CHAPTER - 4

Design and Implementation of PCIe Interface

Summary

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 chapter a high density (FPGA) based architecture using PCI bus interface for high speed data transfer from the DRAM for usage of BIG data analytics is explained. The design is incorporated with programmable PCI master core which act as an interface between PCI bus and the internal logic design of FPGA.

4.1 Premise

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 the designed system 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. Performance issue got eliminated due to the introduction of PCI. 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 its has got very well documented standard support by special group.
4.2 Related work

Due to development in 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. 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. 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. 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. So, it was very much necessary to design and implement a PCIe interface for high speed data transfer to the devices.

4.3 Basics of PCIe

The Peripheral Component Interconnect - Express (PCI Express) architecture is a high-performance I/O bus used to interconnect peripheral devices in applications such as computing and communication platforms. ISA, EISA, VESA and Micro Channel buses were the buses used in first generation, AGP,PCI and PCI-X belongs to second generation bus. PCIe, here ’e’ means Express, the PCIe works at high, cost is low and consumes low power. Due to this advantages PCIe is used in the design .the PCI Express is serial connection which works as a bus.

The plugged devices are determined by the PCIe when the system is switched ON. A map is created to locate the movement of traffic by identifying devices that are in the link so that link width can be negotiated. There is no requirement for any changes to be made in
the operating system when PCIe has to be interfaced. Two pairs of wires are connected in a lane of PCI, out of which one of the lane if for sending the data and other for receiving the data. Only one bit is transmitted per cycle while moving the data packs. The smallest PCIe has x1 connection has four wires that carries one bit per cycle in every direction, two bits are transmitted by x2 link, and four bits are transmitted through x4 link. Few other configurations of PCIe are x12, x16 and x32. PCI-like card and connectors of different sizes are characterized for PCI Express. In addition, a smaller than usual PCI Express card connector and in addition a PCMCIA-like Express card are characterized.

4.4 System Architecture of PCI

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. However I 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.

Fig 4.1 shows the basic block diagram of the designed 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.

4.1 Basic block diagram of PCIe Interface
Here the PCI device interface and memory interface is designed and implemented on FPGA. The PCI device has six sub-units: GLUE Logic, State machine, Retry counter, Parity Generator, Base Address Generator, and Configuration Mux. The FPGA device is designed with a memory interface module.

4.4.1 Glue Logic

The Glue Logic contains programmable rationale which can be associated with the device pins. This permits the use to take out rationale entryways for straightforward glue logic functions. Inputs can be independently veiled. The yield can be combinatorial produced from the inputs. The simulation results of the glue logic in shown in 4.

4.4.2 FIFO

The design has two FIFOs: transmit FIFO and receive FIFO. The transmit FIFO temporarily stores the data from CPU and devices which is to be written into the memory through PCI device. When there is request to write more data than transmit FIFO holds, the FIFO becomes empty and under-run error occurs. 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.

The depth of both FIFOs is defined as eight locations of byte wide. The primary function of FIFOs is to transiently hold the information amid memory read and write operations. The information to be built into the memory is initially put into the FIFO and afterward after building up responsibility for transport to the memory it is perused from the FIFO and written in the memory.

So, the FIFO also stores the information from the memory and then after transfers when they are prepared for receiving. The operating frequency of PCI interface is faster than compared to SDRAM controller hence to perform synchronization and prevent data loss, FIFO is used in the design. Fig 4. 2 Shows the Functional I/O for FIFO.
4.4.3 State Machine

In common, a State Machine in any device stores the status of a unit at a given time and can operate on the input to change the status and results in an action called output which appears as result after the changes takes place. A finite state machine is one that has a restricted or limited number of conceivable states. A Finite state machine can be utilized both as an advancement instrument for drawing nearer and tackling issues and as a formal method for depicting the answer for later designers and framework maintainers.

In PCI interface design, the state machine which have been planned is a Mealy State Machine. The Mealy Machine is chosen as it has less states, at clock edges the yield changes and speedier response to inputs. Fig 4.3 shows the architecture of the state machine that is designed, the state machine has two main branches, one is for Configuration Mux and other for read/write operation.

There are two types of read/write operation, they are 1.) Single clock cycle read operation 2.) Burst operation. During single clock cycle read operation, at a time it will write only one contents of the data and stops the operation. Whereas, Burst R/W operation is a process either we can continuously take the data or control to produce the data at the output which enables to retrieve the data at high rate compared to other protocols. The other operations are retry, abort, read_write and back off. The retry counter keeps on trying and the data is not obtained the operation gets aborted which is shown by abort state.
Fig 4.3 State Machine of the PCIe interface
4.4.4 Parity Generator

Parity generator is used for detection of error when data is transmitted. A combinational logic circuit which generates a parity bit at the transmitter is called parity generator. When data is subjected to noise there are chances where the bits may get altered, to avoid this parity generator is used. We consider that sum of odd number of 1s is always 1. The sum of even number of 1s is always zero.

