Developing an IP core for PCI Using VHDL and its Verification

Dr. Rekha K. R, Ashwin Balakrishna
SJB Institute of Technology, Bangalore, Karnataka, 560060, India (VTU)

ABSTRACT **--- PCI is a computer bus for attaching hardware devices in a computer, these hardware devices are usually referred as the peripheral devices. PCI architecture was started by Intel in 1990, since then PCI has undergone a lot of changes and today PCI stands as one of the most versatile device with well established protocols to communicate with the peripheral devices. PCI allows the peripheral devices to directly access the system memory but it uses the bridge to connect to the front side bus and therefore to the CPU. The communication with the peripheral devices is established through cycles of bus transactions. The PCI bus transactions involve address phase followed by the data phase. The data phase may be from initiator to target or vice versa depending on the type of transactions taking place. A PCI interface will connect any generic devices to the PCI bus. The interface will create the communication between master and the slave devices. The interface will receive the command from the, master and send required response to the master and the slave as per the availability or status of both the master and the slave device. Generally most commands are provided by the master, our interface design includes read and write operations and will be able to communicate to register and RAM through the PCI.**

Index Terms **— PCI Protocol, PCI interface, VHDL.**

I. INTRODUCTION

PCI stands for peripheral component interconnect, as the name suggests PCI is a device which is mainly used to act as an interface between the master and the slave device. Use of devices such as PCI lead to standardization of the interface devices allowing these standard interfaces to connect to the a PC, capable of sustaining the high data transfer rates needed peripheral devices such as modern graphic controllers, storage media, network interface cards and other devices.[1]It may seem that PCI is a set of bus lines and computer users may have mistaken that PCI is just a set of electrical wires which help in transfer of data but PCI is actually a complete set of specification defining how different parts of the computer should interact. PCI bus is an improved bus for PC compatible computers. It has proved to be faster than previously introduces buses like the ISA, VESA. It has faster transfer rates and it can be expandable to both 32 bit and 64 bit. PCI covers most issues relating to computer interface. The PCI has distinct interface pins with specific functionality. We have to interface other devices as per the pin configuration of the PCI. The PCI bus has 4 main characteristics:

Synchronous: PCI bus operation uses one clock, which usually operates at 33 MHz or may choose to operate lower to save power.PCI can also operate at 66MHz if the hardware supports it.

Transaction/burst oriented: bus transactions are followed by an address phase followed by one or more data phases depending on the type of transaction taking place. The transaction is usually address phase followed by one or more data phases. Bus mastering: the operation of PCI works as a master slave relationship where usually the processor acts as an initiator.

Plug and Play: Host CPU/ host OS has the capability to identify the PCI devices connected to the board of the computer, hence it configures itself at the boot up. This feature is most important and is the reason for PCI devices popularity. PCI timing-PCI specifies timing related to its clock with a 33 MHZ clock, we have: 7ns0ns Tsu/Th (setup/hold) constraint on inputs 11ns Tco (click to output) on outputs. [2]

II. OBJECTIVE

The project mainly aims to provide communication between a master and slave device. Systems which include complex components always require interfaces which will act as mediators between to sub systems and allow them to communicate without any glitches. PCI is one such interfacing device which has become popular in today's systems. PCI will act as a mediator between the processor (master) and the peripheral device (slave). Both the master and slave will communicate only with the mediating interface device the PCI, but the versatile characteristics of the PCI device ensures that both the devices communicate successfully with each other.

III. INTERFACE I/O SIGNALS

Frame: It is the activation signal. When it is low the device indicates that it is ready to perform. [3]

Command or Byte Enable: It is a 4 bit signal, in the address cycle it indicates command and in the data cycle it indicates byte enable. [3]

Address: It is a 32 bit bus. It provides both address and data at the address phase and data phase respectively.

Initiator ready: It indicates that the master device is ready for transaction.

Target ready: Target ready is low when the slave device is ready to make transaction.
Device select: The target asserts device select (low) as acknowledgement to indicate that the address has been positively decoded.

