ISSN: 2754-6659

Journal of Artificial Intelligence & Cloud Computing

SCIENTIFIC Research and Community

Review Article

Open Access

A Review of AMBA Bus Protocols: Focus on AHB, AHB-Lite, and AXI

Usha Mohani kavirayani[1]*, Krishna Bhardwaj Mylavarapu[2], Jenitha Pilli[3], Prathik Kumar Jannu[4], Javed Ali Mohammad[5] and Sri Harsha Panchali[6]

[1]Kent State University MS in Computer Science, USA

[2]MS in Computer Science University of Illinois Springfield, USA

[3]MS in Computer Science University of Louisiana at Lafayette, USA

[4]Computer Science Engineering JNTU Hyderabad, India

[5]Masters in Data Science New England College, USA

[6]Information Systems Engineer CrowdStrike Inc, USA

ABSTRACT

A common interface for connecting parts of a system-on-chip (SoC) design is provided by the Advanced Microcontroller Bus Architecture (AMBA). Focusing on the AXI, AHB-Lite, and AHB protocols, this page provides a summary of the AMBA bus protocols. When it comes to communicating between the CPU and high-bandwidth peripherals, the AHB is the way to go, but AHB-Lite is more user-friendly and uses less power. The AXI protocol is an extension of the AHB and includes parallel communication and is also part of the modern SoC architecture. This review discusses the characteristics, functions, and uses of these protocols, their contribution in enhancing system performance, energy efficiency and throughput. Another aspect that has been highlighted in the paper is the development of AMBA protocols in line with the growing complexity of mobile and multi-core processors, with a focus on the capability of supporting high-performance computing systems.

***Corresponding author**

Harsha Vardhan Reddy Kavuluri, Lead Database Administrator, Wissen infotech Inc, USA.

Received: November 27, 2025; **Accepted:** December 07, 2025; **Published:** December 10, 2025

Keywords: AMBA Architecture, Advanced High-Performance Bus (AHB), AHB-Lite, Advanced Extensible Interface (AXI), Soc Interconnect, On-Chip Communication, Bus Protocols

Introduction

The inclusion of buses into nearly all complex System on Chip (SOC) architectures is evident. Traditionally, busses have been meticulously designed to cater to a particular processor or a small subset of users [1]. There have been changes to the necessary buses as a result of several trends that have compelled changes to the designs of the systems. Modern microcontroller bus architecture is frequently used by communication systems to connect various components of a system-on-chip (SoC) design. The Advanced Microcontroller Bus Architecture (AMBA) protocol enables the integration of many intellectual property (IP) standards, including memory controllers, processors, and peripheral interfaces, into a system on a chip (SoC). The AMBA protocol depends on the low-power, low-bandwidth Advanced Peripheral Bus (APB) interface. Enhances bus connectivity for energy-saving devices [2]. Timer, observer, and interrupt controllers, among other simple devices, are commonly connected via the APB protocol because to its user-friendly architecture. Among the many uses for the APB protocol are learning about the steps involved in specifications, how long things take, and how the hardware plan is going to be put into action [3]. Making a product that satisfies the APB protocol's standards for low energy usage and great performance while also meeting time-based constraints is no easy feat.

- **TOP:** The top module is responsible for creating a clock, which is then passed as an argument to the interface. The connections for the DUT, interface, and test bench are then established.
- **TEST BENCH:** The whole process is being executed within the fork-join loop as construct the environment for the entire class handle.
- **BASE PACKET:** Two basic packets are being used here; one for all input ports and another for all output ports.
- **TRANS GENERATOR:** The stimulus is generated in the Trans generator, and the driver task is called to drive into it. Alternatively, Use the mailbox to talk to the driver and generator.
- **DRIVER:** The virtual interface transmits the stimuli to the

DUT, enabling us to execute the same subprogram in different designs and dynamically change the related signal collection. User input is not limited to physical signals but can instead affect a set of virtual signals [4].

- **MONITOR:** All of the DUT's output is being collected here via the virtual interface, and the monitor block is keeping tabs on it.
- **SCORE BOARD:** All the outputs are shown here on the scoreboard.
- **INTERFACE:** The interface is responsible for directing the signals sent to the DUT and test bench from the mod port and connecting them to the test bench.
- **DUT (design under test):** At DUT, putting real design into action.

There is a recommendation in this study to use embedded SOC platforms with excellent performance. An arbitrator and a generic bridge are proposed for the priority encoder-enhanced high-speed AHB protocol to boost performance and throughput. Using a Xilinx simulator, this work designs and tests an effective AMBA controller for data transaction procedures. The read/write procedures of the AMBA technique are illustrated with the aid of many samples [5]. This study proposes an alternative protocol for low-energy applications based on AHB Lite. It enables direct communication between slaves and masters connected to the system and peripheral domains [6]. Furthermore, the AMBA-Lite is capable of operating at high bandwidths. Things like burst transfer, non-tristate implementation, and operating on a single clock edge are all part of this. A single address decoder and a single multiplexor for slave selection make up the bus. The master initiates the transmission of control and address signals.

