ECE593 – Fundamentals of Pre-Silicon Validation

Verification of an AHB2APB Bridge

Milestone 1

Team Members:
Daniel Jacobsen(dj23@pdx.edu): 905696959
Balaji Ginkal Harisha(balajig@pdx.edu): 952724638
Mohith Kumar Bennahatti Chikkegowda(mohithb@pdx.edu): 902084859
Akshaya Kudumalakunte Ravi Kumar(akshayak@pdx.edu): 969375449

Introduction

The project of design and verification of AHB2APB is done using System Verilog and UVM based testbench development methodologies to create Verification IP components of the design.

The APB2APB bridge is a crucial component in SOC architectures as it enables communication between the high-performance AHB(Advanced High-Performance Bus) and the low-performance APB(Advanced Peripheral Bus).

The RTL code for our project is chosen from the github: https://github.com/prajwalgekkouga/AHB-to-APB-Bridge

The tool used to compile the design files and to verify is QuestaSim. QuestaSim was chosen as it provides full support for System Verilog and UVM.

AHB2APB Bridge

The AHB2APB bridge is a crucial interface that enables seamless communication between the Advanced High-Performance Bus (AHB) and the Advanced Peripheral Bus (APB) in AMBA-based system-on-chip (SoC) designs. The AHB is a high-speed, pipelined bus used for performance-critical components, while the APB is a simpler, low-power bus designed for peripherals like timers and UARTs. Since these buses have different architectures and operating mechanisms, the AHB2APB bridge ensures proper data transfer by converting AHB transactions into APB transactions while handling synchronization, protocol adaptation, and control signals efficiently.

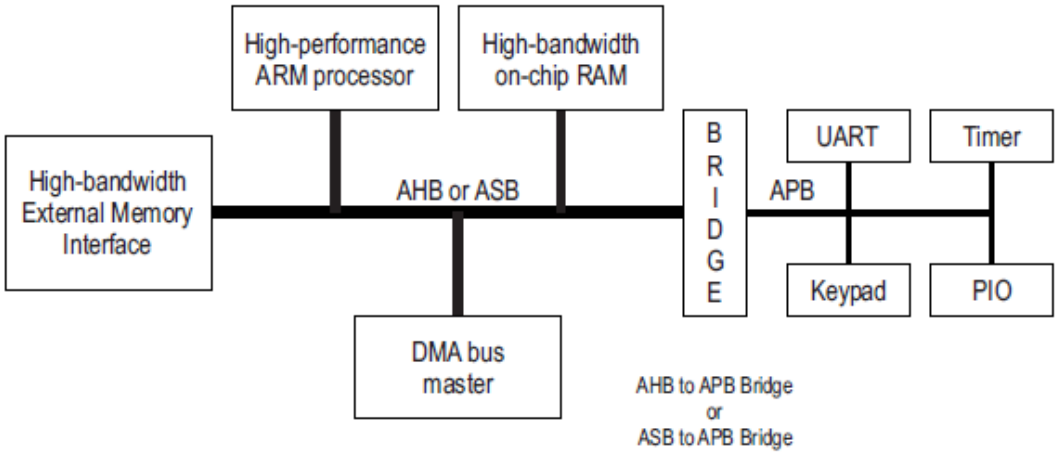

Fig: AHB2APB Bridge

AHB Bus Protocol

AHB is a new generation of AMBA bus which is intended to address the requirements of high-performance synthesizable designs. It is a new level of bus that sits above APB and implements the features required for high performance and high-clock frequency systems as: Burst functions, Split transactions, Single-cycle bus MASTER handover, Single clock edge operation, Non-tristate implementation, Wider dat, configurations(64/128 bits), Larger overhead of approximately 27 control signals, upto 75 MASTERs are allowed, Split transaction phases: Address, data(Pipelined), HREADY signal allows insertion of wait-states

