ISSN 2320-9984 04>
9 772320 998015

Volume 1, Issue 4, December 2013 ISSN: 2320-9984 (Online)

International Journal of Modern Engineering & Management Research

Website: www.ijmemr.org

Design of AHB2APB Bridge for Efficient Utilization of AHB and APB

Geeta Pal
M. Tech Scholar
Department of Electronics & Communication Engg.
Shri Ram Institute of Technology,
Jabalpur (M.P.) [India]
Email :geeta.pal21@yahoo.com

Ravi Mohan
Assistant Professor
Department of Electronics & Communication Engg.
Shri Ram Institute of Technology,
Jabalpur (M.P.) [India]
Email : ravimohan7677@yahoo.co.in

Abstract—*Microprocessor performance has improved rapidly these years. In contrast memory latencies and bandwidths have improved little. The result is that the memory access time is the bottleneck which limits the system performance. In case of larger system design which requires more number of I/O ports and more memory capacity the system designer may interface external I/O ports and memory with the system. We are using advanced microcontroller bus architecture with its advanced high performance bus. The Advanced Microcontroller Bus Architecture (AMBA) is a widely used interconnection standard for System on Chip (SoC) design. In order to support high-speed pipelined data transfers, AMBA supports a rich set of bus signals, making the analysis of AMBA-based embedded systems a challenging proposition. The goal of this is to synthesize and simulate complex interface bridge between Advanced High performance Bus (AHB) and Advanced Peripheral Bus (APB) known as AHB2APB Bridge. This also involves the Back notation for Synthesized of Bridge module and to perform Functional and Timing Simulation using Xilinx ISE.*

In this we are using AHB to APB interfacing via a bridge and using parallel operations which we are divided the large program or modules into a smaller one. Here used parallel pipeline to fast operation. In the thesis we are proposing to design a very high performance AHB to APB and vice versa bridge and to achieve our proposed work we have used pipelining for data transfer.

1. INTRODUCTION

The Advanced Microcontroller Bus Architecture (AMBA) is a widely used interconnection standard for System on Chip (SoC) design[1]. In order to support high-speed pipelined data transfers AMBA supports a rich set of bus signals, making the analysis of AMBA-based embedded systems a challenging proposition. The AMBA specification[2] has become a de-facto standard for the semiconductor industry, it has been adopted by more than 95% of ARM's partners and a number of IP providers. The specification has been successfully implemented in several ASIC designs. Since the AMBA interface is processor and technology independent, it enhances the reusability of peripheral and system components across a wide range of applications. The AMBA specification has been derived to satisfy the following four key requirements.

(i) To facilitate the right-first-time development of Embedded Microcontroller Products with one or more CPUs or signal processors.

(ii) To be 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.

(iii) To encourage modular system design to improve processor independence, providing a development road-map for advanced cached CPU cores and the development of peripheral libraries.

(iv) To minimize the silicon infrastructure required supporting efficient on-chip and off-chip communication for both operation and manufacturing test. This paper is to design the AMBA based AHB2APB Bridge which interfaces AHB and APB buses. 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 is to ensure that there is no data loss between AHB to APB or APB to AHB data transfers. In our work we intend to use Verilog HDL (Hardware Description Language) for designing the RTL (Register Transfer Level) code[3]. Synthesis and Simulation is done using Xilinx[4].

2. TYPICAL AMBA BASED MICROCONTROLLER

An AMBA-based microcontroller typically consists of a high-performance system backbone bus (AMBA AHB or AMBA ASB), able to sustain the external memory bandwidth on which the CPU, on-chip memory and other Direct Memory Access (DMA) devices reside. This bus provides a high-bandwidth interface between the elements that are involved in the majority of transfers. Also located on the high performance bus is a bridge to the lower bandwidth APB, where most of the peripheral devices in the system are located. AMBA APB provides the basic peripheral[5] macro cell communications infrastructure as a secondary bus from the higher bandwidth pipelined main system bus. Such peripherals typically:

