AHB2APB Bridge Verification

Author: Shanthi V A, Principal Engineer at Maven Silicon

Table of Contents

1. Introduction

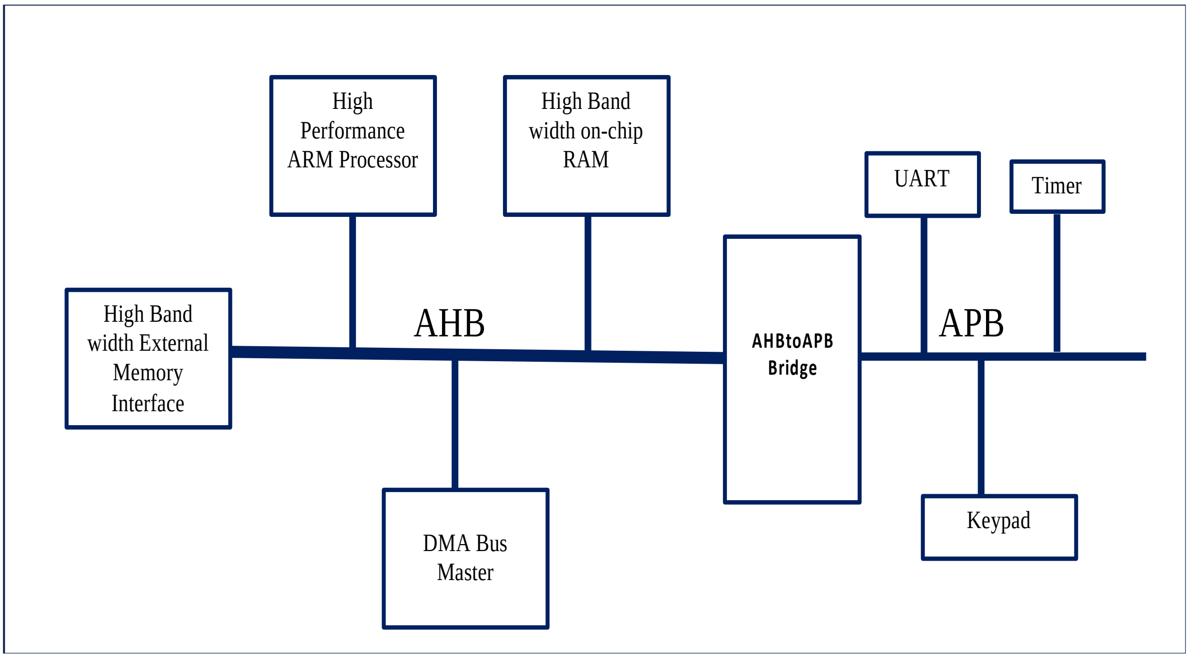
The ARM Advanced Microcontroller Bus Architecture (AMBA) is an open-standard, on-chip interconnect specification for the connection and management of functional blocks in system-on-a-chip (SoC) designs. It facilitates the development of multi-processor designs with large numbers of controllers and components with a bus architecture. Today, AMBA is widely used on a range of ASIC and SoC parts including applications processors used in modern portable mobile devices like smartphones.

AMBA was introduced by ARM in 1996, which includes the buses Advanced System Bus(ASB) and Advanced Peripheral Bus(APB). In its second version, in 1999 AMBA2 Advanced High-performance Bus (AHB) has been added. It is a single clock edge protocol.

AMBA AHB is a bus interface suitable for high-performance synthesizable designs. It defines the interface between components, such as masters, interconnects, and slaves. AMBA AHB implements the features required for high-performance, high-clock frequency systems including:

- Burst transfers.
- Single clock-edge operation.
- Non-tristate implementation.
- Wide data bus configurations, 64, 128, 256, 512, and 1024 bits.