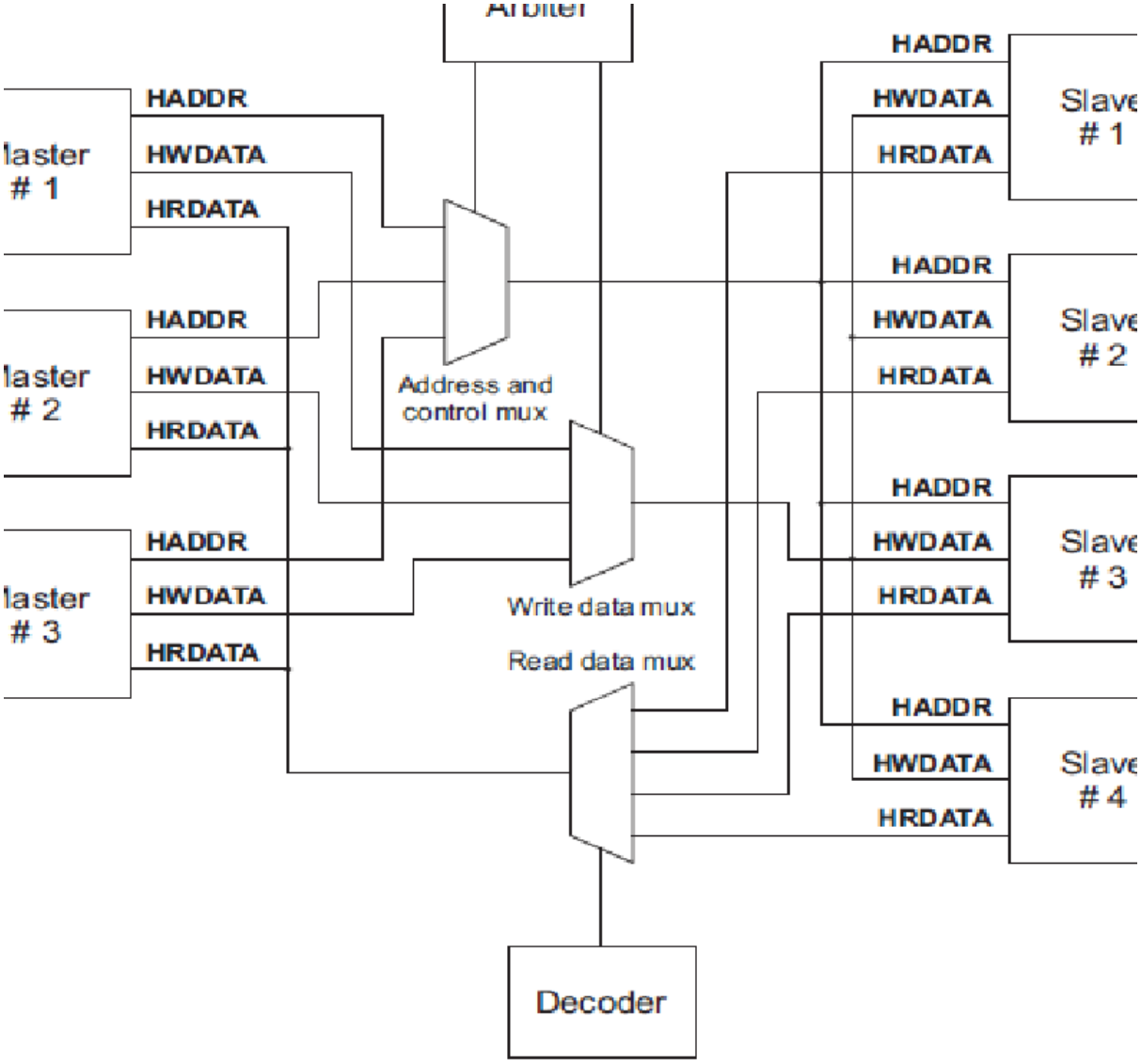

Fig: AHB Architecture

In the above figure, it is depicted that if any MASTER wants to send data, it has to first send request to Arbiter, that gives grant to any 1 of MASTERs and provides select signals to address and write data for MUX. According to this data from that particular MASTER will be reaching the SLAVE.

Decoder will generate the SLAVE Select and decodes the address sent by MASTER and generates the select signal for the SLAVE which is given to read data mux.