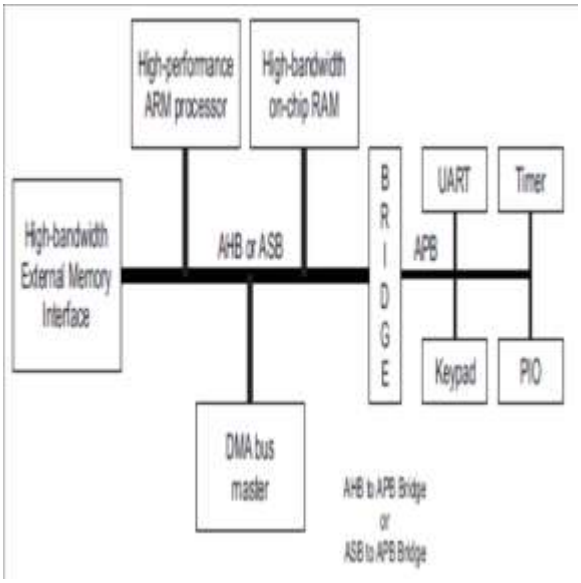

Figure-1. AMBA based Simple Microcontroller

(i) Have interfaces which are memory-mapped registers

(ii) Have no high-bandwidth interfaces

(iii) Are accessed under programmed control.

3. OVERVIEW OF AMBA BUSES

The Advanced Microcontroller Bus Architecture (AMBA) is ARM's no-cost, open specification[2], which defines an on-chip communications standard for designing high performance Embedded Microcontrollers. Three distinct buses are defined within the AMBA specification:

(i) The Advanced High-performance Bus (AHB)

(ii) The Advanced System Bus (ASB)

(iii) The Advanced Peripheral Bus (APB).

A. Advanced high-performance bus (AHB)

AHB is a new generation of AMBA bus, which is intended to address the requirements of high-performance synthesizable designs. It is a high-performance system bus that supports multiple bus masters and provides high-bandwidth operation. AMBA AHB[6]

implements the features required for high-performance, high clock frequency systems including:

- High performance
- Pipelined operation
- Multiple bus masters
- Burst transfers
- Single-cycle bus master handover
- Non-tri state implementation

Wider data bus configurations (64/128bits). Bridging between this higher level of bus and the current ASB/APB can be done efficiently to ensure that any existing designs can be easily integrated.

An AMBA AHB design may contain one or more bus masters typically a system would contain at least the processor and test interface. However, it would also be common for a Direct Memory Access (DMA) or Digital Signal Processor (DSP) to be included as bus masters. The external memory interface, APB Bridge and any internal memory are the most common AHB slaves. Any other peripheral in the system could also be included as an AHB slave. However, low-bandwidth peripherals typically reside on the APB.

B. Advanced System Bus (ASB)

The AMBA ASB is for high-performance modules. It is an alternative system bus suitable for use where high-performances features of AHB are not required. ASB also supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macro cell functions.

Features of ASB:

- Burst transfers
- Pipelined transfer operation
- Multiple bus masters.
- Advanced peripheral bus (APB)

The Advanced Peripheral Bus (APB) is part of the Advanced Microcontroller Bus Architecture (AMBA) hierarchy[7] of buses and is optimized for minimal power consumption and reduced interface complexity.

The AMBA APB should be used to interface to any peripherals which are low-bandwidth and do not require the high performance of a pipelined bus interface. The latest revision of the APB ensures that all signal transitions are only related to the rising edge of the clock. This improvement means the APB peripherals can be integrated easily into any design flow. Features of APB:

- Low power
- Latched address and control
- Simple interface
- Suitable for many peripherals

These changes to the APB also make it simpler to interface it to the new Advanced High- performance Bus (AHB).

4. OPERATION OF AHB2APB BRIDGE

The 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 [8]. The AHB2APB performs transfer of data from AHB to APB for write cycle and APB to AHB for Read cycle [2].

A. Features *of* AHB2APB Bridge

Interface between AMBA high performance bus (AHB) and AMBA peripheral bus (APB)[2], provides latching of address, controls and data signals for APB peripherals. Supports for the following:-

- APB compliant slaves and

peripherals.

- Peripherals which require additional wait states.

