8-12-2021

Bridge Design between AXI Lite and AHB Bus Protocol

Jaymin Patel
San Jose State University

Yash Shah
San Jose State University

Lili He
San Jose State University, lili.he@sjsu.edu

Recommended Citation

Jaymin Patel, Yash Shah, and Lili He. "Bridge Design between AXI Lite and AHB Bus Protocol" *Journal of Physics: Conference Series* (2021). https://doi.org/10.1088/1742-6596/1993/1/012008

Journal of Physics: Conference Series

PAPER • OPEN ACCESS

Bridge Design between AXI Lite and AHB Bus Protocol

To cite this article: Jaymin Patel *et al* 2021 *J. Phys.: Conf. Ser.* **1993** 012008

View the article online for updates and enhancements.

doi:10.1088/1742-6596/1993/1/012008

Bridge Design between AXI Lite and AHB Bus Protocol

Jaymin Patel, Yash Shah and Lili He

San Jose State University, Department of Electrical Engineering, San Jose, CA, USA

lili.he@sjsu.edu

Abstract. Architecture of bridge model between AXI Lite and AHB for this paper were simulated using Synopsys VCS and DC in Verilog HDL. Bridge structure mainly comprises of arbitration techniques, control signals, multiplexing techniques for writing data signals and Decoder for reading data section. In this work, bridge model between AHB and AXI lite was simulated and characterized. The proposed model of bridge design provides efficient communication between on chip bus protocols like AXI and AHB on chip in the era of deep sub-micron technology where channel side is reduced as much as 5 nm.

1. Introduction

The advancement in semiconductor technology makes implementation of SoC design more complex as it integrates lot of component on a single chip. Nowadays SoC embedded different IPs like Logic blocks, Memory blocks, CPU, Complex routing multiplexer techniques etc. As there are greater numbers of IPs present on the same chip, each IP block has its own specific property and will be operating on different frequency. Thus, establishing communication between each IPs on SoC is becoming more and more challenging owing to greater number of components. Thus, different on chip bus protocols are promising solution to establish communication between different integrated IP's which led to increase SoC's efficiency and performance. Each bus protocol exhibit different functions, has its own Specific usage, and utilize different areas, respectively. AXI is considered as a multi-channel bus used for high performances devices which cannot afford high latency such as processor, CPUs, GPU while AHB bus protocol is used with devices which requires higher throughput for example, the cache memory. Therefore, to increase overall performances of SoC, different bus protocols are implemented on same chip. As each protocol has its own specification and different features. Specific hardware circuitry is required known as Bridge to enable the transaction between different Protocols. In this paper bridges designed between AXI lite and AHB bus protocol were by use of System Verilog. System Verilog is the hardware description language. This is used for realization of hardware devices [1].

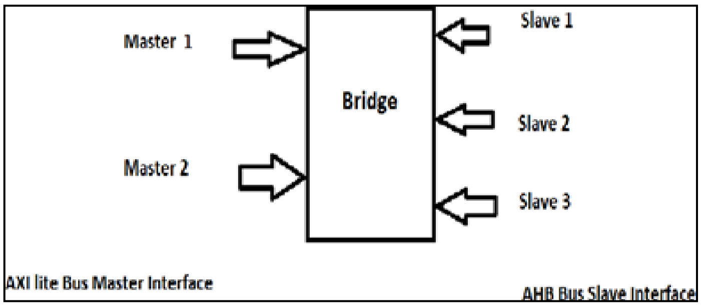


Figure 1. A basic bridge diagram.

The bridge established communication between two master devices of AXI protocol and three Slave devices using AHB Protocol. In general, Bridge can also be referred as an Interconnecting medium between two Protocols. Figure 1 shows a bias bridge diagram.

A bridge can refer as an Interconnect on SoC (System on Chip). It is the Interconnecting medium which makes transaction between two bus protocols compatible as it is not possible to directly connect the signals of both buses. It can be used to connect multiple masters and multiple slaves to increase communication efficiency between two multiple devices to get the desirable outcome of SoC [2].