4.4.5 Base Address Generator

Base address generator is used for generating data in order to generate the address for that specific address. There are two base addresses used in the design they are 0 and 1 and helps in the control of the retry counter.

4.4.6 Retry Counter

When the data has to be read from the memory, suppose by chance if it is not possible to get the data at the first clock cycle, then the retry counter will try again to read the data in the next clock cycle and every time the counter gets decremented as the number of attempts takes place. A track is kept on the number of attempts that is done between the interface and the SDRAM to obtain the data required. And which is mentioned clearly in the state machine. The retry counter keeps on retrying until it gets some information to go for the next state. Always there is a clock dependency for retry to occur. Retry is not endeavored if the reason for disappointment is such that a retry is not prone to be fruitful.

4.4.7 Configuration Mux

Configuration Mux is basically a unit which is used to perform the operation of configuration of the signals like initialization, always it should be either ‘1’ or ‘0’ which are initiated at the starting of our system. For example, the commands like dts=1, pci-ready=0 are the configuration signals. Other than read write operation in the state machine, there are also other operations linked with the Configuration Mux.
4.5 Memory read and write operations

4.5.1 Memory – I/O Read Cycles

The PCI target state machine underpins boundless length burst reads and single cycle reads. Amid a run of the operation, the state machine figures out whether the location is a hit to one of its base location address. It acknowledges the cycle, and illuminates the back end device of an exchange by attesting BASE_REGION0_L or BASE_REGION1_L. After the back end recognizes the cycle by attesting READY_L, the principal two fold word is read.

In the event that the exchange is a burst, the following two fold word is then read. Once the master begins a burst, if the back end device underpins burst, it must declare DATA_STOP_L to illuminate the objective state machine when it can no more give burst read information. The state machine hopes to see DATA_STOP_L declared, two information stages before the back end must stop the burst read.

This gives the state machine time to stop the cycle, easily. On the off chance that a calamitous disappointment happens, the back end gadget can affirm BKEND_ABORT_L whenever once the read operation starts.

4.5.2 Memory – I/O Write Cycles

The PCI target state machine bolsters boundless length burst writes and single cycle writes. Pretty much as the read cycles worked, amid a common operation the state machine figures out whether the location is a hit to one of its base location address.

On the off chance that the objective acknowledges the cycle, the state machine illuminates the back end device of an exchange by declaring BASE_REGION0_L or BASE_REGION1_L. The back end device recognizes the cycle by stating READY_L. The PCI target will state PCI_TRDY_L, and if the initiator has PCI_IRDY_L declared, an information write will happen.

In the event that the exchange is a burst, the following two fold word can be written at the following clock edge. Once the initiator begins a burst, if the back end device bolsters
burst, the back end must declare DATA_STOP_L to illuminate the objective state machine. When it can no more acknowledge burst write information. The state machine hopes to see DATA_STOP_L affirmed, two information stages before the back end must end the burst composes.

This gives the state machine time to stop the cycle, easily. On the off chance that a calamitous disappointment happens, the back end can declare BKEND_ABORT_L whenever once the write operation starts.

The architecture is based on a standard of 33MHz. The PCI design architecture is a 32-bit architecture. The 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 standard bus.

The 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. Fig 4.4 shows the top level module of the PCIe interface.
4.6 Results of PCIe interface

Figure 4.5 shows the design summary for the PCIe interface from which it is observed that number of slices utilized is 1 and LUTs are 1 out of 24576 and 49152 respectively.

![Device Utilization Summary](image1)

![Top level module of PCIe Interface](image2)
Fig 4.7 RTL Internal architecture of PCIe Interface

Fig 4.8 Timing summary of PCIe Interface
Fig 4.9 Simulation results of Glue Logic

Fig 4.10 Simulation Results for Abort operation
Fig 4.11 Simulation results for Retry Counter

Fig 4.12 Simulation results for Parity Generator
Fig 4.13 Simulation results for Base Address Generation

Fig 4.14 Simulation Results for Configuration Mux
4.15 Simulation results for PCI configuration

Fig 4.16 PCI Read operation
Fig 4.17 Simulation results for PCI Write operation

Fig 4.18 Final PCI Top Simulation results
4.6 Conclusion of PCIe interface

The standard was produced to meet all the vital necessities of new age advanced PC framework. If we look into the basic specification of PCI we can find it has got top features like operating speed of 33MHz, 32 bit PCI version can be use 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 its has got very well documented standard support by special group.

The design has been implemented keeping in mind the standard used in PCIe target to meet the requirements. The design was developed on Xilinx Spartan 3E device using Verilog HDL. Complete functional verification has been 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 is fully functional and can be used for any application that requires high speed data transfer using the basic standards of PCIe.