Structure of the Paper: The paper is organized as Section II explain the evolution of AMBA protocols, Section III present analysis of the AHB protocol in amba bus architecture, Section IV describes formal modelling of the AMBA AXI protocol, Section V comparative analysis of AHB, AHB-Lite, AND AXI Protocols. While section VI study of past literatures, Section VII conclude the study and future work.

Evolution of Amba Protocol

A microcontroller system based on the ARM architecture may include any of four AMBA protocols: APB, AHB, ASB, or AXI. The specific protocol used depends on the requirements of the system [7]. The AMBA protocol's primary function is to facilitate the exchange of data between various peripheral devices. The components of a SoC include RAM, a Direct Memory Access Device, and high bandwidth RAM. Understanding the specific applications, historical development, and integration of the AMBA protocols into the SOC architecture is the first step in delving deeper into their inner workings.

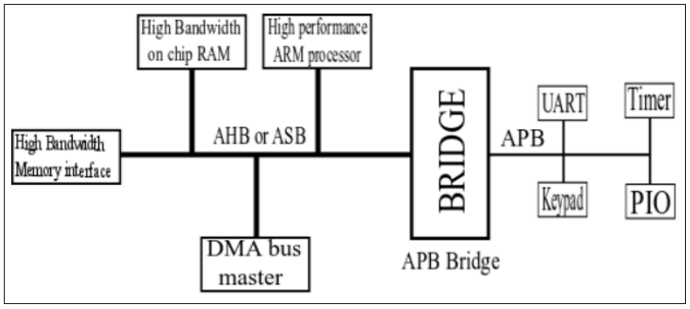

Figure 1: ARM AMBA Bus Architecture

The three parts that make up the AMBA bus architecture are the APB, the ASB, and the AHB Shown in Figure 1. Compared to standard buses, AMBA AHB or ASB offer more bandwidth and better performance. Components linked to the AHB or ASB that necessitate higher bandwidth include high-performance ARM CPUs, high-bandwidth memory interfaces, DMA bus masters, and high-bandwidth on-chip RAM Poor performance and bandwidth are well-known issues with the AMBA APB. Partially due to their lesser bandwidth, the APB is coupled with peripheral devices such as the UART, Keypad, Timer, and PIO (Peripheral Input Output) sensors [8]. Thanks to the bridge, the APB bus can link up with the high-performance AHB or ASB bus. Any additional devices linked to the APB bus must therefore submit to the bridge's authority when using APB. Sending and receiving data for all APB peripheral transactions is handled by the high-performance bus component. The high-performance bus can communicate with the peripheral devices through the bridge.

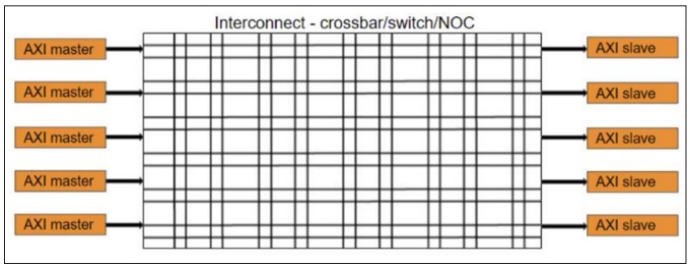

Figure 2: Illustrative Diagram of AXI Interconnect

The connection depicted in Figure 2 could be a unique crossbar, a prefabricated switch setup, or even a commercially available NOC (Chip Network) IP that can accommodate numerous AXI masters and slaves. By connecting to the AXI network, the number of agents can be increased in comparison to the AHB/ASB bus. Typically, in order to establish a connection to a group of peripherals that are shared by the APB bus, it is necessary to link one of the slave ports of the AXI port to the APB bus.

Mobile and smartphone technology necessitated the development of SOCs with multi-core, quad-core, or even octa-core CPUs, shared caches, and hardware-managed memory subsystem coherence, all of which prompted the AMBA protocols to undergo further evolution. This helped bring about the addition of the AXI Coherency Protocol Extension, or ACE, in the fourth version of the AMBA.

Characteristics AMBA Protocols

A Review of AMBA Bus Protocols: Focus on AHB, AHB-Lite, and AXI

- **Advanced Peripheral Bus (APB):** This bus typically only supports one master and has a complicated minimal interface; nevertheless, it can bridge connect to other buses, which means that there are several masters required, and these masters are usually from the same family [9]. There is no transaction pipeline and only a few bands are supported by this power-saving bus.
- **Advanced High-Performance Bus (AHB):** This is a multi-master bus that runs in tandem with broadband. With a large number of instructors (up to 16 buses), it enables transactions in bursts rather than preemptive or delayed ones. In order to achieve optimal performance, the data linked to an address is placed on the bus one cycle subsequent to the address, enabling pipeline operation. A data channel that can be expanded to 128 bits, known as multiplexing, is better than three-state bus lines.
- **Multi-Layer AHB:** Arm intends to circumvent the limitations

of extensibility and guarantee the standard AMBA's (and AHB's) continued existence. The centralized bus replaced by an AHB protocol-based connectivity network [10]. The interconnection matrix assigns a master block for decoding and a slave block for arbitration. Unfortunately, this expansion was mostly unsuccessful because it is still mono-clock, which necessitates global management—a problem that is already difficult for a centralized bus—on a distributed network.

