CHAPTER 1 – INTRODUCTION TO AHB PROTOCOL

1.1 Background of AMBA

The **Advanced Microcontroller Bus Architecture (AMBA)** is an open-standard bus protocol introduced by **ARM Ltd.** in **1996**.
It was developed to simplify the **interconnection and communication** between different blocks inside a System-on-Chip (**SoC**).
The AMBA family of protocols includes:

- **APB (Advanced Peripheral Bus)** – for low-speed peripherals.
- **AHB (Advanced High-performance Bus)** – for high-speed system components.
- **AXI (Advanced extensible Interface)** – for high-bandwidth, low-latency communication.

AMBA has become a **standard in VLSI and embedded systems** because it allows easy integration of IP cores, supports modular design, and improves design reusability.

1.2 History & Evolution of AHB

Year	**AMBA Version**	**Key Changes in AHB**
1996	AMBA 1.0	Introduced ASB & APB (No AHB yet)
1999	AMBA 2.0	**First introduction of AHB protocol** with high-performance pipelined transfers
2003–2004	AMBA 3.0	**AHB-Lite** introduced – simplified version for single-master systems
2010+	AMBA 4.0 & 5.0	AXI took over for high-performance designs, but AHB still used for medium-performance and control-path communication

Key point:
AHB was designed as a **high-speed backbone bus** inside a chip, while APB connects to low-speed peripherals. AHB is still popular in **microcontrollers (e.g., ARM Cortex-M series)** and **mid-range SoCs**.

1.3 Importance in SoC and VLSI Design

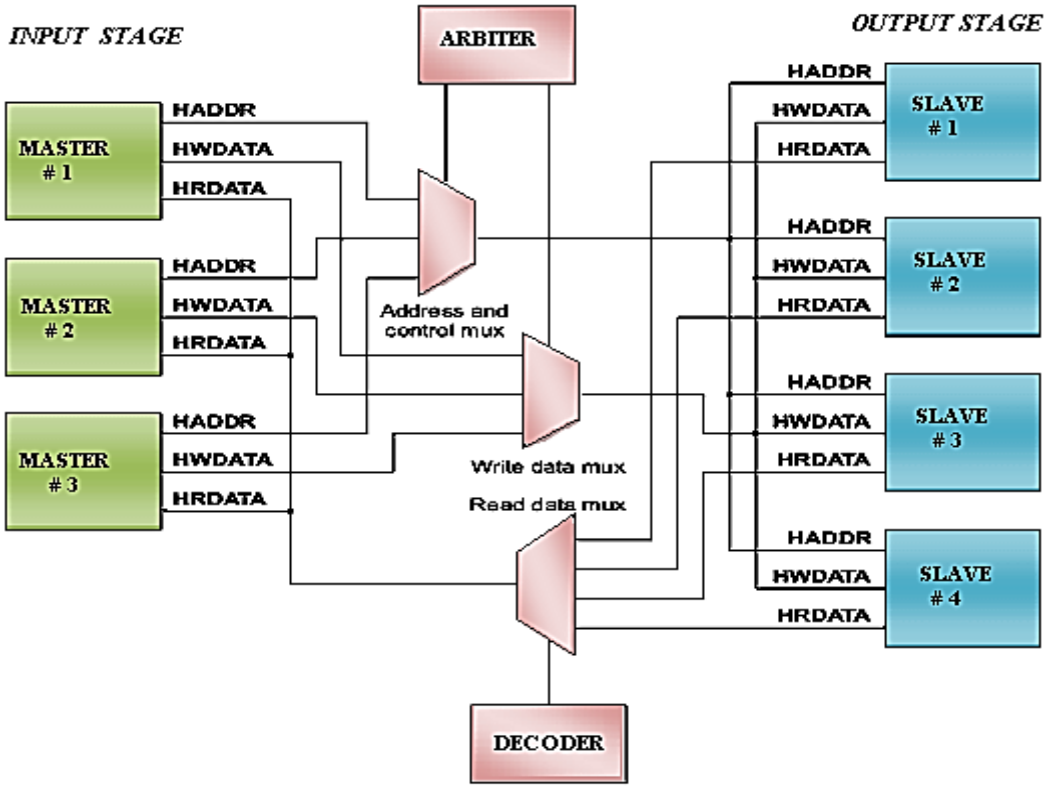
- **Central Data Path** – AHB acts like a "main highway" connecting CPU, memory, and high-speed peripherals.
- **Supports Pipelining** – Allows overlapping of address and data phases for higher throughput.
- **Efficient for Embedded Systems** – Balances complexity and performance, making it suitable for MCUs, FPGAs, and ASICs.
- **Scalability** – Can handle multiple masters and multiple slaves with arbitration control.

