ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Design And Implementation of Efficient FSM For AHB Master And Arbiter

K. Manikanta Sai Kishore,
M.Tech Student,
GITAM University, Hyderabad

Mr. M. Naresh Kumar, M. Tech (JNTUK),
Assistant Professor,
GITAM University Hyderabad

ABSTRACT
Due to Moore's law more and more amount of logic is being placed onto a single silicon die and it is driving the development of highly integrated SoC designs. So this high computational power must be matched with interconnect fabric which can handle it. There are many interconnect buses that are widely used in the industry like AMBA, Wishbone, Core Connect, Avalon etc. AMBA is most proffered among all of them because it has a hierarchy of buses with AHB (Advance high performance bus) can be connected to high performance peripherals and APB (Advance Peripheral Bus) that can be connected to low performance peripherals. Nowadays in industry development of Silicon on Chip (SOC) devices with reusable IP cores are given higher priority, the major challenge faced here is to ensure proper lossless communication between these IP cores in SOC device, this can be ensured with the help of standard communication protocols such as AMBA from ARM Ltd. In this paper we design and synthesize efficient Finite State Machine (FSM) for master and arbiter interface in AMBA AHB.

INTRODUCTION
Advanced Microcontroller Bus Architecture (AMBA)is a protocol that is used as an open standard, on-chip interconnect specification for the connection and management of functional blocks in a system-on-chip (SoC) [1] AMBA assists the progress of right-first-time development of multiprocessor designs with large number of controllers &peripherals

[2] The Advanced Microcontroller Bus Architecture (AMBA) has the ability to re-use designs and here it means it has the ability to re-use IP. IP re-use in today's technology is an important factor in reducing the development costs and timescales for System-on-chip (SoC)

[3] AMBA is a standard interface specification that makes sure of the compatibility between IP components provided by different design teams or vendors.

The worldwide reception of AMBA specifications all over the semiconductor industry has driven a comprehensive market in third party IP products and tools to support the development of AMBA based systems.

OVERVIEW OF AMBA BUSES
There are three different buses defined within the AMBA 2.0 specification –

Advanced High Performance Bus (AHB),
Advanced System Bus (ASB), and
Advance Peripheral Bus (APB).

This paper mainly deals with AMBA AHB and particularly AHB master and arbiter.

Advanced High-performance Bus (AHB)
The AMBA AHB is for high-performance, high clock frequency system modules. The AHB acts as the high-performance system *backbone* bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macro cell functions. AHB is also specified to ensure ease of use in an efficient design flow using synthesis and automated test techniques.

Advanced System Bus (ASB)
The AMBA ASB is for high-performance system modules. AMBA ASB is an alternative system bus suitable for use where the high-performance features of AHB are not required. ASB also supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macro cell functions.

Advanced Peripheral Bus (APB)
The AMBA APB is for low-power peripherals. AMBA APB is optimized for minimal power consumption and reduced interface complexity to support peripheral functions. APB can be used in conjunction with either version of the system bus.

Objectives of the AMBA specification

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The AMBA specification has been derived to satisfy four key requirements:

- To facilitate the right-first-time development of embedded microcontroller products with one or more CPUs or signal processors
- To be technology-independent and ensure that highly reusable peripheral and system macro cells can be migrated across a diverse range of IC processes and be appropriate for full-custom, standard cell and gate array technologies
- To encourage modular system design to improve processor independence, providing a development road-map for advanced cached CPU cores and the development of peripheral libraries
- To minimize the silicon infrastructure required to support efficient on-chip and off-chip communication for both operation and manufacturing test.

A typical AMBA-based microcontroller

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.

