. AMBA 3.0 Protocol

3. AMBA 3.0 Protocol: The key goals of the AMBA 3 AXI protocol is interoperability [15] with the existing AMBA technology. 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.

AMBA AXI is a fully PP connection.[4] It decouples masters and slaves from the underlying interconnect, by defining only master interfaces and symmetric slave interfaces. This approach, besides allowing backward compatibility and interconnect topology independence, has the advantage of simplifying the handshake logic of attached devices, which only need to manage a point-to- point link.

Figure 2a shows how a read transaction uses the read address and read data channels [4]. The write operations over the write address and write data channels are presented in Fig 2b. Data is transferred from the master to the slave using a write data channel, and it is transferred from the slave to the master using a read data channel. In write transactions, where all the data flows from the master to the slave,The AXI protocol[5] has an additional write response channel to allow the slave to signal to the master the completion of the write transaction. The rationale of this split-channel implementation is based upon the observation that usually the required bandwidth for addresses is much lower than that for data (e.g. a burst requires a single address but maybe four or eight data transfers). Thus, it might be possible to allocate more interconnect bandwidth to the data bus than the address bus.[9]

Features of the AMBA 3 AXI protocol:

- 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

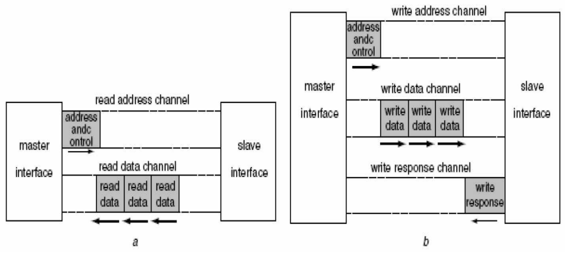

Figure2: (a) Read (b) Write

- System level cache support
- 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

4.AMBA 4.0 Protocol

AMBA AXI4 [3] supports data transfers up to 256 beats and unaligned data transfers using byte strobes. In AMBA AXI4 system 16 masters and 16 slaves are interfaced. Each master and slave has their own 4 bit ID tags. AMBA AXI4 system consists of master, slave and bus (arbiters and decoders). The system consists of five channels namely writeaddress channel, write data channel, read data channel, read address channel, and write response channel. The AXI4 protocol supports the following mechanisms:

- Unaligned data transfers and up-dated write response requirements.
- Variable-length bursts, from 1 to 16 data transfers per burst.
- A burst with a transfer size of 8, 16, 32, 64, 128, 256, 512 or 1024 bits wide is supported.
- Updated AWCACHE and ARCACHE signaling details.

Each transaction is burst-based which has addressed and control information on the address channel that describes the nature of the data to be transferred. The data is transferred between master and slave using a write data channel to the slave or a read data channel to the master. Table 1[3] gives the information of signals used in the complete design of the protocol. The write operation process starts when the master sends an address and control information on the write address channel as shown in fig.3. The master then sends each item of
write data over the write data channel. The master keeps the
VALID signal low until the write data is available. The master sends the last data item, the WLAST signal goes HIGH. When the slave has accepted all the data items, it drives a write response signal BRESP[1:0] back to the master to indicate that the write transaction is complete. This signal indicates the status of the write transaction. The allowable responses are OKAY, EXOKAY, SLVERR, and ECERR. After the read address appears on the address bus, the data transfer occurs on the read data channel as shown in fig. 4. The slave keeps the VALID signal LOW until the read data is available. For the final data transfer of the burst, the slave asserts the RLAST signal to show that the last data item is being transferred. The RRESP[1:0] signal indicates the status of the read transfer. The allowable responses are OKAY.

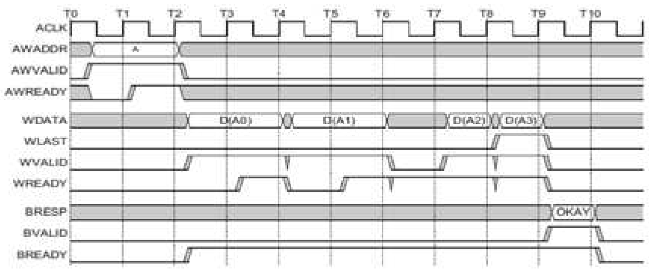

Figure 3: Write address and data burst.

