International Journal of Engineering Research and Science & Technology

ISSN : 2319-5991
Vol. 4, No. 2
May 2015

www.ijerst.com

Email: editorijerst@gmail.com or editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 4, No. 2, May 2015

Research Paper

AN SOC IMPLEMENTATION OF HIGH SPEED OCP SWITCH ARCHITECTURE

Yalavarthi Ramakrishna Paramahamsa[1]*

*Corresponding Author: Yalavarthi Ramakrishna Paramahamsa ✉ ylrkph@gmail.com

OCP presents an efficient solution to address SoC contemporary design issues and shorter time-to-market requirements using an industry standard socket. The on chip permutation follows the industrial OCP protocol. The design which complies with the bus interface protocol to carry out the various advanced bus functionality consequently dominates the communication efficiency of an SOC system. OCP is flexible and configurable to support the communication needs of a wide range of Intellectual Property cores, and is now in widespread use. Various bus transactions defined in AXI and OCP to reduce the communication latency and increase the bus throughput are supported by the proposed bus architecture. The Functional-simulation has been successfully carried out with the results matching with expected ones which shows the effective solution for the communication field. The design functional verification and Synthesis is done by using Xilinx-ISE.

Keywords: System-On-chip, OCP, Xilinx-ISE, Verilog

INTRODUCTION

Current technology trends, scaling, and with end users showing a marked preference for the smaller geometries of deep submicron processes forces a design style where multiple independent circuit implementations are integrated together into single System-on-Chip (SoC). However, contemporary SoC designs have their own share of issues and challenges. The major challenges faced by a design engineer include the ever increasing complexity in modern SoC designs, reusability, time to-market, communication between Intellectual Property (IP) cores, integration of different clocked domain IP cores, and global clock distributions on a chip.. As an organization, OCP-IP guides the development of the industry standard by providing the specification, as well as EDA tools and models, to further ease the task of SoC development. OCP research is mainly targeted to provide an efficient solution to address SoC design challenges by building standard NoC interfaces

[1] Professor, Brindavan Institute of Technology and Science (BITS), Kurnool 518218, AP, India.

using an industry standard socket interface, Open Core Protocol (OCP).

OCP SYSTEM

In an OCP system, communicating components (e.g., processors, memory modules, and I/O devices) need a wrapper which implements the Open Core Protocol interface. This interface enforces a point-to-point unidirectional communication when two components are communicating, the one that utilizes a "master" OCP port sends to the other that utilizes a "slave" OCP port. Communication protocols deal with different types of resource management algorithms used for determining access right to shared communication channels. From this point of view, in the rest of this section, we will give a brief comment related to the main feature of the existing communication protocols. OCP is mainly targeted to provide an efficient solution to address SoC design challenges by building standard NoC interfaces using an industry standard socket interface, Open Core Protocol (OCP). Also there are few motivating factors from an industrial perspective internalization and implementation of this project. Levels of device integration leads to SoC design style SoC provides the platform for integration of different architectural cores such as microprocessor chips, Application Specific Integrated Circuit (ASIC) chips, Random Access Memory (RAM) chips, and peripherals on a single die.

The major advantage of SoC design over custom design is its shorter time-to-market, but at the expense of performance and area. SoC designs help enable IP reusability when they utilize a standard communication interface. Employing a standard socket interface on a SoC enables reuse of the good design swith minimal modification to IPcores. This project targets the shorter time-to-market feature of SoCs to build standard interfaces between IPcores at the expense of power, performance, and area. SoC design Scan Sea NoCoron-chip bus as the on-chip communication medium between IP cores. 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.

The Open Core Protocol (OCP) is a core centric protocol which defines a high-performance, bus-freelance interface between IP cores that reduces design time, style risk, and producing costs for SOC styles. Main property of OCP is that it can be configured with respect to the application needed.

Evaluation and Comparison with Other Buses

An SOC chip usually contains a large number of IP cores that communicate with each other through on-chip buses. As the VLSI process technology continuously advances, the frequency and the amount of the data communication between IP cores increase substantially. As a result, the ability of on chip buses to deal with the large amount of data traffic becomes a dominant factor for the overall performance. 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 inter-connections among the IP cores. The Open Core Protocol (OCP) is an open standard, bus-independent protocol provided by Open Core Protocol-International Partnership (OCP-IP). It meets all the core centric requirements and is one of the protocols which unifies all the inter core communications including sideband control and test signals. OCP defines a high performance, complete interface socket between IP cores facilitating design reuse, and also reduces design time, design risk, and manufacturing costs for SoC designs. OCP supports very high-performance data transfer models ranging from simple request-grants through burst pipelined and tagging objects. OCP protocol compliance verification is one of the distinguishing features from other protocols. OCP-IP not only provides the specification and its member-driven evolution, but also industrial grade tools and services that ensure its members can rapidly confirm compliance and maximize their productivity.B. Overview of the AMBA

