FPGA Design and Implementation of PCIe Target for High Speed Data Transfer to the Devices

Komala M, Dr. Narataj K R, Dr. Mallikarjunaswamy S

***Abstract*— Now a days it is very important to achieve a high speed data transfer and high throughput. A very efficient way to achieve the same is the usage of PCI bus technology. It has been proven that it is a plug and play for the hosts using it. It requires no additional protocols or data integrity as necessary in other protocols. In this paper we have proposed a high density field- programmable gate array (FPGA) based architecture using PCI bus interface for high speed data transfer from the dram for usage of BIG data analytic. The design is incorporated with programmable PCI master core which act as an interface between PCI bus and the internal logic design of FPGA. It also comprises of local FIFO for synchronization. A complete FPGA design has been developed in order to interface FIFO and PCI core. The paper is organized is a way which first give the basic idea and over view of PCI IP core usage and its importance followed by design and logic for developing communication with the bus. Finally the results display the way functionality was achieved and design efficiency is compared with the others.**

***Index Terms*— FPGA, PCI IP Core, FIFO, Xilinx, Bus Interface**

I. INTRODUCTION

The recent development in digital systems has generated various new requirements on the system design engineers. The basic requirement is interface that has high performance and is very much compatible with third party vendors system. The issue like compatibility can be addressed by using a designing a systems with bus interface which has industry standards like ISA, VESA, etc. However, the performance issue was still a very big concern for the system designers. By the introduction of new interface standard named as PCI (Peripheral Component Interconnect) the performance issue was effectively eliminated. The standard was developed to meet all the necessary requirements of new age digital computer system [1]. If we look in to the basic specification of PCI we can find it has got top features like operating speed of 33MHz, 32 bit PCI version can be used for high speed data transfer of 132Mbytes per second as its maximum transfer rate. Not only the compatibility and the performance of the standard make it very useful but also it has got very well documented standard support by special group.

Manuscript received June 29, 2016
Komala M, Research Scholar, JAIN UNIVERSITY, Bengaluru
Dr. Narataj K R, Prof & HOD, Dept. of ECE, SJBIT, Bengaluru
Dr. Mallikarjunaswamy S, Associate Professor, Dept of ECE, SJBIT, Bengaluru

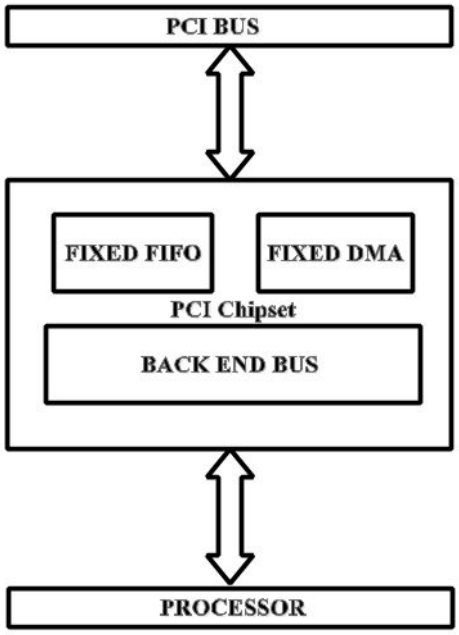

Fig 1 Basic architecture of PCI interface.

II. RELATED WORK

Due to development is technology like big data there is a sudden rise in requirement for high capacity storage systems. These high capacity storage systems generally built based on two methods namely SAN (Storage Area Network) and the other is based on Distributed systems [2].

SAN is designed based on idea of network where there is a need to maintain a server like RAID server and all the storage devices are then connected using a dedicated network. These physical storage networks are Ethernet based using protocol such as iSCSI or FCoE, usage of these software generally makes the system slow by increasing the latency of the overall system. A solution to overcome the problem of latency is the usage of other type of storage mechanism like distributed system. In this technique the storage elements are distributed among the various application hosts and make use of regular network to access them along with distributed system like NFS, Lustre, GFS etc [3]. Another challenging task after reducing the latency is to achieve the high speed data transfer between the memory element and the device in use. This can be done by the usage of high speed dedicated protocol bus named as PCIe.Fig.1 shows the basic architecture of PCI interface

There are enormous design available which are used for the communication between PCI bus and the backend device and the FPGA such as PLX interface chip PCI9054. However we have designed a PCI device on FPGA for the communication with the devices. The design has been built based on standard

32-bit architecture and is capable of communicating with PCI bus protocol and the FPGA internal logic for the specific application.

