# Design and Verification of OCP Master Interface

Aparna R J
Department of EEE
BNMIT
Bangalore, India

Kruthi Jayaram
Department of EEE
BNMIT
Bangalore, India

Niveditha.B.V
Department of EEE
BNMIT
Bangalore, India

***Abstract*— The Open Core Protocol (OCP) delivers the only non-proprietary, openly licensed, core-centric protocol that comprehensively describes the system level integration requirements of intellectual property (IP) cores. While other bus and component interfaces address only the data flow aspects of core communications, the OCP unifies all inter-core communications, including sideband control and test harness signals. The OCP's synchronous unidirectional signaling produces simplified core implementation; integration and timing analysis.OCP eliminates the task of repeatedly defining, verifying, documenting and supporting proprietary interface protocols. The OCP readily adapts to support new core capabilities while limiting test suite modifications for core upgrades. Clearly delineated design boundaries enable cores to be designed independently of other system cores yielding definitive, reusable IP cores with reusable verification and test suites.**

***Keywords—Open Core protocol, Signals and encoding, protocol semantics.***

## I. INTRODUCTION

The Open Core Protocol (OCP) defines a high-performance, bus-independent interface between IP cores that reduces design time, design risk, and manufacturing costs for SOC designs. 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. The Open Core Protocol: Achieves the goal of IP design reuse. 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. This simplifies system verification and testing by providing a firm boundary around each IP core that can be observed, controlled, and validated.

The approach adopted by the Virtual Socket Interface Alliance's (VSIA) Design Working Group on On-Chip Buses (DWGOCB) is to specify a bus wrapper to provide a bus-independent Transaction Protocol-level interface to IP cores. The OCP is equivalent to VSIA's Virtual Component Interface (VCI). While the VCI addresses only data flow aspects of core communications, the OCP is a superset of VCI additionally supporting configurable sideband control signalling and test harness signals. The OCP is the only standard that defines protocols to unify all of the inter-core communication.

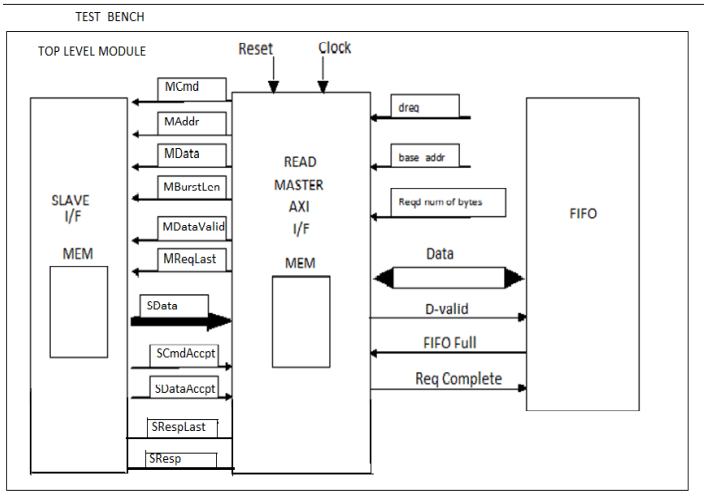


Figure 1: OCP signals for Master-Slave Model Communication

Figure 1 depicts the signals required for this communication, when a blocking hand-shake protocol is used between a master and a slave. When master generates a write/broadcast request, Write Data accompanies the Request. For read and read-exclusive requests, Read Data accompanies the Response. Because of the protocol being blocking, once the master asserts the signals associated with a request, it has to maintain these values until the slave confirm the reception of the request, through the assertion of the Accept Request signal (i.e., SCmdAccept). The transaction is completed upon this confirmation. The request may also be completed at this point, if it is marked as "posted". A write/broadcast request is marked as "posted" when no response is required to acknowledge its reception by the slave. In contrast, a read /read-exclusive request should always be "non-posted", because a response is required to provide the read data. Note that the response channel is also using a blocking hand-shake protocol, thus the master has to assert the Accept Response signal (i.e., MRespAccpt) to confirm the reception of the response. A master may have multiple outstanding requests for different cache lines, but each will create a separate record of a pending response. The stream of pending responses should be returned in-order, to keep the protocol simple.

