# Review on Design and Verification of an Advanced Extensible Interface - 4 Slave Devices

**Harish T.L[1] & Chandrashekhar M.C[2]**
*[1 & 2]Department of Electronics and Communication Engineering,*
*Sri Siddhartha Institute of Technology, Tumkur, India.*
***DOI**: https://doi.org/10.34293/acsjse.v3i2.80*

**Abstract -** *An essential part of a system on a chip (SoC) is not just the components or blocks that it contains, but also how these components and blocks are connected to one another. Advanced Microcontroller Bus Architecture (AMBA) is a system that enables the individual blocks to interact with one another. These protocols have established the de facto standard for 32-bit embedded processors because they are widely documented and may be used without the payment of royalties. High-performance and high-frequency system designs may be designed with the aid of the AMBA AXI 4 protocol. It's ideal for systems that need both large bandwidth and low latency since it enables high-frequency operation without the use of complicated bridges. In addition to being compatible with older versions of the AHB and APB interfaces, it also allows for a wide range of possible configurations for the network's interconnects. Without the need for a bridge, many peripherals may be connected into AMBA-based CPUs via the use of this slave interface. By constructing a wrapper over the AXI4 slave interface, the newly created slave interface may also be used to link non-AMBA-based CPUs to a variety of peripherals. These peripherals include SPI, I2C, and UART, amongst others.*
***Keywords:** Advanced Microcontroller Bus Architecture (AMBA), Advanced Peripheral Bus (APB), AMBA High performance Bus (AHB), Advanced Extensible Interface (AXI).*

## I INTRODUCTION

The firm ARM (Advanced RISC Machines) is responsible for the development of the AMBA, of which the Advanced Extensible Interface (AXI) is a component. It is a communication protocol that takes place on the chip itself. By providing support for the design of high-performance and high-frequency system designs, the AMBA AXI protocol benefits the field. If your system requires both large bandwidth and low latency, the AXI protocol is a good choice. It allows for high-frequency functioning without the need for a complex bridge. It is compatible with the interface specifications of a diverse selection of elements. The AXI protocol offers a degree of flexibility in the manner in which interconnect designs are implemented. It maintains compatibility with the already-established AHB and APB interfaces. A distinct address/control phase and data phase are two of the most important aspects of the AXI protocol. Moreover, the AXI protocol provides support for unaligned data transfers by making use of byte strobes. It processes transactions in bursts, and the sole valid address is the first address that is issued. Direct Memory Access (DMA) is achieved at a reduced cost thanks to its read and write data channels that are kept independent. It allows for the issuance of many unique addresses simultaneously. It allows for the completion of transactions in a non-ordered manner. It simplifies and streamlines the process of adding register stages to allow for timing closure.

## II LITERATURE SURVEY

The AMBA, is a standard for managing the communication between different parts of a system-on-a-chip. This specification is an open standard that is implemented as an on-chip interface. As a result, it simplifies the expansion of multi-processor systems with many controllers and exteriors. AMBA's purview, despite the fact that it was originally intended for use with microcontrollers, has expanded significantly since the organization's foundation. The application processors utilised in today's smart phones and other portable electronic devices are only one example of the many ASIC and SoC components that make use of the AMBA. In 1996, ARM introduced AMBA to the public for the first time. The Advanced System Bus (ASB) and the Advanced Peripheral Bus (APB) were the first two AMBA buses created [1&2]. The AMBA High-performance Bus (AHB) is a single clock-edge protocol introduced with ARM's second version of AMBA, known as AMBA 2 [3-5]. The AMBA 3 was launched by ARM in 2003. It included AXI for enhanced performance connection and the Advanced Tracing Bus (ATB) [6&7]. AMBA 3 was the third iteration of the AMBA. In 2010, the AMBA 4 standards were first made available, beginning with AMBA 4 AXI4. To alleviate traffic, AMBA 4 ACE was released in 2011 with system-wide coherency extension and a rethought high-speed transport layer [8-10]. Metric-driven verification of protocol compliance is made feasible by the AMBA protocol family, allowing for comprehensive testing of interface IP blocks and SoC designs. In comparison to its predecessor, the AMBA AXI4 has the following enhancements: Information on component compatibility, burst lengths up to 256 beats, improved write response requirements, and the removal of locked transactions are just a few of AXI4's new features. AMBA AXI4 is a protocol system that allows interfacing between 16 masters and 16 slaves. Verilog-HDL is used to create an actualization of the design [11-13].

