ISSN: 0975-766X
CODEN: IJPTFI
Available Online through *Research Article*
www.ijptonline.com

VERIFICATION OF MASTER-SLAVE USING AMBA AXI-OCP PROTOCOL

Satheesh Kumar.S, Venugopal P, Rajeev Pankaj Nelapati
School of Electronics Engineering, VIT University, Vellore, Tamilnadu, India.
Email: rajeevpankaj@vit.ac.in

Received on 12-07-2016 *Accepted on 28-08-2016*

Abstract

Bus protocol plays a vital role in System on chip(SOC) design. SOC's is a co-ordinatnation of various intellectual property(IP) blocks, communications are handled by utilizing the proprietary/Non proprietary protocol like AXI and OCP. In this paper we are going to deal how two different protocols(AXI and OCP) are connected with the help of bridge by the name AXI-OCP bridge . Different test cases scenario is developed in order to detect functional failures.

Our methodology robustness and effectiveness is demonstrated from the obtained results by the improvement of code coverage from the 82% to 91%.Which shows the improvement of 9%. As a result this bus protocol/SOC interconnects guarantee proper communication in between the IP cores in a SOC.

Keywords: AXI, OCP, Master, Slave, Bridge.

1. Introduction

This paper presents all about AXI-OCP Protocol(Master and Slave) which is designed and in which communication will takes place in between two different bus protocol such as AXI Master(An ARM proprietary high frequency ,high performance bus protocol) and OCP Slave (a non-proprietary,high performance bus independent protocol)which is designed using a bus bridge. This methodology supports OCP and AXI features like simple burst, pipelined and tagged write-read operation which is implemented using a system Verilog[4].By studying the single set of these both the protocol, and with the help of bus bridge(for signal conversion from one protocol to other AXI-OCP and vice versa).All these features are verified using a Questa Sim tool from Mentor Graphic. The communication on a SOC is handled by the bus protocol of intellectual property. A bus protocol works on the handshaking mechanism like grant-request form which ensures in proper SOC integration. If there is change in bus protocols of Intellectual Property then the maste,Slave

functionality does not match with each other. So this problem is solved by the bus bridge[11]. This work will explain how two different protocol will interact with each other being two different IP's in which one is Master and another is Slave.AXI(MASTER/initiator) and OCP(Slave/target) which are interconnected via AXI-OCP bridge(signal converter from one protocol to anther)[15]. The biggest task for a SOC engineers is to integrate the different IP's from different vendors. Due to evolvement of different IP's in the field of SOC there is a need for a standard like OCP which act as a common interface and communication protocol between ipcores and SOC interconnect

2. Problem formulation

The test bench comprises of AXI(Master), OCP(Slave) ,[1],[2],[6],[10]. AXI-OCP bus bridge and memory. It also ensures that it is working properly. Different test cases scenario are developed in order to detect Functional failuresAXI-OCP bus bridge will helps in signal conversation from AXI protocol to OCP protocol and vice versa[2],[8],[4].. Due to increase in more intellectual property(IP) blocks on a chip, circuit complexity in modern SOC design has been increased. The whole test bench is portioned into top 3 module by the name AXI,OCP and AXI-OCP bridge. In the upcoming section a detailed concept highlighting AXI in section 3, OCP in section 4 and AXI-OCP bridge in section 6 will be covered

3. AMBA-AXI(Advanced Microcontroller Bus Architecture-Advanced eXtensible Interface) Protocol

The amba AXI protocol is a high performance and high frequency bus protocol and which includes a number of features in it[11]. .Every AMBA-AXI transaction have a control , data and address information. The data transaction takes place in between master and slave using a write data channel to the slave or a read data channel from the slave to the master[15]. AMBA-AXI have 5 independent channels which are divided into 2 category write channel and read channel. There are three write channels they are write address channel, write data channel and write response channel. And two read channels Read address channel and read data channel[6].

