Master-Thesis

Development of a receiver system for the RTO

Fabio Di Lorenzo
fabio.dilorenzo@ds.mpg.de

Date: 31.03.2010
Advisors: Prof. Dr. Löffler-Mang
           Prof. Dr. Bodenschatz (eberhard.bodenschatz@ds.mpg.de)
Master’s program: Mechatronic and Sensor Technology
                  Department of Engineering Science
Declaration Of Authorship

I, Fabio Di Lorenzo, declare that the presented work was prepared independently and without using anything other than the named sources.

Göttingen, March 31, 2010

Signed: .....................................................
Acknowledgement

At first I would like to thank my parents and my sister for their encouragement and support. I thank Prof. Martin Löffler-Mang from the University of Applied Sciences in Saarbrücken and Prof. Eberhard Bodenschatz from the Max Planck Institute for Dynamics and Self-Organization in Göttingen for giving me the chance to work at the MPIDS and for their advise. Moreover I thank the members of the LFPN for a great time on and off the job.
## Contents

1 Introduction .......................................................... 1

2 RTO-Interface ...................................................... 5
   2.1 Overview ....................................................... 5
   2.2 RocketIO ....................................................... 6
      2.2.1 Readout sequence ...................................... 6
      2.2.2 8b/10b encoding ...................................... 7
      2.2.3 Camera protocol (confidential) ...................... 9
   2.3 Optical transmitter ......................................... 10
   2.4 Capability of the RTO ....................................... 11
   2.5 Requirements on the receiver system .................... 12

3 FPGA-Board .......................................................... 13
   3.1 Framework Logic ............................................. 15
      3.1.1 Deframer and Stacker .................................. 16
      3.1.2 VFIFO ................................................... 16
      3.1.3 Destacker ............................................... 16
      3.1.4 Alerts ................................................... 16
      3.1.5 Packetizer .............................................. 16
      3.1.6 PCIe ..................................................... 16
      3.1.7 Modification of the Framework Logic ............... 17

4 Configuration of the RocketIO .................................... 18
   4.1 Transfer rate ................................................. 19
   4.2 Comma detection and alignment ............................ 19
   4.3 Polarity ....................................................... 20

5 Implementing the RocketIO ....................................... 21
   5.1 RocketIO Wrapper Tile ...................................... 21
   5.2 Receiver ...................................................... 23
   5.3 Sender ......................................................... 24
   5.4 RocketIO Interface ......................................... 25
   5.5 RocketIO clocks .............................................. 27
   5.6 RocketIO Channel Interface ................................. 28
   5.7 User constraints file ....................................... 29

6 Transceiver-Board .................................................. 30
CONTENTS

7 Results 32

8 Outlook 33
  8.1 FPGA logic .......................................................... 33
  8.2 Host program ......................................................... 34
  8.3 Digital inputs and outputs ....................................... 35
  8.4 Cameras ............................................................... 35

List of Tables 36

List of Figures 37

References 38

Appendix
  A Protocol for 10 bit (confidential)
  B Protocol for 12 bit (confidential)
  C Protocol for 16 bit (confidential)
  D rocketio_wrapper_tile.vhd (confidential)
  E data_receiver.vhd (confidential)
  F data_sender.vhd (confidential)
  G rio_intf.vhd (confidential)
  H rio_clocks.vhd (confidential)
  I rio_chan_intf.vhd (confidential)
  J Schematic of the Transceiver-Board
  K Mechanical drawing of the slot bracket
1 Introduction

In the Laboratory for Fluid Dynamics, Pattern Formation and Nanobiocomplexity (LFPN) of the Max-Planck-Institute for Dynamics and Self-Organization, Lagrangian Particle Tracking (LPT) is used to investigate fluid turbulence. LPT is a measurement technology that detects the trajectories of particles moving through a measurement volume. From these trajectories the velocity and acceleration information can be retrieved.

In experiments analyzed with 3 dimensional LPT, three to four high speed cameras take images of the measurement volume from different angles. As an example fig. 1.1 shows the Lagrangian Exploration Module (LEM). The LEM is an apparatus with the shape of an icosahedron, where 12 propellers generate a homogeneous and isotropic turbulence [1].

Figure 1.1: LPT on the Lagrangian Exploration Module [1]

To make the fluid motion visible, the fluid is seeded with particles that follow the flow. The particles in the measurement volume are illuminated by a laser, and the cameras record high speed movies of the particle motion. Since the cameras are synchronized, corresponding images from all cameras show the measurement volume from different angles at the same time (fig. 1.2). After recording, the images of each camera are downloaded to a computer for post processing.

\[1\] also known as Particle Tracking Velocimetry (PTV)
The following description of the post processing procedure is based on reference [2]. The first step is to find all particles in every image and to determine their coordinates within the image planes of the cameras. In LPT the seeding density is small to reduce the probability of overlapping particles\(^2\). Due to the small seeding density, images taken for LPT consist of a few bright pixels and large dark areas. By comparing every pixel to a threshold value, the image is segmented into groups of bright pixels, that represent individual particles, while dark pixels are discarded.

A particle finding algorithm then determines the position of the particle in the image plane of the camera. With the particle positions from corresponding images of the different cameras, the particle position in 3D can be determined with a stereo-matching process. The 3D particle positions from all time steps are then used to track the particles in time, with the particle trajectories as a result. Different algorithms for particle finding and particle tracking are discussed in [2], but for this work that part of the post processing is of less importance.

---

2One speaks of overlapping particles, when two or more particles are in a line from the viewpoint of a camera, and appear as one particle in the image.
1 Introduction

We now want to focus on the measurement procedure. In a turbulent flow very fast processes occur, i.e. to observe them cameras that are capable of high frame rates are needed. Hence, the cameras used for LPT in the LFPN are from Vision Research’s Phantom V-Series. As an example, with a resolution of 480x480 pixels the Phantom V10 can take images at a frame rate of 6 kHz. This corresponds to a data rate of approximately 1.4 GB/s. As explained before, the images have to be downloaded to a computer for post processing. This is done by use of the Gigabit-Ethernet interface of the cameras. As transfer rate of the Gigabit-Ethernet 28 MB/s was measured. Due to this difference downloading the images to a computer in real-time is not possible with Gigabit-Ethernet. Therefore, the images are stored in the internal RAM of the cameras and are downloaded afterwards. That leads to the problem, that for the use of four cameras, one second of acquisition involves \(\frac{4 \times 1.4 \text{ GB/s \cdot 28 \text{ MB/s \cdot 1 s}}}{28 \text{ MB/s}} = 200 \text{ s}\) of downloading time. During this time no data can be acquired.

