BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

MASTER'S THESIS

Brno, 2016 Bc. Marek Novák

BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF RADIO ELECTRONICS
ÚSTAV RADIOELEKTRONIKY

FSO TRANSCEIVER FOR LINK QUALITY ESTIMATION
FSO VYSÍLAČ/PŘIJÍMAČ PRO MĚŘENÍ KVALITY SPOJE

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE

Bc. Marek Novák

SUPERVISOR
VEDOUCÍ PRÁCE

prof. Ing. Otakar Wilfert, CSc.

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Elektronika a sdělovací technika**
Ústav radioelektroniky

Student: Bc. Marek Novák ***ID:*** 147684
Ročník: 2 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

FSO vysílač/přijímač pro měření kvality spoje

POKYNY PRO VYPRACOVÁNÍ:

Optická komunikace volným prostorem nabízí výhody v podobě velké šířky pásma a možnosti bezlicenčního provozu. Proveďte rešerši nad možnostmi zmírnění negativních atmosferických vlivů na přenos dat touto cestou. Zaměřte se na možnosti odhadu přenosové funkce prostředí a možnostmi zlepšení bitové chybovosti přenosu na základě této znalosti pomocí ekvalizace. Dále se zamyslete nad možnostmi použití MAP a MLSE ekvalizérů pro praktickou realizaci modulu optického přijímače-vysílače pro FSO.

Realizujte prototyp modulu pro zpracování optického signálu a jeho dekódování. Do modulu bude pomocí optického vlákna zaveden přijatý signál. Vaším úkolem bude použít výše zmíněné techniky pro jeho zpracování a správné vyhodnocení pomocí FPGA, na kterém bude také naprogramována MAC vrstva 1000/100/10Mbit/s Ethernetové rozhraní. Skrze toto rozhraní bude tedy možné posílat rámce přes optický most. Dále umožněte přístup k parametrům probíhající komunikace za účelem vytváření statistik.

DOPORUČENÁ LITERATURA:

[1] BOUCHET, O. et al. Free-Space Optics. Propagation and Communication. London: ISTE, 2006. ISBN 10: 1-905209-02-9

Termín zadání: 8.2.2016 ***Termín odevzdání:*** 19.5.2016

Vedoucí práce: prof. Ing. Otakar Wilfert, CSc.
Konzultant diplomové práce:

doc. Ing. Tomáš Kratochvíl, Ph.D., *předseda oborové rady*

ABSTRACT

This thesis deals with bit error ratio diminution of a free space optical link using the principle of reciprocity applied to the communication channel along with a selectable coding. Low-density parity check coding and Reed-Solomon coding is implemented, due to their acceptable performance. Residual frame error rate ratio is calculated and accessible as output of the system, which is implemented on a field programmable gate array chip.

KEYWORDS

FSO, LDPC, FPGA, VHDL, Reed-Solomon coding, Turbulence, SFP, Ethernet

ABSTRAKT

Tato diplomová práce pojednává o zmírnění bitové chybovosti bezkabelového optického spoje s užitím principu reciprocity aplikovaného na komunikační kanál, spolu s možností kódování přenášených dat. V této práci je implementováno LDPC a Reed-Solomonovo kódování pro jejich vyhovující vlastnosti. Zbytková rámcová chybovost je vypočtena a k dispozici jako výstup systému, který je implementovaný v hradlovém poli (FPGA).

KLÍČOVÁ SLOVA

FSO, LDPC, FPGA, VHDL, Reed-Solomonovo kódování, Turbulence, SFP, Ethernet

NOVÁK, Marek *FSO vysílač/přijímač pro měření kvality spoje*: master's thesis. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Radioelectronics, 2016. 83 p. Supervised by prof. Ing. Otakar Wilfert, CSc.

DECLARATION

I declare that I have written my master's thesis on the theme of "FSO transceiver for link quality estimation" independently, under the guidance of the master's thesis supervisor and using the technical literature and other sources of information which are all quoted in the thesis and detailed in the list of literature at the end of the thesis.

