VLSI DESIGN AND VERIFICATION OF AMBA BASED AHB2APB BRIDGE

Feroz Ahmed Choudhary[1], Dr. Rajashekhar C. Biradar[2]
School of ECE, REVA University, Bangalore, India-560064
ferozer3@gmail.com, dir.ece@reva.edu.in

Abstract – ***The Advanced Microcontroller Bus Architecture (AMBA) is an on-chip bus architecture used to Design high performance embedded microcontrollers and strengthen the reusability of IP core and widely used interconnection standard for system on chip (SOC). AMBA AHB (advanced high performance bus) is the high-performance bus means higher bandwidth or high clock frequency system modules. AMBA APB (advanced peripheral bus) as the name suggest used to connect peripheral to the architecture, peripherals like UART, Timer, keypad, PIO etc. this are part of low performance bus and it is optimized for low power consumption and interface reduced complexity to support peripheral functions. In this the functions of the AHB2APB Bridge to make the signals compatible with the high performance bus i.e. AHB with low performance bus i.e. APB, to do so we have to write the DUT code in Verilog and all other test case code in system Verilog, further have verified all the functions of bridge protocol using QuestaSim tool. The code coverage and functional coverage and functional verification of the Bridge RTL design is 97% covered by using QuestaSim.***

Keywords: AMBA, AHB, APB, AHB2APB Bridge, QuestaSim, Arbiter, DUT, Verilog, SVM, Coverage.

1.INTRODUCTION

AMBA architecture is an open standard, on chip interconnects, specification for management and connection of functional block in system on chip (SOC) design. As we know that in computer we use PCI (peripheral component interconnect) when we want to connect hard disk or RAM we connect it externally through the PCI slot to communicate with computer similarly in mobile smart phones when we see the specifications on top of same single real estate chip there are multiple processors, high speed RAM & UART, keypad and other slow peripheral to connect all this the group of wire standard which we use is called Advanced microcontroller bus architecture.

If we look at fig 1.1 the RHS side there is only one master device which is bridge, all transaction etc. are initiated by the bridge, bridge obviously cannot initiate transfer on its own, so one of the device on RHS side i.e. AHB side initiate the transfer through the bridge. Once the bridge initiates the transfer this (RHS) are slave devices and they communicate to the master through the slave at a given point of time the APB bus used to communicate between bridge and any one of the device (UART, keypad, POI etc.). Here master and slave can be termed in other words too as intel they called is as north bridge and south bridge, north bridge specifies master and south bridge as slave.

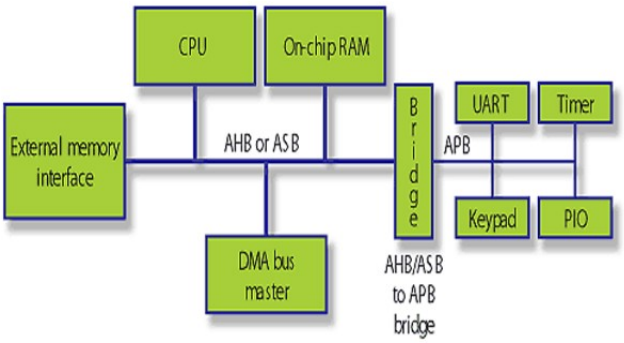


Fig 1 AMBA Based Architecture

multiplexer interconnection scheme is used in the AHB Bus protocol, in this scheme all bus masters drive the address and control signal indicating the transfer to perform and the arbiter arbitrate which master has its control signals and address routed to all of the slaves. The central decoder is also required to control the data read and signal response multiplexer, which choose the appropriate signals from the slaves that is concerned in the transfer. Any peripherals are connected using the APB Bus which are low bandwidth and do not need high performance of a pipelined bus interface.

The BUS Communication may be done in different ways.

(A) Transfer type: - Indicates type of the current transfer, which can be NON-SEQUENTIAL, SEQUENTIAL, IDLE or BUSY.

(B) Transfer direction: - When write HIGH this signal indicates a write transfer and when write LOW a read transfer.

(C) Transfer size: - Indicates this size of the transfer, which is typically byte (8-bit), half word (16-bit) or word (32-bit). The protocol allows larger transfer sizes up to a maximum of 1024 bits.

