Master Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of microelectronics

# FPGA Based Readout System for Particle Detectors

Bc. Marek Janský

Supervisor: Ing. Jakub Jirsa
Field of study: Electronics and communications
Subfield: Electronics
May 2022

# ZADÁNÍ DIPLOMOVÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Janský** Jméno: **Marek** Osobní číslo: **466331**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Elektronika**

## II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

**Vyčítací systém s FPGA pro částicové detektory**

Název diplomové práce anglicky:

**FPGA Based Readout System for Particle Detectors**

Pokyny pro vypracování:

1. Seznamte se s vývojovou deskou AC701 a čipem ColorPix-0B.
2. Prostudujte problematiku multigigabitových transceiverů (GTP) v Xilinx FPGA.
3. V jazyce Verilog2001 implementujte vyčítací systém, který bude přijímat data z čipu ColorPix-0B přes GTP a následně je bude posílat přes rozhraní USB 3.0 do počítače.
4. V jazyce C++ vytvořte jednoduchou aplikaci, která umožní data ukládat do souboru.
5. Charakterizujte přenosový kanál (BER, jitter, eye diagram).
6. Zhodnoťte dosažené výsledky.

Seznam doporučené literatury:

[1] Xilinx AC701 Evaluation Board for the Artix-7 FPGA
[2] Xilinx 7 Series FPGAs GTP Transceivers
[3] Future Technology Devices International Ltd. FT600Q-FT601Q IC Datasheet

Jméno a pracoviště vedoucí(ho) diplomové práce:

**Ing. Jakub Jirsa katedra mikroelektroniky FEL**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **31.01.2022** Termín odevzdání diplomové práce: **20.05.2022**

Platnost zadání diplomové práce: **30.09.2023**

| Ing. Jakub Jirsa | prof. Ing. Pavel Hazdra, CSc. | prof. Mgr. Petr Páta, Ph.D. |
|---|---|---|
| podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | podpis děkana(ky) |

## III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. Datum převzetí zadání

Podpis studenta

# Acknowledgements

I whould like to thank Ing. Jakub Jirsa for his patience and consultations during making of this thesis. Furthermore I would like to thank my family, friends and especially my fiancee for their unprecedented support during my studies at the university.


# Declaration

I declare that I completed the presented thesis independently and that all used sources are quoted in accordance with the Methodological instructions that cover the ethical principles for writing an academic thesis.

Prague, 20th of May, 2022

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 20. května 2022

# Abstract

The aim of this work is the implementation of high-speed readout system for X-ray particle detectors. In the first part, the reader will get acquainted with the principle of X-Ray particle detection and the topics of FPGAs, USB SuperSpeed standard, and multigigabit serial communication. The second part will be focused on implementation of readout system on Artix-7 FPGA and the characterization of the transmission channel and its error rate. Finally, the measured result will be discussed.



**Keywords:** FPGA, gigabit transceiver, GTP, readout system, USB



**Supervisor:** Ing. Jakub Jirsa
Faculty of Electrical Engineering, CTU
Technická 2
160 00 Prague 6 - Dejvice



# Abstrakt

Cílem této práce je realizace vysokorychlostního vyčítacího systému pro částicové detektory. V první části práce se čtenář seznámí s principem detekce rentgenových částic, bude uveden do tématu programovatelných hradlových polí (FPGA), USB SuperSpeed specifikace a do problematiky vysokorychlostních sériových komunikací. Druhá část bude zaměřena na implementaci vyčítacího systému na FPGA Artix-7 a charakterizaci přenosového kanálu včetně jeho chybovosti. Zjištěné výsledky budou diskutovány v závěru.



**Klíčová slova:** FPGA, gigabitový transceiver, USB, vyčítací systém

**Překlad názvu:** Vyčítací systém s FPGA pro částicové detektory

# Contents


**1 Introduction** 1

**2 Applied technology** 3

2.1 Particle detectors . . . . . 3

2.1.1 Interaction of X-Ray photons with matter . . . . . 4

2.1.2 Semiconductor X-Ray detectors 6

2.1.3 Detector frontend electronics . 8

2.2 FPGA . . . . . 10

2.3 USB 3 . . . . . 12

**3 Gigabit Transceivers** 15

3.1 Introduction to multi-gigabit data transmission . . . . . 15

3.1.1 Differential signalling . . . . . 15

3.1.2 Clock synchronisation . . . . . 16

3.2 Physical layer . . . . . 17

3.2.1 IO driver . . . . . 17

3.2.2 Termination . . . . . 18

3.2.3 Clock recovery . . . . . 18

3.2.4 Serialisation and deserialisation . . . . . 18

3.2.5 Alignment . . . . . 18

3.3 8B/10B coding scheme . . . . . 19

3.4 Physical link adaptation . . . . . 19

3.5 State of the art . . . . . 20

3.6 Gigabit transceivers on Xilinx 7 series devices . . . . . 21

3.6.1 Overview . . . . . 21

3.6.2 GTP transceiver on Artix-7 . 21

3.6.3 Physical Medium Attachment 22

3.6.4 Physical Coding Sublayer . . . 24

3.6.5 FPGA interface . . . . . 26

**4 Implementation of readout system** 27

4.1 ColorPix-0B . . . . . 27

4.1.1 Integrated gigabit transmitter 28

4.2 AC701 Evaluation board . . . . . 29

4.3 FPGA readout system . . . . . 30

4.3.1 Readout system clocking . . . . 31

4.3.2 GTP transceiver . . . . . 32

4.3.3 Data pattern generator . . . . . 34

4.3.4 Data frame construction . . . . 34

4.3.5 FTDI USB3 interface . . . . . 38

4.3.6 Command interface . . . . . 40

4.3.7 Message parser . . . . . 40

4.3.8 DDR memory buffering . . . . . 40

4.4 Host readout GUI application . . 41

4.4.1 Data acquisition . . . . . 42

4.5 Design reports . . . . . 44

4.5.1 FPGA utilization . . . . . 44

4.5.2 Implementation timing report 44

4.5.3 Power report . . . . . 44

**5 Transmission characterization** 47

5.1 Eye pattern measurement . . . . . 48

5.2 Jitter measurement . . . . . 49

5.3 Bit Error Rate . . . . . 50

**6 Conclusion** **53**

6.1 Discussion . . . . . 53

6.2 Future improvements and next steps . . . . . 54

**A Bibliography** **55**

**B List of abbreviations** **59**

**C GTP Wizard configuration** **61**

**D Attached files structure** **65**

# Figures


2.1 Detector and readout system diagram . . . . . 3

2.2 Photoelectric effect [4]. . . . . 5

2.3 Pair production [4]. . . . . 5

2.4 Compton scattering [4]. . . . . 6

2.5 Regions of importance for three main particle interactions used for detection [5]. . . . . 6

2.6 Monolithic particle detector, integrating sensing element with a part of front end electronics on a single piece of silicon [7]. . . . . 7

2.7 Hybrid architecture of particle detector [8]. . . . . 8

2.8 A simplified readout front end architecture [8] . . . . . 8

2.9 USB High Speed (HS) vs Super Speed (SS) data transaction, where Host requests data from the device [12] . . . . . 13

2.10 Physical layer of USB Super Speed [12] . . . . . 13

3.1 HSTL driver and input buffer diagram [14] . . . . . 16

3.2 Differential signaling [14] . . . . . 16

3.3 Basic diagram of self-synchronous data transmission [14] . . . . . 17

3.4 CML logic buffers [15] . . . . . 17

3.5 AC coupled CML link [15] . . . . . 18

3.6 Effect of signal pre-emphasis [14] 20

3.7 Eye diagram of NRZ vs PAM4 modulation [17] . . . . . 21

3.8 Topology of GTP transceiver [16] 22

3.9 Block diagram of GTP output buffers with preemphasis drivers [16] 23

3.10 Receiver equalization [16] . . . . . 23

3.11 Receiver analog front end termination [16] . . . . . 24

3.12 Concept of comma alignment [16] . . . . . 25

4.1 ColorPix-0B bonded to a PCB . 27

4.2 ColorPix-0B testing board . . . . . 28

4.3 Block design representation of ColorPix-0B gigabit transmitter . . 29

4.4 AC701 board [20] . . . . . 30

4.5 Simplified system architecture . . 31

4.6 FPGA top entity architecture . . 31

4.7 Design clocking diagram . . . . . 32

4.8 Block diagram of the frame composer module . . . . . 34

4.9 Frame packet protocol . . . . . 35

4.10 State flow diagram of frame composer state machine . . . . . 36

4.11 Signal timing [21] . . . . . 37

4.12 Brute force synchronizer using 2 stages of registers . . . . . 37

4.13 Asynchronous FIFO structure [21] . . . . . 38

4.14 Block diagram of FT600/601 interface chip [22] . . . . . 39

4.15 FTDI 600 bus read transaction timing diagram . . . . . 39

4.16 Message parser state flow . . . . . 41

4.17 GUI readout application written in C++ and Qt framework . . . . . 42

4.18 State flow diagram of data acquisition application . . . . . 43

4.19 Timing report exported after implementation in Vivado toolchain 44

4.20 Reported power draw graph . . . 45

5.1 Measurement workspace . . . . . 47

5.2 Eye diagram without preemphasis 48

5.3 Eye diagram with preemphasis . 48

5.4 Histogram of number of error bits per frame . . . . . 51

C.1 GTP - transceiver selection . . . . 61

C.2 GTP - line rate, clock reference 62

C.3 GTP - encoding configuration . . 62

C.4 GTP - comma detection . . . . . 63

C.5 GTP - configuration summary . 63

# Tables

3.1 Transceivers offered on Xilinx FPGA devices, sourced from [18] . 21

3.2 Table with a sample of the valid 8B/10B encodings . . . . . 25

4.1 GTP configuration summary . . . 33

4.2 Command protocol. All values are in hexadecimal notation and letter x signifies command argument. . . . . 40

4.3 Device utilization report . . . . . 44

5.1 Measured characteristics of eye diagrams . . . . . 49

# Chapter 1

# Introduction

Medical X-Ray imaging dates back into the 20th century. Its principle is based on photon energy attenuation in various materials. X-Ray photons have to be detected after passing through object of interest in order to produce representative image. Most current medical X-Ray imaging systems use film detectors. They are however limited in resolution, sharpness and sensitivity.

Pixel detectors are therefore being developed and slowly applied. Advancements in particle detection that enabled more extensive experiments in the field of particle physics have brought up other applications to utilize those technologies. Pixel detectors are employed almost exclusively in computer tomography. This however leads to increased demands on bandwidth of those systems. Fast digital readout is therefore needed, as it provides the necessary transmission speeds. High speed readout can significantly reduce patient exposure time and allows transmission of intensity image as well as photon energy, which will further improve medical X-Ray diagnosis.

This thesis is focused on implementation and characterization of the readout system for particle detectors using high bit rate transmission. This system is implemented on field programmable gate array with integrated multi-gigabit transceivers, which communicates with PC by the use of USB 3.0 SuperSpeed interface. To control the readout system a graphical application is developed. This application manages the USB 3 communication and ensures storing of received data. The system is designed to handle data rate of 3200 Mbps.

# Chapter 2

# Applied technology

This chapter is a description of used technology stack. Design requirements will be briefly discussed and from those a particular hardware will be introduced.

Principal system components are depicted on figure 2.1. The signal chain starts with detector with ASIC (application specific integrated circuit) frontend chip. It is connected to a readout system, which consists of FPGA and USB interface transceiver. The transceiver is connected to a PC that is receiving the data stream.

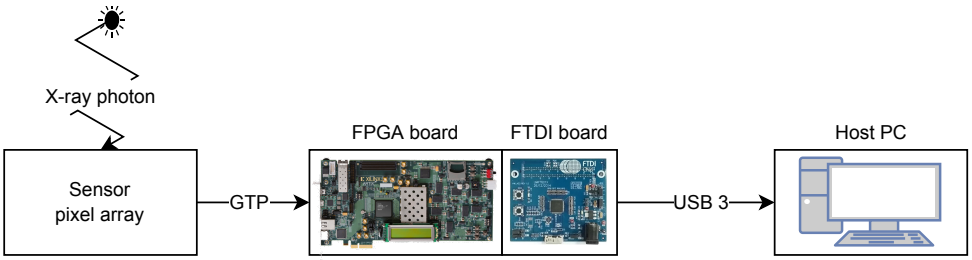



**Figure 2.1:** Detector and readout system diagram

## 2.1 Particle detectors

Particle detectors are devices used to detect and measure particles. Their applications range from medical X-Ray detectors to scientific instruments used in large experiments. Historically detectors were gas based. Geiger counter was one of such devices. They comprise from a tube filled with inert gas and electrodes to which high voltage is applied. If incident particle enters the tube, it causes the gas to briefly conduct due to ionization. These peaks in current can than be counted or integrated. In medical applications,

mostly film detectors are used and new techniques and materials are still under development. However digital screening is despite its infancy on the rise, because it provides novel advancements in radiography applications [1].

### 2.1.1 Interaction of X-Ray photons with matter

This section will describe how particles interact with matter. These are basic principles currently used in X-Ray particle detectors. They determine sensitivity and efficiency of detection.

#### Cross section

The main indicator of the level of interaction is cross section. It quantitatively describes the probability of incident photon interacting with another particle. Cross section is highly dependant on the atomic number of absorbent material. Generally the cross section increases with Z value of material (the material is denser) and decreases with incident particle energy [2].

#### Photoelectric effect

Photoelectric effect (depicted on figure 2.2) is a process where the photon interacts with an electron in the inner atomic shells and completely disappears. Its energy is absorbed by the atom. An energized photo electron is then ejected from the atom shell. Its energy is

$$E_{e^-} = h\nu - E_b, \tag{2.1}$$

where $E_b$ is a binding energy of the released particle, $h$ is the Planck constant and $\nu$ is the frequency of incoming photon [3]. Created vacancy of the ionised atom is then immediately filled by rearrangement of electrons from other shells. The rearrangement leads to either emission of fluorescence photons or release of other electrons from the shell [2]. Generally the incident cross section of this process increases with atomic number of target material and decreases with the photon energy.

#### Pair production

A high energy gamma rays can produce electron-positron pairs if the interaction with electrons is under the influence of strong Coulomb field of the nucleus. There is minimal interaction of the photon and nucleus, almost all energy is shared as kinetic energy between the positron and electron. The interaction is depicted in figure 2.3. Pair production only applies to very high energy particles and those do not occur in medial applications.

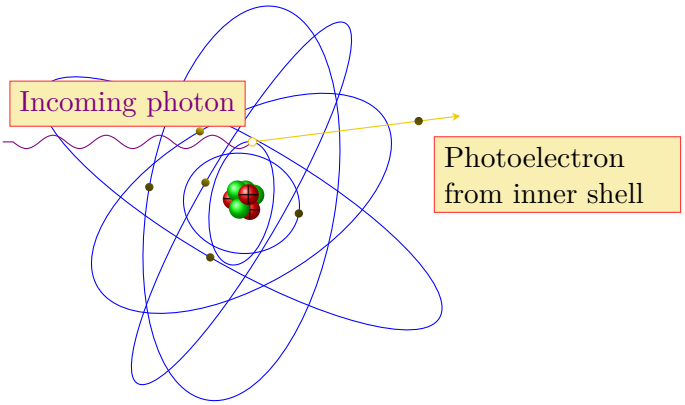



**Figure 2.2:** Photoelectric effect [4].

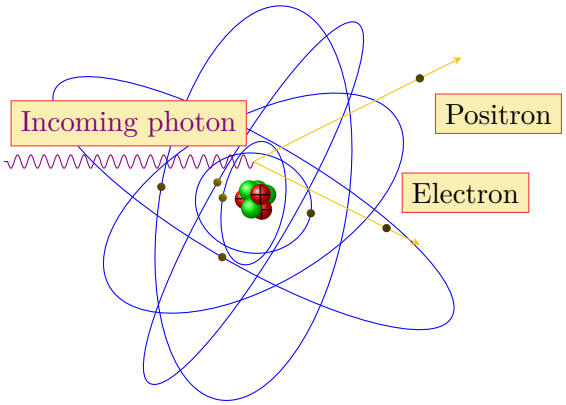



**Figure 2.3:** Pair production [4].

### Scattering

There are two types of scattering: Compton and Rayleigh.

Compton scattering (figure 2.4) is interaction between photon and electron. The energy and momentum that is lost by the photon is transferred to an electron which is emitted at an angle. The photon is scattered from its original trajectory at a different angle. The maximum energy that the electron can gain occurs when the photon is backscattered and electron is emitted in the same trajectory as the incident photon.

Reyleigh scattering is also an interaction between a photon and electron, however the electron is tightly bound to the atom. The photon is slightly deflected and the whole process is elastic, meaning there is no transfer of energy.

