VLSI DESIGN OF AMBA BASED AHB2APB BRIDGE

Aparna Kharade[1] and V. Jayashree[2]

[1]Research Scholar, Electronics Dept., D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji, Maharashtra, India.
[2]Professor, Electronics Dept ., D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji, Maharashtra, India.

ABSTRACT

The Advanced Microcontroller Bus Architecture (AMBA) is an open System-on-Chip bus protocol for high-performance buses to communicate with low-power devices. In the AMBA Advanced High Performance bus (AHB) a system bus is used to connect a processor, a DSP, and high-performance memory controllers where as the AMBA Advanced Peripheral Bus (APB) is used to connect (Universal Asynchronous Receiver Transmitter) UART. It also contains a Bridge, which connects the AHB and APB buses. Bridges are standard bus-to-bus interfaces that allow IPs connected to different buses to communicate with each other in a standardized way. So AHB2APB bridge is designed, implemented using VERILOG tool and tested using Verilog testbench and is reported in this paper. A synthesizable RTL code of a complex interface bridge between AHB and APB is developed and known as AHB2APB Bridge. The simulated AHB2APB Bridge results are promising and can be further tested for its verstality by writing a verification program using UVM in future.

KEYWORDS

AMBA; AHB2APB; SOC; VERILOG; XILINX;

1. INTRODUCTION

The AMBA is used as interconnection standard for system on chip (SOC) design for semiconductor industries. AMBA interconnection standard support for high speed, bandwidth and pipelined data transfer by using rich sets of bus signals. Number of ARM Partners and IP provider adopted this standard because of their successful implementation in Application Specification Integrated Circuit (ASIC) design. In some cases, if AHB side peripherals wants to communicate with APB side peripherals, in between bridge is required for proper communication. Jaehoon Song et al. have reported an efficient testable design technique for an SoC with an on/off-chip bus bridge for the on-chip advanced high-performance bus and off-chip peripheral-component interconnect bus [1]. The author Krishna Sekar designed and reported FLEXBUS, a new architecture that can efficiently adapt the logical connectivity of the communication architecture and the components connected to it [2]. The bus bridge to interface Open Core Protocol (OCP) to AHB protocols which play a vital role in SoC application to avoid application failure in case of malfunctioning. Initially these basic protocols are modeled separately using VHDL and are simulated [3]. Their simulation results show that the communication between AXI 4.0 and APB4.0 through the bridge is appropriate. All of the commands and data are successfully transferred from AXI 4.0 to APB4.0 protocol by the bridge and reduced loss of data or control information was found by Kalluri Usha and T. Ashok Kumar [4]. The Advanced Microcontroller Bus Architecture (AMBA) is an on-chip bus architecture used to strengthen the reusability of IP core and widely used interconnection standard for system on chip (SOC) [5]. Taking literature review into consideration and to ensure less data loss between AHB to APB or APB to AHB data transfers along with reduced in power consumption, in this paper AMBA based AHB2APB Bridge which is required to bridge the communication gap between low bandwidth peripherals on APB with the high bandwidth ARM Processors and/or other high-speed devices on AHB is reported.

Organization of this Implementation of AHB2APB Bridge is as follows. Section II describes the theoretical background on implemented AHB2APB Bridge where as section III describes the methodology adopted with Operation of handshaking signal. Results and observations are reported in section IV. Conclusion and future scope is explained in section V.

2. THEORETICAL BACKGROUND OF AHB2APB BRIDGE

The bridge is designed either by using asynchronous FIFO or by using handshaking signals. Here in this project we used handshaking signals for reducing data loss. In AMBA different bus bridge, protocols such as, ASB2APB, AHB2APB and AXI to APB can be used. The AHB to APB bridge is an AHB slave, providing an interface between the high-speed AHB and the low-power APB. Read and write transfers on the AHB are converted into equivalent transfers on the APB. As the APB is not pipelined, wait states are added during transfers to and from the APB when the AHB is required to wait for the APB. It is required to bridge the communication gap between low bandwidth peripherals on APB with the high bandwidth ARM Processors and/or other high-speed devices on AHB. This ensures that there is no data loss between AHB to APB or APB to AHB data transfers. AHB2APB interfaces AHB and APB. It buffers address, controls and data from the AHB, drives the APB peripherals and return data along with response signal to the AHB. The AHB2APB interface is designed to operate when AHB and APB clocks have ~~the~~ any combination of frequency and phase. TheAHB2APB performs transfer of data from AHB to APB for write cycle and APB to AHB for Read cycle. Interface between AMBA high performance bus (AHB) and AMBA peripheral bus (APB). It provides latching of address, controls and data signals for APB peripherals.