The AMBA AHB 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) proposed by the ARM company. The Advanced Microcontroller Bus Architecture (AMBA) specification defines an on-chip communications standard for designing high-performance embedded microcontrollers. Three distinct buses are defined within the AMBA specification:

1. The Advanced High-performance Bus (AHB)
2. The Advanced System Bus (ASB).
3. The Advanced Peripheral Bus (APB).

A Test methodology is included with the AMBA specification which provides an infrastructure for modular macro cell test and diagnostic access.

OCP Description

Commonly an OCP transfer is made of two separate and temporally decoupled phases, one for the request, and one for the response. Progression within a phase is controlled by a two-way handshake between the communicating entities. A phase begins with one side asserting the signals associated with that phase. The OCP commands are generally accompanied by an address field. In a standard system, it is the responsibility of the chip interconnection network to route the request to the appropriate target, based on this address. As an illustration of the OCP flexibility, it is interesting to point out here that it is perfectly legal for a local OCP interface not to include any address information. More generally, an OCP interface is defined with a very comprehensive set of parameters, which allows enabling/ disabling most of the signals individually. As a consequence, commonly-used interface models such as read-only, write-only, synchronization-only or FIFO interfaces can be easily described. The OCP interface parameters are defined in a set of configuration files, shared by the different tools in the development environment as a fully executable specification.

The Open Core Protocol is an openly licensed, core-centric protocol standard, which

defines a high performance, synchronous, bus independent configurable interface for communication between IP cores and NoC. It is an efficient point-to-point connection standard and because of its configurability, scalability, and generality, it has been widely accepted from low-power to high-performance applications. It can be optimized to use only the necessary features required for communicating between any two components, which saves chip area. It dramatically improves reusability of IP cores independent of the architecture and design of the systems, which leads directly to a more predictable, productive SoC designs and also simplifies the system verification and testing. OCP consists of an aggregation of signals that aims to unify the communication among IP blocks, reducing the system design time significantly. It is comprised of a continuum of communication protocols that share common definition for the whole system where it ensures dramatic time reduction-of functional verification for any future releases of the system.

DESIGN DESCRIPTION

OCP is one of the viable core-centric solutions to address contemporary SoC design implementation requirements. Commercially existing native OCP interfaces (integrated into IP Cores) are fully synchronous in nature which limits an IP core's capabilities to interface with other clocking control methodologies. In order to explore other clocking control methods, this research study designs and builds modular clocked wrapper OCP interfaces (OCP located outside the IP core) for existing IP cores. The Open Core Protocol is an openly licensed, core-centric protocol standard, which defines a high performance, synchronous, bus independent configurable interface for communication between IP cores and NoC. It is an efficient point-to-point connection standard and because of its configurability, scalability, and generality, it has been widely accepted from low-power to high-performance applications. It can be optimized to use only the necessary features required for communicating between any two components, which saves chip area. It dramatically improves reusability of IP cores independent of the architecture and design of the systems, which leads directly to a more predictable, productive SoC designs and also simplifies the system verification and testing. OCP consists of an aggregation of signals that aims to unify the communication among IP blocks, reducing the system design time significantly. It is comprised of a continuum of communication protocols that share common definition for the whole system where it ensures dramatic time reduction -of functional verification for any future releases of the system.

This research develops a new approach to increase modularity, improve reliability, and reduced sign time to interface different IPs to the OCP socket. This consists of splitting the design into common shared components and custom back-ends that are specific to the IP core. The common components consist of OCP master and slave components and a Domain Interface (DI) module. The DI module is used to synchronize mutually asynchronous clocked domains and data flow control. These will be described in more detail in this chapter.

Design Specifications and Supporting Features

The OCP methodology presented enables

intellectual property designers to design core interfaces in standard ways. This facilitates reusing OCP-compliant cores across multiple SoC designs which, in turn, drastically reduces design times, support costs, and overall cost for electronics SoCs. Provides a comprehensive introduction to Open Core Protocol, which is more accessible than the full specification. This interface enforces a point-to-point unidirectional communication when two components are communicating, the one that utilizes a \master" OCP port sends to the other that utilizes a slave OCP port.