Figure 1.3 shows a plot of the number of particles (in arbitrary units) against time in minutes. Each data point represents a measurement of 1 second. The time in between two data points is the downloading time. The fact that the number of particles decreases with time shows that valuable information for the statistics gets lost during the time between the measurements. The decrease of the number of particles in the measurement volume is due to the density mismatch of particles and fluid. The blue curve shows that even tracer particles, which are considered to be neutrally buoyant with a density of only 6% bigger than the density of water, sink below the measurement volume. The red curve shows that the decrease is faster for heavy particles (glass particles, 4 times heavier than water).

![Figure 1.3: Decrease of particle density in the measurement volume](image)

Fabio Di Lorenzo

Master Thesis
To solve this problem Chan et al [3] showed that a real-time image compression can be performed. The basis for this is the fact, that the seeding density in LPT experiments is low and only the bright pixels are needed for the described post processing process. They developed an FPGA-Board that receives the high speed images of the camera, performs the thresholding and sends only the bright pixels with coordinates to a computer. Due to this sparsification of the images, the data rate was reduced below the writing speed of the hard drive and the relevant data of the high speed movies could be saved in real time. Compression rates\(^3\) of 100-1000 were achieved at a frame rate of 500 Hz [3].

The aim of this work is to develop a similar system for the much faster Phantom cameras. These provide, besides the Gigabit-Ethernet interface, an optical output, the so called \textit{Real-Time Output (RTO)}. On this RTO, raw sensor data are sent out at a higher transfer rate (1.5552 GB/s including overhead for Phantom V7 e.g.) than would be possible with Gigabit-Ethernet. The RTO will be described in section 2. An FPGA-Board, the X5-TX, was purchased to receive the RTO signals and to write the data on the hard drive of a computer. This FPGA-Board and the logic provided with it is explained in section 3. The necessary interface to receive the RTO signals is called RocketIO transceiver and was not implemented in the provided logic. Hence, configuration and implementation of the RocketIOs in the FPGA logic was the main task of this work and is described in the sections 4 and 5. Since the RTO sends optical signals and the X5-TX provides eSATA connectors to access the RocketIOs, a circuit board for conversion of optical to electrical signals was developed. This circuit board was called Transceiver-Board and is topic of section 6. The results are presented in section 7 and an outlook is given in section 8.

\(^3\)The compression rate depends on the seeding density
2 RTO-Interface

This section describes the Real-Time-Output of the Phantom V7.2. Understanding how the signal is generated and which information it carries will lead to important requirements for the receiving system. Differences to other camera models like the Phantom V10 will be pointed out. These differences determine which properties of the receiver system have to be flexible, so it can be designed to be compatible with a variety of camera models.

2.1 Overview

The cameras have an built in Field Programmable Gate Array (FPGA). Specifically, it is the model Virtex II Pro from Xilinx. The FPGA reads the data from the CMOS chip and sends them to an optical transmitter using its RocketIO ports (fig. 2.1). The optical transmitter converts the differential electrical signals from the RocketIO ports to optical signals and outputs these through an MTP/MPO receptacle, shown in fig. 2.2.

![Camera block diagram](image1)

**Figure 2.1:** Camera block diagram

![Real Time Output](image2)

**Figure 2.2:** Real Time Output
Different camera models use a different number of RocketIO channels and have different transfer rates. In tab. 2.1 the values for various cameras are shown.

<table>
<thead>
<tr>
<th>camera model</th>
<th>no. of channels</th>
<th>transfer rate per channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>V7.2</td>
<td>8</td>
<td>1.5552 Gb/s</td>
</tr>
<tr>
<td>V10</td>
<td>6</td>
<td>2.125 Gb/s</td>
</tr>
<tr>
<td>V12</td>
<td>8</td>
<td>2.125 Gb/s</td>
</tr>
<tr>
<td>V640</td>
<td>8</td>
<td>2.125 Gb/s</td>
</tr>
</tbody>
</table>

Table 2.1: Transfer rates and channel numbers [4]

2.2 RocketIO

The Xilinx FPGAs provide versatile multi-gigabit transceivers, called RocketIO transceivers. The RocketIOs are used to send the camera data and can be calibrated to satisfy different standards like Gigabit-Ethernet, PCI-Express or Infiniband, for example. Vision Research uses a customized protocol that will be described in section 2.2.3.

2.2.1 Readout sequence

For transmission, every image (frame) is split up to all available RocketIO channels. Every channel sends rows of pixels from left to right, starting with the rows in the middle of the image. As shown in fig. 2.3 and fig. 2.4, one half of the channels send the upper half of the image, while the other half of the channels send the lower half.

Figure 2.3: Readout sequence for 8 channels [4]  
Figure 2.4: Readout sequence for 6 channels [4]
2.2.2 8b/10b encoding

Before data are sent, the RocketIO transceiver performs 8b/10b encoding on every data byte, i.e. that every 8 bit character is assigned to a 10 bit symbol according to a look-up-table [6]. The serial bit stream sent out of the RTO is a sequence of these 10 bit symbols.

One reason to perform 8b/10b encoding is that some of the 8 bit characters do not provide enough bit transitions for the receiver to recover a clock out of the data stream. In tab. 2.2 some of these unfavorable characters are listed. The table shows that these characters are assigned to symbols that provide more transitions and never more than five 1s or 0s in a row.

Another reason for 8b/10b encoding is to keep the data stream DC-balanced. A signal is considered as DC-balanced if the difference between the number of transmitted 1s and the number of transmitted 0s (the disparity) is at most two. Therefore in the look-up-table every 8 bit character in fact corresponds to two 10 bit symbols, one with six 1s (RD-) and one with six 0s (RD+) (tab. 2.2). The RocketIO transceiver increases or decreases the disparity by choosing either an RD+ or an RD- symbol to send.

The full 8b/10b look-up-table is listed in [6].

<table>
<thead>
<tr>
<th>Data Byte Name</th>
<th>8b Character</th>
<th>10b Symbol RD-</th>
<th>10b Symbol RD+</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3.0</td>
<td>000 00011</td>
<td>110001 1011</td>
<td>110001 0100</td>
</tr>
<tr>
<td>D0.1</td>
<td>001 00000</td>
<td>100111 1001</td>
<td>011000 1001</td>
</tr>
<tr>
<td>D31.3</td>
<td>011 11111</td>
<td>101011 0011</td>
<td>010100 1100</td>
</tr>
<tr>
<td>D28.7</td>
<td>111 11100</td>
<td>001110 1110</td>
<td>001110 0001</td>
</tr>
</tbody>
</table>

**Table 2.2:** Extract of the 8b/10b look-up-table [6]
Even with every 8 bit data character being assigned to two 10 bit symbols, there are still half of the possible 10 bit combinations left. Most of them are unused, but some are assigned to 8 bit control characters, the so called K-Characters. The K-Characters used by the FPGA in the Phantom cameras are listed in tab. 2.3. This document will refer to the K-Characters by their meaning for the protocol. The full table of valid K-Characters is listed in [6].