The most common AHB slaves are internal memory devices, external memory interfaces, high-bandwidth peripherals, and a bridge to a narrower APB bus on which the lower bandwidth peripheral devices are located, as shown in the figure below. AHB is widely used on ARM7, ARM9 and ARM Cortex-M-based designs. A simple transaction on the AHB consists of an address phase and a subsequent data phase (without wait states: only two bus cycles).



Typical AMBA System

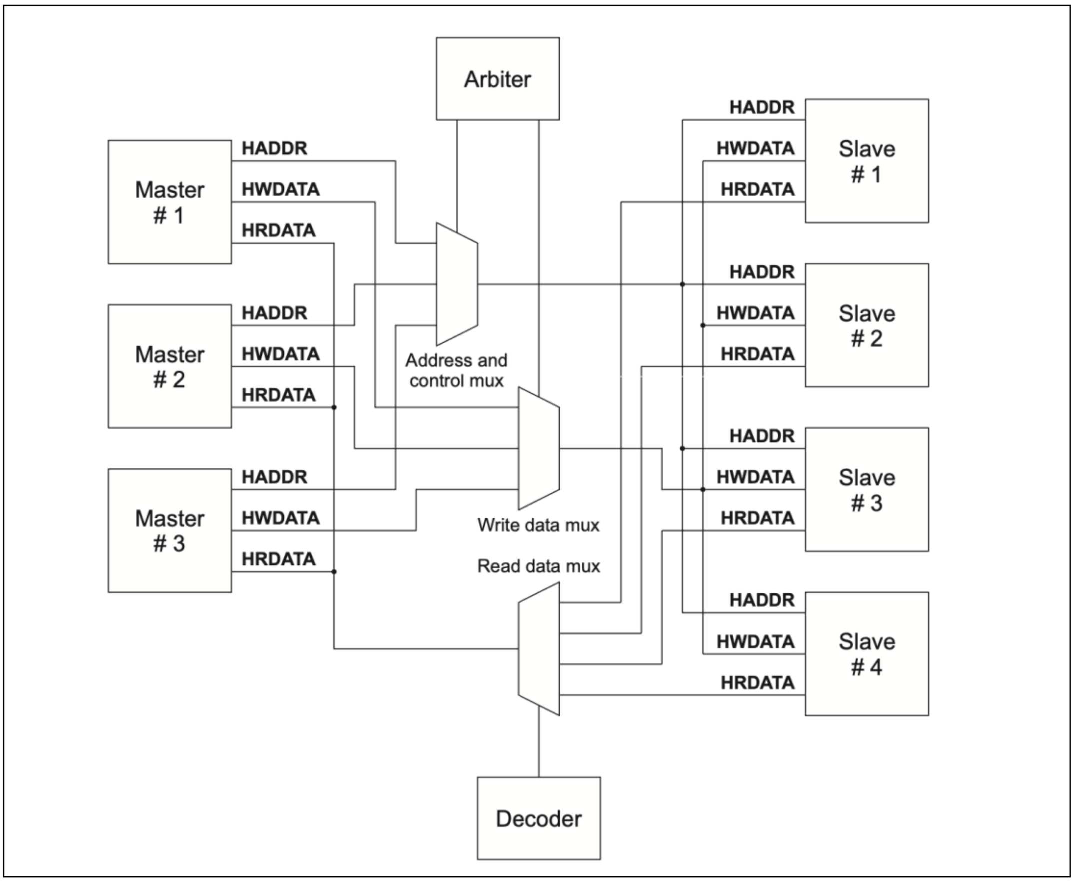
The AMBA APB is designed for low bandwidth control accesses, for example, register interfaces on system peripherals such as Timers, UART, and GPIO. This bus has an address and data phase similar to AHB, but a much reduced, low-complexity signal list (for example no bursts). Furthermore, it is an interface designed for a low-frequency system with a low bit width (32 bits). It is optimized for minimal power consumption and reduced interface complexity.