The following table gives the description of the signals used in AHB:

Signal Name	Description
HCLK(Bus clock)	This clock times all the bus transfers. All signal timings are related to the posedge.
HRESETn	Active LOW and used to reset the system and the bus.
HADDR[31:0]	Indicates the type of current transfer which may be Non-sequential, Sequential, Idle, Busy
HWRITE[1:0]	If 1, indicates as write transfer else read transfer
HSIZE[2:0]	Indicates size of write transfer i.e., 8 or 16 or 32-bit
HBURST[2:0]	Indicates if transfer forms part of a burst. 4, 8 or 16 beat bursts are supported and burst may be incrementing or wrapping.
HWDATA[31:0]	To transfer data from the MASTER to the bus SLAVES during write operations
HRDATA[31:0]	To transfer data from bus SLAVEs to bus MASTER during read operations
HREADY	If 1, it indicates transfer has finished on the bus and if 0, indicates as to extend a transfer
HRESP	Transfer response provides additional information on the status of a transfer, which has 4 different responses like OKAY, ERROR, RETRY, SPLIT
HTRANS[1:0]	It indicates the type of transfer ie., IDLE, BUSY, NON-SEQ, SEQUENTIAL

APB Bus Protocol

APB bus is a overhead which has only 4 control signals. Only 1 MASTER is allowed. It consists of 3 states such as: IDLE, SETUP, ENABLE SLAVE is non-responsive, the transfer takes 2 cycles Makes timing easy to design, as data is latched between 2 cycles.

The following table gives the description of the signals used in APB:

Signals	**Description**
PCLK	Times all the bus transfers. Both the low and high phase of PCLK is used to control signals
PRESETn	Active LOW Asynchronous reset
PENABLE	It is used to time all accesses on the peripheral bus. It indicates 2nd cycle of the APB transfer.
PWRITE	If it is 1, then it indicates APB write else if 0 it is APB read.
PRDATA[31:0]	It is driven by a peripheral bus bridge unit during write cycles.
PSELx	A signal from the secondary decoder, within the peripheral bus bridge unit, to each peripheral bus SLAVEx.
PADDR[31:0]	Address bus from MASTER to SLAVE, can be up to 32 bits
PWDATA[31:0]	Write data from MASTER to SLAVE, can be upto 32- bits
PREADY	It is used by the SLAVE to include wait states in the transfer i.e., whenever SLAVE is not ready to complete the transaction, it will request the MASTER for some time by de-asserting the PREADY

AHB2APB Bridge

AHB2APB bridge is an AHB SLAVE. It acts like an interface between high-speed AHB and low-power APB. The read and write transfers on AHB are converted into equivalent APB transfers and AHB is a pipelined protocol and APB is not pipelined. Therefore, for synchronization, this bridge adds WAIT states during the transfers to and from APB and AHB waits for APB.

AHB bus is MASTER, which drives all the signals except HRDATA and HREADY. AHB2APB bridge drives all other signals on APB bus. The APB bus is a SLAVE, which drives PRDATA.