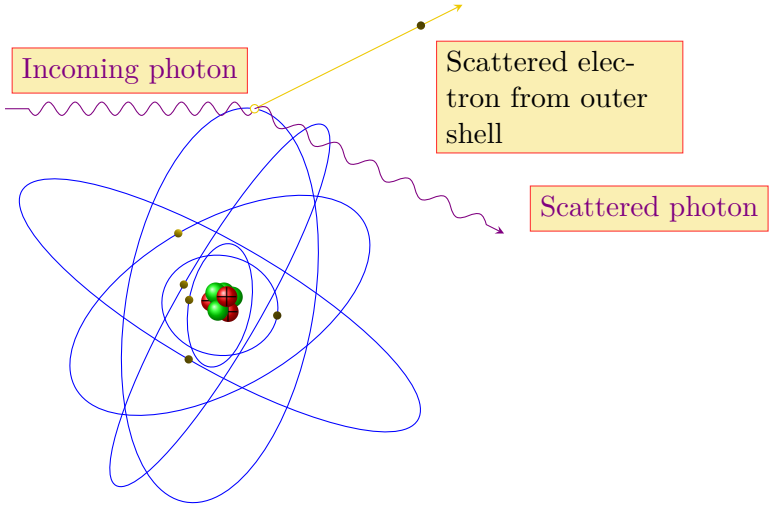



**Figure 2.4:** Compton scattering [4].

### X-ray photon interactions with matter

X-Ray particles which are in the range of hundreds to thousands of hundreds would have the best cross section with high Z materials. Thanks to bigger cross section we gain increased sensitivity to lower energy particles. Figure 2.5 depicts comparison and importance of each process depending on particle energy and material atomic number.

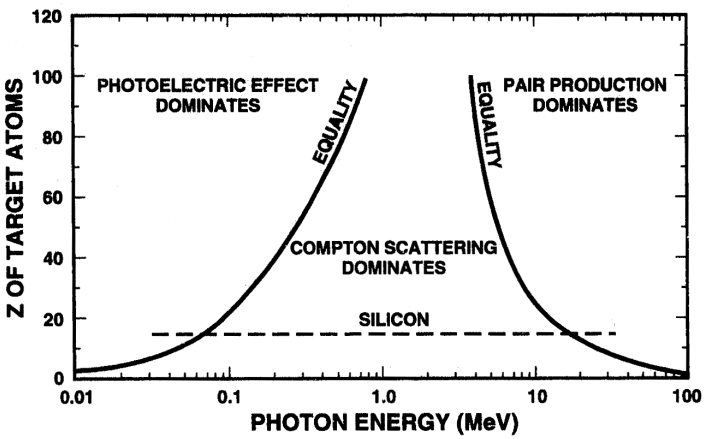



**Figure 2.5:** Regions of importance for three main particle interactions used for detection [5].

## 2.1.2 Semiconductor X-Ray detectors

Previously described processes enable us to detect X-Ray photons for medical imaging in semiconductor materials. Semiconductor detectors are usually

implemented as reverse biased p-n diode [2]. When X-Ray photon interacts it looses its energy. This energy is transferred to other particles, which ultimately excite electrons to the conduction band. That creates electron-hole pairs which under the applied electric field drift through the semiconductor. This current can be than detected.

There are 2 main types of detectors, direct and indirect. Indirect detectors first convert the photon particle into visible light with scintillator materials. The visible light is then detected in a pixelated matrix of photo diodes. However this approach is not efficient due to the two steps. In direct detectors the photons directly create electron-hole pairs inside the semiconductor. These charges then drift in the detection material until they are collected with electrodes.

Most currently used X-Ray detection systems, be it film or digital ones, rely on charge integration - energy collected in the pixel is integrated over a defined exposure time. It means that most of the single photon relevant information is lost in the exposure and integration is also more susceptible to noise. This limits dynamic range and signal to noise ratio of the detection system [6]. Novel method to capture photon interactions is called single-photon counting. It is based around a charge sensitive amplifier and comparator.

There are also two ways to implement a detector system. The first one is Monolithic (on figure 2.6), where the same die is utilized for both sensing and readout electronics. This has the benefit of reduced complexity of manufacturing in commercially available processes. The detector is manufactured in a single process and no sensor bonding is requires as opposed to the hybrid systems. The disadvantage is that it uses silicon photo diodes and those have to be integrated into the sensing pixel, which limits space.

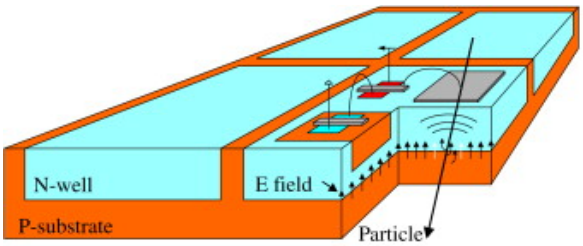



**Figure 2.6:** Monolithic particle detector, integrating sensing element with a part of front end electronics on a single piece of silicon [7].

Hybrid detectors on the other hand comprise of separate sensing layer which can be manufactured from specialized semiconductor. Its electrodes are connected by solder bumps to the readout ASIC as seen on figure 2.7. Front end and readout ASICs are then manufactured usually in a standard CMOS process.

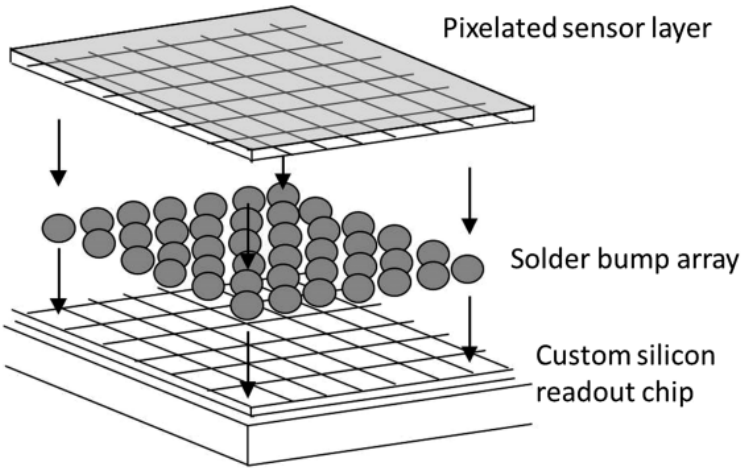



**Figure 2.7:** Hybrid architecture of particle detector [8].

### 2.1.3 Detector frontend electronics

Readout systems are ASICs responsible for conversion of electric current present on the sensing element into a digital signal that can be transferred for further processing. A simplified diagram is depicted on figure 2.8. It consists of a charge sensitive amplifier, pulse shaping circuitry and discriminator.

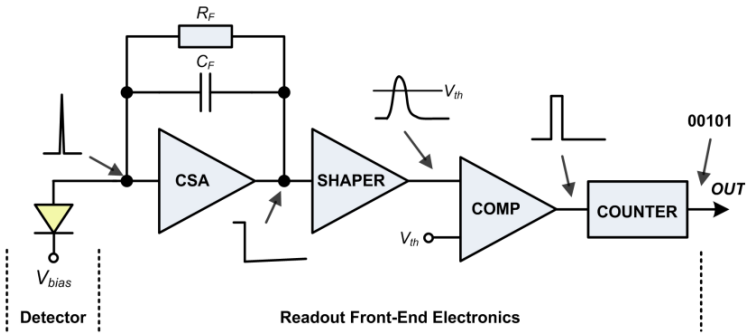



**Figure 2.8:** A simplified readout front end architecture [8]

The discriminator is a comparator that compares detected shaped current pulse with a threshold. If the pulse exceeds the set threshold, counter is incremented. Sometimes there can be multiple levels of such discrimination - these systems are then called spectroscopic and photons with different energies can be distinguished.

After the counter we are strictly in digital domain. Here all columns of pixels are connected in a chain, that ends up in transmit shift registers.

The need for high readout arises from the nature of the detection. For high flux beams, there are lots of particles that interact with the sensor material

and are measured. If the flux is too high, the counter can overflow. Also, the bandwidth needed increases with spatial pixel matrix resolution of the detector.

Count-rate is an indicator of how many pulses (photon interactions) are approximately measured. It is limited by the speed of charge sensitive amplifier. To prevent charge pileup or saturation, the time for each detection event is $1\,\mu$s. The maximum count rate is than derived as

$$Count\ rate\left[\frac{Mcps}{mm^2}\right] = \frac{1}{1\times10^{-6}\,\mathrm{s}} = 1\,\mathrm{Mcps}. \tag{2.2}$$

This would mean that for Colorpix-0B, which has a pixel matrix of 256 by 256 pixels, we would have countrate at maximum

$$Count\ rate = 256^2 \cdot 1\,\mathrm{Mcps} \approx 65{,}6\,\mathrm{Gcps}. \tag{2.3}$$

If we considered 12-bit counter for each pixel, the time until it would overflow would be

$$t_{ovf} = \frac{Counter\ size}{Count\ rate} = \frac{2^{12}}{1\,\mathrm{Mcps}} = 4{,}096\,\mathrm{ms}. \tag{2.4}$$

That implies that we would need to send 12 bits per counter every 4 ms, which equals to

$$BW_{counter} = \frac{Number\ of\ bits}{t_{ovf}} = \frac{12}{4.096e-3} \approx 3000\,\mathrm{bits\ per\ second}. \tag{2.5}$$

As there are multiple discriminators in the Colorpix detector each one has to have its own counter. There are totally 10 counters per each pixel. Total pixel generated data is therefore

$$BW_{pixel} = BW_{counter} \cdot 10 \approx 30\,\mathrm{kbps}. \tag{2.6}$$

If we multiply that by the total number of pixels, we get the total data rate of the whole detector as

$$BW_{detector} = Number\ of\ pixels \cdot BW_{pixel} = 256^2 \cdot 30000 \approx 1920\,\mathrm{Mbps} \tag{2.7}$$

After a dead time is considered, we can approximate the minimum required bandwidth to be at least 2,5 Gbps. The closest standard bitrate is 3,125 Gbps, however Colorpix-0B detector uses non-standard link speed of 3,2 Gbps because of the PLL feedback divider implementation.

## 2.2 FPGA

FPGAs (Field Programmable Gate Arrays) are complex devices that enable developers to implement virtually any logic circuit. These devices enable massive parallelization of data processing and offer dynamic reconfiguration at deployment. Because of that they are mainly used in high speed communications, data processing, space applications and in last few years in machine learning.

FPGA has been chosen in the readout application because of the ability of high parallelization as opposed to general processor systems. Large data throughput combined with reconfigurability makes it the ideal choice for the readout system implementation.

Development on FPGAs is done generally by creating logic modules in hardware description languages, such as Verilog, System Verilog or VHDL. These modules are then synthesised into a netlist, which represents connections between available hardware modules. Next step of the development flow is implementation, where the synthesised netlist is mapped onto real hardware. This process is alternatively called placing and routing a design. Implementation has to comply with constraints, which define timing, placement, clocking and input and output restrictions. Usually the developement tools optimise the design before implementation, reducing nets and cells that wouldn't have any function but were synthesized anyway. The final step is mapping the implemented design onto corresponding device and creating its configuration bitstream.

Most FPGAs today manufactured are SRAM based which means that their configuration is volatile. Each part, be it routing multiplexers, lookup tables or block memories are configured with the bitstream. When bitstream is loaded onto the device, all configuration cells are configured as was implemented in design. Most of the time FPGAs are connected to non-volatile storage, from which they configure themselves at powerup. There exist advanced schemes for bitstream encryption and intellectual property safety.

Field programmable arrays can contain hard IP modules, which means that the module is implemented on silicon and its behaviour can't be modified, only configured.

### Logic cell

The basic building blocks are look up tables (LUT) and registers. These together with carry logic and flip flops create a logic cell. On current Xilinx FPGAs LUTs can be configured as shift registers or memory. They are

also using LUTs which have 6 inputs. That greatly reduces complexity of synthesised designs.

On Xilinx 7 series devices the logic is organised into Configurable Logic Blocks (CLB) [9]. Each of these contains 2 logic slices. Logic slice comprises of:

- Four logic-function generators (or look-up tables)
- Eight storage elements
- Wide-function multiplexers
- Carry logic

### Memories

Almost every design needs a relatively large memory which wouldn't be synthesisable from distributed storage elements. There are two main approaches for using memory in FPGA designs. The first one is to use block memories which are integrated on the device between logic cells. These memories are usually segments of SRAM that can work with synchronous or asynchronous clocks, have at least 2 ports and can work in different modes, for example as an asynchronous FIFO buffer. They can be chained together to create one continuous memory. Xilinx 7 series devices have 36 Kb block memories that can be split to two independent 18 Kb memories. [10]

The other approach is to connect external memory, usually SDRAM. Although other memory technologies can be used, SDRAM is the most used one. Some vendors even include hard IPs to manage the external memories. Dynamic memory is cheaper then the static one, but it needs constant refreshing. That is task of memory controller, which acts as a bridge between memory and internal addressed bus. Memory controllers can support error code correction, automatic interface calibration and so on, usually at the cost of device utilization.

### I/O

FPGA is equipped with wide range of different input/output buffers, serializers and deserializers, signal delay compensators and more. Many times FPGA manufacturers add dedicated hard modules for DRAM, SPI and other interfaces which are critical and standardized. Lately it has become standard to include multigigabit transceivers into the FPGAs. They will be described in more depth later in this work.

Input/output buffers support almost all regular standards. They can behave as differential drivers with specified voltage swing, in classic single ended mode and high impedance output can be enabled [11].

### Clock management and PLL

Every synchronous design needs a clock signal. FPGAs have specialized clock hard IPs and clock routing networks to help with that. Clock management IPs are used to synthesise arbitrary clock frequencies and reduce jitter. They also define the clock routing fabric in the FPGA.

## 2.3 USB 3

This section will shortly describe USB 3.0 specification. USB 3.0 is a standard introduced in 2008 and adds Super Speed (SS) capabilities to the bus. It is an incremental upgrade introducing higher bandwidth up to 5 Gbps, improved power state management, better data flow control and overall robust error handling [12]. These features enable real-time video and audio applications with reduced latency and mass storage devices with faster file transfers.

USB 3.0 preserves previous standard to assure compatibility of the lower speed devices. It is solved by integrating the newer standard into the old one by essentially adding 4 wires - 2 differential pairs in dual simplex mode. These standards are coexistent in the bus. Each of the pairs is responsible for one way of data transfer as opposed to USB 2.0, where one differential pair was used in half duplex mode. Every USB Super Speed device must include the 2.0 specification, because it is used for endpoint enumeration and USB protocol overhead [12].

The Super Speed standard employs multigigabit serial transceivers. That means that proper synchronization is needed as the clock is embedded into the signal. 8b10b encoding as well as scrambling is used to ensure DC balance of the link. With increased link speed protocol enhancements are needed to manage the higher data rate. The new standard introduced burst transfers, unicast transactions and better flow management [13].

Figure 2.9 shows comparison between HS and SS data transactions. USB High Speed host device has to poll for every data packet whereas Super Speed host requests data with ACK message the device sends the requested data. Host then acknowledges the reception. To reduce latency, SS devices can use burst mode. In burst mode the device sends multiple data chunks as soon as they are available even if they are not the length that was requested and the device doesnt wait for ACK messages. Special message headers are employed

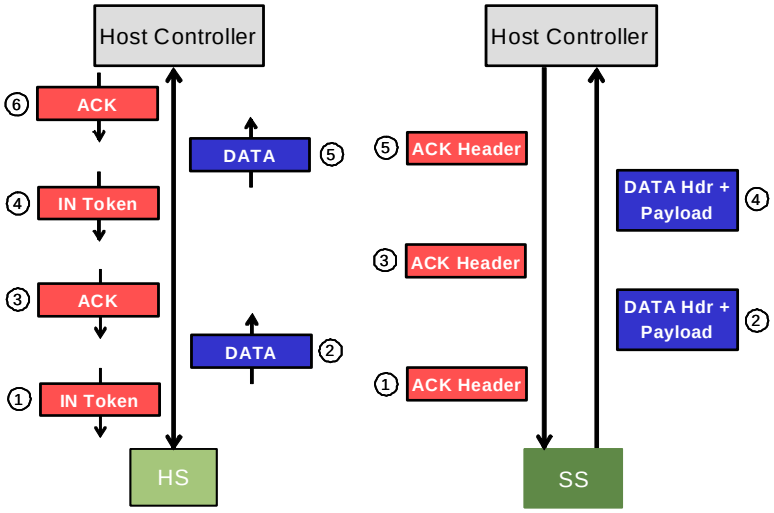



**Figure 2.9:** USB High Speed (HS) vs Super Speed (SS) data transaction, where Host requests data from the device [12]

to ensure correct protocol handling. After the burst is complete, host sends ACK message and can request another chunk of data. In addition Super Speed devices can asynchronously notify Host that new data is ready.

The physical layer of Super Speed devices is described in figure 2.10. TX part of the physical layer is responsible for data scrambling, 8b10b encoding into symbols, serialization and differential transmission. It also employs low frequency periodic signaling, which is used when the link is in idle state achieving receiver synchronisation.