Burst Transactions

A burst transaction is a set of transfers that are linked together, which have a Pre-defined address relation and number of transfers. Compared to single data phase transactions, burst transactions improve data throughput since the address is transferred only during initial bus grant followed by chunks of data. Burst transactions are supported not only to increase the throughput but also to reduce latency and data activity factor across the network router links. Different burst implementations are possible depending on the burst length (size of the burst) and burst sequence (relation between the addresses). This project design supports a maximum burst size of 8 (8* word size) i.e., 256-bits of data. This project implementation supports three types of address sequences (user defined, incremental, and wrapped types), and data can be transferred during each clock cycle.

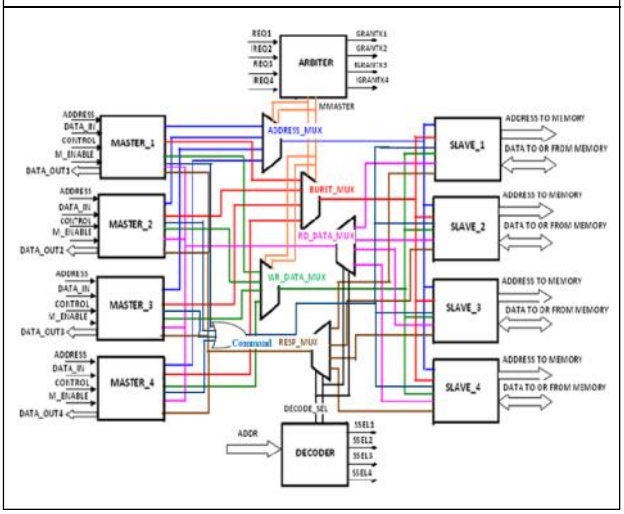
Tagging or Out-of-Order Response

Basically, tags are used to support out-of-order response by directly linking the Slave IP core response to the original request which triggered it. By supporting out-of-order responses, in most cases, the use of tags can improve overall system performance since responses are not halted due to dependencies on previous transactions. In this project implementation, a tag size of 8 is employed.In this work, the numerous OCP profiles will be designed using Verilog and the developed design will be used with respect to its appropriate application in the important time product. Basically the OCP unifies all inter-core communications. The OCP's synchronous unidirectional signaling produces simplified core implementation, integration and timing analysis. The OCP readily adapts to support new core capabilities while limiting check suite modifications for core upgrades.

OCP

Basically OCP has the address is of 8 bits, data is of 8 bits, control signal is of 3bits and burst is of integer type. The 4 k bit memory (28 = 256 bits = 4 k bits) is used in the slave side in order to verify the protocol functionality. The System will

Figure 1: Architecture of OCP Interface



give the inputs to OCP Master during Write operation and receive signals from OCP Slave during Read operation.

OCP Specification

The specifications for the Open Core Protocol are identified for both simple write and read operation supports the pipelining operation and burst operation. The identified specifications are represented in tabulation format. The Open Core Protocol interface addresses communications between the functional units (or IP cores) that comprise a system on a chip. The OCP provides independence from bus protocols without having to sacrifice high-performance access to on-chip interconnects. By designing to the interface boundary defined by the OCP, you can develop reusable IP cores without regard for the ultimate target system. To simplify timing analysis, physical design, and general comprehension, the OCP is composed of uni-directional signals driven with respect to, and sampled by the rising edge of the OCP clock. The OCP is fully synchronous and contains no multi-cycle timing paths. All signals other than the clock are strictly point-to-point.The OCP separates requests from responses. A slave can accept a command request from a master on one cycle and respond in a later cycle. The division of request from response permits pipelining. The OCP provides the option of having responses for Write commands, or completing them immediately without an explicit response.

Simple Write and Read

This simple write and read operation for which the basic and mandatory signals required signals are tabulated in Table 5.1. The Open Core Protocol (OCP) defines a high-performance, bus-independent interface between IP cores. OCP defines a point-to-point interface between two communicating entities. 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. The OCP defines complete standard from the basic data flow signals to the signals that are used for test purposes. Broadly, OCP signals can be divided into two main categories, the basic OCP signals and the optional OCP signals. The presence of basic OCP signals in any core is necessary if it is following the OCP interface. The optional OCP signals can be included according to the requirement. The OCP transforms IP cores making them independent of the architecture and design of the systems in which they are used. Optimizes die area by configuring into the OCP only those features needed by the communicating cores.An IP core can be a simple peripheral core, a high-performance microprocessor, or an on-chip communication subsystem such as a wrapped on-chip bus.

