Design and Synthesis of Efficient FSM for Master and Slave Interface in AMBA AHB

[1]P.Harishankar, [2] Mr.Chusen Duari, [3] Mr. Ajay Sharma
[1]M.Tech Student, [2] Assistant Professor, [3] Manager Engineering
[1]Department of Electronics and Communication Engineering, Manipal University Jaipur, Rajasthan, India
[2] Department of Electronics and Communication Engineering, Manipal University Jaipur, Rajasthan, India
[3] DKOP LABS PVT LTD, Sector 2 Noida

***Abstract* - 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 slave interface in AMBA AHB. The interfaces are capable of responding to split, retry and error responses during a simple read and write transfer. The AMBA AHB system is designed using Hardware description language such as Verilog using Modelsim tool and synthesized using Xilinx ISE tool.**

I. INTRODUCTION

The main aim of this paper is to design efficient finite state machines of master and slave interface in AMBA AHB system. This work is mainly chosen because in industry the demand for system on chip (SOC) devices is growing, and the main problem faced in these SOC devices are the lack of proper communication between the different Intellectual Property (IP) cores. The Advanced Microcontroller Bus Architecture (AMBA) introduced by ARM Ltd is an onchip communication protocol used as an open source in SOC devices. Some of the key features of AMBA are:

- It helps in the right first time development of SOC devices.
- Promotes technology independence and devices
- Encourages modular system design
- Minimizes silicon infrastructure for onchip and off chip communication in devices.

AMBA has Advanced High Performance Bus (AHB) as a standard backbone bus for high performance, high clock frequency synthesizable devices, the AMBA AHB have the following features:

- Split transactions
- Burst transfers
- Single clock edge operations
- Wider data bus configurations
- Single cycle bus master handover.

II. AMBA AHB MASTER INERFACE

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

Figure 1 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.

The design of AHB master for this paper is quite different as it contains some extra signals which are considered to be the signals of the device attached to the AHB master; Table 1 shows the extra signals taken in the Master interface.

Fig 1 Schematic of AMBA AHB Master

Assumptions Taken For Designing Master Module

- At a time only one master can access the bus and can only do a transfer to one slave.
- The size of transfer is fixed at 32bits.
- Only fixed length incrementing transfer of incr4, incr8, incr16 and single burst transfer is used.
- Split and retry condition is included, which are invoked as a result of hresp signal from the slave.
- After each transfer the hready signal of the slave goes to high, indicating the completion of transfer and the slave is free for another transaction.
- The data transfer happens in seqwr and seqrd state until the count taken in this state is equal to hburst-1.

Table 1 the extra signals taken in the Master Interface

EXTRA SIGNAL ADDED TO THE INTERFACE	*TYPE*	*EXPLANATION*
lock	Input	This signal indicates to the master interface that the IP wants to send a locked transfer.
busreq	Input	This signal indicates to the master inter face that the IP core wants to send a request to the arbiter for acquiring the bus.
write	Input	This signal indicates the IP core wants to start a read or write transaction, when write=0 read and when 1 write.
wdata[31:0]	Input	This signal indicates to the master interface the data to be written at the memory location during write transfer.
size[1:0]	Input	This signal indicates the size of individual transfers from the IP core.
burst[4:0]	Input	Indicates the type of burst the master IP core want to send during transfer.
trans[1:0]	Input	Indicates the type of transfer the IP core wants to make.
addr[31:0]	input	Indicates the starting address for every transaction the IP core wishes to perform.

Finite state machine of AHB master interface

The Finite State Machine (FSM) of the master is designed using Mealy machine, in which the output of the states depend on both the input and the current state of the machine. The different states of the FSM for the master is given in Table 2 and Figure 2 shows the FSM of the AHB master interface developed for this paper.

Table 2 the states of AMBA AHB Master Module.