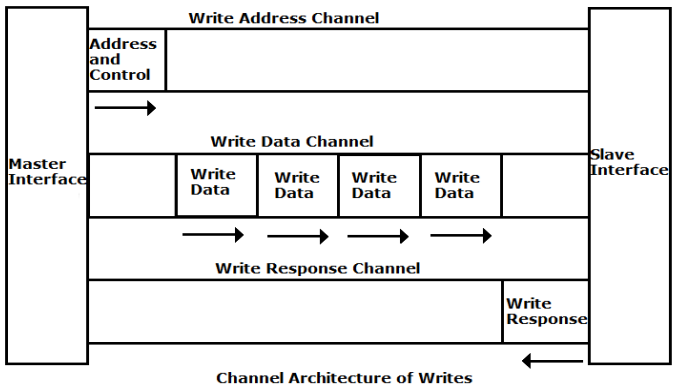

Fig 1:Write Address followed by Write data and acknowledged by slave with response channel.

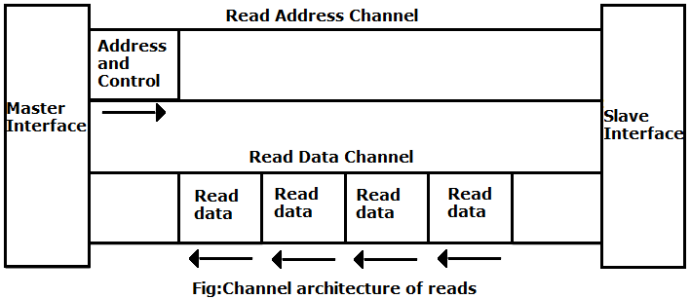

Fig2: Read address followed by read data channel.

3.1. Write Address Channel

This channel Utilize the Control signals like AWVALID, AWREADY and AWADDR. Master will send the address(AWADDR) and along with that it will asserting an AWVALID signal, this signal will indicate the presence of valid Address on an address bus. After receiving the AWADDR and AWVALID it acknowledge the master by sending the AWREADY which indicates that it is prepared to accept the address from master. By this way write address transaction will be completed.

3.2. Write data Channel:

This channel Utilize the Control signals like WVALID, WREADY and WDATA. Master will send the data(WDATA) and along with that it will asserting an WVALID signal ,this signal will indicate the presence of valid WDATA on an data bus.

After receiving the WDATA AND WVALID it acknowledge the master by sending the WREADY which indicates that it is prepared to accept the data from master[5]. BY this way the write data transaction will be completed.

3.3. Write response Channel:

This channel Utilize the Control signals like BRESP,BVALID and BREADY. After the write address and write data transaction completes from master side the slave will respond to it by sending the response signal like BVALID and BRESP in turn master will acknowledge by BREADY signal.

3.4. Read Address Channel:

This channel Utilize the Control signals like ARVALID,ARREADY and ARADDR. Master will send the address(ARADDR) and along with that it will asserting an ARVALID signal, this signal will indicate the presence of valid Address on an address bus[3][4][5][6]. After receiving the ARADDR and ARVALID it acknowledge the master by sending the ARREADY which indicates that it is prepared to accept the address from master[2]. BY this way read address transaction will be completed.

3.5. Read data Channel:

This channel Utilize the Control signals like RVALID, RREADY and RDATA. slave will send the data(RDATA) and along with that it will asserting an RVALID signal ,this signal will indicate the presence of valid RDATA on an data bus[2][11][15]. After receiving the RDATA and RVALID to master it acknowledge the slave by sending the RREADY which indicates that it is prepared to accept the data from slave. BY this way read data transaction will be completed.

3.6. Transaction channel handshake pairs

Transaction channel	Handshake pair
Write address channel	AWVALID ARREADY
Write data channel	WVALID WREADY
Write response channel	BVALID, BREADY
Read address channel	ARVALID, ARREADY
Read data channel	RVALID,READY

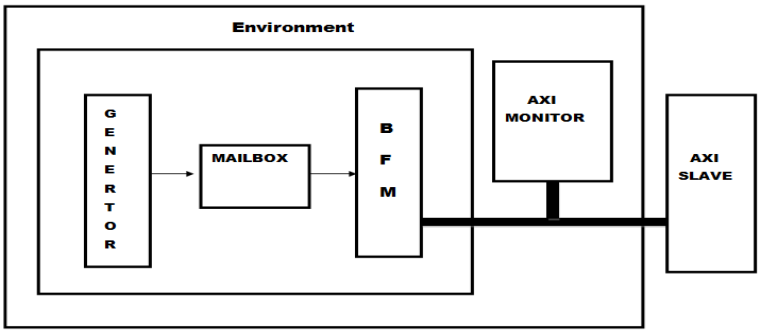

Fig3.Architecture of AXI verification intellectual property.