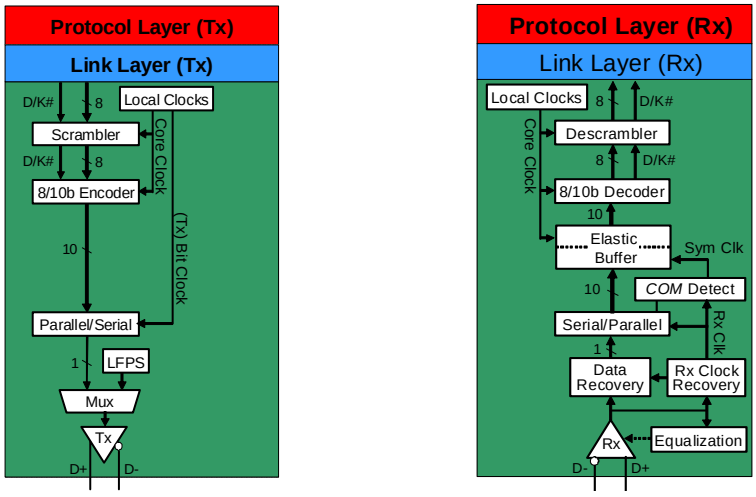



**Figure 2.10:** Physical layer of USB Super Speed [12]

Receiver physical layer receives the differential signal which can be equalized. Clock and data recovery is then performed, transitioning from serial to

parallel data path. The data then passes through elastic buffer, which absorbs variations between recovered and receive clock signals. Before reaching next layer the data is 8B10B decoded and unscrambled. The scrambler used is identical as the one employed in PCIe 2.0.

# Chapter 3

# Gigabit Transceivers

## 3.1 Introduction to multi-gigabit data transmission

Data transmission between devices has always been crucial part of electronic applications. However with advancements in technology communication bandwidth requirements arose. Smaller space and pin count constraints, tougher EMI restrictions paired with low cost and parallel data transmission wasn't feasible anymore. Data path synchronisation also became a problem and was not maintainable anymore with parallel technologies.

As silicon chip development advanced, smaller transistors enabled higher frequencies. With those the option to serialise data into higher speeds arose. Serial links provide solution to all mentioned problems that parallel busses had. Since they are almost exclusively differential, they not only contain electromagnetic fields better but they are much less susceptible to external influence. Lower space and simpler wiring are other benefits.

Gigabit transceivers are modules that enable high speed (>1 Gbps) data communication. They use differential pairs for simplex serial transmission. The main characteristic is that the clock signal is embedded into data line by means of data encoding.

### 3.1.1 Differential signalling

Single ended transmission was standard for a long time. In these systems only one connection is needed to transmit signal between 2 nodes. Based on implemented technology (for example TTL or CMOS), this signal is compared with a defined threshold range or reference. Output of this comparison determines logical level of the signal. These systems have however a number of pitfalls. One is that additive noise on the signal can spontaneously trigger

the comparator and register a false bit. Designers must also take into consideration the switching current. If number of single ended outputs are triggered simultaneously, large current spikes are generated in the design. Low voltage swing standards (HSTL depicted on figure 3.1) were introduced to combat some of the issues, but around the same time differential signalling was starting to emerge in PCB level communication.

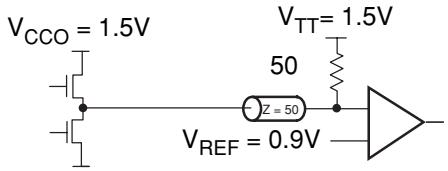



**Figure 3.1:** HSTL driver and input buffer diagram [14]

As imperfections of single ended transmission were becoming obvious, more designers realised that differential signalling, previously used only for long range communication, could be applied even for chip to chip communication on a single board. It uses 2 complementary signals that are transmitted together, as can be seen on figure 3.2. At first it solves biggest disadvantages of single ended standards. As opposed to single ended connection it is much less susceptible to common mode noise because the logic level is derived from difference of two signals. Also, since there is always equal current flowing into the link, differential switching generates much less current noise.

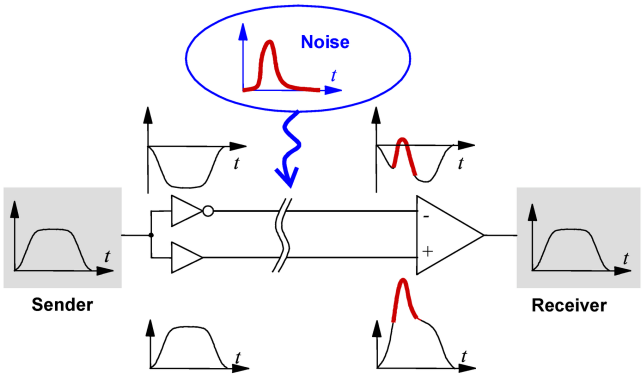



**Figure 3.2:** Differential signaling [14]

### 3.1.2 Clock synchronisation

High speed communication has inherent problems with synchronisation. Signals travel with non infinite velocity and the signal speed is dependant on dielectric surrounding its electric field. This means that every signal travelling in a channel has some propagation delay. For lower speed signals the delay is not significant since it is usually much less than the signal period. For

signals that have bit period of 1 ns or less, skew delay becomes a concern. Managing delay compensation in hardware is thereafter difficult because of manufacturing tolerances and complex verification.

Multi-gigabit systems therefore employ technique called self-synchronisation. The clock signal is embedded into data by means of data encoding and scrambling. These techniques will be described in the following sections. Figure 3.3 shows diagram of said self synchronous system.

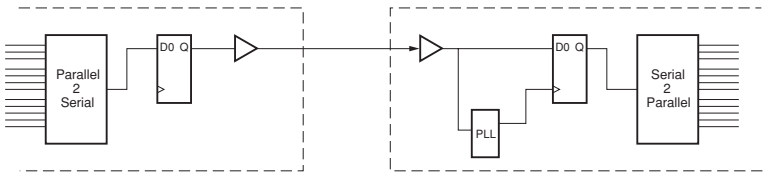
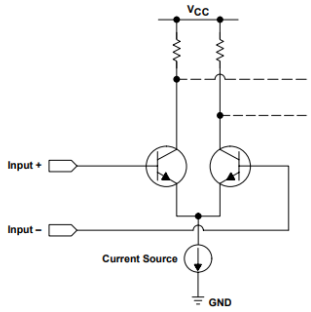



**Figure 3.3:** Basic diagram of self-synchronous data transmission [14]

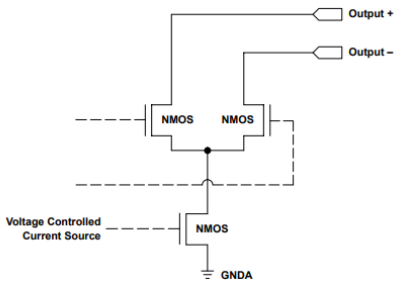
## 3.2 Physical layer

### 3.2.1 IO driver

Gigabit transceivers use differential physical signalling and there are three common differential standards - Low Voltage Differential Signalling (LVDS), Low Voltage Pseudo Emmiter-Coupled Logic (LVPECL) and Current Mode Logic (CML). CML is deployed most often for high speed gigabit links because it offers either DC or AC termination and output drive strength can be selected. Output driver implementation of CML driver and receiver are shown on figures 3.4a and 3.4b respectively.




**(a) :** CML input stage




**(b) :** CML output stage

**Figure 3.4:** CML logic buffers [15]

### 3.2.2 Termination

Every transmission lane has to be properly terminated and that applies especially to CML. The transmission line can be DC or AC coupled. AC coupling means there is an external capacitor connected in series on the link, which works as a high pass filter. It is sometimes called blocking capacitor.

Termination impedance is connected to the common mode voltage at the receiver side to add offset back after it was blocked by AC coupling capacitor. This connection is shown in figure 3.5.

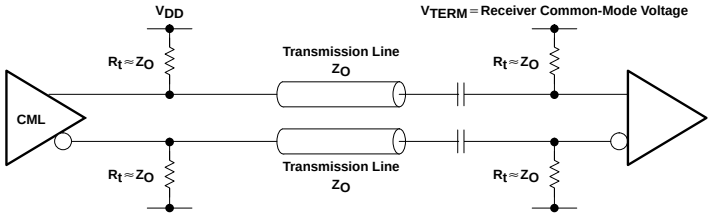



**Figure 3.5:** AC coupled CML link [15]

### 3.2.3 Clock recovery

To properly sample data in receiver, clock signal has to be derived and synchronized to incoming data. Phase interpolators are used for that purpose. Since the incoming line rate will never be identical to reference clock signal, there interpolators produce finely spaced pulses which are used in deserialisers.

### 3.2.4 Serialisation and deserialisation

To get from parallel to serial data shift registers are usually used. There are two regularly used types, one is parallel input and serial output (PISO), the other one is serial input, parallel output (SIPO). There are multiple possible implementations however the most basic one is with a shift register clocked with recovered frequency for receiver part. The transmitter part is directly clocked from reference PLL. To reduce clock frequency multiple phases of lower clock can be used to clock the registers.

### 3.2.5 Alignment

In gigabit receiver the data has to be somehow aligned to byte or word boundaries. There are special control words called K characters which serve exactly that purpose. If input register detects valid starting K word, it will push out its contents and align boundaries to that character.

## 3.3 8B/10B coding scheme

As was previously explained, in multi gigabit communications the clock is embedded into transmitted data. The most crucial part in that process is adding data redundancy with encoding. 8B10B encoding means that for every 8 bits of data, 10 bits are created, increasing overhead by 25 %. Each encoded data byte is called a symbol.

Encoding schemes not only ensure enough transitions in each symbol, they also manage balanced DC state of the link. Every encoded data word has a positive and negative variant. They are usually bitwise complementary. Positive symbols have more bits in ones, negative symbols have more zero bits.

Disparity or running disparity is a metric signifying current DC state of the link. The negative or positive symbols are selected based on current running disparity. If it is high, the negative symbol is chosen. If the disparity is positive, negative symbol is selected.

8B10B encoding also defines control symbols. These are called K characters and are generally used for receiver word alignment. However higher level protocols utilise them as flags or marks. They can signify start of larger continuous data packet or out of band signaling, as well as acknowledgements. Encoders and decoders use additional signal to select if data is K character or regular data.

## 3.4 Physical link adaptation

### Pre-emphasis

Pre-emphasis is a channel optimisation technique which compensates for intersymbol interference (ISI). ISI occurs when transition follows after longer period of bits that have the same value. Pre-emphasis circuit creates intentional distortion to overdrive the channel. If implemented correctly, it should have positive effect on eye pattern opening height, which means that the variance of each logic level is lower. Simple pre-emphasis can implemented with two CML output drive stages connected in parallel, where one is delayed by one bit and its polarity is reversed. It's effect is visible on figure 3.6.

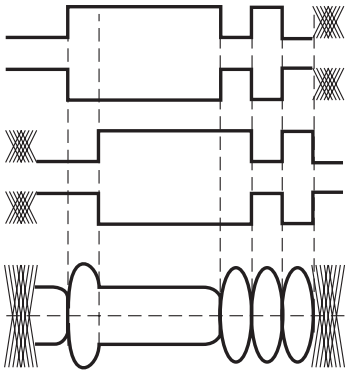


**Figure 3.6:** Effect of signal pre-emphasis [14]

### Equalization

Unlike pre-emphasis, equalization is done at the receiver side. Equalization is used for compensation of non-linear transfer function of the data channel. The corresponding filter should be complementary to transmission losses. There are three types of link adaptation.

The first one is passive, which uses predefined coefficients. They are determined from already characterised channel characteristic. The obvious disadvantage is that for design of such a filter, channel has to be characterised.

Second approach is active adaptation, where the coefficients of equalising filter can be reconfigured at runtime.

The last and most novel approach is the use of adaptive filtering. Filter coefficients are automatically adjusted for any changes of the channel characteristics. These adaptive algorithms find the minimum of error function representing optimal channel state. [16]

## 3.5 State of the art

Current state of the art MGTs use multilevel modulations, such as PAM4 (4 level Pulse Amplitude Modulation), where one symbol transfers 2 bits at a time. These modulations are used by higher rate links, mostly above 50 Gbps, because at lower rates it would waste signal integrity budget.

Advanced adaptive equalization is also making its way into both receiver and transmitter technology. It will enable longer transmission lines within rougher environments as well as better sensibility at higher line rates.

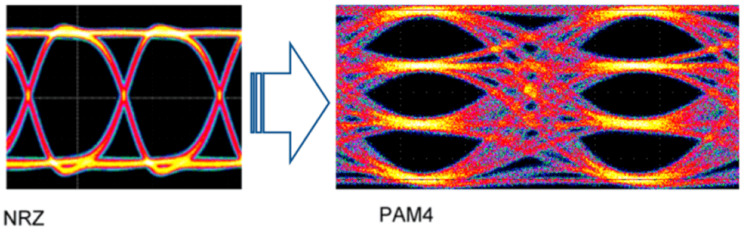



**Figure 3.7:** Eye diagram of NRZ vs PAM4 modulation [17]

## 3.6 Gigabit transceivers on Xilinx 7 series devices

### 3.6.1 Overview

Gigabit transceivers are hardened IP blocks usually on devices such as FPGAs. These blocks need to be "baked" into the silicon because the internal fabric of FPGAs can't run at such high frequencies. There are different types of high speed serial transceivers on Xilinx FPGA devices. Table 3.1 compares the different options Xilinx offers on their FPGAs.

**Table 3.1:** Transceivers offered on Xilinx FPGA devices, sourced from [18]

| Transceiver | Max bandwidth [Gbps] | Devices |
|---|---|---|
| GTP | 6.6 | Artix 7 |
| GTY | 32.75 | UltraScale, UltraScale+ |
| GTX | 12.5 | Virtex 7, Kintex 7, Zynq |
| GTH | 16.3 | UltraScale, UltraScale+ |
| GTM | up to 112.0 | Versal, UltraScale, UltraScale+ |
| GTZ | 28.05 | Virtex 7 |
| GTR | 6.0 | Zynq UltraScale+ |

### 3.6.2 GTP transceiver on Artix-7

Chosen FPGA is an Artix-7. This variant only supports GTP transceivers. These are organised into GTP Quads. Each GTP quad comprises of two parts: `GTP_COMMON` and `GTP_CHANNEL`. `GTP_COMMON` block is responsible for clock generation and distribution. It receives reference clock signal from reserved IO pins and routes them to phase locked loops, which there are two of. These PLLs set the line rate of the GTP instance. This implies that one quad can communicate in full duplex mode on four channels at two different frequencies.

Figure 3.8 shows inner topology of the `GTP_CHANNEL` primitive. Each channel has 2 sections, PMA and PCS. The high speed serial signal flows from traces

on PCB to transceiver RX pads. There it first passes through PMA, than into PCS and finally into FPGA logic interface.

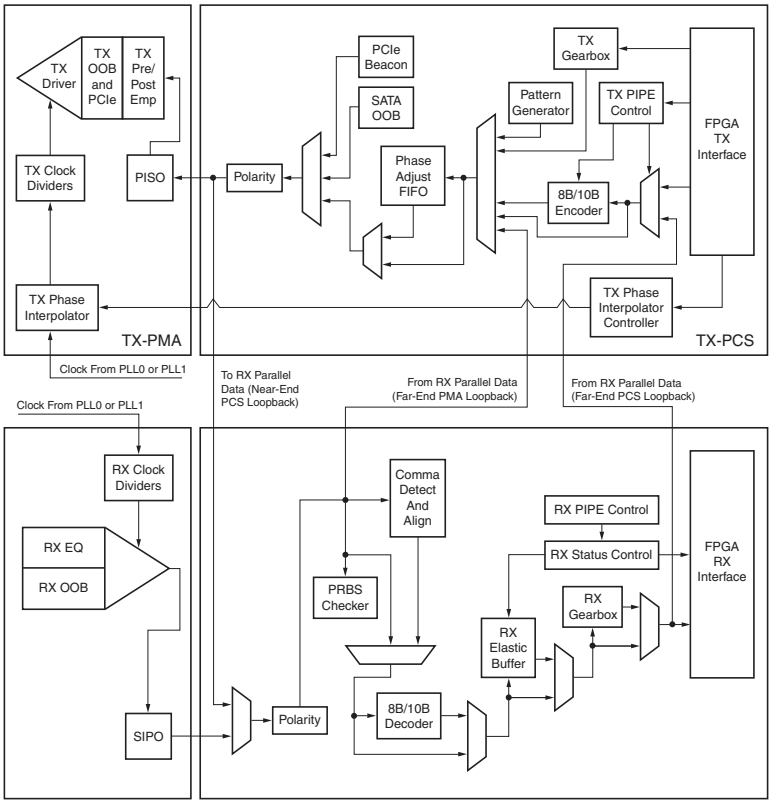



**Figure 3.8:** Topology of GTP transceiver [16]

The GTP enables designer to use loopback modes. There are total of four loopbacks, two in PMA and two in PCS section. Each one is designed to help debug the interface in different stages.

### 3.6.3 Physical Medium Attachment

Physical Medium Attachment (PMA) is a section in both transmitter and receiver implemented in analog circuitry. It comprises of IO buffers, serializers and deserializers, clock data recovery and link equalization modules. It interfaces between physical link to the digital part of receiver/transmitter.

#### Buffers

Input/output differential buffers are blocks that are connected directly to external pads of the package. The transmit channel output driver schematic depicted on figure 3.9 employs pre-cursor and post-cursor preemphasis output

drivers. Parallel to serial shift register is also present. Preemphasis level is configurable with 5 bits of resolution. Post-cursor driver can have gain up to 12 dB whereas pre-cursor driver 6 dB. Both preemphasis coefficients can be optionally inverted.