As the author of the master's thesis I furthermore declare that, as regards the creation of this master's thesis, I have not infringed any copyright. In particular, I have not unlawfully encroached on anyone's personal and/or ownership rights and I am fully aware of the consequences in the case of breaking Regulation § 11 and the following of the Copyright Act No 121/2000 Sb., and of the rights related to intellectual property right and changes in some Acts (Intellectual Property Act) and formulated in later regulations, inclusive of the possible consequences resulting from the provisions of Criminal Act No 40/2009 Sb., Section 2, Head VI, Part 4.

Brno ....

author's signature

ACKNOWLEDGEMENT

I would like to thank my diploma thesis mentor prof. Ing. Otakar Wilfert, CSc. for guidance and support during the elaboration of this thesis. I would also like to thank Ing. Michal Kubíček, Ph.D. for guidance in the field of FPGA system design. Last, but not least, I would like to express my gratitude to my family and friends for moral support and motivation. I would like to point out here the support of Bc. Lukáš Janík with whom I have spent many hours mutually discussing my thesis from both implementation and design point of view. Thank you.

Brno ....

author's signature

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

ACKNOWLEDGEMENT

Research described in this master's thesis has been implemented in the laboratories supported by the SIX project; reg. no. CZ.1.05/2.1.00/03.0072, operational program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

author's signature

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

The optical communication is nowadays, unlike the radio communication, a completely unexhausted means of data exchange. The capacity of an optical channel is uncomparable to the radio one, still it is not often used for free space communication links. The main reason being the atmosphere and the way it interacts with the optical signal.

The aim of this work is to describe these interactions in first place and thereafter implement a technique to reduce the bit error ratio in free space optics. Several possible approaches to reduce the BER will be described with emphasis on the MLSE technique as the most promising and feasible one for a practical FPGA-based communication system. Towards the end of this work, the overall architecture of the communication system will be addressed, the HDL modules it is composed of will be described and some practical benchmarking results will be provided.

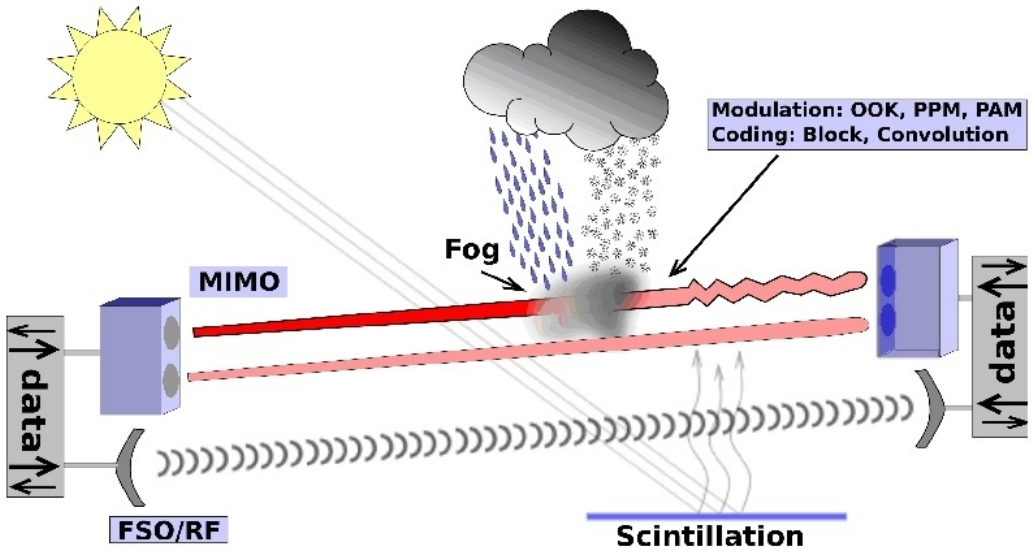

Fig. 1: Illustration of the problem [1]

1 ATMOSPHERIC EFFECTS IN FSO

This section describes how atmospheric turbulence is the major problem in free space optics and how the losses caused by this effect scale with the influence of other interactions.