## II. DESIGN METHODOLOGY

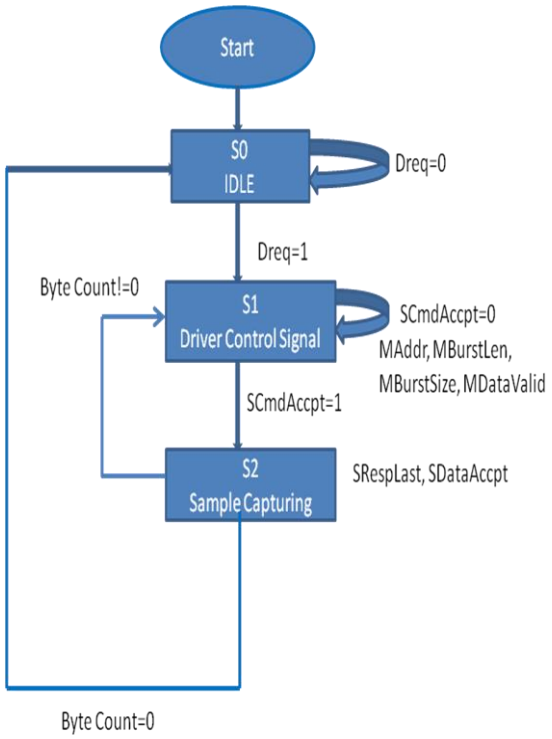


Figure 2: Flow Chart for OCP Master Design

The figure 2 explains that the system will be in idle position S0 at the beginning. Once there is a data request then the state changes from S0 to S1 where S1 is the Driver Control Signal State. Here the state must satisfy the control signals MAddr, MBurstLength, MBurstSize and MDataValid. Depending upon the characteristics of these signals and also if the signal SCmdAccpt is 1 then the state changes from S1 to S2. If SCmdAccpt is 0 then it will be in S1 state only. Now when the state is S2 it means that the data is captured. This means that here the state checks for the signal SDataAccpt means the slave accepts the data from the master and SRespLast where it checks when the last data is transferred. Here a counter is set for checking how many bytes are transferred in each burst. If the byte count is 0 then this state goes back to idle state S0. If the byte count is not equal to 0 then it goes to state S1 for again driving the control signals. The process keeps on continuing until the last data transferring is completed.

## III. THEORY OF OPERATION

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. Given the wide range of IP core functionality, performance and interface requirements, a fixed definition interface protocol cannot address the full spectrum of requirements.

The need to support verification and test requirements adds an even higher level of complexity to the interface. To address this spectrum of interface definitions, the OCP defines a highly configurable interface. The OCP's structured methodology includes all of the signals required to describe an IP cores' communications including data flow, control, and verification and test signals. This chapter provides an overview of the concepts behind the Open Core Protocol introduces the terminology used to describe the interface and offers a high-level view of the protocol.

### *A. Point to point Synchronous interface*

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.

### *B. Bus Independance*

A core utilizing the OCP can be interfaced to any bus. A test of any bus-independent interface is to connect a master to a slave without an intervening on chip bus. This test not only drives the specification towards a fully symmetric interface but helps to clarify other issues. For instance, device selection techniques vary greatly among on-chip buses. Some use address decoders. Others generate independent device select signals (analogous to a board level chip select). This complexity should be hidden from IP cores, especially since in the directly-connected case there is no decode/selection logic. OCP-compliant slaves receive device selection information integrated into the basic command field.

Arbitration schemes vary widely. Since there is virtually no arbitration in the directly-connected case, arbitration for any shared resource is the sole responsibility of the logic on the bus side of the OCP. This permits OCP compliant masters to pass a command field across the OCP that the bus interface logic converts into an arbitration request sequence.