III. System Architecture

Fig2. Shows the basic block diagram for the proposed PCI top that will be used as a communication medium between the backend devices working on PCI protocol to PCI bus. This block works as a data path controller and medium for transfer of data from the bus to the device and vice-versa [4].

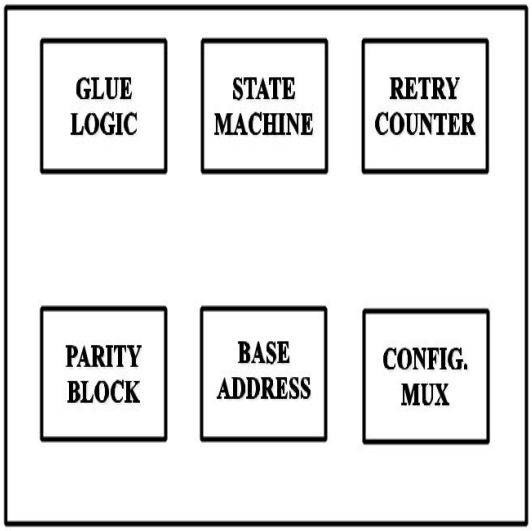

Fig.2. Top level of PCI Target

The architecture is based on 32-bit architecture which works on 33MHz for PCI target reference design. Design initiates the work after PCI initiator sends the signal to start the PCI cycles. It performs various read and write operation with the memory element on the standard bus. PCI target further decodes the address during the address cycle as the base address is hit. An acknowledgement is passed once the cycle is completed. Below we have discussed in detail operation of memory read and write operation performed by the design.

3.1 Memory – I/O Read Cycles

The design developed typically works on a sate machine that can execute single cycle for the read operation and also full length burst read cycle. Certain signals are made high during the operation when the sate machine detects the read command and also finds an address matching to any based address in region [5]. The information is passed to corresponding devices connected to the bus. Once the devices receive command an acknowledgement is pass back to the state machine in form of ready signal to inform sate machine to start the execution. Initially a double word is read and the transaction is completed if the process is related to the single cycle read. The process of reading a double word is continues if the process is a burst read operation and the corresponding device supports the burst operation. If back end device do not support the burst mode of operation is should inform the sate machine by asserting a signal [6]. Once state machine receives this signal it stops providing the burst data and burst read operation mode is terminated. The signal assertion is necessary to be done at-least two clock signal in advance in order to get smooth termination of the operation. Under any unwanted situation corresponding device can raise abort signal to terminate the operation at any instance after the starting of read operation.

3.2 Memory – I/O Write Cycles

This operation is has the same functionality as a read cycle [11]. This also works on a sate machine that can execute single cycle for the write operation and also full length burst write cycle [12]. Certain signals are made high during the operation when the sate machine detects the write command and also finds an address matching to any based address in region. The information is passed to corresponding devices connected to the bus. Once the devices receive command an acknowledgement is pass back to the state machine in form of ready signal to inform sate machine to start the execution. Initially a double word is written and the transaction is completed if the process is related to the single cycle write [7]. The process of writing a double word is continued if the process is a burst write operation and the corresponding device supports the burst operation. If back end device do not support the burst mode of operation is should inform the sate machine by asserting a signal. Once state machine receives this signal it stops providing the burst data and burst write operation mode is terminated [8]. The signal assertion is necessary to be done at-least two clock signal in advance in order to get smooth termination of the operation [9]. Under any unwanted situation corresponding device can raise abort signal to terminate the operation at any instance after the starting of write operation [10].

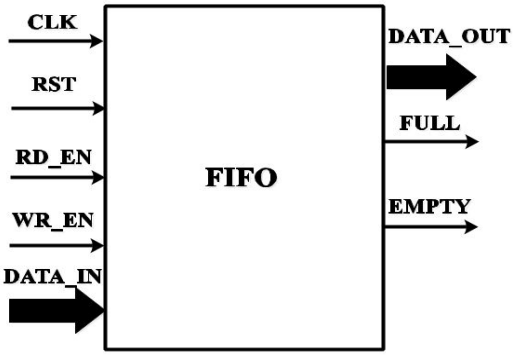

Fig. 3 Functional I/O for FIFO

Apart from this state machine, the design has two FIFOs transmit FIFO and receive FIFO [13]. Fig.3 shows back functional I/O for the designed FIFO [14]. The transmit FIFO temporarily stores the data from CPU and devices which is to be written into the memory through PCI device [15]. And when there is request to write more data than transmit FIFO holds, the FIFO becomes empty and under-run error occurs [16]. The receive FIFO temporarily holds the data read from the memory to CPU or other device through PCI device. Overrun error is associated with this FIFO [17]. The depth of both FIFOs is defined as eight locations of byte wide.