• **AHB-Lite:** The APB protocol services for a single master were improved, but many designers still needed the AHB bus protocol's performance. Boarding the AHB bus Added features that weren't necessary for this scenario. Although it retains the complex transaction potential, the AHB-Lite protocol offers a restricted set of services for a single master: in burst, re-emissive, Various protocol standards and interfaces have been established to make it easier to integrate different components into the same architecture. The goal is to ensure that these protocols are used when creating components.

Analysis of the Ahb Protocol in Amba Bus Architecture

High-performance synthesizable designs are the target audience for AMBA AHB. This high-performance system bus operates at a high bandwidth and supports a large number of bus masters. The features essential to high-clock-frequency, high-performance systems are integrated into AMBA AHB.

- quick swaps
- Separate financial dealings
- transfer between multi-cycle buses
- action on a single clock edge
- absence of tristate execution
- expanded combinations of data buses (64/128 bits).

The present ASB/APB and this higher-level bus can be bridged efficiently to allow for easy integration of any existing designs. The central processing unit (CPU) and test interface are standard components of most systems, but an AMBA AHB architecture allows for the addition of bus masters. Bus masters can be several different types of common components, such as DSPs and DMA [11]. Typically, an APB bridge, any internal memory, the external memory interface, or any internal memory can all serve as AHB slaves also add any other system peripheral as an AHB slave. It is common for the APB to house low-bandwidth peripherals, though. Everything listed below is part of a standard AMBA AHB system design:

• **AHB master:** Once a bus master receives an address and control information, it can begin read/write operations. Only one master may be present on the bus at any one time.

• **AHB slave:** Any activity that falls inside a specific address space range trigger a response from a bus slave. When a data transmission succeeds, fails, or is waiting, the bus slave notifies the active master.

• **AHB arbiter:** A single bus master is required to initiate data transfers, and this is ensured by the bus arbiter. Regardless of the application's requirements, the static arbitration protocol can be used with any algorithm, including fair access and highest priority. In systems with only one bus master, this wouldn't matter, but in an AHB, there would still be just one arbiter [12].

• **AHB decoder:** Each transfer's address is deciphered by the AHB decoder, which then sends a select signal to the relevant slave. No AHB setup is complete without a central decoder.

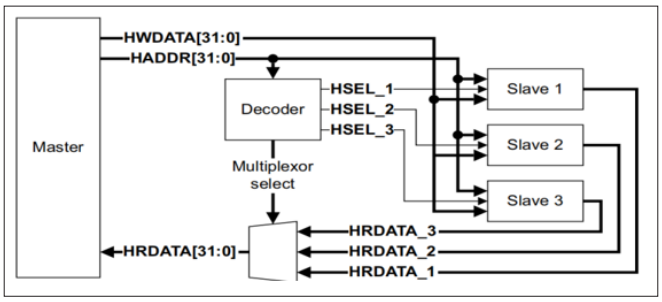

Figure 3: AHB Block Diagram

Three AHB slaves and one AHB master are depicted in Figure 3 of the system design. The logic that connects buses consists of an address decoder and a multiplexor that goes from master to slave. The decoder keeps track of the master's address, which then selects the appropriate slave. The data output from the slave is then transmitted back to the master by a multiplexor. Additionally, AHB is compatible with multitasker systems through its connection component, which allows for the transmission of arbitration and routing signals between several masters and their respective slaves.

A. AHB Master Interface:

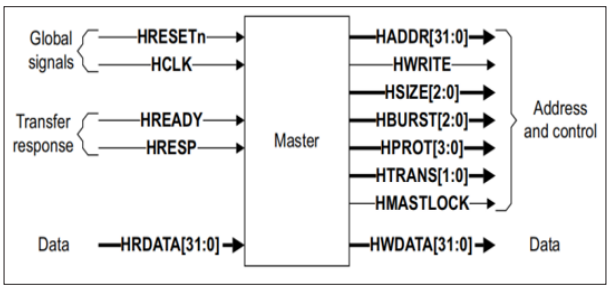

Figure 4: AHB Master Interface

The address and control information are provided by the master to start read/write operations. Presented in Figure 4 is an exhaustive user interface.

