

IMPLEMENTATION OF BUS BRIDGE BETWEEN AHB AND OCP

¹G. Geetha Reddy, ²K. Vanisree, ³David Solomon Raju. Y

¹M. Tech. Student (Embedded Systems), Holy Mary Institute of Technology and Science, AP, India

²Associate Professor, Dept. of ECE, Holy Mary Institute of Technology and Science, AP, India

³Associate Professor, Dept. of ECE, Holy Mary Institute of Technology and Science, AP, India

Abstract-Protocols are commonly used today to connect IP blocks on structured SoCs. Generally Protocol is the back-bone of the SoC and its failure usually leads to a non-functional chip. In present market, various types of standard protocols are available and are used in SoC which requires a bridge to pass the information from one type of protocol to other type of protocol safely and without any data loss. Open Core Protocol (OCP) and AMBA – Advanced High Performance Bus (AHB) protocol are standard and commonly used protocols. In this work, the bus bridge was designed to interface these protocols which plays a vital role in SoC application such as it may lead to application failure, if it doesn't work properly. Initially basic OCP and AHB protocols are modeled separately using VHDL and are simulated. Basically Bus Bridge should convert command and data of OCP formats to acceptable AHB formats. This conversion does not ensure proper communication unless the timings of each protocol were met. Hence the interconnecting Bus Bridge wrapper between Core Centric Protocol (OCP) and Bus Centric Protocol (AHB) was designed with proper timing delay. The simulation results obtained were analyzed to adjust the timing of the design. Overall timing of various signals present in the design was optimized to meet the design objectives.

Keywords – System on Chip, ABV, OCP, AHB
I.INTRODUCTION

Basic idea is to provide a bus bridge between the two protocols in order to perform the proper and lossless communication between the IP cores which using various protocols on the System on Chip (SOC) system. Basically, an SOC is a system which is considered as a set of components and interconnects among them. The dataflow will happen in the system in order to achieve a successful process and hence for which the various interfaces is required. If these interfaces have issues, then the process to be achieved will fail which leads to fail of whole application. Generally, in an SOC system, the various protocols can be used as interfaces based on the application and also the designer. The various interfaces are used because every interface has its own properties which suits for the corresponding application.

This paper is chosen because currently the issues are increased in the industries due to the lack of proper data transferring between the IP cores on the System on Chip (SOC) system. 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. Hence to resolve this issue, the standard protocol buses are used in order to interface the two IP cores. Here the loss of data depends on the standards of protocols used. Most of the IP cores will uses Open Core Protocol (OCP) due to its flexibility and functionalities to communicate between them. Similarly ARM uses the AMBA (Advanced Microcontroller Bus Architecture) which has AHB (Advanced High-Performance Bus). This bus also has its own advantages and flexibilities. Now another important issue comes into picture is that the communication i.e. interface between these interfacing buses. The data or signal flow between the protocols cannot be done easily because each protocol will be having its own properties and advantages which are to be considered mainly. So in order to interface them, a Bus Bridge must be created which should have the properties to covert the signals of one form to another which is of acceptable by the receiving end protocol [4]. One more thing to be considered when designing the Bus Bridge is that the signal loss while converting the signal in acceptable form. Basically, ARM uses the AMBA AHB protocol for its data transfer (reading the stored values from SRAM) where as the SRAM controller uses the OCP which is a core centric protocol for its data transfer. Now, the issue arises in this part i.e. the data should be transferred to the processor from the SRAM which are doing their transaction in different protocols [5]. So in order to fetch the data from SRAM, the OCP to AHB bridge converter is required which converts the data in the acceptable form at other end. This magnified

block has the OCP AMBA AHB Master and Slave wrappers which gives the clear view of the paper. Here ARM processor that uses AHB bus protocol will issue the command which acts as the master and its slave is connected to the bridge. Similarly SRAM uses OCP acts as slave whose master is connected to bridge. Now the bridge is the one which will convert the AHB command to the OCP command and transfer the data to SRAM successfully. Here the importance of paper comes into picture i.e. "Converter between OCP and AHB Bridge plays a vital role by doing its transaction from SRAM to ARM Processor, which will make the application fail when it doesn't work properly".