IV. Results and Discussions

Fig.4 shows the RTL schematic for the PCI top that is developed

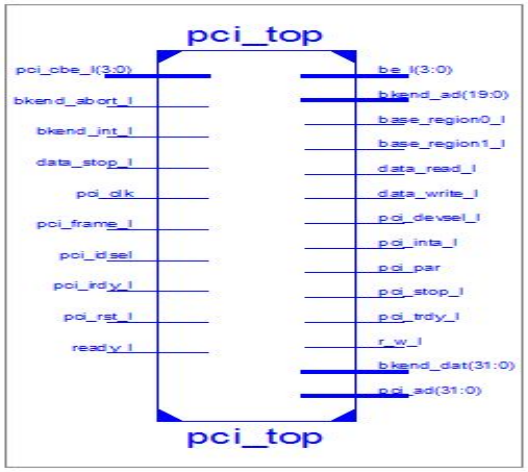

Fig.4 RTL Schematic for the developed PCI top.

The design summary for PCI top is as shown in fig.5

pci_top Project Status (05/13/2016 - 08:16:24)			
Project File:	test_final.xise	Parser Errors:	No Errors
Module Name:	pci_top	Implementation State:	Synthesized
Target Device:	xc3s250e-5tq144	• Errors:	
Product Version:	ISE 13.4	• Warnings:	
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	1	24576	0%
Number of 4 input LUTs	1	49152	0%
Number of bonded IOBs	77	640	12%
Number of GCLKs	1	32	3%

Fig.5 Design Summary for PCI Top.

The simulated waveforms for the read cycle for PCI top is shown in fig.6

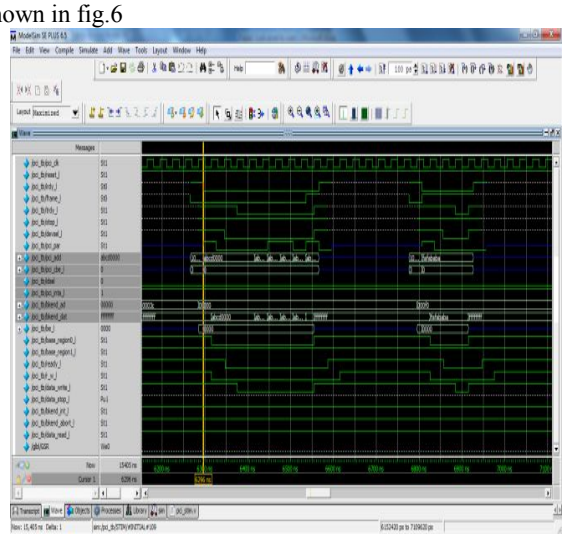

Fig.6 Read cycle for the PCI top.

The write cycle simulation waveform is shown in fig.7 for the PCI top.

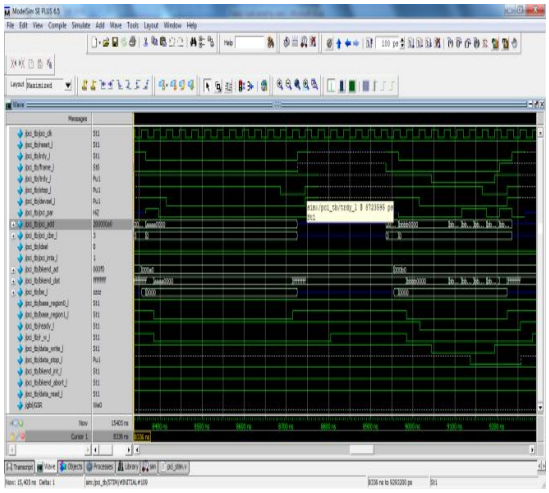

Fig.7 Write cycle for PCI top.

