Journal of Advances in Science and Technology
Vol. 12, Issue No. 25, (Special Issue) December-2016, ISSN 2230-9659

High Performance and Scalable on-chip Bus with Thread Extension using Open Core Protocol

Arun Raj S. R.*

Department of Electronics & Communication Engineering, University B.D.T College of Engineering, Davangere-Karnataka

Abstract – In recent technology most of the core function need an on-chip bus having inter-operability interface irrespective of the core features. The issues that relate to SOC are communication between different core, integration of different clocked domain, increase in performance of system and requirement of a standard protocol interface. This paper presents the open core protocol (OCP) interface implementation with features simple basic OCP signals, burst OCP signals, tag OCP signals. The OCP interface is investigated with two different core systems. The implementation is designed with Xilinx ISE tool in verilog HDL language and verification of OCP interface is carried out.

Keywords— OCP, SOC, OCP interface.

----------♦----------

I. INTRODUCTION

The Integration of different core technology requires an efficient and robust on-chip bus interface. The different core in a system-on-chip (SOC) needed bus-independent design, so that core can operate without pertaining to on-chip bus functionality and communicate with any other core. An on-chip bus is implemented based on the OCP interface specification specified by the organization OCP-IP, OCP is a non-proprietary, openly licensed and a core-centric protocol which describes the system level integration requirements of IP-core. The AMBA specification is defined by the ARM Company which contains the bus architecture details to interconnect more than one master or slave [1]. An OCP specification contains signals and timing which is described by the members of OCP-IP [2]. An introduction to open core protocol is defined in the paper [3]. The OCP is interfaced with AHB bus which shows that OCP can work with different cores and with different bus architecture. OCP require a bridge interface, if two different bus architecture needs to be integrated for communication [4-6]. The on-chip bus using OCP interface is presented in the paper [7-10] the OCP interface is implemented with features simple transaction, burst transaction, tag transaction. The OCP is a core-centric, based on point-to-point interface, OCP provides synchronous interface while the cores connected to the OCP interface can be either synchronous or asynchronous.

II. OCP INTERFACE

An OCP interface follows three types of interconnect as shown in figure2.1. First is the peripheral interconnect which interconnect timer, I/O device, Interrupt controller these are simple basic signals, support single accesses and not necessary to use burst-related signals, a simple read and write communication with added handshake signals may be used. Second is the high speed interconnect these are required for subsystem components that require high throughput such as processor, co-processors, memory controllers and DMA engines. These interconnect require burst-related signals to improve throughput and flow control. These can also carry Tag transaction to implement ordered sequence and out-of-order sequence. Third is the bridging interconnect this is intended to bridge to other interface protocols. The bridge can have either an OCP master or slave port.

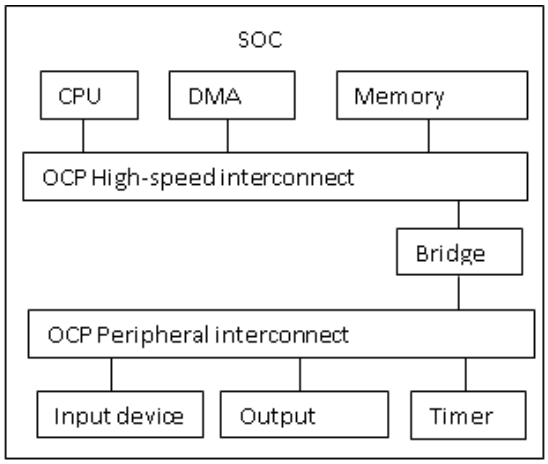

Figure 2.1 : An OCP interconnect for core

A. Simple basic OCP signals

Simple basic OCP signals carry out simple read and write operation this may include handshake signals. The figure2.2 shows three operations and the basic OCP signals. The command signal MCmd denotes the type of operation. WR in the MCmd denote the write operation specified by OCP master. MAddr and MData have the address and data for communication. When OCP slave gets ready to accept data from OCP master the SCmdAccept signal goes high and MData is received by the OCP slave. If OCP master signals MCmd as RD a read operation is performed. OCP master gives the address (MAddr) and when OCP slave is ready (SCmdAccept = 1) the data read is available at the SData and successful transaction is indicated by the Data valid/available signal (DVA). Once DVA is indicated OCP master read data from SData this process shows handshake signaling with OCP slave. If the OCP master signals MCmd as WRNP which represent a non-posted write operation is performed, this is similar to the write operation but a successful transaction is indicated by the DVA.