B. AHB Slave Interface:

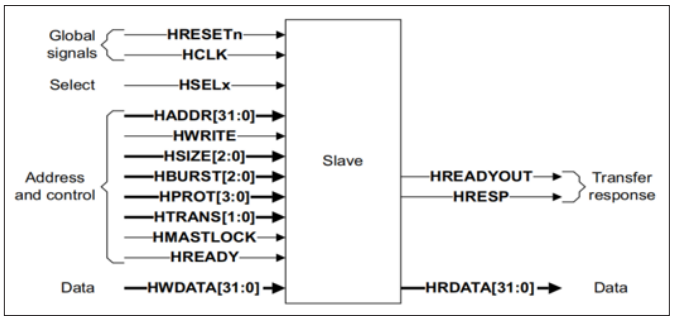

Figure 5: AHB Slave Interface

Slaves in this system are controlled by their masters and react to transfers that they initiate. When a bus transfer occurs, the slave is controlled by the decoder's HSELx select signal. As shown in Figure 5, a slave interface. Slave responds with a signal to master:

- Finishing or extending the bus transfer.
- The bus transfer's success or failure.

II. FORMAL MODELING OF THE AMBA AXI PROTOCOL

The instance AMBA protocol was formally verified using the NuSMV symbolic model checker. This exercise's goal is to suggest a module that can detect deadlocks and test various

solutions to those problems. By using SMV as language, and model each protocol module as an FSM [13]. Users have the ability to specify the initial state of each module as well as the actions for transitioning between them. Using the module descriptions, SMV can construct a model-wide global state transition graph. A small data structure known as a Binary Decision Diagram (BDD) neatly represents the transition relations and states as boolean functions.

The AMBA AXI protocol specification is detailed in this section. Protocol can be thought of as a discrete event system due to its usage of a discrete sense of time, the bus cycle. Discrete event systems can be represented by several popular computer models, including data-flow networks, Petri-nets, and finite state machines [14]. The finite state machine (FSM) is a good computational model to demonstrate the AMBA AXI protocol as a discrete event system. Mainly because they are backed by some model checks, finite state machines were first choice.

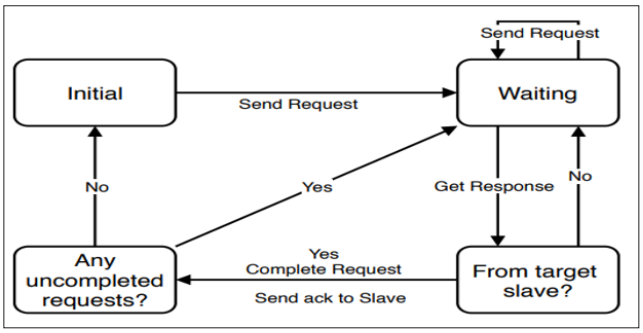

Figure 6: An Abstraction Of Model For One ID

Initialized finite state machines for both the master and the slave. One of the numerous IDs can serve as the master's focal point; from there, the other IDs can be easily modelled after it. Simplified model for a single ID and shown it in Figure 6. No requests with this ID are currently unfinished because the master is in its initial state at startup [15]. After notifying the slave of its requests, it enters a waiting state. At this point, the master is patiently waiting for a response to the request that was given the highest priority at the beginning of the process. The master could keep sending requests even while waiting. Without receiving a response from the slave tasked with the prime request, the master continue to wait before sending an acknowledgement and completing the request. If there are any outstanding requests after that, the master return to the waiting state and await the next highest-priority request. If it isn't the case, it revert to first state.

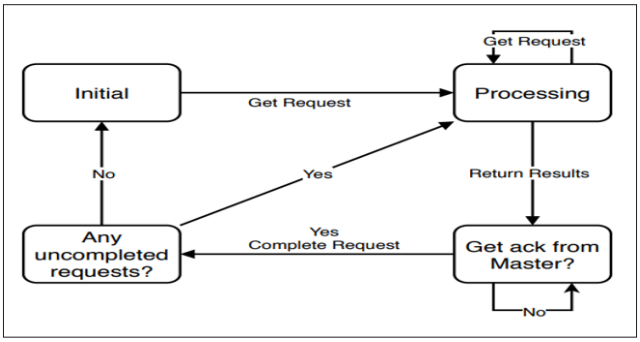

Figure 7: An Abstraction of Slave in Model

Figure 7 depicts the abstract concept of a slave. After entering the initial stage, a slave moves on to the processing state in response to a master's request. It can continue to take requests from masters even while it's in processing. The slave begins to return the result to the specific master as soon as one of these requests is processed [16]. Upon receiving the master's acknowledgement, the slaves proceed to process any remaining requests; otherwise, they return to the original state. The slave patiently wait for the master to acknowledge its requests and block any other unfinished ones until it receives one.

Comparative Analysis of Ahb, Ahb-Lite, and Axi Protocols

The capacity to enable high-performance and scalable connectivity is a crucial component of SoC designs. This relies heavily on the AMBA protocol family developed by ARM. Notable protocols within the AMBA family include AHB, AHB-Lite, and AXI, which stand for Advanced extensible Interface. These protocols cater to varying system needs, ranging from simpler embedded systems to complex high-throughput SoCs.

