International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue-01
January 2018

Design andStudy of On-chip Bus with Open Core Protocol Interface

VenkannaPolagoni, ApuriManasa, MukkeraGouthami
[1,2,3] Assistant professor.Dept.of ECE, Ashoka Institute of Engineering and Technology, Hyderabad,Telanagana, India

Abstract:*As increasingly more IP cores are integrated intoa SOC layout, the conversation flow among IP coreshas expanded considerably and the performance of the on-chipthe bus has become a dominant aspect of the performance ofa system. The on-chip bus design may be divided into parts, particularly the interface and the internal architecture ofthe bus.In this work, the well-described interface well known Open Core Protocol (OCP) is adopted, and the internal bus architecture is designed. The Open Core Protocol(OCP) is a middle-centric protocol which defines an excessive-performance, the bus-unbiased interface between IP cores that reduces layouttime, design threat, and production expenses for SOC designs. The essential belongings of OCP are that it may be configured with appreciating thesoftware required. The OCP is chosen because of its advanced supporting features which include configurable sideband control signalingand check harness alerts whereas as compared to other middle protocols. The OCP defines a point-to-point interface between communicating entities including IP cores and bus interface modules. One entity acts as the master of the OCP instance, and the alternative asthe slave. Only the master can present instructions and is the controlling entity. The slave responds to commands provided to it, either by means ofaccepting data from the master or importing data to the master.*

Index Terms:OPC, SoC, AXI, Xilinx-ISE

I. INTRODUCTION

The On-Chip bus plays a key function inside the system-on-a-chip(SoC) layout by enabling the green integration ofheterogeneous system components which include CPUs, DSPs,software- precise cores, recollections, and custom good judgment.Recently, as the extent of layout complexity has come to bebetter, SoC designs require a device bus with excessivebandwidth to carry out more than one operations in parallel. Toremedy the bandwidth issues, An efficient OCP protocolhas been evolved.A SOC chip typically consists of a lot of IP coresthat speak with every other via on-chip buses.As the VLSI method era continuously advances, thefrequency and the quantity of the data communicationamong IP cores growth notably. As a result, thecapability of on-chip buses to deal with the massive amount of datatraffic turns into a dominant factor for the generaloverall performance. The layout of on-chip buses may be dividedinto two elements: bus interface and bus architecture. The businterface entails a set of interface indicators and theircorresponding timing courting, even as the busstructure refers to the inner components of buses andthe interconnections the various IP cores. The extensivelycustomary on-chip bus, AMBA AHB, defines a fixed of businterface to facilitate simple (single) and burst examine/writetransactions. AHB also defines the inner bus architecture,that's specifically a shared bus composed of multiplexers.The multiplexer-based complete bus architecture works agreeably for alayout with a small range of IP cores. When the rangeof included IP cores will increase, the communicationamong IP cores also boom and it becomes prettyfrequent that two or more master IPs could request datafrom one of a different slaves at the same time.

Each channel involves a fixed of indicators. AXI does no longerrestriction the inner bus architecture and leaves it todesigners. Thus designers are allowed to integrate two IPcores with AXI by means of both connecting the wires without delay orinvoking an in-house bus between them. The different businterface protocol is proposed via a non-worthwhilethe , the Open Core Protocol – InternationalPartnership (OCP-IP). OCP is an interface (or socket)aiming to standardize and hence simplify the deviceintegration issues. It facilitates machine integration viadefining a set of the concrete interface (I/O indicators and thehandshaking protocol) which is unbiased of the busarchitecture.Based on this interface IP center designers can listenon designing the inner functionality of IP cores, busdesigners can emphasize the inner bus structure,and device integrators can attention on the device troubles suchas the requirement of the bandwidth and the whole devicestructure. In this way, device integration turns into muchextra green.

Most of the bus functionalities defined inAXI and OCP are quite similar. The most conspicuousthe difference among them is that AXI divides the deal withchannel into unbiased write cope with channel and studydeal with channel such that examine and write transactions can beprocessed simultaneously. However, the additional location ofthe separated address channels are the penalty.

Some previous work has investigated on-chip buses fromvarious aspects. The work presented in [3] and [4] developshigh-level AMBA bus models with fast simulation speed andhigh timing accuracy. The authors in [5] propose an automaticapproach to generate high-level bus models from a formalchannel model of OCP. In both of the above work, the authorsconcentrate on fast and accurate simulation models at highlevel but did not provide real hardware implementation details.