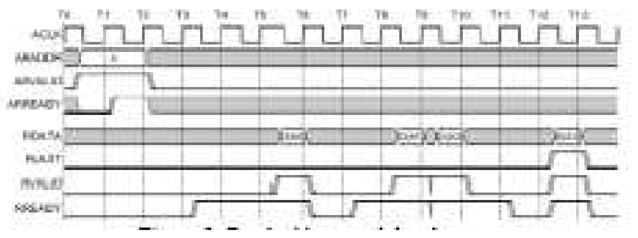

Figure 4: Read address and data burst.

The protocol supports 16 outstanding transactions, so each read and write transactions have ARID[3:0] and AWID [3:0] tags respectively. Once the read and write operation gets completed the module produces a RID[3:0] and BID[3:0] tags. If both the ID tags match, it indicates that the module has responded to right operation of ID tags. ID tags are needed for any operation because for each transaction concatenated input values are passed to module.

The AMBA AXI4 system components consists of
1) Master
2) AMBA AXI4 Interconnect
    2.1) Arbiters
    2.2) Decoders

3) Slave
The master is connected to the interconnect using a slave interface and the slave is connected to the interconnect using a master interface as shown in fig. 3. The AXI4 master gets connected to the AXI4 slave interface port of the interconnect and the AXI slave gets connected to the AXI4 Master interface port of the interconnect. The parallel capability of this interconnects enables master M1 to access one slave at the same as master M0 is accessing the other.

A. AMBA AXI4 Master

To perform write address and data operation the transaction is initiated with concatenated input of [awaddr, awid, awcache, awlock, awprot, awburst]. On the same lines for read address and data operations the concatenated input is [araddr, arid, arcache, arlock, arprot, arburst]. The addresses of read and write operations are validated by VALID signals and sent to interface unit.

***B.* AMBA AXI4 Interconnect**

The interconnect block consists of arbiter and decoder. When two masters initiate a transaction simultaneously, the arbiter gives priority to access the bus. The decoder decodes the address sent by master and the control goes to one slave out of 16. The AMBA AXI interface decoder is centralized digital block. The decoder decodes the address sent by master and goes to one slave out of 16. 0-150 locations are meant for slave-1, next 151-300 addressable locations are meant for slave-2,… and so on till slave-16.

***C.* AMBA AXI4 Slave Read/Write Block Diagram**

The AXI4 slave consists of common read/ write buffer which stores the read/ write address and data as shown in fig. 4. Pending read address register stores the remaining read addresses to be sent; pending write address register which stores the remaining write addresses to be sent and pending write data register which stores the remaining write data to be sent. The read/write state machines receive internal inputs from the read/ write buffer. The AXI4 slave test bench initiates the read or write transaction and the output from the AXI4 slave

are standard read/write channel signals. The AXI4 slave

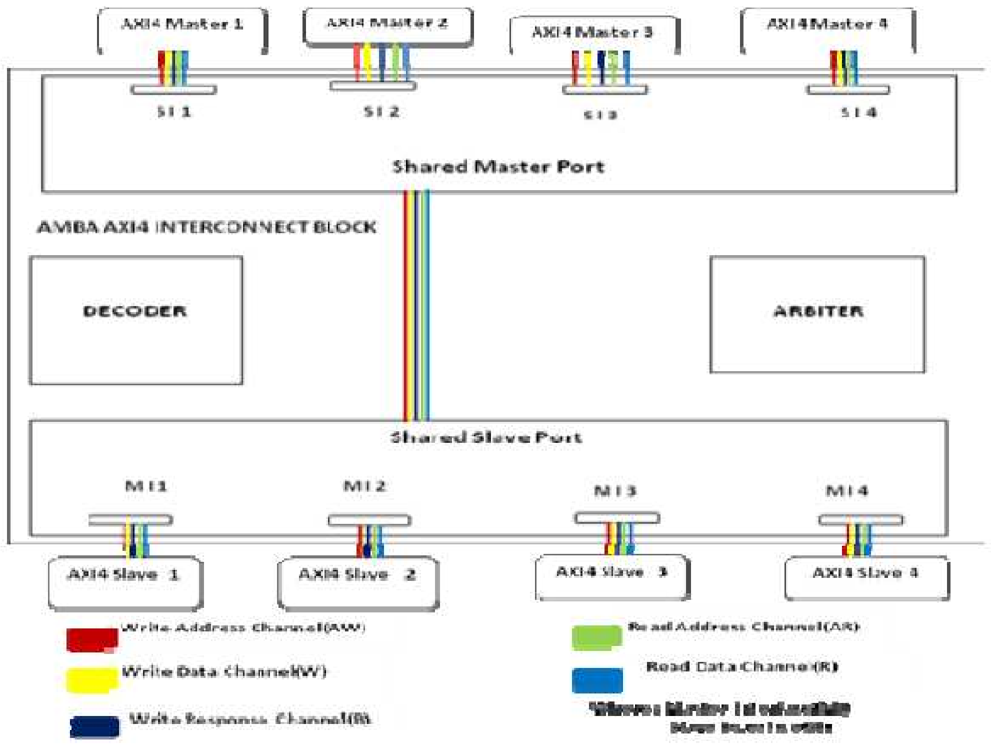

Figure 5: Block diagram of AMBA AXI4 bus interconnect.