The AHB protocol is designed for high-performance systems that need multiple components (or masters) to communicate efficiently. AHB-Lite, a simpler derivative of AHB, is optimized for systems with a single master, providing lower power consumption and simpler design [8]. On the other hand, AXI is geared toward high-performance, high-throughput applications, offering advanced features such as support for multiple masters, out-of-order transactions, and high bandwidth.

In this section, Table I compare these three protocols based on key features such as protocol type, master support, burst transfer capabilities, pipelining, latency, bandwidth, and power efficiency. The following table presents a side-by-side comparison of AHB, AHB-Lite, and AX

Table 1: Comparison of AMBA AHB, AHB-Lite, and AXI Protocols

Feature	AHB	AHB-Lite	AXI
Protocol Type	Shared bus	Shared bus (single master)	Point-to-point (switch-based)
Burst Transfer Support	Yes	Yes	Yes (more flexible burst support)
Master Support	Multiple masters	Single master	Multiple masters
Out-of-Order Transactions	No	No	Yes
Pipelining	Limited	Limited	Full pipelining
Data Bus Width	32/64 bits	32/64 bits	32/64/128/256 bits
Latency	Moderate	Moderate	Low
Address Phase and Data Phase	Shared	Shared	Separate
Power Efficiency	Moderate	High (simple logic)	High
Bandwidth	Moderate	Moderate	High
Application	Mid-performance SoCs	Small-scale embedded systems	High-performance SoCs

Literature Review

A comprehensive study of AMBA Bus Protocols is presented in Table II, which forms the basis of the executive summary.

Deeksha and Shivakumar (2019) The protocols for high-performance on-chip microcontroller communication are specified

in the AMBA specification. AHB is one of the conferences that are part of the AMBA family. The AMBA AHB works best when used with system modules that operate at high clock speeds. Backbone operations are managed via the AHB, a high-performance bus. Low-power macro cells' peripheral functions can be effectively coupled to CPUs, on-chip memory, and external off-chip memory interfaces using AHB. Every signal transition is connected to the clock's rising edge to ensure that AHB peripherals can be easily incorporated into any design flow [17].

Giridhar and Choudhury (2019) AHB Protocol design components include master, slave, decoder, and multiplexer functions. Because of its adaptability, the AMBA-AHB protocol can be utilized in any setting where AHB standards are met. Verilog is utilized by developers to construct the design's master, slave, decoder, and multiplexer devices. SV must be used for building the verification environment. Code and functional coverages are calculated and verified using Quest Asim, an advanced verification tool from Mentor Graphics [18].

Gaikwad and Patil (2018) The world is always changing, and the lead times for new IP and chip designs are getting shorter and shorter. As a result, the quickly growing VLSI sector needs a reliable verification mechanism that can be created quickly. The overview of a common signal bus architecture for connecting varied system modules gave rise to SoC design, the primary integrated technique for reducing designs for whole systems. Traditional approaches are inefficient when it comes to evaluating huge SoC, and one of the major issues now is figuring out how to validate these on-chip bus protocols [19].

Makni et al. (2017) Modern systems on a chip (SoC) designs tackle a lot of important problems, and one of them is how to manage communications effectively. The performance of system-on-chip architectures is frequently impacted by the topologies used for on-chip communication. Modern system-on-chips have far higher requirements than their predecessors' interconnects. Manufacturers of systems on a chip have proposed new protocols to apply high-performance data transfer in an effort to solve this issue. Modern systems on a chip (SoCs) for wireless sensor networks often include AMBA AXI4 as an on-chip bus protocol. Stream, burst, and light are its three separate interface protocols [20].

Fern et al. (2016) The objective is to lay forth a standard framework for building a secret Trojan communication channel between SoC components. These channels aim to interfere with the normal operation of buses by targeting particular protocols and topologies. The construction of circuitry that allows covert information flow based on the channel model is possible for any topology and protocol by changing current bus signals only while they are unobserved. The next step is to design the AMBA AXI4 circuitry and construct a system with several slave and master units connected via an AXI4-Lite connection. Now quantify the Trojan channel overhead and demonstrate that Trojans are capable of evading ARM's assertions on protocol compliance testing [21].

Bhaktavatchalu et al. (2016) details the process of creating AXI bus interface modules that can be programmed using Verilog HDL and then implemented in a Xilinx Spartan 3E FPGA and change the number of transfers per burst, the type of burst, and the data size of each interface module. It is possible for numerous masters to talk to various slave memory regions at the same time. Using a Round Robin algorithm, an arbitrator decides how much burst to provide to each bus master. There is a separate decoder module for each of the following channels: write address, write data, write response, read address, and read data [22].

Srinivas and Musala (2016) Implementing social media isn't complete without verification, which is just as crucial as designing. Verification ensures that a DUT satisfies the specification by examining its actual implementation and functioning. In this study, offers an AMBA 3 AHB_LITE protocol that, according to the requirements, uses a UVM environment to encase the DUT entirely. Incorporating test bench components such as tests, environments, agents, drivers, sequencers, monitors, and scoreboards into the verification environment [23].