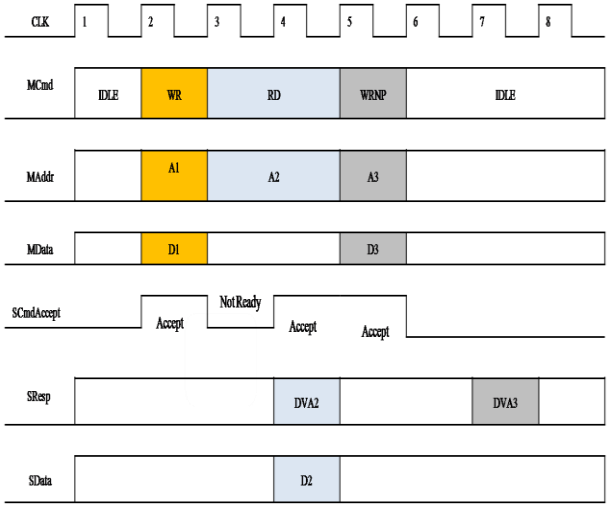

Figure 2.2: Simple read, write and non-post write transaction

A. Burst OCP signals

Burst OCP signals are required for high throughput of transaction. Burst transactions allow multiple transfers and pipeline the request without delay. The figure2.3 shows the Burst OCP signals with two write (WR) operation and two read (RD) operation. The first two write operation uses the MBurstLength to know length of the word to be written, MAddr hold the address, MBurstPrecise must be a constant and MBurstSeq specifies type of burst operation here the address is incremented by its OCP wordsize hence the command used is INCR. DVA1 and DVA2 indicate the write operation is successful. Next two read operation operate similar to the read operation, each read operation read BurstLength size of wordsize data from SData when SResp indicate a DVA. After completion of the request the MReqLast signal goes high to indicate burst request finished. SRespLast signal goes high after completion of the response to indicate burst response finished.

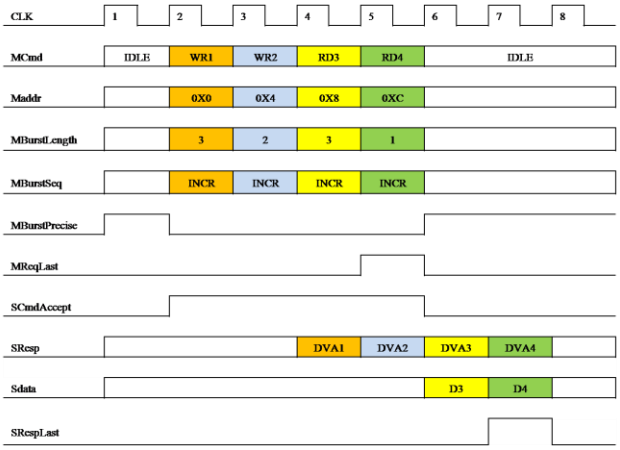

Figure 2.3: Burst OCP signals

B. Out-of-Order OCP signals

Out-of-Order or Tag transactions provide two types of transaction first is the out-of-order transaction which uses the tag ID to identify the transaction, these transfer may be delayed since performance fall is acceptable to these transfers. Where as another type is in-order transaction where the sequence of the request is processed sequentially. Tag transfer allows flow control and priority based transfer. The figure2.4 shows the out-of-order OCP signals have two in-order request and two out-of-order request. MCmd specifies the operation to be carried out, MAddr hold the address, MData and SData hold the write data and read data respectively. The operation is similar to read and write operation of basic signals. The signal MTagID generated by OCP master for a request to be carried out and the same tag ID is generated by signal STagID by OCP slave when response to the request is processed, this is necessary to identify the transaction. An in-order transaction the response to the request is not disordered. MTagInOrder go high when a request phase should be an ordered

sequence and STagInOrder go high to indicate the ordered sequence.

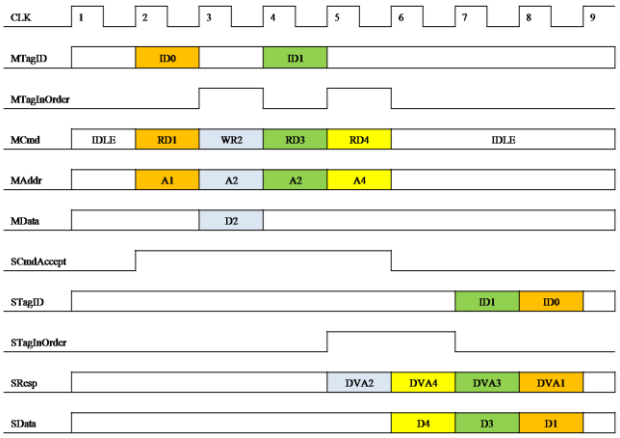

Figure 2.4: Out-of-order OCP signals

III. IMPLEMENTATION OF AN OCP INTERFACE

A core is a logical block that must integrate the system master or system slave in order to obtain a bus independence interface, this establish a unique interface between any core block. A system master initiates the request and controls the operation, while system slave process the request and respond to the system master. The OCP master interconnects to the system slave and OCP slave interconnects to the system master this establishes an OCP interface. The peripheral interconnect or high-throughput interconnect are interfaced to the core as shown in the figure3.1 with a single path interconnect from one core to another core. The OCP Bridge is either an OCP master or an OCP slave.