AHB	APB
High Performance	Low Power
Pipelined Operation	Latched address and control
Multiple Bus Masters	Simple Interface
Burst Transfers	-
Split Transactions	-
-	Suitable for many peripherals

2. AHB Protocol

The AMBA AHB bus protocol is designed to be used with a central multiplexor interconnection scheme. Using this scheme all bus masters drive out the address and control signals indicating the transfer they wish to perform and the arbiter determines which master has its address and control signals routed to all of the slaves. A central decoder is also required to control the read data and response signal multiplexor, which selects the appropriate signals from the slave that is involved in the transfer.

The below figure illustrates the structure required to implement an AMBA AHB design with three masters and four slaves.



Multiplexor Interconnection
Ref:AMBA Specification(Rev 2.0)

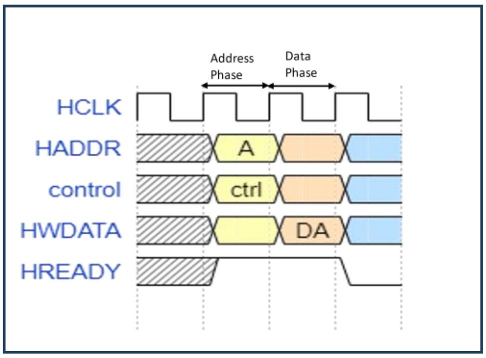
Before an AMBA AHB transfer can commence the bus master must be granted access to the bus. This process is started by the master asserting a request signal to the arbiter. Then the arbiter indicates when the master will be granted use of the bus.

A granted bus master starts an AMBA AHB transfer by driving the address and control signals. These signals provide information on the address, direction and width of the transfer, as well as an indication if the transfer forms part of a burst. A write data bus is used to move data from the master to a slave, while a read data bus is used to move data from a slave to the master.

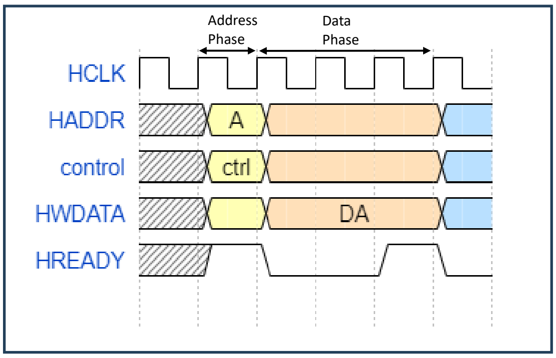
Every transfer consists of:

- an address and control cycle
- one or more cycles for the data.

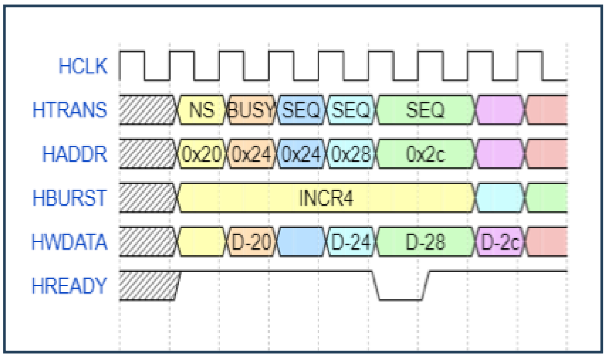
The address cannot be extended and therefore all slaves must sample the address during this time. The data, however, can be extended using the **HREADY** signal. When LOW this signal causes wait states to be inserted into the transfer and allows extra time for the slave to provide or sample data.