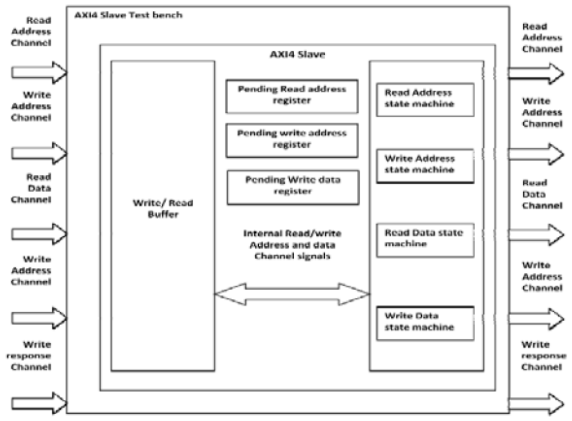

Figure 6: AMBA AXI4 slave Read/Write block Diagram

Features

The AXI4 update to AXI3 includes the following:

- support for burst lengths up to 256 beats
- *Quality of Service* (QoS) signaling
- support for multiple region interfaces
- updated write response requirements
- updated **AWCACHE** and **ARCACHE** signaling details
- additional information on Ordering requirements
- details of optional User signaling
- removal of locked transactions
- removal of write interleaving

5. Performance comparison of AMBA Protocol

Table 1: Comparison of AMBA Protocol Type

	APB	AHB	AXI3/AXI4
Processors	all	ARM7,9,10	ARM11
Control Signals	4	27	77
No. of Masters	1	1-15	1-16
No. of Slaves	1-15	1-15	1-16
Interconnect Type	Central MUX	Central MUX	Crossbar w/ 5 channels
Phases	Setup, Enable	Bus request, Address, Data	Address, Data,Responce
Xact. Depth	1	2	16

5.1 Features Comparison:

AMBA 2 AHB Protocol	**AMBA 3 AXI Protocol**
Fixed pipeline for address and data transfers	Five independent channels for addr/data and response
Bidirectional link with complex timing relationships	Each channel is unidirectional except for single handshake for return path
Hard to isolate timing	Register slices isolate timing
Limits frequency of operation	Frequency scales with pipelining
Inefficient asynchronous bridges	High performance asynch bridging
Separate address for every data item	Burst based--one address per burst
Only one transaction at a time	Multiple outstanding transactions
Fixed pipeline for address and data	Out of order data
Only one transaction at a time	Simultaneous reads and writes

Burst Lengths	1	1-32	1-16
Simultaneous Read & Write	no	no	Yes

5.2 Dependencies between Channel Handshake Signals

Dependencies To prevent a deadlock situation[12], the dependencies that exist between the handshake signals. In any transaction: The VALID signal of one AXI component must not be dependent on the READY signal of the other component in the transaction.The READY signal can wait for assertion of the VALID signal The AXI3 protocol requires that the write response for all transactions must not be given until the clock cycle after the acceptance of the last data transfer In addition, the AXI4 protocol requires that the write response for all transactions must not be given until the clock cycle after address acceptance.