Atmospheric Interaction Types Interactions between the communication laser beam and the atmosphere can be separated in two main types - extinction due to atmospheric aerosols and molecules versus optical turbulence causing random laser beam intensity fluctuations [10].

1.1 Beam Extinction

Extinction is caused either by scattering or by absorption or by a combination of these effects. Depending on the size of the object which interacts with the beam, it is further divided in aerosol extinction and molecular extinction.

In case of the first one, the diameter of the particles is in the range of 0.002 μm to 100 μm - small enough to stay in the atmosphere. The diameter also influences the lifetime of the particle in the atmosphere, where smaller particles stay longer active. For example above the sea water, the sea-spray and higher humidity causes aerosols to be larger in size, but to have a shorter lifetime.

The molecular extinction manifests itself thanks to the absorption of incident radiation energy by air-contained molecules, which appears in discrete quanta. Therefore the absorption spectrum is composed of several discrete lines appearing in certain range of wavelengths, depending on the associated transition in the molecule. Table 1.1 sums up absorption bands. FSO is mainly concerned by vibrational spectra.[13]

To express the attenuation caused by mentioned effects we use the transmittance, which is defined by formula 1.1.

$$T_\nu(s) = \frac{I_\nu(s)}{I_{\nu,0}} = e^{-\beta_\nu s} \tag{1.1}$$

Where $\beta_\nu s$ is the total extinction coefficient comprising all aerosol scattering, aerosol absorption, molecular scattering and molecular absorption. Its dimension is km^{-1}.

The following is a practical example of how the beam extinction demonstrates itself. Typical values of total exticition parameter are given in the table 1.2. In our

Transition mode	**Wavelengths affected**
Rotation	10 cm to 100 μm
Vibration	100 μm to 1 μm
Electronic transition	visible and ultraviolet

Tab. 1.1: Absorption bands [15]

Air conditions	$\beta_\nu s$ [km^{-1}]
Clear	0.1
Haze	1.0
Fog	>10
Dense Fog	391

Tab. 1.2: Total extinction coefficient typical values[13]

case, let's calculate the loss for a link with optical length of 3 km in hazy environment 1.2.

$$\begin{aligned} T &= e^{-\beta_\nu s} = e^{-1 \cdot 3} = e^{-3} \\ L_a &= 10 \log_{10}(T) = 10 \log_{10}(e^{-3}) = -13.03\mathrm{dB} \end{aligned} \tag{1.2}$$

Losses introduced are -13.03 dB.

The scattering can be also divided into two categories - Mie scattering and Rayleigh scattering. The first one is more important for FSO links, since its impact to the SNR (signal to noise ratio) nearly does not depend on the wavelength of the propagating laser beam. Therefore its negative effect cannot be avoided. The second form of scattering is not as important for smaller distance FSO links (with $L < 3$ km), since it depends on the wavelength - careful selection of the wavelength can mitigate its impact.

1.2 Optical Turbulence

Definition

Optical turbulence is an atmospheric effect that consists of random variations in the refractive index of the atmosphere, which are induced by spatially changing air

density and temperature. [23]

Per definition of the refractive index (1.3), its variations cause a distortion to the propagating light wave front, since the velocity at which each of its part propagates is not constant. This produces a fluctuation of the optical intensity at the receiver's optical aperture. The level of fluctuation can be measured using the scintillation index and depends on the refractive index structure parameter C_n^2, which is further described in subsection 2.3.2.

$$n = \frac{c}{v} \tag{1.3}$$

Where n is the refractive index, c is the speed of light in vacuum and v is the speed of light in a medium. [8]

Furthermore, when the size of the "turbles" produced by the optical turbulence is larger than the transmitted beam diameter, the beam tends to be deflected, which causes a complete signal loss. This effect is known as beam wander. [22]

1.2.1 Beam Wander

When beam wander starts to appear, it may cause the incident beam to completely miss the receiver's aperture, making the signal completely disappear. Since the apparition of optical turbles is a relatively slow process, the beam wander can be well compensated by the use of the adaptive optics, which can compensate these signal losses optically. In next chapter, an alternative technique will be presented based only on a electronic-based compensation, which can be more resource saving compared to expensive adaptive optics setup.