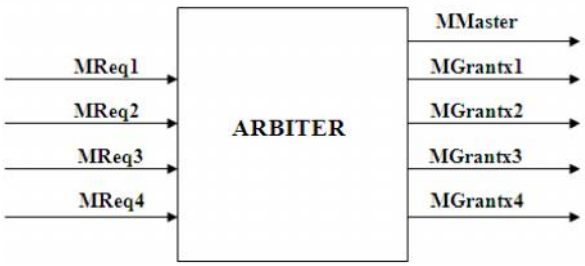
OCP Arbiter

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.

OCP Master

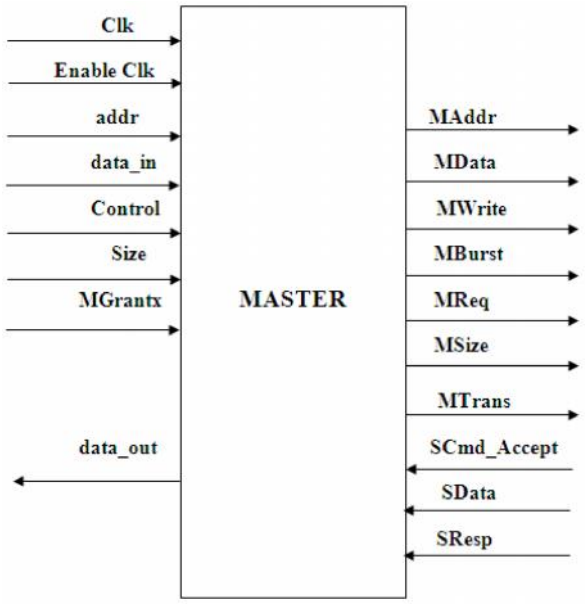
OCP master have Clk, Enable Clk as inputs. And also the master gets the addr, data_in,control, size signal as input and gives

Figure 2: OCP Arbiter



MAddr, Mdata, MWrite, and Mburst as outputs. MReq signal is output goes to the arbiter and gets MGrantx as input. Other inputs are SCmd_accept, SData, and SResp from the slave. When master is write operation data on the data_in signal port will transfer to the MData and when master is write operation data on the SData signal port will transfer to the data-out signal. And master also gives the MTrans signal which specifies data transfer in to or from the memory write or read in SEQ or NON_SEQ. MSize signal for no bits in each transfer here MSize width is 8-bit.

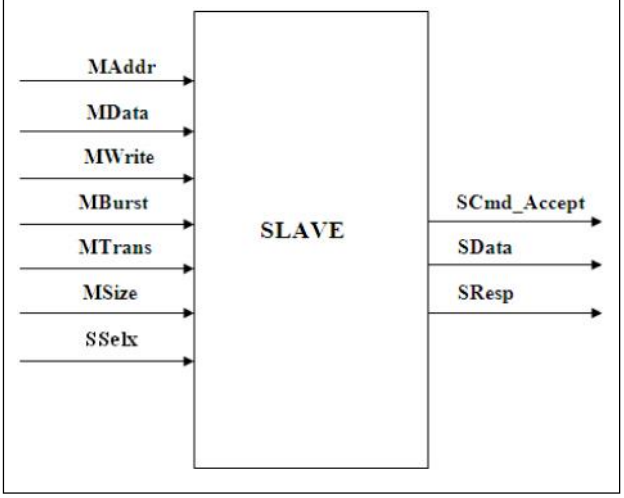
Figure 3: OCP Master



OCP Slave

The below figure shows the MAddr, Mdata, MWrite, Mburst. MTrans, MSize and SSelx signal as inputs and SCmd_accept, SData,SResp as outputs. Whenever slave makes write or read operation SCmd_accept signal will be high to Low and Low to High logic level. SResp, signal is acknowledgement signals to the master that slave has performed read or write operation.

Figure 4: OCP Slave



Write and Read Operation

The simple write and read operation in OCP has the mandatory signals whose specification is mentioned in the Table 1.