3. METHODOLOGY OF IMPLEMENTATION

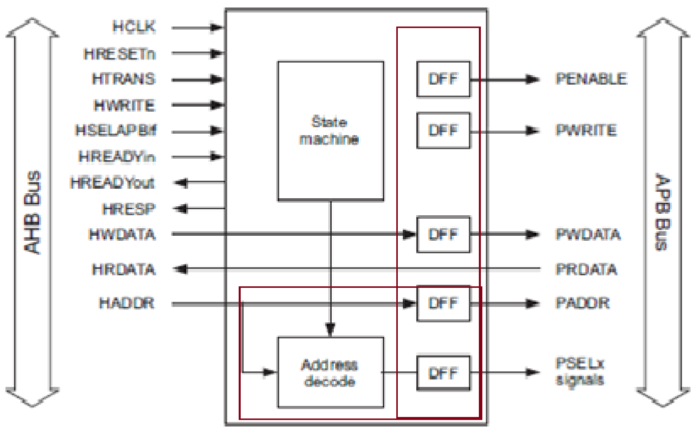


Figure 1.Architecture of AHB2APB Bridge

3.1 ARCHITECTURE OF AHB2APB BRIDGE

Architecture of AHB2APB is as shown in Fig 1. The AHB is a pipelined bus, designed for high performance and high bandwidth operations. It can support up to 16 bus masters and slaves. The address bus can be up to 32 bits wide and the data bus can be up to 128 bit wide. Architecture of AHB2APB bridge shown in Figure 1 includes three modules viz.; bridge top, AHB slave interface and APB controller which are explained further.

BRIDGE TOP: It includes two basic modules first is AHB slave interface and second is APB controller. Instantiation of these modules into main bridge top and connection is made through signal flow which is mentioned in the port list of sub module of both AHB interface and APB controller.

AHB SLAVE INTERFACE: This module includes the logic required to implement TSELx that is used to select peripheral memory map, valid and pipelined address, data and control channel as well as all the input and output signals which defines the module for further instantiation and connection of this module.

APB CONTROLLER: It is the heart of the design this module is useful for taking the decision which is required for proper operation of design according to present state, next state logic and output logic the coding is performed.

3.2 OPERATION OF HANDSHAKING SIGNAL

The generation of all APB output signals is based on the status of the transfer state machine and A standard AHB slave interface consists three outputs are explained further

THE APB OUTPUT SIGNAL GENERATION

The generation of all APB output signals is based on the status of the transfer state machine:

PENABLE is only set HIGH during one of three enable states, in the last cycle of an APB transfer. A register is used to generate this output from the next state of the transfer state machine.

PSELx outputs are decoded from the current transfer address. They are only valid during the read, write and enable states, and are all driven LOW at all other times so that no peripherals are selected when no transfers are being performed.

PADDR is a registered version of the currently selected address input (*HADDR* or the address register) and only changes when the read and write states are entered at the start of the APB transfer.

PWRITE is set HIGH during a write transfer, and only changes when a new APB transfer is started. A register is used to generate this output from the next state of the transfer state machine.

AHB OUTPUT SIGNAL GENERATION

A standard AHB slave interface consists of the following three outputs HRDATA, PRDATA and HREADY Yout.

HRDATA is directly driven with the current value of *PRDATA*. APB slaves only drive read data during the enable phase of the APB transfer, with *PRDATA* set LOW at all other times, so bus clash is avoided on *HRDATA* (assuming OR bus connections for both the AHB and APB read data buses).

HREADYout is driven with a registered signal to improve the output timing. Wait states are inserted by the APB bridge during the *ST_READ* and *ST_WRITEP* states, and during the *ST_WENABLEP* state when the next transfer to be performed is a read. *HRESP* is continuously held LOW, as the APB bridge does not generate SPLIT, RETRY or ERROR responses.