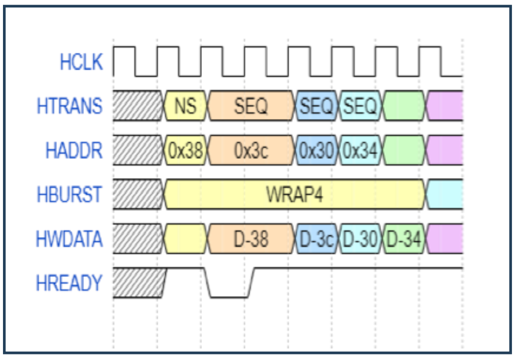
Basic AHB Transfer



Transfer with wait state



Burst Transfer(Increment)- Pipelined Opereration



Burst Transfer(Wrapping) – Pipelined Operation

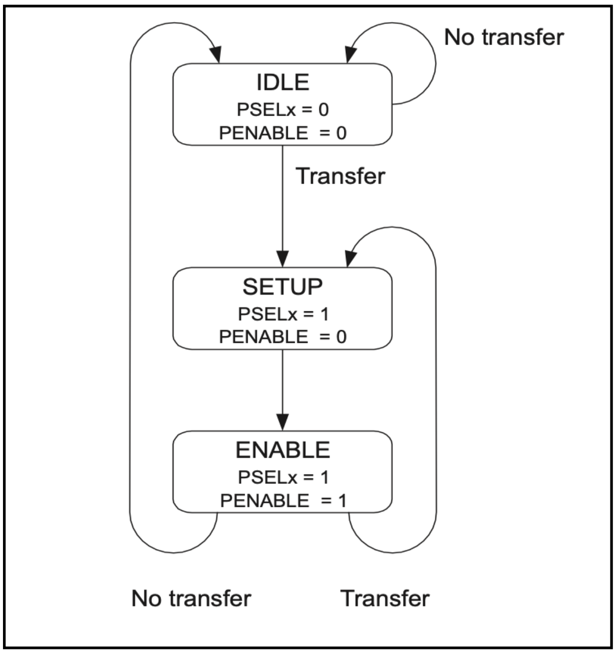
AHB Signals

Signal Name	**Source**	**Description**
HCLK Bus clock	Clock Source	This clock times all bus transfers. All signal timings are related to the rising edge of **HCLK.**
HRESETn Reset	Reset Controller	The bus reset signal is active LOW and is used to reset the system and the bus. This is the only active LOW signal.
HADDR[31:0] Address bus	Master	The 32-bit system address bus.
HTRANS[1:0] Transfer type	Master	Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY.
HWRITE Transfer direction	Master	When HIGH this signal indicates a write transfer and when LOW a read transfer.
HSIZE[2:0] Transfer size	Master	Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit) or word (32-bit). The protocol allows for larger transfer sizes up to a maximum of 1024 bits.
HBURST[2:0] Burst type	Master	Indicates if the transfer forms part of a burst. Four, eight, and sixteen beat bursts are supported and the burst may be either incrementing or wrapping.
HWDATA[31:0] Write data bus	Master	The write data bus is used to transfer data from the master to the bus slaves during write operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation.
HRDATA[31:0] Read data bus	Slave	The read data bus is used to transfer data from bus slaves to the bus master during read operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation.
HREADY Transfer done	Slave	When HIGH the **HREADY signal indicates that a** transfer has finished on the bus. This signal may be driven LOW to extend a transfer. Note: Slaves on the bus require **HREADY as both** an input and an output signal.
HRESP[1:0] Transfer response	Slave	The transfer response provides additional information on the status of a transfer. Four different responses are provided, OKAY, ERROR, RETRY and SPLIT.

3. APB Protocol

AMBA APB provides a low-cost interface that is optimized for minimal power consumption and reduced interface complexity. The APB interfaces to any peripherals that are low-bandwidth and do not require the high performance of a pipelined bus interface. The APB has an unpipelined protocol.

All signal transitions are only related to the rising edge of the clock to enable the integration of APB peripherals easily into any design flow. Every transfer takes two cycles in AMBA2, APB or more cycles in the later releases.



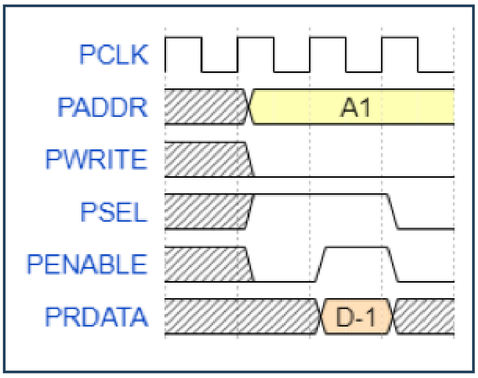
State Diagram
Ref: AMBA Specification(Rev 2.0)

The APB2 protocol can be explained with the help of the above state diagram.