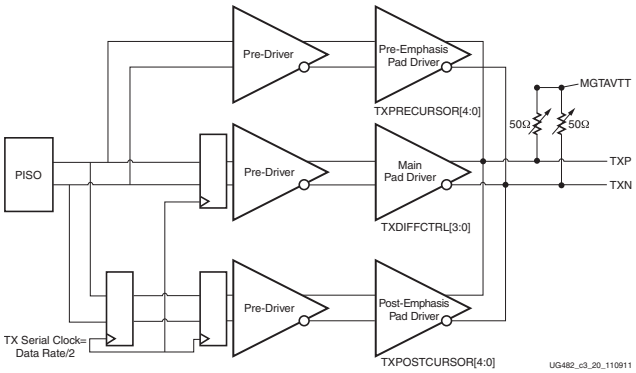



**Figure 3.9:** Block diagram of GTP output buffers with preemphasis drivers [16]

The signal flow in receiver is depicted figure 3.10. First the signal is terminated, than equalized. After that, clock signal is reconstructed from the data and it's used to clock deserialiser circuit.

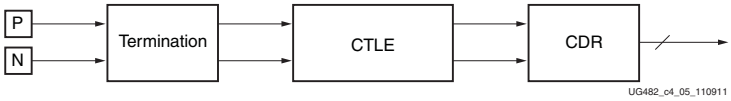



**Figure 3.10:** Receiver equalization [16]

## Termination

Termination circuitry, which schematic is on figure 3.11, implements configurable calibrated termination resistors and programmable common mode termination voltage. Also JTAG boundary check connections are available. The options of common mode termination are also shown on figure 3.11. Programmable common mode voltage can be configured in range of $100-1100$mV.

## Equalization

After termination the equalization is performed. The GTP transceivers use continuous time linear equalisers. Their function is to compensate distortion in the channel caused by high frequency losses. It can compensate up to 12 dB of losses at the Nyquist frequency [16]. The physical channel insertion loss is worse at high frequencies, that means the channel can be characterised as low pass filter. Equaliser usually performs high pass filtering to balance the signal spectral components. Careful consideration has to be taken when

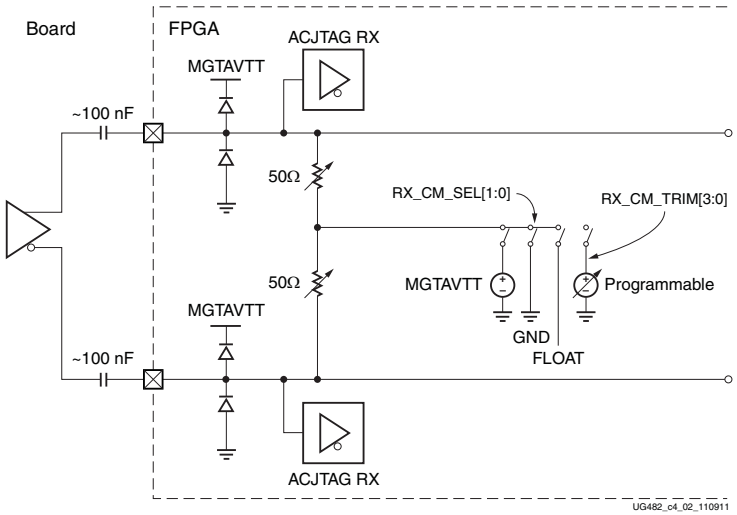



**Figure 3.11:** Receiver analog front end termination [16]

selecting CTLE coefficients to prevent over or under equalization. Xilinx 7 series transceivers offer auto-adaptation feature, that will select optimal CTLE coefficients autonomously. The automatic CTLE selection can run continuously or be triggered on request.

To properly sample incoming data stream, clock signal has to be first extracted. CDR in GTP transceivers employ phase rotator architecture. Two samplers are used - data and edge sampler. CDR state machine than determines the phase difference of incoming data to reference clock and controls the phase interpolators. Phase of the edge detector is aligned to the transition of signal, data sampler phase is aligned to the middle of the bit interval.

PMA is also responsible for generating out-of-band signalling which is used in various protocols. These symbols are defined out of encoding and physical specification and usually in separate frequency band, which means that the needed circuitry has to be integrated into analog domain of the receiver. They can be used for synchronization, handshaking and flow control.

### 3.6.4 Physical Coding Sublayer

Physical Coding Sublayer (PCS) is a section of gigabit transceivers that is mostly in digital domain. It is responsible for data alignment, comma detection, decoding, clock domain transitioning and receiver state management. Another useful feature of PCS is data polarity control. If design error in PCB occurs, the receiver is able to flip the assignment of RXP and RXN pads by inverting the parallel data.

## Data alignment

Data alignment and comma detection are used to find packet boundaries in the incoming serial data stream. To make alignment possible, transmitter sends unique sequence of recognizable bits called comma. The comma is usually K char as defined by 8b10b coding scheme. When the detector finds this pattern, it moves the data to a byte boundary as shown on figure 3.12. That way the parallel data words are matched to transmitted ones.

Each time the comma detector encounters the comma pattern, it aligns the data again. Automatic comma alignment can be switched off at runtime and manual triggering of the procedure is supported. There is also functionality to manually "slip" the byte boundary. Exact comma value can configured when instantiating the GTP. Comma detector can be optionally sensitive to both positive or negative value of the comma.

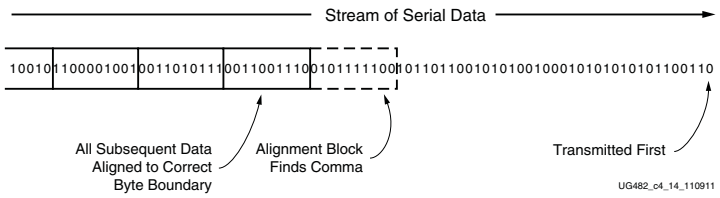



**Figure 3.12:** Concept of comma alignment [16]

## 8B/10B encoder and decoder

GTPs include hardware 8B/10B encoders and decoders. These can be used optionally if used protocol already uses another coding scheme. Table 3.2 shows a few of the valid characters. Control characters have prefix `K`, data characters prefix `D`.

**Table 3.2:** Table with a sample of the valid 8B/10B encodings

| Data Byte Name | Input bits | Negative character | Positive character |
|---|---|---|---|
| D0.0 | 000 00000 | 100111 0100 | 011000 1011 |
| D20.0 | 000 10100 | 001011 1011 | 001011 0100 |
| D24.2 | 010 11000 | 110011 0101 | 001100 0101 |
| D9.3 | 011 01001 | 100101 1100 | 100101 0011 |
| K28.5 | 101 11100 | 001111 1010 | 110000 0101 |

The encoder generates words based on input data word and control character indication flags. It also calculates running disparity and chooses suitable polarity data characters to ensure links DC balance state.

Decoder located in receiver outputs decoded words with corresponding indication of control characters. It can also detect if data is not a valid 8B/10B

encoded word. The decoder also calculates running disparity and can output flag if disparity error is detected.

#### Elastic buffer and clock correction

Elastic buffer is used to bridge two clock domains - clock derived from the recovered clock signal and user design clock derived from GTP common PLL. There will always be difference between those domains, even if they are running at the same frequency. This difference can cause the elastic buffer to overrun or underrun. Special characters are transmitted so that receiver can insert them twice or ignore them to compensate the timing differences. Following data processing logic has to consider those characters.

#### Channel bonding

Channel bonding is a technique used to combine multiple connections into one higher throughput channel. It is widely employed in PCI Express, DisplayPort and others. Since multiple lanes (each connection is refered to as lane) are used, time difference between them has to be compensated. Specialized channel bonding characters are used to initiate the communication and calibrate the skew. Calibration process on the receiver side determines delay of every line and adjusts the corresponding latency of elastic buffers. Data in FPGA fabric then appears without any time difference for each lane.

### 3.6.5 FPGA interface

The GTP exposes interface for common logic, RX and TX subsystems. Common logic is a wrapper for reset finite state machines, it ensures proper reset procedure as the multiple complex GTP components have defined reset behaviour. Receiver and transmitter PCS part is interfaced with the logic fabric inside the FPGA.

Transmitter provides `TXUSRCLK` which is derived from the reference clock and is in a relationship with the synthesized transmit clock. If 20bit internal datapath is used, this clock is 160 MHz for 3,2 Gbps linerate. Every signal that is connected to the TX must be synchronous with this clock. The basic signals used to transmit data are the data itself and control character flags.

Receiver exposes more signals. Data vector and K char identification remain the same, but error signals for events such as disparity error and character not in table, which signifies non-valid character that couldn't be decoded, are added. Alignment signals can be optionally also exposed.

# Chapter 4

# Implementation of readout system

This chapter is focused on implementing the gigabit readout system for ColorPix-0B. Gigabit transmitter implemented on the chip will be briefly described first and then the FPGA readout system implementation.

## 4.1 ColorPix-0B

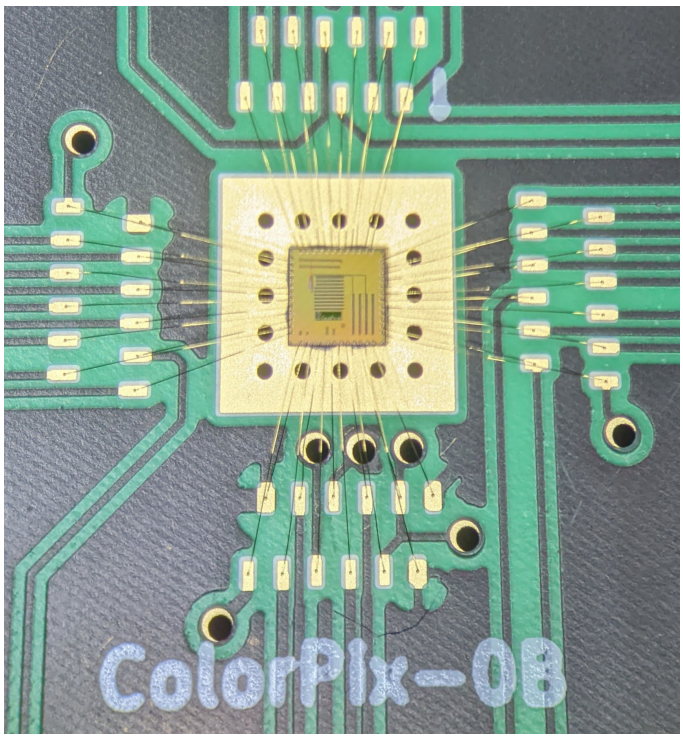



**Figure 4.1:** ColorPix-0B bonded to a PCB

Colorpix-0B is a prototype of electronic system for X-Ray particle detector.

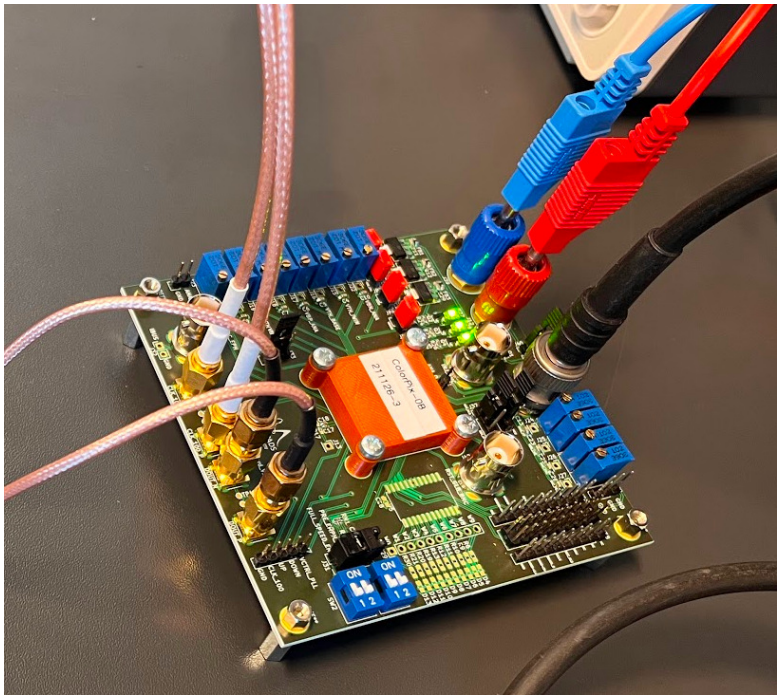


**Figure 4.2:** ColorPix-0B testing board

It was manufactured in CMOS 65 nm process. It's purpose is to test the pixels analog front end, windowing, pixel clustering and integrated gigabit transmitter with relevant components. The bonded chip is depicted on figure 4.1 and the testing board on figure 4.2.

### 4.1.1 Integrated gigabit transmitter

Block diagram on figure 4.3 represents simplified digital domain design interpretation of the integrated gigabit transmitter on ColorPix-0B. The whole design is synchronized by differential clock input. There is an input for enabling preemphasis of the CML output driver and option to reduce PLL frequency to half which is 800 MHz. Output driver bias current can be set by external potentiometer.

The design transmits test pattern stored in memory. Each clock cycle generates byte and K character signal. These signals are fed into 8B10B encoder, which will generate encoded 10 bit word. It is then transferred to a series of serializers. The first level reduces the data flow to 2 bits wide at 1600 MHz. Second level consists of a series of differential registers clocked with a phase difference of 180 degrees. The output CML driver has a multiplexer on its input that selects which bit is currently transmitted. The multiplexer switches with the link frequency of 1600 MHz.

To use the integrated transmitter the chip needs to be powered and all

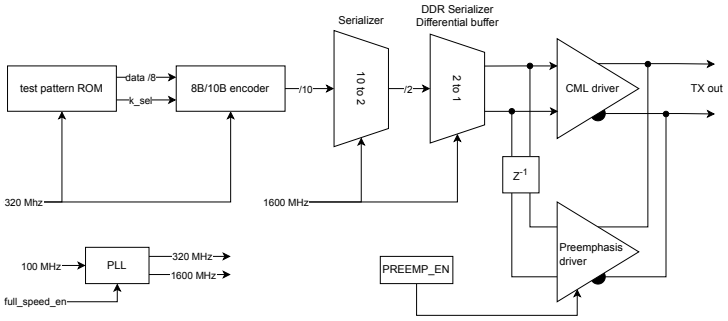



**Figure 4.3:** Block design representation of ColorPix-0B gigabit transmitter

necessary signals have to be connected. Evaluation board for ColorPix-0B which is shown on figure 4.2 was used. It has all necessary power circuitry and takes only 5 V from external power supply. Differential clock source with 100 MHz has to be connected to a pair of SMA connectors as well as data, which is also AC coupled with capacitors on the board. Output CML driver current bias can be set with a potentiometer, preemphasis and link speed are controlled with a switch.

## 4.2 AC701 Evaluation board

The readout system has been implemented on Xilinx AC701 [19] evaluation board. The board features an Artix-7 FPGA with 215360 equivalent logic cells, 13 Mb of block RAM, 16 GTP transceivers and total of 500 IO pins. It also provides robust power supply options. Main the features are:

- Gigabit ethernet on RJ-45 connector
- SFP module cage
- HDMI output
- PCI Express® x4 Gen2
- 1 GB DDR3 SODIMM
- Alphanumeric LCD display
- User buttons, LEDs, rotary encoder, SD card slot, etc.
- Integrated JTAG adapter

The board also provides the developer with complex clocking options:

- Fixed 200 MHz oscillator with differential output
- Programmable differential oscillator with range of 10 MHz to 810 MHz
- Differential SMA clock input
- Differential SMA clock input for GTP reference clock
- Jitter attenuation/clock recovery clocking IC (with optional standalone mode)

All of the programmable generators as well as power management can be configured via I2C bus or PM bus respectively. I2C devices are also isolated by I2C multiplexer.

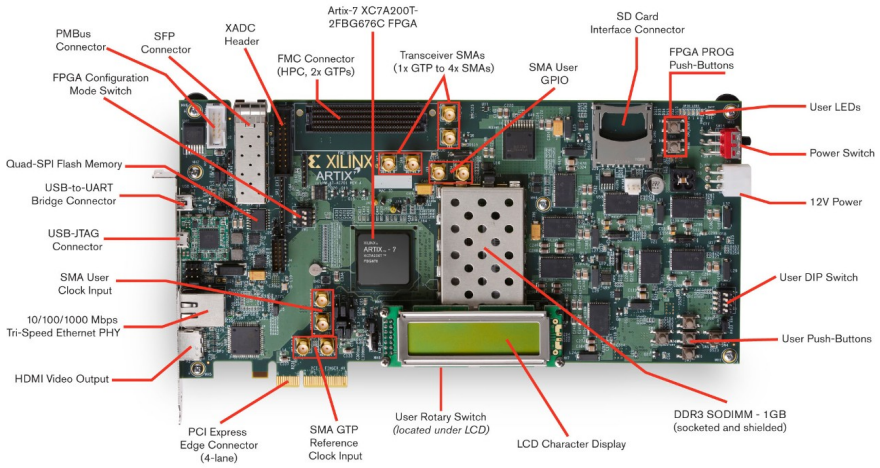



**Figure 4.4:** AC701 board [20]

## 4.3 FPGA readout system

The readout system consists of FPGA and ColorPix boards connected together with coaxial cables. The main component of the system is AC701 evaluation board made by Xilinx. Purpose of this system is to capture data stream generated on Colorpix-0B, perform any needed buffering and data composition and transmit this stream into the PC. Figure 4.5 shows the simplified architecture.