STATES	*STATE ASSIGNMENTS*
IDLESTATE	0
BUSREQUESTSTATE	1
NSEQRD	2
NSEQWR	3
RDWAIT	4
LASTRD	5
SEQRD	6
WRWAIT	7
LASTWR	8
SEQWR	9

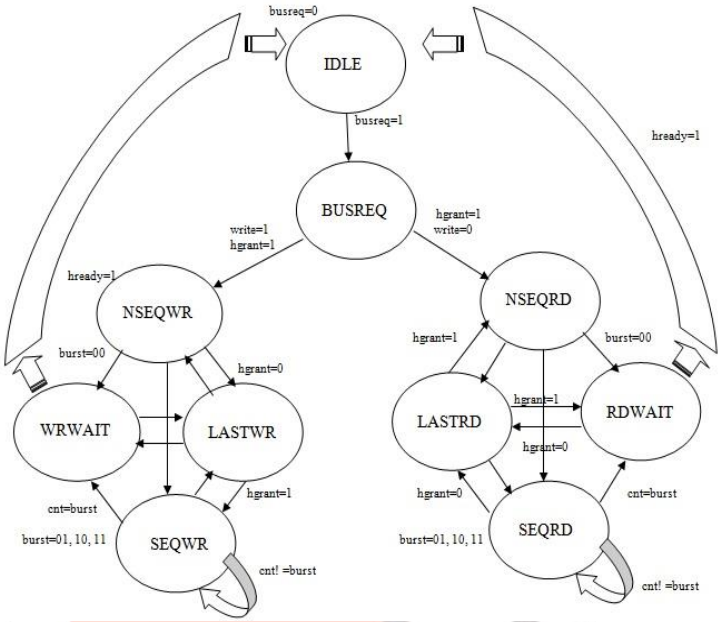

Fig 2 FSM of AHB master interface

The different states of the FSM changes according to the different inputs given, and also due to the current state in which it is present. The working of the FSM is as follows

The hrstn is the reset signal which is an active low signal. When hrstn signal is high the system is ready for communication.

- ***IDLESTATE:*** initially when the system is on the master will be in its IDLESTATE; the master will remain in its idlestate 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.
- ***BUSREQUESTSTATE:*** 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 need to be

followed 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 hready signal 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.

III. AMBA AHB SLAVE INTERFACE

An AHB bus slave responds to transfers initiated by bus masters within the system. The slave uses a HSELx select signal from the decoder to determine when it should respond to a bus transfer, in this work initially for the design of slave module the ***hselx*** signal was given from the test bench, later while simulating the top module the ***hselx*** signal is given from the decoder itself according to the address requested by the master module. The other signals required for the transfer, such as address and control information is generated by the bus master, and is given to the slave module through arbiter module. The slave in this work does use the ***hsplitx*** signal i.e. the slaves in this paper is capable of ***SPLIT*** and ***RETRY*** functions. *SPLIT* transfers improve the overall utilization of the bus by separating (or splitting) the operation of the master providing the address to a slave from the operation of the slave responding with the appropriate data. When a transfer occurs the slave can decide to issue a *SPLIT* response if it believes the transfer will take a large number of cycles to perform. This signals to the arbiter that the master which is attempting the transfer should not be granted access to the bus until the slave indicates it is ready to complete the transfer. Therefore the arbiter is responsible for observing the response signals and internally masking any requests from masters which have been *SPLIT*. During the address phase of a transfer the arbiter generates a tag, or bus master number, on ***hmaster* [1:0**] which identifies the master that is performing the transfer. Any slave issuing a *SPLIT* response must be capable of indicating that it can complete the transfer, and it does this by making a note of the master number on the ***hmaster* [1:0]** signals. Later, when the slave can complete the transfer, it asserts the appropriate bit, according to the master number, on the ***hsplitx* [1:0]** [Refer Appendix A] signals from the slave to the arbiter. The Figure 3 shows the AHB slave interface synthesized in this work.

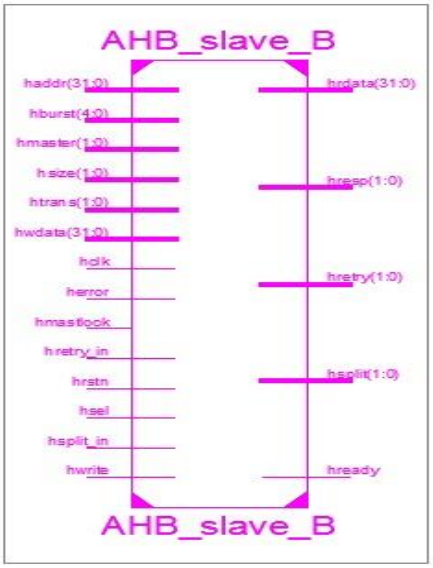

Fig 3 Schematic of AMBA AHB Slave

The design of AHB slave for this paper is quite different as it contains some extra signals which are considered to be the signals of the device attached to the AHB slave; Table 3 shows the extra signals taken in the slave interface.