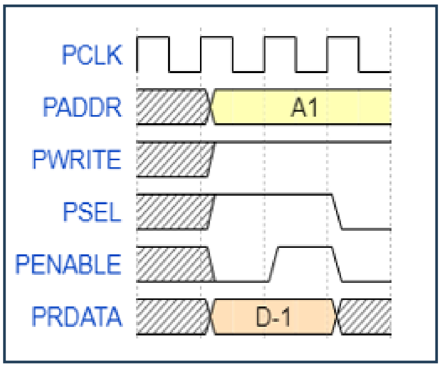
IDLE: The default state for the peripheral bus.

SETUP: When a transfer is required the bus moves into the SETUP state, where the appropriate select signal, **PSELx**, is asserted. The bus only remains in the SETUP state for one clock cycle and will always move to the ENABLE state on the next rising edge of the clock.

ENABLE: In the ENABLE state the enable signal, **PENABLE** is asserted. The address, write and select signals all remain stable during the transition from the SETUP to ENABLE state. The ENABLE state also only lasts for a single clock cycle and after this state, the bus will return to the IDLE state if no further transfers are required. Alternatively, if another transfer is to follow then the bus will move directly to the SETUP state.



Read Transfer



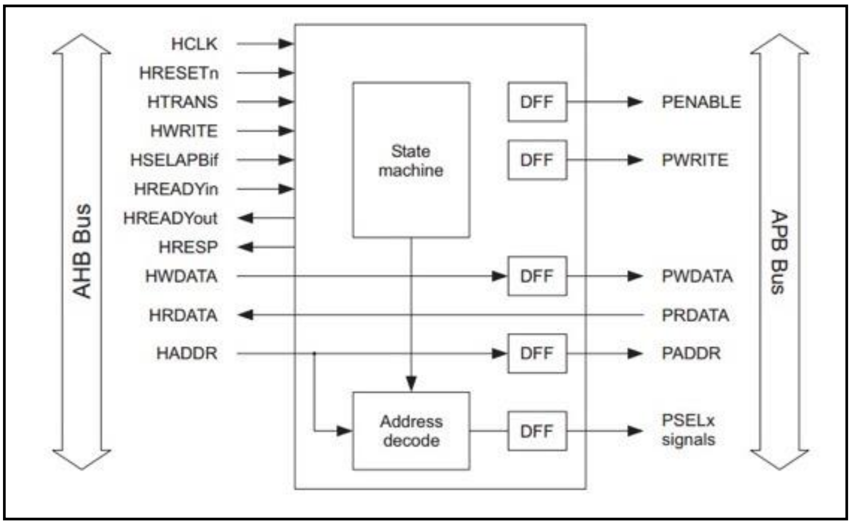
Write Transfer

APB Signals

Signal Name	Description
PCLK Bus Clock	This clock times all bus transfers. Both the LOW phase and HIGH phase of PCLK are used to control transfers.
PRESETn APB Reset	The bus reset signal is active LOW and is used to reset the system.
PENABLE APB Strobe	This strobe signal is used to time all accesses on the peripheral bus. The enable signal is used to indicate the second cycle of anAPB transfer. The rising edge of PENABLE occurs in the middle of the APB transfer.
PADDR[31:0] APB address bus	This is the APB address bus, which may be up to 32-bits wide and is driven by the peripheral bus bridge unit.
PWRITE APB transfer direction	When HIGH this signal indicates an APB write access and when LOW a read access.
PRDATA APB read data bus	The read data bus is driven by the selected slave during read cycles (when PWRITE is LOW). The read data bus can be up to 32-bits wide.
PWDATA APB write data bus	The write data bus is driven by the peripheral bus bridge unit during write cycles (when PWRITE is HIGH). The write data bus can be up to 32-bits wide.
PSELx APB select	A signal from the secondary decoder, within the peripheral bus bridge unit, to each peripheral bus slave x. This signal indicates that the slave device is selected and a data transfer is required. There is a PSELx signal for each bus slave.