2. The objective

In this project, objective of bridge design is to connect two master devices of AXI lite bus protocol with three slave devices of AHB bus protocol. In this manner bridge plays a significant role in establishing connection between multiple devices in each SoC. Out of two masters, which master will have an access of bus is decided by Arbitration mechanism. First, master will assert a request signal to access bus. Arbiter will decide which master will have an access of bus. After determination with the help of implemented arbitration techniques, arbiter asserts the grant signal to the master for accessing the bus in order to commence transaction of address and data along with control signals. In this Paper priority arbitration technique is implemented. After receiving grant signal from arbiter, handshaking mechanism is required between master’s AWVALID signal and slave’s HREADY signal. AWVALID signal indicates the master’s address and data are valid while Slave’s HREADY signal indicates slave is ready to accept the address and data of master device.

AXI comprise of five different channels supporting read, write and responses of data and address signal separately. While AHB uses a single channel to transmit data. Both the AHB and AXI support the Burst transfer techniques. In terms of area usage AXI bus occupies more area due to presence of five different channels in comparison with AHB bus. In today’s technology power is also an important constraint.

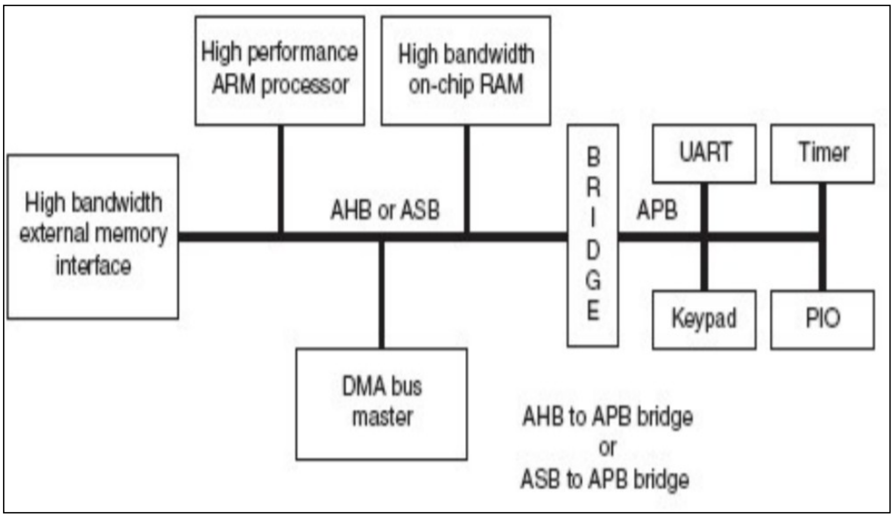

High performance ARM processor
High bandwidth on-chip RAM
High bandwidth external memory interface
AHB or ASB
B R I D G E
APB
UART
Timer
Keypad
PIO
DMA bus master
AHB to APB bridge or ASB to APB bridge

Figure 2. A typical AMBA System

Choosing the bus as in submicron technology it is necessary to maintain the low power and reduce the overall leakage which also makes AHB sometimes preferable choice. AXI bus Protocol offers higher performances due to features like out of order transaction, multiple outstanding and interleaving [3].