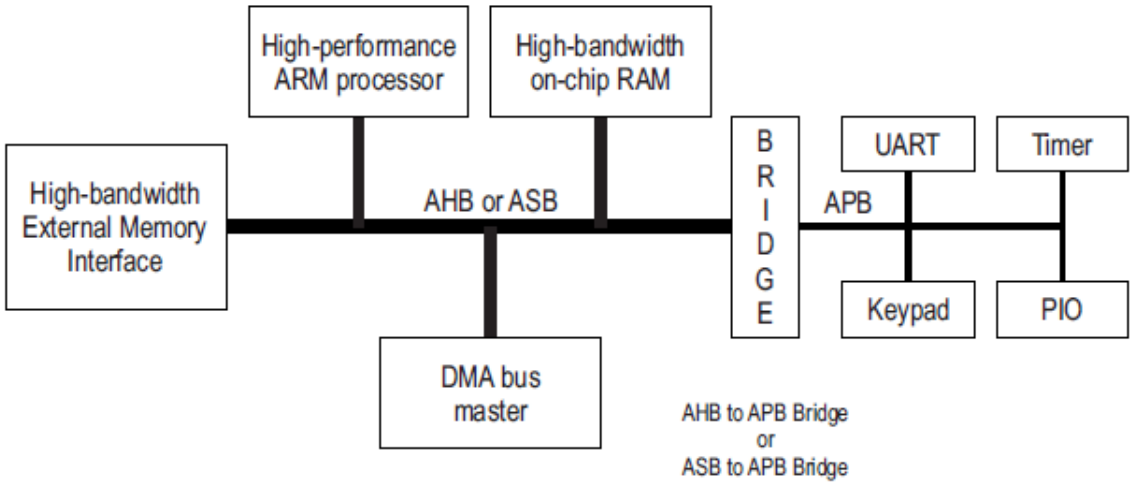

Figure1: AMBA based Microcontroller

AHB INTERCONNECTION

AHB is a new generation of AMBA bus which is intended to address the requirements of high performance synthesizable designs. It is a high performance system bus that supports multiple busmasters and provides high-bandwidth operation. AMBA AHB implements the features required for high-performance, high clock frequency systems including:

- Burst transfers
- Single-cycle bus master handover
- Wider data bus configurations (64/128, up to1024 bits).
- SEQ, NON-SEQ, BUSY IDLE transfer types
- Address decoding

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

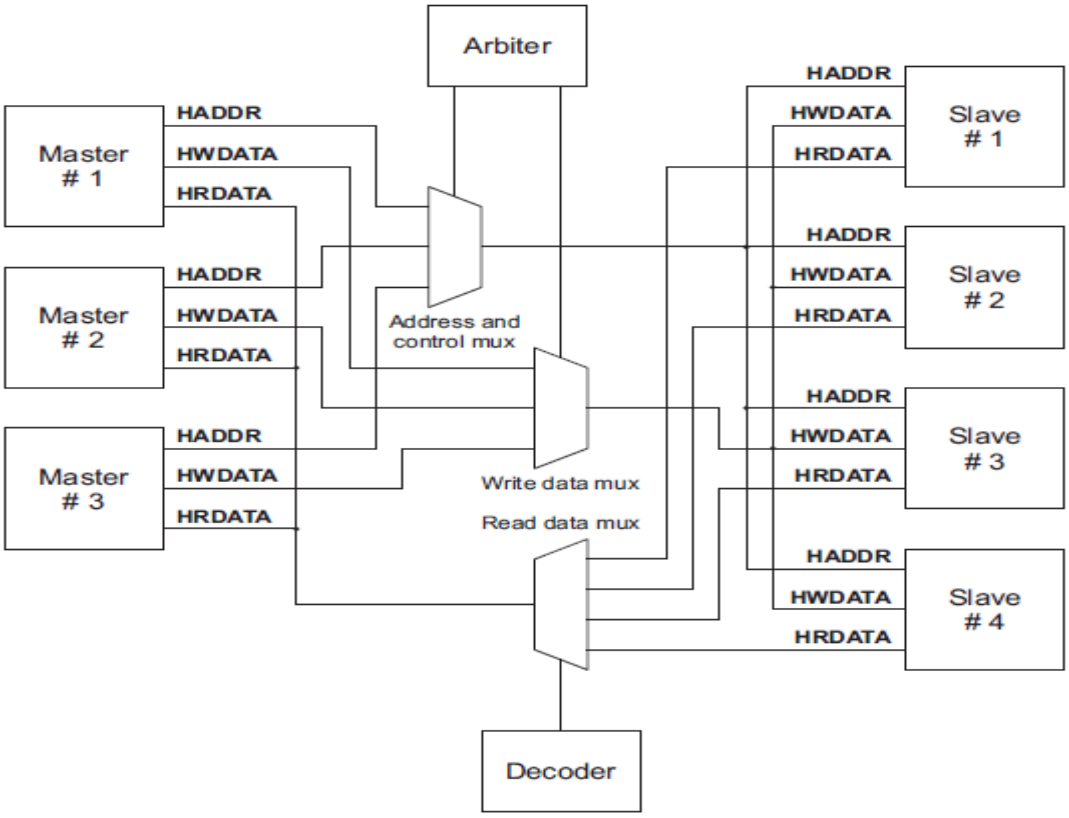

Figure2: AHB Interconnection Diagram

An AMBA AHB design is having following components:

AMBA AHB MASTER:

An AMBA AHB bus master is able to initiate read and write operations by making use of an address and control information. Only one bus master at a time is allowed to actively use the bus.