1.2.2 Omnipresence of Optical Turbulence

Unlike the process of absorption, which can be reduced by selecting appropriate wavelengths for communication (e.g. 1550 nm), the optical turbulence does not have a "window", where its effects would not apply. It is therefore crucial to develop a method to avoid its effects, since it cannot be removed from the system by just selecting a right wavelength. Although with decreasing wavelength, the influence of turbulence slightly decreases as well, as you can see on the figure 1.1. According to the figure, for links with the range smaller than 5 km, the wavelength does not influence much the level of the attenuation. On the same figure we can notice the influence of the diameter of the receiver's lens - smaller the diameter, bigger impact does the turbulence have.

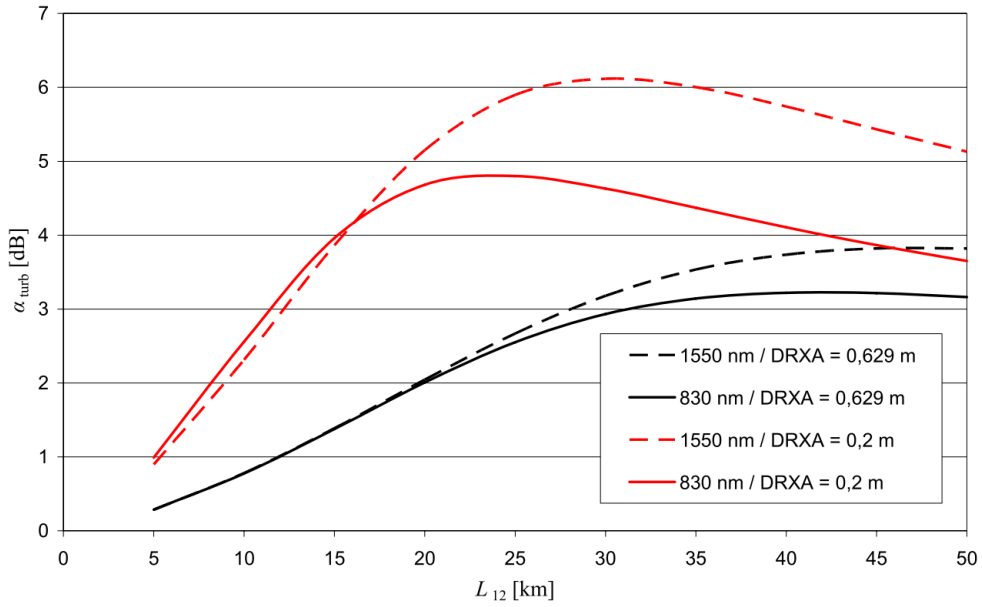

Fig. 1.1: Attenuation caused by turbulence for two values of receiver's lens diameter and for two wavelengths in function of the link range [11]

2 LASER BEAM PROPAGATION

In order to understand, how certain parameters of the communication system affect the scintillation index and consequently the bit-error ratio, let's define a commonly used laser beam model - the Gaussian beam.

2.1 Gaussian Beam

In optics, the Gaussian beam is a particular solution of the Helmholtz equation 2.1 (same as a planar wave). This model produces the best description of coherent radiation such as the laser beams, even if it does not describe completely the effect of diffraction.

$$\nabla^2 A + k^2 A = 0 \tag{2.1}$$

Where ∇^2 is the Laplacian, A is the amplitude and k is the wavenumber.

More specifically, a Gaussian beam is a beam, whose transversal profile evolution of amplitude in function of the spatial propagation is proportional to a Gaussian function.

