Zynq SoC Based High Speed Data Transfer Using PCIe: A Device Driver Based Approach

Pramod Kumar Tanwar[1], Om Prakash Thakur[1], Kritik Bhimani[2], Gaurav Purohit[1], Vipin Kumar[1], Sanjay Singh[1], Kota Solomon Raju[1]
[1]Cyber Physical Systems, CSIR-CEERI, Pilani, India
[2]BITS-Pilani, Goa Campus, Goa, India

Idaku Ishii, Sushil Raut
Department of System Cybernetics
Hiroshima University
Hiroshima, Japan

Abstract— **High Speed Data Transfers is a typical requirement of data intensive applications like image and video processing. Speed efficiency can be ensured by handling the data transfers at both Hardware and Software level. A complete system has been developed by implementing the hardware architecture on FPGA and writing corresponding Software device driver to perform the speedy data transfers from endpoint to a root complex device using PCIe interface. This paper describes the approach to design and verify this system. The speed of data transfer achieved practically for PCIe (2.0) x4 is 4Gib/s. The developed hardware architecture is resource constrained and low power. The hardware is implemented on Xilinx Zynq device. This work has a good potential in the field of image and video processing and can be used to perform large data transfer operations at high speed.**

Keywords— PCIe, Zynq, XMD, High Speed, DMA

I. INTRODUCTION

Image and video processing has revolutionized the world with its high performance digital cameras having flexible interface to achieve high throughput. The camera captured images are put to hardware accelerators to perform filtering, processing and displaying operations. There are huge number of computations performed on the captured images using FPGA boards. FPGA devices are used to create reconfigurable hardware architectures to perform the above said operations at high speed. After applying the image and video processing algorithms on the captured images, these images are sent to Central Processing Unit (CPU) for displaying and further processing. In this paper, a hardware architecture is developed in Xilinx Vivado using IPs to send the processed images data from the FPGA board to CPU with very low latency. Peripheral Component Interconnect Express (PCIe) is used here as the high speed serial interface to create the communication link between FPGA board and the CPU. To create and manage the PCIe interface, a software device driver is written on Linux Ubuntu OS which performs the data transfer operations smoothly and speedily [1].

For a typical high speed vision requirement, frames are captured at more than 250 frames/sec, pre-processed and sent to CPU for further processing and usage. A high speed frame grabber which captures image at more than 250 frames/sec with resolution of (512*512 pixels), needs a high speed (Gib/s) interface through which the data could be sent to the host system for further processing. In order to fulfill the high throughput need of high-speed digital data processing and to achieve high-speed communication between digital front-ends and computer, PCIe is the best suitable interface now a days [2]. This interface has a number of versions and lanes which could be used as per the application requirement. In this paper, PCIe (2.0) x4 is used to analyze the capability and efficiency of this interface. The developed hardware architecture is having Zynq SoC, which has abundant logic cells and dual hard core ARM Cortex A9 processors to make the complex decisions based on the arrived data from the peripherals like high speed camera. The Zynq SoC is also having the Programmable Logic (FPGA) and is capable in processing large data, applying massively parallel algorithms and performing speedy computations. Fig. 1 describes the interaction between PCIe endpoint and root complex CPU. Xilinx ZC706 board having Zynq-7000 all programmable SoC (Z-7045) is used as a PCIe endpoint.

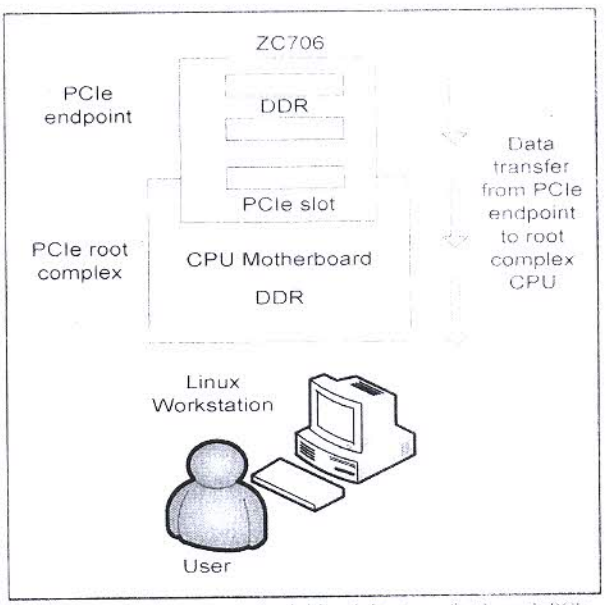

Fig. 1 Introduction to endpoint initiated data transfer through PCIe

ZC706 has PCIe (2.0) x4 interface. 1 GB DDR3 for processing systems (PS) and another 1 GB for Programmable Logic (PL). The hardware design is made using reconfigurable

hardware available in the PL part of this board. The developed hardware creates an interface between the endpoint and root complex CPU. To manage the data transfer operation, a software device driver is written on Linux core which could be run in online mode. The implemented software driver can work on all the Zynq SoC boards by modifying the configurations [3].

In the remaining sections of this paper, hardware design and software driver development are discussed. The developed hardware and software are put on the FPGA board and CPU, it performs the required data transfer operation and verifies the data buffers.

