EasyChair Preprint

№ 7890

Design of AMBA Based AHB2APB Bridge Protocol

Peram Bhanu Prakash, Panta Nishith Reddy,
Maddireddy Sathish Reddy, Rachapalyam Vignesh Kumar and
G. Bharatha Sreeja

May 2, 2022

Design of AMBA Based AHB2APB Bridge Protocol

Peram Bhanu Prakash #[1],Panta Nishith Reddy #[2], Maddireddy Sathish Reddy#[3], Rachapalyam Vignesh Kumar#[4], G.Bharatha Sreeja#[5]
1,2,3,4, UG Student, ECE Dept, R.M.K College of Engineering and Technology, Chennai, India.
5 Assistant Professor, ECE Dept, R.M.KCollege of Engineering and Technology

***Abstract*—** The main abstract of this project is to design an Ip for a soc level purpose that converts AH B signals coming out from the AHB interface which is connected with high-performance devices like DMA, CPU to the APB signals which are low-performance devices like UVART, keypad connected to the APB interface. Creates a bridge between two interfaces AHB and APB..

KEYWORDS—
IP,AHB,APB,DMA,CPU,UVART,BRIDGE,interface

I.INTRODUCTION

The Advanced Microcontroller Bus Architecture (AMBA) has three types of signals.

- The advanced high-performance bus(AHB)
- The Advanced System Bus (ASB)
- The Advanced Peripheral Bus (APB)

we are only discussing AHB and APB.

Advanced High-performance Bus (AHB):

The AHB act as a high-performance bus and fast transmission. AHB supports the efficient connection of processors, off-chip external memory and on-chip memories interfaces with low-power peripheral macrocell functions, which sits above the APB and implements the features required for high clock frequency and high-performance systems including:

- ✓ Pipeline data operation
- ✓ split transactions
- ✓ burst transfers
- ✓ single cycle bus master handover
- ✓ single clock edge operation
- ✓ non-tristate implementation
- ✓ wider data bus configurations (64/128 bits).

Advanced Peripheral Bus (APB):

APB (Advanced Peripheral Bus) is one of the component of the AMBA bus architecture. APB is low-performance and low consumption bandwidth bus used to connect the peripherals like Keypad, Timer, UART and other peripheral devices to the bus architecture. APB can be used in conjunction with either version of the system bus.

- ✓ Low power consumption
- ✓ Small bandwidth
- ✓ Reduce interface complexity
- ✓ Pipelined operation is not supported by APB, so it makes communication with ASB or AHB.

Structure of AMBA microcontroller:

An AMBA-based microcontroller typically consists of a high-performance system (AHB or ASB), able to sustain the external memory bandwidth, on-chip memory, on which the CPU and other Direct Memory Access (DMA) devices reside. This bus provides a bandwidth interface between the elements that are involved in the transfers. Also located on the high-performance bus is a bridge to the lower bandwidth APB, where most of the peripheral devices in the system are located (see Figure 1-1).

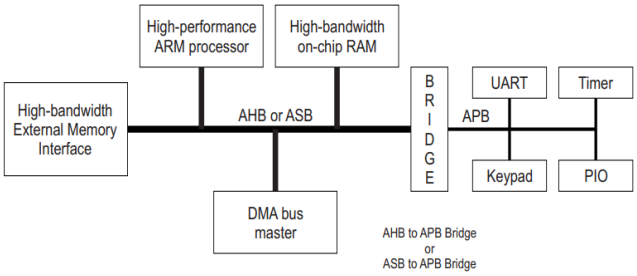

Fig1 typical AMBA bridge

An AMBA-based microcontroller typically consists of a high-performance system (AHB or ASB), able to sustain the external memory bandwidth, on-chip memory, on which the CPU and other Direct Memory Access (DMA) devices reside. This bus

AMBA signal prefix denotations:

provides a bandwidth interface between the elements that are involved in the transfers..