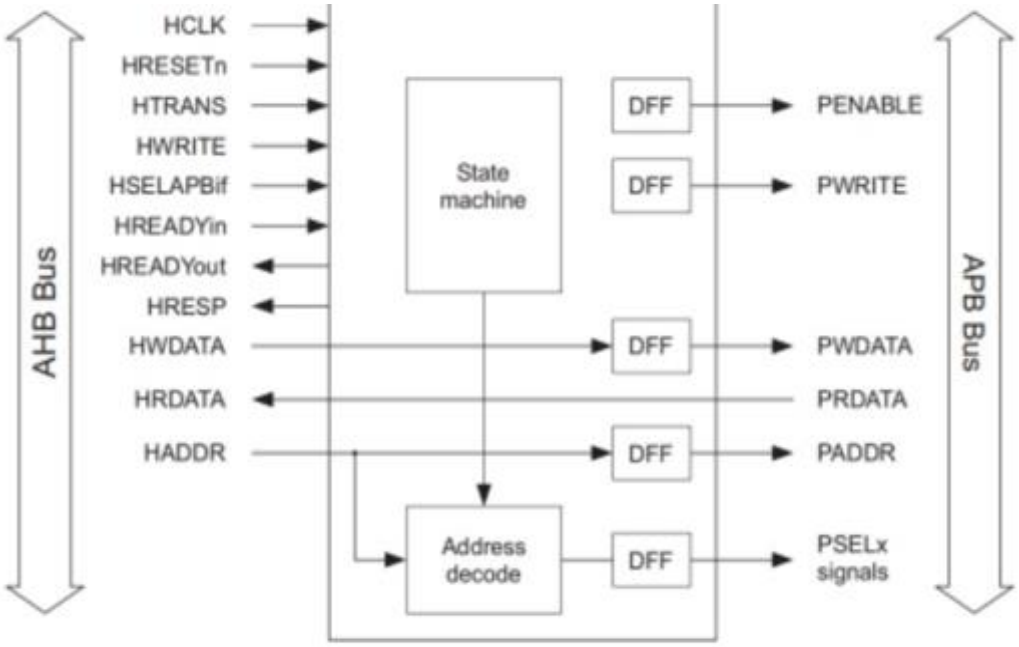

Fig. AHB2APB Block Diagram

The bridge uses Address Decoder for generating PSEL signal.
The flipflops are used because whenever SLAVE inserts WAIT state. MASTER will wait until APB is ready. At that time same address and control information should be held by the bridge. Hence, D-ffs are used.

Some of the bridge functions are as follows:

- Latches the address and holds it valid throughout the transfer.
- It decodes the address and generates a peripheral select-PSEL.
- It drives the data onto APB for write transfer.
- For read transfer, it drives APB data onto the system bus.
- Generates a timing strobe i.e., PENABLE for transfer

During write transfer, the bridge converts HRDATA into PWDATA and during read transfer, PRDATA is converted into HRDATA.