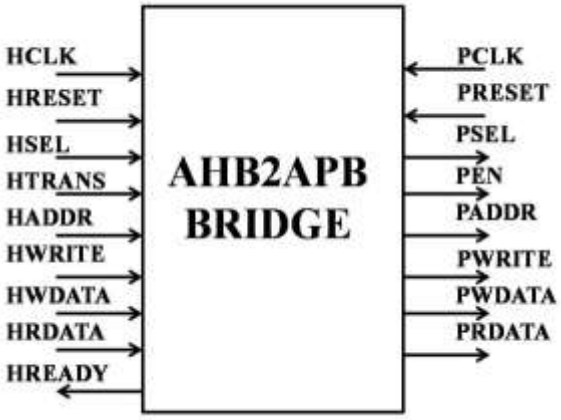

Figure 2. Pin details of AHB2APB Bridge

B. AHB Response

The sub-module AHB Response sequences the way that the AHB2APB responds to AHB requests. Valid commands are forwarded to control transfer for action. Invalid commands are not forwarded and an error message is generated. It operates on AHB CLOCK and RESET. The control Transfers block in Figure 3 transfers AHB control signal to the APB access with appropriate delays inserted to map the pipelined AHB protocol to the two cycle APB protocol. It ensures that only one request is presented to the APB access while it is processing a request. It operates on AHB CLOCK and RESET.

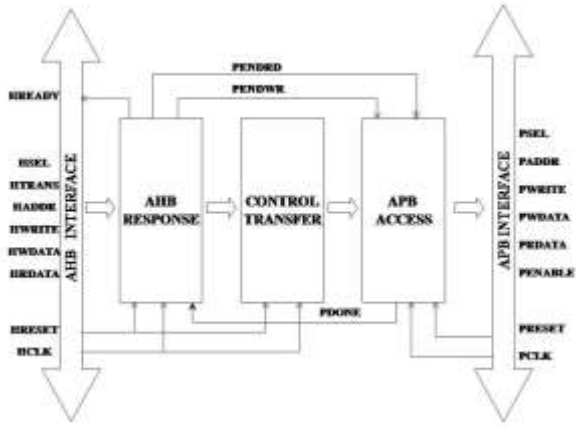

Figure 3. Internal architecture of the bridge

C. APB *Access*

The APB access generates the control signals on the APB for read and writes cycles. It operates on APB CLOCK and RESET. The APB Bridge is the only bus master on the AMBA APB. In addition, the APB Bridge is also a slave on the higher-level system bus. The bridge unit converts system bus transfers into APB transfers and performs the following functions:

- Latches the address and holds it valid throughout the transfer.
- Decodes the address and generates a peripheral select (PSEL). Only one select signal can be active during a transfer.
- Drives the data onto the APB for a write transfer.
- Drives the APB data onto the system bus for a read transfer.
- Generates a timing strobe, PENABLE, for the transfer.

5. DESIGN OF AHB2APB BRIDGE

AHB2APB Bridge operates on HCLK and APB access sub module operates on PCLK. AHB response and Control transfer is together termed as AHB interface and APB access is termed as APB interface to ensure the correct generation of suitable control signals and address we use three internal signals in the bridge module namely:

- PENDWR (Pending Write)
- PENDRD (Pending Read)
- PDONE (Peripheral operation done)

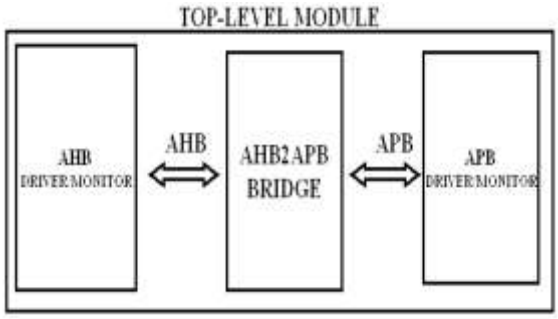

Figure 4. Design of AHB2APB Bridge

The capture of address & control for Write or Read operation is done when HREADY, HTRANS and HSEL are valid. READY is the only signal that is the output from the bridge to AHB master to cope up the communication between AHB and APB. Hence the generation of HREADY signal is very significant in the bridge module. By using the internal signals PENDWR and PENDRD and double synchronized signal (Double synchronization is explained later in this section) PDONE, HREADY generation is made easy to capture the next control for Write or Read operation from AHB to APB. Since the sub modules operate on different clock domains namely HCLK and PCLK, there is a need for interfacing these clock domains. Any two systems are considered asynchronous to each other:

- When they operate at two different frequency
- When they operate at same frequency, but at two different clock phase angles

This interfacing is difficult in the sense that design becomes asynchronous at the boundary of interface, which results in setup and hold time violation, Meta stability and unreliable data transfers. Hence we need to go out for special design and interfacing techniques. In such a case if we need to do data transfer, there are very few methods to achieve this namely:

- Handshake signalling method
- Asynchronous FIFO

Both have its own advantages and disadvantages. In our paper we have used Handshake signalling Method. In Handshake signalling method the AHB interface sends data to APB interface based on the handshake signals PENDWR (or PENDRD) and PDONE signals. The protocol for this uses the same method that is found with 8155 chip used with 8085 based on handshake signals Request and Acknowledge.

6. PROTOCOL

AHB interface asserts the PENDWR (or PENDRD) signal, makes the APB interface to accept or to send the data on the data bus.APB interface asserts the PDONE signal, asserting that it has accepted or sent the data. This method is straightforward but it has got loop holes. when APB interface samples the AHB interface‘s PENDWR (or PENDRD) line and AHB interface samples APB interface’s PDONE line, they are done with respect to their internal clock, so there will be setup and hold time violation. To avoid this we use double stage synchronizers, which are immune to meta stability to a good extent. The figure 5, shows how this is done

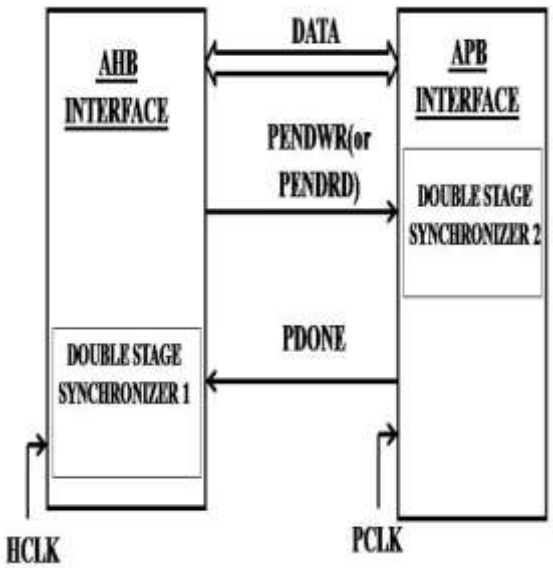

Figure 5. Handshake signalling method

The figure below shows the internal blocks of double stage synchronizer for PENDWR

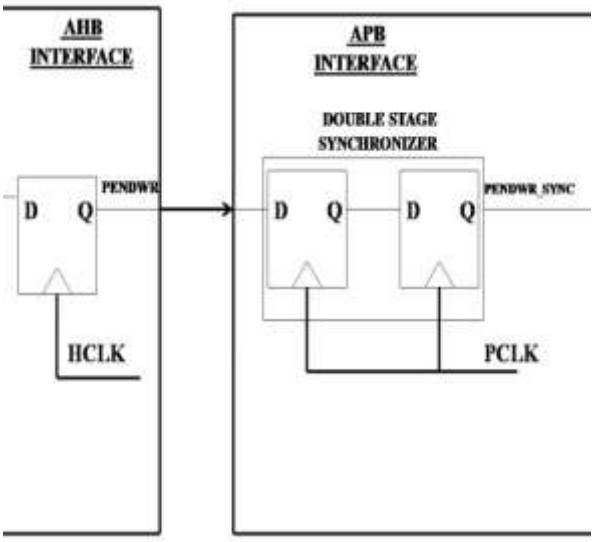

Figure 6. Double Stage Synchronizer

The double synchronizers for PENDRD and PDONE will be same as double synchronizers for PENDWR with the only difference that synchronizer for PDONE is made to operate on HCLK unlike PENDWR and PENRD which operate on PCLK. If we do the double synchronizing, then the transfer rate comes down, due to the fact that a lot of clock cycles are wasted just handshaking.