### *C. Interrupts, Errors, and other Sideband Signalling*

While moving data between devices is a central requirement of on-chip communication systems, other types of communications are also important. Different types of control signalling are required to coordinate data transfers (for instance, high-level flow control) or signal system events (such as interrupts). Dedicated point-to-point data communication is sometimes required. Many devices also require the ability to notify the system of errors that may be unrelated to address/data transfers.

The OCP refers to all such communication as sideband (or out-of-band) signalling, since it is not directly related to the protocol state machines of the dataflow portion of the OCP. The OCP provides support for such signals through sideband signalling extensions.

Errors are reported across the OCP using two mechanisms. The error response code in the response field describes errors resulting from OCP transfers that provide responses. Write-type commands without responses cannot use the in-band reporting mechanism. The second method for reporting errors across the OCP uses out-of band error fields. These signals report more generic sideband errors, including those associated with posted write commands.

## IV. SIGNALS AND ENCODING

OCP interface signals are grouped into dataflow, sideband, and test signals. The dataflow signals are divided into basic signals, simple extensions, burst extensions, and thread extensions. A small set of the signals from the basic dataflow group is required in all OCP configurations. Optional signals can be configured to support additional core communication requirements. All sideband and test signals are optional.

The OCP is a synchronous interface with a single clock signal. All OCP signals are driven with respect to, and sampled by, the rising edge of the OCP clock. Except for clock, OCP signals are strictly point-to-point and uni-directional. The complete set of OCP signals is shown in Figure 3.

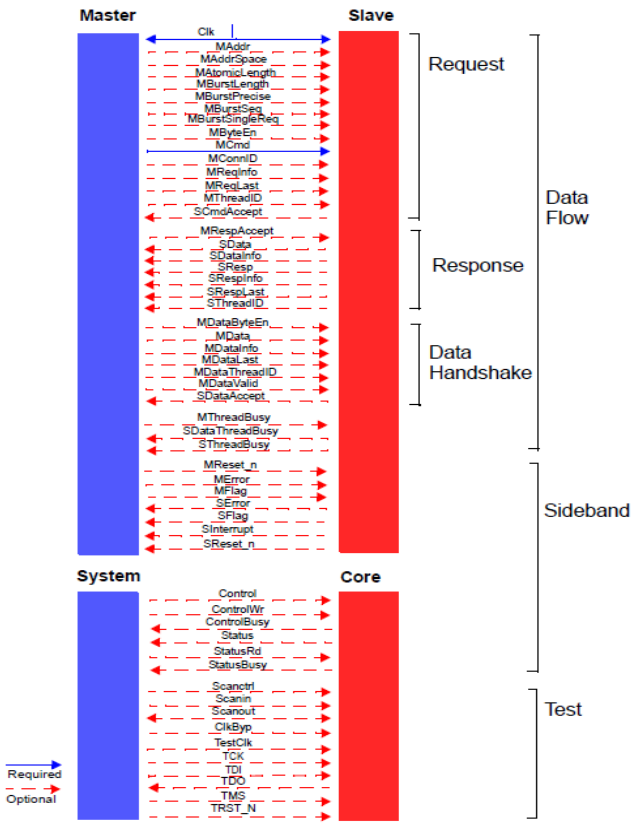


FIGURE 3: OCP SIGNALS

## V. PROTOCOL SEMANTICS

Figure 4 provides a graphic view of the hierarchy of elements that compose the OCP.

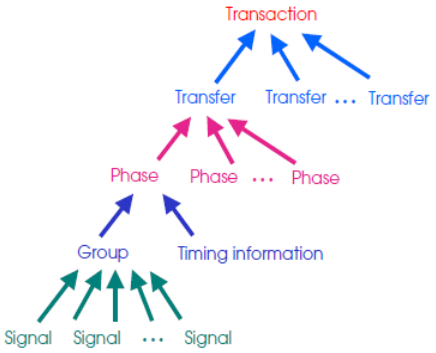


Figure 4: Hierarchy of Elements

Some OCP fields are grouped together because they must be active at the same time. The data flow signals are divided into three signal groups: request signals, response signals, and data handshake signals. A list of the signals that belong to each group is shown in Table 1.