Fixed test pattern of 32 bytes is sent from the Colorpix-0B. This stream of data should be captured and deserialized inside the FPGA. This data sequence was designed to test the performance of implemented gigabit transmitter.

For design validation without Colorpix-0B electronics, local loopback of the GTP was used. GTP transmitter can be connected to the GTP receiver with

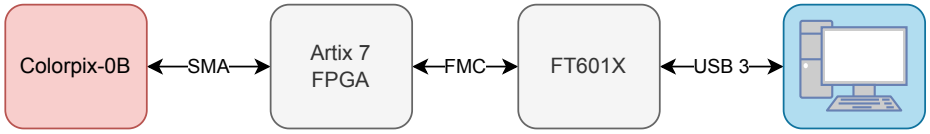



**Figure 4.5:** Simplified system architecture

external coaxial wires. Both channels work on the same line rate since they are clocked from the same PLL.

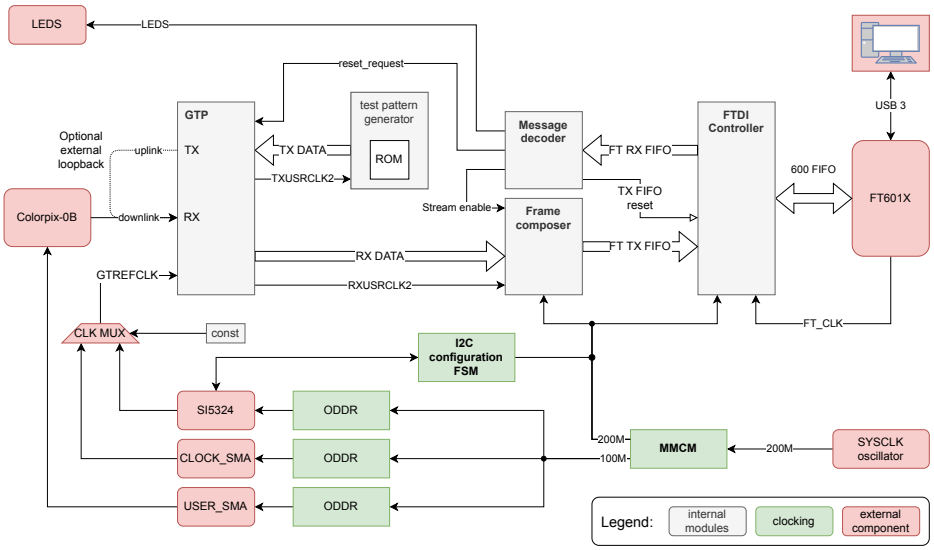



**Figure 4.6:** FPGA top entity architecture

### 4.3.1 Readout system clocking

AC701 board provides numerous options to generate and distribute clock signals. The available GTP Quad 213 has two reference clock inputs, each connected through multiplexers to a fixed clock generator, SMA connector and programmable oscillator/jitter attenuator. Because of strict performance requirements, the GTP reference clock cannot be provided inside the FPGA. Figure 4.7 shows a schematic of external clock distribution. Because the ColorPix electronics were designed for 3,2 Gbps line rate, GTP PLL restricts us to have to use a 100 Mhz clock reference.

One solution is to program the external jitter attenuator Si5324. It can run in both free and jitter attenuation mode. In free mode it works as a standard clock synthesizer. In jitter attenuation mode it generates phase coherent clock signal with relation to provided reference signal. Is is intended to be

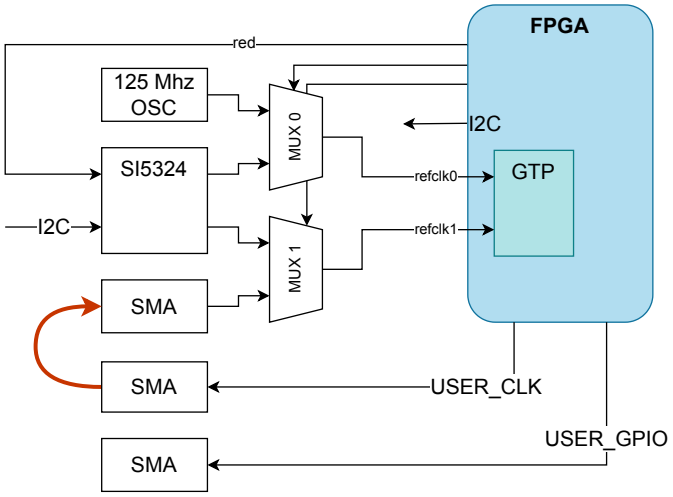



**Figure 4.7:** Design clocking diagram

used in clock recovery applications, for example to recover reference clock from SFP connectors. For this reason a simple I2C configuration module was implemented, however because of unresolved bugs and synthesis irregularities this solution was abandoned as the Si5324 couldn't be properly configured.

Second solution is to use available SMA connectors and loop back the clock signal generated inside the FPGA externally. At the end this was the preferred variant. The design has to also generate a reference 100 MHz clock for the ColorPix transmitter. This clock should be too available on external SMA connectors.

The reference clocks are synthesized inside the FPGA from the SYSCLK clock source. They are then transitioned to the IO level with ODDR output buffers and connected to OBUFDS drivers, which output LVDS25 signal.

### 4.3.2 GTP transceiver

GTP transceivers on Xilinx devices can be only instantiated through GT Wizard IP generator. It is one of many standard Xilinx tools for IP core automated generation. The block can't be inferred from RTL source, but with reverse engineering and proper constrains one could instantiate the hard IP block manually, however for more complex protocols such as SATA or PCIe the setup would be impossible.

The GTP requires additional logic that is responsible for TX and RX channel reset, proper initialization of PLL and reconfiguration. These modules are optionally generated with the IP core, however GTP block has also a dynamic reconfiguration port, that can be useful if the design needs to change

transmission parameters when already active.

The generated IP core brings out IO ports which need to be constrained, even though they are hard-assigned to the respective pins of the GTP channels. The SMA coaxial connectors on AC701 board are directly connected to both the RX and TX channels of GTP channel X0Y3 without any coupling capacitors.

### Transceiver configuration

The transceiver wizard IP core configuration is included in appendix C. First the MGT type has to be selected, which is in Artix-7 FPGA restricted to GTP transceiver. Next, particular GTP Quad and channel is selected. Line rate of the transmission and eventually used protocol has to be specified, together with clock reference and common PLL.

The IP wizard allows entry of any line rate and it automatically provides possible clock reference frequencies which can be selected after. Further wizard pages are dedicated to additional inputs and outputs, comma specification, data path width and encoder selection.

Table 4.1 shows summary of the transceiver configuration.

**Table 4.1:** GTP configuration summary

| Features | GTP |
|---|---|
| Protocol File | Start_from_scratch |
| TX Line Rate(Gbps) | 3.2 |
| TX reference clock(MHz) | 100.000 |
| Encoding | 8B/10B |
| TX Internal Data width | 20 |
| TX External Data width | 16 |
| TXUSRCLK(MHz) | 160.0 |
| TXUSRCLK2(MHz) | 160.0 |
| TX Buffer Enabled | true |
| RX Line Rate(Gbps) | 3.2 |
| RX reference clock(MHz) | 100.000 |
| Decoding | 8B/10B |
| RX Internal Data width | 20 |
| RX External Data Width | 16 |
| RXUSRCLK(MHz) | 160.0 |
| RXUSRCLK2(MHz) | 160.0 |
| RX Buffer Enabled | true |

### 4.3.3 Data pattern generator

This is a simple module that generates testing data pattern for the GTP. This is the same pattern that is transmitted from the ColorPix-0B. It is implemented as an array of 32 elements. Each element is 9 bit wide word that stores data byte and control character identification. The array is initiated in verilog from a text file.

Since the GTP FPGA data interface is 2 bytes wide, 2 elements have to be read from the memory at one clock cycle. The data is then concatenated into a single 16 bit wide vector and 2 bit control vector. The module is clocked from `GT_TXUSRCLK` which ensures that everything is synchronized with internal GTP circuitry.

By connecting the RX and TX SMA connectors with coaxial cables external loopback is created. This was helpful for development when ColorPix electronics were not available.

### 4.3.4 Data frame construction

The main module responsible for data propagation in FPGA is the frame composer module. It transitions data from the clock domain of GTP RX (160MHz) into 200 MHz domain where frame generation is done. It generates packets in defined format and pushes them into the FTDI transmit FIFO. Figure 4.8 shows diagram of implemented frame composer module.

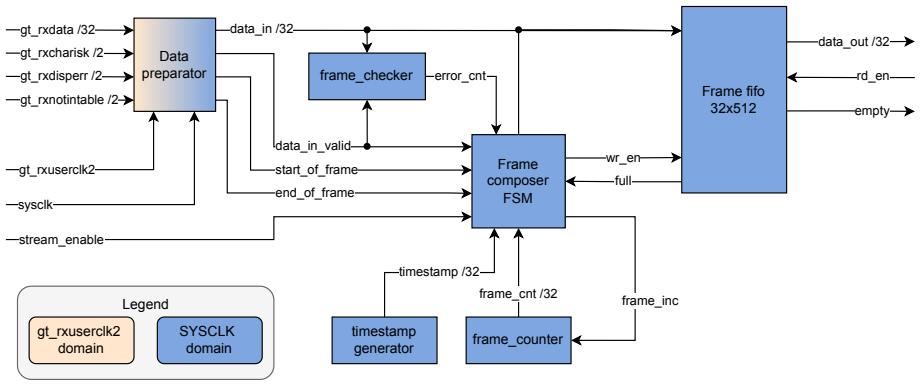



**Figure 4.8:** Block diagram of the frame composer module

To ensure that only whole packets are generated, another FIFO buffer was created. As the USB 3 PIPE interface operates in burst transfers, the transmit FIFO is not continuously read. The frame FIFO provides a flow separation point when the USB PIPE is full. In that case the frame generation is suspended until the frame fifo is empty again.

### Data preparator

This module is connected to the GTP and it aligns the data stream to control character boundaries. The generated GTP wrapper provides 4 output signals - `rxdata[15:0]`, `rxcharisk[1:0]`, `rxdisperr[1:0]` and `rxnotintable[1:0]` as well as `rxuserclk2`, which is the clock for those signals.

Data frames received by the GTP are bounded with 2 control characters - K28.5 and K28.0. Data preparator takes these bounds and filters out only the inside of the frame. For used test pattern this leaves 28 bytes that were inside the bounds. Preparator module has to align the data, since the first frame byte can occur in low and in high byte of `rxdata`. It creates 4 byte long words, because this word length is used further in the design.

Output interface of this module is connected to a FIFO which has read always enabled. This way it can't overflow, because the read clock is faster than write clock. FIFO has to be used because it transitions between `rxuserclk2` and `sysclock` clock domains.

### Frame composer

Frame composer receives stream of data from the GTP and generates a frame packet. The frame protocol is visualized on figure 4.9.

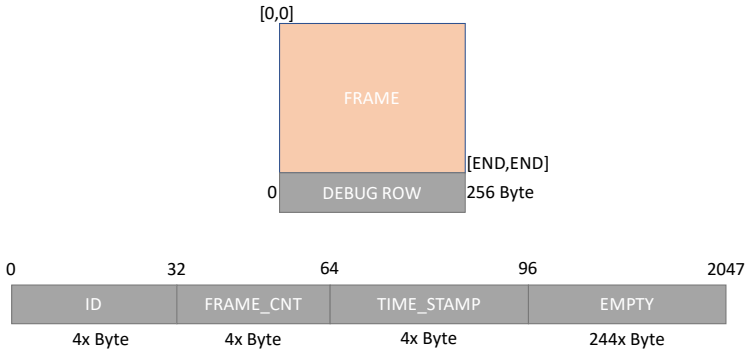



**Figure 4.9:** Frame packet protocol

The frame composer is implemented as a state machine, which flow is depicted on figure 4.10. Frame packet generation starts only if outbound frame fifo is empty. This ensures that all 284 bytes that the frame protocol defines are generated.

The composer is designed such as frames that are received when header is being generated are discarded. Once the frame FIFO is empty and start of frame flag active the frame generation process is started.

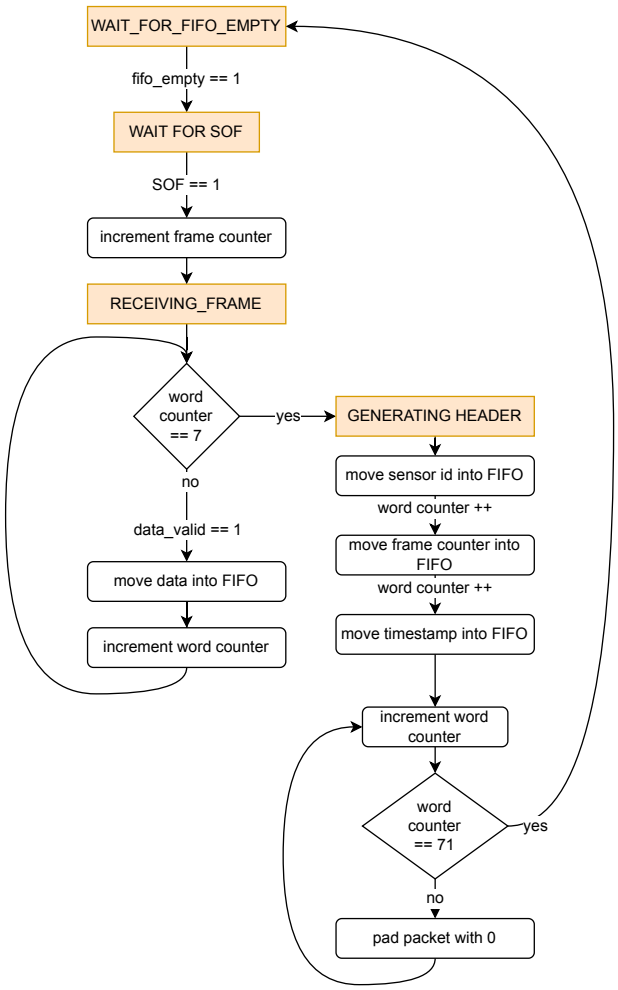



**Figure 4.10:** State flow diagram of frame composer state machine

Once the frame header generation process is started, all incoming frame data are discarded. This is due to increased overhead of the protocol. In future the relative size of frame and header will be much bigger.

### Frame checker

Frame checker is a module that compares incoming data stream with stored reference. The stream is synchronized to start of frame flag. If the incoming frame is not equal to the reference, for each frame this difference occurs frame error counter is incremented and for duration of the frame a `frame_error` flag is driven high. Errors are counted for the whole frame, even if the difference is only in one bit.

### Clock domain crossing

Whenever data is transferred between multiple clock domains, it needs to be synchronized. Since the readout system works in 3 clock domains, these techniques had to be applied. There are 2 main cases of transition - from higher frequency clock domain to lower frequency clock domain and vice versa.

Each clocked register has defined $t_{setup}$ and $t_{hold}$ times relative to clock edge that are required for input signal. If these timing requirements are not met, meta-stability on the output of the register occurs. That means that we can't say with certainty if the logic output of the register is 1 or 0.

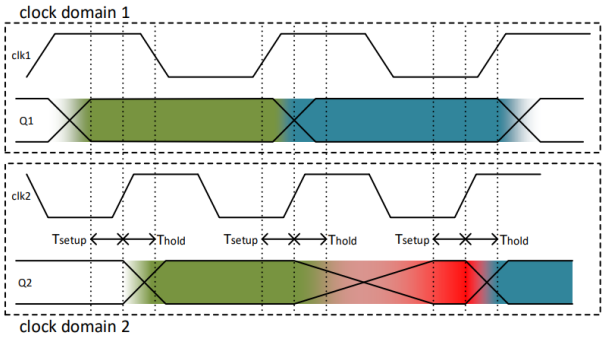



**Figure 4.11:** Signal timing [21]

### Brute force synchronizer

This is the simplest way to transfer signals between two clock domains. If the first flip flop violates the timing requirements and registers in metastable signal, there is enough time for the second flip flop to capture stabilised level. Figure 4.12 shows how is the basic circuit implemented.

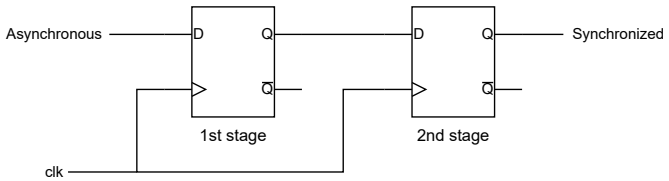



**Figure 4.12:** Brute force synchronizer using 2 stages of registers

### Asynchronous FIFO

Asynchronous FIFOs are also used in clock domain crossing. They are synchronization entities which are suitable for large and fast data transfers.