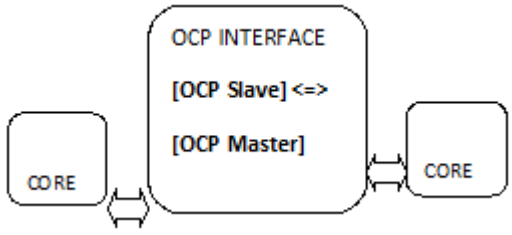

Figure 3.1: Block diagram of OCP interface

A. Synchronous and Asynchronous transfer

OCP interface operate in synchronous mode, this enables buffer usage. Initially the configuration is established with certain frequency to communicate with any core, and then the required operation is placed. This synchronous mode requires that both the core should support the same frequency. In Asynchronous mode the core will have different working frequency therefore a handshake signals are utilized for proper transfer of data and FIFO buffer used for flow control.

The system master or system slave may operate on two clock frequency. One is the system clock frequency which is obtained from the system core and another OCP clock having common clock frequency to work with OCP Interface this enables OCP interface to be a synchronous communication. The figure 3.2 illustrates the clock and signal interface. The cross link in the system master or slave block shows the changes in clock and signals this configuration enables the OCP interface to work with any two different core systems, hence Inter-operability is established since all cores communicate using OCP signals. If system master or slave does not have system clock the OCP clock will drive the system master or slave block.

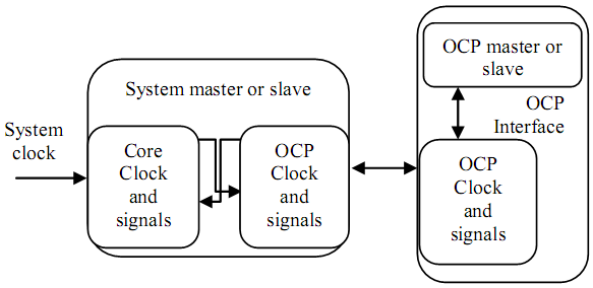

Figure 3.2: Illustration of clock and signals interface

IV. RESULT

The work is coded in verilog HDL language and simulated using Xilinx ISE tool. Verified timing waveforms using the Modelsim, Simple OCP Signals, Burst OCP signals and pipelined OCP signals are shown in the following figures.

