MODELING OF AMBA AHB BUS USING VERILOG HDL

Dr. Khaja Mujeebuddin Quadry ,Dr. M Khaleelullah Khan, Mr. Mohammed Abdul Raqeeb ,Dr. S.P. Venu Madhava Rao

Professor ECE,Vignan Institute of Technology and Science, Telangana, India.
mujeebqd@yahoo.com
Associate Professor ECE,Vignan Institute of Technology and Science, Telangana, India.
mkkcr9@gmail.com
Assistant Professor ECE,Vignan Institute of Technology and Science, Telangana, India.
abdul.raqeeb1993@gmail.com
Professor, Department of ECE, Maturi Venkata Subbarao Engineering College (MVSREC), Telangana, India.
spvmrao_ece@mvsrec.edu.in

Abstract: *This paper describes modeling of AMBA AHB protocol using Verilog HDL. The Advanced High Speed Bus (AHB) is part of the Advanced Micro Controller Bus Architecture protocol family. The AMBA AHB is for high clock frequency, high-performance system modules. The AHB acts as the high-performance system backbone bus. Efficient interfacing of processors with on-chip memories, off-chip external memory, and low-power peripheral macro cell functions is achieved with the support of AHB [1-3]. Integration of AHB peripherals into any design flow is done easily as all the signal transitions are related to the rising edge of the clock. Every transfer takes at least two cycles. The AHB can be interfaced with the AMBA APB-Line (Advanced Peripheral BUS Line) and AMBA-AXI (Advanced Extensible interface). The programmable control registers of peripheral devices are accessed through AHB. Once the modeling done with Verilog HDL the verification of the functionality of the design is carried out by the applying the sequence of test cases and compared with expected results.*

Index Terms: *AHB, APB, AXI, Bus function model*

DOI:10.48047/ecb/2023.12.8.773

I.INTRODUCTION

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

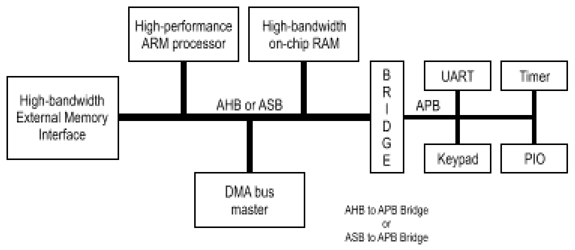
High-performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth External Memory Interface
AHB or ASB
DMA bus master
BRIDGE
APB
UART
Timer
Keypad
PIO
AHB to APB Bridge or ASB to APB Bridge

Figure 1.1 A typical AMBA system

II. OVERVIEW OF AMBA AHB OPERATION

AHB is a new generation of AMBA bus, which is intended to address the requirements of high-performance synthesizable designs. AMBA AHB is a new level of bus which sits above the APB and implements the features required for high-performance, high clock frequency systems [5].

A. Bus interconnection

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.

Figure 1.2 illustrates the structure required to implement an AMBA AHB design with three masters and four slaves.

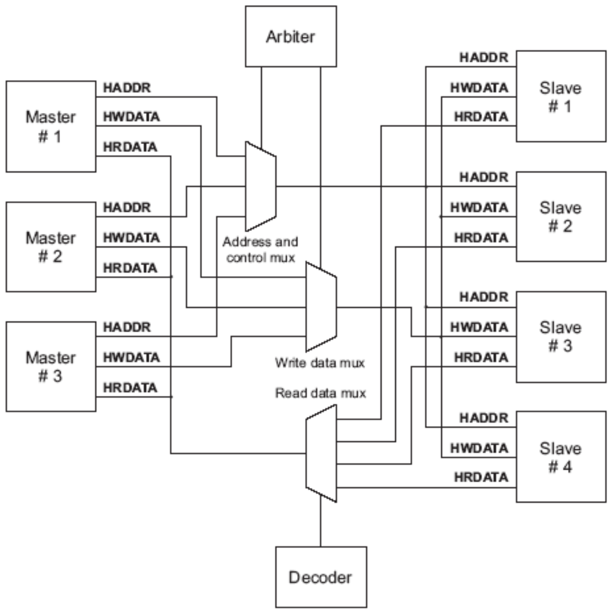

Figure 1.2 AMBA AHB design with three masters and four slaves.