AMBA AHB SLAVE:

An AMBA AHB bus slave responds to read and write operation initialized by master within a given address space range. The bus slave signals back to active master about the success, failure or waiting of the data transfer.

AMBA AHB ARBITER:

An AMBA AHB bus arbiter gives an assurance that only one bus master at a time is allowed to initiate the data transfers.Even though the arbitration scheme is fixed, any arbitration scheme can be used like Round Robin, Fair Chance etc. depending on the application requirement.

AMBA AHB DECODER:

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

Figure 2 shows the working principle of AHB Bus. Before starting the AMBA AHB transfer, the bus master must have to be granted access to the bus. In this process first of all master asserts a request signal to an arbiter. Now the arbiter will indicate when the master will get the grant of the bus. This decision of granting the access to bus is achieved using some arbitration mechanism like priority based or round robin mechanism etc. A granted bus master then starts the AHB transfer by first driving an address and control signals. These address and control signals provide information about an address, direction and width of the transfer, burst transfer information if the transfer forms the part of the burst

AHB MASTER INTERFACE

The AHB bus master is the part which initiates the read or write transfers, it can be said as the most complex part in the AMBA AHB system. It is the module in AMBA

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

AHB that tarts the transfer by sending a request signal hbusreq to the arbiter module to grant access of the bus.

Figure shows the schematic of AHB master developed for this paper. The bus master waits for the hgrantx signal to be active high, as soon as the bus master gets the hgrantx signal, AHB master takes the appropriate action according to the Transfer response signals, and sends the required address, control and data signals to the slave and arbiter. AHB master is responsible for sending the address and control signal to the slave device weather the operation is read or write. The AHB master output depends upon the hready and hresp [1:0] provided by the slave device. The hrdata carries the read data signal from the slave device and hwdata carries the data to be written to the slave device by the master device. hwrite signal tells weather the read operation is being carried or the write operation is taking place. The data width and the address width are of 32 bit in size. The hsize [2:0] tells the size of the transfer, and the type of data transfer is given by the htrans [1:0] signal. The hburst [2:0] signal gives the information about the burst transfer, i.e. single type, increment by 4, increment by 8, increment by 16.