The architecture of AHB2APB Bridge shown in Figure.1. is implemented in Verilog and is tested for functionality simulation is carried out in UVM Environment

3.3 VERILOG CODING STEPS FOR IMPLEMENTATION OF AHB2APB BRIDGE

3.3.1 STEPS TO IMPLEMENT BRIDGE TOP

a. Define the AHB2APB bridge top consisting of 9 input signals are *HTRANS, HADDR, HWDATA, HREADYin, HWRITE, HCLK, HRESETn, PRDATA, HSIZE, HRESP* and 8 output signals are *PENABLE, HREADYout, PWRITE, PWDATA, PADDR, HRADATA, PSELX, HRESP*.
b. Instantiate AHB interface module in top using order-based instantiation.
c. Instantiate FSM controller module in top based on order-based instantiation. The Syntax for module instantiation is; module name user defined name (portlist).

3.3.2 STEPS TO IMPLEMENT AHB INTERFACE

a. Define the AHB interface module consisting of 8 input signals from that reference signal HCLK, reset signal HRESET, handshaking signal HWRITE, HREADYin, address signal HADDR, Data signal HWDATA, internal Handshaking signals HSIZE, HTRANS and and 8 output signals are handshaking signals VALID, HWRITEreg, TSELx, transmission address signals TADDR1, TADDR2, transmission data signals TPWDATA1, TPWDATA2, PSIZEreg.
b. Implement the logic for *TSELx*, valid and pipelining of signals by using different always blocks includes conditional statements.
c. End the AHB interface module by using verilog command end module. The Syntax for module instantiation: module name user defined name (portlist)

3.3.3 STEPS TO IMPLEMENT APB CONTROLLER

a. Define the Top APB module consisting of 13 input signals viz.; reference signal *HCLK*, reset signal *HRESETn*, handshaking signal *VALID*, *TSELx*, Transmition addresses signals,,*TPADDR1*, *TPADDR2*, Transmission data signals, *TPWDATA1*, *TPWDATA2*, *HWRITEreg*, *HRDATA*, *PRDATA*, *HWRITE*, *HSIZE*, and 7 output signals viz.; control signals *PENABLE*, *PSELx*,slave address signal *PADDR*, Transmission data signal *PWDATA*, *HRDATA*, *PWRITE*, Response signal *HRESP*
b. Define FSM states as a parameter by using inbuilt verilog function *parameter*.
c. Write the present, next state, output logic according state diagram by using three different procedural blocks.

3.4 FLOWCHART AND LOGIC IMPLEMENTATION

Bridge top: Includes two basic modules first one is AHB slave interface and second one is APB controller shown in Figure 2. verification of AHB interface is performed by using FSM controller states. Lastly bridge is verified by using test bench which is written in Verilog environment.

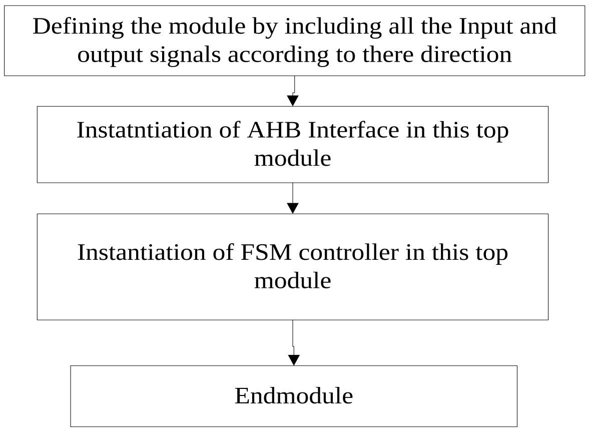

Defining the module by including all the Input and output signals according to there direction
Instatntiation of AHB Interface in this top module
Instantiation of FSM controller in this top module
Endmodule

Fig2.Flowchart of Bridge Top

3.4.1*AHB SLAVE INTERFACE:*

In this, three decisions with respect to

i) Peripheral Memory map and Tselx signal
ii) Valid
iii) Pipelining of signals are taken as these are required to operate bridge correctly and those are presented further. Flowchart of implementation is shown in figure 3.

i. **PERIPHERAL MEMORY MAP AND TSELX SIGNAL***:* According to Tselx signal controller operates bridge in particular area where user wants to workout. This signal is given to the APB controller and status of Telx w.r.to range of address is explained in Table 1. In accordance with Haddr and peripheral memory map is shown in Table 2. There are four regions in the peripheral memory map namely Interrupt controller, counter and timers, remap and pause and undefined.