AXI Module Consists Of

1. Generator
2. Mailbox
3. BFM
4. AXI Monitor
5. Driver
6. Slave

Generator: The Generator will generate all the random test case to be tested.

Mailbox: All the test transaction which is generated by the generator will be of high transaction rate so in order to synchronize with the lower frequency module(slave mail box act as storage.

BFM: Bus Functional Module which drives the transaction generated from the virtual interface to the physical interface

Monitor: This will take the care of slave and master transaction and verify it

Slave: this is the place all transaction are going and retrieving for the master

Driver: Which drive the signals from virtual interface to the physical interface

4. Open Core Protocol(OCP)

The open Core protocol(OCP) is a not-for-profit,transaparently,core-centric protocol that has an capacity of describing integration requirements of IP's cores in System on chip.OCP along with the dealing the data flow between intellectual property(IP) cores, it binds all inter-core communications like test signal and sideband signals. Because of all these reason we can say OCP provides a high performance ,bus-independent interface between IP cores which will reduce the manufacturing costs , design time and design risk for SOC designs. A built up point to point interface between any two IPs cores and bus interface module will have one of the two IP cores acting as an OCP slave and another as a OCP master[13][14][15]. The master starts initiating the command and slave in turn reacts to it by accepting data from the master(for write),or transferring the data to the master(for read).

In this the system initiator act as an OCP master which issues a command and its associated request to the connected slave(bus initiator).The command request is placed across the on chip bus system. System will target the (OCP slave) and receives the command and takes the necessary action.

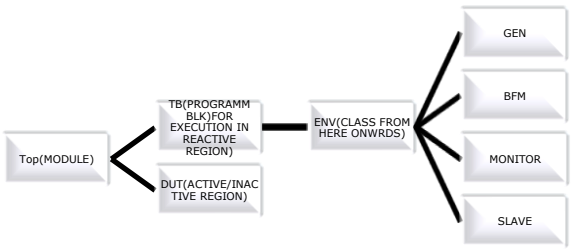

Fig 4. Testbench Architecture.

An open, non-benefit and configurable standard protocol like OCP for IP block interface is rapidly getting to be important for productive on-chip interconnection configuration and SOC coordination. Even in case of native protocol differs, an OCP complaint bus bridges will help to overcome the in compatibility between IP cores

The initaitor has an AXI master and OCP Slave and communications happens through handshaking mechanism by request-grant.Figure 2 which it contain a OCP complaint(master) and other side core having an OCP complaint (slave).The OCP master can send write/read request by presenting a request command from the request line to slave. Along with the request command master presents will send the data lines to the slave .In turn slave will acknowledge the master request by accepting the request from master by registering the request information.Slave internally will presents the request to the core2 and then it wait for the response from core2[9].

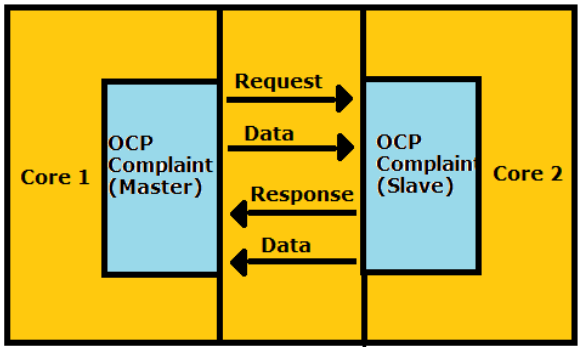

Figure 5: contain a basic building block of open core protocol(OCP).

After the request is processed in the core2, slave will send the response information and response status to the master through the response line and data line.