Shrivastava and Sharma (2016) Implementing social media isn't complete without verification, which is just as crucial as designing. verification checks the functionality and execution of a DUT to see if it matches the specification. In this study, offers an AMBA 3 AHB_LITE protocol that, according to the requirements, uses a UVM environment to encase the DUT entirely. The test bench contains the following components: test, environment, agent, driver, sequencer, monitor, and scoreboard. Together, they form the verification environment [24].

Table 2: Research on Review of AMBA Bus Protocols on AHB, AHB-Lite, and AXI

Author(s), Year	Key Focus / Contributions	Methods / Findings	Limitations / Challenges Identified	Future Work Directions
Deeksha & Shivakumar, 2019	Overview of AMBA and AHB for high-performance on-chip communication	AMBA AHB supports efficient integration of low-power peripherals; single-edge clock simplifies design integration	Complexity increases with larger SoC systems; limited flexibility for heterogeneous IPs	Explore scalable AHB-based interconnects for heterogeneous SoCs; evaluate low-power optimization techniques for AHB designs
Giridhar & Choudhury, 2019	AHB protocol design including master, slave, decoder, mux	Verilog implementation; SV verification; simulation using QuestaSim	Verification overhead increases with system complexity; manual testbench creation is time-consuming	Develop automated verification frameworks using UVM; create reusable verification IPs for AMBA AHB
Gaikwad & Patil, 2018	Need for rapid and robust verification in VLSI; challenges in verifying on-chip bus protocols	Highlights inadequacy of traditional methods for large SoCs	Traditional verification methods do not scale; limited automation	Introduce ML-driven verification methods; design scalable assertion-based verification frameworks

Makni et al., 2017	Challenges in modern SoC communication; comparison of AXI4 protocols	AXI4 (stream, burst, lite) proposed for high-performance data transfer	Traditional interconnects cannot meet modern SoC demands	Explore adaptive AXI interconnects for real-time systems; integrate NoC with AXI protocols
Fern et al., 2016	Covert Trojan channels in AXI4-based SoCs	Developed Trojan model altering unspecified bus signals; validated using AXI4-Lite system	Existing compliance checks cannot detect sophisticated Trojans	Design stronger security and anomaly detection mechanisms for AMBA protocols; develop formal verification for Trojan detection
Bhaktavatchalu et al., 2016	Programmable AXI bus interface in Verilog and FPGA implementation	Reconfigurable modules; round-robin arbitration; concurrent master/slave communication	Limited scalability for very large SoCs; FPGA resource constraints	Develop programmable AXI interfaces for high-end SoCs; integrate advanced arbitration (e.g., QoS-based scheduling)
Srinivas & Musala, 2016	Verification of AMBA 3 AHB-Lite using UVM	UVM-based testbench with driver, sequencer, monitor, scoreboard	Verification complexity increases with protocol variations	Develop reusable UVM components for multiple AMBA versions; implement coverage-guided stimulus generation
Shrivastava & Sharma, 2016	On-chip communication and arbitration in MPSoC architectures	Emphasizes need for flexible, high-performance communication; discusses arbitration for shared buses	Bus architectures face limitations in large MPSoCs; arbitration latency grows with traffic	Explore hybrid NoC–bus architectures; develop dynamic arbitration based on workload prediction

Conclusion and Future Scope

The development of AMBA bus protocols: AHB to AHB-Lite and AXI, supports the increasing needs of system-on-chip (SoC)-based designs. AHB is required to communicate with high-performance and high-bandwidth communication, AHB-Lite fits well in low-power applications, and AXI supports parallel communication and further improves throughput. The invention of these protocols has played a pivotal role in facilitating faster and efficient transfer of data in embedded system especially mobile and multi-core processors. Moreover, the convergence of such features as burst transfers, high clock-frequency, and low-energy operations have contributed to high system performance and power efficiency. These protocols form the basis of SoC designs, as they are constantly improved and new designs with more complexity are supported. They will continue to evolve and will probably bring even more complex architectures, which would further expand the capabilities of embedded systems.

The future of AMBA protocols is bright as it is constantly being integrated with multi-core processors and as fast, low-energy embedded systems are on the rise. The evolution might also see some of the new features, such as, better memory management, coherency protocols designed to support multi-core systems, and support of newer technology, such as, 5G and AI/ML accelerators. As the mobile and IoT devices continue to get more complex, additional improvements in the AMBA protocols, especially in the AXI and AHB-Lite will be needed to accommodate the issues of performance, scalability, and energy efficiency. Also, the prospects of combining AMBA protocols with other interconnects that are industry-standard will almost certainly allow more flexible and adaptable SoC designs, leading to embedded systems of the next generation [25-46].