1.4 Where AHB is Used

- **ARM Cortex-M microcontrollers** (MCU internal bus)
- **FPGA SoCs** (e.g., Xilinx, Intel) for memory & DMA transfers
- **ASICs** for connecting processor cores to on-chip SRAM, ROM, and high-speed peripherals

CHAPTER 2 – AHB PROTOCOL BASICS

2.1 AHB Architecture Overview

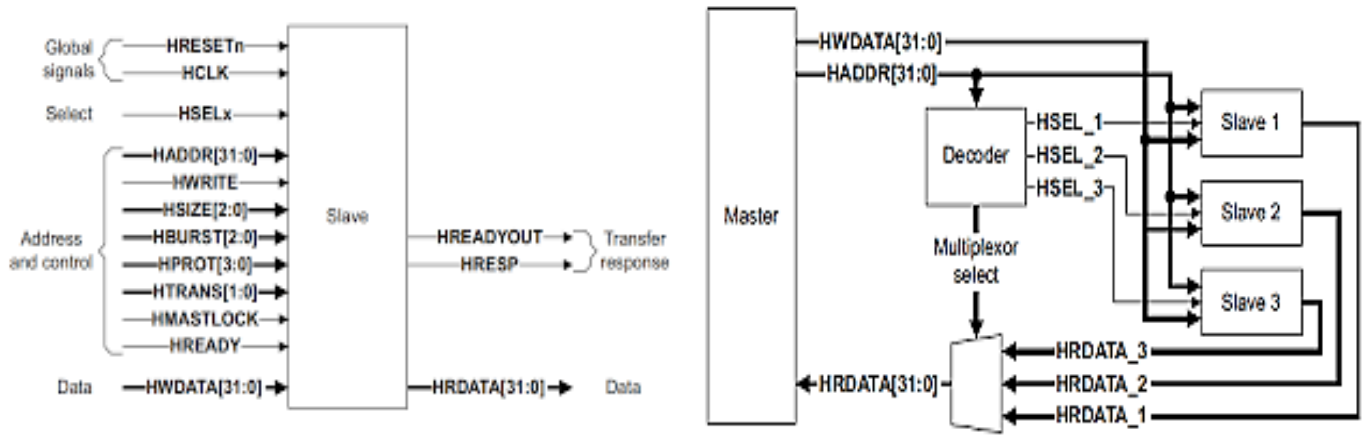


The **Advanced High-performance Bus (AHB)** is a **system-level bus** in the AMBA architecture. It is designed for high-performance, high-bandwidth communication between **high-speed system components** such as the CPU, on-chip memory, and DMA controllers.

An AHB-based system typically consists of:

- **Masters** → Components that initiate data transfers (e.g., CPU, DMA controller).
- **Slaves** → Components that respond to read/write requests (e.g., memory, peripherals).
- **Arbiter** → Decides which master gets control of the bus when multiple masters request access.
- **Decoder** → Identifies the target slave based on the address.
- **Multiplexer** → Routes signals between the active master and selected slave.

Block Diagram (Simplified)



2.2 Key Features of AHB Protocol

- **High Performance** – Uses pipelined architecture for faster transfers.

- **Multiple Masters** – Can connect more than one data-initiating unit.
- **Multiple Slaves** – Supports a variety of memory and peripheral devices.
- **Burst Transfers** – Transfers multiple data words in a single transaction.
- **Split Transfers** – Allows a slave to release the bus if it cannot respond immediately.
- **Single Clock Edge Operation** – Simplifies timing and improves synchronization.
- **Separate Address & Data Phases** – Overlap in execution improves throughput.

2.3 AHB vs. Other AMBA Protocols

Feature	**AHB**	**APB**	**AXI**
Speed	High	Low	Very High
Pipelining	Yes	No	Yes (Advanced)
Burst Support	Yes	No	Yes
Multiple Masters	Yes	No	Yes
Usage	CPU ↔ Memory/High-speed peripherals	Low-speed peripherals (UART, GPIO)	High-performance SoCs, multi-core
Complexity	Medium	Low	High