In [6], the authors implement the AXI interface on a sharedbus architecture. Even though it costs less in area, the benefitof AXI in the communication efficiency may be limited by theshared-bus architecture.In this paper we propose a high-performance on-chip busdesign with OCP as the bus interface. We choose OCPbecause it is open to the public and OCP-IP has providedsome free tools to verify this protocol. Nevertheless, most busdesign techniques developed in this paper can also be appliedto the AXI bus. Our proposed bus architecture featurescrossbar/partial-crossbar based interconnect and realizes mosttransactions defined in OCP, including 1) single transactions,2) burst transactions, 3) lock transactions, 4) pipelinedtransactions, and 5) out-of-order transactions. In addition, theproposed bus is flexible such that one can adjust the busarchitecture according to the system requirement.

One key issue of advanced buses is how to manipulate theorder of transactions such that requests from masters andresponses from slaves can be carried out in best efficiencywithout violating any ordering constraint. In this work wehave developed a key bus component called the scheduler tohandle the ordering issues of out-of-order transactions. Wewill show that the proposed crossbar/partial-crossbar busarchitecture together with the scheduler can significantlyenhance the communication efficiency of a complex SOC.

Another notable feature of this work is that we employboth transaction level modeling (TLM) and register transferlevel (RTL) modeling to design the bus. We start from theTLM for the consideration of design flexibility and fastsimulation speed. We then refine the TLM design intosynthesizable and cycle-accurate RTL codes which can besynthesized into gate level hardware to facilitate accuratetiming and functional simulation. The proposed bus has beenemployed in a multimedia SOC design and the results showthat not only our TLM model has better simulation efficiencycomparing to a bus obtained through a commercial ESL tool,but also our RTL on-chip bus design performs much moreefficient than the multiplexer-based buses or those withoutout-of-order feature in real SOC design.

II. OCP SYSTEM

In an OCP system, communicating components (e.g.,processors, memory modules, and I/O devices) need awrapper which implements the Open Core Protocolinterface.Network-on-chip is an efficient communication medium compared to bus because of its advantages like the following:

1. Efficiency improvement in speed, bandwidth, area , and power consumption.
2. Supports concurrency – effective spatial reuse of resources.
3. Low latency.
4. Scalable bandwidth
5. Modularity

We first describe the various bus functionalities including1) burst, 2) lock, 3) pipelined, and 4) out-of-order transactions.

Burst transactions: The burst transactions allow the grouping of multipletransactions that have a certain address relationship, and canbe classified into multi-request burst and single-request burstaccording to how many times the addresses are issued.

Lock transactions: Lock is a protection mechanism for masters that have lowbus priorities. Without this mechanism the read/writetransactions of masters with lower priority would beinterrupted whenever a higher-priority master issues a request.

Pipelined transactions: Pipelined transactions (outstanding transactions)show the difference between nonpipelined and pipelined (also called outstanding in AXI) readtransactions.

Out-of-order transactions: The out-of-order transactions allow the return order ofresponses to be different from the order of their requests.These transactions can significantly improve

thecommunication efficiency of an SOC system containing IPcores with various access latencies.

III. PROPOSED FRAMEWORK

The block diagram which explains the basic operation andcharacteristics of OCP is shown in Figure 2.The OCP definesa point-to-point interface between two communicatingentities such as IP cores and bus interface modules. Oneentity acts as the master of the OCP instance, and the other asthe slave. Only the master can present commands and is thecontrolling entity. The slave responds to commands presentedto it, either by accepting data from the master, or presentingdata to the master. For two entities to communicate thereneed to be two instances of the OCP connecting them such asone where the first entity is a master, and one where the firstentity is a slave.

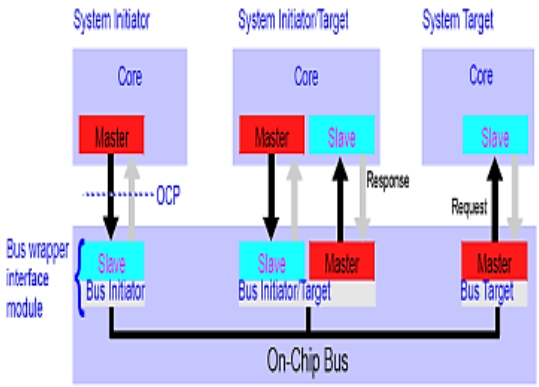

Fig.1 Basic block diagram of OCP instance

Figure.1 shows a simple system containing a wrapped busand three IP core entities such as one that is a system target,one that is a system initiator, and an entity that is both. Thecharacteristics of the IP core determine whether the coreneeds master, slave, or both sides of the OCP [5] and thewrapper interface modules must act as the complementaryside of the OCP for each connected entity. A transfer acrossthis system occurs as follows.