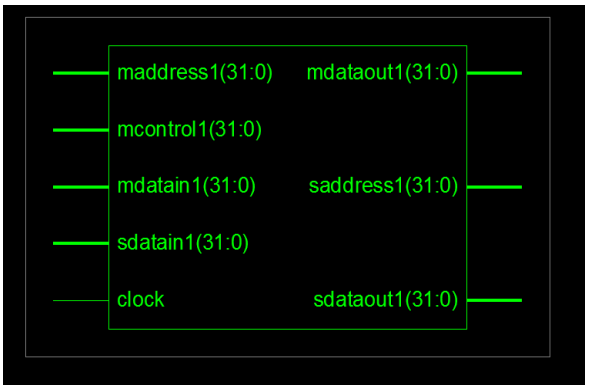

Figure 4.1: OCP Interface module

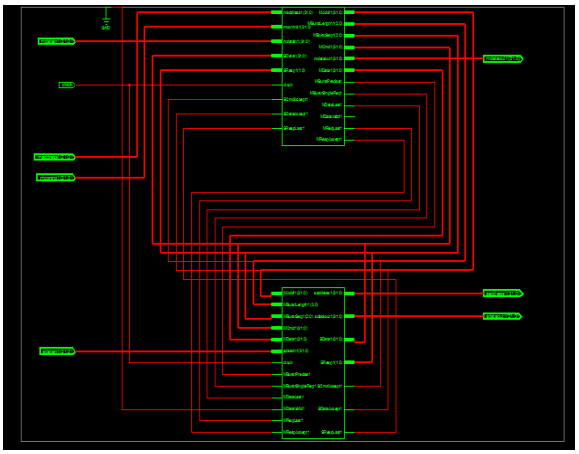

Figure 4.2: master-slave interconnect

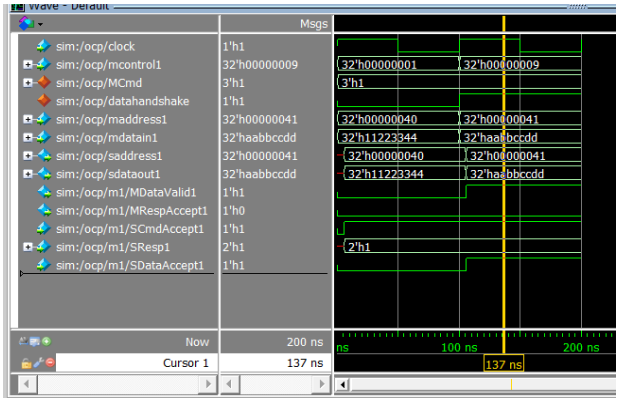

Figure 4.3: Basic Write with datahandshake OCP signals

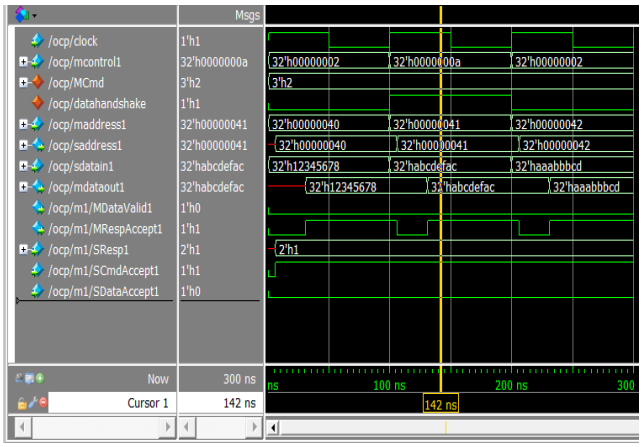

Figure 4.4: Basic Read with datahandshake OCP signals

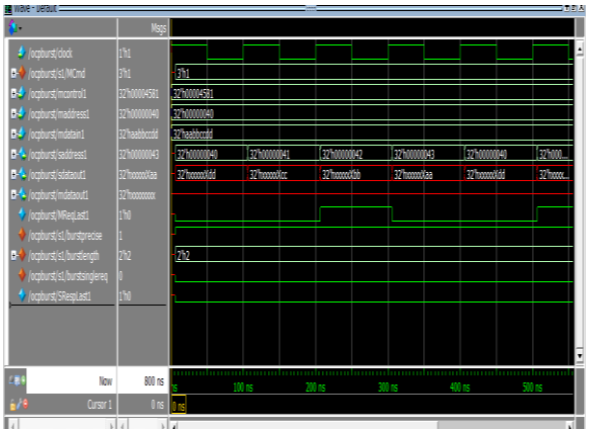

Figure 4.4: Burst request with pipelined transaction

V. CONCLUSION

The OCP interface is successfully implemented with two core interfaces and signal flow diagram for simple read and writes, burst read and write, out-of-order read and write is verified. Most of the required cores can be interfaced with the functionality of OCP interface to communicate with peripheral devices or high performance required cores. The OCP interface allows any two cores with system master or slave integrated could communicate without worrying of on-chip bus interface design and its operation, this also enable IP core reusability.

REFERENCES

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

Open Core Protocol (OCP) Specification 3.0, http://www.ocpip.org/home.

Wolf-Dietrich Weber “Enabling Reuse via an IP Core-centric Communications Protocol: Open Core Protocol” © 2000 Sonics, Inc.

Shen-Fu Hsiao, Chi-Guang Lin, Po-Han Wu, and Chia-Sheng Wen “Asynchronous AHB Bus Interface Designs in a Multiple-clock-Domain Graphics System” 2012 IEEE

Cheng-Ta Wu, Feng-Xiang Huang, Kuan-Fu Kuo, Ing-JerHuang “An OCP-AHB Bus Wrapper with Built-in ICE Support for SOC Integration” 2012 IEEE

Ramesh Bhakthavatchalu, Deepthy G R, Vidhya S, Nisha V “Analysis of Low Power Open Core Protocol Bridge Interface Using VHDL” 2011 IEEE

Chin-yaochang, yi-jiunchang, kuen-jonglee, jen-chiehyeh,shih-yin linandjui-liangma, “Design

of On-Chip Bus with OCP Interface" 2010 IEEE

ElinaRajanVarughese, Rony Antony P "IMPLEMENTATION OF EXTENDED OPEN CORE PROTOCOL INTERFACE MEMORY SYSTEM USING Verilog HDL " 2013 IEEE

Naga Prasad Reddy.T, Avinash.K "Design and PSL Verification of SoC Interconnect Using Open Core Protocol (OCP)" IJCTT Sep 2013

Vikas S. Vij, Raghu Prasad Gudla, Kenneth S. Stevens "Interfacing Synchronous and Asynchronous Domains for Open Core Protocol" 2014 IEEE.

Corresponding Author

Arun Raj S. R.*

Department of Electronics & Communication Engineering, University B.D.T College of Engineering, Davangere, Karnataka

E-Mail – arunrajsr5@gmail.com