(D) Burst type: - Indicates if the transfer forms the part of a burst. Four and eight and sixteen beat bursts are supported and the burst may be either incrementing or wrapping.

 ISBN 978-93-81288-18-4

2. AHB2APB BRIDGE

General architecture of AHB2APB Bridge consists of five main building blocks:

- AHB Master
- AHB2APB Bridge
- AHB Interface
- APB FSM Controller
- APB Interface

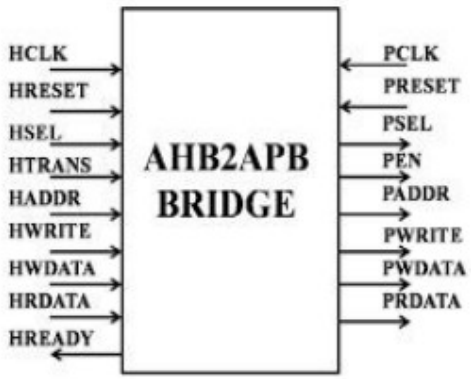


Fig 2 Architecture of AHB2APB Bridge

1. AHB Master: The read and write operations are initiated by the bus master by providing an address and control information, one bus master is allowed to actively use the bus at any one time.

2. AHB Arbiter: one bus master is allowed to initiate data transfer and this is ensured by the arbiter. Even though the arbitration protocol is fixed, any arbitration algorithm, such as fair access and highest priority can be implemented depending on the application requirements.

3. AHB decoder: The AHB decoder is used to decode address of each transfer and provide a select signal for the slave that is involved transfer single centralized decoder is required in all AHB implementations.

4. APB Interface: Read and a write operation are responded by a bus slave within a given address-space range. The bus slave signals back to the active master the success, failure and waiting of the data transfer address and data that received from the bridge correctly used for data transaction from bridge to this module and vice versa depending whether it is a write and a read operation. These modules contain block for P_CLK generation, which is divided to AHB2APB bridge module, and the P_CLK generated obviously used in this module also.

5. AHB2APB Bridge: Out of all modules present, this module is larger and simple. All the signals are taken as wire to interconnect the various modules present in the top module. In the module, all the three modules namely

- AHB Master
- AHB2APB bridge
- APB interface These modules are all instantiated used Positional assignments which is again simple compared naming assignment which is little tedious.

These modules are all instantiated used Positional assignments which is again simple compared naming assignment which is little tedious.

6. FSM Controller:

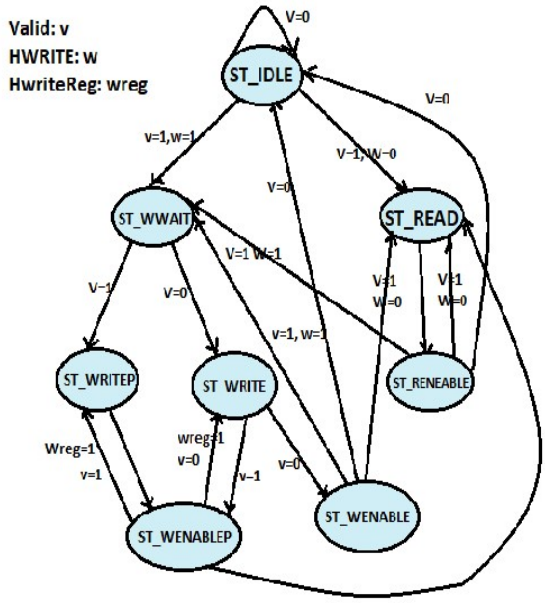


Fig 3. State Machine

3. APB AND ITS STATE DIAGRAM

i. APB State diagram

APB is used to interface to any peripherals which are of low bandwidth and do not required high performance of pipeline bus interface.

