

# DESIGN & IMPLEMENTATION OF PCI TO AMBA AHB BUS INTERFACE

Snehal Thesia

VLSI & Embedded System

Gujarat Technological University PG School, Gujarat, India

**Abstract**—The purpose of this paper is to integrate the PCI CORE TO AMBA AHB BUS INTERFACE. The design accounts for crossing clock domains as it is inevitable in a system on chip design with multiple components running at different frequencies. The Bridge maps various control signals and address spaces from one bus into those of another bus. This high performance bridge enables any embedded processor connected to an AMBA AHB to act either as a host bridge or a simple bridge on a PCI bus. A host bridge connects CPU to on board PCI devices, it contains PCI arbiter and is used to initialize and communicate with PCI devices. A simple bridge is usually implemented in PCI plug-in board and makes CPU act as a master/target PCI device. The interface will be useful in many SoC having AMBA bus in the RISC processor which will be connected to other peripherals.

**Index Terms**:- PCI, AMBA AHB, AMBA Master, AMBA Slave, PCI INTIATOR, PCI TARGET

## I. INTRODUCTION

Before PCI old buses like ISA, EISA, VL/VESA bus were used to connect peripheral components but, during 90's there were huge development in Graphical user interface (GUI) and multimedia so old bus architecture became the bottleneck. The computer's developing high-performance needed the relevant rapid bus to complete the transaction. PCI local bus met the need.

The AMBA AHB is a new generation bus, which was released by ARM Ltd. It is for high-performance, high clock frequency system modules. The AMBA AHB acts like the high-performance system backbone bus. AHB supports the efficiently in connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral microcells functions. AHB is also specified to ensure ease of use in an efficient design flow using synthesis and automated test techniques. This paper is to design PCI/AHB Bridge which connects the SoC platform with peripheral components based on PCI Bus<sup>[3]</sup>.

### A. Scope of PCI to AHB Bridge

System Chips with the PCI – AHB Bridge have been widely adopted for various applications as mentioned below:

- Sound
- Graphics
- Network Cards

The PCI – AHB Bridge is suitable for a wide range of applications, especially those needing to achieve high performance with low cost. Its synchronal design and standard-cell based approach allow the users to quickly integrate the PCI – AHB Bridge into their SoC designs.

- Sound
- Graphics
- Network Cards

The PCI – AHB Bridge is suitable for a wide range of applications, especially those needing to achieve high performance with low cost. Its synchronal design and standard-cell based approach allow the users to quickly integrate the PCI – AHB Bridge into their SoC designs.

## II. PROPOSED SPECIFICATIONS

### A. PCI - AHB Bridge Specifications

Operating frequency of PCI is 66MHz<sup>[2]</sup>. Operating frequency of AHB is 100MHz<sup>[3]</sup>. Communication between AHB and PCI is achieved by using Asynchronous FIFO in the interfaces.

### B. PCI - AHB Bridge Supported Features<sup>[1][2][6][7]</sup>

- 1) AHB and PCI bus operates at independent clock domains.
- 2) Supports 32-bit bidirectional data transfer
  - a) 32-bit Bus width and Max Data Size is 256 bytes
  - b) Minimum Latency of Design is 2 clock cycles.
- 3) **CONFIGURATION INTERFACE**
  - a) **PCI SUPPORTS**
    - Address Bus Width of 6 bits.
  - b) **AHB SLAVE SUPPORTS**
    - Two Clock cycles needed for Read & Write Transfer.
    - Transfer Size varies from 8 bit to 32 bit.
    - It supports all four Transfer Types. (NONSEQ, SEQ, IDLE, BUSY)
    - It supports all four Response Types (OKAY, ERROR, SPLIT, RETRY)
- 4) **SLAVE INTERFACE**
  - a) **PCI SUPPORTS**
    - Single write cycle.
    - Burst (burst length of 8 bytes to 64 bytes) write cycle.
    - Single prefetchable read cycle.
    - Single Non-prefetchable read cycle.
    - Burst (burst length of 8 bytes to 64 bytes) prefetchable read cycle
    - Burst (burst length of 8 bytes to 64bytes) Non-prefetchable read cycle
  - b) **AHB SLAVE SUPPORTS**
    - Two clock cycles needed for Read & Write Transfer.
    - It support 4/8/16 beat incrementing or wrapping Burst mode.
    - It support single transfer and incrementing burst mode with unspecified length.
    - Transfer Size varies from 8 bit to 32 bit.
    - It supports all four Transfer Types. (NONSEQ, SEQ, IDLE, BUSY)