Definition

There are multiple ways to define the Gaussian beam. Historically, the Gaussian beams were used in optics as a solution to the Helmholtz equation 2.1 using the paraxial approximation, which allows only a small divergence of the beam towards its axis of propagation. The maximum allowed angle is about 20 degrees.

There are other approaches coming from the electromagnetism allowing to obtain a formulation of the Gaussian beam. This way, we can define monomode and multimode Gaussian beams as a particular case in the paraxial approximation of one or multiple complex sources of rays.[2]

2.2 Propagation of Gaussian Beam

The approximation done here presumes scalar Gaussian beam, where the electric field is considered linearly polarised in a direction orthogonal to its direction of propagation. This approximation gives good results when the beam waist w_0 is superior to the wavelength. If this constraint is not met, a simple scalar description is not valid.

Assuming propagation in the z direction ($+z$) and polarization in the x axis, the electric field in phasor notation is given by 2.2.

$$\mathbf{E}(r,z) = E_0\hat{x}\frac{w_0}{w(z)}\exp(\frac{-r^2}{w(z)^2})\exp(-i(kz + k\frac{r^2}{2R(z)} - \psi(z))) \tag{2.2}$$

Where: [20]

r is the radial distance from the center axis of the beam

z is the axial distance from the beam's waist

$i = sqrt(-1)$

$k = 2\pi/\lambda$ is the wave number

$w(z)$ is the radius at which the field amplitudes fall to $1/e$ of their axial values, at the plane z along the beam

$w_0 = w(0)$ is the waist size

$R(z)$ is the radius of curvature of the beam's wavefronts at z

$\psi(z)$ is the Gouy phase at z

$\hat{x}$ is the unit vector in the x direction

More commonly, however, the intensity or irradiance is used to describe the beam, which is more easily measurable, since it averages the electrical field over time per 2.3.

$$I(r,z) = \frac{|\mathbf{E}(r,z))|^2}{2\eta} = I_0(\frac{w_0}{w(z)})^2\exp(\frac{-2r^2}{w^2(z)}) \tag{2.3}$$

Where η is the characteristic impedance of the medium ($\eta_0 = 377\Omega$ for free space).

Thus $I_0 = E_0^2/2\eta$ is the intensity at its waist, at the centre of the beam.

2.2.1 Evolving Parameters

In the equations 2.2 and 2.3, several parameters are evolving along the beam following the z axis.

$$w(z) = w_0\sqrt{1 + (\frac{z}{z_0})^2} \tag{2.4}$$

Where $w(z)$ is denoted as "evolving beam width" and z_0 is called the Reyleigh range 2.7. This parameter defines the spot size at position z.

$$R(z) = z\left[1 + (\frac{z_0}{z})^2\right] \tag{2.5}$$

This parameter is called "evolving radius of curvature" and defines the curvature of the wavefronts. At the beam waist and as z approaches infinity, the $R(z)$ approaches infinity as well. For $z = z_0$, we obtain $R(z_0) = 2z_0$.

Next parameter is the Gouy phase, which is not normally observable during experiments, but has its importance in theoretical description of the Gaussian beam. 2.6

$$\psi(z) = \arctan(\frac{z}{z_0}) \tag{2.6}$$

This parameter, called Gouy phase, changes between $-\pi/2$ and $+\pi/2$ and explains the apparition of higher-order Gaussian modes.[17]

2.2.2 Beam parameters

The shape of the beam is determined by multiple parameters - wavelength λ, beam waist w_0. All other subparameters depend on these two parameters.

Rayleigh range, confocal parameter and beam divergence

$$z_0 = \frac{\pi w_0^2}{\lambda} \tag{2.7}$$

This parameter is called Rayleigh distance or Rayleigh range. It defines a point, where the wavefront curvature is greatest $(1/R)$.[26] Rayleigh distance is sometimes used in form of confocal parameter $(b = 2z_0)$.

