ISSN:0975-9646

PCI Express Interface Development and Simulation for High Speed Data Transmission

Vijitha.C.V[1], Najla.A.P[2], Jayaraj.U.Kidav[3]

[1]*M.E Applied Electronics,*
VCEW, Tamil Nadu, India

[2]*Mtech Electronics Design Technology,*
NIELIT, Calicut, India

[3]*Scientist, NIELIT, Calicut, India*

Abstract **- This paper discusses the development of PCI Express Interface used for high speed data transfer to PC. This serial bus lane-wise architecture with scalability has many advantages over conventional parallel PCI bus. The applications that require low speed can use only one lane of PCI Express wherein applications that requires high speed can use 2, 4, 8, 16 or 32 lanes of PCI Express depending on the speed requirement. Each serial lane of PCI Express has data transfer speed of 2.5 Giga bits per second and is duplex. The interface uses Virtex-6 series of FPGA for implementing the PCI Express.**

Keywords: PCI Express, FPGA, Virtex-6

I INTRODUCTION

From the earliest digital computers, a means of inter-process communication between the various components that built a computing framework was as important as the components themselves. Since those early days, computers have come to transform the digital landscape and in turn became a vital element in almost every process. All this extra demand on computing power necessitated the rapid development and production of faster processors and co-processors. But as processor clock frequencies and memory sizes get increased, the simple bus inter-connecting all these blocks lagged behind to become one of the weakest links in the computing chain. In order to meet the demand of high-speed digital data processing and achieve high-speed communication between digital front-ends and pc, we implement a transmission system based on PCI-Express protocol.

The first IO buses generation was introduced in the 1980s, including the Industry Standard Architecture (ISA), which enables a bandwidth of 16.7 Mbytes/s. Extended ISA (EISA) and Video Electronics Standards Association (VESA) are other buses of this generation. The second IO buses generation was introduced in 1990s. In 1993 a 32-bit PCI 33 MHz bus was released to deliver a bandwidth of 133 Mbytes/s and a 64-bit PCI bus that delivers a bandwidth of 266 Mbytes/s [1]. However the increase in the processor speeds and the bandwidth needs of newer IO technologies, the PCI bus frequency was increased in 1995 from 33 to 66 MHz, to increase the bandwidth from 133 Mbytes/s to 266 Mbytes/s for a 32-bit PCI, and from 266 Mbytes/s to 533 Mbytes/s for a 64-bit PCI, correspondingly [2]. Several limitations of the PCI 66 MHz bus and the emerging of new high end system technologies that continued demand for higher bandwidths led in 1999 to the release of a new derivation of the PCI called the PCI-X bus. The PCI-X bus has frequencies of 66 and 133 MHz and enables a bandwidth up to 1 Gbytes/s. These frequencies were increased to 266 and 533 MHz in 2002, to increase the bandwidth provided up to 4 Gbytes/s [2]. Another bus system in the second generation is the Accelerated Graphics Port (AGP). However, in order to meet the higher bandwidth requirements and to satisfy the bandwidth hungry devices, a new bus system was still needed.

The latest generation IO bus system is the PCIe. It is evolved from the PCI and overcame the limitations of it. An x1 PCIe bus provides theoretically a bandwidth of 500 Mbytes/s, an x16 PCIe can provide up to 8 Gbytes/s, and a x32 provides 16 Gbytes/s [2]. In this paper, the capabilities of this PCIe bus system are demonstrated by designing and simulating a PCIe based system. This system enables data communication between the CPU through the Root Complex and the Endpoint device. It also offers an overview of the physical and transaction layers of PCI Express and the benefits of PCI Express.

II PCI EXPRESS

PCI Express, the next-generation of the PCI bus, was introduced to overcome the challenges of PCI. A PCI Express topology contains a Bridge and many endpoints (I/O devices) as shown in Figure 1. The switch replaces the multi-drop bus and is used to provide fan-out for the I/O bus. A PCI Express switch provides fan-out capability and enables a series of connectors for add-in, high-performance I/O. A switch provides peer-to-peer communication between different endpoints and their traffic without involving the host bridge provided processes do not involve cache-coherent memory transfers.