Connecting and managing functional blocks in system-on-a-chip (SoC) designs is the focus of the AMBA. This specification is an open standard that is implemented as an on-chip interface. Hence, it facilitates the creation of multi-processor systems that use several controllers and peripherals. AMBA's purview, despite the fact that it was originally intended for use with microcontrollers, has expanded significantly since the organization's foundation. The application processors utilised in today's smart phones and other portable electronic devices are only one example of the many ASIC and SoC components that make use of the AMBA. In 1996, ARM introduced AMBA to the public for the first time. The first two AMBA buses to be developed were the Advanced System Bus (ASB) and the Advanced Peripheral Bus (APB) [1&2]. The AMBA High-performance Bus (AHB) is a single clock-edge protocol that debuted in ARM's second edition of AMBA, dubbed AMBA 2. [3-5] The Advanced eXtensible Interconnect (AXI) and the Advanced Tracing Bus (ATB) are components of ARM's Core Sight on-chip debug and trace solution, which was unveiled in 2003 [6&7]. The most recent release of the AMBA is version 3. In 2010, the AMBA 4 standards were first made available, beginning with AMBA 4 AXI4. The following year, in 2011, AMBA 4 ACE was released, which extended system wide coherency and included a redesigned high-speed transport layer in addition to features meant to alleviate congestion [8-10]. The AMBA protocol family offers metric-driven protocol compliance verification,

which enables thorough testing of interface IP blocks and system-on-chip (SoC) designs. The AMBA advanced extensible interface 4 (AXI4) adds many improvements over its predecessor, the AMBA AXI3: New features in AXI4 include support for burst durations up to 256 beats, revised write response requirements, and the elimination of locked transactions. Interfacing between 16 masters and 16 slaves is supported by the AMBA AXI4 protocol system. Verilog-HDL is used to create an actualization of the design [11-13].

The AMBA AXI protocol was developed with the intention of being used in high-frequency system designs. It's well-suited for usage as a super-fast submicron link due to a number of features. A function that is capable of supporting up to 256 data transfers in a single data burst has been suggested for inclusion in this project [3]. A total of 16 masters and 16 slaves are connected via the AMBA AXI4 system. In this society, each master and slave is assigned a unique four-bit number. A master, a slave, and a connecting bus make up the system [4]. Among other mechanisms, the following ones are supported by the AXI4 protocol:

- There are two different varieties of address mode: aligned and unaligned.
- There are three different kinds of bursts: fixed, incremental, and wrap.
- A selection of sixteen different burst lengths, with values ranging from 1 to 256.
- Four distinct forms of responses available, including OKAY, EXOKAY, SLVERR, and DECERR.

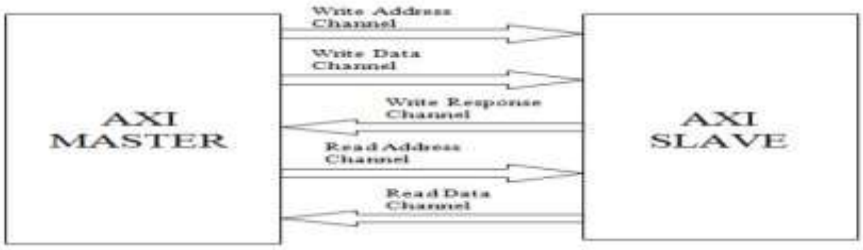


Figure 1 System Methodology Every transmission in the AXI protocol is completed via a process called the hand shake. When it comes to transferring control and data information, every channel employs the same VALID/READY handshake. The master and the slave may both adjust the data and control information transfer rates using this bidirectional flow control system. The VALID signal is sent by the source when either the data or the control information may be accessed. If the destination is all set to receive the data or control information, it will send out a READY signal. The VALID and READY signals must both be high in order for the transfer operation to take place. No combinatorial paths should exist between the input signals and the output signals at either the master or the slave interface.

