The Hardware System of Online-teaching Database System Based on FPGA and Multi-core DSP

Yaqin Li[1, a *], Cao Yuan[1, b] and Cheng Gong[1, c]

[1]School of Mathematic & Computer Science. Wuhan Polytechnic University, Wuhan China

[a]leeyaqin.li@gmail.com, [b]yuancao1980@gmail.com, [c]863787503@qq.com

Keywords: Online-teaching; PCI; FPGA; DSP.

Abstract. These online-teaching database systems are generated from online transactions, emails, videos, audios, images. They are stored in databases grow massively and become difficult to capture, form, store, manage, share, analyze and visualize via typical database software tools. The adventure of large amount of data and real-time requirements presents great challenges using traditional desktop computers. In this paper, to tackle the specific problem with a critical requirement of 10ms for whole data processing pipeline, we proposed a high performance embedded system as opposed to personal computer. Peripheral Component Interconnect (PCI) for data transferring from computer to Field Programmable Gate Array (FPGA) is proposed to solve the problem, and a custom-designed dual-port dual-channel RAM to realize simultaneous data exchange and data processing which was implemented using a 6-core Digital Signal Processor (DSP). The hardware system was designed and tested the performance of individual modules as well as the integration of them as a whole. We reported a total time using such pipeline of 7.5ms, meeting the critical requirement and demonstrating its feasibility in practical application.

Introduction

Especially in science big data, it is hard for scientists to analysis big data remotely because the consumed time for read/write process of big data is very big. Therefore, scientists request data center to move big data near to computing farm in their favor data center first. And then, analyze them with thousands CPUs in high speed local networking environment. One of big science data is the data from large hadron collider such as LHC in Swiss CERN. CERN generates multi-peta byte data per year. A peta-byte LHC data analysis needs approximately 3000 CPUs. For the analysis of big science data, thousands of CPUs are usually required for the analysis. Therefore, science big data center has to utilize Grid computing technology that aggregates thousands of CPUs scattered in the data center and orchestrates them to work as if they are single supercomputer. This Grid computing causes increases traffic in the most part of data center networks. The mutual interference between the Grid computing traffic generated by thousands of CPUs and the big data transfer traffic for local archiving make the local network of the data center network to be congested frequently. Therefore, the capability of data analysis in the data center drops rapidly. In order to solve these problems, data centers competitively raise the network bandwidth of data center or enhance QoS function in networking devices. But, these approaches have a limit because the both approach of raising the bandwidth and improving the QoS function require very high expenditure and because the expenditure increases exponentially according to the increase of the volume of big data. Thus, this paper suggests the method for the extension of big data networking capability of data center by separating big data traffic from the ordinary data center traffic. The feature of our separating method is basically based on the concept that the separating of the traffic at the lowest end of network architecture is more convenient and economic than the traffic separating at highest end of network architecture under the big science data computing environment.

These library Big data and its analysis are at the center of modern science and business. These data are generated from online transactions, emails, videos, audios, images, science data, sensors and their applications [1]. The major challenges include how we approach these large volume of data with various types, sources and high velocity and so on. Traditional solutions includes using

powerful processors, large amount of memory, intelligent algorithms and clever software on personal computer, which is economically expensive and computationally inefficient . In contrast, embedded system has been playing a more and more important role in tackling such tasks, for example, digital communication, medical device, electric commerce and so on, where data is highly digitalized and high performance data processing systems are in great demand. Such alternative solutions feature application specific, parallel computing, real-time performance, low power requirements and low cost [2]. In general, embedded systems gain more and more prevalence due to three main features: embed ability, specificity, computer, et. al. These features enable us to design highly integrated, application specific, computationally efficient and low cost data processing systems.

In the paper, we propose to tackle this critical problem by building a low cost while high performance embedded system which consists of Peripheral Component Interconnect (PCI), Field Programmable Gate Array (FPGA), and high performance multi-core Digital Signal Processor (DSP). In particular, we design our system as below: image data from camera/probe is buffered to FPGA via PCI, and then transmitted to DSP through custom-designed dual-port dual-channel Random-access memory (RAM), the multi-core DSP performs high performance parallel computation and return the results to the host computer through FPGA using external memory interface (EMIF). The EMIF module in DSP is implemented using its internal resource, while we realize such function by building a corresponding module using Verilog in the FPGA. We designed the hardware and tested the performance of the included individual modules and all of them as a whole. We reported results of 3.2ms for data transmitting from FPGA to DSP and 4.1ms for parallel computation implemented in the multi-core DSP, the resultant total time of data transmitting and processing was 7.5ms, meeting the critical requirement of 10ms/100Hz and outperforms that implemented on personal computer (PC).