They usually wrap a standard dual port RAM. Dual port RAM has a read and write port and each can be clocked from different domain. The structure is shown on figure 4.13. The FIFO wrapper usually uses gray codes to do provide basic error detection. Gray codes are codes where transition to next state changes only one bit. This code is used as an address counter for both memory ports. Handshakes with standard synchronizers are then connected between the two domains.

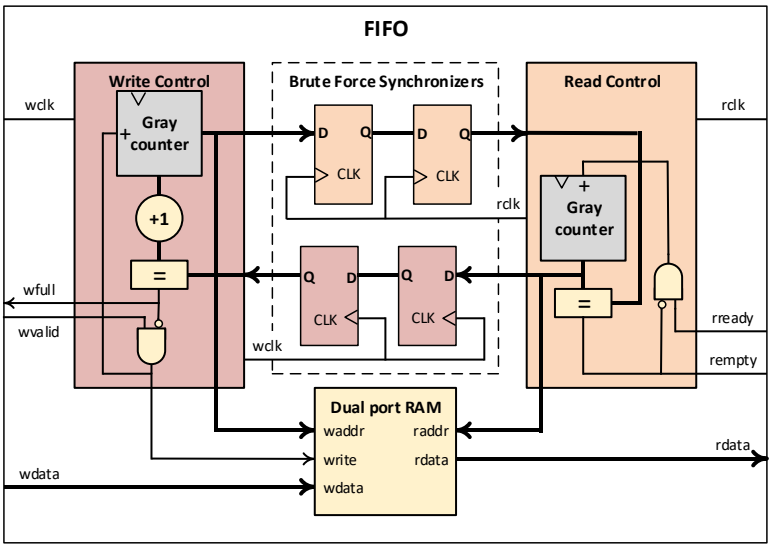



**Figure 4.13:** Asynchronous FIFO structure [21]

### 4.3.5 FTDI USB3 interface

In this work the FT601 interface chip is used. Its block diagram is on picture 4.14. It simplifies all USB communication because it uses simple FIFO master/slave bus. The manufacturer provides proprietary `ftd3xx` driver with three example projects and limited documentation.

The AC701 board has an FMC mezzanine connector [19] that is connected to the FT601X-B expansion card with FT601Q interface chip. This chip is a bridge between FIFO slave interface and USB PIPE endpoint.

The FTDI interface complies with USB 3.1 Gen 1 specification, thus enabling maximum speeds of 5 Gbps [13]. However this theoretical maximum speed can't be reached with this interface chip. The maximum practical transfer speed can be determined from the bus parameters:

$$\begin{aligned} BW &= bus\ width * bus\ frequency \\ BW &= 32\,\mathrm{b} * 100\,\mathrm{MHz} \\ BW &= 3200\,\mathrm{Mbps} = 400\,\mathrm{MBps}. \end{aligned} \tag{4.1}$$

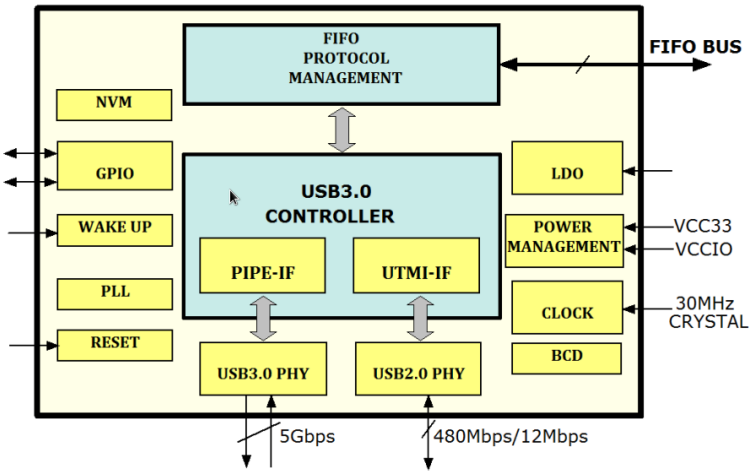



**Figure 4.14:** Block diagram of FT600/601 interface chip [22]

FTDI interface is configured to communicate in "600" mode, which is a proprietary bus protocol specification for data transfers between 2 endpoints. In this case the Artix-7 FPGA is initiating all transfers and thus is in master role. The bus read transaction is shown on figure 4.15. The protocol defines handshake commands transferred via data lines. Master initiates the transfer, waits for response from slave and then starts reading or writing to the bus.

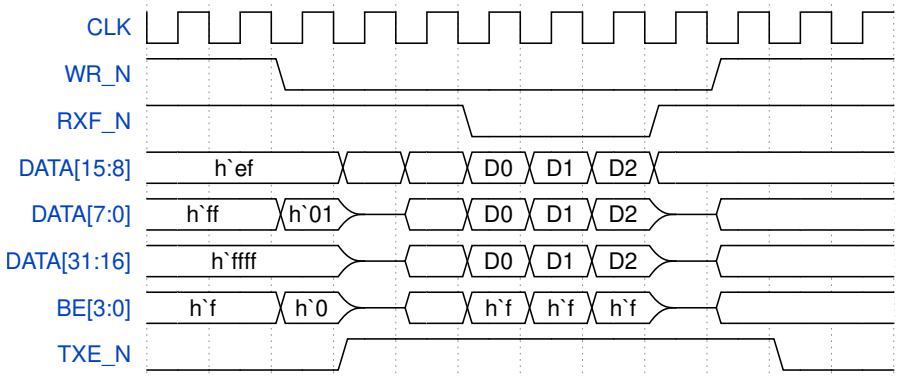



**Figure 4.15:** FTDI 600 bus read transaction timing diagram

The communication module that handles bus communication was provided as functional IP. Since it was previously implemented on Spartan-6 FPGA and used instantiated block memory and IO buffers, these had to be remapped to instances available in Artix-7 FPGA. New IO constraints had to be created to comply with the AC701 FMC connector. The FT601X expansion board uses total of 40 IO pins in single ended mode. The buffers were configured to use LVCMOS25 standard.

The design has a problem which means that the bus couldn't communicate at full speed. Depending on the particular implementation some bits would be erroneous. When the interface speed was decreased the errors disappeared.

This behaviour implies that there is a problem with timing of the individual bits on the bus. As the behaviour is changing with each new implementation it only confirms this theory. However correct timing constraints and grouping could not be determined. Sadly the FTDI documentation is not too eloquent in this manner.

### 4.3.6 Command interface

A simple protocol was designed for control of the internal logic inside the FPGA. As depicted in table 4.2, there are 3 command types: STREAM, LED and RESET. STREAM command can start and stop frame composer finite state machine. LED command is mapped to external LEDs on the board and is used mainly as indication of working communication between the host PC and FPGA board. RESET command issues a reset for the GTP system.

**Table 4.2:** Command protocol. All values are in hexadecimal notation and letter x signifies command argument.

| Command | Byte 0 | Byte 1 | Byte 2 | Byte 3 |
|---|---|---|---|---|
| Stream start | 01 | 01 | 00 | 00 |
| Stream stop | 01 | 02 | 00 | 00 |
| Leds set | 20 | 0x | 00 | 00 |
| Reset request | 02 | 00 | 00 | 00 |

### 4.3.7 Message parser

This verilog module is responsible for reading incoming 32 bit messages and parsing them. It is implemented as a very simple finite state machine. Whenever the receive FTDI FIFO is not empty, one word is read from it. Depending on the command type which is determined by zeroth byte in the message next state transion is made.

### 4.3.8 DDR memory buffering

Originally it was planned to buffer incoming frame data to DDR memory because of further needed descrambling and reorganisation. However later it was removed from requirements because the exact format of incoming data was not known yet. Despite that the DDR3 SDRAM on AC701 board was tested and configured. Xilinx provides IP core tool Memory Interface Generator which abstracts all complex pin and timing constraining from the developer. It also assures proper calibration of the DDR interface. The memory is than usable as an AXI4 slave. AXI4 is a point to point memory mapped interface compliant with AMBA standard. AXI4 FIFO IP would be

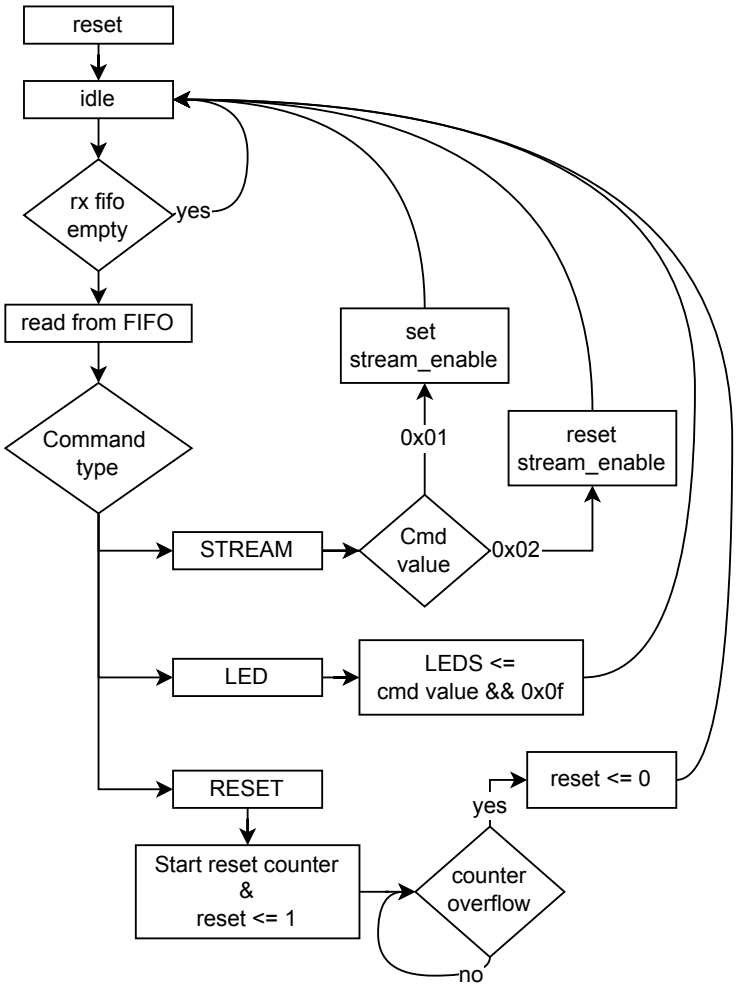



**Figure 4.16:** Message parser state flow

than used to take advantage of the DDR memory and use it as a FIFO buffer. This high level buffering would enable the design to capture large bursts of data coming from the detector and process it later, as the data stream would have a clear path.

## 4.4 Host readout GUI application

Control GUI application was developed as a part of this work. It is used to debug the communication stack between host computer and FTDI chip, send messages to the FPGA and receive frame data. This data can be saved to a file.

The application was written in C++ language using the Qt framework version 5.15.2. Qt was selected for the sake of simplicity and performance. The user interface is static and completely written in C++ source code.

As a part of the implementation, abstraction class for handling the communication with the FPGA was developed. It offers layer above the FTDI driver libraries and uses the defined protocol to control state of the readout. FTDI `ftd3xx` libraries were first evaluated using the provided examples.

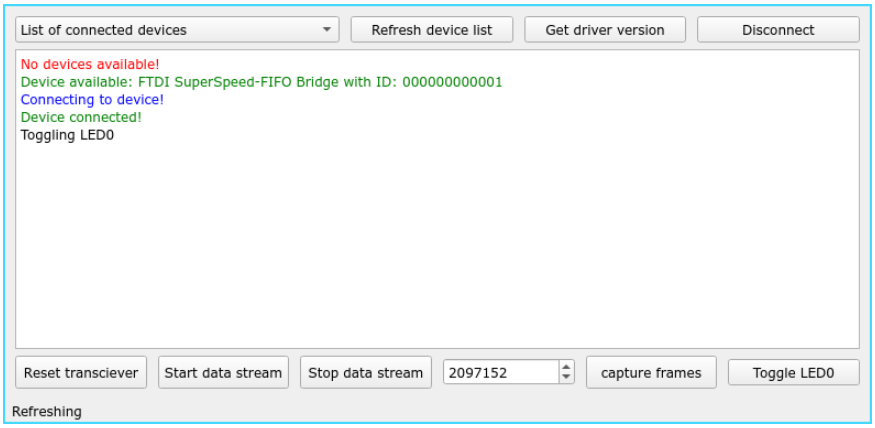



**Figure 4.17:** GUI readout application written in C++ and Qt framework

Any connected devices which the FTDI driver recognizes are detected after start. Connection can than be estabilished. This will create an handle for the FTDI library. There are two modes of data transfer operation. One is to request fixed amount of frames to be read. These frames are saved into a binary file. The other mode is to simply enable and disable the stream. Running BER analysis will be implemented for the continuous mode since it will be useful for further characterisation of the data link.

### 4.4.1 Data acquisition

The readout state flow diagram is shown on figure 4.18. This application was used for debugging the FTDI API and the transfers. Later it was used as an template for the main GUI application.

The application has no user configurable inputs, all properties are set in source code. At first it checks if `FT_Connect` executes and returns a valid `FT_HANDLE`, which is a void pointer to unknown structure on which the `ftd3xx` driver operates. Then it proceeds to configure the FTDI driver and interface. Next LED command is sent for debugging purposes. Stream stop command is sent so the transmit FIFO is flushed and reset. The last initialization command is stream start which presumably starts the frame generator.

The data is received in chunks which are 32 frames long. Each frame is

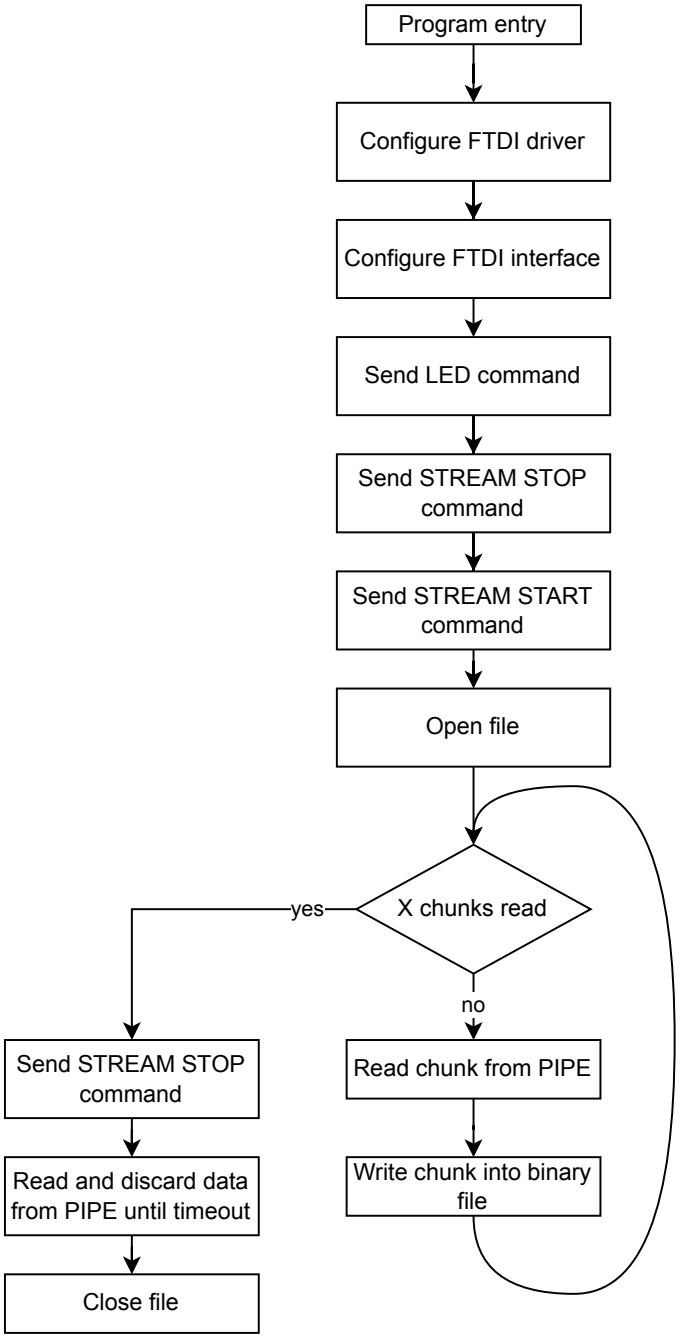



**Figure 4.18:** State flow diagram of data acquisition application

284 B, thus the chunk length equals to 9088 B. After a chunk is received, it is written to a binary file. This process repeats until X number of chunks is received. After all chunks are successfully received, the stream stop command is issued and read from the pipe performed to clear out all remaining data that frame composer generated.

## 4.5 Design reports

### 4.5.1 FPGA utilization

The readout design utilisation report was generated. Table 4.3

**Table 4.3:** Device utilization report

| Element | Used | Available | Percentage [%] |
|---|---|---|---|
| LUT | 1963 | 133800 | 1,47 |
| LUT RAM | 194 | 46200 | 0,42 |
| Flip Flops | 3350 | 269200 | 1,24 |
| Block memories | 22 | 365 | 6,03 |
| IO | 59 | 400 | 14,75 |
| Transceivers | 1 | 8 | 12,50 |
| Global clock buffers | 7 | 32 | 21,88 |
| Clock management | 1 | 10 | 10,0 |

### 4.5.2 Implementation timing report

Implementation timing analysis was extracted from the design. It shows that all requirements were met and no endpoints are failing the timing check. Timing has to be properly constrained, which means defining clock periods, signal clock domains and possible asynchronous domain relations. Figure 4.19 shows that the worst positive slack is 0,369 ns. This value expresses the worst delay created by combinatorial logic.