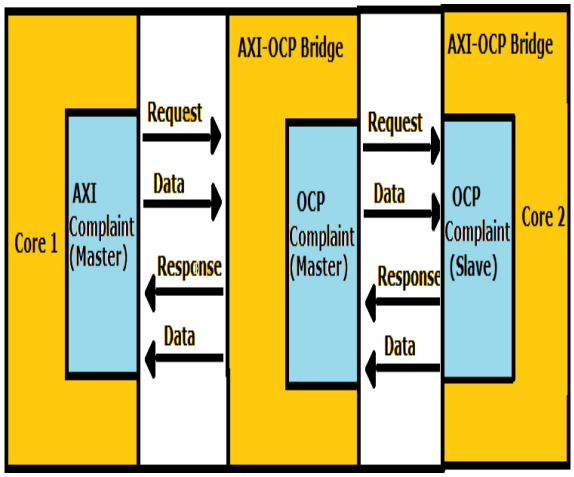

Figure 6: shows the AXI-OCP bridge with an AXI(master) in core1.

Which will generates the request for transaction and sends to the bus bridge through the 'request' and 'data' lines. The bus bridge will accepts the transaction request from the AXI and convert the signal into the OCP protocol format. In turn the OCP master generates the valid OCP request. These OCP request command and its data are forwarded to the core2 OCP slave by the bus bridge through the data and request lines. The OCP slave will accepts the request from the OCP compliant master and sends back the response information and response status via reponse and data lines. The AXI-OCP

bus bridge will accept the the OCP response signal and convert them to valid AXI response signal. By this way OCP complaint bus bridge will act as an efficient communication between the two protocols.

The basic signals in the master side consist of Write/Read address(MAddr),Write data(MData),write command(MCmd) and slave signals consist of response(SResp),response data(SData),acknowledgment(SCmdAccept).On the basis of connected core the requirement the configurable width of a signal depends on the connected core requirement .The three bit MCmd will carry demand information such as write , read , readex ,read linked , writeNonpost ,Write Conditional and Broadcast .The two bit SResp will carry the response status information such as NULL, DVA, FAIL and ERR .The basic signals act as fundamental signal for building a simple OCP master-slave Unit

Table.1 OCP design specifications

Sl.no	Signal Name	Width	Driver	Function
1	MAddr	5	Master	Transfer addres
2	MData	5	Master	Write data
3	MCmd	3	Master	Transfer Command
4	MBurstlength	3	Master	Burst length
5	MBurstPrecise	1	Master	Given burst length is precise
6	MReqlast	1	Master	Last request is burst
7	MTagid	3	Master	Request tag id
8	SCmdaccept	1	Slave	Slave accepts transfer
9	SResp	2	Slave	Transfer response
10	SData	5	Slave	Read data
11	SRes	1	Slave	Last response in burst
12	STagid	3	Slave	Response tag id

OCP Burst Extensions:

Burst transaction means for a single address sending numerous amount of data in order to achieve high transfer efficiency. The burst request information is carried with OCP burst extension signals ,which incorporates number of burst transfers(MBurstLength),address sequence for a burst requests (MBurstSeq) , precise / imprcise sort of burst transfer (MBurstPrecise),indication of last request for the burst(MReqLast), and indication of last response for the burst request(SReqLast).For a precise/exact type of burst (MBurstPrecise='1') which indicates the total number of transfer or burst length is steady/constant in the throughout the burst and an impress burst type(MBurstPrecise=0) which indicates that the number of transfer keeps varying throughout the burst[6]. For an imprecise kind of burst transfer ends when the master places the burst length equals to one. The MBurst Sequence indicates the sequence of address during an burst such as incremental/wrapping/exclusive/ streaming lind of address.

OCP tag extension tags

It is used for the controlling of out of order responses. The extensions of OCP tag includes response tag ID(STagID)and request tag ID(MTagID)[8][9]. The master will generates the tag ID for a request and in repose to master request slave will the acknowledgment with response associating the same tag ID

Simulation Result of simple Read-Write Operation

In broad level in order to setup the functionality verifying the OCP master-Slave block is as shown below in the figure 6.The top rtl level for OCP will be as shown in appendix A. The OCP master will get the request signal from the testbench. The master will send the request to the slave and in turn slave will acknowledge the master by performing its write/read transaction on the memory[1]. For even simple write-read operation the test bench need the controlling signals like wreq, rreq, maddrin and mdatain. The simulation results are shown below in the figure 7.