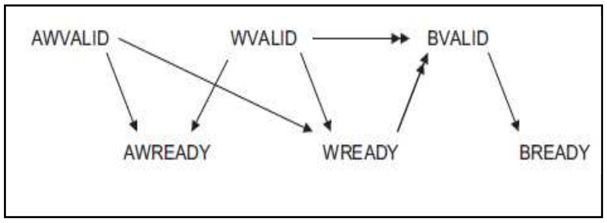

Figure7: AXI3

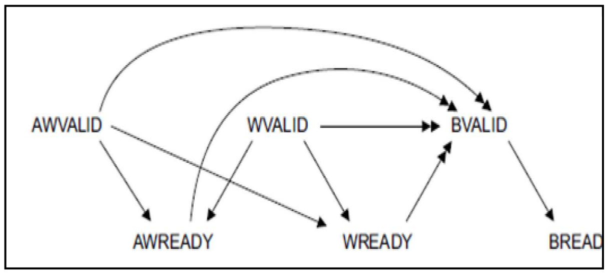

Figure 8: AXI4

5.3 Comparison of AMBA AXI3 and AXI4

AMBA AXI3 protocol has separate address/control and data phases, but AXI4 has updated write response requirements and updated AWCACHE and ARCACHE signaling details. AMBA AXI4 protocol supports for burst lengths up to 256 beats and Quality of Service (QoS) signaling. AXI4 has additional information on Ordering requirements and details of optional user signaling. AXI3 has the ability to issue multiple outstanding addresses and out-of order transaction completion, but AXI4 has the ability of removal of locked transactions and write interleaving. One major up-dation seen in AXI4 is that, it includes information on the use of default signaling and discusses the interoperability of components which can't be seen in AXI3. In this paper features of AMBA AXI4 listed above are designed and verified.

Conclusions

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.
4.The AHB is a single-channel, shared bus. The AXI is a multi-channel, read/write optimized bus.
5. AXI4 address width -32 bits with 16 master and 16 slave device
6.Write interleaving. This feature was retracted by AXI4 protocol. AXI3 master devices must be configured as if connected to a slave with Write interleaving depth of one.
7. The AMBA AXI4 has limitations with respect to the burst data and beats of information to be transferred.
8.The AMBA 4 protocol new features such as 256-beat burst support for improved performance, Quality of Service (QoS) signaling for multi-master support, the AXI4-Stream protocol for non-address-based, point-to-point communication among masters and slaves (especially useful for data streams commonly used for moving video and audio on an SoC), multiple region interfaces.
9. The AXI4 protocol supports an ordering model based on the use of the AXI ID transaction
10.The AXI3 protocol requires that the write response for all transactions must not be given until the clock cycle after the acceptance of the last data transfer. In addition, the AXI4 protocol requires that the write response for all transactions must not be given until the clock cycle after address acceptance
11.Ttransactionwrite interleaving support in AXI4.
12.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).
13.AXI3 protocol: Early termination of bursts it not supported.A burst must not cross a 4-kbyte boundary.AXI4 protocol longer burst support(16 beats).

References

1. ARM, “AMBA Specification Overview”, available at *http://www.arm.com/*.
2. ARM, “AMBA Specification (Rev 2.0)”, available at *http://www.arm.com*.
3. ARM, “AMBA AXI Protocol Specification”, available at *http://www.arm.com*.
4. Benini and D. Bertozzi, “Network-on-chip architectures and design methods,” IEE Proc.-Comput. Digit. Tech., Vol. 152, No. 2, March 2005
5. Chien-Hung Chen, Jiun-Cheng Ju, and Ing-Jer Huang, “A Synthesizable AXI Protocol Checker for SoC Integration,”IEEE,,ISOCC 2010.
6. ng. Dries Driessens and. Tom Tierens, “Overview Embedded Buses,” 2003 by Patrick Pelgrims.
7. *Mick Posner, Darrin Mossor* ,”Designing Using the AMBA (TM) 3 AXI (TM) Protocol -- Easing the Design Challenges and Putting the Verification Task on a Fast Track to Success” *from Synopsys.*
8. Denali Memory Blog *Steve Leibson* ,”New White Paper discusses the challenges of chip design based on AMBA 4”From 2010, Cadence. [9]Marcus Harnisch , ”Migrating from AHB to AXI based SoC Designs”,from Doulos, 2010.
9. Krishna Sekar, Kanishka Lahiri,, Anand Raghunathan ,”Dynamically Configurable Bus Topologies for High-Performance On-Chip Communication”, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 16, NO. 10, OCTOBER 2008.
10. Sangkwon Na*, Sung Yang**, and Chong-Min Kyung,”Low-Power Bus Architecture Composition for AMBA AXI ,JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE, VOL.9,2009.
11. AMBA® 4 AXI4™, AXI4-Lite™, andAXI4-Stream™ Protocol Assertions User Guide 2010 ARM.
12. NY Liao,”Analysis of shared-link AXI” IET Comput. Digit. Tech., 2009, Vol. 3, Iss. 4, pp. 373–383 .
13. Bohm,Tom Melham,”Design and verification of on-chip communicatiom protocol”,proceedings of designing correct circuits 2008
14. Advanced Microcontroller Bus Architecture free encyclopedia Roychoudhury,Tulika,”Using formal techniques to debug the AMABA SOC Bus Protocol”,Design,automation and test Europe conference,2003,IEEE.