References

1. Sharma S, Mukherjee C, Gambhir A (2014) A Comparison of Network-on-chip and Buses, Proc. Natl. Conf. Recent Adv. Electron. Commun. Eng., no. March pp 28-29.
2. Kumar KS, Deepthi P (2013) Implementation of Complex interface bridge for LOW and HIGH bandwidth Peripherals Using AXI4-Lite for AMBA. Int. J. Eng. Res. Appl 3: 1005-1010.
3. Thakkar B, JV (2018) "Verification of Driver Logic Using AMBA-AXI UVM," Int. J. VLSI Des. Commun. Syst., 2018, doi: 10.5121/vlsic.2018.9303.
4. Saluja H, Grover N (2019) Memory Controller and Its Interface using AMBA 2.0, Int. J. Eng. Manuf 9: 33-44.
5. Aggarwal A, Sharma N (2016) Implementation of AMBA AHB protocol using verilog HDL, Int. J. Recent Innov. Trends Comput. Commun 4: 573-576.
6. Suan WH, Jambek AB (2017) Design and analysis of microcontroller system using AMBA-Lite bus, EPJ Web Conf 162: 1-6.
7. Chang NYC, Liao YZ, Chang TS (2009) Analysis of shared-link AXI, IET Comput. Digit. Tech 3: 373-383.
8. Shankar D, Girdhar, Shukla NK (2014) Design and Verification of AMBA APB Protocol, Int. J Comput. Appl 95: 21.
9. Saluja H (2017) a Review of Ahb Protocols With Memory Controller, Int. J. Pure Appl. Math., vol. 114: 349-361.
10. Shete PS, Oza S (2014) Design of an Efficient FSM for an Implementation of AMBA AHB Master," Int. J. Adv. Res. Comput. Sci. Softw. Eng 4: 267-271.
11. Rinku R, Dahiya PK (2017) Advance High Performance Bus Arbitration Techniques (AHB): A State-of-the-Art Review, IOSR J. VLSI Signal Process 07: 51-56.
12. Shrivastav A, Tomar GS, Singh AK (2011) Performance Comparison of AMBA Bus-Based System-On-Chip Communication Protocol, in 2011 International Conference on Communication Systems and Network Technologies, IEEE 449-454.
13. Velpula S, Obilineni V, Inthiyaz S, Munuswamy SK (2013) AMBA AHB Bus Protocol Checker," Int. J. Eng. Res. Technol 2: 3717-3724.
14. Swamy AMK, Uma BV (2015) A Literature Review on Wishbone Bus Technique for Network on Chip Architecture," Int. J. Innov. Res. Eng. Manag 2: 5-8.