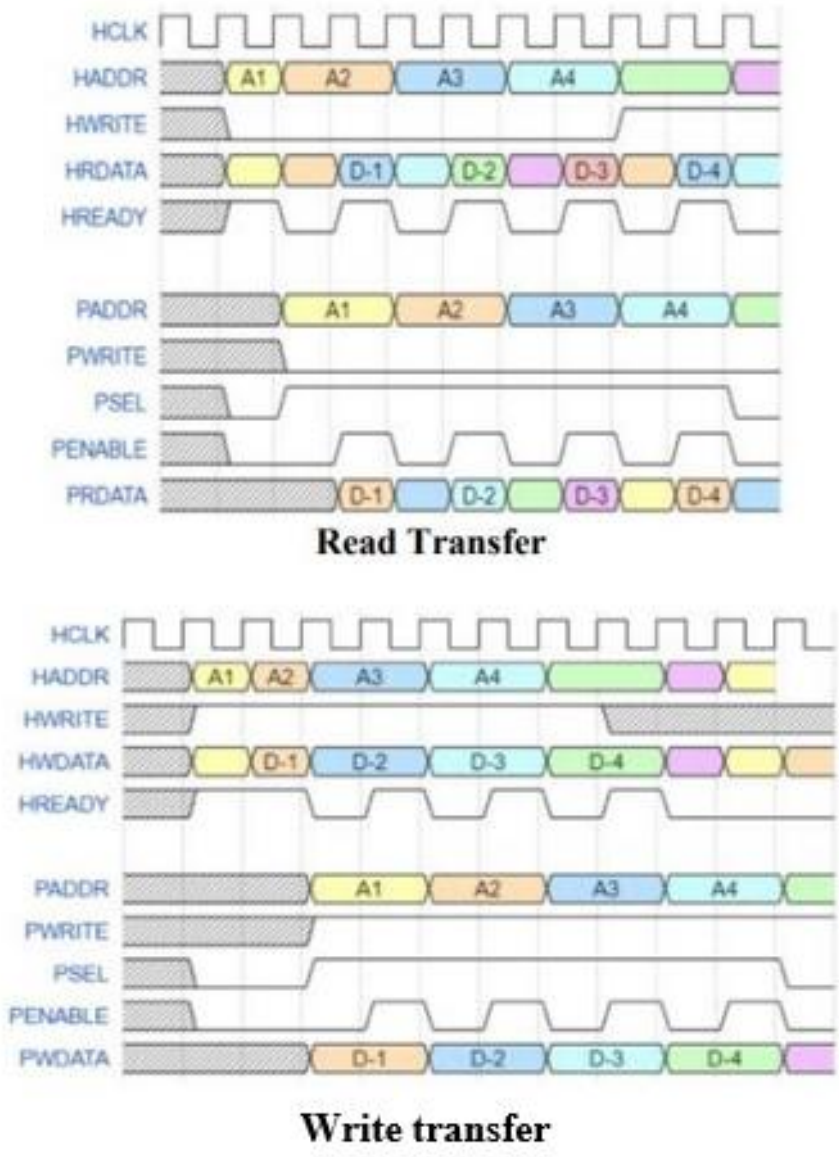

Fig: Read and Write Transfer Waveform

Write Transfer:

The AHB MASTER send 1st address and control information in the 1st cycle. The bridge samples immediately in next cycle and will be latched into local registers and in next cycle, MASTER send address and data for the previous address and again bridge samples in the immediate next posedge..

The currently sampled address is latched into local registers on APB bus i.e., PADDR and PWDATA.

Once PWRITE and PWDATA are on APB bus, it needs 2 cycles to complete the transfer. In the 2nd cycle only, the bridge will generate PENABLE and the transfer on APB is completed, after only bridge can sample new address and the previous data. Therefore, bridge can insert WAIT state by making HREADY = 0 and wait for APB peripheral to complete the transfer.

Read Transfer:

First MASTER send address which is sampled by bridge in next cycle. Once sampled, the same address and write is reflected as PADDR and PWRITE. Once, there are available on APB bus, it requires 2 cycles to complete the transfer. Therefore, in 2nd cycle, bridge generates the strobe signal PENABLE, were PRDATA is available on APB side until 1st transfer is completed. Here, bridge cannot sample next address, so insert WAIT state by making HREADY = 0. So that it can wait for previous transfer to complete. Therefore, the same address at that cycle will get extended and in next cycle, HREADY = 1, because in cycle transfer on APB side is completed.

Therefore, bridge can sample address in the next cycle and HREADY = 1.

Once address is sampled, that will be reflected as PADDR in APB bus and again needs 2 cycles to complete transfer. Therefore, bridge cannot sample the address immediately in the next cycle and insert WAIT state and wait for APB to complete transfer. Whenever PRDATA is reflected onto HRDATA, it wont take any time.

Verification IP Devolopment:

Verification of AHB2APB bridge involves components using SystemVerilog and UVM, which include Bus Functional Models (BFMs), monitors, checkers, scoreboards, assertions, coverage metrics, and testbenches.

- Master BFM(Master Bus Functional Model): This generated AHB transactions such as read/write and drives them to the bridge. This acts as an AHB Master and is responsible for stimulating the design under various conditions.
- Slave BFM(Slave Bus Functional Model): This acts as an APB slave, which responds to APB transactions from the AHB2APB Bridge.
- Monitor: This module captured all the transactions and signals between AHB and APB interfaces.
- Checker: This module is used to verify the correctness of the AHB2APB bridge operation. It ensures that AHB requests are correctly translated to APB transactions.
- Scoreborad: This is a critical verification component that compares the expected vs. observed transactions.
- SVA(System Verilog Assertions): These are used to enforce design rules and detect violations. These capture and verify the specific properties of AHB2APB bridge.
- Coverpoint and Covergroups: Functional coverage is crucial to measure how well the verification environment exercises the design. We define the coverpoints and covergroups tailored to the AHB2APB bridge
- Testbench: A comprehensive testbench is developed to simulate Multiple AHB Masters which generates transactions and multiple APB slaves which responds to those transactions and also the synchronization mechanisms.
- Tests: Here 2 types of tests can be developed that is directed tests and the random tests. Directed tests are focused on specific functionalities like Basic read/write operations, reset and initialization sequences. Random tests are used to constrain the random stimulus to explore the corner cases.