International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-1 Issue-5, October 2012

Open Core Protocol for High Performance on Chip Bus

Asha latha P, Rambabu B

Abstract: ***The need for on-chip bus protocols are increased drastically for efficient and lossless communication among large number of IP cores of SOC design. This paper proposes a high-performance, highly scalable, bus-independent interface between IP cores named as Open Core Protocol-International partnership. The Open Core Protocol (OCP) is a core centric point to point protocol which provides lossless communication and reduces design time, design risk, and manufacturing costs for SOC designs . Main property of OCP is that it can be configured with respect to the application required. The OCP is chosen because of its advanced supporting features such as configurable sideband control signaling and test harness signals, when compared to other core protocols. The OCP defines a point-to-point interface between two communicating entities such as IP cores and bus interface modules. One entity acts as the master of the OCP instance, and the other as the slave .In this paper, the most efficient bus architecture was adopted to support most advanced bus functionalities including simple transactions, burst transactions, lock transactions, pipelined transactions, and out-of-order transactions withrespect to its suitable application in the real time product. The Open Core Protocol (OCP) was designed and the hardware modeling for that architecture was done using VHDL. This design is Simulated and Synthesized. An experimental result shows the efficiency of the proposed bus architecture and interface.***

Key words: Open core protocolIP Core,SOC,Vhdl

I. INTRODUCTION

In recent days, the development of SOC chips and the reusable IP cores are given higher priority because of its less cost and reduction in the period of Time-to-Market. So this enables the major and very sensitive issue such as interfacing of these IP cores. These interfaces play a vital role in SOC and should be taken care because of the communication between the IP cores property. The communication between the different IP cores should have a lossless data flow and should be flexible to the designer too. The design of on-chip buses can be divided into two parts: bus interface and bus architecture. The bus interface involves a set of interface signals and their corresponding timing relationship, while the bus architecture refers to the internal components of buses and the interconnections among the IP cores. The AMBA AHB[2], which is mainly a shared bus composed of multiplexors, it can be permitted to a design with small number of IP Cores. When the IP Cores increases then the overall performance can be reduced.

In order to improve the communication efficiency among the large no of IP Cores two more Protocols have been proposed. One is Advanced extensible Interface protocol (AXI) [3], proposed by the ARM company. AXI defines five independent channels (write address, write data, write response, read address, and read data channels). Each channel involves a set of signals. AXI does not restrict the internal bus architecture and leaves it to designers. Thus designers are allowed to integrate two IP cores with AXI by either connecting the wires directly or invoking an in-house bus between them. The AXI divides the address channel into independent write address channel and read address channel such that read and write transactions can be processed simultaneously. However, the additional area of the separated address channels is the penalty. The other bus interface protocol is proposed by a non-profitable organization, the Open Core Protocol – International Partnership (OCP-IP) [1]. OCP is an interface (or socket) aiming to standardize and thus simplify the system integration problems. It facilitates system integration by defining a set of concrete interface (I/O signals and the handshaking protocol) which is independent of the bus architecture. Based on this interface IP core designers can concentrate on designing the internal functionality of IP cores, bus designers can emphasize on the internal bus architecture, and system integrators can focus on the system issues such as the requirement of the bandwidth and the whole system architecture. In this way, system integration becomes much more efficient.

II. OPEN CORE PROTCOL (OCP)

The Open Core Protocol (OCP) is a core centric protocol which defines a high-performance, bus-independent interface between IP cores that reduces design time, design risk, and manufacturing costs for SOC designs. Main property of OCP is that it can be configured with respect to the application required. The OCP is chosen because of its advanced supporting features such as configurable sideband control signaling and test harness signals, when compared to other core protocols. The OCP defines a point-to-point interface between two communicating entities such as IP cores and bus interface modules. One entity acts as the master of the OCP instance, and the other as the slave. Only the master can present commands and is the controlling entity. The slave responds to commands presented to it, either by accepting data from the master, or presenting data to the master. For two entities to communicate there need to be two instances of the OCP connecting them such as one where the first entity is a master and one where the first entity is a slave.