- It supports all four Response Types (OKAY, ERROR, SPLIT, RETRY)
- 5) **MASTER INTERFACE**
- a) **PCI SUPPORTS**
- Single write cycle
  - Burst write cycle (burst length of 8bytes to 64 bytes)
  - Single prefetchable read cycle
  - Single Non-prefetchable read cycle
- b) **AHB MASTER SUPPORTS**
- Burst prefetchable read cycle (burst length of 8bytes to 64bytes)
  - Burst Non-prefetchable read cycle (burst length of 8 bytes to 64bytes)
- Two clock cycles needed for Read & Write Transfer.
- It support 4/8/16 beat incrementing or wrapping Burst mode.
  - It support single transfer and incrementing burst mode with unspecified length.
  - Transfer Size varies from 8 bit to 32 bit.
  - It supports all four Transfer Types. (NONSEQ, SEQ, IDLE, BUSY)
  - It support all four Response Types (OKAY, ERROR, SPLIT, RETRY)

### III. ARCHITECTURE OF PCI – AHB BRIDGE

As shown in Fig. 1 the PCI – AHB Bridge is an interface, which is used to connect an agent that is both master and target, to the PCI Local Bus. Because of this double capability the Bridge is divided into a Master Block, a Target Block and a Configuration Block.

There is a clear division between the master and the target part, not only from the functional, but also from the architectural point of view. These two parts have its own state machine that can operate independently from other in order to avoid possible deadlocks. Although it should be possible to build the other functions into single blocks with master/slave capability, a separation between the two parts was chosen.

Except from the Configuration block, the others are divided into four functional parts. The core is a bridge between the PCI bus and the AMBA AHB bus. The core is capable of connecting to the PCI bus via both a target and initiator/master interface. The connection to the AMBA bus is an AHB master interface for the PCI target functionality and an AHB slave interface for the PCI initiator functionality.

The core uses the PCI initiator to connect to the PCI bus and an AHB master to connect to the AMBA bus. Configuration registers in the core are accessible via PCI Master and Target Interface Block.

#### A. AHB Master

AHB Master is able to initiate read and write operations by providing an address and control information. Only one bus master is allowed to actively use the bus at any one time.

#### B. AHB Slave

An AHB slave responds to read or write transfers initiated by masters in the system within specified address range. The slave signals back to the master:

- The success
- Failure
- Waiting of the data transfer.

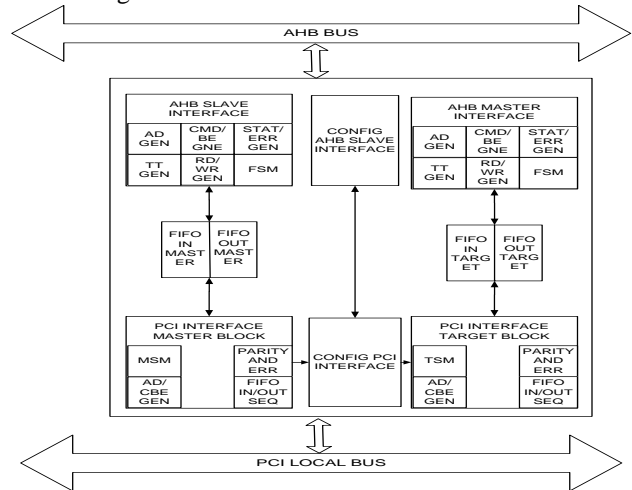


Fig. 1: Block Diagram of PCI/AHB Bus Interface

#### C. Configuration Interface

The Configuration Interface consists of configuration registers and handles the read and writes cycles. The configuration registers in the configuration space of the PCI Core can be accessed through the PCI Bus in a direct access or through the Local Configuration Interface during AHB's access. These registers control the modes and options in the PCI and provide vital information to the PCI host.