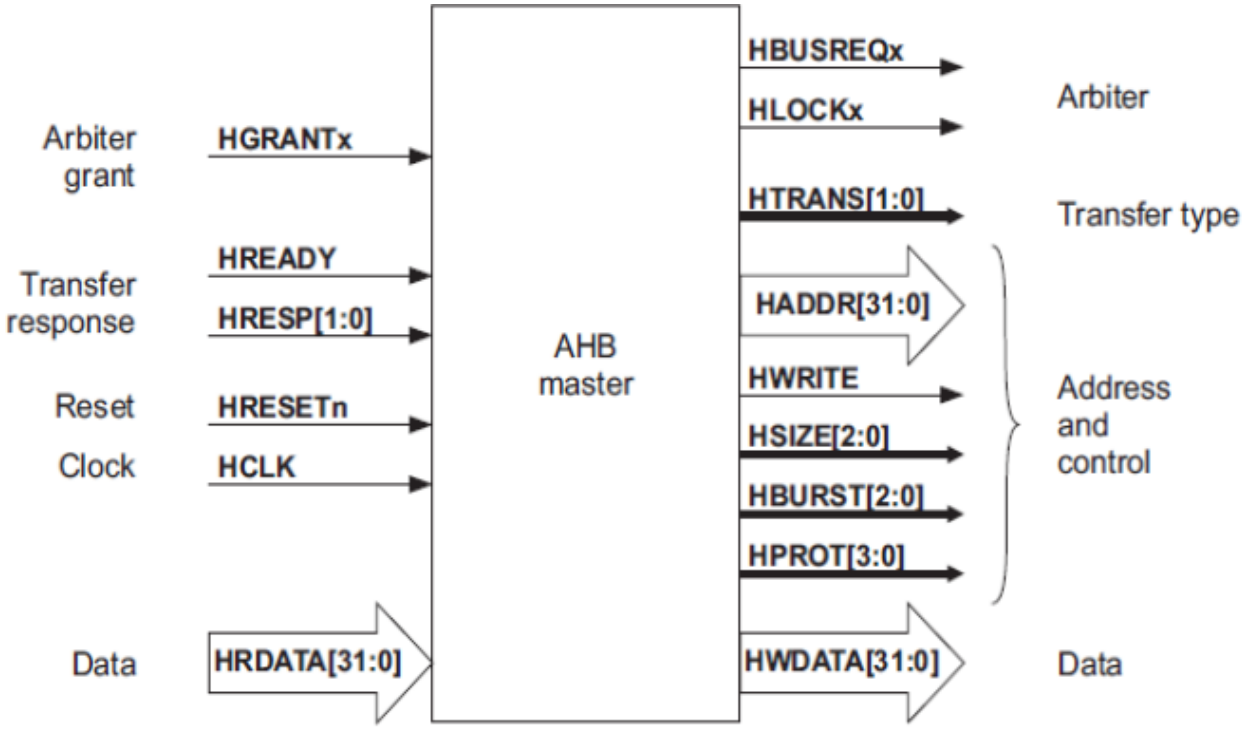

Figure3: AHB Master Interconnection Diagram

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

FINITE STATE MACHINE OF AHB MASTER

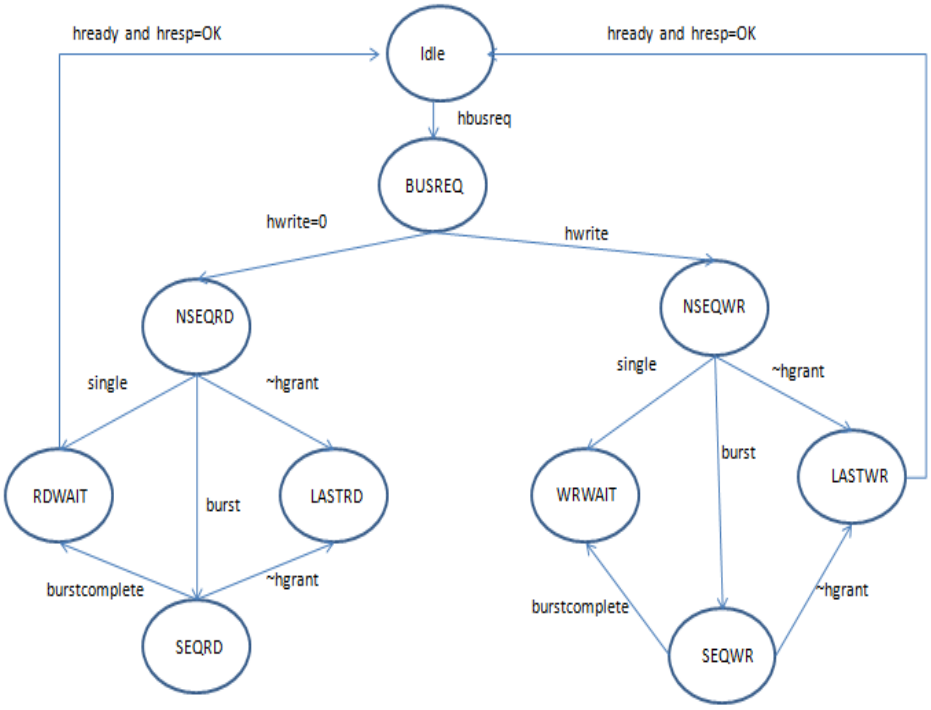

Figure 4: FSM of AHB Master

The following section contains the brief description of every state in the diagram.

IDLE STATE: Initially when the system is on the master will be in its IDLESTATE; the master will remain in its idle state until the busreq input from test bench is not made high. Here the idea is, for starting a transfer the master sends a bus request signal hbusreq to the arbiter and moves on to the next state BUSREQUESTSTATE.