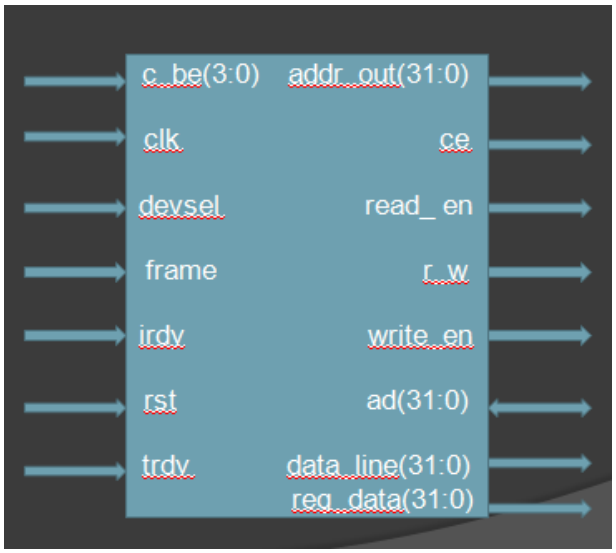

Fig.1. Interace block

IV. ADDRESS MAPPING

In our design, the method we have used to select the slave device is called the address mapping. PCI device has 32 bit address bus which means it can have 2^32 address which is 4294967296 number of addresses. Considering the address lines which are present in the PCI, it is evident that PCI has the ability to address many devices. Although PCI is capable of communicating with many devices at once due to electrical issues most of the PCI based systems usually restrict their communication to few devices.

V. SIMULATION ENVIORNMENT

In our system we consider a processor as the master of the system and a PCI target device as the interface. The master i.e. the processor provides the commands and indicates what type of transaction has to take place; basically the processor will act as an initiator. The transaction initiated by the processor will be received by the PCI target and further processing will be done as per the master's request.

VI. TIMING DIAGRAM

A. READ TRANSACTION:

The following timing diagram illustrates a read transaction on the PCI bus:

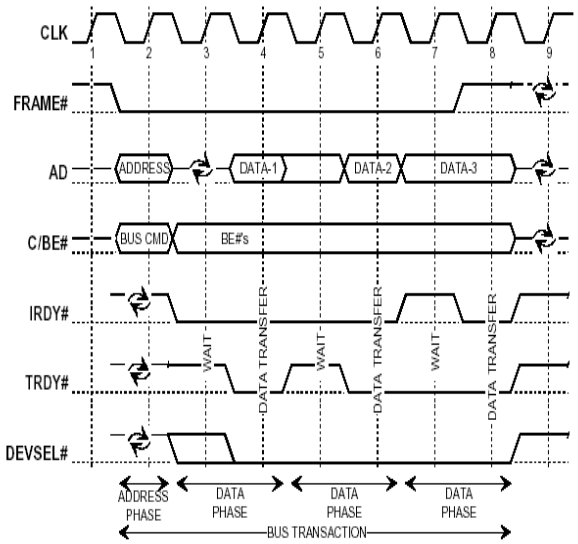

Fig.2. Read cycle of PCI.

Cycle 1- The bus is idle.
Cycle 2- The PCI initiates every transaction with an address phase to indicate which device it wishes to communicate with. This cycle is called the address phase where address of the device is received by the PCI bus from the processor. In this phase the AD lines indicate the address and the C/BE lines indicate the command or the type of transaction required to take place.
Cycle 3- The initiator tri-states the address in preparation for the target driving read data. Here the initiator will drive the valid byte enable signal C/BE indicating which are the valid data among the 32 bit data being transferred. IRDY# is low here indicating that the initiator is ready to make the transfer. Target will also assert the DEVSEL# low as an acknowledgment signal to indicate that the address has been correctly decoded.
Cycle 4- valid data is present at the AD lines. TRDY# is also low here indicating that the target is ready for the transfer of data. When the IRDY# and the TRDY# both are low the data transaction takes place here.
Cycle 5-Target will deassert TRDY#, making it high indicating it needs more time to prepare the next data transfer.
Cycle 6- The second data phase occurs when both the IRDY# and the TRDY# are low and again the data transaction takes place.
Cycle 7-The target provides valid data for the third data phase, initiator delays by indicating it's not ready for the transfer of the data.
Cycle 8- Initiator reasserts the IRDY# signal to indicate it is ready for the transfer of the data. Initiator captures the data provided by the target. Initiator drives the FRAME# high indicating the loast data phase.
Cycle 9-FRAME#, AD, and C/BE# are tri-stated, as IRDY#, TRDY#, and DEVSEL# are driven inactive high for one cycle prior to being tri-stated. [5]

B. WRITE TRANSACTION:

The following timing diagram illustrates a write transaction on the PCI bus:
Cycle 1- The bus id IDLE.
Cycle 2- The initiator asserts a valid address And places a write command on the C/BE# signals. This is the address phase.