Table 3 the extra signals taken in the Slave Interface

EXTRA SIGNAL ADDED TO THE INTERFACE	*TYPE*	*EXPLANATION*
herror	input	When herror is high it indicates that the transfer did not take place properly and an error has occurred.
hsplit_in	input	This signal invokes the split condition when high.
hretry_in	input	This signal invokes the retry condition when high.

Finite state machine of AHB Slave interface

The Finite State Machine (FSM) of the Slave is designed using Mealy machine, in which the output of the states depend on both the input and the current state of the machine. The different states of the FSM for the slave is given in Table 4 and Figure 4 shows the FSM of the AHB slave interface developed for this paper.

Table 4 The states of AMBA AHB Master Module

STATES	*STATE ASSIGNMENT*
IDLE	0
ADDRPHASE	1
WRDATAPHASE	2
RDDATAPHASE	3
ERRORSTATE	4
SPLITSTATE	5
RETRYSTATE	6

Fig 4 FSM of AHB Slave interface

The working of the Slave FSM is as follows:

- ***IDLE:*** the slave is in IDLE state before the start of a transfer, when the slave module is selected with the help of the hsel signal from the decoder unit it jumps to the next state i.e. ADDRPHASE, but if the hsel signal to the particular slave module is low (0) it stays in the IDLE state itself. Initially before the start of any transfer the default output of slave module will be hready signal 1 and hresp 00. when the transfer starts the slave drives the hready signal to 0 and will make it 1 when the transfer finishes.
- ***ADDRPHASE:*** this state is called address phase, here after being activated the slave checks the address signal haddr given by the master module and tells the master whether this address location is a valid address location or not, if it's a valid location it checks for the hwrite signal from the master if its high it jumps to WRDATAPHASE, if the address given is an invalid location it generates an error response hresp and goes to ERRORSTATE.
- ***WRDATAPHASE:*** this state is called write data phase state, the slave module checks the hwrite signal in the ADDRPHASE and if its 1 jumps to this state.in this state according to the control signals and address generated by the master module the slave performs the write operations, here a counter is internally generated which determines when the data transfer ends and the ready signal is driven to high again, the counter in this state is working with respect to the burst generated by the master module. If split, retry or error conditions are invoked in this state the slave goes to the corresponding SPLITSTATE, RETRYSTATE or ERRORSTATE.
- ***RDDATAPHASE:*** this state is called read data phase state, the slave module checks the hwrite signal in the ADDRPHASE and if its 0 jumps to this state. In this state according to the control signals and address generated by the master module the slave performs the write operations, here a counter is internally generated which determines when the data transfer ends and the ready signal is driven to high again, the counter in this state is working with respect to the burst generated by the master module. If split, retry or error conditions are invoked in this state the slave goes to the corresponding SPLITSTATE, RETRYSTATE or ERRORSTATE.
- ***SPLITSTATE:*** the slave enters the SPLITSTATE when the hsplit_in is made high, the slave IP decides when to give the split transfer, and the split transfer is given in order to reduce the traffic on a particular slave. When hsplit_in is made high during WRDATAPHASE the slave immediately gives the hresp 11and hready is made high or remains low depending on if any other master wants to access the slave . If a new master access the slave the ready is made high then goes low else it remains low. when the split function is invoked the slave will remain in split state for a fixed amount of time after that it goes back to idle state while entering split state the slave indicates to the arbiter for which master its issuing the split through hsplit [1:0] signal with the help of which the arbiter removes the grant for the particular master. The master gets the grant back it performs the task from the point when the split was invoked and completes the transfer. While split is invoked any master can access the bus , here priority of the master does not matter, once the split condition is over then only the master that was given the split can continue the transfer.
- ***RETRYSTATE:*** this state is similar to SPLITSTATE with a difference that during RETRYSTATE only masters with a higher priority than the current master can access the bus. The slave enters the RETRYSTATE from the WRDATAPHASE when the hretry_in signal is made high; it indicates the slave IP communicating through the interface that the current transaction didn't perform properly so send it again. Here also the slave stays for a predetermined time and then jumps to IDLE state, or if a master with higher priority than the current master access the bus it jumps to ADDRPHASE.