FPGA

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing—hence "field-programmable". The FPGA configuration is generally specified using a hardware description language(HDL), similar to that used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare).

Contemporary FPGAs have large resources of logic gates and RAM blocks to implement complex digital computations. As FPGA designs employ very fast I/Os and bidirectional data buses it becomes a challenge to verify correct timing of valid data within setup time and hold time. Floor planning enables resources allocation within FPGA to meet these time constraints [3].FPGAs can be used to implement any logical function that an ASIC could perform. The ability to update the functionality after shipping, partial re-configuration of a portion of the design and the low non-recurring engineering costs relative to an ASIC design (notwithstanding the generally higher unit cost), offer advantages for many applications [4].

FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together"—somewhat like many (changeable) logic gates that can be inter-wired in (many) different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like ANDand XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

To simplify the design of complex systems in FPGAs, there exist libraries of predefined complex functions and circuits that have been tested and optimized to speed up the design process. Other predefined circuits are available from developer communities such as OpenCores (typically released under free and open source licenses such as the GPL, BSD or similar license), and other sources.

In a typical design flow, an FPGA application developer will simulate the design at multiple stages throughout the design process. Initially the RTL description in VHDL or Verilog is simulated by creating test benches to simulate the system and observe results. Then, after the synthesis engine

has mapped the design to a netlist, the netlist is translated to a gate level description where simulation is repeated to confirm the synthesis proceeded without errors [5]. Finally the design is laid out in the FPGA at which point propagation delays can be added and the simulation run again with these values back-annotated onto the net list.

Methods

Recently a number of Multi-core architectures have been proposed for the streaming DSP application domain. Examples are: Silicon Hive (an incubator of Philips Research); the PACT-XPP; the Maya and Pleiades chips fromBerkeley; and the Chameleon/Montium architecture from the University of Twente and Recore Systems. For an overview we refer to.A multi-core architecture has a number of advantages:

It is a future-proof architecture, as the processing cores do not grow in complexity with technology. Instead, as technology scales, simply the number of cores on the chip grows.

A multi-core organization can contribute to the energyefficiency of a SoC. The best energy savings can be obtained by simply switching off cores that are not used, which also helps for reducing the static power consumption. Furthermore, the processing of local data in small autonomous cores abides by the locality of reference principle. Moreover, a core processor might not need to run at full clock speed to achieve the required QoS at a particular moment in time.

When one of the cores is discovered to be defect (either due to a manufacturing fault or discovered atoperating-time by the built-in-diagnosis) this defective core can be switched-off and isolated from the rest of the design [6].

A multi-core approach also eases verification of an integrated circuit design, since the design of identical cores only has to be verified once. The design of a single core is relatively simple and therefore a lot of 37effort can be put in (area/power) optimizations on the physical level of integrated circuit design.

The computational power of a multi-core architecture scales linearly with the number of cores. The more cores there are on a chip, the more computations can be done in parallel (providing that the network capacity scales with the number of cores and there is sufficient work).

Although cores operate together in a complex system,an individual tile operates quite autonomously. In a multi-core architecture every processing core is configured independently. In fact, a core is a natural unit of partial reconfiguration. Unused cores can be con-figured for a new task, while at the same time other cores continue performing their tasks. That is to say, a multi-core architecture can be reconfigured dynamically