3. State transitions and output results

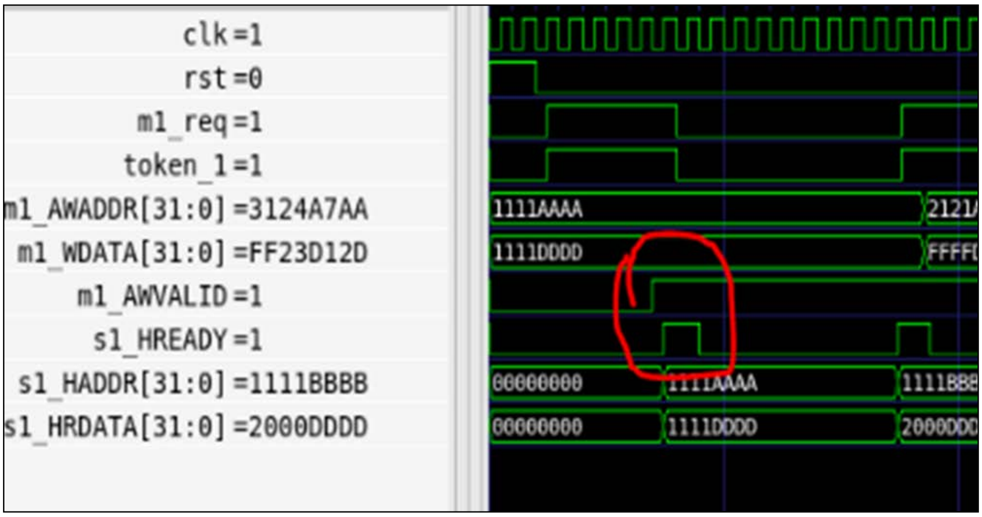


Figure 3. Display of the handshaking mechanism

After the successful handshaking address and data, signal will transfer into slave devices. Figure 3 displays the handshaking mechanism in output.

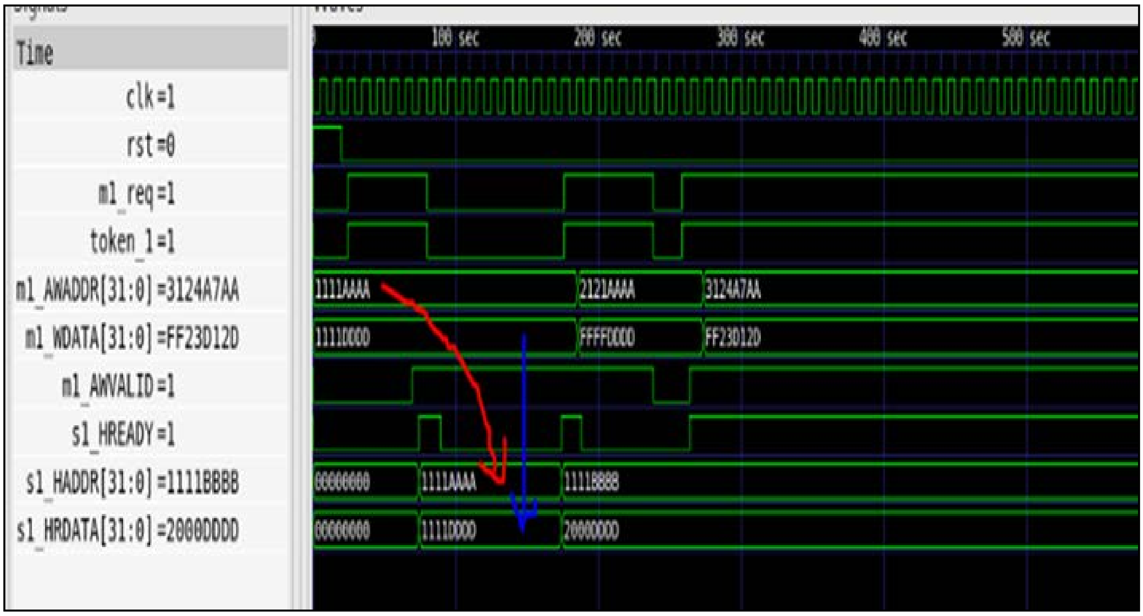


Figure 4. Transfer data to the slave device

In figure 4 the red arrow indicates on slave_1's address where master'1 data is written by blue. On successful transmission of address and data, slave sends response signal to indicate whether slave device was able to successfully receive master's data or failed in transaction. To receive response signal from slave master device should have computability to accept the response signal generated by slave to acknowledge master.

Central decoder is another important component in this structure. It is a combinational circuit which provides n inputs to 2^n outputs. In this paper decoder is mainly used to decode address range. Decoder sends data, address, and control signals to appropriate slave device for which master is transferring data or from which master is accepting data. Along with central decoder there is Multiplexers in architecture for transmitting ready signal [4].