Revised Manuscript Received on 30 October 2012
*****Correspondence Author(s)
P. Asha latha, Department of ECE, JNTUK University, Kaushik College of Engineering, Visakhapatnam, India.
Ram Babu B, Assoc. Professor, Department of ECE, JNTUK University, Kaushik College of Engineering, Visakhapatnam, India.

This high-performance on-chip bus design with OCP as the bus interface. We choose OCP because it is open to the public and OCP-IP has provided some free tools to verify this protocol. Our proposed bus architecture features crossbar/partial-crossbar based interconnect and realizes most transactions defined in OCP, including 1) single transactions, 2) burst transactions, 3) lock transactions, 4) pipelined transactions, and 5) out- of-order transactions. In addition, the proposed bus is flexible such that one can adjust the bus architecture according to the system requirement.

III. HARDWARE DESIGN OF THE ON-CHIP BUS

The architecture of the proposed on-chip bus is illustrated in FIGURE 1.This crossbar architecture is employed such that more than one master can communicate with more than one slave simultaneously.

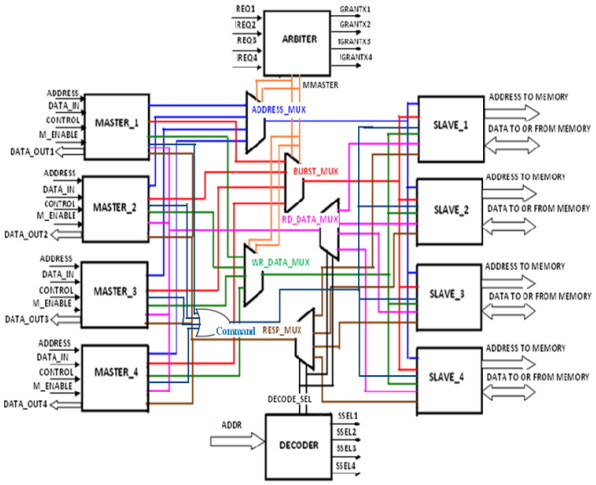

FIGURE1: Bus Architecture

The main blocks of the Bus architecture

A) OCP Arbiter Arbiter inputs are the request signals from the all four master's which are MReq1, MReq2, MReq3 and MReq4. Arbiter will issue the MGarantx1, MGarantx2, MGarantx3 and MGarantx4 to any one of the master's. Whichever the master sent MReq signal that master will get the MGrantx signal. Arbiter gives MMaster signal to the Address mux, Burst mux and Write Data mux as a selection signal. Two other inputs to the Arbiter are MBurst and SCmd_Accept signals. MBurst signa1 comes from the master to arbiter specifies that how many time units arbiter should give grant signal to the master by making a MGrantx signal to High. SCmd_Accept signal come from the slave to the master to increment the counter depending on the number ofs write or read operation.

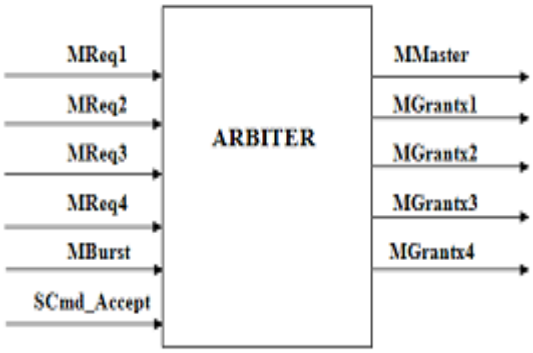

FIGURE1.1: OCP ARBITER

B) OCP Decoder:

In the ocp protocol decoder selects any one of the slave among four slaves depending on the MAddr signal come from the master to the decoder. The ocp decoder selects any one of the slave by making that slave SSelx signal to High.Decoder also gives another output signal Decode_sel, this signal becomes selection signal for the Resp and Read_Data_mux.