IV. RESULT ANALYSIS

After designing the finite state machine of the AHB master and slave, Verilog hardware description language is used to implement it and check its functionality and correctness. In this paper the simulation is done with the help of Modelsim 10.03a and the synthesis is done with the help of Xilinx ISE design tool.

AHB Master

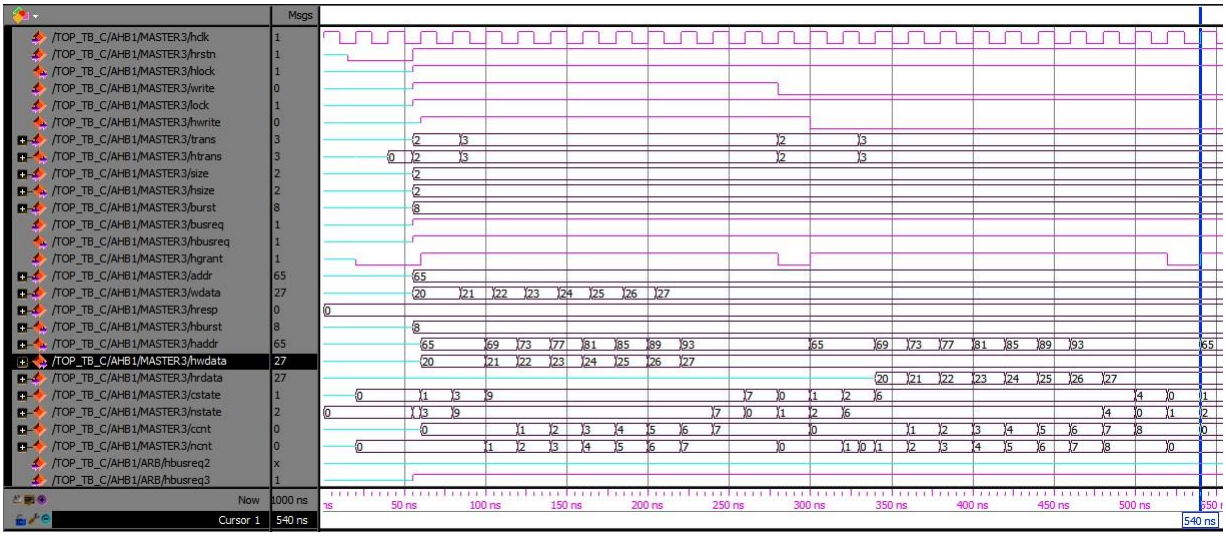

Fig 5 simulation result of AHB master during write and read transfer.

Figure 5 shows the simulation result of AHB Master for simple read and right transfer all the state changes happening in the FSM are also shown in Fig 5.

Figure 6 shows the simulation result of AHB Master for simple write and read transfer with Split and Retry condition invoked by the slave interface.