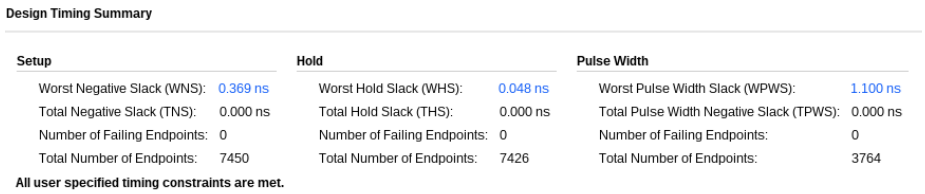



**Figure 4.19:** Timing report exported after implementation in Vivado toolchain

### 4.5.3 Power report

Device power consumption is most crucial for final implementation when power delivery circuitry is being designed. This design doesn't have much power draw, mostly due to the low utilization. Figure 4.20 visualises the reported power drains in the design. Total power consumtion is 618 mW.

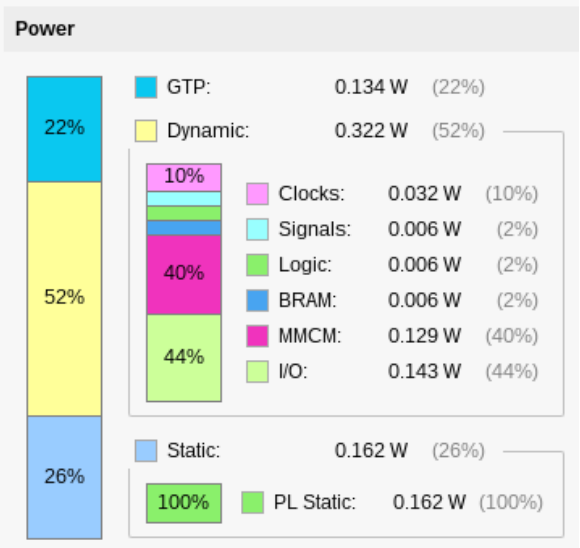



**Figure 4.20:** Reported power draw graph

# Chapter 5

# Transmission characterization

In this chapter the communication channel between Colorpix-0B and AC701 board was characterized. Eye diagram was captured for both enabled and disable preemphasis, jitter was measured and bit error rate calculated from captured data and in high speed oscilloscope.

The measurement workspace is depicted on figure 5.1. Colorpix-0B is connected to OWON XDG3000 signal generator which is generating the clock reference. The transmitter is connected to SMA adapter board where the link is terminated. High speed differential probe connected to the Teledyne WavePro HD oscilloscope is attached to the termination resistor.

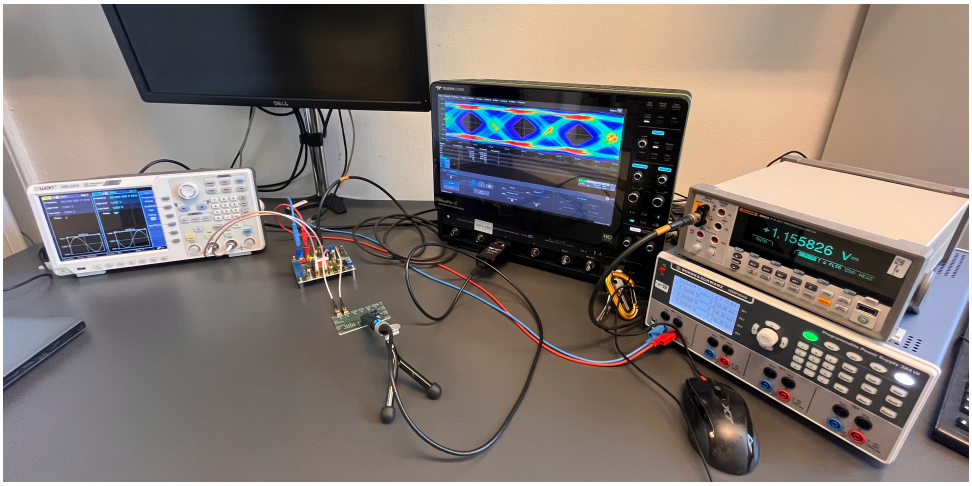



**Figure 5.1:** Measurement workspace

## 5.1 Eye pattern measurement

Eye pattern or diagram is a visualisation of repetitively sampled signal. The samples are overlaid on each other and density color map is applied. It represents a probability density function of the signal over time constraints. Eye diagram is frequently used in telecommunications and signal analysis as a tool to measure and analyse communication system performance. Its name was derived from its resemblance to an eye. When the signal quality is overall poor, the inner area (eye) should be smaller [23].

From the eye diagram one can observe additive noise in the system, bad synchronization to clock, inter-symbol interference, overshoot and undershoot, RMS value of logical levels, transition times and other multitude of characteristics. When the eye is open it usually signs minimal distortion of the analysed signal. The less the area of the eye opening is, the more interference and noise is present in the signal.

Figures 5.2 and 5.3 depict measured eye diagrams of transmit channel of Colorpix-0B. The transmission line was differentially terminated with 100 Ω resistor, on which the high speed oscilloscope probe was attached.

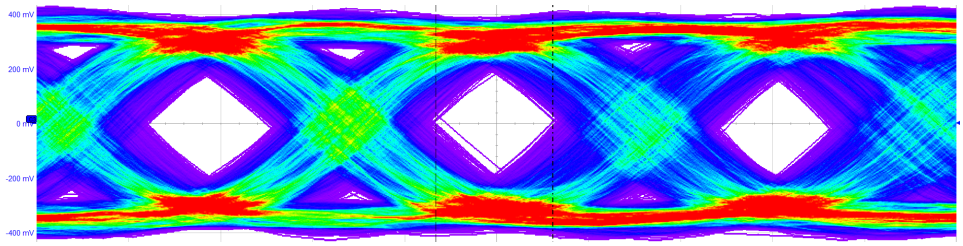



**Figure 5.2:** Eye diagram without preemphasis

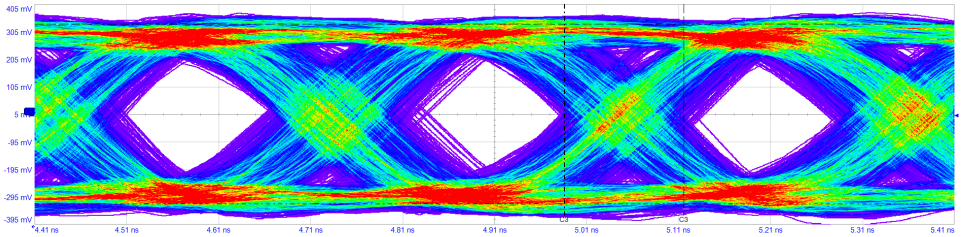



**Figure 5.3:** Eye diagram with preemphasis

The positive effect of preemphasis can be noticed, however as the measurement was done at far-end, thus the significance is not very high. Measured eye diagram measurements are depicted in table 5.1. On both diagrams open eye can be seen, which signifies overall good signal integrity [24]. However the jitter, which is indicated by the width of transition crossing, is on both measurements high. Probable causes of that will be proposed in the following sections.

**Table 5.1:** Measured characteristics of eye diagrams

| Measurement | Preemphasis on | Preemphasis off |
|---|---|---|
| Eye width [ps] | 127 | 136 |
| Eye height [mV] | 366 | 373 |
| Amplitude [mV] | 704 | 577 |
| Rise time [ps] | 183 | 164 |
| Fall time [ps] | 174 | 163 |
| Jitter [ps] | 88 | 130 |
| Bit period [ps] | 313 | 317 |
| One level [mV] | 331 | 290 |
| Zero level [mV] | -321 | -287 |

## 5.2 Jitter measurement

Jitter is an undesired measure of deviation with respect to periodicity of a signal. In electronics signals are usually propagated with respect to a clock signal, which has a presumably periodic characteristic. However in real world imperfections and tolerances occur [25].

There are 2 types of jitter: deterministic and random.

Random jitter is unpredictable and usually follows Gaussian normal distribution. In electronic circuits it is mainly caused by thermal noise [26].

Deterministic jitter is predictable and has a known probability distribution. It can be caused by reflections on the channel, cross talk and other interference [26].

Deterministic jitter has been measured from the eye diagram and can be seen in table 5.1. Random jitter couldn't be reliably measured because of the bandwidth limitations of used oscilloscope.

Relatively high jitter was measured in previous section. That can be caused by multitude of reasons, the main being unreliable reference clock source. As was described previously, the internal PLL of Artix-7 FPGA was used as a source and clock reference routing was not used as intended by the board designer. Since high speed communication most crucial requirement is as good clock source as can be, there is definitively room for improvement. Also clock references should be more rigorously validated.

Used oscilloscope couldn't measure jitter natively. Period analysis was performed instead, however since the link rate is high there was not enough data points to precisely measure the value.

Measured values were: mean of signal period

$$\mu_{T_s} = 612{,}8\,\text{ps},$$

standard deviation of signal period

$$\sigma_{T_s} = 23{,}97\,\text{ps}.$$

## 5.3 Bit Error Rate

Bit error rate (BER) is a unitless measure for quality of transmission channel. It can be calculated as

$$BER = \frac{Bit\ errors}{Total\ received\ bits}, \tag{5.1}$$

where *bit errors* is the number of single bit flips that occurred over a given time period of transmission and *total received bits* is the number of bits received over the same period.

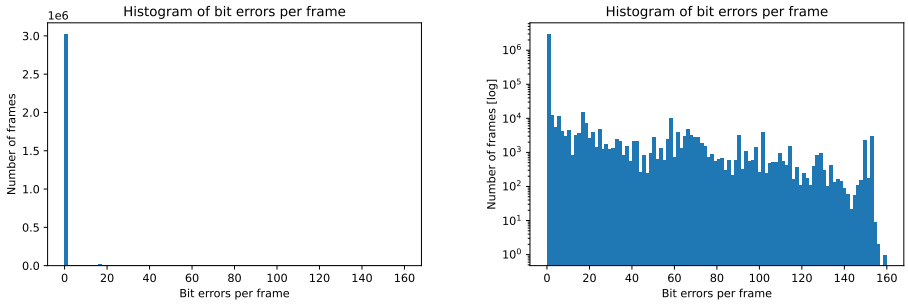
Colorpix-0B board and AC701 were connected by coaxial cables with SMA connectors. One pair was used for data and second pair was used for clock reference coming from the FPGA to Colorpix-0B. One more differential connection was a loopback providing clock reference to the GTP.

To determine BER for transmission channel between Colorpix-0B and AC701 board the C data acquisition application was used. Each received frame was written to a binary file from which it was later read by a python script in Jupyter notebook. Each frame was compared to a reference, which is the known fixed data sequence of 28 bytes, that is without synchronization K chars and interframe data. Bitwise XOR operation was performed between the received frame and reference. XOR operation leaves bits that are not equal in logic one. If the number of active bits is counted it represents the number of errors in current frame.

From equation 5.1 we can calculate

$$BER = \frac{8062747}{716800000} \approx 0.011248 = 1.125\%. \tag{5.2}$$

As can be seen from the histogram (figure 5.4), most of the frames don't have any errors. Median of sample is 0. However the mean BER for the whole sample is 1,12 %. Cause of such high value (in contrast, regular 1000 Base-T Ethernet BER requirement is 1 error in $1 \times 10^{10}$ bits [27]) can be attributed to high jitter of reference clock or mismatched termination at the receiver.




**(a) :** Y axis in linear scale **(b) :** Y axis in logarithmic scale

**Figure 5.4:** Histogram of number of error bits per frame

Since the calculated error rate of the whole chain was suspiciously high, it was decided to cross check this information with high speed oscilloscope and integrated 8b10b decoder. The same high speed differential probe was used to capture data. This time the trigger was set to specific pulse width. The decoder works on waveforms in captured time window. This way around 8 GB of decoder results were captured in CSV files for duration of around 2 hours. The capture speed was limited by the oscilloscopes ability to save decoded data, thus it could not capture at full bandwidth of the channel.

Another python script in Jupyter notebook was used to parse and analyse the acquired data. For some reason each CSV file ended with unaligned frame, so the script had to filter every last frame. In total 393623040 bits were tested with reference. The comparison was done in a same way as previously described and resulted in 0 detected errors. The worst case would be if the next captured bit was erroneous. That would imply bit error rate of

$$BER = \frac{1}{393623041} \approx 2.54 \cdot 10^{-9},$$

which is still higher than required BER of 1000 Base-T Ethernet, but it's better overall result. To consider this value more meaningful, we would have to acquire data for much longer period of time.

# Chapter 6

# Conclusion

This thesis has been devoted to implementation of high speed digital readout for X-Ray particle detector using FPGA. In the beginning, principles and concepts behind particle detectors were presented. X-Ray photon interaction with matter and possible utilization in medical applications was described. Next, FPGA technology and its basics were explained with examples on where this technology is most applied. Afterwards USB 3 specification was explained.

Next chapter was dedicated to gigabit transceivers. Its principal concepts were introduced and description of Xilinx GTP transceiver was made.

Implementation of gigabit readout with Artix 7 FPGA was presented in chapter 4. RTL modules were developed and verified in simulation. These modules interface the gigabit transceiver on the FPGA with USB 3 communication card. The design is responsible for proper data alignment, propagation and generation of frame packets in compliance with devised specification. Used IP modules were also described.

Results measured by the readout system showed one percent bit error rate. That differs from the error rate that was analyzed from oscilloscope decoder, which shows no errors. However, oscilloscopes use different data recovery approaches, making them more immune to jitter and frequency variations. Thus, solutions have been proposed to investigate the cause of discrepancy in bit error rate measured by the oscilloscope and readout system.

## 6.1 Discussion

Development of this work has faced multiple difficulties. One of them being incorrect simulation for the GTP core. Since reset handshake signals are generated within the transceiver IP and their timeout is around 2 ms, the

simulation of GTP design took at least 30 minutes for all timeouts to expire and transfer the first data. There is an attribute that sets the simulation mode for the IP, however it was not working. After long debugging, the correct attribute inside the GTP was discovered and correctly overridden afterwards. This change reduced needed simulation time to only 60 us, after which the first data was received.

Regarding the measurement, as the available oscilloscope was operating almost at the limit of its bandwidth, it is assumed that timing and jitter analysis results are not reliable. That biased the jitter measurements because there was not enough data for period detection. Characterisation with specialized high speed serial analyser is currently planned. Total peak to peak jitter measured on the eye diagram was 88 ps when preemphasis was turned on, which equals 281 150 ppm. That is three orders of magnitude from the tolerated value of the GTP, which is ±1250 ppm.

Probable cause for high bit error rate results could be high jitter and phase noise of the reference clock or incorrectly selected termination voltage at receiver side. At the beginning it was configured to 500 mV, however it might not be enough to generate sufficient common mode offset.

## 6.2 Future improvements and next steps

To optimise channel performance numerous methods can be employed. First problem to tackle would be clock quality. Low reference clock jitter is the uppermost requirement for any high speed communication design. Onboard jitter attenuator Si5324 could be utilized to produce low jitter reference clock signal. Furthermore, additional receiver termination analysis and far-end channel characterization should be performed.

Despite the system implementing a frame checker module, it doesn't provide complex insight. Bit error rate analyzer should be integrated into both the GUI application as well as into FPGA fabric. It will provide better perspective to where errors occur as it will analyze the error rate in real time.

# Appendix A

## Bibliography


1. LEWIN, John M.; D'ORSI, Carl J.; HENDRICK, R. Edward. Digital mammography. *Radiologic Clinics of North America*. 2004, vol. 42, no. 5, pp. 871–884. ISSN 00338389. Available from DOI: 10.1016/j.rcl.2004.06.004.
2. BALLABRIGA, Rafael. *The Design and Implementation in* 0.13$\mu$*m CMOS of an Algorithm Permitting Spectroscopic Imaging with High Spatial Resolution for Hybrid Pixel Detectors*. 2009. Available also from: http://cds.cern.ch/record/1259673. Presented on 24 Nov 2009.
3. TALLA, Patrick. *Investigation of photon counting pixel detectors for X-ray spectroscopy and imaging*. 2011.
4. *Example: Gamma interaction*. TEXample.net, 2007. Available also from: https://texample.net/tikz/examples/gamma-interaction/.
5. SCHWANK, J R. Basic mechanisms of radiation effects in the natural space radiation environment. 1994. Available also from: https://www.osti.gov/biblio/10158182.
6. KLECZEK, Rafal; GRYBOS, Pawel; SZCZYGIEL, Robert; MAJ, Piotr. Single Photon-Counting Pixel Readout Chip Operating Up to 1,2 Gcps/mm$^2$ for Digital X-Ray Imaging Systems. *IEEE Journal of Solid-State Circuits*. 2018, vol. 53, no. 9, pp. 2651–2662. Available from DOI: 10.1109/JSSC.2018.2851234.
7. PERIĆ, Ivan. A novel monolithic pixelated particle detector implemented in high-voltage CMOS technology. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. 2007, vol. 582, no. 3, pp. 876–885. ISSN 0168-9002. Available from DOI: https://doi.org/10.1016/j.nima.2007.07.115. VERTEX 2006.