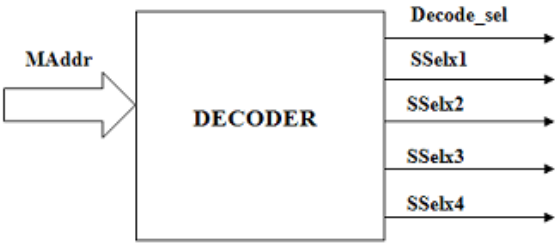

FIGURE 1.2: DECODER

C) MULTIPLEXERS:

There are five different types of multiplexors will be used are

Address Mux: It is used to select any one of the address which are coming from four different masters.

Burst Mux: It is used to select any one of the burst signals which are coming from four different masters.

Write data Mux : It is used to select the data signal which is coming from the enabled master to write into the defined address location.

Read data Mux: It is used to select the data signalwhich has to read by the slave from the location which is given by the master.

Response Mux: After the completion of each transaction the responded slave will give an acknowledgement to the master through this multiplexer.

IV. ON CHIP BUS FUNCTIONALITIES

The advanced bus functionalities supported by this protocol are

A) Simple transactions
B) Burst transactions
C) Lock transactions
D) Out-of-order transactions
E) Pipelined transactions

A) Simple transactions:

Allows the simple read and write operations.

B) Burst transactions:

The burst transactions allow the grouping of multiple transactions that have a certain address relationship, and can be classified into multi-request burst and single-request burst according to how many times the addresses are issued. FIGURE 2 shows the two types of burst read transactions. The multi-request burst as illustrated in FIGURE 2(a) where the address information must be issued for each command of a burst transaction (e.g., A11, A12, A13 and A14).

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

This may cause some unnecessary overhead. In the more advanced bus architecture, the single-request burst transaction is supported. As shown in FIGURE 2(b), which is the burst type defined in AXI, the address information is issued only once for each burst transaction

C) Lock transactions:

Lock is a protection mechanism for masters that have low bus priorities. Without this mechanism the read/write transactions of masters with lower priority would be interrupted whenever a higher-priority master issues a request. Lock transactions prevent an arbiter from performing arbitration and assure that the low priority masters can complete its granted transaction without being interrupted.

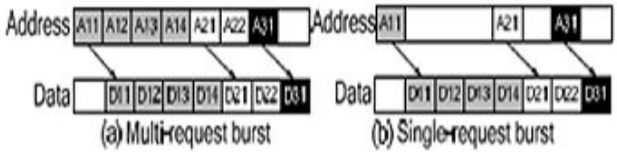

Figure 2 .Burst transactions

D) Out-of-order transactions:

The out-of-order transactions allow the return order of responses to be different from the order of their requests. hese transactions can significantly improve the communication efficiency of an SOC system containing IP cores with various access latencies as illustrated in FIGURE 3. In FIGURE 3(a) which does not allow out-of-order transactions, the corresponding responses of A21 and A31 must be returned after the response of A11. With the support of out-of-order transactions as shown in FIGURE3(b), the response with shorter access latency (D21, D22 and D31) can be returned before those with longer latency (D11-D14) and thus the transactions can be completed in much less cycles.

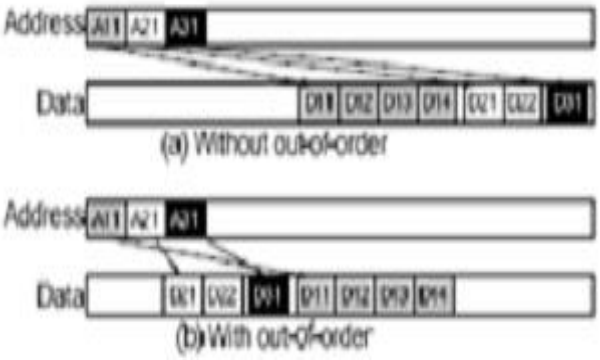

Figure 3.Out of order transactions

E) Pipelined transactions (outstanding transactions)

Figure 4(a) and 4(b) show the difference between non-pipelined and read transactions. In FIGURE 4(a), for a non-pipelined transaction a read data must be returned after its corresponding address is issued plus a period of latency. For example, D21 is sent right after A21 is issued plus t. For a pipelined transaction as shown in FIGURE8(b), this hard link is not required. Thus A21 can be issued right after A11 is issued without waiting for the return of data requested by A11 (i.e., D11-D14).