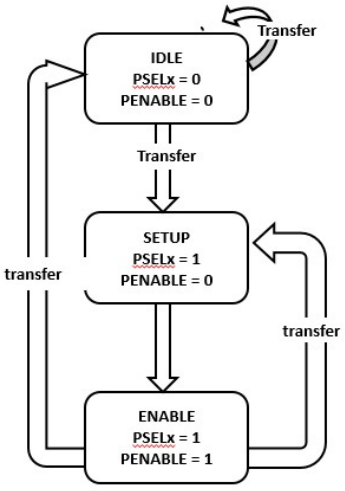


Fig 4. APB State Diagram

 ISBN 978-93-81288-18-4

Abstract state machine is defined in 2 signals i.e. Pselx and Penable. Here in Pselx 'x' stands for the peripheral device which is selected. If we see in fig 1 the bridge has to select between the four devices, so there is independent line for the selection. For example, Psel1 might be selecting UART, Psel2 might be selecting Timer and so on.

1. When Pselx is 0 & Penable is also 0, then it is an IDLE State.
2. When it wants to transfer it will raise Pselx is 1, that's what bridge is doing.
3. Then it will raise Penable that will initiate one data, it can be on and data can be transferred in a loop. When Pselx is 0 and Penable is 0 it will go back to idle state. While its doing several transfer Psel remains high and Penable toggles to 0 1 0 1... for each item of data i.e. transferred.

ii. APB Bridge Interface

- APB bridge is the only bus master on the AMBA APB.
- APB bridge is also a slave on the higher level system bus.

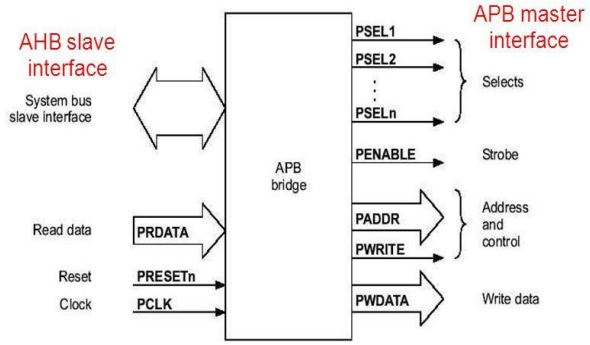


Fig 5. APB Bridge Interface

iii. APB Slave Interface

- APB Slave have a simple, yet flexible interface.
- Exact implementation of the interface will be dependent on the design style employed and many different options are possible.

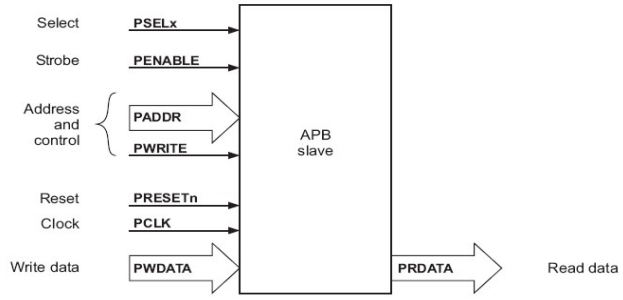


Fig 6. APB Slave Interface

- This is one peripheral device, it receives Psel for one device and Penable will go to all peripheral devices.
- It receives the address and the write signal.
- There is PRESETn which is reset pin here which can be used to reset the slave. This is driven by the controller and not by the APB.
- Pclk is the clock for the slave and there is write data.

iv. APB Write & Read Transfer Timing Diagram

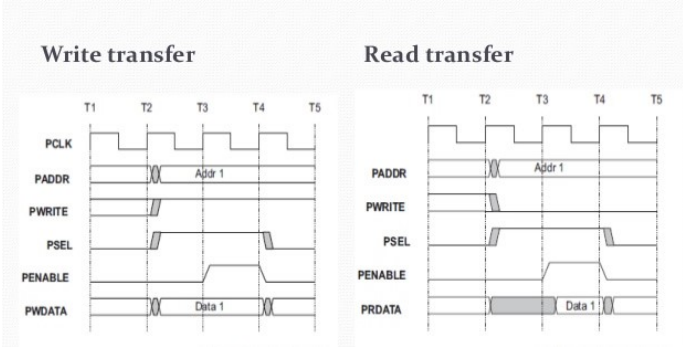


Fig. 7. APB Read and write timing diagram

APB Write Transfer

- First the master will raise Psel and at the same time it is floating the address.
- Slave observes the Psel high so it also gets the address this is for write transfer data; this also floats the data.
- When slave gets all this 3 it gets time upto next clock pulse to latch in the data (to save the data).
- And in the next clock pulse Penable is raised.