The PCI Express bus implementation is similar to a point to point network protocol utilizing dedicated lines, flow control, error detection and re-transmissions. Despite this fact it behaves and interacts with other components as its old versions using a load-share flat-address space memory architecture derived PCI addressing model.

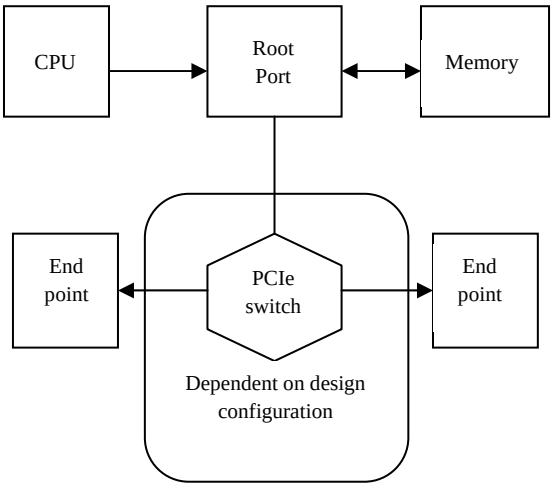

Fig 1 Switched Fabric Serial Interconnect Bus

III PCI EXPRESS ARCHITECTURE

PCIe has a layered architecture as shown in Figure 2. It includes the Transaction Layer, the Data Link Layer and the Physical Layer. On the top of these 3 layers the Software Layer, or device core exists. Each of these layers is further divided into two: transmitter and receiver. The transmitter is responsible for processing the TLPs requested from the device core before transmitting across the PCIe link. The receiver processes the incoming TLPs before sending them to the device core. To demonstrate the functionality of the PCI Express protocol and for the purpose of this paper, 32-bit addressable memory write/read and Completion with Data (CPLD) TLPs will be considered.

The memory write TLP is considered to be a posted transaction where the requester transmits a request TLP to the completer. This in turn does not return a completion TLP back to the requester, unlike the memory read TLP, where the completer is supposed to return a completion TLP back to the requester. The completer returns either a CPLD, if it is able to provide the requested data, or a Completion without data (CPL), if it fails to obtain the requested data. Figure 2 also shows the PCIe TLP. The device core sends to the Transaction Layer the information required to assemble the TLP. This information contains the header and the Data Payload, if exists.

The main functionality of the Transaction Layer is the generation of TLPs to be transmitted across the PCIe link and the reception of TLPs received from the PCIe link. This layer appends a 32-bit End to End Cyclic Redundancy Check (ECRC) to the TLP. These 32 bits are stripped out by the same layer at the receiver side. The Data Link Layer (DLL) is responsible for ensuring a reliable data transport on the PCIe link. The received TLP from the transaction layer is concatenated with a 12-bit sequence ID and a 32-bit Link CRC (LCRC) as shown in Figure 2 [4]. These added bits are stripped out from the incoming TLP by the same layer in the receiving device before being transferred to the Transaction Layer.

