1 Firt slide

Dear professors, students, lady's and gentleman. My name is Piotr Zdunek and I would like to present you my master thesis which is a design of a High speed, multichannel camera with a 10 GbE interface.

The thesis is being prepared under a supervision of Phd. Grzegorz Kasprowicz, at the Institute of Electronics Systems, in Photonics and Web Engineering group.

2 Agenda

I would like to start my presentation with an agenda. Firstly I will make a brief introduction I say a little bit about cameras, then I will talk about my thesis, it's goals and it's genesis.

After that I will present a concept of my design. The assumptions that I have made as well as the requirements.

Next part is the realisation itself, I will explain what I have done till now, how I have done it, what I have not done, what are the outcomes and failures that I have had during the development.

In the end I will summarize my work and add some information about the possibilities of further development of my project.

3 Introduction - Cameras

Let's begin with a brief introduction about the cameras. Camera is an electromechanical device used for capturing images or videos.

It consists of few main parts: optics, sensor, processing system and interface.

All of those parts are necessary for a camera to work. A special kind of cameras are scientific cameras used for astronomical or medical purposes. Their design is much more complex than a design of a typical camera due to specific requirements like sensitivity or high speed operation.

4 Introduction - camera firmware design

The purpose of my thesis is to address the issue of the scientific camera design. One cannot deny the fact that there is no open framework or base for such systems. All designs are custom and what is needed on the market is a system which would allow for a faster time-to-market, less error prone development and high throughput connectivity. Also an easy way of adding a support for different sensors is essential.

4.1 Genesis and goals

The goal of my thesis is to design a firmware for a scientific camera with the following requirements:

- high processing performance - for support of high resolutions
- ease of adding a support for a different sensor

- high speed communication - to send high amounts of data live
- multichannel operation - astronomical as well as medical applications require it

5 Concept of design

So where does one start the realization of a project with the before-mentioned requirements? As engineering practise shows the best way to start is from breaking down the system into multiple smaller chunks and choosing the simplest solution which fulfils all or most of the requirements.

Firstly I have prepared a base block diagram of my design. Which can be seen in the slide.

My design consists of:

- sensor acquisition block [IP (Intellectual Property - digital system in FPGA)]
- processing system
- high speed interface

(Picture of simple system)

Okay, this is a joke of course. This is the actual breakdown structure of my design. (Very complicated picture)

After breaking down the design I have made a few key design choices. I have decided to use a modern System on a Chip IC - Xilinx Zynq SoC as a heart of the system. This device incorporates an FPGA and two core Cortex A9 chip. It's the most versatile and cost-efficient device that was available on the market when the project started. It provides:

- high speed connectivity - using GTX transceivers - dedicated hardware (up to 12.5 Gbps per link, number of links depending on the device size)
- versatility - FPGA fabric allows for connecting to any sensor (sometimes a dedicated AFE is needed)
- high processing capability - two core Cortex A9

As an sensor I have chosen CMV4000 from CMOSIS company. It is a high speed 4 MP chip which can run at 100 fps generating 400 MBps of raw data.

For the connectivity I have chosen few interfaces at the beginning: 1 GbE, 10 GbE, SATA and PCIe. During the design the decision of which interface to use was done.

6 Realization

As for the realization of the design itself. I would like to start with presenting you the hardware that I am using. The test stand looks like this (picture). I use ZC706 development board with Zynq 7000 Z7045 in FFG900 package. The sensor is placed on a dedicated FMC card connected to the development board. And as you can see in the picture I have tested the SATA interface (cause the discs are connected) as well as Ethernet connection via SFP transceiver (1 Gbps or 10 Gbps Ethernet).

I think that the best way to describe how does my system work it to start from the sensor and then go all the way to the gigabit interface.

