International Journal of Research in Engineering, Science and Management
Volume 4, Issue 7, July 2021
https://www.ijresm.com | ISSN (Online): 2581-5792


# Design and Implementation of Multiple-master, Multiple-slave Interface in AMBA AHB Protocol

S. Bhavana Surya[1*], C. Hema[2], M. Shruthi Priya[3], U. Devikarani[4], Ashwini S. Rathod[5]
[1,3,4,5]*Student, Department of Electronics and Communication Engineering, East West Institute of Technology, Bangalore, India*
[2]*Assistant Professor, Department of Electronics and Communication Engineering, East West Institute of Technology, Bangalore, India*

***Abstract*: The development of VLSI technology makes it possible to combine millions of transistors on a single chip to form a SOC. The main disadvantage confronted here is to verify correct and loss-less communication between the IP cores within the SOC. This can be achieved using standard communication protocols like the Advanced Microcontroller Bus Architecture (AMBA). The Advanced High-Performance Bus (AHB) is an integral part of the AMBA protocol series. It is designed for high-performance system modules. In the proposed paper, the AMBA AHB system is designed and implemented using Verilog HDL. The design is simulated using Xilinx ISE 14.6.**

***Keywords*: AHB, AMBA, SOC.**

## 1. Introduction

Owing to the progress of the semiconductor industry, it has become possible to integrate an oversized range of functional or IP blocks equivalent to processor cores, GPUs, and wireless communication models on one chip. To achieve economical information transmission and reliableness between SOC blocks and bus protocols that accommodates specific commonplaces are used. The AMBA bus protocol was introduced by ARM in the year 1996. AMBA is an open standard that describes how various components or blocks in the SOC are connected and controlled [2]. It has become the factual standard for front-end components in SOC. AMBA has an Advanced High-Performance Bus (AHB) as the standard backbone for high-performance synthesizable devices with high clock speeds.

## 2. Advanced High-Performance Bus (AHB)

AHB is a new generation of AMBA bus which is intended to address the requirements of high-performance synthesizable designs [1]. It is a high-performance system bus that supports multiple bus masters and provides high-bandwidth operation [1]. AMBA AHB implements the features required for high-performance, high clock frequency systems including: burst transfers, split transactions, single-cycle bus master handover, single clock edge operations, non-tristate implementation, wider data bus configurations (64/128 bits) [1].

## 3. Architecture of AMBA AHB

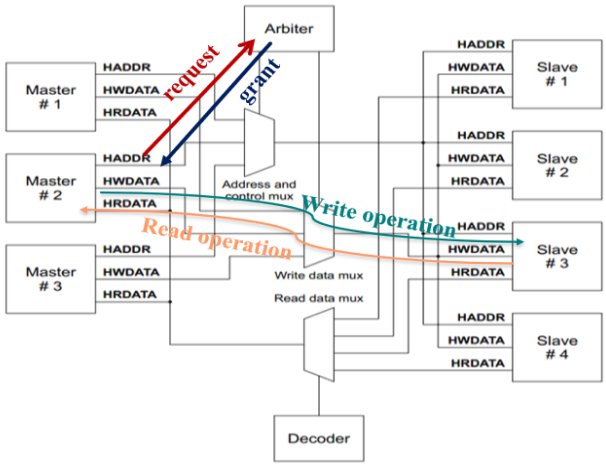


Fig. 1. AMBA AHB architecture

The architecture of the AMBA AHB is shown in Fig. 1. The AHB system uses a central multiplexer interconnection scheme to establish the connection between the master and the slave. The same architecture is implemented that consists of three masters and four slaves.

AHB is divided into three parts: master, slave, and control. The control part consists of an arbiter, a write multiplexer, a read multiplexer, an address control multiplexer, and an address decoder.