2.4 Typical AHB System Example

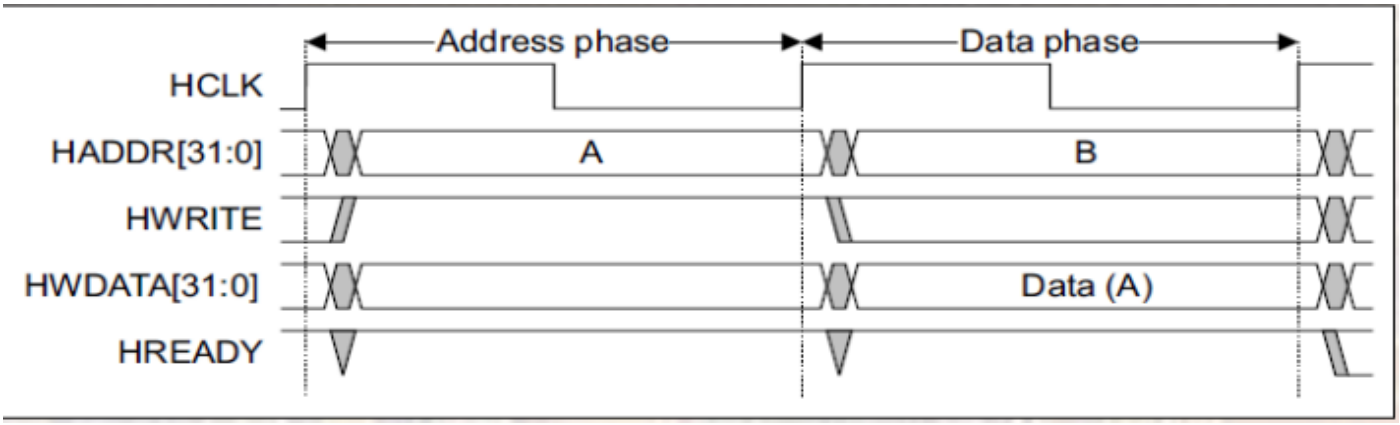
In a **microcontroller** like ARM Cortex-M4:

- The **CPU** is an AHB master.
- On-chip **SRAM & Flash** are AHB slaves.
- A **DMA controller** is also a master.
- The **APB bridge** connects AHB to slower peripherals like timers and UART.

CHAPTER 3 – AHB SIGNAL DESCRIPTIONS & OPERATIONS

3.1 Overview

The AHB protocol uses a **set of signals** to control and transfer data between masters and slaves.
These signals can be grouped into **Address**, **Data**, **Control**, and **Response** categories.
Transfers occur in **two main phases**:



1. **Address Phase** – The master places the address and control signals on the bus.
2. **Data Phase** – The actual data is transferred between the master and slave.

3.2 AHB Signal Categories

A. Address & Control Signals

Signal	Direction	Description
HADDR[31:0]	Master → Slave	32-bit address of the transfer
HTRANS[1:0]	Master → Slave	Transfer type (IDLE, BUSY, NONSEQ, SEQ)
HSIZE[2:0]	Master → Slave	Size of the transfer (byte, halfword, word)
HBURST[2:0]	Master → Slave	Defines burst type (SINGLE, INCR, WRAP)
HWRITE	Master → Slave	1 = Write, 0 = Read
HPROT[3:0]	Master → Slave	Protection control (privileged/user, cacheable, bufferable)

B. Data Signals

Signal	Direction	Description
HWDATA[31:0]	Master → Slave	Write data from master
HRDATA[31:0]	Slave → Master	Read data from slave

C. Handshake & Control Signals

Signal	Direction	Description
HREADY	Slave → Master	Indicates the slave is ready for the next transfer
HRESP[1:0]	Slave → Master	Response status (OKAY, ERROR, RETRY, SPLIT)

D. Clock & Reset

Signal	Direction	Description
HCLK	Global	Clock signal for AHB
HRESETn	Global	Active-low reset

3.3 Transfer Types (HTRANS)

HTRANS	Transfer Type	Meaning
00	IDLE	No transfer is happening
01	BUSY	Master keeps the bus but delays new transfer
10	NONSEQ	First transfer of a burst or single transfer
11	SEQ	Sequential transfer in a burst

3.4 Transfer Phases

1. **Address Phase**
   - Master provides **HADDR**, **HTRANS**, **HWRITE**, **HSIZE**.
   - Control signals are set according to transfer type.
2. **Data Phase**
   - For **write**: Master places **HWDATA** on the bus.
   - For **read**: Slave places **HRDATA** on the bus.
   - **HREADY** signal indicates completion.

3.5 Read & Write Cycle Examples

ALL AHB TRANSFERS you can refer this:https://faculty-web.msoe.edu/johnsontimoj/ELE4142/files4142/parallel_communications_ahb.pdf

Write Cycle

1. Master sets address and control signals.
2. On the next clock, master sends write data (**HWDATA**).
3. Slave sets **HREADY** high when ready.