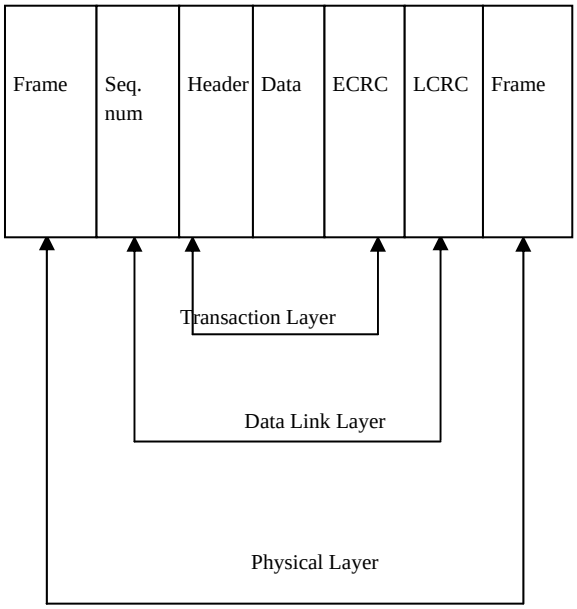

Fig 2 Transaction Layer Packet

The physical layer of a PCIe device is responsible for driving and receiving the Low Voltage Differential Signals (LVDS) at a high speed rate of 2.5 Gbps each way. It interfaces the device to the PCIe fabric. Such an interface is scalable to deliver a higher bandwidth. The TLPs are transferred to this layer for the purpose of transmission across the link. This layer also receives the incoming TLPs from the link and sends them to the Data Link Layer. This layer appends 8-bit Start and End framing characters to the packet before being transmitted. The physical layer of the receiving device in-turn strips out these characters after recognizing the starting and ending of the received packet, and then forwards it to the Data Link Layer. In addition to that, the physical layer of the transmitter issues Physical Layer Packets (PLPs) which are terminated at the physical layer of the receiver, such PLPs are used during the Link Training and Initialization process. In this process the link is automatically configured and initialized for normal operation; no software is involved. During this process the following features are defined: link width, data rate of the link, polarity inversion, lane reversal, bit/symbol lock per lane, and lane-to-lane deskew (in case of multi-lane link) [2].

IV PCI EXPRESS ENDPOINT DESIGN

In this paper, the x1 PCIe Endpoint is considered. In Figure 1, the Endpoint is an intelligent device which acts as a target for downstream TLPs from the CPU through the Root Complex and as an initiator of upstream TLPs to the CPU. This Endpoint generates or responds to Memory Write/Read transactions. When the Endpoint acts as a receiver, the CPU issues a store register command to a memory mapped location in the Endpoint. This is done by having the Root Complex generate a Memory Write TLP with the required memory mapped address in the Endpoint, the payload size (a DW in this design), byte enables and other Header contents. This TLP moves

downstream through the PCIe fabric to the Endpoint. Routing of the TLP in this case is based on the address within its Header. A termination of the transaction takes place when the Endpoint receives the TLP and writes the data to the targeted local register. To read this data back, the CPU issues a load register command from the same memory mapped location in the Endpoint. This is done by having the Root Complex generate a Memory Read TLP with the same memory mapped address and other Header contents. This TLP moves downstream through the PCIe fabric to the Endpoint. Again, routing here is based on the same address within the Header. Once the Endpoint receives this Memory Read TLP, it generates a Completion with Data TLP (CPLD). The Header of this CPLD TLP includes the ID number of the Root Complex, which is used to route this TLP upstream through the fabric to the Root Complex, which in-turn updates the targeted CPU register and terminates the transaction.