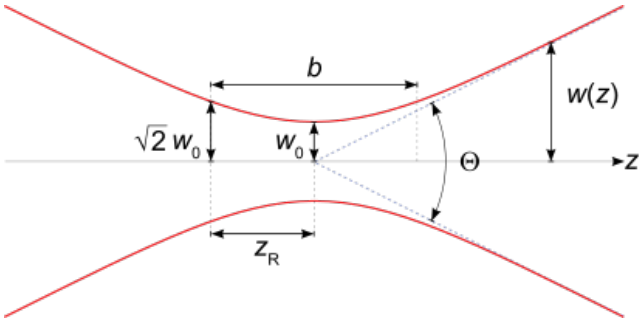

Fig. 2.1: Gaussian beam parameters [21]

$$\theta \approx \frac{\lambda}{\pi w_0}[rad] \tag{2.8}$$

The divergence of the beam is a parameter defined by the angle between the central axis of the beam and the lines of the cone, which forms the edge of the beam for $z \gg z_0$.

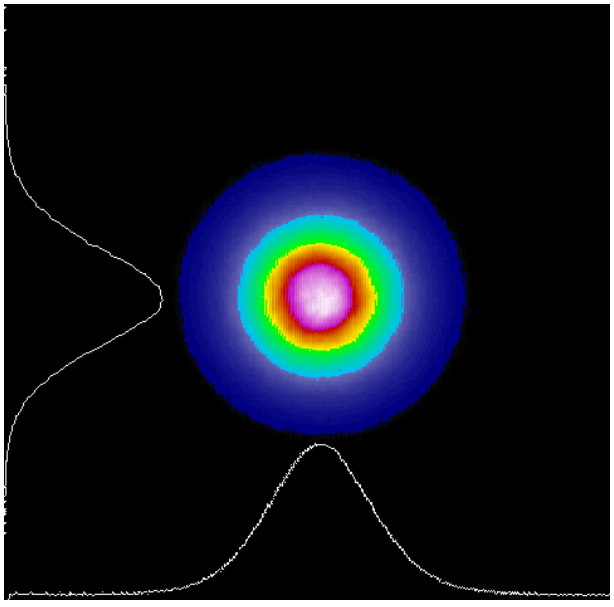

Fig. 2.2: Gaussian beam spot measured by a detector [16]

2.3 System Parameters Influencing the BER

In previous section, a collimated beam was assumed. Let's now generalize the relation 2.4 to a situation, where the focusing parameter 2.10 is not equal to 1 ($\hat{r} \neq 1$). This will cause the beam width to grow faster as in case of the collimated beam. Obviously, when the beam width becomes larger, a bigger receive optics are required otherwise the received optical intensity is smaller, which induces a worse SNR.

Another parameter influencing the BER is the length of the link. Longer the link, greater the scintillation index 2.14, as shown in figure 2.4.[13]

2.3.1 Optical Aperture Parameters

The generalized form for the beam size is expressed by 2.9.

$$w(z) = w_0\sqrt{(\hat{r}^2 + \zeta\hat{z}^2)} \tag{2.9}$$

Where $\hat{r}$ is the focusing parameter defined in 2.10, ζ is the global coherence parameter defined in 2.11 and $\hat{z}$ is distance with its diffractive component ($\hat{z}_d$) defined in 2.13. We have $\hat{z} = \frac{z}{\hat{z}_d}$.

$$\hat{r}(z) = \frac{R_0 - z}{R_0} \tag{2.10}$$

Where R_0 is the radius of curvature at optimum focusing. This parameter is called focusing parameter, the hat symbol expresses it is a vector.

$$\zeta = \zeta_S + \frac{2w_0^2}{\rho_0^2} \tag{2.11}$$

This parameter is called global coherence parameter and have two additive components - ζ_S, which describes spatial coherence properties of the signal-carrying laser beam when leaving the transmitter ($\zeta_S = 1$ for coherent beam, $\zeta_S > 1$ for partially coherent beam). The second term depends on ρ_0, which is a coherence length of a spherical wave propagating in optical turbulence as defined in 2.12.

$$\rho_0 = (0.55 C_n^2 k^2 z)^{-3/5} \tag{2.12}$$

This parameter is called coherence length and in case optical turbulence is present (anywhere in the Earth's atmosphere), it depends on the refractive index structure parameter C_n^2. The C_n^2 parameter depends on the atmosphere propreties, thermal turbles presence and their dynamic propreties.

$$\hat{z}_d = \frac{k w_0^2}{2} \tag{2.13}$$

This parameter is called the diffractive distance and it depends on the beam size at the origin.