BUS REQUEST STATE: In this state the master waits for the hgrant signal from the arbiter for the hbusreq signal sent from the previous state. If the hgrant signal is not given by the arbiter it waits in the same state, if the hgrant signal is obtained it then jumps to NSEQWR or NSEQRD states according to the input write given through the test bench. If the write signal is 1 then write mode is activated and master will generate the output hwrite as 1. At this state the address and control signals are also given to the master through the test bench. The address signal is addr, and the control signals being size, trans, burst, lock. The hready signal and hresp signal from the slave is also checked in this state.

NSEQWR: This state is the new sequence write state. In this state the master will check the control signals trans, size, burst. If the burst of the transfer is 00 which indicates single, the transfer is of single burst and the state will change to WRWAIT state. The trans signal value for the start of each transfer is 00 which indicates non sequential. If the burst signal is incr4, incr8, incr16 the next state will be SEQWR. In general we can say that at this state the master interface will check whether it need to transfer a single transfer or a sequential incrementing transfer.

SEQWR: This state is the sequential write state. The master comes to this state from NSEQWR state if the burst is incrementing burst. The master will be in this state until it finishes the number of burst to be transferred. For achieving this a counter is generated inside the state which counts the number of burst sent by the master when the count reaches burst-1 value the master changes to WRWAIT state. After sending each burst the master checks the hresp and hready signal from the slave, if hresp signal is 00 it means the transfer was successful and next transfer can be sent. If the hresp from the slave module is 01, 10 or 11 it means the transfer didn't happen successfully and different routines

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

need to befollowed for each hresp signal. Throughout this state while the transfer is being performed the hready signal from the slave interface will be 0, indication transfer is taking place right now.

WRWAIT: This state is named write wait state, as the name suggests here after completion of transfer of burst the master waits until the hready signal from the slave becomes 1. If the hreadysignal still remains 0 in this state that means the slave has not finished the transfer and requesting some time for wait, once the hready is made high it means that the transfer has been completed.

LASTWR: This state is the last write state; here in this the master can come from any other state during a write transfer. the master go into LASTWR state whenever the hgrant signal from the arbiter goes to 0, i.e. it literally means that the master has lost the control over the bus for some reason, so in order to store the current activity of master, the master moves in to LASTWR state, when the hgrant to the particular master becomes 1, the master moves back to the state from which it went to LASTWR state in the first place.

NSEQRD: This state is the new sequence read state. In this state the master will check the control signals trans, size, burst. If the burst of the transfer is 00 which indicates single, the transfer is of single burst and the state will change to RDWAIT state. The trans signal value for the start of each transfer is 00 which indicates non sequential. If the burst signal is incr4, incr8, incr16 the next state will be SEQRD. In general we can say that at this state the master interface will check whether it need to transfer a single transfer or a sequential incrementing transfer.

SEQRD: This state is sequential read state. The master comes to this state from NSEQRD state if the burst is not single. The master will be in this state until it finishes the number of burst to be transferred. For achieving this a counter is generated inside the state which counts the number of burst sent by the master when the count reaches burst-1 value the master changes to RDWAIT state. After sending each burst the master checks the hresp and hready signal from the slave, if hresp signal is 00 it means the transfer was successful and next transfer can be sent. If the hresp from the slave module is 01, 10 or 11 it means the transfer didn't happen successfully and different routines need to be followed for each hresp signal.

RDWAIT: This state is named read wait state; the logic behind this state is that when the master has finished sending all the burst transfers it waits for the hready signal from the slave module to change from 0-1, indicating the completion of a read transfer.

LASTRD: This state is named last read state; the uniqueness of this state is the master can come in to this state from any other state during a read transfer. the master go into LASTRD state whenever the hgrant signal from the arbiter goes to 0, i.e. it literally means that the master has lost the control over the bus for some reason, so in order to store the current activity of master, the master moves in to LASTRD state, when the hgrant to the particular master becomes 1, the master moves back to the state from which it went to LASTRD state in the first place.

AHB ARBITER

The arbitration mechanism is used to ensure that only one master has access to the bus at any one time. The arbiter performs this function by observing a number of different requests to use the bus and deciding which is currently the highest priority master requesting the bus. The arbiter also receives requests from slaves that wish to complete SPLIT transfers.

Any slaves which are not capable of performing SPLIT transfers do not need to be aware of the arbitration process, except that they need to observe the fact that a burst of transfers may not complete if the ownership of the bus is changed.