First section gives a basic idea and need for this paper and also a brief introduction to the generalized block diagram of the application targeted, where this paper comes in that application. Second section of the thesis literature review is carried out which includes merits and demerits, basic block diagram and the data flow signal diagram of OCP and AHB. Based on the review, the specifications of the protocols have been arrived. Finally the architecture is identified for the Bus Bridge. This section also includes the merits and demerits of the paper referred and comparison of the OCP and AHB protocols. Third section discusses problem definition and objectives of the paper. Methodologies to achieve the objectives of the paper are clearly mentioned. Fourth section deals with the FSMs developed for the OCP and AHB for simple write and read and burst operation. Then OCP and AHB are modeled with respect to the developed FSMs and based on which the Bus Bridge is designed. The screen shots of the simulated waveforms of each supporting feature is explained. Fifth section discusses the Assertion Based Verification (ABV) and the verification scenarios for each module. Sixth section gives the conclusion and future work required for the paper.

II. LITERATURE REVIEW

The literature survey is carried out for the protocols OCP and AMBA-AHB separately and is compared with respect to their own functionalities.

Basically, this paper can be classified into three major parts.

- Open Core Protocol (OCP)
- Advanced High-Performance Bus (AHB) Protocol
- Bus Bridge between OCP and AHB

The above classification will be discussed individually in the following sessions.

a) Open Core Protocol (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 [6]. 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 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 [6]. The OCP readily adapts to support new core capabilities while limiting test suite modifications for core upgrades.

The block diagram which explains the basic operation and characteristics of OCP is shown in Figure 2.1. 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.

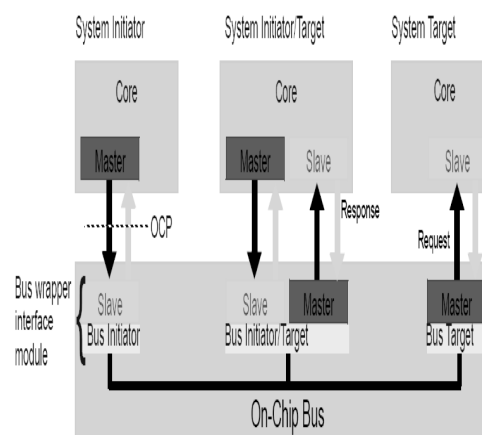


Figure 1: Basic block diagram of OCP instance

Figure1 shows a simple system containing a wrapped bus and three IP core entities such as one that is a system target, one that is a system initiator, and an entity that is both. The characteristics of the IP core determine whether the core needs master, slave, or both sides of the OCP and the wrapper interface modules must act as the complementary side of

the OCP for each connected entity. A transfer across this system occurs as follows.

A system initiator (as the OCP master) presents command, control, and possibly data to its connected slave (a bus wrapper interface module). The interface module plays the request across the on-chip bus system. The OCP does not specify the embedded bus functionality. Instead, the interface designer converts the OCP request into an embedded bus transfer. The receiving bus wrapper interface module (as the OCP master) converts the embedded bus operation into a legal OCP command. The system target (OCP slave) receives the command and takes the requested action.

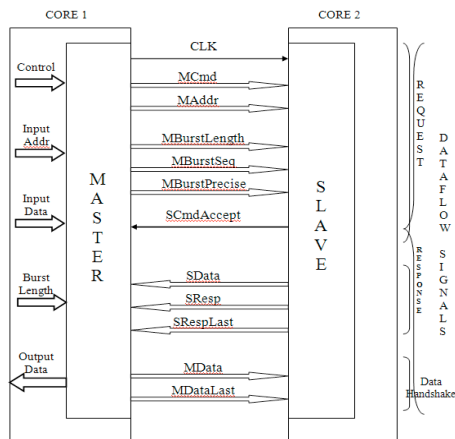


Figure 2: OCP dataflow signals

From the Figure 2.2, the inputs and outputs of the OCP are clearly identified which are discussed as follows. The specifications for the Open Core Protocol are identified for both simple write and read operation and burst operation. The identified specifications are represented in tabulation format. Simple write and read operation for which the basic and mandatory signals required signals are tabulated in Table 1