Table 1. Status of Tselx [2:0] signal

Range of Haddr	**Status of Tselx [2:0]**
32'h800000000 to32'h84000000	001
32'h840000000 to32'h88000000	010
32'h880000000 to32'h8C000000	100
>32'h8C000000	000

Table 2. Address range of peripheral memory map

Range of Address	**Peripheral Memory Map**
0x80000000 to 0x84000000	Interrupt controller
0x84000000 to 0x88000000	Counter and timers
0x88000000 to 0x8C000000	Remap and pause
0x8C000000 to 0xBFFFFFFF	Undefined

ii. **VALID SIGNAL**: This signal is useful for indicating that valid data is coming from master to slave and it is high during non-sequential and sequential operation and for Idle and busy signal mentioned in table III. This valid is goes low that means no valid transfer is remaining. It is implemented using the procedural always block and implemented using if else statement and using conditional operator by using that the address range according to peripheral memory map is explained in Table 3.

Table 3. Status of VALID signal

Condition of signal	Status of valid
Hresetn =low	Valid = low
Haddr=between32'h800000000 to32'h8C000000 && Hready=High, && && Htrans=2'b1OR2'b11)	Valid = high
Haddr>32'h8C000000 OR Hready=low OR OR(Htrans!=2'b10OR2'b11)	Valid=low

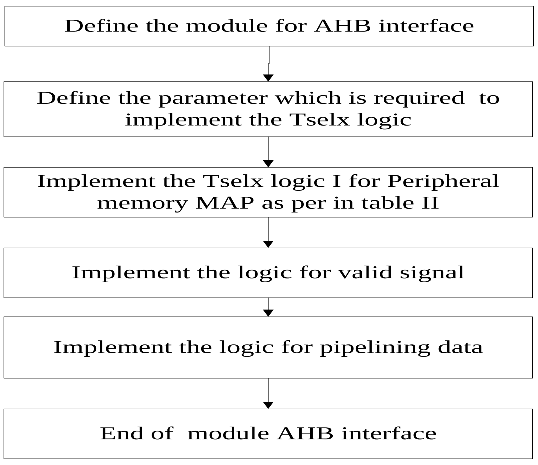

Define the module for AHB interface
Define the parameter which is required to implement the Tselx logic
Implement the Tselx logic I for Peripheral memory MAP as per in table II
Implement the logic for valid signal
Implement the logic for pipelining data
End of module AHB interface

Figure 3.Flowchart of AHB Slave Interface

i. **PIPELINING OF SIGNALS**: By using Procedural always block and non-blocking assignment (<=) the signal data channel, address channel as well as control signals i.e. H write are delayed by two clock cycles and it is given to the APB Interface.

3.4.2 APB CONTROLLER:

As per the protocol specification of AMBA, the state diagram is as shown in Figure 4. and state table of APB controller shown in Table 4 are used. This logic implemented in APB controller is as shown in Fig.5. The states are defined as per the control signal coming to the input. There are three main logics for implementation of the APB controller according to Present state logic, Next state logic and Output logic.

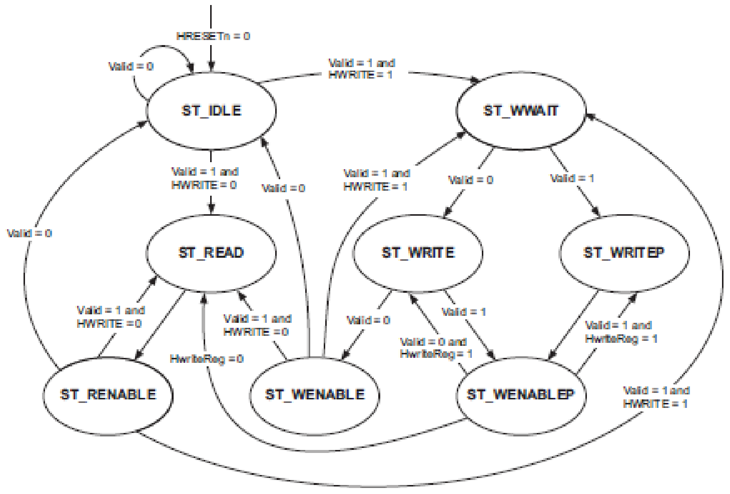


Figure 4. State diagram of APB controller

The state diagram of APB controller is explained in AHB2APB bridge protocol specification which is provided by ARM.

3.4.2.1 ST_IDLE

During this state the APB buses and PWRITE are driven with the last values they had, and PSEL and PENABLE lines are driven LOW.