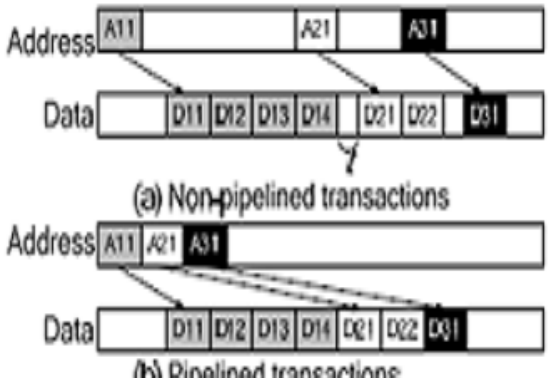

Figure4: Pipelined transactions

Table1: control inputs

Control	Command
000	Idle
001	SimpleWrite
010	Simple Read
011	Burst Write
100	Burst Read
101	Out of Order Write
110	Out of order Read

VI.EXPERIMENTAL RESULTS

The simulation results when 4 masters and 4 slaves are used are shown below where all masters can issue all transactions. In the figure5-for simple write operation the corresponding control input is 001 as shown in Table1 and the given data(mwdata signal) will be stored in the applied address location(maddr) .The selection of master will be done by high m_enable signal.Then the arbiter will generates M grantsignal for selected master. As maddr and mwdata signals passes through the multiplexors and converted to multiplexed signals. The decoder will select any one of the slave according to maddr,then ihe simple write operation is performed by this selected slave. After completion of transaction slave will send scmd accept signal as acknowledgement to the master. similarly for simple read operation the control input is 010 and m_addr is the location from where we need data. Slave will give scmdaccept signal to master after transaction. The slave will give read data as s_data signal to the master. .Figure6-shows burst operations. For burst write operation the control input is 011 and we have to give address and data signals, burst size also. For 4- burst ,ocp_top/size signal is 000 and for 8-burst 001 and for 16-burst 010 is the size. We have given address as 0000000000000 then 4 consecutive locations can be generated for 4-burst and the input data can be write in that locations. Similarly for burst read control signal is 100. The data in consecutive locations can be read by the slave which is selected by the decoder and will give that data as sdata to the master.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