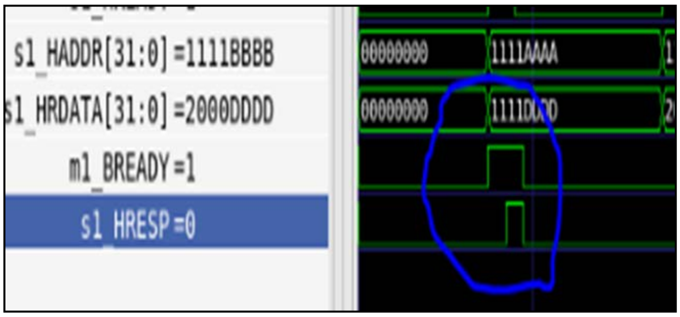
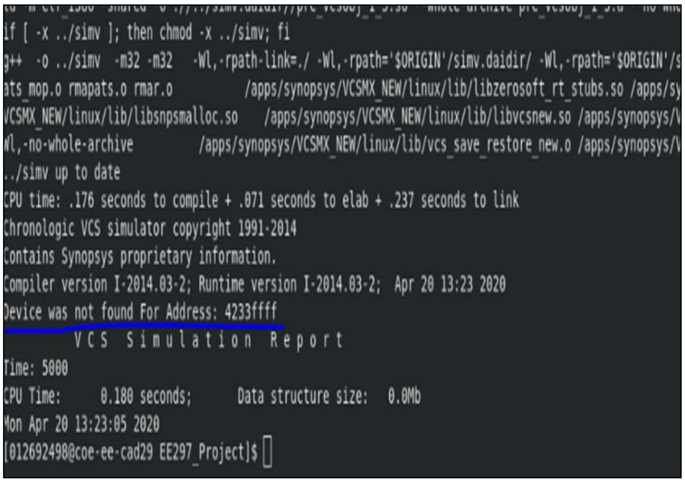


Figure 5. Response of the signal from slave

Decoder is used to discover which slave master wants to transmit data. It firstly decodes the address available from master and then decodes it to know the corresponding slave device to that address. In this project we have divided the master's address in corresponding manner. Slave 1 can only accept data whose addresses are in the range of 32'h1000_0000 to 32'h1FFF_FFFF. Similarly, slave 2 can receive data within range of 32'h2000_0000 to 32'h2FFF_FFFF while slave 3 can receive data in 32'h3000_0000 to 32'h3FFF_FFFF range. Any other address except above address gives an error messages "Slave devices does not exist". Figure 5 shows the response of the signal from slave.



```
if [ -x ../simv ]; then chmod -x ../simv; fi
g++  -o ../simv  -m32 -m32   -Wl,-rpath-link=./ -Wl,-rpath='$ORIGIN'/simv.daidir/ -Wl,-rpath='$ORIGIN'/s
ats_mop.o rmapats.o rmar.o        /apps/synopsys/VCSMX_NEW/linux/lib/libzerosoft_rt_stubs.so /apps/sy
VCSMX_NEW/linux/lib/libsnpsmalloc.so   /apps/synopsys/VCSMX_NEW/linux/lib/libvcsnew.so /apps/synopsys/V
Wl,-no-whole-archive        /apps/synopsys/VCSMX_NEW/linux/lib/vcs_save_restore_new.o /apps/synopsys/V
../simv up to date
CPU time: .176 seconds to compile + .071 seconds to elab + .237 seconds to link
Chronologic VCS simulator copyright 1991-2014
Contains Synopsys proprietary information.
Compiler version I-2014.03-2; Runtime version I-2014.03-2;  Apr 20 13:23 2020
Device was not found For Address: 4233ffff
           V C S   S i m u l a t i o n   R e p o r t
Time: 5000
CPU Time:      0.180 seconds;       Data structure size:   0.0Mb
Mon Apr 20 13:23:05 2020
[012692498@coe-ee-cad29 EE297_Project]$
```