When Present state, ST_IDLE

i. If valid is asserted and Hwrite is asserted, then next state is ST_WWAIT.
ii. If valid is asserted and Hwrite is desserted then next state is ST_READ.
iii. If valid is deasserted then next state is ST_IDLE.

3.4.2.2 ST_WWAIT

This state is needed due to the pipelined structure of AHB transfers, to allow the AHB side of the write transfer to complete so that the write data becomes available on HWDATA. The APB write transfer is then started in the next clock cycle.

When Present state ST_WWAIT

i. If valid is asserted and Hwrite is asserted, then next state is ST_WRITEP.
ii. If valid is de-asserted, then next state is ST_WRITE.

3.4.2.3 ST_WRITEP

During this state the address is decoded and driven onto PADDR, the relevant PSEL line is driven HIGH, and PWRITE is driven HIGH. A wait state is always inserted, as there must only ever be one pending transfer between the currently performed APB transfer and the currently driven AHB transfer.

When Present state ST_WRITEP:

i. Unconditional jump to ST_WENABLEP

3.4.2.4 ST_WENABLEP

A wait state is inserted if the pending transfer is a read because, when a read follows a write, an extra wait state must be inserted to allow the write transfer to complete on the APB before the read is started.

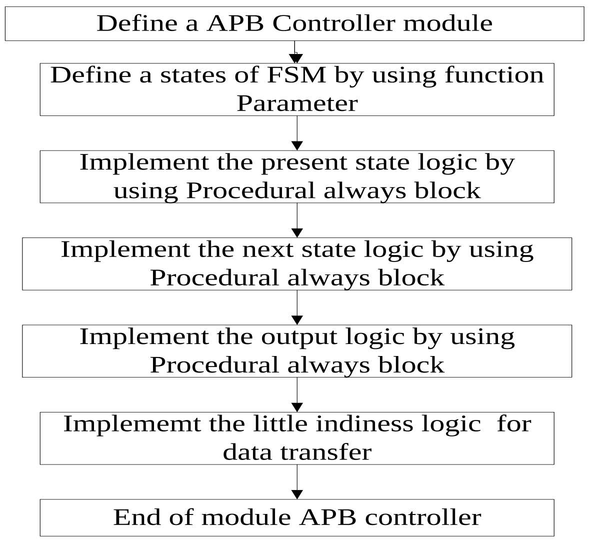

Define a APB Controller module
Define a states of FSM by using function Parameter
Implement the present state logic by using Procedural always block
Implement the next state logic by using Procedural always block
Implement the output logic by using Procedural always block
Implememt the little indiness logic for data transfer
End of module APB controller

Figure 5.Flowchart of APB Controller

when Present state ST_WENABLEP:

i. If valid is de-asserted and Hwritereg is asserted, then next state is ST_WRITE.
ii. If Hwritereg is de-asserted, then next state is ST_READ.

3.4.2.5. ST_WENABLE

During this state the **PENABLE** output is driven HIGH, enabling the current APB transfer. All other APB outputs remain the same as the previous cycle.

When Present state ST_WENABLE:

i. If valid is de-asserted, then next state is ST_IDLE.
ii. If valid is asserted and Hwrite is de-asserted, then next state is ST_READ.
iii. If valid is asserted and Hwrite is de-asserted, then next state is ST_WWAIT.

3.4.2.6. ST_RENABLE

During this state the PENABLE output is driven HIGH, enabling the current APB transfer. All other APB outputs remain the same as the previous cycle.

When Present state ST_RENABLE:

i. If valid is de-asserted, then next state is ST_IDLE.
ii. If valid is asserted and Hwrite is de-asserted, then next state is ST_READ.
iii. If valid is asserted and Hwrite is de-asserted, then next state is ST_WWAIT.