Table 1: Basic OCP signal specification

S. No.	NAME	WIDTH	DRIVER	FUNCTION
1	Clk	1	Varies	OCP Clock
2	MCmd	3	Master	Transfer Command
3	MAddr	13 (Configurable)	Master	Transfer address
4	MData	8 (Configurable)	Master	Write data
5	SCmdAccept	1	Slave	Slave accepts transfer
6	SData	1	Slave	Read data
7	SResp	2	Slave	Transfer response

The request issued by system is given to slave by MCmd signal. Similarly, in Write operation, the input address and data provided by the system will be given to slave through the signal MAddr and MData and when those information's are accepted, slave will give SCmdAccept signal which ensures that the system can issue next request. During Read operation, system issues the request and address to slave which will set SResp and fetch the corresponding data that is given to output through SData. Thus the OCP basic block diagram, dataflow signal diagram and its specifications are tabulated and hence give the clear view in designing the Open Core Protocol bus.

b) AMBA-AHB Protocol

The AHB (Advanced High-performance Bus) is a high-performance bus in AMBA (Advanced Microcontroller Bus Architecture) family. This AHB can be used in high clock frequency system modules. The AHB acts as the high-performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macro cell functions. AHB is also specified to ensure ease of use in an efficient design flow using automated test techniques [2]. This AHB is a technology-independent and ensure that highly reusable peripheral and system macro cells can be migrated across a diverse range of IC processes and be appropriate for full-custom, standard cell and gate array technologies.

The block diagram of the Advanced High-Performance Bus Protocol is shown in the Figure 2.3.

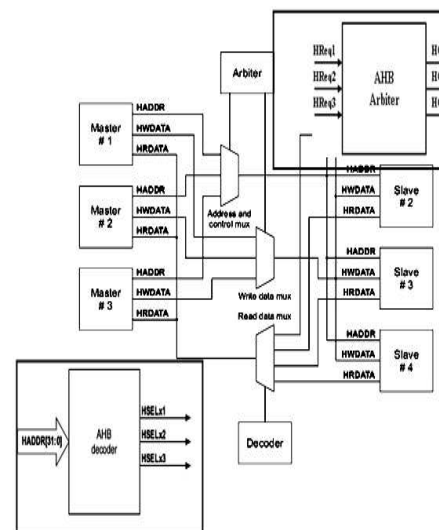


Figure 3: AMBA – AHB block diagram

Totally this block diagram comprises of four components- Arbiter, Master, Slave, and Decoder

Arbiter - The arbitration mechanism is used to ensure that only one master has access to the bus at any one time. The arbiter performs this function by observing a number of different requests to use the bus and deciding which is currently the highest priority master requesting the bus.

Master - A bus master is able to initiate read and write information by providing address and control information. Only one bus master can use the bus at the same time. An AHB bus master has the most complex bus interface in an AMBA system. Typically an AMBA system designer would use predesigned bus masters and therefore would not need to be concerned with the detail of the bus master interface. No provision is made within the AHB specification for a bus master to cancel a transfer once it has commenced.

Slave - After a master has started a transfer, the slave then determines how the transfer should progress. Whenever a slave is accessed it must provide a response which indicates the status of the transfer. The HREADY signal is used to extend the transfer and this works in combination with the response signal HRESP which provide the status of the transfer. The slave can complete the transfer in a number of ways. It can:

- Complete the transfer immediately
- Signal an error to indicate that the transfer has failed
- Delay the completion of the transfer, but allow the master and slave to back off the bus, leaving it available for other transfers.

Decoder - The AHB decoder is used to decode the address of each transfer and provide a select signal for the slave that is involved in the transfer. A central address decoder is used to provide a select signal 'HSELx' for each slave on the bus. The select signal is a combinatorial decode of the high-order address signals. A slave must only sample the address and control signals and HSELx is asserted when HREADY is HIGH, indicating that the current transfer is completing.

The AMBA AHB bus protocol is designed with a central multiplexor interconnection scheme. Using this scheme all bus masters drive out the address and control signals indicating the transfer, they wish to perform and the arbiter determines which master has its address and control signals routed to all of the slaves. Before which initially the master who needs to perform the operation should give the request signal to the arbiter and the arbiter will give the grant signal to the master for further proceedings. Similarly, a decoder is used to

select the slave which has to be active during the operation based on the address given by the master. A central decoder is also required to control the read data and response signal multiplexor, which selects the appropriate signals from the slave that is involved in the transfer. These make the read and write operation smoothly. Thus the working of AMBA AHB protocol is explained with the help of its block diagram shown in Figure 3.

c) Specification

The signals involved in designing the AMBA AHB are listed in the Table 2.3 which also gives the specification of each signal.