### 2.1.1 Address Write Channel (AW Channel)

Only when ARESETn is HIGH does AXI MASTER drive the write command signals; in all other circumstances, it drives all signals as zero. The AXI MASTER is responsible for driving the following address write command signals: To validate the driving signals, set

AWVALID to HIGH and use AWID, AWADDR, AWBURST, AWLEN, AWSIZE, AWCACHE, AWLOCK, and AWPROT. The AWPROT signal is a safeguard. AXI MASTER will not set AWVALID to LOW until it has received the AWREADY signal from the DESTINATION SLAVE, which indicates that it has received the address write instruction signals. This is because it has received the address write instruction signals, as shown by the AWREADY signal. When AWREADY is set to LOW, AXI MASTER's settings will not change.

### 2.1.2 Write Data Channel (W Channel)

After providing the write address instruction signals, the AXI MASTER is the one responsible for driving these Write Data signals. Only when ARESETn is HIGH does it cause these signals to be driven; otherwise, it causes all signals to be driven to zero. The WDATA signal is driven by the AXI MASTER with the WVALID bit set to HIGH, and it maintains the same value until it receives the WREADY signal. When WREADY is HIGH, it causes the subsequent WDATA to be driven. The data for the AWLEN No. is driven by the AXI MASTER. During the process of driving the most recent data, it sets the WLAST to the HIGH position.

### 2.1.3 Write Response Channel (B Channel)

Only when ARESETn is HIGH does the DESTINATION SLAVE cause these Write Response signals to be driven; in all other cases, it causes all signals to be driven as zero. DESTINATION SLAVE is now holding out for the WLAST signal. It maintains a HIGH value of BVALID and drives the response signals in response to the WLAST signal. Awaiting the BREADY signal from the AXI MASTER, it will remain at its current value. Each signal will be reset to zero on the next positive edge of ACLK if BREADY is HIGH, but it will retain its current value if it is not HIGH.

### 2.1.4 Address Read Channel (AR Channel)

To be clear, AXI MASTER simply controls the signalling for commands. When ARESETn is HIGH, AXI MASTER controls the drive of the command signals; in all other circumstances, it drives all signals as zero. The AXI MASTER is responsible for driving the following address read command signals: The driven signals are valid when ARVALID is HIGH, which affects ARID, ARADDR, ARBURST, ARLEN, ARSIZE, ARCACHE, ARLOCK, and ARPROT. AXI MASTER will drive the ARVALID signal as LOW in response to address read instruction signals from SOURCE SLAVE. Given that it has received the address read instruction signals as shown by the ARREADY signal, this is the case. In the event that ARREADY is LOW, the settings of AXI MASTER will remain unchanged.

### 2.1.5 Read Data Channel (R Channel)

After receiving the read command signals, the SOURCE SLAVE is responsible for driving these Read Data signals. ARESETn only drives these signals when it is HIGH;

otherwise, it drives all of the signals with a value of zero. When the RDATA signal is driven by SOURCE SLAVE with RVALID set to HIGH, it maintains its previous value until it gets the RREADY signal. In the event that RREADY is HIGH, the succeeding RDATA value will be determined by that state. The data's ARLEN Value is determined by its SOURCE SLAVE. It drives the final data and puts the RLAST in the HIGH position.

**Verification Environment of AXI Protocol** System Verilog is used in the development of the AXI bus's verification environment. The hierarchical layering of this environment facilitates both its continued upkeep and its re-use with other designs now undergoing verification.