AMBA AHB signals list:

H illustrates an AHB signal. For example, HCLK is the signal used to indicate that This clock times all bus transfers. It is active HIGH
All signals are prefixed with the letter H, ensuring that the AHB signals are differentiated from other similarly named signals in a system design.

Table 1 AHB Signals and Arbitration Signal.

Name	source	description
HCLK Bus clock	Clock source	This clock times all bus transfers
HRESETn Reset	Reset controller	It is used to reset the system and the bus.
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
HWRITE Transfer direct	Master	l Indicates a write transfer and read transfers.
HWRITE Transfer direct	Master	Indicates the size of the transfer
HBURST[2:0] Burst type	Master	Indicates the transfer forms part of a burst
HWDATA[31:0] Write data bus	Master	used to transfer data from the master to the bus slaves
HSELx Slave select	Decoder	Indicates the current transfer is intended for the slave
HRDATA[31:0] Read data bus	Slave	Transfer data from slave to master bus
HREADY Transfer done	Slave	Indicates that a transfer has finished
HRESP[1:0] Transfer response	Slave	Provides extra info about transfer
Arbitration signals		
HBUSREQx Bus request	Master	This indicates that the bus master requires the bus.
HLOCKx Locked transfers	Master	This indicates that the master requires locked access to the bus
HGRANTx Bus grant	Arbiter	Indicates the highest priority master
HMASTER[3:0] Master number	Arbiter	Indicates which bus is performing the transfer.
HMASTLOCK Locked sequence	Arbiter	This indicates that the current master is performing a locked sequence of transfers
HSPLITx[15:0] Split completion request	Slave (SPLIT-capable)	Used by a slave to indicate to the arbiter

AMBA APB signals list:

P indicates the AMBA APB signals. Some APB signals, such as the reset, may be connected directly to the system bus equivalent signal.

Table 2 APB signal

Name	Description
PCLK Bus clock	PCLK is used to time all transfers
PRESETn APB reset	The APB bus reset signal
PADDR[31:0] APB address bus	This is the APB address bus
PSELx APB select	A signal from the secondary decoder, l indicates that the slave device is selected and a data transfer is required.
PENAL APB strobe	This strobe signal is used to time all accesses on the peripheral bus.
PWRITE APB transfer direction	Indicates the read and write access
PRDATA APB read data bus	The read data bus is driven by the selected slave during reading cycles
PRDATA APB read data bus	The write data bus is driven by the peripheral bus bridge unit during write cycles

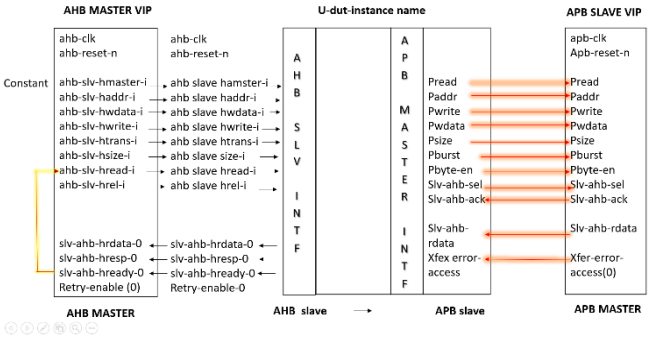

Fig 2 architecture of Ahb2Apb bridge

AHB Slave: AHB master commences write as well as read operations by coming up with control and address signals. Only once the bus can be used by a single bus master**.**

Bridge FSM: It is a sequential type machine that defines each step in the sequence. In this project, State machine control:

1. AHB transaction with HREADYout signal

2.Generation of each product signal of APB.

Then APB location there is not a single peripheral get preferred.

APB Interface: Slave answers to both operations read and write in the allotted span of address. The slave signal returns to the master which is active and that master is acknowledged by a response like success, failure, and waiting of the signals(data, address) collected from the bridge.

Top Module: It is the leading chunk which is not small compared to others. To connect various elements present in the top, all signals behave as wires and connect all modules within the chief top chunk. This top factor has AHB slave, AHB2APB bridge element & APB interface.