<table>
<thead>
<tr>
<th>Control Byte Name</th>
<th>Hex Value</th>
<th>8b Character</th>
<th>10b Symbol RD-</th>
<th>10b Symbol RD+</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>K28.4</td>
<td>9C</td>
<td>100 11100</td>
<td>001111 0010</td>
<td>110000 1101</td>
<td>SOF$^1$</td>
</tr>
<tr>
<td>K27.7</td>
<td>FB</td>
<td>111 11011</td>
<td>110110 1000</td>
<td>001001 0111</td>
<td>SOP$^2$</td>
</tr>
<tr>
<td>K29.7</td>
<td>FD</td>
<td>111 11101</td>
<td>101110 1000</td>
<td>010001 0111</td>
<td>EOP$^3$</td>
</tr>
<tr>
<td>K28.5</td>
<td>BC</td>
<td>101 11100</td>
<td>001111 1010</td>
<td>110000 0101</td>
<td>Comma</td>
</tr>
<tr>
<td>K28.0</td>
<td>1C</td>
<td>000 11100</td>
<td>001111 0100</td>
<td>110000 1011</td>
<td>Idle</td>
</tr>
<tr>
<td>K28.2</td>
<td>5C</td>
<td>010 11100</td>
<td>001111 0101</td>
<td>110000 1010</td>
<td>Non-Idle</td>
</tr>
<tr>
<td>K28.3</td>
<td>7C</td>
<td>011 11100</td>
<td>001111 0011</td>
<td>110000 1100</td>
<td>Non-Idle</td>
</tr>
</tbody>
</table>

Table 2.3: Extract of K-Characters [6]

$^1$SOF = Start Of Frame
$^2$SOP = Start Of Packet
$^3$EOP = End Of Packet
2.2.3 Camera protocol (confidential)

*Due to a non-disclosure agreement with Vision Research, the information in this section is confidential.*

**Figure 2.5:** Transmission of packets

**Figure 2.6:** Data packet [5]

**Table 2.4:** Transfer protocol for 8 bit pixels [5]
2.3 Optical transmitter

After 8b/10b encoding, the 10 bit sequences of all RocketIO channels are sent to the optical transmitter as differential electrical signals. The optical transmitter is a ZL60101 TX (fig. 2.7) from Zarlink Semiconductor. It provides 12 channels to convert electrical signals to optical signals of 850 nm wavelength [7]. It can be connected to a fiber cable with an MTP/MPO connector (fig. 2.8). These cables have 12 fiber cores. As shown in tab. 2.1 only 6 or 8 channels are used, depending on the camera type. Therefore the channels in the middle of ZL60101 TX are not used [5].

![Optical transmitter ZL60101 TX](image)

**Figure 2.7:** Optical transmitter ZL60101 TX [7]

![Fiber cable with MTP/MPO connector](image)

**Figure 2.8:** Fiber cable with MTP/MPO connector
2.4 Capability of the RTO

Unfortunately, the RTO is not fast enough for transmission of images at the highest possible frame rate and full resolution. If the amount of data caused by the requested frame rate and image size is too large, the RTO will automatically output the images at a submultiple of the desired frame rate [5]. This section presents an equation to calculate the maximum possible frame rate for a given image size.

Eq. 1 can be formed from the protocol described in tab. 2.4. It shows that the amount of data to be transferred caused by one frame, multiplied by the frame rate equals the transfer rate. Due to 8b/10b encoding the transfer rate has to be reduced by 20% to fulfill the equation.

\[(FSP + n \cdot DP + IFG) f_{\text{max}} = TF \cdot 0.8\] (1)

- \(FSP\) : frame start packet including SOP and EOP (42 B)
- \(n\) : number of data packets
- \(DP\) : data packet including SOP, EOP, Comma and IPG (140 B)
- \(IFG\) : inter frame gap, minimum size is used to calculate the maximum framerate (1600 B)
- \(f_{\text{max}}\) : maximum frame rate for a given image size
- \(TF\) : transfer rate

The number of data packets is a function of the image size, the depth of the pixels and the number of channels of the camera, taking into account that one data packet carries 128 B of information (eq. 2).

\[n = \frac{xy \cdot d}{ch \cdot 128 B}\] (2)

- \(n\) : number of data packets
- \(xy\) : resolution
- \(d\) : pixel depth
- \(ch\) : number of channels
2.5 Requirements on the receiver system

With eq. 2 in eq. 1, the maximum frame rate for a given image size can be calculated (eq. 3).

\[
f_{\text{max}} = \frac{TF \cdot 0.8}{(FSP + \frac{xy \cdot d_{\text{ch}}}{128} \cdot DP + IFG)}
\]

\[
f_{\text{max}} = \frac{TF \cdot 0.8}{(\frac{xy \cdot d_{\text{ch}}}{128} 140 + 1642 B)}
\] (3)

An example calculation (eq. 4) shows the maximum frame rate that can be outputted by the RTO of a Phantom V7.2 (8 channels, 8 bit depth, 1.5552 Gb/s) for a resolution of 256x256 pixels.

\[
f_{\text{max}} = \frac{1.5552 \text{Gb/s} \cdot 0.8}{(\frac{256^2 B}{8} \frac{140}{128} + 1642 B)}
\]

\[
f_{\text{max}} = 14.67 \text{kHz}
\] (4)

For comparison, if images are written only to the internal RAM of the camera, one can record with up to 36.7 kHz at that resolution [8]. However, the RTO outputs, in this case, at a total data rate of \(256^2 B \cdot 14.67 \text{kHz} = 961.4 \text{MB/s}\). This is about a factor of 100 faster than transferring through Gigabit-Ethernet, which was measured to transfer at approximately 10 MB/s for the same camera.

2.5 Requirements on the receiver system

The requirements on the receiver system arising from the previous sections are:

- it has to receive the optical signals of up to 2.125 Gb/s per channel and convert them to electrical signals for processing with an FPGA
- the FPGA has to be able to perform 8b/10b decoding
- it has to be flexible in the number of channels and transfer rate in order to be compatible with all types of cameras
- it needs a fast interface to a computer for storing image data
3 FPGA-Board

To match the requirements on the receiver system while keeping the task of hardware development small, we decided to use an off-the-shelf FPGA module that provides the necessary interfaces and software. After reviewing commercially available FPGA modules on the market, we chose the module X5-TX Rev. B (fig. 3.1) from Innovative Integration:

Advantages: + Integrated FPGA is a Virtex 5 SX95T from Xilinx and provides RocketIOs
+ The pins of 8 RocketIO channels are routed to eSATA connectors (fig. 3.1)
+ FPGA logic is provided by the manufacturer and is fully customizable
+ An adapter card provides a fast interface (PCIe) to a host computer (fig. 3.1)
+ Host computer programs for communication with and controlling of the FPGA board are provided by the manufacturer and are fully customizable
+ It provides an adjustable PLL-Clock. This is important to source the RocketIOs with different frequencies to match the different transfer rates of the cameras.