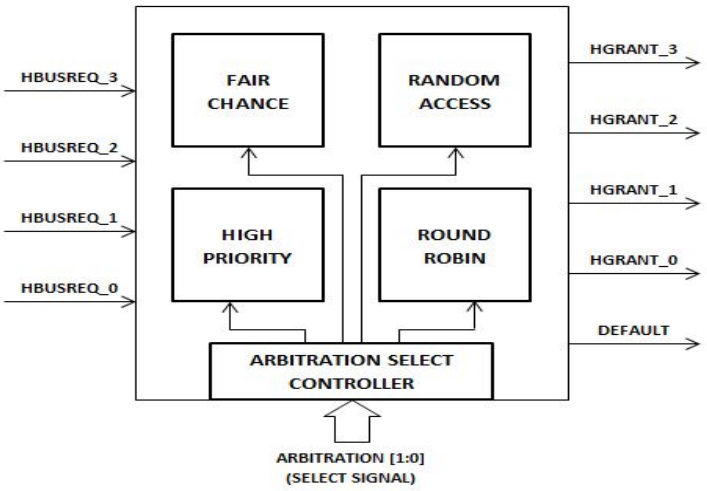

Figure 5: Functional Diagram of AHB Arbiter

With the reconfigurable functionality, it can assign any arbitration scheme among four which are designed, depending onthe requirement of IP cores. First of all, any

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

master among the four can request for an access of the bus. Then dependingon the requirement of an application, we can choose any arbitration scheme using input signal ARBITRATION [1:0].Now, as per the selected arbitration scheme, grant signal will be generated to any particular master and hence master willget bus access. Choice of the arbitration algorithm can selected as follows.

ARBITRATION [1:0]	Arbitration Selection Algorithm
00	High Priority Algorithm
01	Fair Chance Algorithm
10	Random Access Algorithm
11	Round Robin Algorithm

Table 1: Selection of Arbitration Algorithm

FINITE STATE MACHINE FOR AHB ARBITER

Master asserts request to an arbiter and arbitergrants access to the bus master based on any particular arbitration scheme. Arbiter gives an assurance of granting requestto only one bus master at a time.

The following section contains the brief description of every state in the diagram.

IDLE:

FSM of AHB arbiter starts with IDLE state and this is the default state in the state machine. When reset signal HRESETnis active, master has finished the transaction or master has lost the grant in the middle of the transaction anyhow, at thattime arbiter will stay in this state. When master wants to perform any transaction, the master asserts request signal to anarbiter and then arbiter will move into next state i.e. ARBITRATION state to choose arbitration scheme, else arbiter willwait in this state until it gets any request or requests.

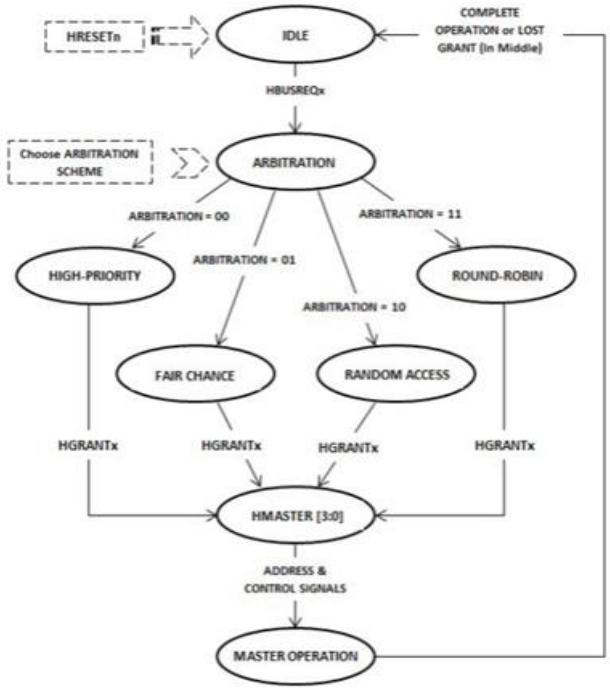

Figure 6: FSM of AHB Arbiter

ARBITRATION:

When an arbiter will get any request or requests i.e. HBUSREQx is active, it will move to this state from IDLE state. Inthis state depending on the application requirement any particular arbitration scheme can be chosen among the four withthe help of a signal ARBITRATION [1:0]. Proposed arbitration schemes are High Priority, Fair-Chance, Random Accessand Round Robin. Now, as per the selected arbitration schemearbiter will move into next state i.e. suppose, if "ARBITRATION = 01" an arbiter will move to Fair-chance state to grantan access.