**Table 1: OCP Signal Groups**

| Group | Signal | Condition |
|---|---|---|
| **Request** | MAddr | Always |
| | MAddrSpace | Always |
| | MAtomicLength | Always |
| | MBurstLength | Always |
| | MBurstPrecise | Always |
| | MBurstSeq | Always |
| | MBurstSingleReq | Always |
| | MByteEn | Always |
| | MCmd | Always |
| | MConnID | Always |
| | Mdata* | Data Handshake=0 |
| | MdataInfo* | Data Handshake=0 |
| | MReqInfo | Always |
| | MReqLast | Always |
| | MThreadID | Always |
| **Response** | Sdata | Always |
| | SDataInfo | Always |
| | SResp | Always |
| | SRespInfo | Always |
| | SRespLast | Always |
| | SThreadID | Always |
| **Data Handshake** | Mdata* | Data Handshake=1 |
| | MDataByteEn | Always |
| | MdataInfo* | Data Handshake=1 |
| | MDataLast | Always |
| | MDataThreadID | Always |
| | MDataValid | Always |

Combinational paths are not allowed within the sideband and test signals, or between those signals and the data flow signals. The only legal combinational dependencies are within the data flow signals. Data flow signals, however, may be combinationally derived from MReset_n and SReset_n.

## VI. OCP CORE PERFORMANCE

### A. *Report Instructions*

To document the core, you will need to provide the following information:

- Core name. Identify the core by the name you assigned.
- Core ID. Specify the precise identification of the core inside the system-on- chip. The information consists of the vendor code, core code, and revision code.
- Core is/is not process dependent. Specify whether the core is process dependent or not. This is important for the frequency, area, and power estimates that follow. If multiple processes are supported, name them here and specify corresponding frequency/area/power numbers separately for each core if they are known.
- Frequency range for this core. Specify the frequency range that the core can run at. If there are conditions attached, state them clearly.
- Specify the area that the core occupies. State how the number was derived and be precise about the units used.
- Power estimate. Specify an estimate of the power that the core consumes.
- Special reset requirements.
- Number of interfaces.
- Interface information. For each OCP interface that the core provides, list the name and type.

## VII. SIMULATION RESULTS

The simulation result shown in figure 5 is the Finite State Machine for reading the data from the master.

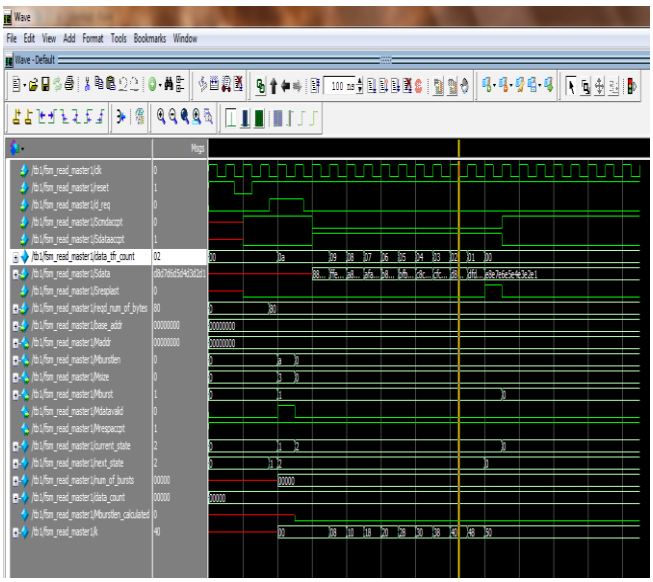

Figure 5: FSM Read Master

In this result there are 3 stages. First is the idle state. Second is the Driving the control signals and third is the capturing of the data. The input given to the master is Mdata. Once the input is valid i.e. if MDataValid is 1 it will read the signal and then sent as the request signal to the slave. Now the slave will accept the data by making the signal SDataAccpt valid i.e.1. Now the slave responses back to master by reading the valid data. Once the data reaches the master the master acknowledges the slave by accepting the response from slave making the signal MRespAccpt 1. The required number of bytes taken here is calculated and also the base address will be given which will be constant for all the simulation. For simplification we have taken number of bytes as 8 for each transfer.