II. HARDWARE DESIGN DEVELOPMENT

Hardware design is realized using Xilinx Vivado tool and implemented on Xilinx Zynq-7000 SoC. The main IPs which are used in this project are: 1. AXI CDMA, 2. AXI memory mapped to PCI express, 3. Zynq-7000 processing system, 4. AXI BRAM generator, 5. Block memory generator. Other small IPs like binary counter, AXI interconnect, processing system reset, clock buffers and other glue logic IPs are also used in this design. The goal is to transfer data packets from ZC706 board DDR3 memory to Intel-i7 CPU using PCIe bus. PCIe is more like a network where different endpoints are connected to a switch or bridge and the switch or bridge is connected to root complex using dedicated paths. In the presented work, endpoint is directly connected to the root complex device. The endpoint which initiates the transaction is called requester and the responder is called completer. Here, the ZC706 Board is configured as endpoint and the CPU board is functional as root complex. When the PCIe device is connected to CPU then the configuration address spaces are filled. The PCIe device has three PCIEBARs (base address register) but only one is required here, which is used by the CPU to communicate with the ZC706 FPGA board. The source and destination addresses are of different bits (32 bits in ZC706 DDR and 64 bits in Intel-i7 CPU address space). It is necessary to use an address translator *i.e.* AXI memory mapped to PCI express block. To store the translation vectors, one Block Memory (BRAM) is needed and the controller of this memory is AXI BRAM controller [4]. Vivado 2015.1 is used to create the Zynq based hardware architecture.

Zynq processing system is required to execute the instructions. DDR controller sends the data buffers from Zynq DDR to CPU DDR. CDMA block manages data transfer and CPU usage. The remaining blocks like Processor system reset, counter, AXI Interconnect *etc.* are supporting IP blocks which are used to generate clocks, resets and interconnections between IP blocks. Some logical gates are also used to bring down the frequency of reference signals to act as debug monitors to check the proper working of the developed design on the hardware board. An implemented design has a low visibility to be looked into for debugging. A designer has to connect a JTAG debugger and uses Chipscope or other similar In Circuit Debugger (ICD) to dwell into the FPGA design once it is configured. We have used an old-school technique of LEDs is used to check the clocks and basic handshaking. Some user LEDs on the board are attached to show the clock and link up. Link up is the basic step towards the handshaking of PCIe interface. The block diagram in Fig. 2 shows the main IPs used to perform data transfer operations speedily. The Zynq-7000 processing system, AXI CDMA and AXI PCIe IPs configures each other. The Block memory generator IP is accessed by Zynq, AXI PCIe and AXI CDMA IPs to get the translated addresses during transfer operations. The DDR is interfaced with Zynq-7000 and data are accessed from it through its slave ports. The accessed data are put on the PCIe link after encoding and parallel to serial conversion. The PCIe link transfers the data on the CPU side and after serial to parallel conversion and decoding, the data buffers are received and displayed on the CPU.

Data is transferred in the form of standard packets. A data transfer packet consists of overheads which decreases the efficiency of transfer but increases the reliability. The data transfer packet looks like as shown in Fig. 3 [5]. PCIe protocol ensures the transfer of data payload from one device to other reliably.

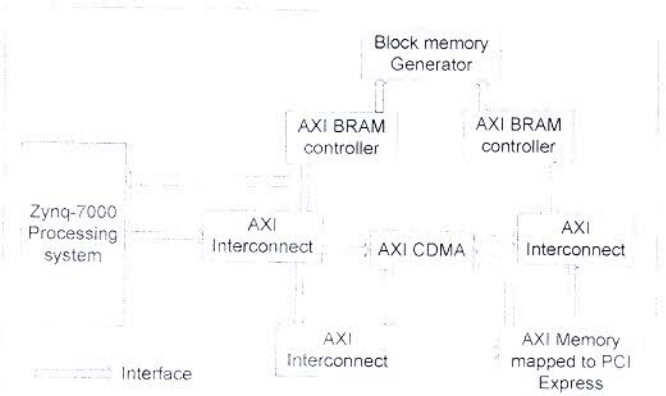

Fig. 2 Hardware design to transfer data from Zynq DDR to CPU DDR

For large data transfers, the whole data is divided into packets and sent in sequence, the error cyclic redundancy check (ECRC) and linear CRC are appended to the packet in transaction and data link layers. In the physical layer, start and stop bits are added and then 8/10b encoding is done. Data is sent bit by bit after conversion from parallel to serial [6].

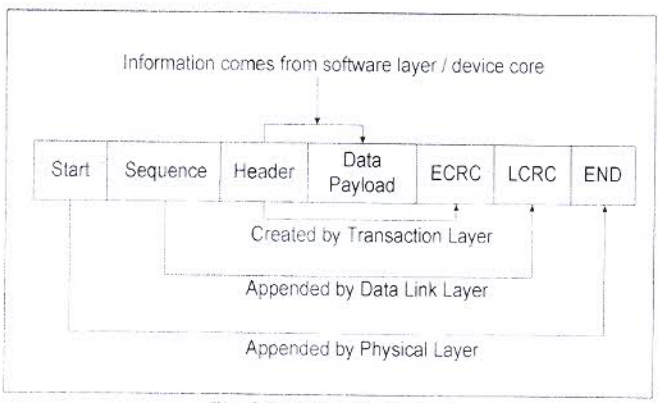

Fig. 3 Data Transfer packet