3.4.2.7. ST_READ: During this state the address is decoded and driven onto PADDR, the relevant PSEL line is driven HIGH, and PWRITE is driven LOW. A wait state is always inserted to ensure that the data phase of the current AHB transfer does not complete until the APB read data has been driven onto HRDATA

When Present state ST_READ:

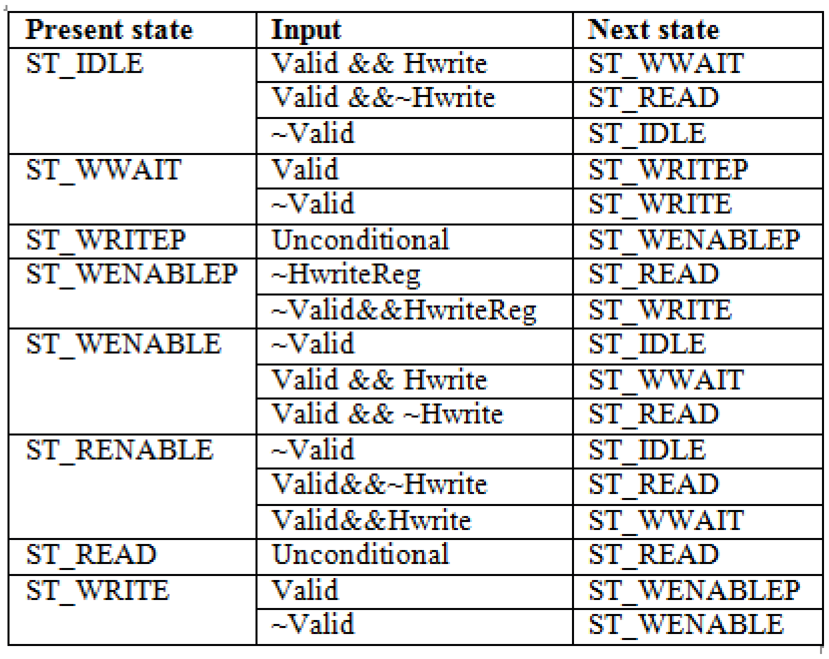
i. Unconditional jump then next state is ST_READ

3.4.2.8. ST_WRITE: During this state the address is decoded and driven onto PADDR, the relevant PSEL line is driven HIGH, and PWRITE is driven HIGH.

When Present state ST_WRITE:

i. If valid is de-asserted then next state is ST_WENABLE.
ii. If valid is asserted and then next state is ST_WENABLEP

Table 4. State Table of APB Controller



Present state	Input	Next state
ST_IDLE	Valid && Hwrite	ST_WWAIT
	Valid &&~Hwrite	ST_READ
	~Valid	ST_IDLE
ST_WWAIT	Valid	ST_WRITEP
	~Valid	ST_WRITE
ST_WRITEP	Unconditional	ST_WENABLEP
ST_WENABLEP	~HwriteReg	ST_READ
	~Valid&&HwriteReg	ST_WRITE
ST_WENABLE	~Valid	ST_IDLE
	Valid && Hwrite	ST_WWAIT
	Valid && ~Hwrite	ST_READ
ST_RENABLE	~Valid	ST_IDLE
	Valid&&~Hwrite	ST_READ
	Valid&&Hwrite	ST_WWAIT
ST_READ	Unconditional	ST_READ
ST_WRITE	Valid	ST_WENABLEP
	~Valid	ST_WENABLE

4. RESULTS AND OBSERVATIONS

There are 11 signals coming from AHB2APB bridge and are given to the AHB Master and 6 signals are coming from the AHB2APB bridge which are given to the APB Interface. These signals are as shown in RTL schematic results of Figure 6.

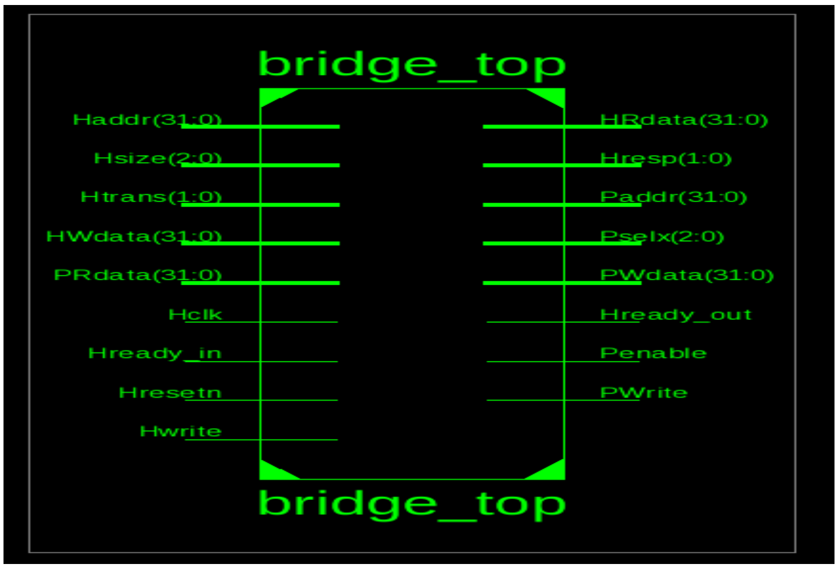


Figure 6. RTL Schematic of AHB2APB Bridge

The enhanced RTL schematic of Fig. 7 shows the AHB slave interface and APB controller inside the AHB2APB Bridge top which is as per the Standard Design.