A system initiator (as the OCP master) presents command,control, and possibly data to its connected slave (a buswrapper interface module). The interface module plays therequest across the on-chip bus system. The OCP does notspecify the embedded bus functionality. Instead, the interfacedesigner converts the OCP request into an embedded bustransfer. The receiving bus wrapper interface module (as theOCP master) converts the embedded bus operation into alegal OCP command. The system target (OCP slave) receivesthe command and takes the requested action.

A crossbar architecture is employed such that more than onemaster can communicate with more than one slavesimultaneously. If not all masters require the accessing pathsto all slaves, partial crossbar architecture is also allowed.

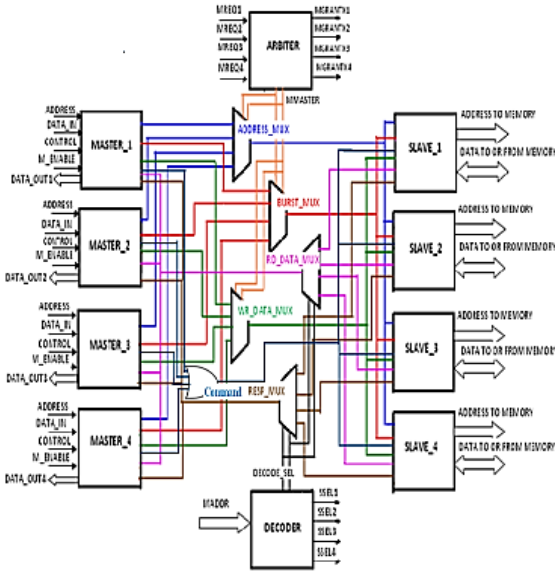

Fig. 2 Architecture of Proposed OCP

Arbiter: In traditional shared bus architecture, resource contentionhappens whenever more than one master requests the bus atthe same time. For a crossbar or partial crossbar architecture,resource contention occurs when more than one master is toaccess the same slave simultaneously. In the proposed designeach slave IP is associated with an arbiter that determineswhich master can access the slave.

Decoder: Since more than one slave exists in the system, the decoderdecodes the address and decides which slave return responseto the target master. In addition, the proposed decoder alsochecks whether the transaction address is illegal ornonexistent and responses with an error message if necessary.

FSM-M & FSM-S: Depending on whether a transaction is a read or a writeoperation, the request and response processes are different.For a write transaction, the data to be written is sent outtogether with the address of the target slave, and thetransaction is complete when the target slave accepts the dataand acknowledges the reception of the data. For a readoperation, the address of the target slave is first sent out andthe target slave will

issue an accept signal when it receivesthe message. The slave then generates the required data andsends it to the bus where the data will be properly directed tothe master requesting the data. The read transaction finallycompletes when the master accepts the response and issues anacknowledge signal. In the proposed bus architecture, weemploy two types of finite state machines, namely FSM-Mand FSM-S to control the flow of each transaction. FSM-Macts as a master and generates the OCP signals of a master,while FSM-S acts as a slave and generates those of a slave.

These finite state machines are designed in a way that burst,pipelined, and out-or-order read/write transactions can all beproperly controlled.

IV. RESULTS AND DISCUSSION

The proposed design is coded in VHDL language andsimulated using Xilinx ISE tool. The simulated waveformsfor simple transactions, burst transactions, pipelinedtransactions and out of order transactions are shown infollowing figures.