15. Godhal Y, Chatterjee K, Henzinger TA (2011) Synthesis of AMBA AHB from formal specification: a case study, Int. J. Softw. Tools Technol. Transf 15: 585-601.
16. Yadav MK, Paliwal RK, Mahajan KC (2018) AXI Interface with a Multiple Master and Slave through Interconnect, IJARCCE 7: 132-135.
17. Deeksha L, Shivakumar BR (2019) Effective Design and Implementation of AMBA AHB Bus Protocol using Verilog," in 2019 International Conference on Intelligent Sustainable Systems (ICISS), IEEE 1-5.
18. Giridhar P, Choudhury P (2019) Design and Verification of AMBA AHB, in 2019 1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE), IEEE 310-315.
19. Gaikwad N, Patil VN (2018) Verification of AMBA AXI On-Chip Communication Protocol," in 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), IEEE 1-5.
20. Makni M, Baklouti M, Niar S, Abid M (2017) Performance Exploration of AMBA AXI4 Bus Protocols for Wireless Sensor Networks, in 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA) 1163-1169.
21. Fern N, San I, Koç ÇK, Cheng KT (2016) Hardware Trojans in incompletely specified on-chip bus systems, in 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE) 527-530.
22. Bhaktavatchalu R, Rekha BS, Divya GA, Jyothi VUS (2016) Design of AXI bus interface modules on FPGA," in 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), IEEE 141-146.
23. Srinivas MBR, Musala S (2016) Verification of AHB_ LITE protocol for waited transfer responses using re-usable verification methodology, in 2016 International Conference on Inventive Computation Technologies (ICICT), IEEE 1-3.
24. Shrivastava A, Sharma SK (2016) Various arbitration algorithm for on-chip(AMBA) shared bus multi-processor SoC, in 2016 IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS), IEEE 1-7.
25. Gupta AK, Polu AR, Narra B, Buddula DVKR, Patchipulusu HHS, et al. (2024) Leveraging deep learning models for intrusion detection systems for secure networks. Journal of Computer Science and Technology Studies 6: 199-208.
26. Narra B, Buddula DVK R, Patchipulusu H, Vattikonda N, Gupta A, et al. (2024) The integration of artificial intelligence in software development: Trends, tools, and future prospects Available at SSRN 5596472.
27. Achuthananda RP, Bhumeka N, Dheeraj Varun Kumar RB, Hari Hara SP, Navya V (2024) Evaluating machine learning approaches for personalized movie recommendations: A comprehensive analysis. J Contemp Edu Theo Artific Intel: JCETAI-115.
28. Waditwar P (2024) The Intersection of Strategic Sourcing and Artificial Intelligence: A Paradigm Shift for Modern Organizations. Open Journal of Business and Management 12: 4073-4085.
29. Bitkuri V, Kendyala R, Kurma J, Mamidala JV, Attipalli A, et al. (2024) A Survey on Blockchain-Enabled ERP Systems for Secure Supply Chain Processes and Cloud Integration. International Journal of Technology, Management and Humanities 10: 126-135.
30. Mamidala JV, Bitkuri V, Attipalli A, Kendyala R, Kurma J, et al. (2024) Machine Learning Approaches to Salary Prediction in Human Resource Payroll Systems. Journal of Computer Science and Technology Studies 6: 341-349.
31. Waditwar P (2024) AI for Bathsheba Syndrome: Ethical Implications and Preventative Strategies. Open Journal of Leadership 13: 321-341.
32. Attipalli A, Kendyala R, Kurma J, Mamidala JV, Bitkuri V, et al. (2024) Privacy Preservation in the Cloud: A Comprehensive Review of Encryption and Anonymization Methods. International Journal of Multidisciplinary on Science and Management IJMSM 1: 35-44.
33. Tamilmani V, Maniar V, Singh AA, Kothamaram RR, Rajendran D, et al. (2024) A Review of Cyber Threat Detection in Software-Defined and Virtualized Networking Infrastructures. International Journal of Technology, Management and Humanities 10: 136-146.
34. Singh AAS, Kothamaram RR, Rajendran D, Deepak V, Namburi VT, et al. (2024) A Review on Model-Driven Development with a Focus on Microsoft PowerApps. International Journal of Humanities, Science Innovations and Management Studies 1: 43-56.
35. Padur SKR (2024) AI-augmented platform engineering: Redefining developer experience through autonomous, self-optimizing enterprise systems. International Journal of Science, Engineering and Technology https://ijsret.com/wp-content/uploads/IJSRET_V10_issue6_672.pdf.
36. Gangineni VN, Tyagadurgam MSV, Pabbineedi S, Penmetsa M, Bhumireddy JR, et al. (2024) AI-Powered Cybersecurity Risk Scoring for Financial Institutions Using Machine Learning Techniques (Approved by ICITET 2024). Journal of Artificial Intelligence & Cloud Computing 3: 1-9.
37. Sagili SR, Goswami C, Bharathi VC, Ananthi S, Rani K, et al (2024) Identification of Diabetic Retinopathy by Transfer Learning Based Retinal Images, 2024 9th International Conference on Communication and Electronics Systems (ICCES), Coimbatore 1149-1154.
38. Sagili SR, Kinsman TB (2024) Drive Dash: Vehicle Crash Insights Reporting System," 2024 International Conference on Intelligent Systems and Advanced Applications (ICISAA) 1-6.
39. Padur SKR (2024) Securing Oracle Integration Cloud ERP ecosystems, zero trust architecture, data governance, and compliance automation. International Journal of Science, Engineering and Technology 12: 10-5281.
40. Sagili SR, Chidambaranathan S, Nallametti N, Bodele HM, Raja L, et al. (2024) NeuroPCA: Enhancing Alzheimer's disorder Disease Detection through Optimized Feature Reduction and Machine Learning, 2024 Third International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT) 1-9.
41. SR Sagili VKB, Puli P, Sundaramoorthy MR, KNV (2025) Advancing Cervical Cancer Identification using Generative-based Adversarial Networks: An Integrative Learning Methodology, 2025 6th International Conference for Emerging Technology (INCET) 1-5.
42. Routhu KK (2024) Beyond Automation: AI-Powered Employee Engagement Journeys in Oracle HCM Cloud. KOS Journal of AIML, Data Science, and Robotics 1: 1-6.
43. Routhu KK (2024) The future of HCM: Evaluating Oracle's and SAP's AI-powered solutions for workforce strategy. Journal of Artificial Intelligence, Machine Learning & Data Science 2: 2942-2947.
44. Sannapureddy R, Nadella VM, Nelavelli S (2024) Edge–Cloud Continuums for Latency-Sensitive Tasks. International Journal of AI, BigData, Computational and Management Studies 5: 189-201.

45. Arigela AK, Brahmareddy A, Sreenivas TS, Selvan MP, Venu N, et al. (2024) Optimizing Energy Efficiency and Latency in IoT Devices Through AI-Based Adaptive Protocols in Fog-Edge Computing Environments. In Congress on Smart Computing Technologies 595-607.

46. Nadella VM (2024) Al-Native 6G Network Management. American International Journal of Computer Science and Technology 6: 23-37.

Copyright: ©2025 Usha Mohani kavirayani. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.