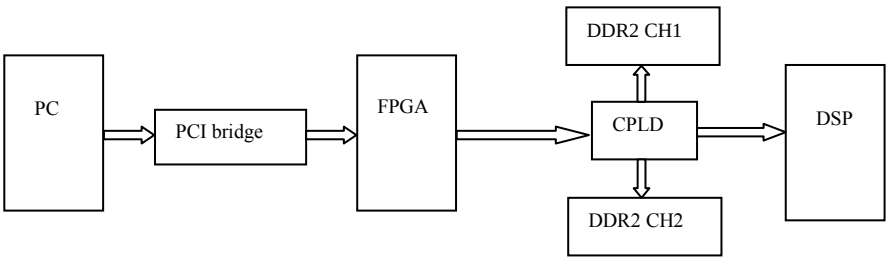

Figure 1. System structure

The main structure of this high performance system can be described as below: (1) image data received by PC from camera/probe will be fed into the First-In First-out (FIFO) designed in FPGA through PCI bridge. (2) FPGA then signals the following Complex programmable logic device (CPLD) to change the dual-channel switch status so that both the FPGA and DSP can access different channel DDR2 at the same time, for example FPGA writes data to channel 2 DDR2 while DSP reads data from channel 1 DDR2. The dynamic connection relationship allows for simultaneous reading and writing operation that initiated by different hosts, resulting in fast speed data transferring between FPGA and DSP. (3) DSP allocates received image data equally to its

internal 6 cores to perform parallel computation. (4) Computation results from DSP are then sent back to FPGA through HPI. Fig. 1 shows the framework of this system [7].

From left to right, PC transfers image data to FPGA using PCI bridge. FPGA buffers its received data to DSP through a custom-built dual-port dual-channel RAM.

Experiments

With the hardware framework introduced above, we conduct experiments and evaluate the performance of individual modules and then all of them as a whole. We evaluated data transfer rate between PC and FPGA, and parallel computation within the multi-core DSP. Due to the efficient dual-port dual-channel RAM scheme, the data exchange time between FPGA and DSP can be ignored [8].

We developed the PCI driver such that our PC can identify our custom-built card with a PCI9056 as the bridge mounted. The PCI driver was developed under Visual C++ 6.0 using Driver Development Toolkit and DriverStudio. The user application was developed to communicate with the driver to pass user command to and receive response from the kernel. We tested data transferring from PC to FPGA and obtained the below data size and transferring rate relationship. We obtained a date transferring rate of approximately 80Mbytes/s when transferring 1000-bytes data from PC to FPGA (3.2ms). DMA mode was not used in this preliminary study, and the performance will be improved due to its advance over the Direct Transfer mode [9].

We evaluated the performance of data transferring from FPGA to DSP by buffering synthetic data to one channel DDR2. We designed a data-generation module which was interfaced to the DDR2 IP core and the simulated signal sequences agreed well with the standard DDR2 signal sequences as is shown in figure. The time required to transferring a 512x512byte image is approximately 0.16ms. Fig. 2 below depicts the signal sequence generated by FPGA using Quartus II 9.0. By comparing the generated signal sequences and that required to operate the aforementioned DDR2s, we found that they match quite well, indicating the success in data exchange between the FPGA and DSP.