6.1 Sensor acquisition

As for the sensor acquisition, the data from the sensor are being send to the FPGA fabric of Zynq via multiple serial links. As you may know nowadays strictly parallel buses are not used because of the throughput limitations. So in order to send such an amount of data from the sensor multiple serial links are used each sending raw data (without encoding) with 400 MHz frequency DDR (data are latched on the both edges of the clock). The downside of serial links is that the data needs to be deserialized when acquired. And this is done in an FPGA and send to the DMA IP Core. Another IP Core tightly coupled with data acquisition is driving the sensor control signals. The control is register based, with registers set via software.

6.2 Internal data handling

DMA is responsible for transferring of the data from the sensor to the DDR3 memory and optionally to the processing system.

This way the data can be buffered in order to send them through dedicated 1 GbE Hardware IP available in Zynq or e.g. PCIe IP Core given by Xilinx.

In case we want to use a 10 GbE interface there is no need for DMA, because the translation would have to be made. When 10 GbE interface is needed data are being send directly to this IP Core without the use of DMA.

So at this point we know how are the data acquired form the sensor.

6.3 Connectivity

As for the connectivity the data from the sensor can be send directly via 1 GbE Ethernet using dedicated hardware in Zynq. This solution provides only 1 Gbps of throughput for the Ethernet, but is very well supported by the software (drivers are available).

I have also tried using SATA interface from Open Cores Repository (opencores.org), but unfortunately this IP core behaves in a very unstable way and had to be discarded.

The solution that I have chosen, that would provide 10 Gbps of throughput is to use a dedicated FADE protocol IP, which allows for simple FPGA-PC 10 Gbps Ethernet-based connectivity. This IP core is also available on the opencores. It was designed by PhD. Wojciech Zabolotny and is extensively used at this moment.

I can only add that PCIe also can be used in this design, but it needs further development.

I would like to move further and describe the processing and control.

6.4 Processing system

As far as processing and control is concerned - Cortex A9 is used for this purpose. Because it's a two core chip, the designer (that's me) can take advantage of asymmetric multiprocessing. A scenario in which two cores are running different operating systems in parallel. This approach eases the development, due to the fact that Linux can easily provide high system level features like TCP/IP stack whereas RTOS is running in a deterministic way and eases the development of sometimes complex sensor IP control and interrupt handling.

In my design Linux is responsible for control of camera operation as well as data processing (optional), and FreeRTOS is responsible for sensor data acquisition.

6.5 Multichannel operation

Anther key requirement is a multichannel operation. In order to provide synchronization between multiple cameras PTP (Presicion Time Protocol) or MLVDS (multipoint LVDS) interfaces can be used. First is based on ethernet whereas second is a multipoint hardware serial link. PTP is an ethernet based solution which allows for microsecond synchronization and requires dedicated and costly time reference hardware (Meinberg 100K PLN). Another solution is to use MLVDS interface where multiple cameras are connected via this links and are tightly synchronized in hardware, but the implementation is more complicated.

Both of this approaches are available in my design, but are not fully implemented and tested.

7 Work status

At this point what I have done is:

- realized sensor acquisition
- successfully run both operating systems on Cortex A9
- tested PTP for synchronization
- started integration of FADE IP into my design

What still has to be done:

- finish and polish software - Hardware (FPGA) integration
- run 10 GbE - using FADE IP from PhD. Zabolotny
- MLVDS synchronization

8 Further development

Because mostly all cameras has got ethernet interface, a key issue is security. What I think might be imporant to add in future development of my project is virtualization. A technique known from x86 architecture, which has lately become very popular in embedded systems like smartphones. It provides the possibility of having a supervisor which looks after all operating systems and is working beneath them, closer to hardware. When any of the system running becomes corrupted or wants to access a peripherial it shouldn't have access to (e.g. in case of intrusion). Supervisor will not allow for that and will deny the access to a particular peripheral or will reset the operating system.

9 Summary

This slide summarizes my presentation. What you have learned during my short talk is that a camera firmware design can be a complex task. My project is a novel framework which can be used in any camera development and can provide flexibility, high performance as well as faster time-to-market.

Thank you for your attention.