Considering an example where Master-2 is making an attempt to communicate with Slave-3 to perform read and write operations. For this, the Master must first send a signal to request the Arbiter to use the bus. At this time, the Master will begin the AHB transmission. The Master sends out address and control signals. These signals in the main provide address information, transmission direction, and burst type and it conjointly sends the information to be written.

With the help of the address transferred by the Master, the slave will be selected by the decoder

The slave that receives the address and control information will send a response signal to the master, and the master

*Corresponding author: bhavanasuryas@gmail.com

samples the response signal.

This means, that the primary write operation and read operation between the master and slave is completed. Verilog code is written for master, slave, arbiter, decoder, address control multiplexer, write data multiplexer, and read data multiplexer.

## 4. Design Implementation

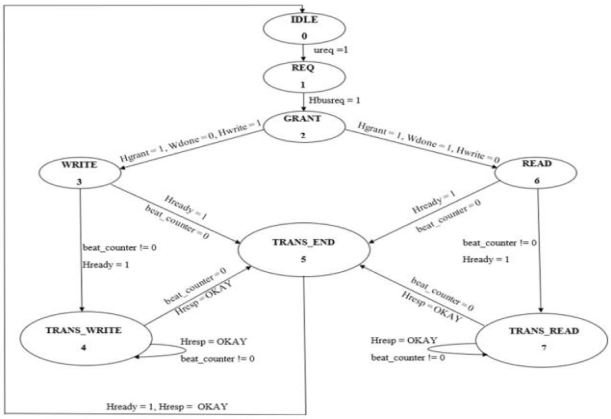


Fig. 2. FSM for AMBA AHB

AMBA AHB is designed using FSM, as shown in Fig.2. FSM includes a total of 8 states, namely state_idle, state_request, state_grant, state_write, state_trans_write, state_trans_end, state_read, and state_trans_read.

The working of FSM is as follows:

### *A. Idle State*

The AHB FSM starts from the idle state, which is the default state in the state machine. If the master has no transaction to perform or it has simply completed one, it will stay in this state. If the Master desires to execute any other transaction, it will enter the request_state to request the bus from the arbiter. Since master is the controlling module in the design few users described signals are used. For example, the Ureq signal is used to control the selection of Master.

### *B. Req State*

Here, the master can set the HBUSREQ signal high indicating to the arbiter that it needs to own access to the bus.

### *C. Grant State*

The master will remain in the same state until it gets access to the bus. Through the utilization of some arbitration mechanisms, the arbiter grants access to the bus. The arbitration mechanism used with inside the design of AMBA AHB is the highest priority. Wherein, Master-1 is given the highest priority among the three masters followed by Master-2 and Master-3. Once the selected master has access to the bus, it will move to state_write for write transfer or state_read for reading operations relying upon HWRITE signal.

### *D. Write State*

This state is a non-sequential write state. Here HREADY signal is monitored to test whether or not it is asserted high due to the fact HREADY = 1 is an indication that there aren't any formerly pending transfers. Beat_counter is used to count the number of beats to be transferred at some stage in the 4, 8, or 16 beat burst operation. It is in this state, the first data of the burst is transferred.

### *E. Trans_Write*

This state is a sequential write state i.e., the second one or subsequent data of a burst is transferred. Once all the data are transferred, beat_counter would be zero, and the master moves directly to the trans_end_state. If beat_counter isn't zero it will continue to be withinside the same state due to the fact beat_counter!=0 is a sign that all data aren't transferred yet.

### *F. Trans_End State*

In this state, it is checked whether or not HREADY is 1 and the response from the slave is OKAY. These signals are monitored because of the very fact HREADY = 1 and HRESP = OKAY shows the transaction has finished with success at the bus after which the master moves on to the idle_state.

### *G. Read State*

This state is a non-sequential read state here HREADY signal is monitored to test whether or not it is asserted high due to the fact HREADY=1 is a sign that there aren't any formerly pending transfers. Beat_counter is used to count the number of beats to be received at some stage in the 4, 8, or 16 beat burst operation. It is in this state, the first data of the burst is received.