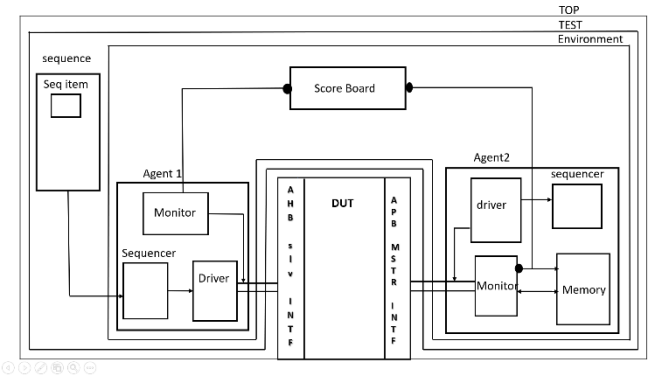

fig 3 Block diagram

II.PROPOSED WORK

Testbench Components and Architecture

The following are the required and important components of UVM based verification.

Design Test:

This gives the design that is intended to be proved. This is usually an RTL us in any of the HDL (System Verilog, VHDL, and Verilog). This completely describes the functionality of the design as well the features to be verified.

Interface:

The interface serves as the actual link between the design-under-verification and the verification environment. The interface describes the pin-level description of the DUT. An interface is a bundle of nets or wires**.**

Virtual Interfaces:

Virtual interfaces provide a mechanism for separating abstract models from the actual signals of the design. A virtual interface gives access to the subprogram to operate in different places of the design.

Transaction (class uvm_sequence_item):

It is an object that represents communication abstraction such as a bus cycle, data packet, or handshake. A transaction class contains user-defined properties of the specific protocol, and user-defined specific methods to perform a few operations like print, pack, unpack, copy, compare, and record those members. Typically transactions are generated by a sequence and these are passed to a driver or collected by a monitor and passed to zero or more subscribers. Between components, transactions are passed using ports and exports.

Sequence (class uvm_sequence):

It is an object that generates basic transactions or starts other sequences in the verification methodology. A sequence generates a sequence item which is nothing but stimulus scenarios that are passed to the driver through a sequencer. In a sequence class that contains a user-defined task body that is called when the sequence is started. The task body does the work of the sequence. A sequence that directly generates transactions must always execute on a sequencer. A sequence indicates its readiness to generate a transaction by using the calling method start_item and delivers the transaction by calling method finish_item. A sequence may retrieve a response to a transaction by calling the method get_response. A sequence may be synchronized with other parts of the verification environment using events.

Sequencer:

Multiple sequences may be trying to send sequence items ‟s to the driver so this component coordinates and arbitrates between transactions generated by sequences.

Driver:

The driver is defined by extending uvm_driver. The driver takes the transactions from the sequencer using seq_item_port and sends the transactions to the DUT as per the interface signal specifications. Then using uvm_analysis_port in the monitor transactions will be sent to the scoreboard. Tasks are used here after to reset DUT. In the environment class, an instance of the driver class is created and a sequencer is connected to it. The following figure 4 shows the connection between the uvm_sequencer and uvm-driver and the connection between uvm_driver to uvm_scoreboard.

Figure 4: Connection between Driver and Sequencer

Monitor:

The monitor is used to observe specific DUT activity through an interface and convert it into a higher-level transaction. Monitors collect and perform protocol checking. The monitor does the following:

- Extracts signal information from an interface as per the protocol and translates the information into a transaction & this is made available to other components.
- Passes the information collected from the DUT to coverage collectors using analysis ports/exports.

It is implemented by extending the uvm monitor class and an instance is created in the environment for with DUT signals.

Env (class uvm_env):

For all the components Environment is the top-level container. Inside the environment, all the agents are instantiated and configured. The top-level environment is instantiated from the test.

Testcases:

Class is instantiated inside this Test class. The uvm_test class defines the test case for the testbench for the DUT and as specified in the test. Each test is derived from the uvm_test. The virtual interfaces declared in the verification environment are pointed to the physical interfaces which are declared in the top module. Virtual interfaces pointing to the top module are made to point to the physical interface in the test case.