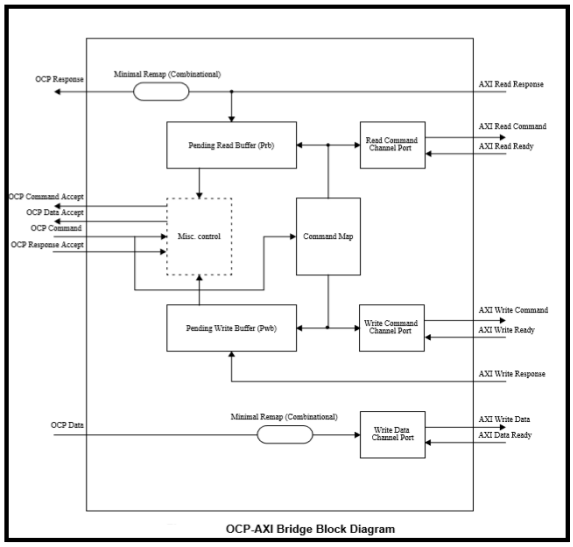

Figure 7 shows the AXI-OCP bridge.

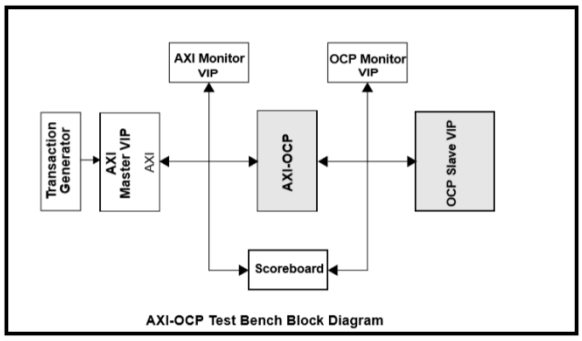

fig 8:AXI-OCP.

Output port:

The bridge consist of 3 output ports to the amba-AXI interface one each for the write address channel, write data channel, read address channel. The 3 output ports are identical with respect to the logic point of view but the difference is with

respect to port width. Pending read buffer: it can track the 16 pending read transacation.each and every read request are entered into the pending read buffer. When the last read transaction are requested is received from the pending read buffer the associated entry is deleted.

Pending write buffer: This is used to keep track of the latest 8 entries of the write transaction. This will store the address of the request and associated burst length will be recorded. when the competition response for the last write transaction is requested it will get deleted from that.

Simulation Results

Write Transaction only

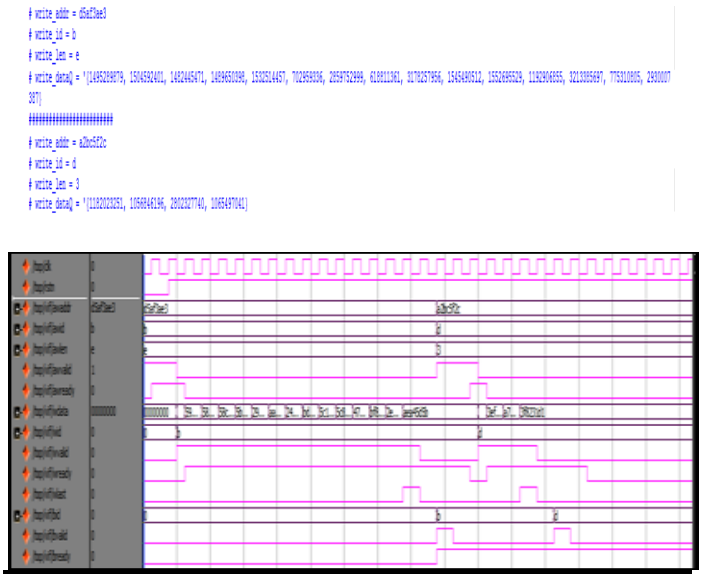

fig 9.Write Address Simuation.