Figure 6. Error showing wrong address entered

After completion of data transmission from master 1 device to slave devices. Master 2 devices can start transmitting data based on the same approach used for master 1 device for selection of slave will get Priority. After completion of data transmission from master 1 device to slave devices. Master 2 devices can start transmitting data based on the same approach used for a master 1 device for selection of slave. Moreover, as priority arbitration is implemented Master 1 has been allocated higher priority. Thus, if Master 1 wants to transmit any device will get priority. After completion of data transmission from master 1 device to slave devices. Master 2 devices can start transmitting data based on the same approach used for master 1 device for selection of slave.

Here the final output waveforms depict all the associated signals which are used in this project and along with proper clock signals. As the whole project is synchronously working with clock, change in each signal will take place in accordance with changes in clock signal. Many times, it happens that slave is not ready to accept the data sent by the master [5]. In this kind of scenarios, Address cannot be extended but data can be extended to complete the transaction. Thus, for extension of data cycle so that slave can accept the transaction within time, HREADY signal is used. When HREADY Signal goes low, it introduced wait states and allows extra time for the slave to provide or sample data. Figure 6 shows the error which showing the wrong address was entered.

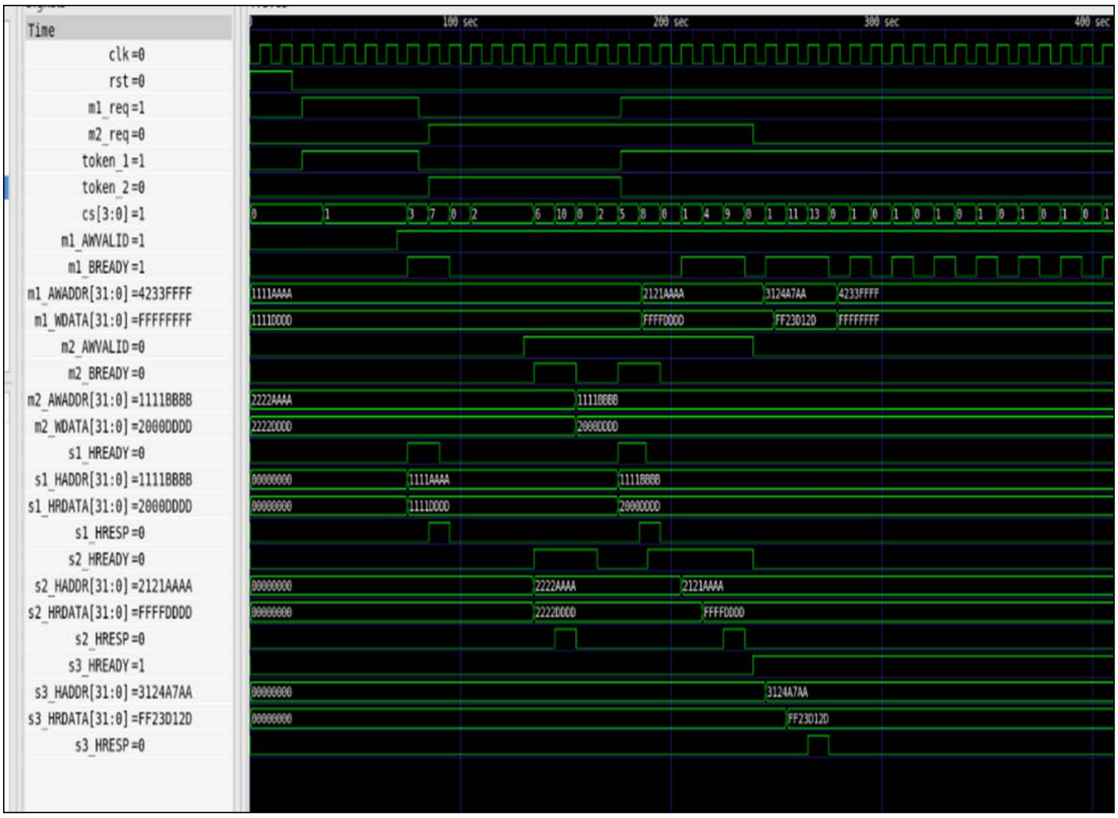


Figure 7. Final output waveform

In this project, this operation can be observed when HREADY from slave1 and slave 2 goes low. Another Important signal is HRESP signal which is a bit signal provided by slave devices. HRESP signal shows the status of the slave whether it is ready to accept data, there is Error or Whether SPLIT or RETRY Operation is needed [6]. Moreover, in Verilog code it is important to create Interfaces so that both the master and slave device can communicate with Bridge. To achieve that two interfaces have been created in Verilog code so that bridge can communicate with master and slave. In this working model we used mod port to create multiple instances of master and slave.

After completion of data transmission from master 1 device to slave devices. Master 2 devices can start transmitting data based on the same approach used for master 1 device for selection of slave will get Priority. After completion of data transmission from master 1 device to slave devices. Master 2 devices can start transmitting data based on the same approach used for a master 1 device for selection of slave. Moreover, as priority arbitration is implemented Master 1 has been allocated higher priority. Thus, if Master 1 wants to transmit any device will get priority. After completion of data transmission from master 1 device to slave devices. Master 2 devices can start transmitting data based on the same approach used for master 1 device for selection of slave.