#### D. PCI Master

PCI Master accepts bus transactions from the AHB SLAVE Interface and passes those transactions on to the PCI Master State Machine. It drives the address phase and transaction boundary (FRAME\_N). The Master initiates a transaction and drives data handshaking (IRDY\_N) with the target and performs the asynchronous decoupling between the peripheral clock domain and the PCI clock domain for master transactions. This block implements the address translation for PCI Master Transactions and control registers.

#### E. PCI Target

This block accepts transactions from the PCI Target State Machine and passes those transactions to the AHB MASTER Interface. It claims the transaction by asserting DEVSEL\_N and handshakes the transaction (TRDY\_N) with the PCI Master and performs the asynchronous decoupling between the PCI clock domain and the peripheral clock domain for Target transactions. It also implements the address translation for PCI Target transactions and control registers.

#### F. Asynchronous Fifo

Communication between AHB and PCI is achieved by using Asynchronous FIFO in the interfaces. An asynchronous FIFO uses different clocks for reading and writing.

IV. DETAILED PIN DIAGRAM OF PCI CORE TO AHB BUS BRIDGE

A. AHB Slave Interface

- Fig. 2 shows the top module of AHB SLAVE INTERFACE which includes state machine, BE generator, Command generator, AD generator and Transfer type generator.
- State Machine: It generates the control signals which controls the flow when to send Data, when to send Address, this is selected by AD\_sel. When to send command and when to send Byte Enable is controlled by CB\_sel generated by state machine. 2-bit HRESP signal to acknowledge AHB bus whether transaction are okay or not. Read request sends read request to FIFO\_OUT module and Write request sends write request to FIFO\_IN module.
- Address Generation: It sends address/data as per the select line.
- Command Generation: It sends command to inform whether it is a read operation or write operation.
- Byte Enable Generation: It shows which byte lane is having the valid data.
- Transfer Type Generator: It indicates which kind of transfer is taking place. Is it a single transfer, incremental, incremental4, wrap 4, incremental8, wrap8, incremental16, wrap16?

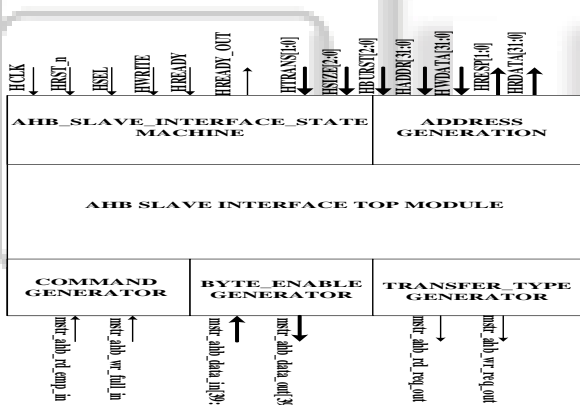


Fig. 2: Top module of AHB SLAVE INTERFACE

B. PCI Master Interface

Fig. 3 shows Top module of PCI MASTER INTERFACE. It includes:

- PCI Master Interface State Machine: It controls the operation of Master interface generates signal as per requirement.
- AD\_BUFFER: It is consist of tristate buffer which pass data from AD\_IN\_REG to AD when OE is 1 and is assigned AD to fifo\_out\_data\_in when OE is 0.
- CBE\_BUFFER: It is consist of tristate buffer which passes data from CBE\_IN\_REG to CBE when OE is 1 and is assigned CBE to fifo\_out\_CBE\_in when OE is 0.
- Parity Generator: In this block Parity is generated for Address/Data and for CBE.
- Mem\_IO Generator: This block decides whether it is memory address space or IO address space.
- Transfer\_type Generator: For Memory space it works either in linear mode or in wrap mode.