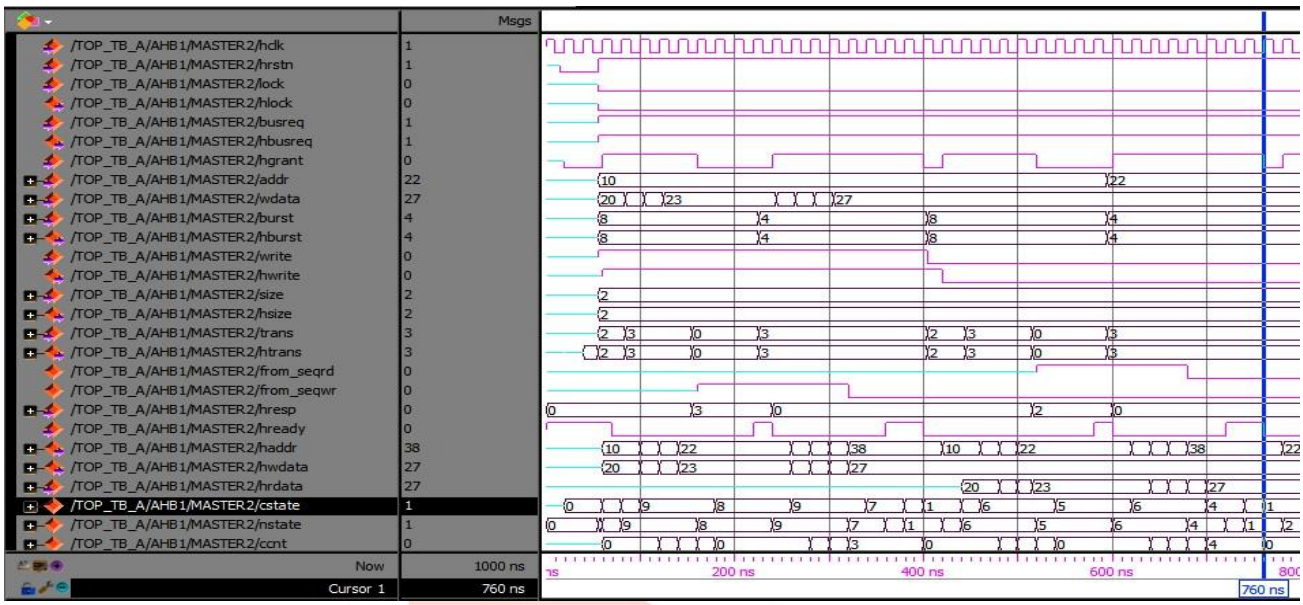

Fig 6 simulation result of AHB master during write and read transfer with split and retry condition invoked.

AHB Slave

Figure 7 shows the simulation result of AHB Master for simple read and right transfer all the state changes happening in the FSM are also shown in Fig 7.

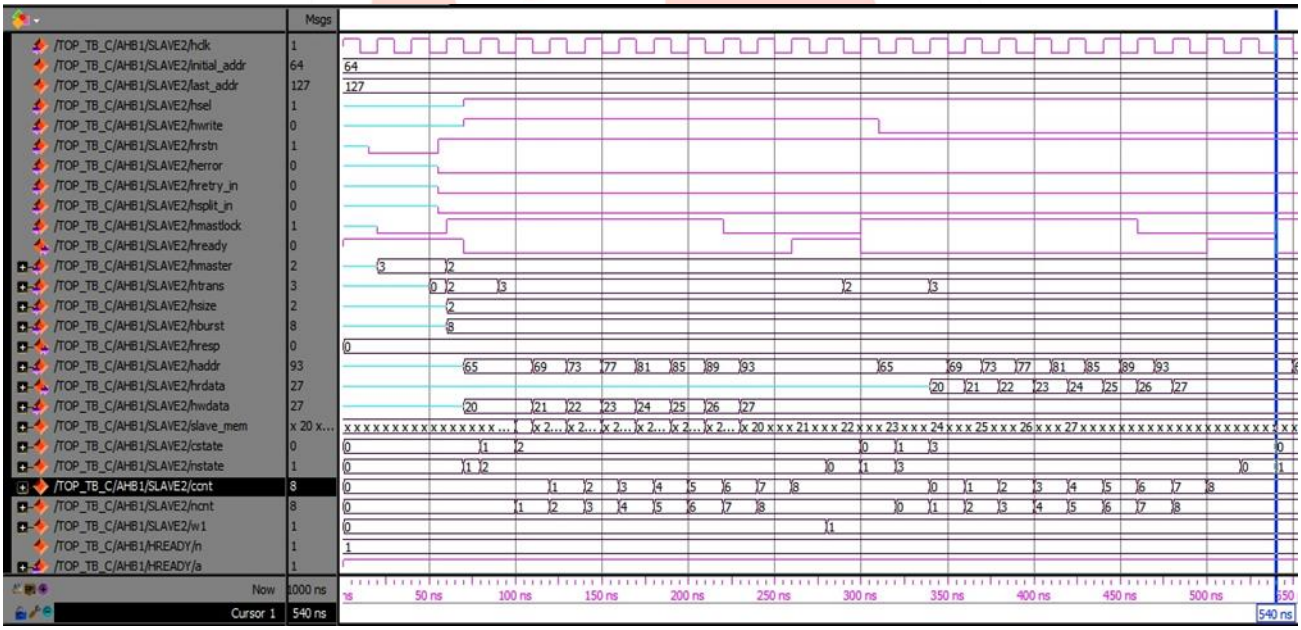

Fig 7 simulation result of AHB slave during write and read transfer.