Table 2: AMBA AHB signal specification

S. No.	NAME	WIDTH	DRIVER	FUNCTION
1	HCLK	1	Clock Source	This clock times all bus transfers at the rising edge of HCLK
2	HADDR	32	Master	The system address bus of width 32-bit
3	HTRANS	2	Master	Indicates the type of the current transfer happening
4	HWRITE	1	Master	When HIGH this signal indicates a write transfer and when LOW a read transfer
5	HSIZE	3	Master	Indicates the size of the transfer
6	HBURST	3	Master	Indicates if the transfer forms part of a burst.
7	HWDATA	8	Master	The write data bus is used to transfer data from the master to the bus slaves during write operations.
8	HSELx	1	Decoder	Each AHB slave has its own slave select signal and this signal indicates that the current transfer is intended for the selected

				slave.
9	HRDATA	8	Slave	The read data bus is used to transfer data from bus slaves to the bus master during read operations.
10	HREADY	1	Slave	When HIGH the HREADY signal indicates that a transfer has finished on the bus. This signal may be driven LOW to extend a transfer.
11	HRESP	2	Slave	The transfer response provides additional information on the status of a transfer

The table also includes the function of each signal and the source from which the each signal is driven. The width of each signal is specified and hence the address has 32 bits and both write and read data also has 32 bits. The operation is performed in a synchronized clock frequency and hence the signals should be changed with respect to the rising edge of the clock.

d) Bus Bridge

Basically a bus can be defined as a tool designed to interconnect the functional blocks in a systematic manner. It provides standardization in communication protocols between board-level devices. The Master and Slave AMBA AHB bus wrappers are used to connect bus independent Intellectual Properties (IPs) to an AMBA AHB bus controller. The bus bridge is the one which bridges any two protocols such as AHB and OCP in order to process the data from one block to the other through interconnection. This allows integrating OCP compliant IPs in their system core by enabling a practical reuse methodology.

Based on the basic block diagram and the specifications of the OCP and AHB, designing a bus bridge between these two protocols are made clear. In the bus bridge design, totally two versions of the wrapper are available, one for bridge level and another for top level. They both manage simple and burst transfers in order to increase the bus throughput [10].

e) Signal Flow Diagram

The Signal flow diagram of OCP-AHB Bridge & Top Level Wrapper is shown clearly in the Figure 2.4. This enables the clear understanding of the bus bridge designing and the overall operation. This also reveals the number of input and output signals required to design a bus bridge between the AHB and OCP buses. The internal signals also made visible in the signal flow diagram which makes the design much simple and the description of these identified signals are given in the Table 2.2 and Table 2.3. The OCP-AHB bus bridge is designed with a synchronous clock and is maintained in the whole design. On-chip buses are generally synchronous and in a synchronous bus all the operations are synchronized to a global bus clock.

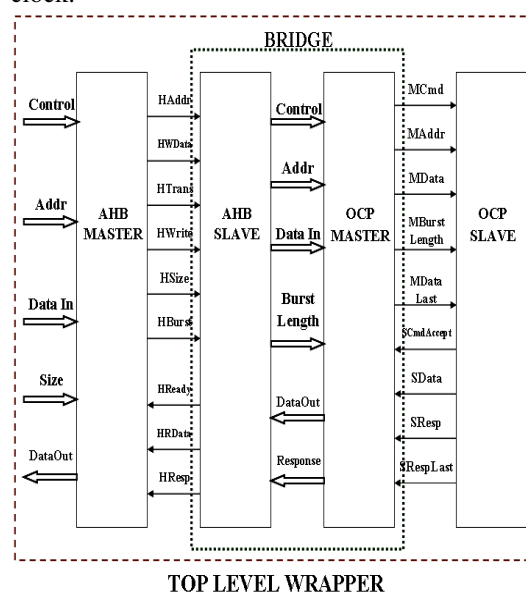


Figure 4: Signal flow diagram of OCP-AHB bridge & top level wrapper

Generally, AHB will be connected to the processor which will be communicating to an external memory in order to fetch or store data in it. This external memory will be using the OCP bus for its data transferring. Thus the processor will be acting as a system which will be providing the control or command through AHB master to the memory through OCP slave in order to transfer data.