Disadvantages: - It does not provide the necessary optical interface. That means a circuit board has to be developed which provides an optical receiver and routes the converted signals to eSATA connectors, see section 6.
- RocketIOs are not implemented in the FPGA logic, see section 4 and section 5.
The *X5-TX* is a D/A converter card. It is designed to read digital data from the hard drive of the host computer, convert these to analog signals and provide them on SMA outputs. Figure 3.1 shows the X5-TX plugged onto the adapter card that provides the connectors. The goal is to change the FPGA logic so that it reads data from the camera on the RocketIO inputs and writes these to the hard drive.

![Figure 3.1: X5-TX with adapter card](image)

As a starting point, the FPGA logic delivered by the board manufacturer, the so called *Framework Logic*, was used. The following section describes briefly the structure of the *Framework Logic* and shows what has to be modified for the *X5-TX* to become a receiver card for the RTO.
3.1 Framework Logic

The Framework Logic described here is not the official version usually delivered with the X5-TX. It is a version modified by the customer service of Innovative Integration. Basically all functions related to D/A conversion were removed. The rest is important for board to host communication. To keep it simple this document will refer to this version as Framework Logic.

Figure 3.2 shows a block diagram of the components of the Framework Logic that are relevant for board to host communication. One can see that in this state the FPGA just receives data from the host computer through the PCIe interface and sends them right back. This loopback structure is the starting point for further modifications.

![Framework Logic block diagram](image)

**Figure 3.2:** Framework Logic block diagram

The individual components shown in the block diagram will be explained in more detail in the following sections. The informations about these components are taken from [9].
3.1 Framework Logic

3.1.1 Deframer and Stacker

The components Deframer and Stacker receive data packets from the PCIe interface and are used to interpret the header information and route the data to their destination device.

3.1.2 VFIFO

The X5-TX is equipped with four 64M x 16 SRAM devices. Together they form the component VFIFO (virtual first in first out). A FIFO is used as a buffer if data is passed from one clock domain to another. It simply stores data from the Stacker until the Destacker is ready to receive them.

3.1.3 Destacker

The component Destacker receives 128 bit words from the VFIFO and passes these on to the Packetizer as two 64 bit words.

3.1.4 Alerts

In the component Alerts, critical system parameters like the buffer state or temperature are monitored. If an alert occurs a packet is sent to the host computer through Packetizer and PCIe interface for interpretation by the host software.

3.1.5 Packetizer

The component Packetizer prepares data packets to be sent through the PCIe interface. It attaches a header to each packet that will be interpreted by the host software.

3.1.6 PCIe

The Virtex-5 SX95T has 16 RocketIO transceivers in total. Eight of them are used by the Framework Logic for the 8 lane PCIe interface to the host computer. The component PCIe encapsulates the logic needed for fast data transfer between host and FPGA board. It sends data from the Packetizer to the computer and data from the computer to the Deframer. The component additionally provides a so-called command channel for status and control of the FPGA.
3.1.7 Modification of the Framework Logic

The goal is to receive signals from the RocketIO transmitters of the camera with the FPGA and write them to a file on the host computer through the PCIe interface. As shown in fig. 3.2, the Framework Logic already provides components to stream data from the FPGA to the host. However, in order to receive camera data the RocketIOs have to be implemented into the logic. Then the data can be routed from the RocketIOs to the VFIFO and make its way through Destacker, Packetizer and PCIe interface. Since in the planned application no data streaming from the host to the FPGA is intended, Deframer and Stacker will not be needed (fig. 3.3).

![Figure 3.3: Framework Logic block diagram with RocketIO](image)

The red block called RocketIO in (fig. 3.3) shows what was added to the Framework Logic to enable the X5-TX to receive the camera signals. In the sections 4 and 5 this red block will be described in more detail. The circuit board that had to be developed to convert the optical signals to electrical signals is the topic of section 6.
4 Configuration of the RocketIO

To implement RocketIO transceivers a component from the Xilinx library, a so called Xilinx primitive, is used. In the Virtex 5, the primitive encapsulating RocketIOs is called *GTP Dual*. A GTP Dual operates two RocketIO transceivers, hence two transmitters and two receivers. This section deals with the configuration of the GTP Dual. It is configured by assigning values to configuration ports and attributes of the component. For test purposes the RocketIO transmitters are also configured. Table 4.1 and table 4.2 list the most important configuration ports and attributes, the rest is either self explanatory or assigned to its default value. A full list is provided in [6].

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLL_DIVSEL_FB</td>
<td>2</td>
<td>Transfer rate</td>
</tr>
<tr>
<td>PLL_DIVSEL_REF</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PLL_TXDIVSEL_COMM_OUT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PLL_TXDIVSEL_OUT_x</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PLL_RXDIVSEL_OUT_x</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ALIGN_COMMA_WORD_x</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>COMMA_DOUBLE_x</td>
<td>FALSE</td>
<td>Comma detection and alignment</td>
</tr>
<tr>
<td>MCOMMA_10B_VALUE_x</td>
<td>1010000011</td>
<td></td>
</tr>
<tr>
<td>PCOMMA_10B_VALUE_x</td>
<td>0101111100</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Attributes of the GTP Dual [6]

<table>
<thead>
<tr>
<th>Port</th>
<th>Value</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTDATAWIDTH</td>
<td>1</td>
<td>Transfer rate</td>
</tr>
<tr>
<td>RXDATAWIDTHx</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TXDATAWIDTHx</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RXDEC8B10BUSEx</td>
<td>1</td>
<td>Comma detection and alignment</td>
</tr>
<tr>
<td>TXENC8B10BUSEx</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RXCOMMADETUSEx</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RXENMCOMMAALIGNx</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RXENPCOMMAALIGNx</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TXPOLARITY0</td>
<td>0</td>
<td>Polarity</td>
</tr>
<tr>
<td>TXPOLARITY1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RXPOLARITY0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RXPOLARITY1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Configuration ports of the GTP Dual [6]
The following sections contain brief descriptions of the configurations made with the listed configuration ports and attributes. A configuration port or an attribute with an ‘x’ at the end of its name indicates that this parameter appears twice, one for each channel of the GTP Dual.