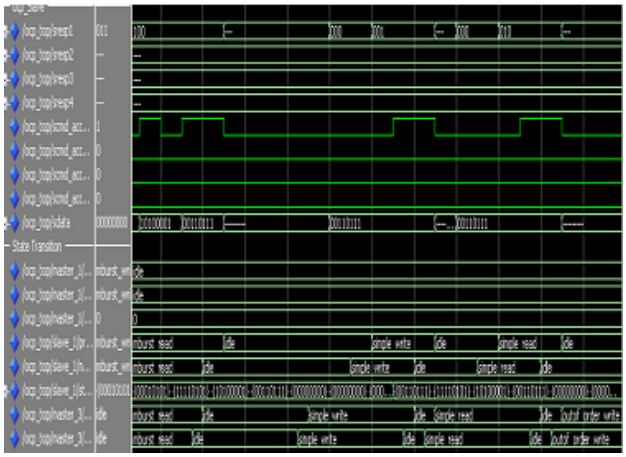

FIGURE 5-SIMPLE TRANSACTIONS

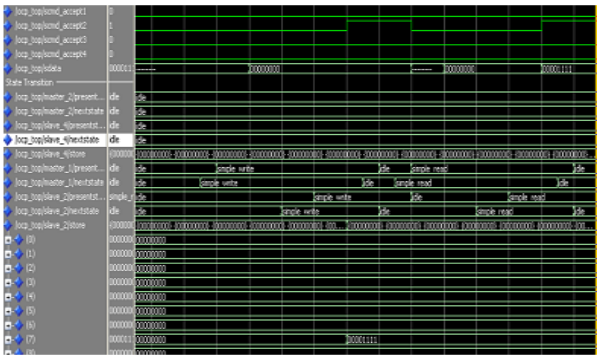

FIGURE 6 BURST TRANSACTIONS

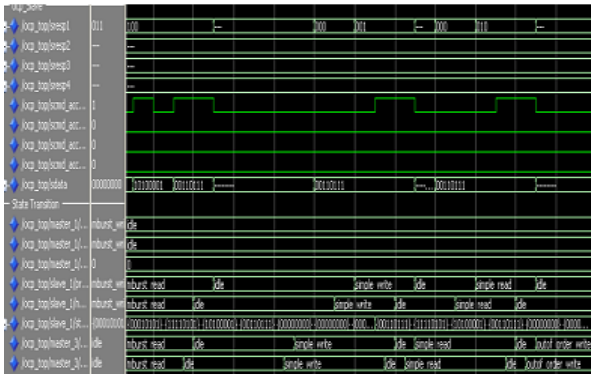

FIGURE 7-PIPELINED TRANSACTIONS

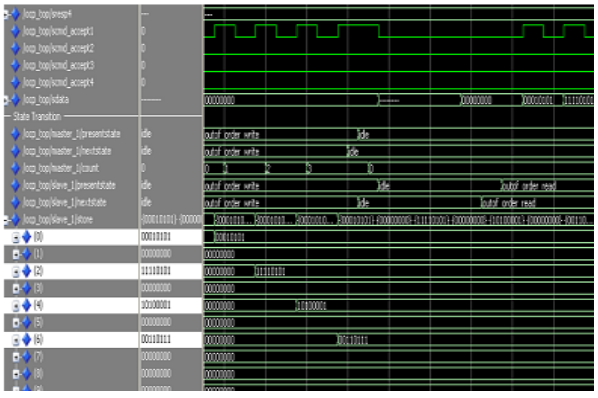

FIGURE 8 OUT OF ORDER TRANSACTIONS

Figure7 shows pipelined transactions by which we can issue control signals for different transactions without waiting for return of data requested.Without pipelined transactions period of latency will occur.Figure8-shows out-of-order transactions in which the control input for out-of-order write is 101 and for out-of-order read 110.This is a non-sequential transaction which allows the order of responses to be different from the order of their requests.we have signal inputs as address,data and masterenableand size which gives number of non sequential locations in which data can be written.after operation slave will send acknowledgement to master.similarly for out-of-order read the data can be read from non cequential locations by the selected slave and given as data to the master Timing Summary: Minimum period: 25.498ns (Maximum Frequency: 39.219MHz)

Minimum input arrival time before clock: 8.158ns Maximum output required time after clock: 4.063ns Maximum combinational path delay: No path found

REFERENCES

1. Open Core Protocol (OCP) Specification, http://www.ocpip.org/home.
2. Advanced Microcontroller Bus Architecture (AMBA) Specification Rev 2.0 & 3.0, http://www.arm.com.
3. N.Y.-C. Chang, Y.-Z. Liao and T.-S. Chang, “Analysis of Shared-link AXI,” IET Computers & Digital Techniques, Volume 3, Issue 4, pages 373-383, 2009.
4. Y.-T. Kim, T. Kim, Y. Kim, C. Shin, E.-Y. Chung, K.-M. Choi, J.-T. Kong, S.-K. Eo, “Fast and Accurate Transaction Level Modeling of an Extended AMBA2.0 Bus Architecture,” Design, Automation, and Test in Europe, pages 138-139, 2005.
5. G. Schirner and R. Domer, “Quantitative Analysis of Transaction Level Models for the AMBA Bus,” Design, Automation, and Test in Europe, 6 pages, 2006.
6. CoWarewebsite,http://www.coware.com

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)