Bridge - Basically the bridge contains the AHB slave and OCP master at the synchronized clock. So the AHB slave is fed by the AHB master and the output of the AHB slave is managed in order to feed those signals to the OCP master. The output signals of AHB slave are port mapped in such a way that the OCP master accepts it. The mapping between the AHB slave and OCP master is done such as the HAddr signal is mapped to Addr, HSize to BurstLength, HWDat to data_in during write operation and data_out to

HRData during read operation. AHB does not support the pipelining operation whereas the OCP will support the pipelining operation. Due to this variation, the designed OCP-AHB Bridge will not support the pipelining operation which can be said as a limitation of the design, whereas this bridge design supports the simple and burst operation. Hence inside the bridge, the AHB slave and OCP master is synchronized by taking care of delay of each signals and the timing of control, address and data.

Write operation - Initially the system will give the control, address and the data to the AHB master which in turn gives that information to the bridge. Then the bridge will pass the information to the OCP slave which will write the given data in to its internal memory for that particular address during the write operation.

Read operation - Similarly during the read operation, the address will be given by the system and the OCP slave will fetch the data for that particular address and will give that to the bridge. Then the bridge will give the data to AHB master and hence the output data is achieved at the "data_out". Thus the basic blocks, specification and merits and demerits of the OCP, AHB and the AHB-OCP Bridge are identified.

The literature survey is carried out with merits and demerits of OCP and AHB and the signal flow diagram is identified. The specification for the signals shown in the signal flow diagram is identified and its working is explained with the help of its block diagram. The comparison of the Core Centric Protocol (OCP) and Bus Centric Protocol (AHB) is made with respect to their properties and is tabulated. Finally the signal flow diagram is developed with respect to the identified specification and the working of the OCP and AHB.

III. DESIGN OF OCP, AHB AND BUS BRIDGE

The literature survey on the OCP, AHB and Bus Bridge is made and the basic signal flow block diagram is identified. In the mentioned dataflow signal diagram, the basic signals are identified and are used in the simple read and write and burst operation in OCP and AHB Master and Slave. Initially the Finite State Machine (FSM) is developed and the modeling of the developed FSM is done using the VHDL. Based on this developed OCP and AHB, the Bus Bridge is designed. The design of each protocol such as OCP and AHB is discussed and their simulations are verified with the basic operation. Similarly the design of Bus Bridge is also discussed with the simulation results.

a) Design of Open Core Protocol

The design of the Open Core Protocol starts with the initial study based on which the development of FSM for the various supporting operation after which the development of VHDL for the FSM. The notations used while designing the OCP are listed in the Table 3.1, Table 3.2 and Table 3.3 which are as follows.

Table 3: Input control values

Control	Notations Used	Command
000	IDL	Idle
001	WR	Write
010	RD	Read
011	INCR_WR	Burst_Write
100	INCR_RD	Burst_Read

Table 4: OCP master command (MCmd) values

MCmd	Notations Used	Command
000	IDL	Idle
001	WR	Write
010	RD	Read

Table 5: Slave response (SResp) values

SResp	Notations Used	Response
00	NUL	No Response
01	DVA	Data Valid / Accept

b) Design of AMBA - AHB

The AHB takes on many characteristics of a standard plug-in bus. It's a multi-master with arbitration, putting the address on the bus, followed by the data. It has a data-valid signal (HREADY). This bus differs in that it has separate read (HRDATA) and write (HWDATA) buses whose connections are multiplexed, rather than making use of a tri-state multiple connection. AHB supports bursts with 4, 8, and 16 beat bursts and single transfers. The notations used while designing the AHB for the system Control signals are mentioned in the Table 3.1 and for others are listed in the Table 3.4 and Table 3.5 which are as follows.