The final output wave forms depict all the associated signals which are used in this project and along with proper clock signals. As shown in figure 7, the whole project is synchronously working with clock, change in each signal will take place in accordance with changes in clock signal. Many times, it happens that slave is not ready to accept the data sent by the master. In this kind of scenarios, Address cannot be extended but data can be extended to complete the transaction. Thus, for extension of data cycle so that slave can accept the transaction within time, HREADY signal is used. When HREADY signal goes low, it introduced wait states and allows extra time for the slave to provide or sample data. In our Project, this operation can be observed when HREADY from slave1 and slave 2 goes low. Another Important signal is HRESP signal which is a bit signal provided by slave devices. HRESP signal shows the status of the slave whether it is ready to accept data, there is Error or Whether SPLIT or RETRY Operation is needed. Moreover, in Verilog code it is important to create Interfaces so that both the master and slave device can communicate with Bridge. To achieve that two interfaces have been created in Verilog code so that bridge can communicate with master and slave. In this working model we used mod port to create multiple instances of master and slave.

4. Summary and conclusion

Bridge model between AHB and AXI lite was simulated and characterized. The proposed model of bridge design provides efficient communication between on chip bus protocols like AXI and AHB on chip in the era of deep sub-micron technology where channel side is reduced as much as 5 nm. Thus, because of bridge, communication between different protocols can be possible which helps to increase chip's efficiency and frequency. Hence Bridge would have high impact in VLSI and SoC design.

5. References

[1] Kavaldjeiv and G.M. Smith, "A survey of efficient on chip Communications for SoC" in University of Twente, Netherlands.

[2] S. Guruprasad, K. Sudharshan, "Design and Analysis of master module for AMBA AXI-4"in (NCECS-2011) at Siliguri Institute for technology, Sept 2011.

[3] S. Pasricha and N. Dutt, "On-Chip Communication Architecture Standards'', in section of "System on Chip Interconnect", USA, Chapter 3, Sec. 3.2, pp. 88-96.

[4] L. Deeksha, B.R. Shivakumar, "Effective Design and Implementation of AMBA AHB bus Protocol Using Verilog", International Conference on Intelligence Sustainable System (ICISS), 2019.

[5] M. Milti and M. Stocjev, "An Overview of on chip bus", a research paper published in Jan. 2006

[6] A. Shrivastava, S K Sharma, "Various Arbitration Algorithm for on-chip (AMBA) Shared bus Multiprocessor SoC", IEEE Student's Conference on Electrical, Electronics and Computer Science (SCEECS), Bhopal, India, March 16