8. PENNICARD, David; PIRARD, Benoît; TOLBANOV, Oleg; INIEWSKI, Krzysztof. Semiconductor materials for x-ray detectors. *MRS Bulletin*. 2017, vol. 42, no. 06, pp. 445–450. ISSN 0883-7694. Available from DOI: 10.1557/mrs.2017.95.

9. *UG474: 7 Series FPGAs Configurable Logic Block* [online]. Xilinx, 2016-09. Available also from: https://docs.xilinx.com/v/u/en-US/ug474_7Series_CLB. v1.8.

10. *UG473: 7 Series FPGAs Memory Resources* [online]. Xilinx, 2019-07. v1.14.

11. *UG471: 7 Series FPGAs SelectIO Resources* [online]. Xilinx, 2018-05. Available also from: https://docs.xilinx.com/v/u/en-US/ug471_7Series_SelectIO. v1.10.

12. ANDERSON, Donovan. Introduction to USB 3.0. *MindShare, Inc., nd,[retrieved Sep. 30, 2010]*. 2009.

13. *USB 3.2 Specification* [online]. USB-IF [visited on 2022-04-20]. Available from: https://www.usb.org/document-library/usb-32-specification-released-september-22-2017-and-ecns.

14. ATHAVALE, Abhijit; CHRISTENSEN, Carl. *High-Speed Serial I/O Made Simple: A Designers' Guide, with FPGA Applications*. 1st ed. Xilinx, Inc., 2005.

15. HOLLAND, Nick; SOLUTIONS, Serial Gigabit. Interfacing between lvpecl, vml, cml, and lvds levels. *Texas Instruments Application Report*. 2002, vol. 192.

16. *UG482: 7 Series FPGAs GTP Transceivers User Guide* [online]. Xilinx, 2016. v1.9.

17. NIEHOFF, Brian. *Understanding NRZ and PAM4 Signaling* [online]. Samtec, 2020 [visited on 2022-05-15]. Available from: https://blog.samtec.com/post/understanding-nrz-and-pam4-signaling/.

18. DEVICES, Advanced Micro. *High Speed Serial* [online]. 2022 [visited on 2022-05-10]. Available from: https://www.xilinx.com/products/technology/high-speed-serial.html.

19. *UG952: AC701 Evaluation Board for the Artix-7 FPGA User Guide* [online]. Xilinx, 2019. v1.4.

20. *Xilinx Artix-7 FPGA AC701 Evaluation Kit*. XILINX. Available also from: https://www.xilinx.com/products/boards-and-kits/ek-a7-ac701-g.html.

21. HAFTING, Yngve. *Metastability and Clock domain crossing* [online]. University of Oslo, 2021 [visited on 2022-05-13]. Available from: https://www.uio.no/studier/emner/matnat/ifi/IN3160/v21/timeplan/in3160-l92-clock-domains.pdf.

22. *FT600Q-FT601Q IC Datasheet* [online]. Future Technology Devices International Ltd, 2017. Available also from: `https://www.xilinx.com/support/documentation/user_guides/ug473_7Series_Memory_Resources.pdf`. 1.05.

23. MILLER, CM. High-Speed Digital Transmitter Characterization Using Eye Diagram Analysis. *HEWLETT-PACKARD JOURNAL*. 1994, vol. 45, no. 4, pp. 29–37. ISSN 94303-0724.

24. DEHLAGHI, Behzad; MAGIEROWSKI, Sebastian; BELOSTOTSKI, Leonid. A 12.5-Gb/s On-Chip Oscilloscope to Measure Eye Diagrams and Jitter Histograms of High-Speed Signals. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2014, vol. 22, no. 5, pp. 1127–1137. Available from DOI: `10.1109/TVLSI.2013.2265895`.

25. HAGEDORN, Julian; ALICKE, Falk; VERMA, Ankur. *How to Measure Total Jitter (TJ)* [online]. Texas Instruments, 2012 [visited on 2022-05-17]. Available from: `https://www.ti.com/lit/an/scaa120b/scaa120b.pdf`.

26. TRISCHITTA, Patrick R.; VARMA, Eve L. *Jitter in Digital Transmission System*. USA: Artech House, Inc., 1989. ISBN 089006248X.

27. GUMMALLA, Ajay. *BER Requirements* [online]. IEEE, 2001 [visited on 2022-05-12]. Available from: `https://www.ieee802.org/3/efm/public/nov01/khermosh_3_1101.pdf`.

# Appendix B

# List of abbreviations

| Abbrievation | Meaning |
|---|---|
| FPGA | Field Programmable Gate Array |
| CPLD | Complex Programmable Logic Device |
| GTP | Gigabit Transciever Peripheral? |
| BER | Bit Error Rate |
| FIFO | First In First Out |
| SIPO | Serial In, Parallel Out |
| PISO | Parallel In, Serial Out |
| PLL | Phase Locked Loop |
| ILA | Integrated Logic Analyser |
| IP | Intellectual Property |
| DC | Direct Current |
| AC | Alternating Current |
| SoM | System on Module |
| ISO | International Organization for Standardization |
| USB | Universal Serial Bus |
| ECAD | Electronic Computer-Aided Design |
| $I^2C$ | Inter-Integrated Circuit |
| ADC | Analog to Digital Converter |
| LED | Light Emitting Diode |
| MOSFET | Metal Oxide Semiconductor Field Effect Transistor |
| SRAM | Static Random Access Memory |
| DRAM | Dynamic Random Access Memory |
| UART | Universal asynchronous receiver-transmitter |
| CMOS | Complementary Metal–Oxide–Semiconductor |
| EMI | Electro-Magnetic Interference |
| ESD | Electric Static Discharge |
| DDR | Double-Data-Rate |
| LVDS | Low Voltage Digital Signalling |
| PAM | Pulse-Amplitude Modulation |
| BERT | Bit Error Rate Test |
| BER | Bit Error Rate |
| RTL | Register-Transfer Level |
| GUI | Graphical User Interface |

# Appendix C

# GTP Wizard configuration

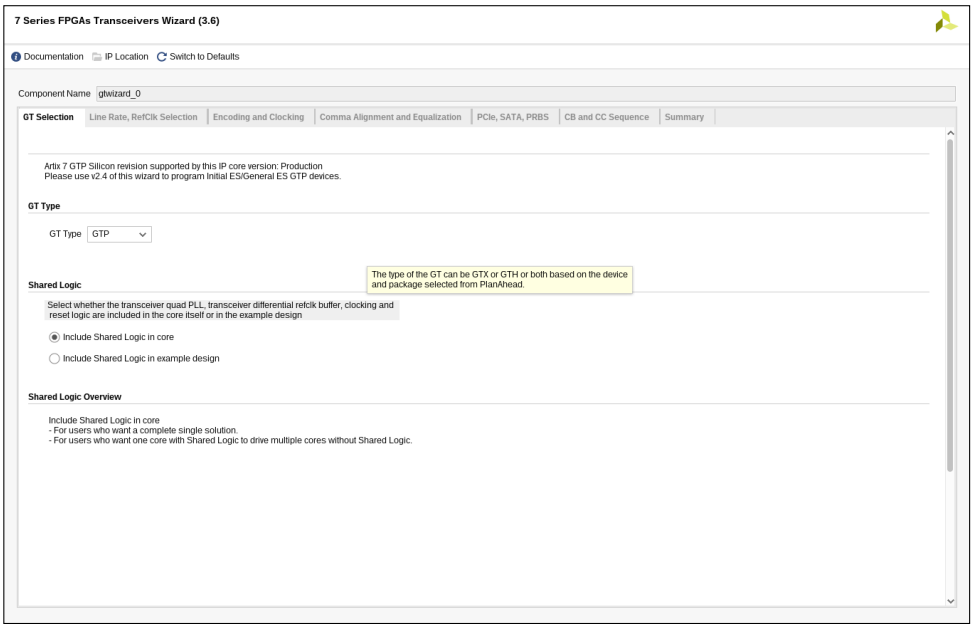



**Figure C.1:** GTP - transceiver selection

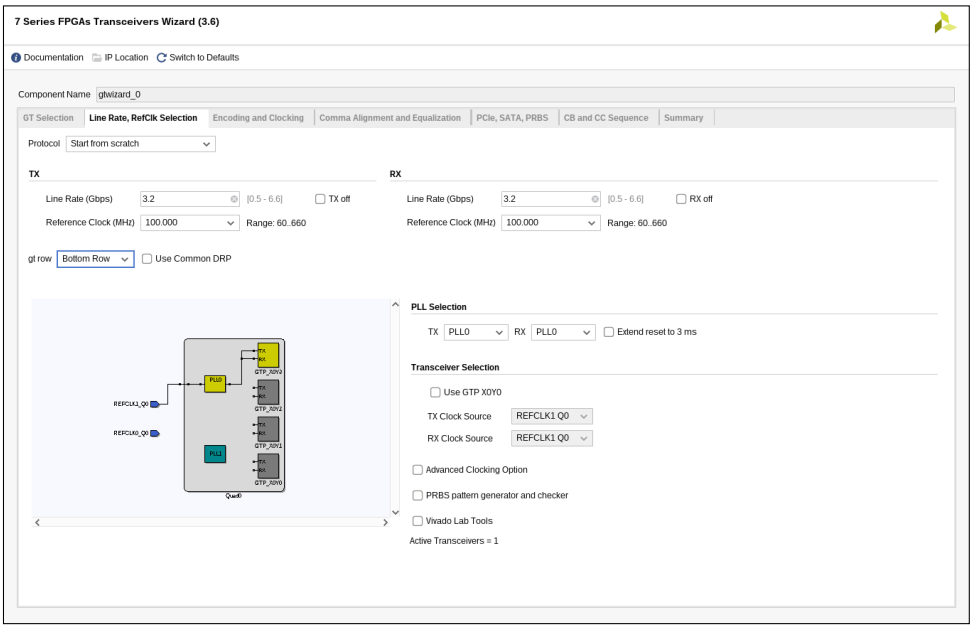



**Figure C.2:** GTP - line rate, clock reference

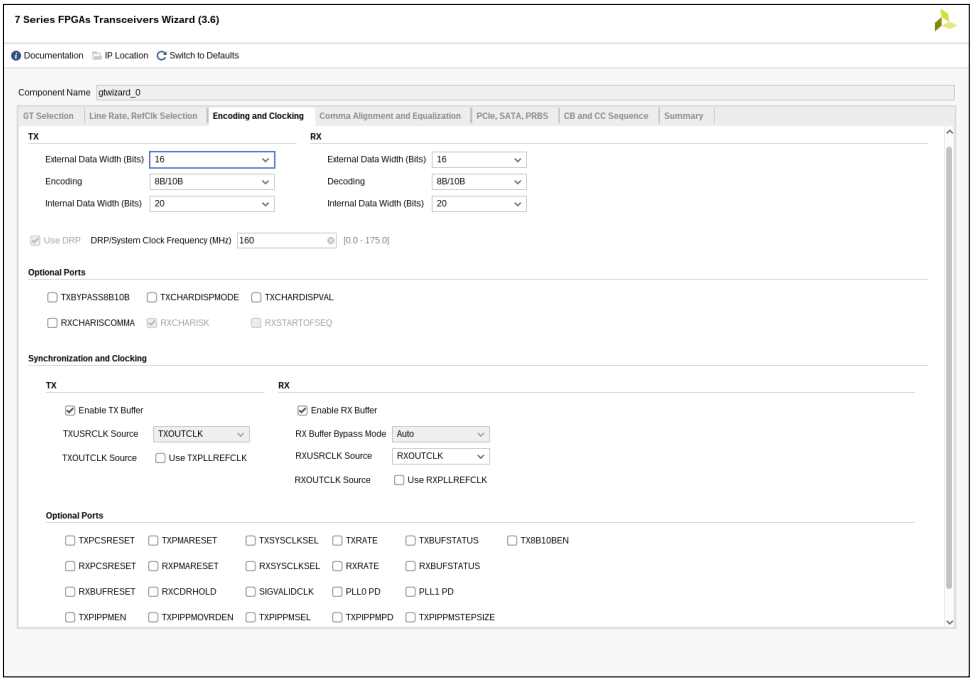



**Figure C.3:** GTP - encoding configuration

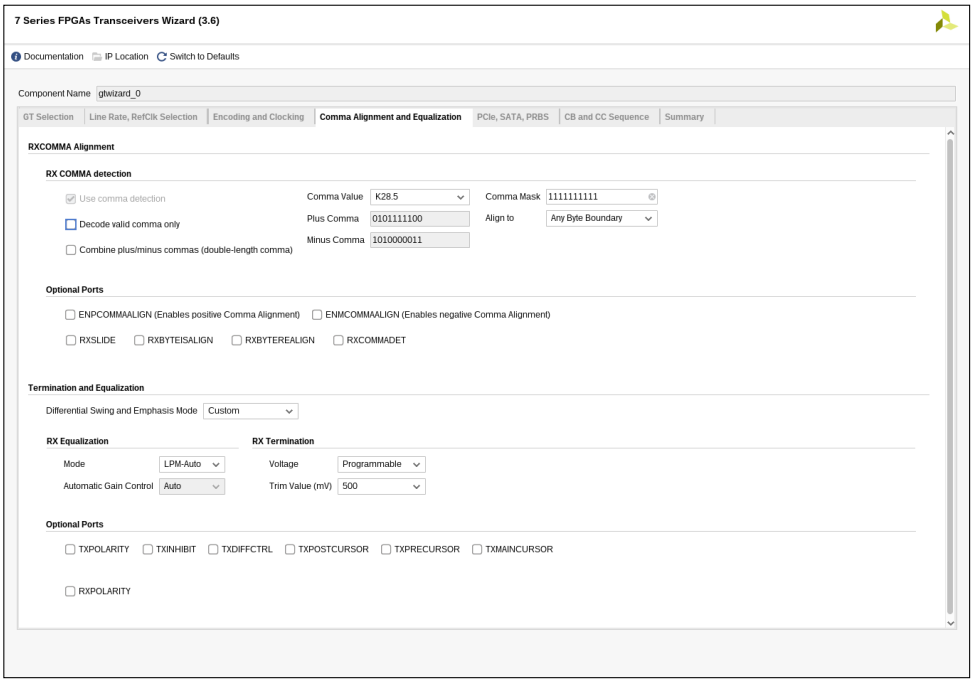



**Figure C.4:** GTP - comma detection

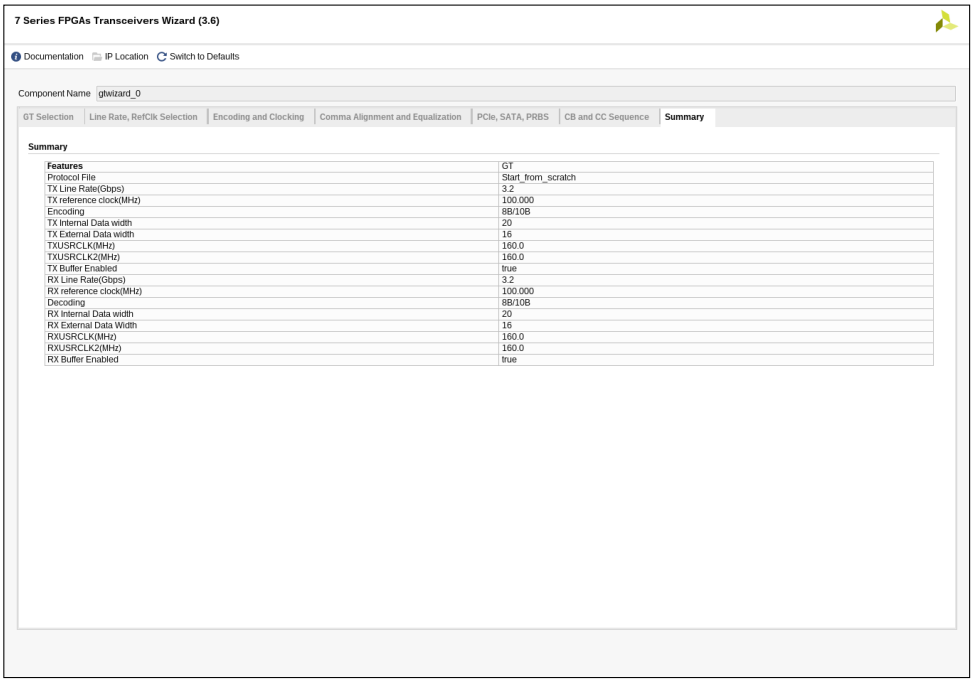



**Figure C.5:** GTP - configuration summary

# Appendix D

## Attached files structure

```
/
├── Colorpix_MGT_Readout/
│   ├── FTDI/
│   ├── ip/
│   └── readout/
│       ├── RTL/
│       └── SIM/
├── gtp_readout_gui/
├── c_sample_ft600/
├── data_analysis_jupyter/
└── thesis_latex_source
```