Figure 6 shown below is as same as 5 but only the required number of bytes is changed.

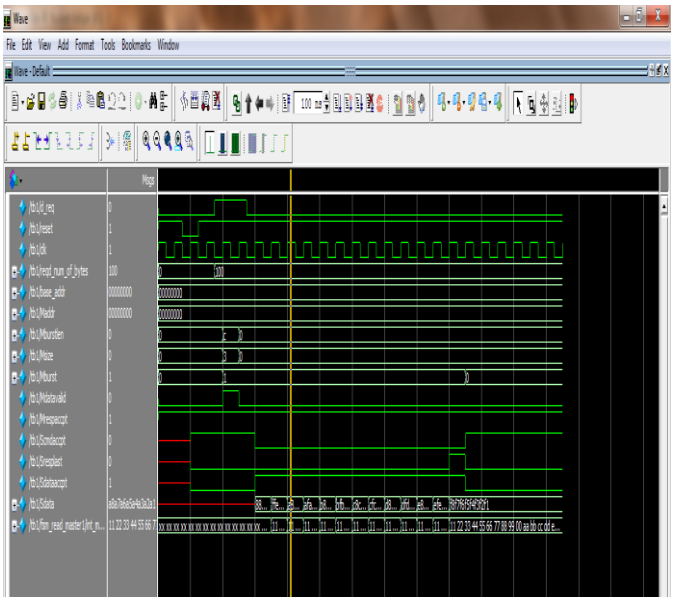

Figure 6: FSM Read Master for 100 bytes transfer

As shown in the figure 5 we see that 100 bytes cannot be transferred in a single burst since we have taken only 8 bytes of transfer per bursts. Since there is 100 bytes to be transferred multiple bursts are executed continuously without any delay in the data transfer.

The result shown in figure 7 is for the memory of OCP.

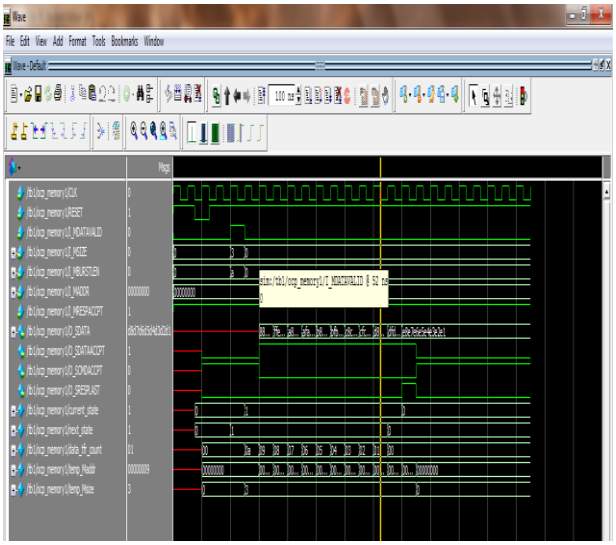

Figure 7: OCP Memory

This design consists of only 2 stages. One is the address stage and second one is the data stage. When the data from the master is given to the slave the address in which the data is stored will be given back to the master by making SDataAccpt as 1. Once the data is received by the master from slave the master acknowledges the slave by making the signal MRespAccpt 1. Here only the data is transferred from slave to master which will be the input to FIFO from the master output. From FIFO the output will be given to the required system application as input. The transcript is as shown below.