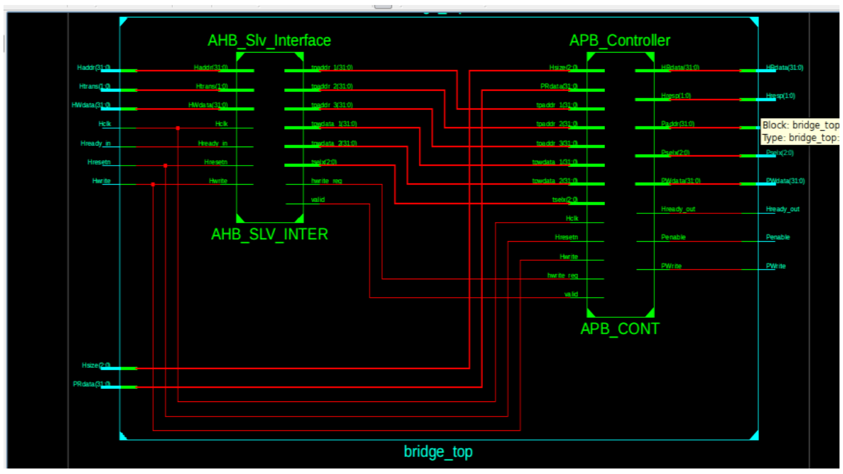


Figure 7. Enhanced RTL Schematic of AHB2APB Bridge

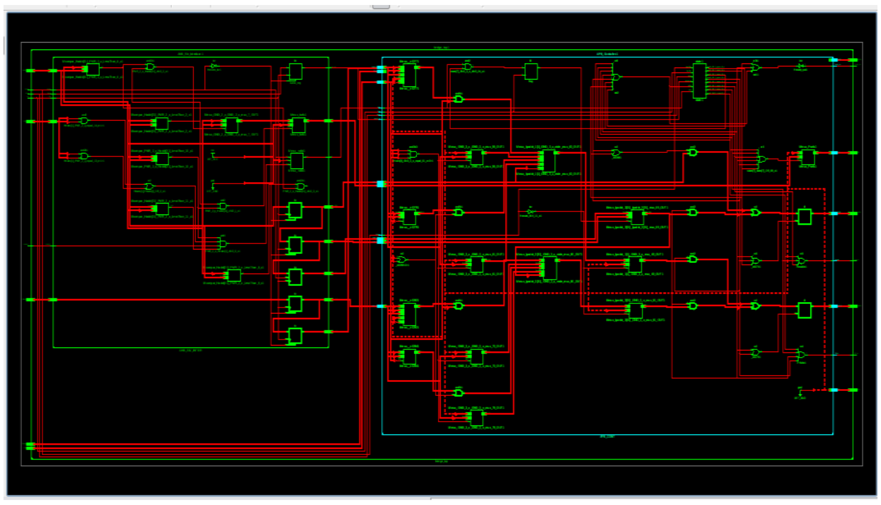


Figure 8.Enhanced RTL Schematic of AHB2APB Bridge

Gate level schematic of AHB2APB bridge showing detailed components view of AHB slave interface and APB controller has been shown in Figure.8. It consists of flipflops, basic gates and decoders etc.

For implementation of AHB2APB bridge, project settings of Family-Spartan 6, device-XC6SLX4, package-TQG144, speed grade -3 etc. are used.