Read Cycle

1. Master sets address and control signals.
2. Slave responds with data on **HRDATA** when ready.
3. Master latches data when **HREADY** = 1.

3.6 AHB Timing Diagram (Simplified)

Write Transfer Example:

Clock: ↑ ↑ ↑ ↑ ↑

HADDR: A1 A2 -- -- --

HWDATA: -- D1 D2 -- --

HREADY: 1 1 1 1 1

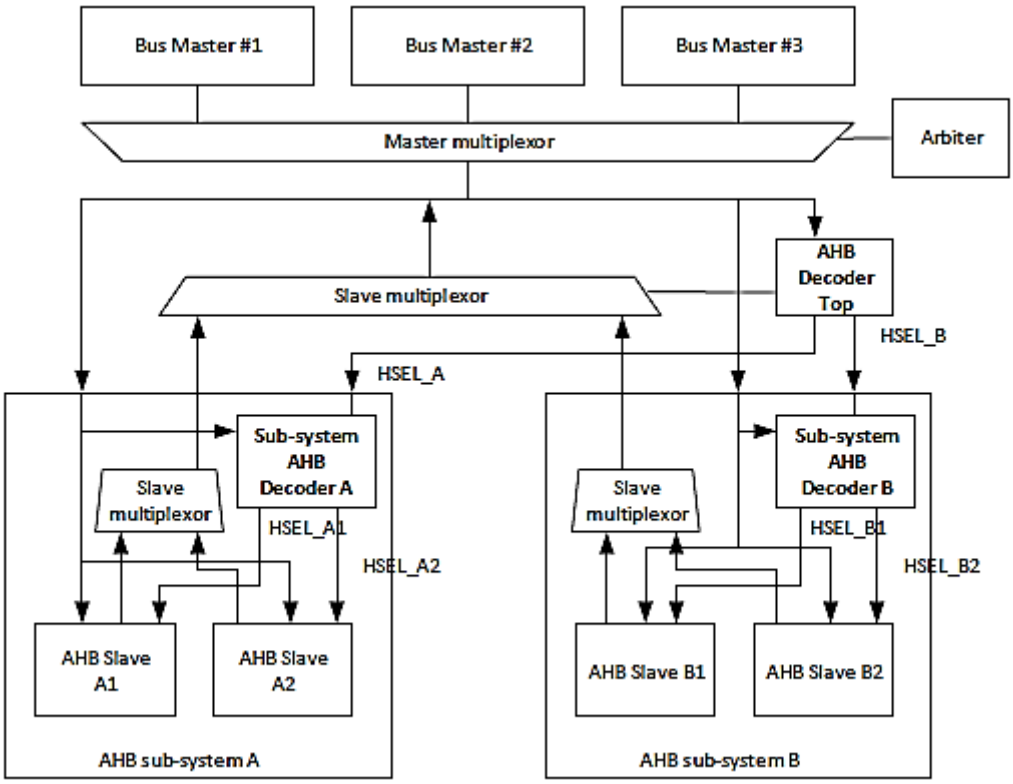
Here:

- **A1** = Address for first transfer, **D1** = Data for that transfer.
- Address phase and data phase are **pipelined** for better performance.

CHAPTER 4 – AHB ARBITRATION, DECODING & PIPELINING

4.1 Overview

In an AHB-based system, **multiple masters** may need to use the bus at the same time (e.g., CPU and DMA controller).
The protocol includes **arbitration**, **address decoding**, and **pipelining** to ensure smooth, high-performance operation.



4.2 Arbitration in AHB

Arbitration is the process of deciding which master gets control of the bus when two or more masters request it.

4.2.1 Arbitration Process

1. Each master asserts a **bus request signal** to the arbiter when it needs the bus.
2. The arbiter decides which master to grant based on a priority scheme.
3. The arbiter issues a **bus grant signal** to the selected master.

4.2.2 Arbitration Schemes

- **Fixed Priority** – One master always has higher priority over others.
  Advantage: Simple design.
  Disadvantage: Lower-priority masters may starve.
- **Round-Robin** – Masters take turns in a cyclic manner.
  Advantage: Fair access.
  Disadvantage: Slightly more complex logic.