APB Write Transfer

- When it raises the Psel at that time it can float the address from where it wants to read It but it doesn't have the data, data is going to come from slave.
- When slave sees Psel is high for it, it uses the address and reads the corresponding address and floats the data in the PR data line, once this data is available in the next cycle master raises Penable and reads the data and then lower it

4. SYSTEM VERILOG WORKING METHODOLOGY

i. Architecture

The below show Fig-4 is for the System Verification Architecture as can be seen there are various blocks which are need to coding in the tool, but the question is why Verilog only can be used and why in what way does System Verilog have advantage over Verilog. DUT Block is the Device under test i.e. the top module for which coding is done by using Verilog. If all the remaining Blocks are coded via Verilog then we have to instantiate coding by defining module name for each block, module to module communication is very hectic thus we don't prefer Verilog for coding of other blocks instead prefer System Verilog. The main reason is System Verilog includes OOPs concepts thus defining each blocks codes in a

 ISBN 978-93-81288-18-4

class format provides easy way of coding as compare to Verilog.
The following Figure can be divided as two parts for master and slave configuration, the left hand side is for master and the right hand side is for the slave. Depending upon the no of slave and master the figure can be modified for different design aspects.

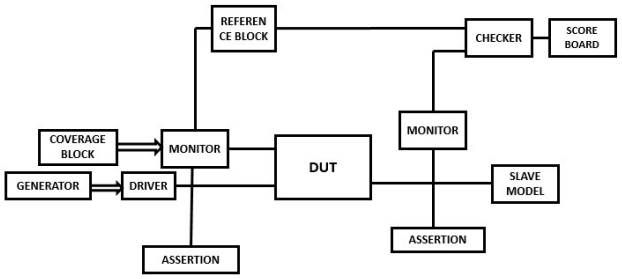


Fig 8. System Verilog Architecture

We can briefly define each block role independently as follows

(i) Generator Block: Responsible for generating all the signals required for all scenarios example clock & reset signals etc.
(ii)Driver/Bus function model (BFM): Routing Signals generated from the generator.
(iii)Monitor: To check signals and has inputs from coverage & assertion blocks.
(iv)Coverage Block: checking the given input to the DUT whether it's complete or not.
(v)Assertion Block: checking at the interface as per the protocol or not i.e. indicates violation of protocols.
(vi)Reference: kind of having desire results.
(vii)Checker: Compare the outputs from the reference and DUT block.
(viii)Scoreboard: Displays the result of Checker.

ii. Verification Steps

(i) Features listing down.
(ii) Scenario listing down.
(iii) Test plan development.
(iv) Functional Coverage Point listing down.
(v) Testbench architecture definition.
(vi) Testbench component coding.
(vii) Sanity test case development.
(viii) Sanity test case bring up.
(ix) other test cases.
(x) Setting up regression.
(xi) Running regression and debugging regression results.
(xii) Generating coverage results.
(xiii) Analyze coverage results.
(xiv) Closing functional Coverage.

iii. Verification Tools

There are many companies which provides simulating advanced verification tools namely QuestaSim, ModelSim, Xilinx, Cadence etc.
Usually the tools work in 3 steps which includes:

(i) Compilation.
(ii) Elaboration.
(iii) Simulation.
Tool like QuestaSim has an inbuilt writing notepad otherwise the code written in a separate notepad can be linked to the tools. For every Design prospective we need to create a new project in the tool and make sure all the required libraries linked to the directories are included for the same project.