Transfer type (HTrans)

Table 6: Transfer type (HTrans)

HTrans	Notations Used	Description
00	IDE	No Data Transfer
10	NON_SEQ	The address and control signals are unrelated to the previous transfer
11	SEQ	The address is related to the previous transfer

HBurst values

Table 7: HBurst values

HBurst	Description
000	Represents Burst Size of 4
001	Represents Burst Size of 8
010	Represents Burst Size of 16

IV. SIMULATION RESULTS

Initially the Control, input address and data to the AHB Master based on which it will go to either write or read state. If write request (Control = "001") is given, AHB Master will go to WRITE state only when the grant signal is given from the arbiter and hence it will issue HAddr, HWDData and HWrite = 1.

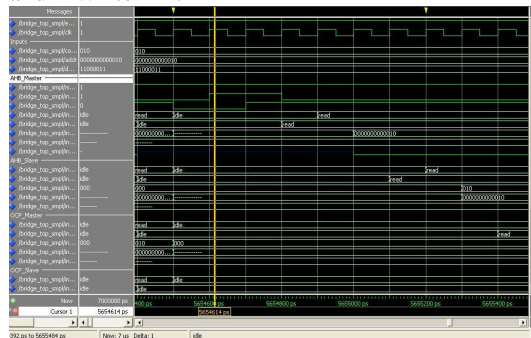


Figure 5: Bridge Simple Write

These output will leads the AHB Slave go to WRITE state which will provide Control, Addr and data_in to the next OCP Master. With this inputs OCP Master will go WRITE state and will issue MCmd, MAddr and MData which in turn lead the OCP Slave to WRITE state. Once the input data is written to the specified slave memory location, it will give 'SCmdAccept' signal as response to OCP Master which in turn gives 'Response' signal to AHB Slave. Now the AHB Slave will issue the 'HReady' signal to AHB Master and hence both master and slave of OCP and AHB will go IDLE state

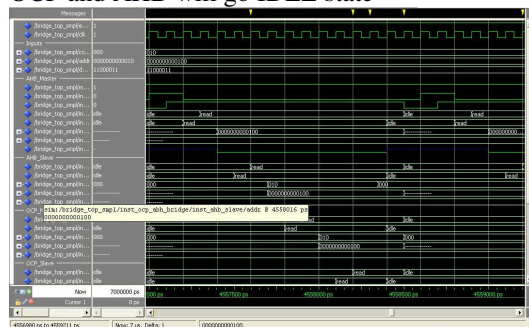


Figure 6: Bridge Simple Read

In the same way, when read request (Control = "010") is given with the proper address, AHB Master will go to the READ state after getting the grant from arbiter and will make the HWrite to Low (HWrite = 0). This makes AHB Slave go to READ state and issue Control and Addr which will leads the OCP Master to go READ state and in turn give MCmd and MAddr.

Now the OCP Slave will go READ state and will fetch the data for the given address location. Once it issued 'SCmdAccept and SResp' which means that data is ready and is passed to the OCP Master. This will pass the data to AHB Slave through data_out signal and makes 'Response' signal high.

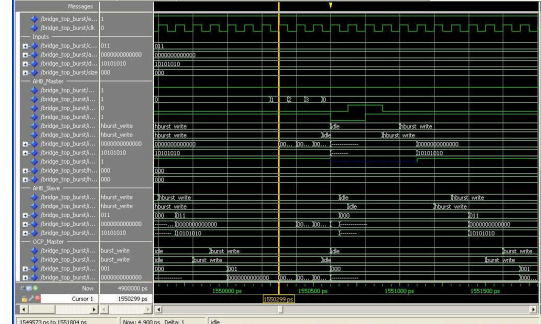


Figure 7: Bus Bridge Write and Read

The simulation result for Bus Bridge of Burst size_4 is shown which represents the proper sequence of Write and Read operation. Here HSize is given as "000" which represent the four burst operation continuously. The input data is written to the memory with respect to the Response signals and the same way is followed in reading a data from a particular memory location. Here the count is used in order to generate the address in sequence and the data is given as input to write in to the memory location. The count get resets only when the burst operation is over and hence all the master and slave will go to IDLE state