Scoreboard (class uvm_scoreboard):

A component that receives the transactions from multiple active/passive agents and typically performs checking of DUT functionality using cover groups and assertions and helps to collect functional coverage information. A scoreboard may or may not incorporate a reference / golden model of DUT functionality.

Top Module:

System Verilog interface instance is created in this module. The clock generator is implemented here. run_test method is called. The implementation will be discussed in further sections.

Agent (class uvm_agent):

This is a component (depending on active/passive type) that contains one sequencer, one driver, and one monitor and which also senses and drives the signals of the SystemVerilog interface.

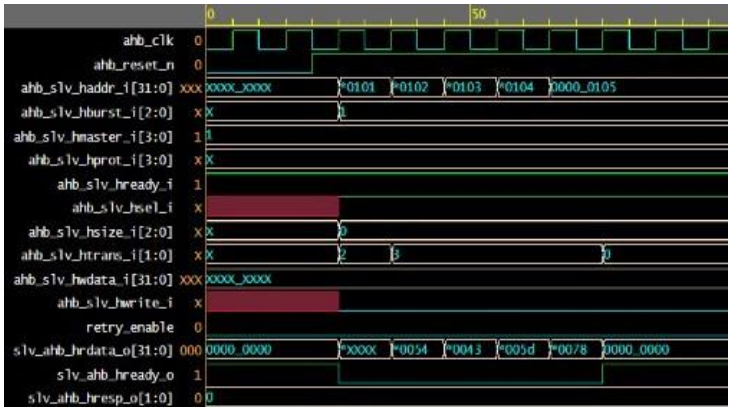

Fig :5 AHB Signals

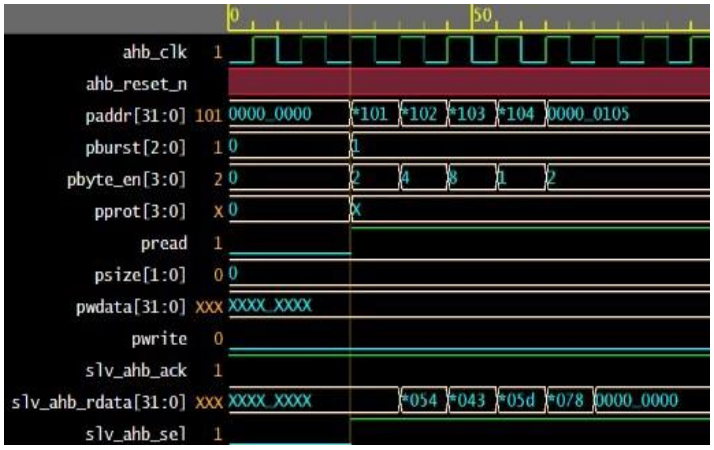

Fig:6 APB Signals

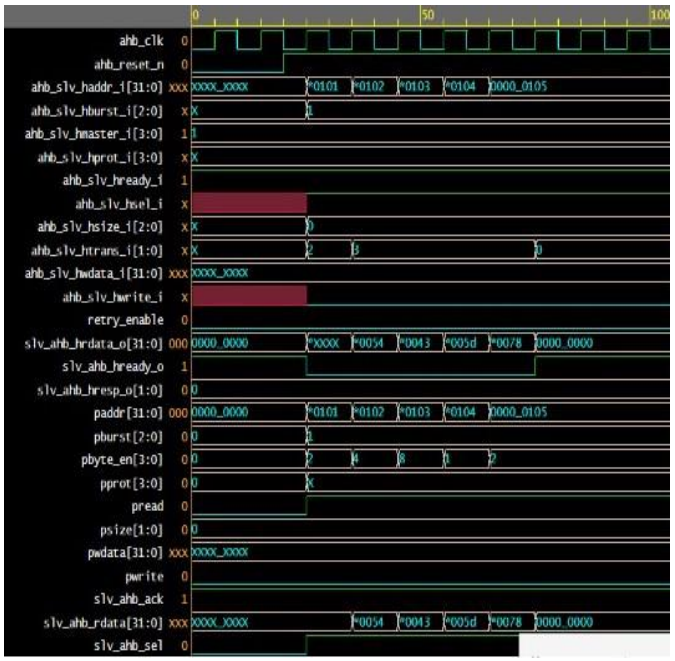

Fig:7 AHB2APB Signals

III.CONCLUSION AND FUTURE WORK

AHB2APB bridge design is implemented in system Verilog HDL for Read transfer, Write transfer, Read burst transfer, Write burst transfer, back to back read and write transfer and all these designs are verified by simulating Xilinx ISE. By adding the timeout concept, data loss can be overcome and the design will become more generic.

REFERENCES:

1. M. kiran Kumar, Amrita Sajja, Dr, Fazal Noorbasha, “Design and FPGA Implementation of AMBA APB bridge with clock skew minimization Technique” (IOSR-JVSP),Vol.7,Issue.3,June 2017.
2. Miss Pooja, Kawale, , “Design of AMBA based AHB2APB bridge” IJSRD, Vol.4, Issue.8, Nov.2016.
3. N.G.N. Prasad, “ Development and Verification of AHB2APB Bridge Protocol using UVM Technique”. IJSRR, Vol.6, Issue.12, 2017.
4. Sujata Mallappa chajagauda, Abdullah Gubbi.,” Using Verilog and System Verilog design and verify communication bridge between APB and I2C protocol.”. IJSTE ,Vol.3 , Issue. 1, 2016.