ROUND ROBIN:

When arbitration scheme selection signal "ARBITRATION = 11", arbiter enters into this state. In order to processrequests fairly, a Round Robin algorithm employs time sharing, giving each master a time slot and interrupting themaster if it is unable to complete the transaction within prescribed time slots.

The time slots are assigned to masters based on the number of beat burst operation i.e. HBURST [2:0]. There is 4-beat, 8-beat or 16-beat burst operations either incrementing or wrapping. If 4-beat burst operation, 4 time slots to complete thetransaction and so on. After granting access to any particular master arbiter will move to the

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

HMASTER state to generate the master number which has granted the access.

HMASTER:

An arbiter enters into this state, when any particular master has granted bus access. In this state arbiter generates master number who has just granted access. There are maximum 16 masters supported by AMBA 2.0 specification and accordingly HMASTER [3:0] can generate up to 16 numbers. Numbers for Master_0, Master_1, Master_2 and Master_3are assigned to 0000, 0001, 0010 and 0011 respectively. Now, after generating the master number an arbiter moves to the MASTER OPERATION state where master starts its read and write operation.

MASTER OPERATION:

After accessing the grant, master starts read and write data operations in this state with the help of address and control information. If master lost grant in the middle, it has to again assert the request to have an access of the bus. After completion of data transfer or anyhow if master lost its grant, an arbiter moves to an IDLE state and the cycle starts again.

SIMULATION RESULTS

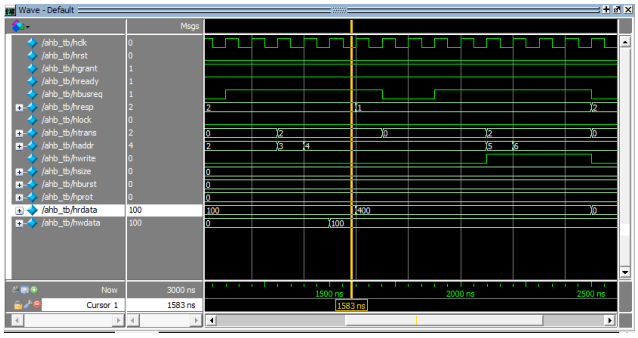

Figure 7: Simulation Waveform of AHB Master for Read and Write Operations

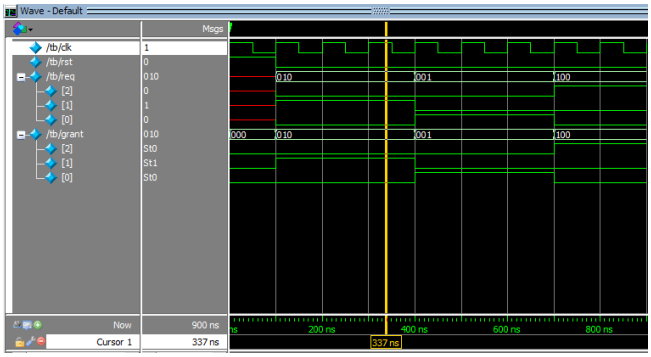

Figure 8: Simulation waveform of AHB Arbiter using Round robin Algorithm

CONCLUSION

The AHB master interface and arbiter interface are designed using the finite state machines in Verilog hardware description language and the design is simulate with the help of Questa Sim. The completed AMBA AHB system is then checked for proper lossless communication between master and slave interface.

REFERENCES

[1] AMBA specification, version 2.0

[2]"AHB Example AMBA System", Technical Reference Manual, ARM Inc.

[3]BhaumikVaidya, Anupamdevani "Design of an efficient finite state machine for the implementation of AMBA AHB Master",2013

[4] PriyankaGandhani, Charu Patel "Moving from AMBA AHB to AXI Bus in SoC Designs: A Comparative Study", 2011

[5] P.Harishankar, Mr. ChusenDuari, Mr. Ajay Sharma, "Design and synthesis of efficient FSM for master and slave interfaces in AMBA AHB", IJEDR 2014

[6]Pravin S. Shete, Dr. ShrutiOza"Design of an AMBA AHB reconfigurable Arbiter for On-chip Bus Architecture", IJAIEM 2014