4. AHBtoAPB Bridge

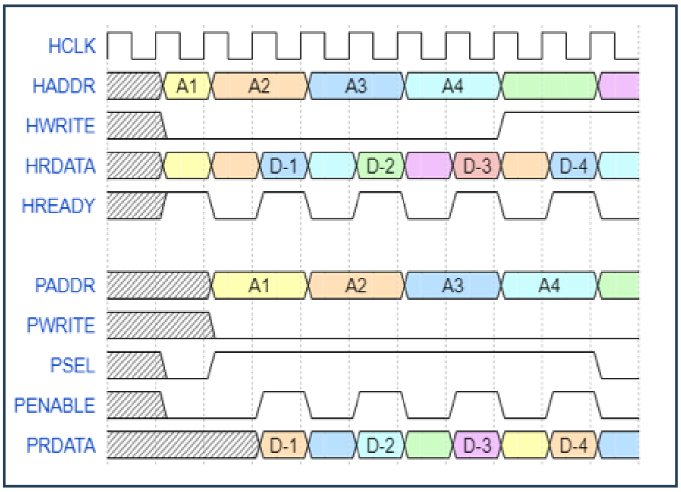
The AHBtoAPB Bridge is an AHB slave, providing an interface between the high-speed AHB and the low-power APB. Read and write transfers on the AHB are converted into equivalent transfers on the APB. As the APB is not pipelined, then wait states are added during transfers to and from the APB when the AHB is required to wait for the APB.



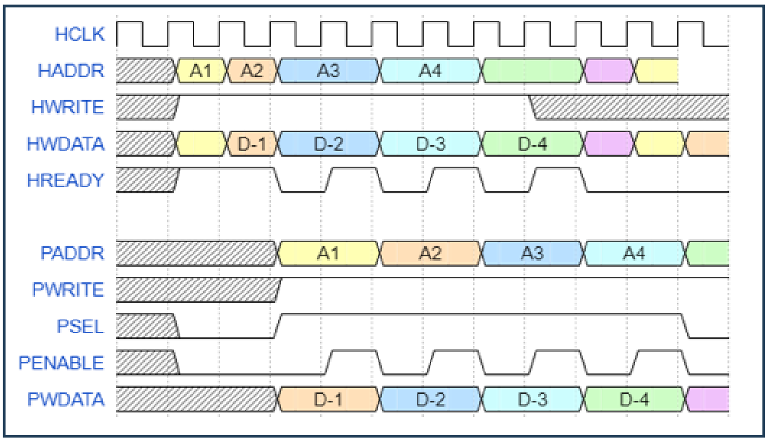
Block Diagram of AHBtoAPB Bridge
Ref: AHB Example AMBA System Technical Reference Manual

The bridge unit converts system bus transfers into APB transfers and performs the following functions:

- Latches the address and holds it valid throughout the transfer.
- Decodes the address and generates a peripheral select, PSELx. Only one select signal can be active during a transfer.
- Drives the data onto the APB for a write transfer.
- Drives the APB data onto the system bus for a read transfer.
- Generates a timing strobe, PENABLE, for the transfer.