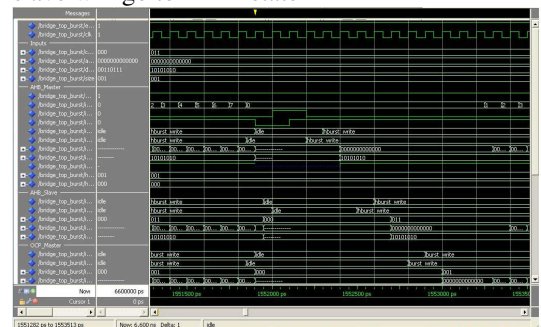


Figure 8: Bus Bridge Write Operation Burst Size_8

Here the size is given as "001" which represents the burst size of 8 and this contains the eight continuous operation of write or read operation. The simulation result for the write operation is shown in the Figure 4.4. Similarly the Burst read operation of size 8 is shown in the Figure 4.5 which clearly shows the generation of 8 addresses and corresponding data is read out. The slave will go IDLE state when the memory locations are full i.e. when the address value exceeds the number of memory locations in a slave, then that slave will go to IDLE and next slave will get activated

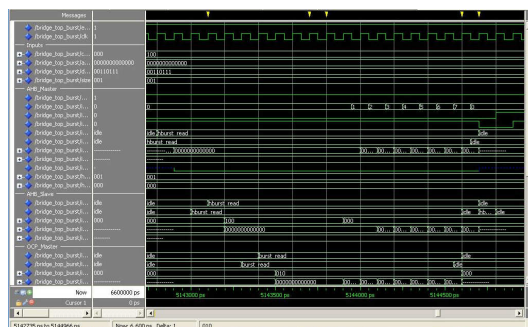


Figure 9: Bus Bridge Read Operation Burst size_8

The Read operation for the burst size of 8 is shown in the waveform in Figure 4.5 which clearly shows that the HWrite is assigned to 0. Similarly the Response signals such as “SCmdAccept, Response and HReady” also clearly represents that each signal is activated based on the previous signal i.e. each signal depends on the other signals. The similar operation is carried out as discussed in the burst of size 4. The only difference is that the count is getting extended up to 8 based on the burst size

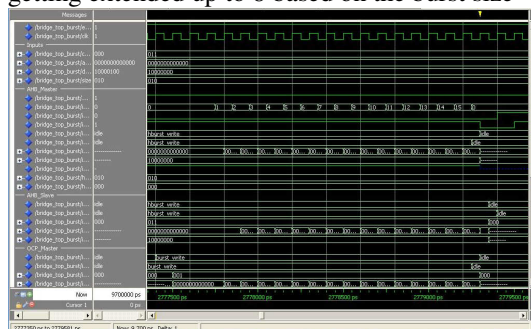


Figure 10: Bus Bridge Write Operation Burst Size_16

The burst size is given as “010” which says that the AHB Master should perform 16 write or read operations and then it should go IDLE state. The Write operation of burst size 16 is shown in the Figure 4.6. Similarly the read operation for the burst size of 16 is shown and it shows the data corresponding to the given address is read out successfully from the memory present in the OCP Slave. Once the all the process in the burst got over, both master and slave of OCP and AHB will go IDLE state.

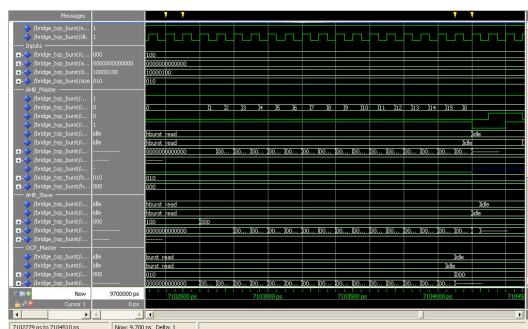


Figure 11: Bus Bridge Read Operation Burst Size_16

V. CONCLUSION

This paper work presents the Bus Bridge design which acts as an interface between two different protocols such as Open Core Protocol (OCP) and AMBA- Advanced High Performance Bus (AHB) protocol. In this work, initially the investigation on the OCP is carried out and the basic commands and its working are identified based on which the signal flow diagram and the specifications are developed for designing the OCP using VHDL. This OCP will include three types of operation such as Simple Write and Read and Burst Operation. Similar way is followed in order to identify the specification for designing AHB. This AHB includes two types of operations such as Simple Write and Read and Burst Operation. Based on the OCP and AHB design, the signal flow diagram is identified and developed for the Bus Bridge design.