### *H. Trans_Read State*

This state is a sequential read state i.e., the second one or subsequent data of a burst is obtained. Once all of the data are obtained, beat_counter would be zero, and the master moves directly to the trans_end_state. If beat_counter is not 0 it will continue to be withinside the same state due to the fact beat_counter != 0 is a sign that all data aren't received yet.

## 5. Results

As multiple Masters, multiple Slaves are involved in the design of AHB, incremental burst operation is implemented between a random combination of Masters and Slaves.

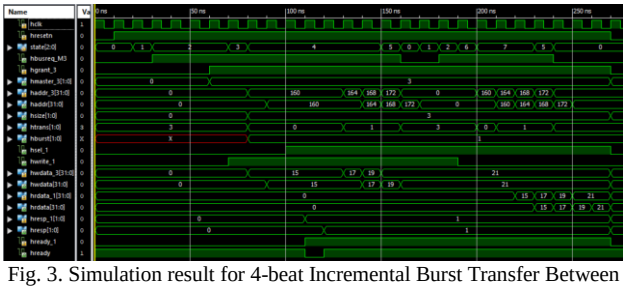

Fig. 3. Simulation result for 4-beat Incremental Burst Transfer Between Master-3 And Slave-1

Here 0-5 states are used during a write operation in which 0 is state_idle(where all signals are initialized to zero), 1 is state_request(where HBUSREQ_M3 is made high), 2 is state_grant(where HGRANT_3 is asserted high along with HAMSTER_3), 3 is a non-sequential state and 4 is the

sequential state. During this state, HADDR_3 is given with inside the test bench after which via add and control mux, the address is placed on HADDR. 160 is the beginning address and since a 4-beat burst transfer is carried out, 160 along with 3 more subsequent addresses are transferred. Since Slave-1 is involved in the transfer HSEL_1 is made high. Once all this is done, HWRITE_1 is made high for a write operation and later made low for read operation. HWDATA_3 is given in the test bench and via write data mux the 4 consecutive data are written into the memory. After trans_end state, the following state is the idle state, and all over again signals are initialized to 0 to begin the read operation. Once HWRITE_1 is made low, 4 data are read from the memory via read_data mux using HRDATA_1 and then placed on HRDATA. For every write and read operation HREADY_2 and HRESP_2 is made high.

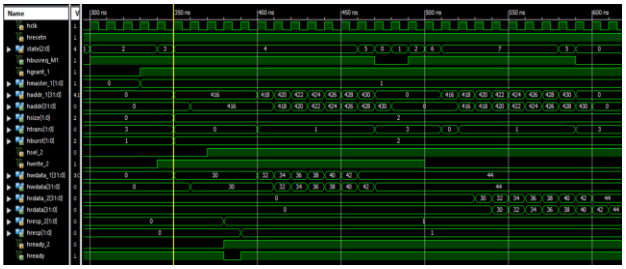

Fig. 4. Simulation result for 8-beat Incremental Burst Transfer Between Master-1 And Slave-2

In this simulation result, Master-1 and Slave-2 are interacting and an 8-beat incremental burst operation is carried out. So, 8 addresses are routed out and 8 data are written into the memory and read from the memory.

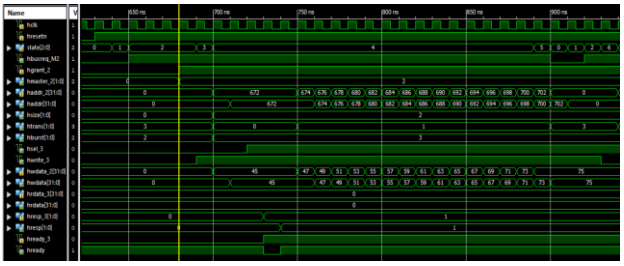

Fig. 5. Simulation Result for 16-beat write Incremental Burst Transfer Between Master-2 and Slave-3