5. Aparna charade, Jayshree Sengupta, “VLSI Design of AMBA based AHB2APB bridge” (IJCSNS) VOL.9 No.3 June 2018.
6. Sowmya Aithal, Dr. J. S. Baligar, Guruprasad S. P. “FPGA Implementation of AHB to APB Protocol” (IJSR)-Volume 5 Issue 5, May 2016.
7. Prof. ravi Mohan Sairam ,Prof. Sumit Sharma, Miss Geeta Pal, “FSM abd handshaking based AHB2APB bridge for high speed system”. IJERT, Volume.2 , Feb 2019.
8. Ankem Kiran1, V Thrimurthulu, “Verification Of Amba Ahb2apb Bridge Using Universal Verification Methodology (UVM), IJITE, Vol.04 Issue-12, ISSN: 2321-1776.
9. Clifford E. Cummings, "Coding And Scripting Techniques For FSM Designs With Synthesis-Optimized, Glitch-Free Outputs," SNUG (Synopsys Users Group Boston, MA 2000) Proceedings, September 2000.
10. Sowmya Aithal1, Dr. J. S. Baligar, Guruprasad S. P. “FPGA Implementation of AHB to APB Protocol” IJSR-Volume 5 Issue 5, May 2016.
11. Samir Palnitkar, “ Verilog HDL: A guide to digital design and synthesis(2nd Edition), Pearson, 2008.
12. Bergeron, “ Writing testbench using system Verilog”,Springer,2009.
13. AHB to APB Bridge (AHB2APB) Technical Data Sheet Part Number: T-CS-PR-0005-100 Document Number: IIPA01-0106-USR Rev 05 March 2007.
14. Mukunthan, J., et al. "Design And Implementation Of Amba Apb Protocol." *IOP Conference Series: Materials Science and Engineering*. Vol. 1084. No. 1. IOP Publishing, 2021.
15. Yuan, Conggui, et al. "An Easy-to-Integrate IP Design of AHB Slave Bus Interface for the Security Chip of IoT." *Mathematical Problems in Engineering* 2021 (2021).
16. HK, Meghana Jain, and Punith Kumar. "Verification of Advanced Peripheral Bus Protocol (APB V2. 0)." (2021).
17. Giri, Davide, et al. "Accelerator Integration for Open-Source SoC Design." *IEEE Micro* 41.4 (2021): 8-14.