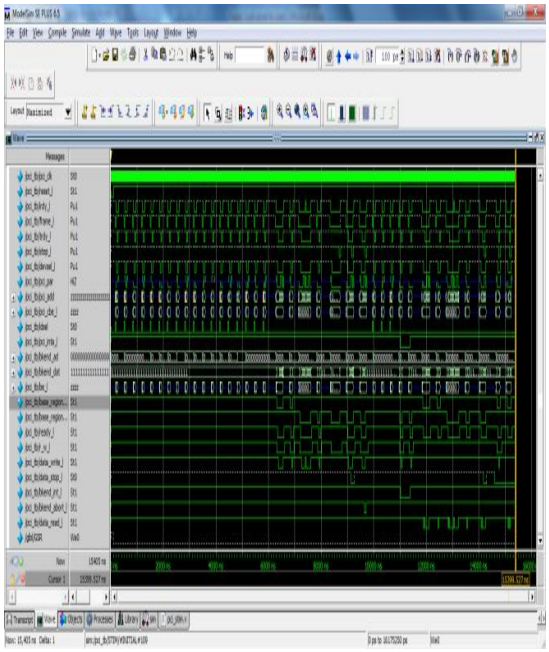

Fig.8 Final Simulation for the design.

The final simulation waveform for the PCIe target is obtained which is shown in fig.8

CONCLUSION

This project has been Implemented keeping in mind the standard used in PCIe target to meet the requirements. The design was developed on Xilixn Spartan 3E device using verilog HDL. Complete functional verification has beed done for various mode of operation and found to work satisfactory as per the design requirements. Little improvement has also been achieved as compared to the previous design implemented on similar device. The design si fully functional and can be used for the any application that requires high speed data transfer using the basic standards of PCIe.

REFERENCES

[1] Ray Bittner "Speedy bus mastering PCI express" 22nd International Conference on Field Programmable Logic and Applications (FPL) Aug. 2012, PP 523 – 526.

[2] Sven Heithecker ; Rolf Ernst" FlexWAFE - A High-end Real-Time Stream Processing Library for FPGAs" 2007 44th ACM/IEEE Design Automation Conference, 4-8 June 2007,PP- 916 – 921.

[3] J. Gong, J. Chen, H. "EPEE: An Efficient PCIe Communication Library with Easy-host-integration Property for FPGA Accelerators" FPGA '14 Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays,PP 255-255.

[4] M. Jacobsen and R. Kastner "RIFFA 2.0: A reusable integration framework for FPGA accelerators" 2013 23rd International Conference on Field programmable Logic and Applications, 2-4 Sept. 2013, PP 1 – 8.

[5] J. A. Jake Wiltgen. "Bus Master Performance Demonstration Reference Design for the Xilinx Endpoint PCI Express S"lutions" September 29,PP 23-24.

[6] H. Kavianipour, S. Muschter, and C. Bohm. High performance FPGAbased DMA interface for PCIe. In RT 2012.

[7] G. Marcus, W. Gao, A. Kugel, and R. Manner. The MPRACE framework: An open source stack for communication with custom FPGA-based accelerators. In SPL 2011.

[8] T. S. Ravi Budruk, Don Anderson. PCI Express System Architecture. Addison Wesley, 75 Arlington St., Suite 300, Boston, MA 02116, 2003.

[9] Y. Thoma, A. Dassatti, and D. Molla. FPGA2: An open source framework for FPGA-GPU PCIe communication. In ReConFig 2013.

[10] K. Vipin, S. Shreejith, D. Gunasekera, S. Fahmy, and N. Kapre. Systemlevel FPGA device driver with high-level synthesis support. In FPT 2013.

[11] Q. Wu, J. Xu, X. Li, and K. Jia. The research and implementation of interfacing based on PCI express. In ICEMI 2009.

[12] C. Hinkelbein, A. Kugel, R. Männer, and M. Müller, "Reconfigurable hardware control software," in RSP '02: Proceedings of the 13th IEEE InternationalWorkshop on Rapid System Prototyping (RSP'02). Washington, DC, USA: IEEE Computer Society, 2002, p. 84.

[13] G. Marcus, G. Lienhart, A. Kugel, and R. Männer, "On buffer management strategies for high performance computing with reconfigurable hardware," Field Programmable Logic and Applications, 2006. FPL '06. International Conference on, pp. 1-6, Aug 2006.

[14] M. Müller, "Evaluation of an fpga and pci bus based readout buffer for the atlas experiment," Ph.D. dissertation, 2004.

[15] Eugin Hyun, Kwang-Su Seong, Desing and Verification for PCI Express Controller. Proceedings of the 4.Third International Conference on Information Technology and Applications, 2005.

[16] Adam Wilen, Justin P, Schade, Ron Thornburg, Introduction to PCI Express: A Hardware and Software Developer's Guide. Intel Press, 2003.