Write-Read Transaction

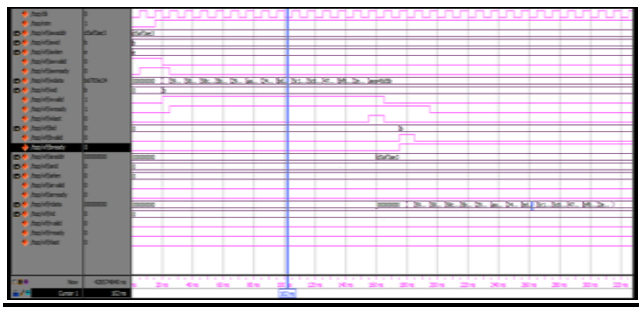

fig 10.Write_Read Simulation

Conclusions

the OCP designed is simple burst, precise imprecise burst ,tags and pipeline are functionally proper with respect to the data that are written to the memory which are successfully read back. By the overall result we can conclude that AXI transaction are converted into OCP transaction with the help of AXI-OCP bridge and vice versa.

Recommendations for Future Work

Data handshaking signals are used with along the normal signals for setting a high performance.OCP Burst of operation can be extended up to different burst sequence like fifo burst and wrap burst etc.

ocp tag are implemented with the upgrade with sideband signals that the master can request with the out of order transaction for ocp arbiter with multiple masters.

References

1. Chih-Wea Wang, Chi-Shao Lai, Chi-Feng Wu, Shih-Arn Hwang and Ying-Hsi Lin, On-chip Interconnection Design and SoC Integration with OCP, IEEE International Symposium on VLSI Design, Automation and Test, pp: 25-28, April 2008
2. Krishnan Srinivasan and Erno Salminen, A Methodology for Performance Analysis ofNetworkon-Chip Architectures for Video SoC, OCP-IP, April 2009
3. OCP-IP, Open core protocol Specification v2.1, 2005. Last accessed on 01/09/2009 from http://www.ocpip.org/
4. Shihua Zhang, Asif Iqbal Ahmed and Otmane Ait Mohamed, A Re-UsableVerification Framework of Open Core Protocol (OCP), IEEE North-East Workshop on Circuits and Systems, pp: 1-4, June 2009
5. ARM, AMBA AXI Protocol Specifcation v1.O, 2003-04. Last accessed on 08/09/2009 from http://www.arm.com/
6. Derek Graham, Scott Roy and Fernando Rodriguez, A low-tech solution to avoid the severe impact of transient errors on the IP interconnect, IEEE/IFIP International Conference on Dependable Systems & Networks, pp: 478-483, 2009
7. Konstantinos Aisopos, Chien-Chun Chou and Li-Shiuan Peh, Extending Open Core Protocol to Support System-Level Cache Coherence, IEEE International Conference on Hardware/Software codesign and system synthesis, pp: 167-172, 2008
8. Fu-ming Xiao, Dong-sheng Li, Gao-ming Du, Yu-kun Song, Duo-li Zhang and Ming- lun Gao, Design of AXI Bus Based MPSoC on FPGA, 3rd International Conference on Anti-counterfeiting, Security and Identification in Communication, pp: 560-564, August 200

9. Xiao, F., Dong-sheng, L., Gao-ming, D., Yu-kun, S., Duo-li, Z., Ming-lun, G.,2009, "Design of AXI Bus Based MPSoC on FPGA". 3rd International Conference on Anti-counterfeiting, Security and Identification in Communication (ASID), Hong Kong, pp. 560-564.
10. Anurag Shrivastavastava, G.S. Tomar, Kamal K Kalra, 2010 "Efficient Design and Performance analysis for AMBA bus Architecture based System-on-Chip" International Conference on Computational Intelligence and Communication Systems
11. Sanghun Lee, Hyuk-Jae Lee, "A new multi-channel on-chip-bus architecture for system-on-chips," in Proceedings of IEEE international SOC Conference, September 2004.
12. Xiongfei Liao, Jun Zhou and Xin Liu, "Exploring AMBA AXI On-Chip Interconnection for TSV-based 3D SoCs"
13. Salita Sombatsiri, Kazuhiro Kobashi, Keishi Sakanushi, Yoshinori Takeuchi and Masaharu Ima, "An AMBA Hierarchical Shared Bus Architecture Design Space Exploration Method Considering Pipeline, Burst and Split Transaction".
14. Anurag Shrivastava, G.S. Toma, Ashutosh Kumar Singh 2011, "Performance Comparison of AMBA Bus-Based System-On-Chip Communication Protocol", International Conference on Communication Systems and Network Technologies.

Corresponding Author:
Rajeev Pankaj Nelapati
Email: *rajeevpankaj@vit.ac.in*