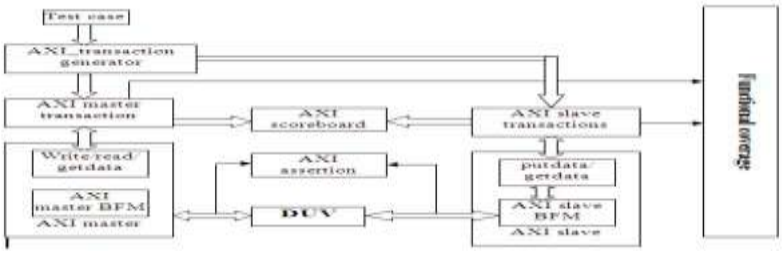


**Figure 2: The Testbench Architecture**

## III Conclusion

An example of a plug-and-play IP standard is the AMBA AXI4 protocol. ARM supplies this resource, which includes a bus specification and a technology-agnostic approach for creating and testing customised high-integrity embedded interfaces. To read or write data from or to a certain address point on a slave, it is assumed that the master will provide the necessary data. This article argues that an effective verification environment can perform simulations of the vast majority of conceivable AXI signal scenarios, analyse all of the data that is supplied automatically, and complete coverage analysis all while the simulation is operating. In order for the environment to enhance the coverage and significantly cut down the amount of time spent on verification. When it comes to the amount of burst and beats information that may be communicated, the AMBA AXI has certain restrictions. The data for the burst must stop short of the 4k barrier. The INCR burst type is the only one that supports bursts that are longer than sixteen beats. A limit of 16 beats per burst is still enforced for both the WRAP and FIXED kinds. These are the challenges that need to be conquered in order to use the AMBA AXI system.

## IV REFERENCES

[1] ARM. (1996). *AMBA Peripheral Bus Controller Data Sheet*. Advanced RISC Machines Ltd (ARM).

[2] Ramagundam, S., Das, S. R., Morton, S., Biswas, S. N., Groza, V., Assaf, M. H., & Petriu, E. M. (2014). Design and implementation of high-performance master/slave memory controller with microcontroller bus architecture. *IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*. https://doi.org/10.1109/i2mtc.2014.6860513.

[3] ARM. (1999). *AMBA™ Specification (Rev 2.0)*. ARM Limited.

[4] Hwang, S. Y., & Jhang, K. S. (2005). An improved implementation method of AHB BusMatrix. *IEEE International SOC Conference*. https://doi.org/10.1109/socc.2005.1554497

[5] YueLi, H., & Ben, Y. (2011). Building an AMBA AHB Compliant Memory Controller. *Third International Conference on Measuring Technology and Mechatronics Automation*. https://doi.org/10.1109/icmtma.2011.167.

[6] ARM (2003). *AMBA AXI Specification (AR500-DA-10008)*. ARM Limited.

[7] Paunikar, A., Gavankar, R., Umarikar, N., & Sivasankaran, K. (2014). Design and implementation of area efficient, low power AMBA-APB Bridge for SoC. *International Conference on Green Computing Communication and Electrical Engineering*. https://doi.org/10.1109/icgccee.2014.6922239

[8] ARM. (2011). *AMBA AXI 4 and ACE Protocol Specification*. ACE Protocol Specification.

[9] Yang, X., Qing-li, Z., Fang-fa, F., Ming-yan, Y., & Cheng, L. (2007). NISAR: An AXI compliant on-chip NI architecture offering transaction reordering processing. *International Conference on ASIC*. https://doi.org/10.1109/icasic.2007.4415774.

[10] Math, S. S., Manjula, R. B., Manvi, S. S., & Kaunds, P. (2011). Data transactions on system-on-chip bus using AXI4 protocol. *International Conference on Recent Advancements in Electrical, Electronics and Control Engineering*. https://doi.org/10.1109/iconraeece.2011.6129797

[11] Open Verilog International. (1996). *Verilog-A Language Reference Manual Analog Extensions to Verilog HDL, Version 1.0*.

[12] Accellera Systems Initiative. (2009). *Verilog-AMS Language Reference Manual*.

[13] Accellera Systems Initiative. (2014). *Verilog-AMS Language Reference Manual.*