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. Two different forms of burst transfers are allowed:

• Incrementing bursts, which do not wrap at address boundaries

• wrapping bursts, which wrap at particular address boundaries. 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 during a transfer the slave shows the status using the response signals, **HRESP [1:0]**: OKAY The OKAY response is used to indicate that the transfer is progressing normally and when **HREADY** goes HIGH this shows the transfer has completed successfully. The ERROR response indicates that a transfer error has occurred and the transfer has been unsuccessful. RETRY and SPLIT Both the RETRY and SPLIT transfer responses indicate that the transfer cannot complete immediately, but the bus master should continue to attempt the transfer.

In normal operation a master is allowed to complete all the transfers in a particular burst before the arbiter grants another master access to the bus. However, in order to avoid excessive arbitration latencies it is possible for the arbiter to break up a burst and in such cases the master must re-arbitrate for the bus in order to complete the remaining transfers in the burst.

B. Basic transfer

An AHB transfer consists of two distinct sections:

• The address phase, which lasts only a single cycle.

• The data phase, which may require several cycles. This is achieved using the **HREADY** signal.

Figure 1.3 shows the simplest transfer, one with no wait states. In a simple transfer with no wait states:

• The master drives the address and control signals onto the bus after the rising edge of **HCLK**.

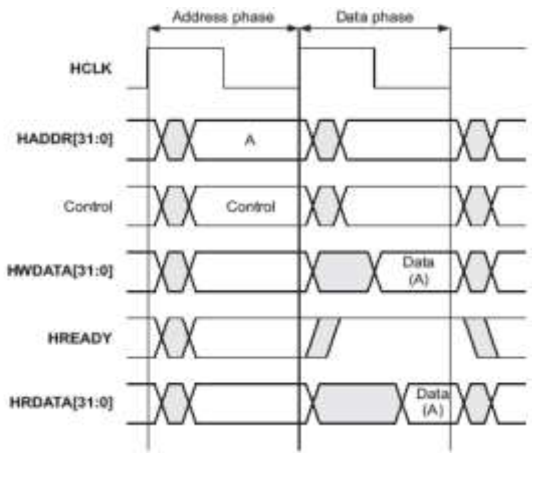

Figure 1.3 Simple Transfer

• The slave then samples the address and control information on the next rising edge of the clock.

• After the slave has sampled the address and control it can start to drive the appropriate response and this is sampled by the bus master on the third rising edge of the clock.

This simple example demonstrates how the address and data phases of the transfer occur during different clock periods. In fact, the address phase of any transfer occurs during the data phase of the previous transfer. This overlapping of address and data is fundamental to the pipelined nature of the bus and allows for high performance operation, while still providing adequate time for a slave to provide the response to a transfer.

A slave may insert wait states into any transfer which extends the transfer allowing additional time for completion.

III. MODELING STEPS

1. Identification of the AHB components is done and implemented as Verilog Modules. The AHB protocol consists of several key components, including the AHB master, AHB slave, AHB arbiter, AHB decoder, and AHB multiplexer.
2. Defined the AHB signals: The AHB protocol has a set of standard control signals that govern the communication between the master and the slave. These signals include the address bus (HADDR), data bus (HRDATA and HWDATA), control signals (HSEL, HTRANS, HREADY, etc.), and various handshaking signals (HREADYOUT, HRESP). These signals are defined as Verilog wire or reg variables.
3. Implementation of the AHB master: The AHB master initiates transactions and sends requests to the AHB bus. Implementation the AHB master module is done in Verilog. It should drive the appropriate control signals and transfer data between the master and slave based on the AHB protocol rules.
4. Implementation of the AHB slave: The AHB slave receives requests from the AHB bus and responds accordingly. Implementation of the AHB slave module is done in Verilog. It should monitor the control signals and decode the address to determine if the requested data is available. It should also receive and transmit data based on the AHB protocol.
5. Implementation of the AHB arbiter: The AHB arbiter resolves conflicts when multiple masters request access to the AHB bus simultaneously. Implementation of the AHB arbiter module is done in Verilog HDL. It should prioritize and grant access to masters based on the arbitration policy defined for the system.
6. Implementation of the AHB decoder and multiplexer: The AHB decoder decodes the address to identify the appropriate slave and select it for the transaction. The AHB multiplexer selects the appropriate data based on the decoded address and presents it on the data bus. Implementation of these modules is done in Verilog according to the specifications [3-5].
7. Interconnection of the the modules: Instantiation of the AHB master, slave, arbiter, decoder, and multiplexer modules is done in a top-level module. Interfacing of the AHB signals between the modules is done to enable communication and data exchange according to the AHB protocol.
8. Simulation and verification: written testbench modules to simulate and verify the functionality of the AHB model. Created appropriate test scenarios and monitoring of the AHB signals is done to ensure correct protocol adherence and data transfer.