Design of AHB driver/monitor

The AHB Driver/Monitor is the module that drives the AHB2APB Bridge with suitable control signals, address and data and also monitors the input data that is received from the bridge, so as to create the environment of the AHB Master. The control signals, address and data are fed into this module with the use of ".mem" file. If the user needs to change the values of control, address and data, he has to change the entries in this ". mem" file. This " mem" file is copied with suitable test cases and it has a command code of suitable width. From this file, control signals, address and data are stored in an internal memory and then it is transmitted to the bridge depending on the signal from the bridge saying whether it is ready or not namely HREADY. If the transaction is a write operation, HWDATA is sent to the bridge in the very next clock cycle of HCLK.

The HWDATA that is sent to the bridge during a write operation is also written into a memory, later on this data is used to compare with the HRDATA that is received from the bridge during read operation. After comparison of these two data, suitable messages are displayed for equality and also if read operation is done without write. After completing all the transactions for the controls present in the file, simulation is stopped by giving suitable command code in the ".mem" file. This module also contains the block for HCLK generation distributed to module AHB2APB Bridge and RESET generation distributed to the two other modules namely AHB2APB Bridge and APB Driver/Monitor and these two signals generated are obviously used in this module also.

Design of APB driver/monitor

The APB Driver/Monitor is the module that drives the AHB2APB Bridge with suitable data and also monitors the control signals, address and data that is received from the bridge, so as to create the environment of the APB Slave. The control signals, address and data that is received from the bridge is suitably used for the data transaction from bridge to this module or vice versa depending whether it is a write or a read operation. The PWDATA that is received from the bridge during a write operation is stored in a memory so that the same data can be sent to the bridge during a read operation as PRDATA. If there were no prior written data in this memory then the PRDATA that is sent to the bridge is a garbage value which can be any combination of 0's and

1's.This module contains block for PCLK generation, which is distributed to AHB2APB bridge module, and the PCLK generated is obviously used in this module also.

Design of top module

This module is simplest and also very prominent. All the signals are taken as wire to interconnect various modules present under this top module.

In this module all the three modules namely:

- AHB Driver/Monitor
- AHB2APB bridge
- APB Driver/Monitor

These modules are all instantiated using Positional assignments which is again simple compared to naming assignment which is little tedious.

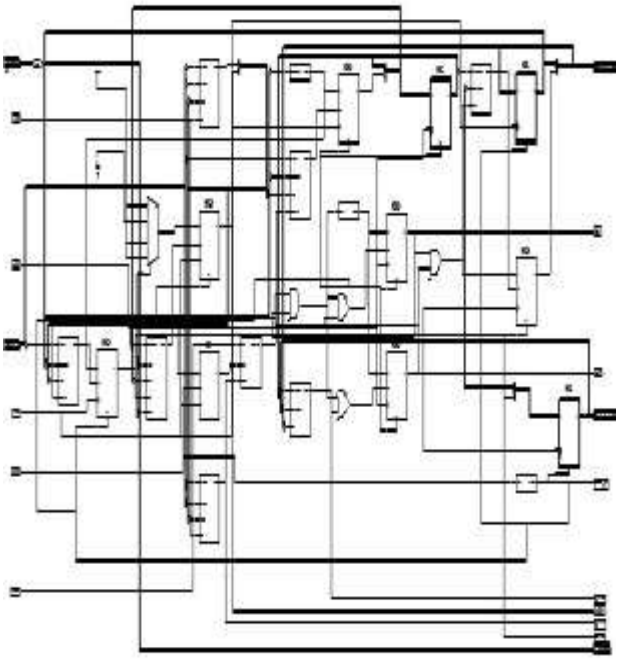

Figure 7. Synthesized Netlist of Bridge Module

7. BACK ANNOTATION

Back annotation is the translation of a routed or fitted design to a timing simulation Netlist. Back annotation was performed on the Xilinx generated synthesis file for the AHB2APB Bridge module. In our paper only Bridge module is the synthesized module and AHB driver and APB monitor were test bench modules.