Figure 8 shows the simulation result of AHB slave for simple write and read transfer with Split and Retry condition invoked by the slave interface.

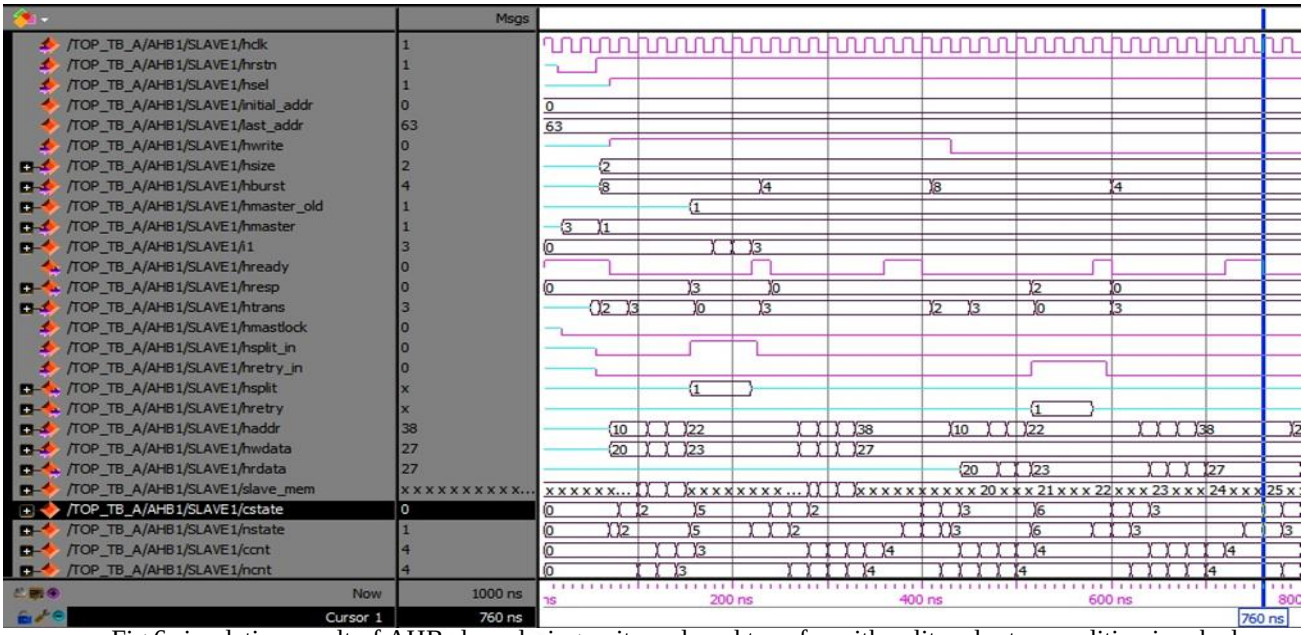

Fig 6 simulation result of AHB slave during write and read transfer with split and retry condition invoked

Synthesis Result of AHB Master And AHB Slave Interface

Here Vertex 5 package is used in which The Selected device is XC5vlx50t-2ff1136 for which the synthesis report has been obtained.

Device utilization and timing results of master interface

The Table 5 gives the information about the device utilization in terms of certain parameters which are important to be considered for the synthesis of the design.