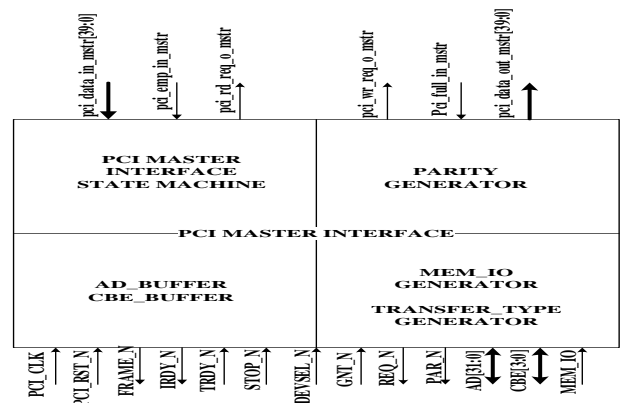


Fig. 3: Top module of PCI MASTER INTERFACE

C. FIFO

- Fig. 4 shows the top module of FIFO which includes Write Pointer and Read Pointer address generation along with FIFO Full and Empty signal generation. It also includes FIFO Memory block with configurable Width and Depth.
- Write addresses are generated using 6-bit counters and then these address are sent to read block. Before sending it to read block they are converted into gray encoding to reduce error generation as only one bit change when we move from one count to another. FIFO FULL is generated by comparing write pointer and read pointer if lower 5-bits are same and msb's are not same of each other than FIFO is full otherwise we can write data into it.
- Read addresses are generated using 6-bit counters and then these address are sent to write block. Before sending it to write block they are converted into gray encoding to reduce error generation as only one bit change when we move from one count to another. FIFO EMPTY is generated by comparing read pointer and writes pointer if lower 5-bits then FIFO are empty otherwise we can read data into it.

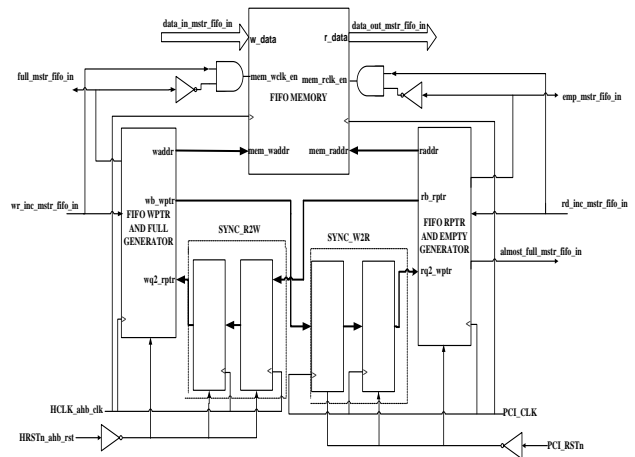


Fig. 4: Top module of FIFO

V. EXPERIMENTAL RESULTS

- Results are verified functionally by giving directed test bench and the simulator used for Functional Verification is Xilinx. Bottom up Approach was used to design this bridge.

- First of AHB SLAVE INTERFACE was designed and the simulation results obtained are shown under Section A. Then after FIFO was designed, the simulation results obtained are shown in Section B. Then after PCI MASTER INTERFACE was designed, the simulation results are shown in Section C. The combined top module was then designed and Simulation Result of the top Interface is shown in Section D.