5. RESULTS AND DISCUSSION

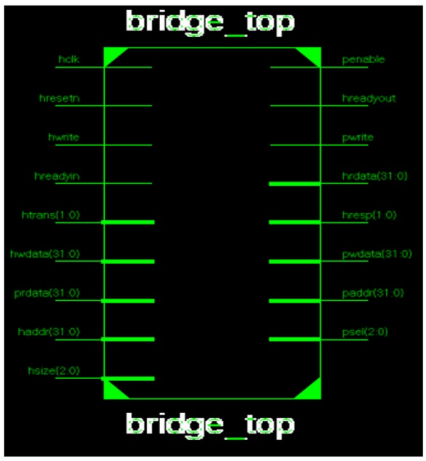
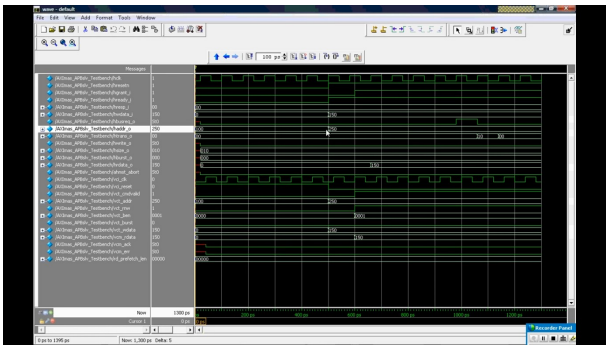


Fig 9. Bridge top Schematic Diagram from Synthesis

The Bridge is carried out for the functional verification using the SVM techniques for both the read as well as write operation. The functional verification of the RTL design is the Bridge is yields the complete code and functional coverage. Functional Verification of the Bridge Using SVM as verification methodology plays important phase in the circuit design. The read operation of the Bridge for the RTL design and the verification methodologies are carried out using Questasim. The design is carried out using in Verilog and the verification is carried out in SVM. The Bridge is set up as DUT the functional verification and the code coverage is obtained for 97%.

6. READINGS AND COVERAGE REPORT



 ISBN 978-93-81288-18-4

Fig 10. Result and coverage Report

7. CONCLUSION

This paper gives an outline of the Advanced microcontroller bus architecture and explains the Advance Peripheral Bus bus in detail and the working of bridge. The design has taken care of balance between and speed and area overhead. The read and write operation is accomplished with zero wait states from the external ROM and the write operation with zero states to the external RAM. The Advance Peripheral Bus is designed using the Verilog Hardware Description Language according to the specification and is verified using QuestaSim.

REFERENCES

[1] "Design and Implementation of HighPerformance Master/Slave Memory Controller with Microcontroller Bus Architecture" Shashisekhar Ramagundam1, Sunil R.Das1, 2, Scott Morton1, Satyendra N. Biswas4, Voicu Groza2, Mansour H. Assaf3, and Emil M. Petriu2 1Department of Computer Science, College of Arts and Sciences, Troy University, Montgomery, AL 36103, USA School of Information Technology and Engineering, Faculty of Engineering, University of Ottawa, Ottawa, ON K1N 6N5, Canada 3School of Engineering and Physics, University of the South Pacific, Suva 19128, Fiji 4School of Engineering and Technology, Kaziranga University, Jorhat 785006, India.

[2] Y. Hu and B. Yang, "Building an AMBA AHB compliant memory controller", Proceedings of the Third International Conference on Measuring Technology and Mechatronics Automation, Vol. 01, 2011, pp. 658–661.

[3] S. Rao and A. S. Phadke, "Implementation of AMBA compliant memory controller on a FPGA", International Journal of Emerging Trends in Electrical and Electronics, Vol. 2, April 2013, pp. 20–23.

[4] AMBA Specification (Rev 2.0), ARM Inc., 1999.

[5] AHB Example – AMBA System, Technical Reference Manual, ARM Inc., 1999.

[6] Sudeep Pasricha, "On-Chip Communication Architecture Synthesis for Multi-Processor Systems-on-Chip", University of California, 2008.

[7] "PrimeCell Synchronous Static Memory Controller", Technical Reference Manual, ARM Inc, 2001-2005.

[8] URL:http://wwwmicro.deis.unibo.it/~magagni /amba99.pdf

[9] ARM, "AMBA Specification Overview", available at http://www.arm.com/.

[10] ARM, "AMBA Specification (Rev 2.0)", available at http://www.arm.com.

[11] URL:http://www.differencebetween.net/techno logy/difference-between-ahb-and-apb [12]. Samir Palnitkar, "Verilog HDL: A guide to Digital Design and Synthesis (2nd Edition), Pearson, 2008. [13]. URL:http://www.testbench.com.

 ISBN 978-93-81288-18-4