Table 5 device utilization of AHB master interface

LOGIC UTILIZATION	*USED*	*AVAILABLE*	*UTILIZATION*
Number of Slice Registers	65	28800	0%
Number of Slice LUTs	511	28800	1%
Number of fully used LUT-FF pairs	62	514	12%
Number of bonded IOBs	158	480	32%
Number of BUFG/BUFGCTRLs	3	32	9%

Table 6 gives the timing analysis obtained during synthesis for the master interface

Table 6 Timing summary for the AHB Master module

TIMING	*VALUE*
Minimum period	6.173ns
Minimum input arrival time before clock	8.341ns
Maximum output required time after clock	8.965ns
Maximum combinational path delay	9.526ns

Device utilization and timing results of slave interface

The Table 7 gives the information about the device utilization in terms of certain parameters which are important to be considered for the synthesis of the design.

Table 7 device utilization of AHB slave interface

LOGIC UTILIZATION	USED	AVAILABLE	UTILIZATION
Number of Slice Registers	8270	28800	28%
Number of Slice LUTs	11276	28800	39%
Number of fully used LUT-FF pairs	8263	11283	73%
Number of bonded IOBs	119	480	24%
Number of BUFG/BUFGCTRLs	32	32	100%

Table 8 gives the timing analysis of the slave module during synthesis.

Table 8 Timing summary for the AHB Master module

TIMING	VALUE
Minimum period	1.112ns
Minimum input arrival time before clock	5.026ns
Maximum output required time after clock	3.044ns
Maximum combinational path delay	No path found

V. CONCLUSION

The AHB master interface and slave interface are designed using the finite state machines in Verilog hardware description language and the design is simulate with the help of Mentor graphics tool Modelsim 10.03a and the synthesis of the design is done in Xilinx ISE design tool. The completed AMBA AHB system is then checked for proper lossless communication between master and slave interface.

REFERENCES

[1] AMBA Specification (Rev 2.0) © Copyright ARM Limited 1999. ARM IHI 0011A, Chapter1, 2, 3.

[2] AHB Example AMBA SYSTEM Technical Reference Manual, Copyright © 1999 ARM Limited. DDI0170A

[3] AMBA 3 AHB-Lite Protocol Specification, Copyright © 2001, 2006 ARM Limited. ARM IHI 0033A.

[4] SystemVerilog for Verification: A Guide to Teach the Testbench Language Features Third Edition".

[5] Manish A Lakhiyar and Mitesh M Dalwadi, "*Implementation of Different Operations for Data Transfer for Amba-Advanced High Performance Bus*", International Journal of Futuristic Science Engineering and Technology, Vol 1 Issue 3 March 2013,

[6] Rishabh Singh Kurmi and Shruti Bhargava, "*Design of AHB Protocol Block for Advanced Microcontrollers*", International Journal of Computer Applications (0975 – 8887) Volume 32– No.8, October 2011, Page 23-29.

[7] Soo Yun Hwang, and Kyoung Sun Jhang, "*An Ameliorated Design Method of ML-AHB BusMatrix*", ETRI Journal, Volume 28, Number 3, June 2006.

[8] Dr.R.Shashikumar and 2C.N. Vijay Kumar, "AHB Compatible DDR SDRAM Controller IP Core for ARM BASED SOC", (IJCSIS) International Journal of Computer Science and Information Security, Vol. 7, No. 1, 2010.

[9] Mohamed Ben-Romdhane Sudeep Pasricha, Nikil Dutt. Fast exploration of bus-based on-chip communication architectures. In *CODES and ISSS*, Stockholm, Sweden, September 2004.

[10] Thorsten Gr¨otker, Stan Liao, Grant Martin, and Stuart Swan. "*System Design with SystemC*." Kluwer Academic Publishers, 2002.

[11] Yashdeep Godhal, Krishnendu Chatterjee, Thomas A.henzinger. "Synthesis of AMBA AHB from Formal Specification: A Case Study, IST Austria (Institute of Science and Technology Austria)

[12] Manish A Lakhiyar, Mitesh M Dalwadi, Dharmesh N Kandhar. "Implementation of Different Operations for Data Transfer for AMBA AHB" ISSN: 0975-6779|NOV 12 to OCT 13|Volume -02, ISSUE-02.Page 679-683.

[13] Bhaumik Vaidya, Anupam devani. "Design of an Efficient Finite state Machine for the Implementation of AMBA AHB Master" Journal of information, Knowledge and Research in Electronics and Communication Engineering ISSN: 0975-6779|NOV 12 to OCT13| volume-02, Issue-02, Page484-487.