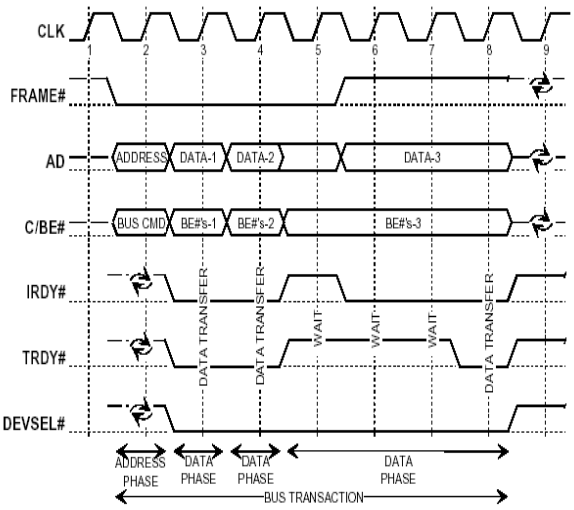

Fig.3. Write cycle of PCI.

Cycle 3- the initiator drives valid data and byte enable signals. The initiator asserts IRDY# low indicating valid write data is available. The target asserts DEVSEL# low as an acknowledgment it has positively decoded the address. The target drives TRDY# low indicating it is ready to capture data. The first data phase occurs the write data.
Cycle 4- The initiator provides new data and byte enables. The second data phase occurs as both IRDY# and TRDY# are low. The target captures the write data.
Cycle 5- The initiator deasserts IRDY# indicating it is not ready to provide the next data. The target deasserts TRDY# indicating it is not ready to capture the next data.
Cycle 6- The initiator provides the next valid data and asserts IRDY# low. The initiator drives FRAME# high indicating this is the final data phase. The target is still not ready and keeps TRDY# high.
Cycle 7- The target is still not ready and keeps TRDY# high.
Cycle 8- The target becomes ready and asserts TRDY# low. The third data phase occurs as both IRDY# and TRDY# are low. The target captures the write data.
Cycle 9- FRAME#, AD, and C/BE# are tri-stated, as IRDY#, TRDY#, and DEVSEL# are driven inactive high for one cycle prior to being tri-stated.[5]

VI. FINITE STATE MACHINES

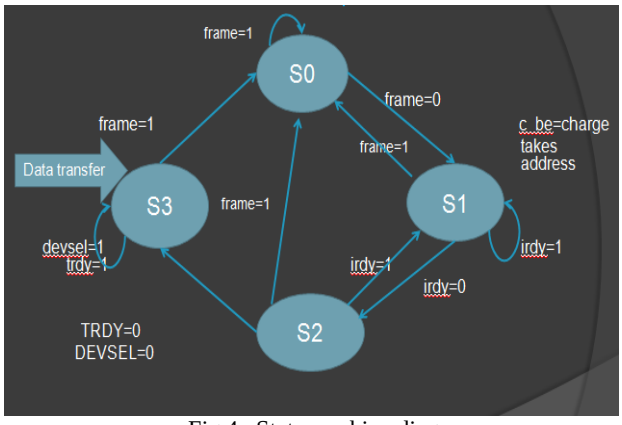

Fig.4. State machine diagram

S0=STATE_IDLE
S1=STATE_TAKE_ADDRESS
S2=STATE_TAKE_DATA
S3=STATE_LAST_DATA

A. STATE_IDLE:

In the idle state all the signals are reset to their initial states and if the system is completely idle here. PCI will wait for the response of the master for any operation to take place. PCI becomes active and ready for operation only when it detects that the FRAME # is low indicating the start of any operation be it configuration cycles or read and write cycles.

B. STATE_TAKE_ADDRESS:

The second state id the take address state where the address of the device to which master wishes to communicate is received by the PCI bus. Here the IRDY# is still at high state once the initiator is ready for the operation to take place the IRDY# is made low and this results in the change of state from the STATE_TAKE_ADDRESS state to the next state.

C. STATE_TAKE_DATA:

The third state is the take data state and if it is found that the DEVSEL# and the TRDY# are low it indicates that the device address has been decoded correctly and further data transaction takes place.

D. STATE_LAST_DATA:

The last and the final state of the data transaction is the STATE_LAST_DATA state where the last data is transferred and this is indicated by the FRAME# signal when it goes high. Hence this completes one full cycle of the data transaction and the PCI reverts back to the IDLE state.