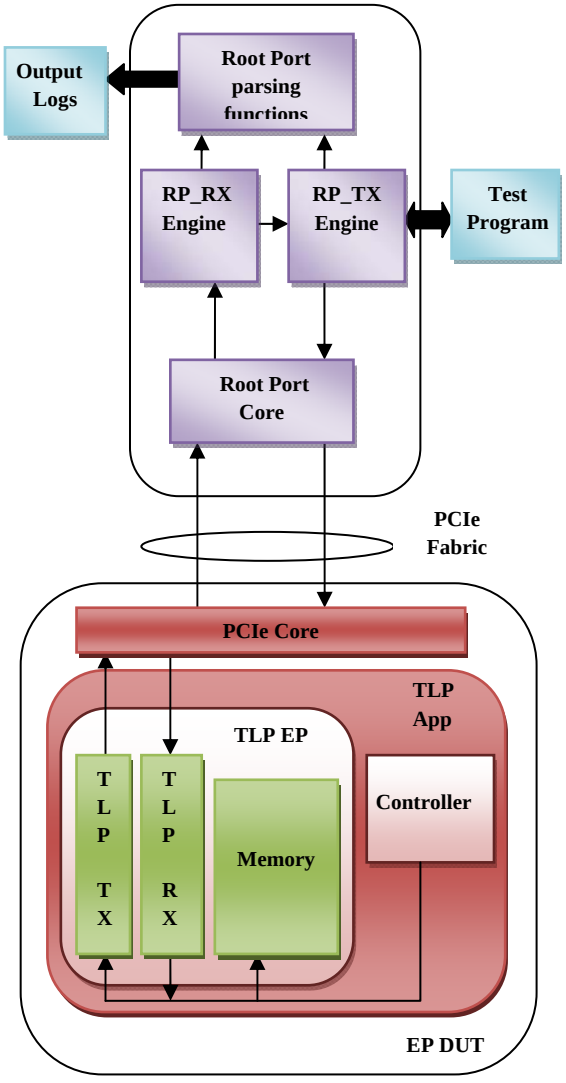

Fig 3 Root Port Model and Top-level Endpoint

The other way around, is to have the Endpoint act as a bus master and initiate a Memory Write TLP to write 1 DW to a location within the system memory. This TLP is routed upstream toward the Root Complex which in turn writes the data to the targeted location in the system memory. If the Endpoint wants to read the data it has written, it generates a Memory Read TLP with the same address. This is steered to the Root complex, which in-turn accesses the system memory, gets the required data and generates a Completion with this data TLP. This CPLD TLP is routed downstream to the Endpoint through the PCIe fabric. The Endpoint receives this TLP, updates its local register and terminates the transaction. Figure 3 shows the layered structure of the PCIe Endpoint device. There are two different solutions for the physical layer (PHY). In the first solution, this layer can be integrated with the other layers in the same chip. Doing so increases the complexity of this chip and provides a higher integration level. This integrated solution has one key advantage when designing using an FPGA. It uses a smaller number of IO pins, which enables easier timing closure. An example of this integrated solution is offered by Xilinx in their newly introduced Xilinx Virtex-6 PCIe Endpoint block [5].

V Running Simulation

The simulation can be run on multiple environments, i.e, XILINX ISE, Mentor Graphics ModelSim, Synopsys VCS, etc. For faster processing and better debugging the VCS platform was the preferred choice of compiler and simulator for this project.

The files needed to perform the simulation are listed in board.f and xilinx_lib_vcs.f, both can be found in the Functional directory under simulation. In case the simulation is being run on VCS, the files listed in xilinx_lib_vcs.f have to be copied from the Xilinx installation directory and placed in the appropriate paths that are being pointed-to from the file list.

Once the global variable files have been placed and the project directory in its entirety is available on the target systems, the simulation task can begin. The testbench file is the board.v which is also located in the same directory as the file lists. The test cases to stimulate and drive the testbench and capture responses in located in the tests folder under simulation. Various test cases can be developed that utilize expectation task to check response and the virtual program to generate and consume TLPs for memory or configuration transactions.

VI Result

The various capabilities of the PCIe bus protocol were demonstrated. The PCIe core was generated, configured and customized using the Xilinx CORE generator. In a modified version of a PCIe Testbench and with the help of the simulation tool Synopsys VCS or ModelSim, the functionality of the designed Endpoint was simulated and verified. Several test cases were conducted to simulate the functionality of this designed Endpoint device.

Fig 4 Simulation Result

VII CONCLUSION

The simulation results clearly show that packet transmission between two devices on the bus is taking place. This has helped form the basis of understanding the core's operation and the intricacies related to packet generation transmission in particular. The Xilinx modules have allowed the implementer of the design to gain a quicker understanding of the device operation and the method of interfacing successfully with the core.

The future potential of this IP Core is evident from the start and was one of the intended purposes stated for this project undertaking. As any future IP design involving a microprocessor or micro-controller requires the utilization of bus, the PCI Express IP Core form Xilinx provides a standard industry ready bus that is easily portable to an FPGA board, thus being practically invaluable. The standard itself may undergo several more iterations before it reaches a performance ceiling. This project has stated the qualities and benefits of this bus and described the practical usage of this core design**.**

REFERENCES

[1] Don Anderson and Tom Shanley, "PCI System Architecture", MINDSHARE INC., 1999.

[2] Don Anderson, Ravi Budruk, and Tom Shanley, "PCI Express System Architecture, MINDSHARE INC., 2004.

[3] Ajay V. Bhatt "Creating a PCI Express Interconnect", Technology and Research Labs, Intel Corporation, 2002.

[4] "PCI Express Base Specification", Revision 3.0, November 10, 2010

[5] "Virtex-6 Integrated Endpoint Block for PCI Express ", User Guide, UG517 (v5.1), September 21, 2010.

[6] Virtex-6 FPGA Integrated Block for PCI Express - User Guide - Xilinx v14.3

[7] LogiCORE IP Virtex-6 FPGA Integrated Block v2.5 for PCI Express, January 18, 2012.