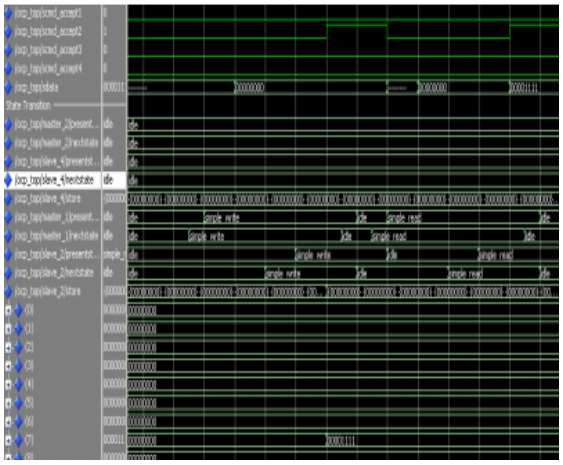

Fig.3 Simple transaction

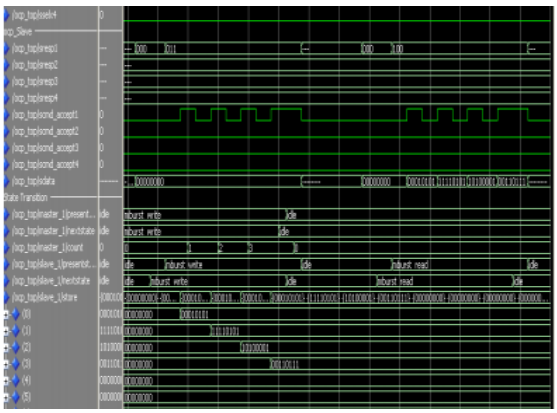

Fig.4 Burst transaction

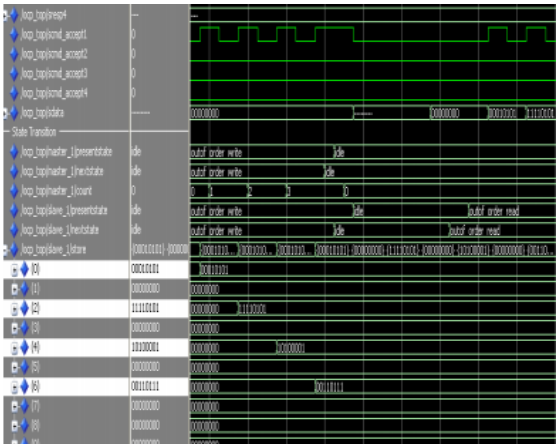

Fig.5 Out of order Transaction

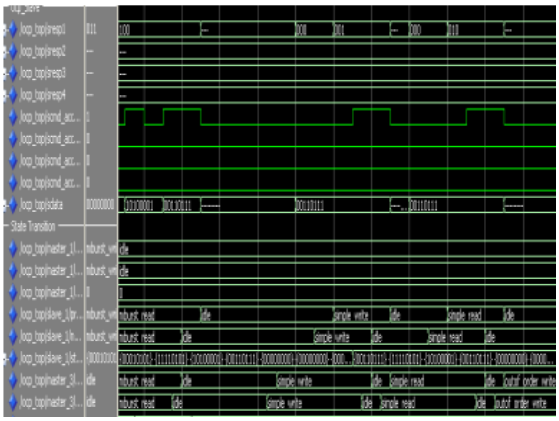

Fig.6 pipelined Transaction

V. CONCLUSION

This work provides the OCP (Open Core Protocol)the design which acts as an interface between two exceptional IPcores. In this paintings, first of all, the investigation on the OCP iscarried out and the basic commands and its running areidentified based on which the signal flow diagram and thespecifications are developed for designing the OCP the usage ofVHDL. This permits the designer of the cores and the device to work in parallel and shortens layout times. In addition, no longer having device good judgment inside the cores permits the cores to be reused and not using an additional time for the center to be re-created. Depending upon the real-time application those intellectual properties may be used.

REFERENCES

[1] Advanced Microcontroller Bus Architecture (AMBA)Specification Rev 2.0 & 3.0, http://www.arm.com.

[2] Open Core Protocol (OCP) Specification,http://www.ocpip.org/home.

[3] Y.-T. Kim, T. Kim, Y. Kim, C. Shin, E.-Y.Chung, K.-M.Choi, J.-T.Kong, S.-K.Eo, "Fast and Accurate TransactionLevel Modeling of an Extended AMBA2.0 Bus Architecture,"Design, Automation, and Test in Europe, pages 138-139, 2005.

[4] G. Schirner and R. Domer, "Quantitative Analysis ofTransaction Level Models for the AMBA Bus," Design,Automation, and Test in Europe, 6 pages, 2006.

[5] C.-K. Lo and R.-S. Tsay, "Automatic Generation of CycleAccurate and Cycle Count Accurate Transaction Level BusModels from a Formal Model," Asia and South Pacific DesignAutomation Conference, pages 558-563, 2009.

[6] Automation, and Test in Europe, 6 pages, 2006.

[7] C.-K. Lo and R.-S. Tsay, "Automatic Generation ofCycle Accurate and Cycle Count Accurate TransactionLevel Bus

[8] Models from a Formal Model," Asia and South PacificDesign Automation Conference, pages 558-563, 2009.

[9] N.Y.-C. Chang, Y.-Z.Liao and T.-S. Chang, "Analysisof Shared-link AXI," IET Computers & DigitalTechniques, Volume 3, Issue 4, pages 373-383, 2009.

[10] IBM Corporation, "Prioritization of Out-of-Order DataTransfers on Shared Data Bus," US Patent No.7,392,353, 2008.

BIO DATA

Author 1

VenkannaPolagonipresently working as Assistant Professor in.Dept. of ECE, Ashoka Institute of Engineering and Technology, Hyderabad, Telanagana, India.

Co-Author-1

ApuriManasapresently working as Assistant Professor in.Dept. of ECE, Ashoka Institute of Engineering and Technology, Hyderabad, Telanagana, India.

Co-Author-2

MukkeraGouthamipresently working as Assistant Professor in.Dept. of ECE, Ashoka Institute of Engineering and Technology, Hyderabad, Telanagana, India.