IV. SIMULATION RESULTS

Simulation is carried out using Xilinx ISE 9.2 Tool, Simulation results for master control unit for some of the test cases are shown in the Figure 1.4

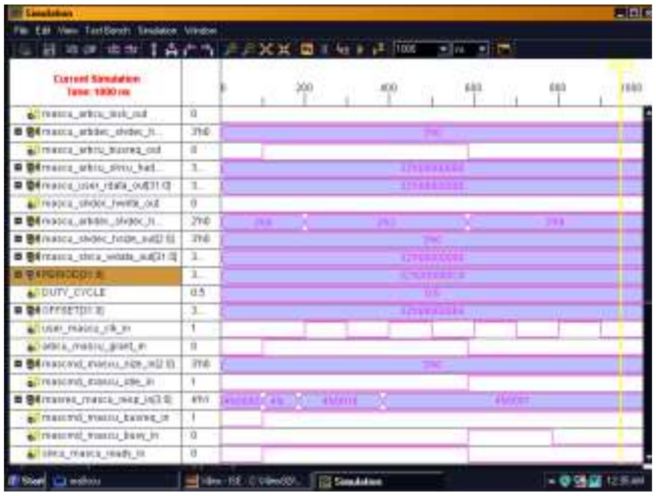
Simulation
File Edit View Test Bench Simulation Window
Current Simulation Time: 1000 ns
DUTY_CYCLE
0.5
200
400
600
800
1000

Figure 1.4 Simulation of Master Control Unit

The Figure 1.4 Shows the simulation results for the Master control unit. In this the mascmd_mascu_reset_in_n is set to 0 then the default values of the signals as shown in the figure will be assigned. When mascmd_mascu_reset_in_n is set to 1 then depending upon the value of {mascmd_mascu_busreq_in,, mascmd_mascu_ lock_ in} the assignments will be done as shown in the figure.

V. CONCLUSIONS

In this we have described the modeling of the AMBA AHB bus architecture. Five major models have been modeled: AHB Master model, Arbiter model, Decoder model, Slave model multiplexer model. The correctness of each model in terms of functionality and in terms of timing has been validated. The AMBA models have been integrated and tested with the Xilinx design environment.

The test cases are written for testing the bus function model for Single master bus testing, No overlap between masters bus access, Single slave, multiple slaves, Simple transfers, multiple transfers with and without waits and splits [4]. However, while the design of a AHB Bus functional model is a useful show-case for our framework, there is much to be done for a complete verification. Priorities are verifying datapath properties for the AHB, implementing locked access, having a more sophisticated arbitration policy and non-word-aligned transfers. In future work, the AHB model will be extended to support more complex bus transactions like split transfers, which will expand the usability of the model. For a further performance increase, multi-threaded master / slave bus interfaces will be included in the model. This will allow the modeling environment to take full advantage of the pipelined access even within bus accesses of the same master. Furthermore, it is planned to model the peripheral bus APB for an efficient connection to peripheral devices. Design of the UART model for serial transmission and reception of data through the APB Protocol. The Bridge can be designed between the AHB and APB for the communication of the protocol. Further all the above models can be designed in the System Verilog and transaction level modeling can be used where master and slave transactor, monitor and drive can be designed to make environment reusable [7]. The object-oriented feature of System Verilog can greatly enhance the reusability of test bench components [6].

VI. REFERENCES

[1] ARM, “AMBA Specification (Rev 2.0)”, available at http://www.arm.com.

[2] ARM, “AMBA AXI Protocol Specification”, available at http://www.arm.com.

[3] Advanced Digital Design with the Verilog HDL Ciletti, Michael D. Published by Prentice Hall PTR, 2002.

[4] Writing Test benches using System Verilog, Janick Bergeron, Synopsis Inc, Springer, 2006.

[5] Verilog LRM. https://www.accellera.org/images /downloads /standards/v-ams/VAMS-LRM-2-4.pdf.

[6] IEEE Standard for System Verilog—Unified Hardware Design, Specification, and Verification Language, IEEE Computer Society and the IEEE Standards Association Corporate Advisory Group, 2017.

[7] Krste Asanovi´c, ”Transactors for Parallel Hardware and Software Co-Design” IEEE International High Level Design Validation and Test Workshop, November 2007.