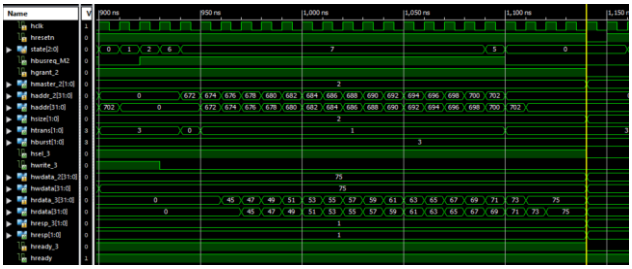

Fig. 6. Simulation result for 16-beat read Incremental Burst Transfer Between Master-2 And Slave-3

In this simulation result, Master-2 and Slave-3 are interacting and 16-beat incremental burst operation is carried out. So, 16 addresses are routed out and 16 datas are written into the memory and read from the memory.

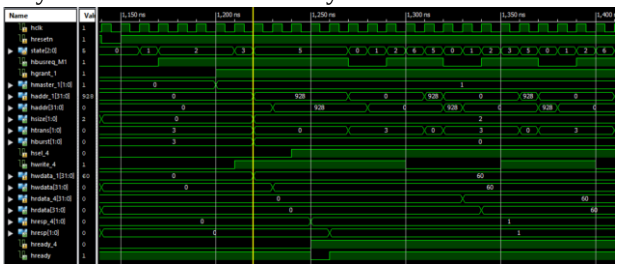

Fig. 7. Simulation Result for Single Transfer Between Master-1 and Slave-4

In this simulation result, Master-1 and Slave-4 are interacting and single burst operation is carried out.

## 6. Conclusion

The AMBA AHB multi-master and multi-slave communication protocol have been successfully developed and implemented. The AHB communication protocol is designed to improve system performance by implementing incremental burst read/write transfers. The simulation results show that the communication between the master and the slave through the AHB protocol proceeds as expected.

## 7. Future scope

In the present work, the AHB system is designed which supports 3 - Masters and 4- Slaves. Developing the system that can support 16 – Master and 16 – Slaves for both incremental and wrapping burst is the future scope of this project. Future extension of the project also includes split transaction compatible AHB system.

## References


[1] AMBA Specification Rev 2.0. ARM Ltd., 1999

[2] funzen.net.https://www.funzen.net/2020/12/07/the-advanced-microcontroller-bus-architecture-an-introduction.

[3] AHB Example AMBA System Technical Reference Manual. ARM Ltd., 1999.

[4] Deeksha. L and Shivakumar B R, *"*Effective Design and Implementation of AMBA AHB Bus Protocol using Verilog*"*, in *ICISS*, 2019.

[5] Permalla Giridhar and Dr Priyanka Choudhury, *"*Design and Verification of AMBA AHB*"*, in *1st ICATIECE*, 19-20 March 2019.

[6] K Manikanta Sai Kishore and M Naresh Kumar, *"*Design and Implementation of Efficient FSM for AHB Master and Arbiter*"*, in *International Journal and Magazine of Engineering, Technology, Management and Research*, vol. 2, no. 3, March, 2015.

[7] J S C Varma Nagaraju and N. H. N. S. Srinivasa Murthy, *"*Design of Multiple Master/Slave Memory Controllers with AMBA Bus Architecture*"*, in *International Journal of VLSI System Design and Communication Systems*, vol. 3, no. 10, pp.1563-1567, December 2015.

[8] Shraddha Divekar and Archana Tiwari, "Interconnect Matrix for Multichannel AMBA AHB with Multiple Arbitration Technique," in *ICGCCEE*, 6-8 March 2014.

[9] Bhaumik Vaidya and Anupam Devani, *"*Design of an Efficient finite state machine for the implementation of AMBA AHB master*", in Journal of Information, Knowledge and Research in Electronics and Communication Engineering*, vol. 2, no. 2, 2013.