The simulation result shows that the communication between AHB and OCP through the bridge is proper. All of the commands and data are successfully transferred from AHB to OCP protocol by the bridge. There is no loss of data or control information. The Bus Bridge supports the simple write and read operation and the burst extension (with sizes supported by AHB). Based on the result obtained, the burst extension is seen to automate the address generation. The initial address alone is provided to the bridge. The Various Scenarios for each component in the Bus Bridge design are verified effectively during the simulation with respect to its behavior.

a) Recommendations for Future Work

The design can be further extended by developing a total system around it. For Example, we can use this bridge to interface between an ARM processor and any OCP based device (like SRAM). Burst Mode can be extended further to include various supported types of burst. The wrapping burst is that once the burst size is reached, the burst operation will be continued until the specified value of the wrapping burst.

- 4-beat wrapping burst
- 8-beat wrapping burst
- 16-beat wrapping burst

This paper work provides an ideal platform for enhancement or further development of the Bus Bridge Design between OCP and AHB protocols.

VI. ACKNOWLEDGMENTS

I am gratefully thankful to management and principal of Holy Mary Institute of Technology and Science, for giving the opportunity to utilise the tools available in the dept. of ECE in completing the above task with successful completion.

VII. REFERENCES

- [1] Jeongwoo Park, Kwangjae Lee, Jeonghun Kim, Kwang-Hyun Baek and Suki Kim, "An MDDI-Host Architecture with Low Complexity for SoC Platforms", IEEE Transactions on Consumer Electronics, Vol. 53, pp. 1668-1673, No. 4, Nov 2007.
- [2] S. Osborne, A.T. Erdogan, T. Arslan and D. Robinson, "Bus encoding architecture for low-power implementation of an AMBA-based SoC platform", IEE Proc. - Comput. Digit. Tech., Vol. 149, pp. 152-156, No. 4, July 2002.
- [3] Cheng-Ta Hsieh and Massoud Pedram, "Architectural Energy Optimization by Bus Splitting", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 21, pp. 408-414, No. 4, Apr 2002.
- [4] Kenneth L. Calvert and Simon S. Lam, "Formal Methods for Protocol Conversion", IEEE Journal on Selected Areas in Communications, Vol. 10, pp. 127-142, No. 1, Jan 1990.
- [5] Chih-Wea Wang, Chi-Shao Lai, Chi-Feng Wu, Shih-Arn Hwang, and Ying-Hsi Lin, "On-chip Interconnection Design and SoC Integration with OCP", Proceedings of the IEEE International Symposium on VLSI Design, Automation and Test (VLSI-DAT '08), pp. 25-28, 23-25 Apr 2008.
- [6] OCP-IP, "Open core protocol Specification v2.2", Document Revision 1.1, <http://www.ocpip.org/>, 2007.
- [7] S. Watanabe, K. Seto, Y. Ishikawa, S. Komatsu and M. Fujita, "Protocol Transducer Synthesis using Divide and Conquer approach", Proceedings of the Asia and South Pacific - Design Automation Conference (ASP-DAC '07), pp. 280-285, 23-26 Jan 2007.
- [8] Philippe Martin, "Design of a virtual component neutral network-on-chip transaction layer", Proceedings of the IEEE conference on Design, Automation and Test in Europe, 2005, pp. 336 – 337, 2005.
- [9] Ruibing Lu and Cheng-Kok Koh, "A high performance bus communication architecture through bus splitting", Proceedings of the Asia and South Pacific - Design Automation Conference (ASP-DAC '04), pp. 751 – 755, 27-30 Jan 2004.
- [10] V. Dapos silva, S. Ramesh and A. Sowmya, "Bridge over troubled wrappers: automated interface synthesis", Proceedings of 17th International Conference on VLSI Design (VLSID '04), pp. 189 – 194, 2004.
- [11] ARM, "AMBA Specification v2.0", <http://www.arm.com/>, 1999.
- [12] Douglas L.Perry, "VHDL Programming by Example", Fourth Edition, Tata McGraw-Hill publishing, ISBN: 0-07-140070-2.
- [13] Kuang-Chien (KC) Chen, "Assertion-Based Verification for SoC Designs", IEEE Transactions on Electronic Engineering Times, Vol. 22, pp. 584-587, No. 4, Nov 2003.
- [14] ARM, "AMBA 3 AHB-Lite Protocol Specification v1.0", <http://www.arm.com/>, 2006.