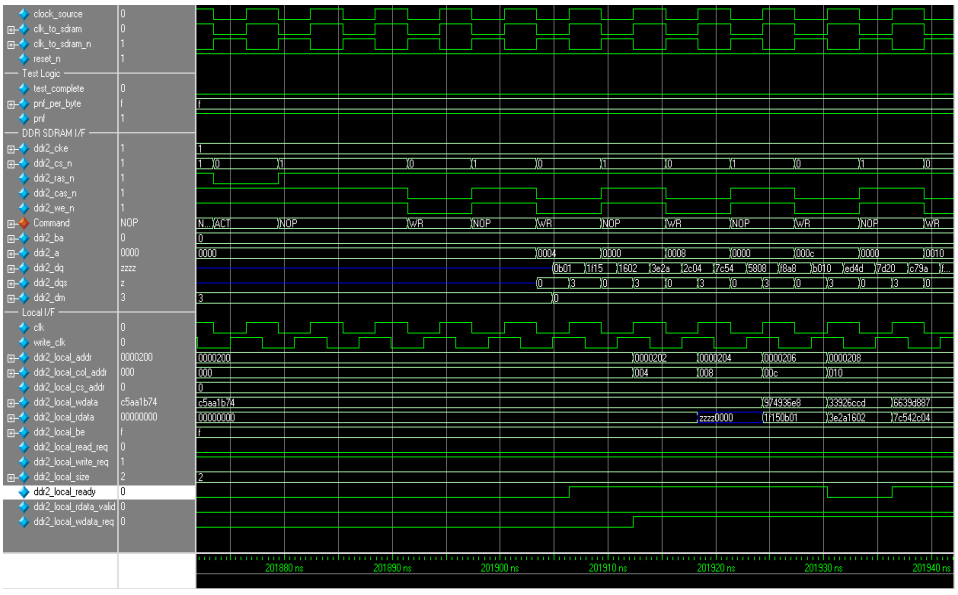

Figure 2. Signal sequences generated by FPGA module to operate following DDR2s

High performance image processing in multi-core DSP.Image data received by the multi-core DSP was used to register to the pre-embedded reference image. The registration involves block-matching technique followed by an affine transformation [10]. The block size was set to 4 voxels and the best correspondence between the source block and the target block was determined using the Normalized correlation coefficient (NCC). The corresponding set of matched points were then used to find the best affine transformation. A coarse-to-fine scheme was used where the images

were first down sampled and the pervious transformation was used to transform all the points, and finally the registration was performed on full resolution images as described. The comparison of implementations on different platforms are shown in Table 1. The 6-core DSP performed the computation intensive task in parallel way and achieved a 4.1ms, outperforming both CPU and the DSP using single core.

Table 1 Comparison of computation efficient of the same block matching registration implementations on CPU, single-core DSP and the 6-core DSP.

Computation time(ms)	CPU	Single-core DSP	6-core DSP
Block matching registration	45.2	12.4	4.1

Summary

In this study, we addressed the image processing problem with critical real-time requirements using a custom-built embedded system that includes PCI communication between PC and FPGA, a custom-built dual-channel DDR2 as the buffer between FPGA and multi-core DSP which was used to perform the intensive computation involved in the specific image registration problem in parallel. To the best of our knowledge, we are the first to propose to solve such problems using such an embedded system with novel structure such as high performance FPGA-DSP framework, efficient dual-channel DDR2 and multi-core DSP for parallel computation.

Using our 512x512bytes image with a frame rate of 100Hz, we obtained a 3.2ms for data transferring from PC to FPGA, 0.16ms for transferring the received data from FPGA to DSP and 4.1ms for parallel image processing in the multi-core DSP. In total, we achieved a 7.5ms for such an image transmitting and processing pipeline, meeting the critical requirement of 10ms for such task. We also compared the parallel image processing performance using multi-core DSP with that on the same DSP in single-core mode and a CPU, the results demonstrated the advantage of multi-core DSP in computation over either CPU or single-core DSP.

Acknowledgements

This work and the paper are being supported by the Natural Science Foundation of China (Grant No. 61502362).

References

[1] Grosheide, F.W., Copyright issues and the information society: Dutch perspectives. Electronic Journal of Comparative Law (EJCL), 2002. 6(4).

[2] Cukier, K. and V. Mayer-Schoenberger, Rise of Big Data: How it's Changing the Way We Thinkabout the World, The. 2013.

[3] Derakhshan, R., M.E. Orlowska, and X. Li. RFID data management: challenges and opportunities. in RFID, 2007. IEEE International Conference on. 2007. IEEE.

[4] Labrinidis, A. and H. Jagadish, Challenges and opportunities with big data. Proceedings of the VLDB Endowment, 2012. 5(12): p. 2032-2033.

[5] Hamilton, M., et al., A System for Real-time Parallel Scientific Computation and Visualization using the IBM Blue Gene/L Supercomputer.

[6] Cotofana, S., et al. Embedded processors: Characteristics and trends. in Proceedings of the 2001 ASCI Conference. 2001. Citeseer.

[7] Digital Media Processing. Embedded Systems. 2010.

[8] Fletcher, B.H. FPGA embedded processors. in Embedded Systems Conference. 2005.

[9] Maxfield, Clive (2004). The Design Warior's Guide to FPGAs: Devices, Tools and Flows. Elsevier. p. 4.ISBN 978-0-7506-7604-5.

[10] Xilinx. The role of distributed arithmetic in FPGA-based signal processing. Xilinx Application Note,2009,http://www.xilinx.comJ appnotesltheory 1.pdf