### 4.1 Transfer rate

As shown in tab. 2.4, the incoming data are handled as 16 bit words [5]. Therefore INTDATAWIDTHx, RXDATAWIDTHx and TXDATAWIDTHx have to be set to 1 [6]. To match the transfer rate of the camera the frequency of the clock provided at the main clock input CLkin of the GTP Dual is very important. This frequency is set to 1/20 of the transfer rate (tab. 2.1), since 20 bits of the incoming 10 bit symbols have to be gathered to produce one 16 bit word after 8b/10b decoding. As a result CLkin will be fed with \(77.76 \, MHz\) for a transfer rate of 1.5552 \(Gb/s\) or with 106.25 \(MHz\) for 2.125 \(Gb/s\), see section 5.5. In addition to providing one of the mentioned frequencies (depending on the camera model) several parameters have to be set according to eq. 5 taken from [6].

\[
\frac{PLL_{DIVSEL_{FB}}}{PLL_{DIVSEL_{REF}}} = \frac{\text{transfer rate}}{CLKIN \cdot 10} = \frac{1.5552 \, GHz}{77.76 \, MHz \cdot 10} = 2 \quad (5)
\]

As shown in eq. 5 the ratio between PLL_DIVSEL_FB and PLL_DIVSEL_REF has to be 2. Therefore PLL_DIVSEL_FB was set to 2 and PLL_DIVSEL_REF was set to 1. With this configuration it is possible to switch between the two transfer rates just by changing the frequency at CLkin.

The GTP Dual provides the possibility to have different transfer rates on its two channels. Since this is not needed the dividers for this purpose - PLL_RXDIVSEL_OUT_x, PLL_TXDIVSEL_COMM.OUT, PLL_TXDIVSEL.OUT_x - are all set to 1.

### 4.2 Comma detection and alignment

While receiving a bit stream from the camera, the receiver initially does not know where a 10 bit symbol starts and where it ends. Therefore the camera sends a Comma before every packet (tab. 2.4). A Comma detection and alignment circuit in the GTP Dual searches for this K-Character in the bit stream and aligns to its boundaries [6]. This section explains how to configure the Comma detection and alignment circuit. 8b/10b decoding has to be enabled by setting RXDEC8B10BUSEx and TXENC8B-10BUSEx to 1. The Comma detection and alignment circuit is enabled by assigning 1 to RXCOMMADETUSEx, RXENMCOMMAALIGNx and RXENPCOM-
MAALIGNx. To clarify which K-Character is used as Comma the attribute PCOMMA_{10B}_VALUE_{x} is set to 0101 111100, and MCOMMA_{10B}_VALUE_{x} is set to 1010 000011. These values correspond to the 10 bit symbols for Comma listed in tab. 2.3. As mentioned in section 2.2.2 every 8 bit character corresponds to two 10 bit symbols for disparity control. Therefore the GTP Dual distinguishes between PCOMMA and MCOMMA. The protocol used by the camera sends either MCOMMA or PCOMMA. Therefore COMMA_DOUBLE_{x} has to be set to FALSE. Setting ALIGN_COMMA_WORD_{x} to 1, tells the receiver that there is no preferred position in the internal data path, which means that if a Comma is found and the receiver aligns to it, the Comma can be written to the upper or to the lower byte of the 16 bit word.

4.3 Polarity

On the X5-TX the pins of both receivers of a GTP Dual are routed to one eSATA connector on the adapter board. The same is done for the pins of the transmitters (fig. 3.1). In both cases the polarity of channel one is switched to avoid vias [10]. For compensation RXPOLARITY1 and TXPOLARITY1 have to be set to 1, while RXPOLARITY0 and TXPOLARITY0 remain 0.
5 Implementing the RocketIO

The following sections describe how the RocketIos were implemented into the Framework Logic. For this purpose several components were programmed in VHDL (very high speed integrated circuit hardware description language), just like the Framework Logic. The following description of the components starts with the components on the lowest hierarchy level and moves on to the higher ones.

5.1 RocketIO Wrapper Tile

The rocketIO_wrapper_tile encapsulates the Xilinx primitive GTP Dual described in section 4. This component only provides the most important inputs and outputs of the GTP Dual to other components at the same hierarchy level (fig. 5.1). The rocketIO_wrapper_tile itself is encapsulated in the component RocketIO interface (rio_intf). The most important inputs and outputs are listed in tab. 5.1. Appendix D shows the VHDL code of the rocketIO_wrapper_tile.

<table>
<thead>
<tr>
<th>Port name</th>
<th>Direction</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXDATAx</td>
<td>OUT</td>
<td>Received 16 bit words (tab. 2.4) after 8b/10b decoding</td>
</tr>
<tr>
<td>RXCHARISKx</td>
<td>OUT</td>
<td>After 8b/10b decoding data characters cannot be distinguished from K-Characters, this 2 bit value indicates if the upper or the lower byte of the received 16 bit word is a K-Character</td>
</tr>
<tr>
<td>TXOUTCLK</td>
<td>OUT</td>
<td>A clock generated by the GTP Dual according to the frequency of CLKIN and the settings for the transfer rate, see section 4.1. The frequency of TXOUTCLK is transfer rate/10 and used to generate USRCLK and USRCLK2, see section 5.5</td>
</tr>
<tr>
<td>TXDATAx</td>
<td>IN</td>
<td>16 bit word to be sent over the RocketIO</td>
</tr>
<tr>
<td>TXCHARISKx</td>
<td>IN</td>
<td>2 bit value that indicates if the upper or the lower byte of TXDATAx is a K-Character</td>
</tr>
<tr>
<td>USRCLK</td>
<td>IN</td>
<td>Clock needed for the internal logic of the GTP Dual. The frequency is transfer rate/10</td>
</tr>
<tr>
<td>USRCLK2</td>
<td>IN</td>
<td>Clock needed for the interface between GTP Dual and the Framework Logic. The frequency is transfer rate/20</td>
</tr>
<tr>
<td>CLKIN</td>
<td>IN</td>
<td>Reference clock for the transfer rate settings</td>
</tr>
</tbody>
</table>

Table 5.1: Inputs and outputs of rocketIO_wrapper_tile [6]
Figure 5.1: RocketIO interface
5.2 Receiver

The component `data_receiver` handles the data stream of one RocketIO receiver channel and is therefore instantiated twice in `rio_intf` (fig. 5.1). It interprets the K-Characters of the incoming data stream and raises the output `DATA_RDY_TRIG` to 1 if an SOP is found and resets it to 0 if an EOP is found (tab. 2.4). Hence, `DATA_RDY_TRIG` indicates the following component if the data is part of a packet or not, so that the Idle and Non-Idle sequences of the IPG and IFG can be skipped. Moreover the `data_receiver` calculates the CRC of the incoming data and compares it to the CRC at the end of each packet. In the current design a wrong checksum has no consequence, but later this can be used to detect transmission errors.

The most important inputs and outputs are listed in tab. 5.2. Appendix E shows the VHDL code of the `data_receiver`.