VII. SIMULATION AND IMPLEMENTATION

For simulation process, we have used Xilinx 10.1. For synthesis and verification we have used Xilinx 10.1 and ModelSim 6.4 SE.The simulation results captured by the ModelSim software are shown below:

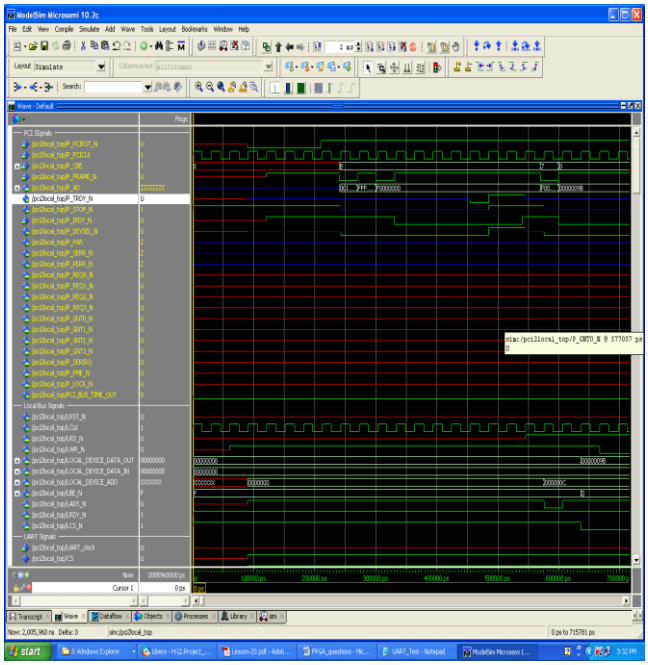

Fig.5. Waveform – Configuration cycle.

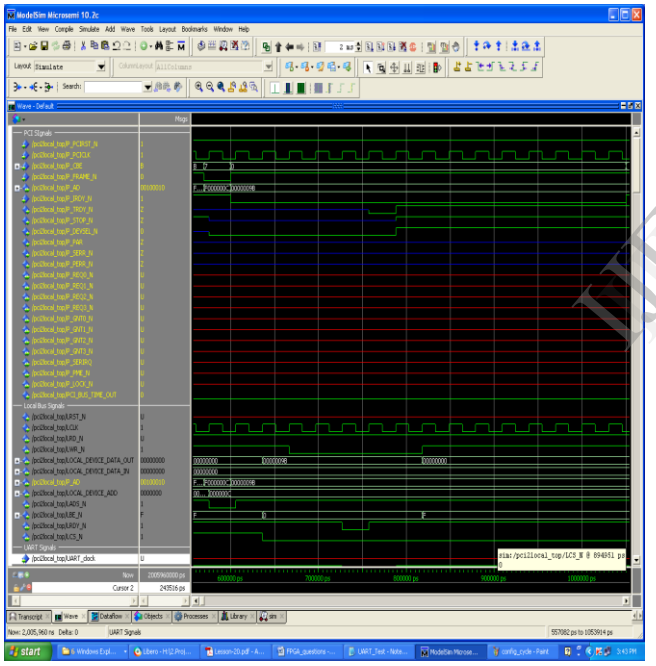

Fig.6. Waveform -Data transaction 1.