Table 1: OCP Operation

Control	Command
000	Idle
001	Simple Write
010	Simple Read

FSM for OCP Master

The Finite State Machine (FSM) is developed for the simple write and read operation of OCP Master. The simple write and read operation indicates that the control goes to IDLE state after every operation. The FSM for the OCP Master –

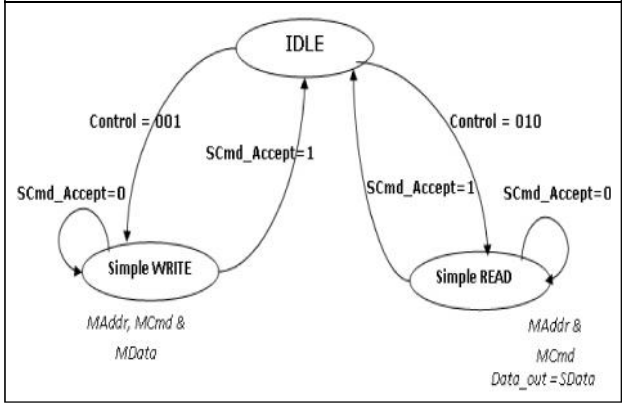
Simple Write and Read is developed and is shown in the Figure 6.1. Totally there are four states are available in this FSM such as IDLE, WRITE, READ and WAIT. Basically, the operation in the OCP will be held in two phases.

1. Request Phase
2. Response Phase

Initially the control will be in IDLE state (Control = "000") at which all the outputs such as MCmd, MAddr and MData are set to "don't care". The system will issue the request to the master such write request which leads to the WRITE state (Control = "001"). In this state, the address and the data will be given to the slave that is to be written and hence the process will get over only when the SCmd_Accept is asserted to high.

If SCmd_Accept is not set, this represents that the write operation still in process and the control will be in the WRITE state itself. Once the write operation is over the control will go to the IDLE state and then it will check for the next request.

Figure 5: FSM for OCP Master-Simple Write and Read

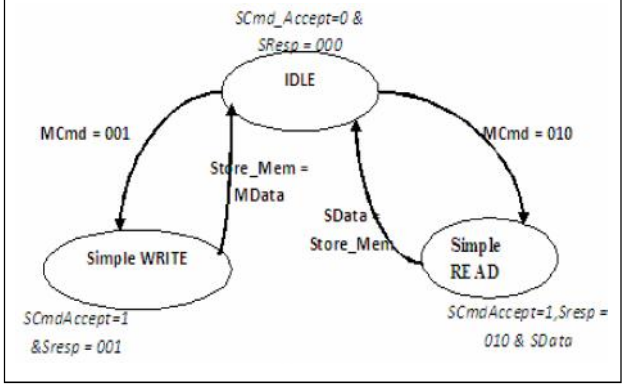


The Finite State Machine (FSM) for the OCP master simple write and read operation is developed and is shown in the Figure 5. OCP Master will be in IDLE State at the initial stage and will give the request signal to the arbiter which in turn provides the grant signal. Hence based on the control and the grant signal, OCP Master either goes to WRITE or READ state. If control from the system is a write request (Control = "001"), then the OCP Master go WRITE state and will issue the address (MADDR) and input data (MDATA) to the slave and also makes the MWRITE signal to high. Once these signals are issued, it will wait for the SCMD ACCEPT signal which will come from the slave after finishing the operation. Once the SCMD ACCEPT signal occurred, then the OCP Master will go to the IDLE state again. Similarly, when system gives read request (Control = "010"), master goes to the READ state and will give address (MADDR) and make MWRITE signal to Low. The data in the given address will be read out by the data out signal. Master go IDLE state when the SCMD ACCEPT signal is made high which represents the operation got over.

FSM for OCP Slave

The FSM for the OCP Slave which has the simple write and read operation is developed and is shown in the Figure 6.

Figure 6: FSM for OCP Slave-Simple Write and Read



The slave will be set to the respective state based on the MCmd issued by the master and the output of this slave is that the SCmd_Accept

and SResp. Initially control will be in the IDLE state and when the master issues the command as write request, and then the control will go the WRITE state in which the data will be written to the corresponding memory address location which is sent by the masters. Once the write operation is finished, the SCmd_Accept signal is set to high and is given to the master.When MCmd is given as read request, then the control will move to the READ state in which the data will read from the particular memory address location that is given by the master. Hence the SCmd_Accept is set to high and the SResp is set to the DVA which represents that the read operation over and control goes to the IDLE state.