<table>
<thead>
<tr>
<th>Port name</th>
<th>Direction</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATAIN</td>
<td>IN</td>
<td>Input for 16 bit words received from the RXDATAx of the rocketIO_wrapper_tile</td>
</tr>
<tr>
<td>CHAR_IS_K</td>
<td>IN</td>
<td>Input for the 2 bit value received from the RXCHARISKx of the rocketIO_wrapper_tile</td>
</tr>
<tr>
<td>DATAOUT</td>
<td>OUT</td>
<td>Data from DATAIN are routed to DATAOUT without being changed</td>
</tr>
<tr>
<td>DATA_RDONLY</td>
<td>OUT</td>
<td>A signal that indicates whether the 16 bit word in DATAOUT is part of a packet or not</td>
</tr>
</tbody>
</table>

**Table 5.2**: Inputs and outputs of `data_receiver`
5.3 Sender

The component *data_sender* was implemented for test purposes. It sends the same data stream to both transmitter channels of the *rocketIO_wrapper_tile* and is encapsulated in *rio_intf* (fig. 5.1). This data stream consists of a repeated data packet of the form shown in fig. 2.6. In the current design the 128 data characters of the data packet are all 0xFF, and therefore the data stream corresponds to a white image.

This data stream can be received by connecting an eSATA connector of the Rocke-tIO transmitters to an eSATA connector of the RocketIO receivers (fig. 3.1). For the development of the data receiver it was important to have this source of known data.

The outputs of *data_sender* are listed in tab. 5.3. Appendix F shows the VHDL code of the *data_sender*.

<table>
<thead>
<tr>
<th>Port name</th>
<th>Direction</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR_IS_Kx</td>
<td>OUT</td>
<td>Output of a 2 bit value for TXCHARISKx of the rocketIO_wrapper_tile</td>
</tr>
<tr>
<td>DATAOUTx</td>
<td>OUT</td>
<td>Data output for TXDATAx of the rocketIO_wrapper_tile</td>
</tr>
</tbody>
</table>

**Table 5.3:** Outputs of *data_sender*
5.4 RocketIO Interface

The component rio_intf was programmed for structural purposes. The components explained in the previous sections - rocketio_wrapper_tile, data_receiver and data_sender - are all encapsulated and connected with each other inside the rio_intf (fig. 5.1). By providing access only to important inputs and outputs of these components, the rio_intf becomes the interface for one GTP Dual, i.e. two RocketIO transceivers. In order to receive data from 8 RocketIO transceivers rio_intf has to be instantiated four times in the RocketIO Channel Interface (fig. 5.2).

The most important inputs and outputs of rio_intf are listed in tab. 5.4. Appendix G shows the VHDL code of the rio_intf.

<table>
<thead>
<tr>
<th>Port name</th>
<th>Direction</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATAOUTx</td>
<td>OUT</td>
<td>Provides access to DATAOUT of the data_receiver</td>
</tr>
<tr>
<td>DATA_RDY_TRIGx</td>
<td>OUT</td>
<td>Provides access to DATA_RDY_TRIG of the data_receiver</td>
</tr>
<tr>
<td>TXOUTCLK</td>
<td>OUT</td>
<td>Provides access to TXOUTCLK of the rocketio_wrapper_tile</td>
</tr>
<tr>
<td>USRCLK</td>
<td>IN</td>
<td>Provides access to USRCLK of the rocketio_wrapper_tile</td>
</tr>
<tr>
<td>USRCLK2</td>
<td>IN</td>
<td>Provides access to USRCLK2 of the rocketio_wrapper_tile</td>
</tr>
<tr>
<td>CLKIN</td>
<td>IN</td>
<td>Provides access to CLKIN of the rocketio_wrapper_tile</td>
</tr>
</tbody>
</table>

Table 5.4: Inputs and outputs of rio_intf
Figure 5.2: RocketIO channel interface
5.5 RocketIO clocks

The component rio_clocks provides USRCLK and USRCLK2 to the rio_intf by deriving these clocks from TXOUTCLK. Figure 5.2 shows that TXOUTCLK is routed from one rio_intf to rio_clocks, while USRCLK and USRCLK2 are routed from rio_clocks to all four instantiated rio_intf.

As explained in tab. 5.1 TXOUTCLK and USRCLK have the frequency of 1/10 of the transfer rate, while USRCLK2 has the frequency of 1/20 of the transfer rate. A digital clock manager in rio_clocks ensures USRCLK and USRCLK2 have these frequencies and are edge aligned to each other. This way of clocking is suggested in [6].

The most important inputs and outputs of rio_clocks are listed in tab. 5.5. Appendix H shows the VHDL code of the rio_clocks.

<table>
<thead>
<tr>
<th>Port name</th>
<th>Direction</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXOUTCLK</td>
<td>IN</td>
<td>TXOUTCLK from the rocketio_wrapper_tile. The frequency is transfer rate / 10</td>
</tr>
<tr>
<td>USRCLK</td>
<td>OUT</td>
<td>USRCLK for the rocketio_wrapper_tile. The frequency is transfer rate / 10. It is edge aligned to USRCLK2</td>
</tr>
<tr>
<td>USRCLK2</td>
<td>OUT</td>
<td>USRCLK2 for the rocketio_wrapper_tile. The frequency is transfer rate / 20. It is edge aligned to USRCLK</td>
</tr>
</tbody>
</table>

Table 5.5: Inputs and outputs of rio_clocks
5.6 RocketIO Channel Interface

The component rio_chan_intf is implemented on the top level in the Framework Logic and corresponds to the red block in fig. 3.3. It encapsulates the components rio_clocks and four rio_intf (fig. 5.2) and is therefore the interface between the eight RocketIO transceivers and the Framework Logic.

The most important inputs and outputs of rio_chan_intf are listed in tab. 5.6. Appendix I shows the VHDL code of the rio_chan_intf.

<table>
<thead>
<tr>
<th>Port name</th>
<th>Direction</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLL</td>
<td>IN</td>
<td>Programmable PLL clock routed through to CLKin of the rocketio_wrapper_tile. The frequency has to be transfer rate / 20</td>
</tr>
<tr>
<td>DATAOUT</td>
<td>OUT</td>
<td>128 bit wide signal merged from the eight 16 bit wide outputs of the data receiver. It is sent to the data input of the VFIFO</td>
</tr>
<tr>
<td>DATA_RDY</td>
<td>OUT</td>
<td>Signal sent to the write enable input of the VFIFO. It indicates to the VFIFO when new data is available</td>
</tr>
</tbody>
</table>

**Table 5.6:** Inputs and outputs of rio_chan_intf
5.7 User constraints file

The final step to implement the RocketIOs is to connect the programmed logic to the hardware, i.e. every instantiated GTP Dual has to be assigned to the hardware core it represents [6]. This is done by inserting the following entries in the so called user constraints file (cam_intf.ucf).