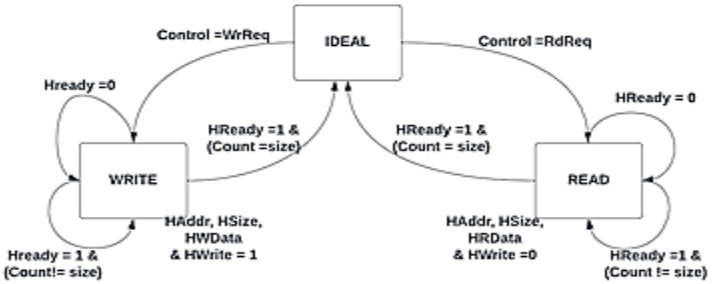
4.3 Address Decoding

The **decoder** determines which slave should respond to a transfer based on the **HADDR** value.

- Each slave is assigned a **specific address range**.
- The decoder activates the **select signal (HSELx)** for the matching slave.
- Only the selected slave responds to the transfer; others remain idle.

4.4 Pipelining in AHB

Pipelining allows the address phase of one transfer to overlap with the data phase of the previous transfer.



Benefits:

- Increases throughput.
- Reduces idle cycles on the bus.

Example:

- Cycle 1: Address for Transfer 1 sent.
- Cycle 2: Data for Transfer 1 + Address for Transfer 2 sent simultaneously.
- Cycle 3: Data for Transfer 2 + Address for Transfer 3 sent, and so on.

4.5 Split and Retry Transfers

When a slave cannot respond immediately:

- **SPLIT response**: The slave releases the bus and later signals the arbiter to re-grant it to the same master.
- **RETRY response**: The slave asks the master to retry the transfer later without losing ownership.

4.6 Error Handling

The **HRESP** signal indicates the status of a transfer:

- **OKAY (00)** – Successful transfer.
- **ERROR (01)** – Transfer failed (e.g., invalid address).
- **RETRY (10)** – Retry the transfer later.
- **SPLIT (11)** – Slave releases the bus for other masters.

CHAPTER 5 – APPLICATIONS, ADVANTAGES & LIMITATIONS OF AHB PROTOCOL

5.1 Applications of AHB Protocol

The AHB protocol is widely used in **SoC (System-on-Chip)** and **VLSI design** where high-speed communication is required between system components.

Common use cases:

1. **Microcontrollers (MCUs)**
   - ARM Cortex-M series uses AHB or AHB-Lite as the main system bus.
2. **Embedded SoCs**
   - Connecting CPU cores to on-chip SRAM, ROM, and high-speed I/O.
3. **FPGA-based Systems**
   - Implementing memory controllers, DMA units, and high-speed data paths.

4. **ASIC Designs**
    - Linking multiple processor cores, accelerators, and peripherals.
5. **Bridging to Other Buses**
    - AHB-to-APB bridge for connecting slower peripherals like UART, GPIO, SPI.

5.2 Advantages of AHB Protocol

- **High Performance** – Pipelining and burst transfers increase throughput.
- **Supports Multiple Masters** – Efficient in systems with CPU + DMA + other initiators.
- **Scalable** – Easily integrates new masters and slaves without major changes.
- **Standardized** – AMBA compliance ensures compatibility with IP cores from different vendors.
- **Reduced Latency** – Overlapping address/data phases minimizes idle time.
- **Error Handling** – Built-in response signals (OKAY, ERROR, RETRY, SPLIT).

5.3 Limitations of AHB Protocol

- **Lower Performance than AXI** – For extremely high-speed, multi-threaded applications, AXI is preferred.
- **More Complex than APB** – Requires more control logic, making it less suitable for simple peripherals.
- **Limited Support for Out-of-Order Transactions** – Unlike AXI, AHB processes transactions in order.
- **Not Ideal for Very High-Frequency Designs** – For >500 MHz designs, AXI often replaces AHB.

5.4 Future Trends

While **AXI** has become the dominant choice in high-end SoCs, AHB continues to thrive in:

- **Low-to-medium performance systems** where design simplicity is preferred.
- **Microcontrollers** where deterministic timing is more important than peak bandwidth.
- **IoT devices** and **medical electronics** where low power and reliable transfers are needed.

With evolving low-power and embedded applications, AHB (especially **AHB-Lite**) is expected to remain relevant for years to come.

REFERENCES

1. ARM Ltd. – *AMBA AHB and AHB-Lite Protocol Specification (AMBA 5)*
   🔗 https://developer.arm.com/documentation/ihi0033/latest
2. ARM Ltd. – *AMBA 3 AHB-Lite Protocol Specification*
   🔗 https://developer.arm.com/documentation/ihi0033/b/
3. ARM Ltd. – *AMBA 2.0 Specification (Original AHB Introduction)*
   🔗 https://developer.arm.com/documentation/ihi0011/latest
4. ARM Ltd. – *AMBA Design Kit and Technical Resources*
   🔗 https://developer.arm.com/architectures/system-architectures/amba