8. SIMULATION RESULTS RTL SIMULATION RESULTS

(1) With HCLK and PCLK having a phase difference of 90₀ and same frequency

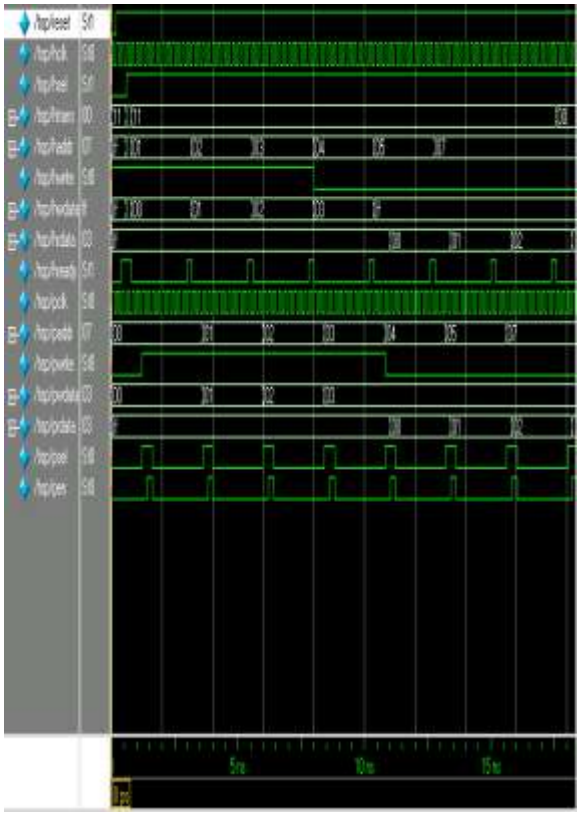

Figure 8. Burst of Write & Read Transfers from AHB master to APB peripheral through AHB2APB Bridge with HCLK(AHB Master clock) and PCLK(APB Peripheral clock) having a ratio of 1:3 with phase difference of 900 and same frequency

(2) With HCLK and PCLK having a ratio of 1:2

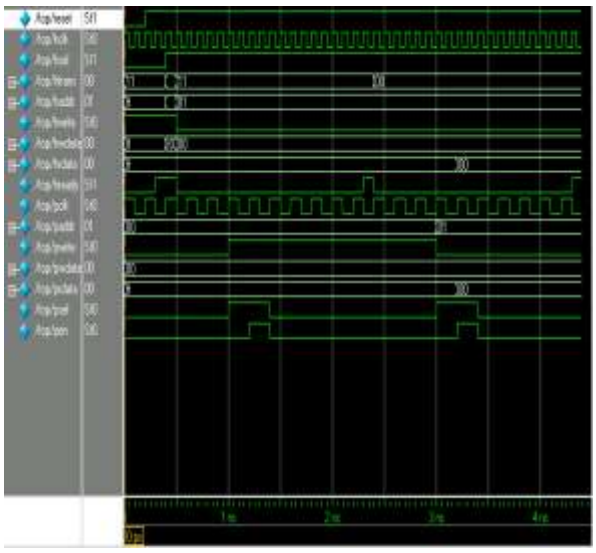

Figure 9. Single Write followed by Single Read from AHB master to APB peripheral through AHB2APB Bridge with HCLK(AHB Master clock) and PCLK (APB Peripheral clock) having a ratio of 1:2, with same phase difference and different frequency.

Back Annotation Simulation Results

(3) With HCLK and PCLK having a ratio of 1:2.

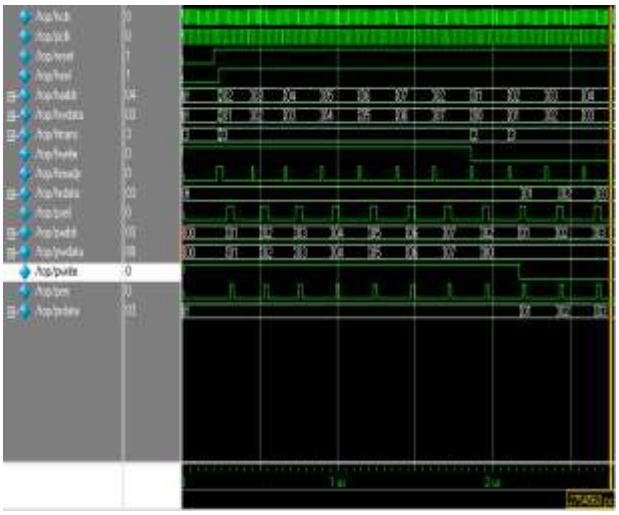

Figure 10. Burst of Write & Read Transfers from AHB master to APB peripheral through AHB2APB Bridge with HCLK(AHB Master clock) and PCLK (APB Peripheral clock) having a ratio of 1:2, with same phase difference and different frequency.