\[
\begin{align*}
\text{INST rio\_channel\_interface/rio\_chan01/rocketio\_rx/gtp\_dual\_i} \\
\text{LOC=GTP\_DUAL\_X0Y0;} \\
\text{INST rio\_channel\_interface/rio\_chan23/rocketio\_rx/gtp\_dual\_i} \\
\text{LOC=GTP\_DUAL\_X0Y5;} \\
\text{INST rio\_channel\_interface/rio\_chan45/rocketio\_rx/gtp\_dual\_i} \\
\text{LOC=GTP\_DUAL\_X0Y6;} \\
\text{INST rio\_channel\_interface/rio\_chan67/rocketio\_rx/gtp\_dual\_i} \\
\text{LOC=GTP\_DUAL\_X0Y7;}
\end{align*}
\]

The first part of the entry starting with \texttt{INST} is followed by a hierarchy-path that identifies one specific GTP Dual instantiated in the logic. Every component is instantiated with a unique instantiation name. To keep it simple the instantiation names were neglected before. The hierarchy-paths consist of the instantiation names of the previously described components and point to the individual GTP Duals. For clarification, written with component names the hierarchy-path is:

\[
\text{rio\_chan\_intf/rio\_intf/rocketio\_wrapper\_tile/gtp\_dual}
\]

The reader might compare to fig. 5.2 and fig. 5.1.

The second part of the entry starting with \texttt{LOC} assigns the component specified by the hierarchy-path to its physical unit. As mentioned before, the FPGA integrated in the X5-TX provides eight GTP Duals, called GTP\_DUAL\_X0Y0 to GTP\_DUAL\_X0Y7 [6]. Four of them are in use for the PCIe [10]. The rest is enabled to receive the RTO signals by the entries in the user constraints file mentioned above.
6 Transceiver-Board

The Transceiver-Board (fig. 6.1) was developed to convert the optical signals from the camera to differential electrical signals for the RocketIOs. This job is done by the optical receiver \textit{ZL60102 RX} from \textit{Zarlink Semiconductors} which is the counterpart of the optical transmitter used in the cameras, see section 2.3.

![Transceiver-Board](image)

\textbf{Figure 6.1:} Transceiver-Board

As explained before the camera uses the eight outer channels of the optical transmitter. Hence, the electrical outputs of the eight outer channels of the optical receiver are routed to four eSATA connectors. Thus the Transceiver-Board can be connected to the X5-TX by common eSATA cables. Since the X5-TX is plugged in a PCIe slot of the host computer, the Transceiver-Board is equipped with a slot bracket and is also mounted in the computer. For power supply a receptacle was chosen that fits the power supply connectors available in the computer. The optical receiver is accessible through the slot bracket from outside the computer (fig. 6.2).

The Transceiver-Board is equipped with an MDR68 connector that routes its signals through to an MDR36 connector, accessible through the slot bracket. The MDR68 connector is meant to be connected to the MDR68 connector of the X5-TX. This was done to make the 16 digital inputs and outputs of the X5-TX accessible from outside the computer. At present the digital inputs and outputs are not used.
Figure 6.2: X5-TX and Transceiver-Board in the host computer

The schematic of the Transceiver-Board is shown in Appendix J and the mechanical drawing of the slot bracket is shown in Appendix K.
7 Results

The X5-TX with the modified Logic in combination with the Transceiver-Board form a receiver system for the RTO of the Phantom cameras. Data of all channels excluding the inter frame gaps is written to a binary file. Then MATLAB is used to assemble the data to an image.

A test image of 64 x 64 pixels from a chessboard was taken with the Phantom V7.2. Figure 7.1 shows this image. As mentioned before the RTO is sending raw images. Hence, every pixel has to be corrected by subtracting an offset and then multiplying a gain. The offsets and gains are different for every pixel and are provided in the .stg-files of each camera [4]. In the .stg-file, the pointer to the offsets is at address\(^1\) 0x34 and the pointer to the gains is at address 0x40. Both pointers are little-endian 4 byte numbers and represent the addresses where the first offset and the first gain are listed. The values are listed from left to right, starting with bottom row. MATLAB is used to perform the correction. Figure 7.2 shows the corrected image.

\[\text{Figure 7.1: Raw image} \quad \text{Figure 7.2: Corrected image}\]

---

\(^1\)The address of a byte is the number of the byte in the .stg-file, starting to count on zero.
8 Outlook

8.1 FPGA logic

The developed receiver system is now capable to receive the data output by the RTO. However, due to the limitation in the writing speed of common hard drives ($\approx 70 - 80 \text{ MB/s}$), images of high resolution recorded at high frame rates cannot be stored in real-time. Therefore future improvements will aim on performing a sparsification on the data stream.

As mentioned before, the images taken for Lagrangian Particle Tracking consist of large dark areas and very few bright pixels where particles are represented. Only the bright pixels are of interest for post processing and there is no need to save dark ones, too. Hence, the sparsification algorithm will be implemented in the X5-TX logic and compare received pixels to a threshold. Only if the pixel is brighter than the threshold it will be saved with its coordinate. The following example shows that the data rate will be reduced below the writing speed limit of hard drives.

If images with a resolution of 256x256 are recorded at the highest possible frame rate for the RTO of $14.67 \text{ kHz}$ (section 2.4) the data rate is:

$$\text{data rate} = 256^2 \cdot 14.67 \text{ kHz}$$

$$\text{data rate} = 961.41 \text{ MB/s}$$

This number represents pure image data after all K-Characters and overhead has been filtered out. It is too high to be written on hard drives.

The sparsification algorithm would write only the bright pixels with one coordinate to file. One coordinate is sufficient, since the information from which channel the pixel came is available and does not require memory. The size of this coordinate depends on the resolution and is $256^2/8 = 8192 \equiv 13 \text{ bit}$ for this example, if an 8 channel camera is used. Assuming 500 particles per frame and each represented by five 8 bit-pixel the data rate would be:

$$\text{data rate} = 500 \cdot 5 \cdot (8 \text{ bit} + 13 \text{ bit}) \cdot 14.67 \text{ kHz}$$

$$\text{data rate} = 96.4 \text{ MB/s}$$

This shows, using sparsification can reduce the data rate (and the amount of data produced by an experiment) drastically, 90% in this example. Even if the data rate is still too high for a common hard drive, a simple RAID0 system can easily handle data rates in that range. Hence, this improvement will make it possible to stream
only the needed data of high speed movies directly to the hard drive and increase the
duration of the experiment. A movie recorded into the $4\,\text{GB}$ RAM of the Phantom
V7.2 will last only:

\[
\text{experiment duration} = \frac{4\,\text{GB}}{961.41\,\text{MB/s}} \approx 4.16\,\text{s}
\]

While a movie recorded to the hard drive with a size of $4\,\text{GB}$ after sparsification will
last:

\[
\text{experiment duration} = \frac{4\,\text{GB}}{96.4\,\text{MB/s}} \approx 41.5\,\text{s}
\]

Moreover the size of the hard drive can be increased easily and with low costs. A hard
drive of $1\,\text{TB}$, full with sparsified data would contain 2.9 hours of a continuous ex-
periment. This will make it possible to acquire data efficiently as long as the particle
density in the measurement volume is high enough (fig. 1.3). Moreover slow processes
can be observed, together with the fast ones, with a high temporal resolution.