**A. Simulation Results of AHB Slave Interface**

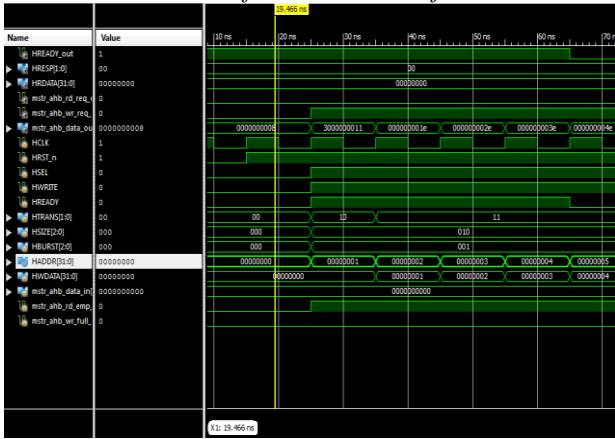


Fig. 5:

**B. Simulation Result of FIFO**

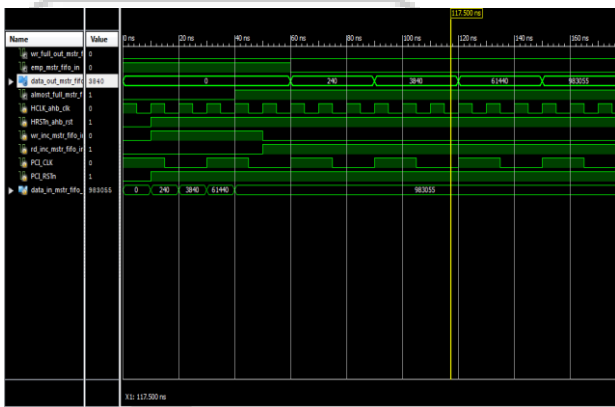


Fig. 6:

**C. Simulation Result of PCI MASTER INTERFACE**



Fig. 7:

**D. Simulation Results of AHB WRITE PCI READ**



Fig. 8:

**VI. CONCLUSION**

In this paper the PCI CORE TO AHB bus interface has been achieved which is functionally verified with Xilinx ISIM simulator. The operating frequency of AHB is kept as 100MHz and the operating frequency of PCI Bus is kept 66.66MHz. Synchronization between two protocols is achieved using Asynchronous FIFO. The bridge which is proposed in this paper is generic so that in future if one wants to replace AHB with Wishbone architecture then it will be done easily.

**ACKNOWLEDGEMENT**

I pay my sincere gratitude to researchers whose research work was helpful to achieve these results.

**REFERENCE**

- [1] ARM LTD, "AMBA 3 AHB-LITE PROTOCOL SPECIFICATION", 2001.
- [2] PCI Special Interest Group, "PCI Local Bus Specification Revision 3.0 (volume1)", 2002 pp. 15-136.
- [3] WANG Zhonghai, YE Yizheng, WANG Jinxiang, W Mingyan,"Designing AHB/PCI Bridge" Shanghai. ASIC, 2001. Proceedings. 4th International Conference on 2001 pp. 578 – 580.
- [4] AMBA-AHB PCI Bridge IP Core Introductory Document PLDA Ltd [Online]. Available: [http://www.plda.com/download/press\\_release/launch\\_a\\_hb\\_pci\\_jan\\_2002.doc](http://www.plda.com/download/press_release/launch_a_hb_pci_jan_2002.doc)
- [5] Aeroflex. Gaisler; "Gaisler Research IP Core's Manual. Colorado Springs, CO: Gaisler Res.", 2013 [Online]. Available: <http://www.gaisler.com/cms/>, ver. 1.3.1 pp. 876-933.
- [6] E. Lama Vaquero "PCI Interface", March 2000 [Online]. Available: <http://microelectronics.esa.int/papers/PCI-AMBA-ELV.pdf>
- [7] E. Lama Vaquero "PCI core/ AHB Bus Interface", Nov 2000 [Online]. Available : <http://microelectronics.esa.int/papers/PCI-AMBA-ELV.pdf>