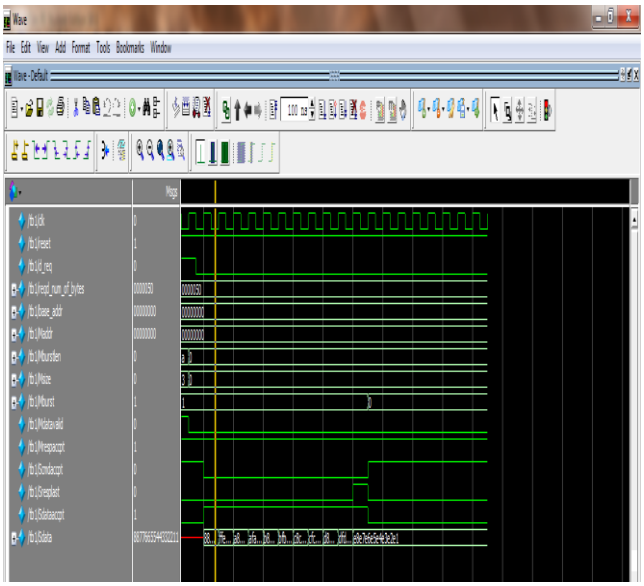

Figure 8: Test Bench for overall Verification of OCP Master Interface

This result is for the verification of both slave and master model. In the figure when the control signal MRespAccpt and SDataAccpt is 1 with Mburst value as 1 the transfer of data in each burst where the required number of bytes is transferred without any delay. The reading of the data is done from master as well as the response from slave is done sequentially which will be given as the input to the FIFO from master. This result shows that the data is transferred from master to slave or vice versa to read the given data i.e. input and produce the required output without having any errors

## VIII. CONCLUSION

This paper is aimed at providing the overview of function of the OCP protocol and deeper understanding about its characteristics and how it operates. This project also improves our knowledge in knowing the protocol and its usage in the present technology.

Here we will be able to know the application of OCP protocol in the latest technologies like encoders, decoders, imaging applications, mobile world, etc. It also briefs us about different types of OCP timing diagrams, core information, etc that are available and also their advantages and disadvantages. We also learn the fundamentals of the protocol and its signals with their interfacing are explained. More features can also be added in future to enhance the system operation with respect toitsapplication.

## REFERENCES

[1] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An open source tool for symbolic model checking. In Proc. Computer Aid Verification, pages 241–268, Jul 2002.

[2] E. A. Emerson. Temporal and Modal Logic, Handbook of Theoretical Computer Science, volume B. Elsevier and MIT Press, 1990.

[3] D. Flynn. Amba: Enabling reusable on-chip designs. IEEE Micro, 17(4):20–27, Jul 1997.

[4] M. Loghi, M. Poncino, and L. Benini. Cache coherence tradeoff s in shared-memory MPSoCs. ACM Transactions on Embedded Computing Systems, 5(2):383 – 407, May 2006.

[5] OCP-IP, “Open core protocol international partnership”, http://www.ocpip.org/, 2007.

[6] Vesa Lahtinen, “System level design experiences and the need for standardization”, in Proc. IEEE Int’l Conf. on System-on-Chip, Tampere, Finland, Nov. 2006, pp. 1–4.

[7] James Aldis, “Use of http://www.ocpip.org/, 2005.

[8] Partha Pratim Pande, Cristian Grecu, Michael Jones, Andr´eIvanov, and Resve Saleh, “Performance evaluation and design trade-offs for network-on-chip interconnect architectures”, IEEE Trans. Computers, vol. 54, no. 8, pp. 1025–1040, Aug. 2005.

[9] Natale Barsotti, Riccardo Mariani, Matteo Martinelli, and Mario Pasquariello, “Dynamic verification of OCP-based SoC”, in Proc. IEEE Int’l Conf. on System-on-Chip, Tampere, Finland, Nov. 2005

[10] Sonics Inc., “SMART http://www.sonicsinc.com/, 2007.

[11] ArterisT M, “Automated http://www.arteris.com/, 2007.

[12] Silistix, “CHAINworksT M”, http://www.silistix.com/, 2007.