\section*{8.2 Host program}

The host program used at this stage of the project is the \textit{Wave Example} delivered by
Innovative Integration. Just like the X5-TX it was developed for D/A conversion of
data from a file. It is only used to start and stop the process of writing camera data
to file.

As mentioned before it is fully customizable. Hence, it will be adapted to the applica-
tion. The user could give information about the experiment like frame rate, resolution
and camera model to the program and therefore to the X5-TX. Then the FPGA can
configure itself to the number of channels in use, for example. Moreover the host
program could inform the user if the chosen frame rate is within the capability of the
RTO according to eq. 3.
8.3 Digital inputs and outputs

When the receiver system with sparsification is used in LPT experiments, four of these systems will be used - one for each camera. Then synchronization is necessary. As explained in section 6 the 16 digital inputs and outputs of the X5-TX are available on an MDR36 connector of the Transceiver-Board and are unused up to now. One of these inputs of every receiver system could be connected to an external device and be used as the trigger signal for all systems to start and stop recording simultaneous. This external device could be a simple button or another computer. If a computer is used, more digital outputs can be used and information of all four receiver systems can be monitored.

8.4 Cameras

At present the RTO outputs image data without saving them in the dedicated RAM of the camera. Due to this fact and the limited transfer rate of the RTO the maximum frame rates which can be achieved when the RTO is in use, are much lower than the maximum frame rates of the camera itself, as explained in section 2.4. That means, high speed movies taken at the limit of the cameras capability have to be stored in the RAM and can only be downloaded by Ethernet. The camera manufacturer confirmed to deliver firmware updates in the future that will allow the RTO to send images from the RAM. Then the user can chose to use the RTO with:

Live images: The frame rate is limited by the transfer rate of the RTO, but therefore experiment duration is only limited to the size of the hard drive.

Images from RAM: The possible frame rates are much higher, but the experiment duration is limited to the size of the RAM of the camera. However, the downloading of the recorded images is much faster than through ethernet.

In a cooperation with the camera manufacturer the sparsification could even be implemented in the camera. Then the RTO could output sparsified data.
## List of Tables

2.1 Transfer rates and channel numbers [4] ..................................... 6
2.2 Extract of the 8b/10b look-up-table [6] ................................. 7
2.3 Extract of K-Characters [6] ................................................... 8
2.4 Transfer protocol for 8 bit pixels [5] ....................................... 9
4.1 Attributes of the GTP Dual [6] ............................................. 18
4.2 Configuration ports of the GTP Dual [6] ................................. 18
5.1 Inputs and outputs of rocketIO_wrapper_tile [6] ...................... 21
5.2 Inputs and outputs of data_receiver ................................. 23
5.3 Outputs of data_sender .................................................... 24
5.4 Inputs and outputs of rio_intf ........................................... 25
5.5 Inputs and outputs of rio_clocks ....................................... 27
5.6 Inputs and outputs of rio_chan_intf ................................... 28
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>LPT on the Lagrangian Exploration Module [1]</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Measurement volume</td>
<td>2</td>
</tr>
<tr>
<td>1.3</td>
<td>Decrease of particle density in the measurement volume</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Camera block diagram</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Real Time Output</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>Readout sequence for 8 channels [4]</td>
<td>6</td>
</tr>
<tr>
<td>2.4</td>
<td>Readout sequence for 6 channels [4]</td>
<td>6</td>
</tr>
<tr>
<td>2.5</td>
<td>Transmission of packets</td>
<td>9</td>
</tr>
<tr>
<td>2.6</td>
<td>Data packet [5]</td>
<td>9</td>
</tr>
<tr>
<td>2.7</td>
<td>Optical transmitter ZL60101 TX [7]</td>
<td>10</td>
</tr>
<tr>
<td>2.8</td>
<td>Fiber cable with MTP/MPO connector</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>X5-TX with adapter card</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>Framework Logic block diagram</td>
<td>15</td>
</tr>
<tr>
<td>3.3</td>
<td>Framework Logic block diagram with RocketIO</td>
<td>17</td>
</tr>
<tr>
<td>5.1</td>
<td>RocketIO interface</td>
<td>22</td>
</tr>
<tr>
<td>5.2</td>
<td>RocketIO channel interface</td>
<td>26</td>
</tr>
<tr>
<td>6.1</td>
<td>Transceiver-Board</td>
<td>30</td>
</tr>
<tr>
<td>6.2</td>
<td>X5-TX and Transceiver-Board in the host computer</td>
<td>31</td>
</tr>
<tr>
<td>7.1</td>
<td>Raw image</td>
<td>32</td>
</tr>
<tr>
<td>7.2</td>
<td>Corrected image</td>
<td>32</td>
</tr>
</tbody>
</table>
References

[1] *The Lagrangian Exploration Module*  
Generation of homogeneous and isotropic turbulence with little mean flow for Lagrangian experiments  
Diploma Thesis  
Robert Zimmermann  
November 04, 2008

[2] *A quantitative study of three-dimensional Lagrangian particle tracking algorithms*  
N.T. Ouellette, H. Xu, E. Bodenschatz  
Experiments in Fluids, 40(2):301-313, 2006

[3] *Real-time image compression for high-speed particle tracking*  
K.Y. Chan, D. Stich, G.A. Voth  
Review of scientific instruments 78, 023704, 2007

[4] *Confidential information from Vision Research*

Vision Research  
R. Corlan  
Rev 1.0 July 01, 2005  
www.visionresearch.com

Xilinx  
UG196 (v2.0) June 10, 2009  
www.xilinx.com

[7] *ZL60101 TX / ZL60102 RX Data Sheet*  
12 x 2.7 Gbps Parallel Fiber Optic Link Transmitter and Receiver Data Sheet  
Zarlink  
Issue 1.1 January, 2003  
www.zarlink.com
[8] *Phantom V7.3 Data Sheet*
Vision Research
November 11, 2007
www.visionresearch.com

Innovative Integration
Rev 1.0 December 19, 2007
www.innovative-dsp.com

[10] *X5-TX User’s Manual*
Innovative Integration
Rev 1.1 April 22, 2009
www.innovative-dsp.com
Appendix J

Schematic of the Transceiver-Board
Appendix K

Mechanical drawing of the slot bracket