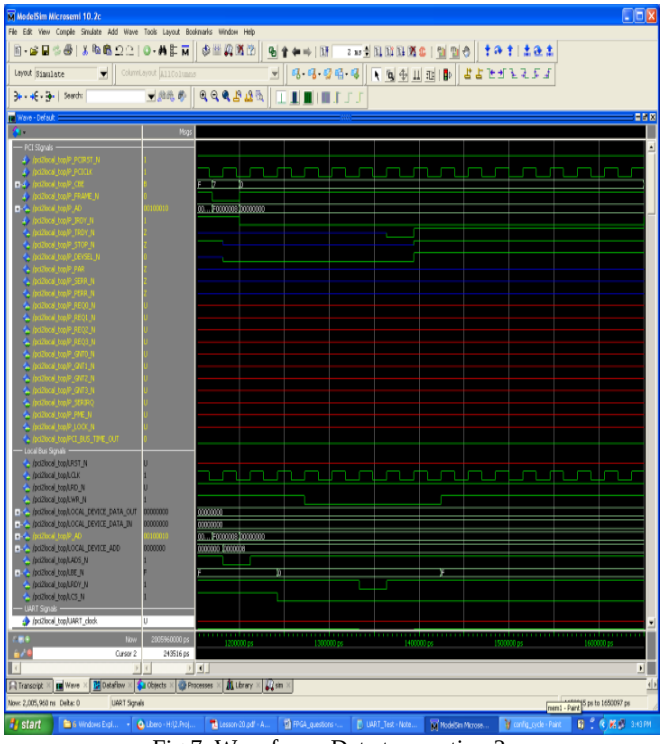

Fig.7. Waveform -Data transaction 2.

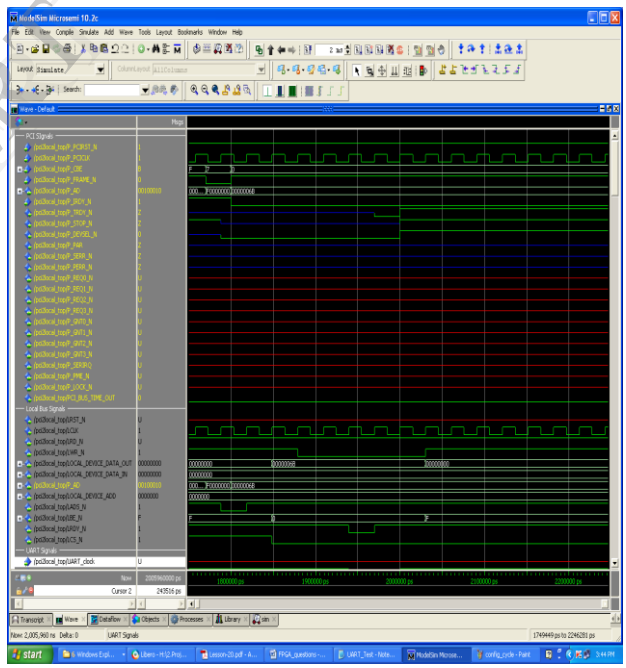

Fig.8. Waveform -Data transaction 3.

VIII. CONCLUSION

Today's computer systems, with their emphasis on high resolution graphics, full motion video, high bandwidth networking, and so on, go far beyond the capabilities of the architecture that ushered in the age of the personal computer. Today's PC s demand high transfer rate and deal with sophisticated peripheral devices hence interfaces like PCI is very much necessary. It is a well thought out standard with a number of forward looking feature that should keep it

relevant well into the next century. PCI was originally conceived as a mechanism for interacting with the master and a slave in a system and to serve as an interface in the system. But today it stands as a well defined device which provides glitch free communication between the master and the slave device.

Though in this project we have not illustrated the communication of the PCI device with any peripheral device the simulation results proves that successful communication will be possible using the PCI. We hope that this project will help in further PCI related invention. PCI has been evolving and adapting since its birth and has proven one of the best interface and today we also have many advanced version of the PCI to match higher speeds and be compatible with more sophisticated systems. Nonetheless PCI has retained its popularity and has played a remarkable role with its significant speed and reliability.

ACKNOWLEDGMENT

The authors would like to acknowledge HOD and faculty of SJBIT (ECE Department) for their consultation and advice throughout the research work.

REFERENCES

[1] Tech-pro.net. (n.d.). Retrieved August 5, 2010, from tech-pro.net: http://www.tech-pro.net/intro_pci.html

[2] Fpga4fun. (2010, September 07). Retrieved September 07, 2010, from fpga4fun: http://www.fpga4fun.com/PCI1.html

[3] PCI Local Bus Technical Summary. (2010, October 10). Retrieved October 10, 2010, from techfest: www.techfest.com/hardware/bus/pci.htm

[4] Figure Block Diagram of Static RAM Table Truth Table. (2010, July 27). Retrieved July 27, 2010, from docstoc:http://www.docstoc.com/docs/4219029/Figure-Block-Diagram-of-Static- RAM-Table-Truth-Table

[5] PCI Bus Timing Diagram. (2010, August 17). Retrieved august 17, 2010, fromsilverhawk:http://silverhawk.net/notes/tutorials/hardware/pcitiming.html

[6] Altera DE1 board. (n.d.). Retrieved December 10, 2010, from terasic:http://www.terasic.com.tw/cgibipage/archive.pl?Language=English No=83