Interfacing AHBtoAPBRead transfer

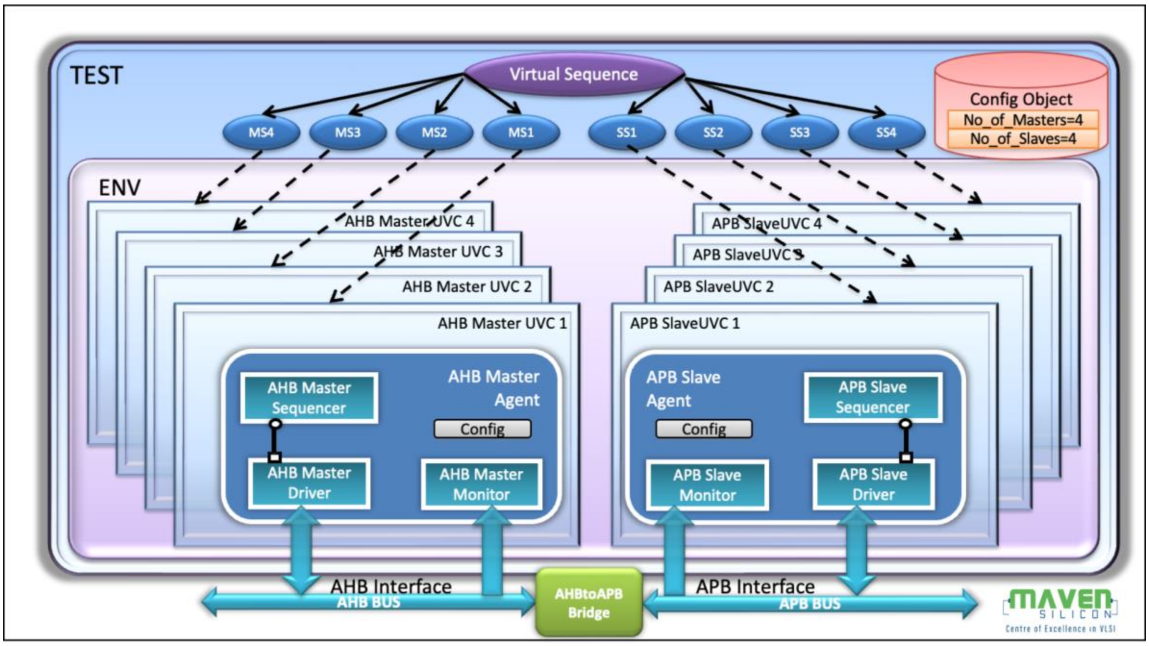


Interfacing AHBtoAPBWrite transfer

5. AHB2APB Bridge Verification

The verification process starts with the preparation of the Verification Plan(Vplan). The verification plan defines the verification intent of the DUV/DUT[Design Under Verification/Test]. It captures all the design features and defines how each feature can be verified and tracked closely, including TB architecture, Coverage Model, Verification Strategy, Test Scenarios, list of assertions etc. The verification team will develop the TB and write multiple testcases taking the Vplan as a reference.

A typical AMBA system supports connections of up to 16 AHB Masters. Among all the masters, the granted bus masters communicate with the AHB slave, in this case, AHB2APB Bridge. The Bridge inturn communicates with one of the APB peripherals.



AHB2APB Bridge UVM Verification Environment

The functionality of the AHB2APB Bridge is verified by connecting the DUT with an AHB master agent and an APB slave agent and by initiating the write and read transfers from the AHB agent.

Testbench/VE is developed in such a way that from the test we can configure the number of AHB masters, and the number of APB Peripherals. With the help of a virtual sequence, in each test case, we can control the communication between different AHB bus masters and APB peripherals. Different sequences are developed to verify scenarios like increment & wrapping type bursts of various data sizes for both read and write operations.

Based on the coverage analysis, we may need to add additional test cases to cover the missed features by overriding the existing constraints. This is how coverage is going to be improved and we can achieve coverage closure.

6. Model Interview Questions

1. Is HREADY an input or output from slaves?
2. Can a BUSY transfer occur at the end of a burst?
3. The specification recommends that AHB slave can insert only 16 wait states. What should an AHB slave do if it needs more than 16 cycles?
4. When can Early burst Termination occurs?
5. What type of response does an AHBtoAPB Bridge generate?
6. Is a dummy master really necessary?
7. Will a master always lose the bus after a SPLIT response?
8. Why does a non-okay response require two cycles in AHB?