RESULTS

Results for the OCP interface had taken for all modules individually and test cases are passed through testbench, written in Verilog 2001 and the results for module are described as follows:

Figure 7: Schematic View of the Top Module

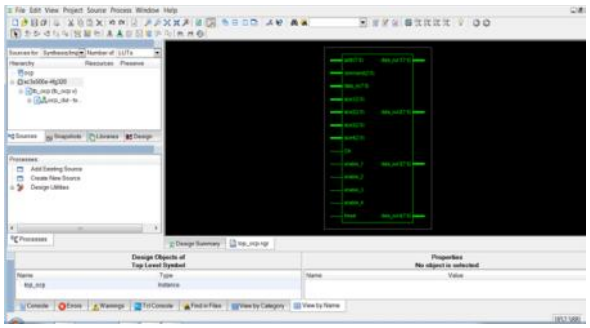


Figure 8: RTL Schematic View of the Top Module

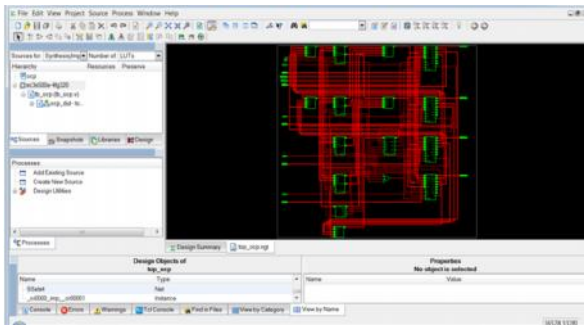


Figure 9: Wave Form of the Top Module

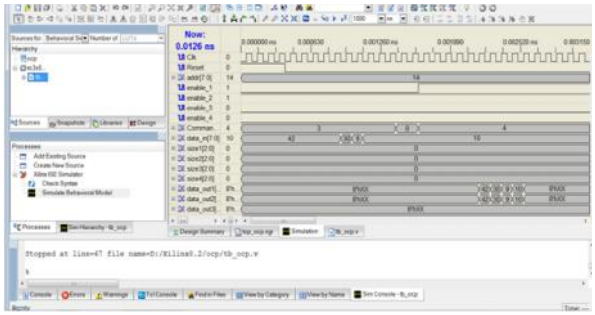


Table 2: Design Summary

Device Utilization Summary (Estimated Values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	81	4656	1%
Number of Slice Flip Flops	44	9312	0%
Number of 4 input LUTs	152	9312	1%
Number of bonded IOBs	44	232	18%
Number of GCLKs	1	24	4%

From the waveform it is clearly demonstrate that data in signal sends the user transmitted data to the corresponding multiprocessors according to the address being given through the enable signal. The RTL Schematic describes the user view of the system through which the flow of the signal will be easily understood. The design summary report describes the utilization factor among the available resources. Based on the number of 4 input LUTs the area utilization should be calculated.

CONCLUSION

This project can be extendable between more than two IP-cores interacting simultaneously using arbiter and Mutual exclusion elements. The current trend of the bus standard is to define an explicit bus interface and leave the internal bus

architecture to the bus interface and leave the internal bus architecture to the bus designer. The design which compiles with the bus interface protocol to carry out the various advanced bus functionality consequently dominates the communication efficiency of an SOC system. In this paper it is clearly shown the functionality of the On-Chip Permutation Network for Multiprocessor System-On-Chip which should be verified by XILINX ISE simulator.This project has been implemented by considering bursting data flow features of OCP. The other features of OCP, such as sideband and test signals, can be supported as extensions to this project. bursting sizes, as well as address and data widths, can also be increased to support higher data flow requirements.

REFERENCES

1. Advanced Microcontroller Bus Architecture (AMBA) Specification Rev 2.0 & 3.0, http://www.arm.com
2. AMBA AHB Specification
3. AMBA AXI Protocol Specification
4. ARM, AMBA Overview
5. Corp S (2002), "Wishbone System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores", http://www.silicore.net/pd les/wishbone.pdf
6. OCP3.0 Specification (www.ocpip.org).
7. OCP-IP, "Socket-Centric IP Core Interface Maximizes IP Applications", http://www.ocpip.org/white papers.php
8. OCP-IP, "The Importance of Sockets in SoC Design", http://www.ocpip.org/ white papers.php
9. Open Core Protocol (OCP) Specification, http://www.ocpip.org/home
10. Open Core Protocol Specification 2.2.1
11. Open Cores, "Open Cores", http://http://opencores.org/projects
12. Technical Information on Open Core Protocol, http://www.ocpip.org/, Open Core Protocol-International Partnership (OCP-IP).

International Journal of Engineering Research and Science & Technology
Hyderabad, INDIA. Ph: +91-09441351700, 09059645577
E-mail: editorijlerst@gmail.com or editor@ijerst.com
Website: www.ijerst.com