(4) With HCLK and PCLK having a phase difference of 90^0 and same frequency

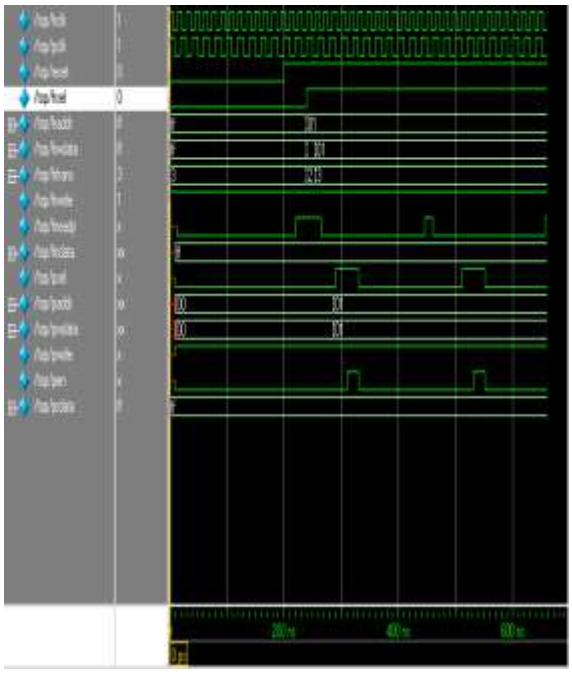

Figure 11. Single Write Transfer from AHB master to APB peripheral through AHB2APB Bridge with HCLK (AHB Master clock) and PCLK(APB Peripheral clock) having a ratio of 1:3 with phase difference of 900 and same frequency

9. CONCLUSION

The RTL Simulation of AHB2APB Bridge has been verified and validated by using suitable test benches namely AHB Driver/Monitor and APB Driver/Monitor. The AHB2APB Bridge has been successfully synthesized by the extraction of Synthesized Netlist with unit delays and verified by comparing the Gate level Simulation with RTL Simulation results. The Back Annotation of AHB2APB Bridge has also been successfully completed by the extraction of Synthesized Netlist with suitable delays & verified by the comparison of Gate level simulation with RTL simulation results. Thus AHB2APB Bridge is a standalone solution to extract the advantages of newly developed ARM based AMBA AHB bus by bridging the common gap between AHB and the existing APB bus.

REFERENCES:

[1] Jaehoon Song, Student member, IEEE, Hyunbean Yi, Member,IEEE, Juhee Han, and Sungju Park, Member, IEEE,"An Efficient SOC Test Technique by Reusing On/Off-Chip Bus Bridge"IEEE Transcactions on Circuits and Systems-I: Regular Papers, Vol,56,No.3,March2009.

[2] AMBA Specification (Rev 2.0).

[3] Sameer Palnitkar, Verilog HDL, A Guide to Digital Design and Synthesis, 2nd Edition, 2008.

[4] Xilinx ISE Synthesis and Verification Design Guide

[5] Sangik Choi and Shinwook Kang, Mobile SamsungElectronics Co., Ltd, "Implementation of an On-Chip Bus Bridge between Heterogeneous Buses with Different Clock Frequencies". IEEE, IDEAS'05, 1098-8068/2005.

[6] AHB Example AMBA system; Technical Manual ARM1999.

[7] Flynn, D. Adv. RISC Machines Ltd., Cambridge, "AMBA: enabling reusable on-chip designs", IEEE Micro, Publication Date: Jul/Aug 1997.

[8] Wang Zhonghai, Ye Yizheng, Wang Jinxing, and Yu Mingyan, "Designing AHB/PCI Bridge," in Proceedings of 4th International Conference on ASIC, Oct 2001,pp.578-580.