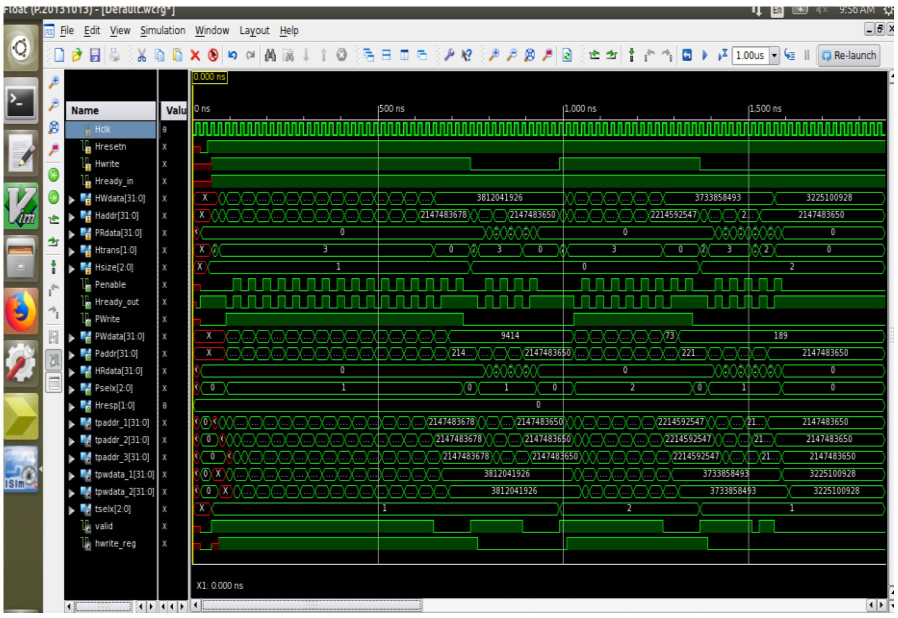


Figure 9.Verification Result of AHB2APB Bridge using Verilog TB

By driving the different Hburst and Hsize the obtained result has shown the Hwdata of AHB Master is arriving to the Pwdata of APB slave and Prdata of APB Slave is arriving to Hrdata of AHB Master so no data loss is as shown in figure 9. Pclk is not used by the APB peripherals only when they are communicating through AHB-APB bridge because Penable is used as a strobe signal by the APB peripherals hence they are low power peripheral device.

5. CONCLUSION

We have implemented AHB2APB Bridge successfully by using Verilog tool and simulated the implemented design in Xilinx ISE environment. Testing of this bridge was done by using Universal verification Methodology Environment using an industrial standard Aldec Riviera-Pro simulator. Here we are using master clock, Hclk alone with Penable which will help for reducing the power consumption. Implementation of AHB2APB Bridge for multi master and multi slave is one of the future scope.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude towards Prof.Dr. Mrs.V.Jayashree Professor in Electronics Engineering of DKTE Society's Textile & Engineering Institute, Ichalkaranji (An Autonomous Institute) , who has been my supervisor. She has been my philosopher, mentor and guide. She provided me with many helpful suggestions, important advice and constant encouragement in this work.

REFERENCES

[1] Jaehoon Song, Student member, IEEE et al.," An Efficient SOC Test Technique by Reusing On/Off-Chip Bus Bridge". IEEE Transactions on Circuits and Systems-I: Regular Papers, Vol. 56, No.3, March 2009.

[2] Krishna Sekar, Member, IEEE et al." Dynamically Configurable Bus Topologies for High-Performance On-Chip Communication". IEEE transactions on very large-scale integration (Vlsi) systems, vol. 16, no. 10, October 2008.

[3] G. Geetha Reddy et al.," Implementation of bus bridge between AHB and OCP". Global Journal of Advanced Engineering Technologies, Vol 1, Issue4-2012 ISSN: 2277-6370.

[4] Kalluri Usha and T. Ashok Kumar," AXI To Apb Interface Design Using Verilog". IOSR Journal of Electronics and Communication Engineering, p- ISSN: 2278-8735.Volume 8, Issue 5 (Nov. - Dec. 2013), pg. 01-09.

[5] Vani.R.M and Merope," Design of AMBA based AHB2APB bridge" IJCSNS international journal of computer science and network security, vol.10, no.11, November 2010

[6] Priyanka M Shettar1, Ashwin Kumar "Verification IP of AMBA AXI V1.0 Using UVM". IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) Volume 6, Issue 3, Ver. II (May -Jun. 2016), pg. 54-58.

AUTHORS

Aparna Ramesh Kharade is currently pursuing Mtech in electronics from Shivaji University Kolhapur.completing the project in DKTE Society's Textile &Enginnering Institute She obtained her bachelor's degree in Electronics from Shivaji University, Kolhapur. She obtained her diploma in Medical Electronics from Government Polytechnic Miraj. Her research interests are VLSI based SOC design and verification.