How the coherence propreties of the transmitted beam (ζ_S) involve increase in the beam size is depicted in the figure 2.3. As already mentioned, widening the beam size means losses at the receiving aperture, since it cannot grow without limitation (economic reason, feasibility etc.).

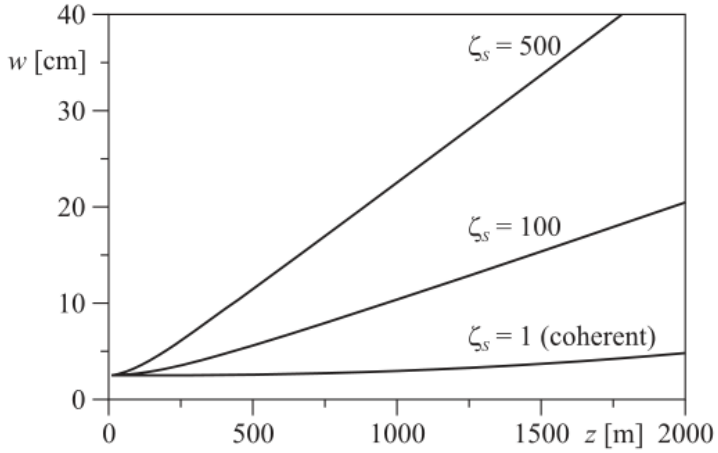

Fig. 2.3: Collimated beam size as a function of range for several choices of phase diffuser ($\lambda = 0.785$ µm, $w_0 = 2.5$ cm and $C_n^2 = 10^{-}14\ \mathrm{m}^{-2/3}$)[10]

2.3.2 Scintillation Index

While the generalized formula for the beam width (2.9) implies, that in case of bad coherence, the received intesity will diminuate faster in function of the range than in case of coherent beam. This would not be the major problem if an optical noise component were not present. This component can be measured and expressed by a parameter called scintillation index and is defined in 2.14. This parameter depends on the refractive index structure parameter C_n^2, which can be seen in the figure 2.4.

$$\sigma_{\ln S}^2 = \frac{\langle I^2(r,z)\rangle}{\langle I(r,z))\rangle^2} - 1 \tag{2.14}$$

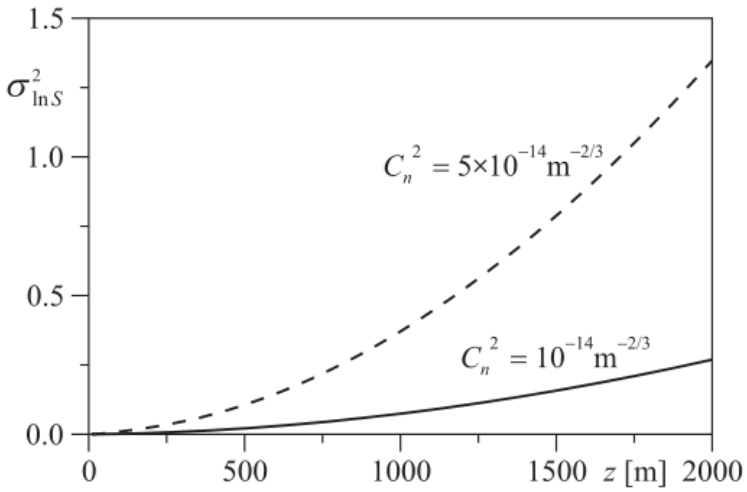

Fig. 2.4: Scintillation index in function of the range ($\hat{r} = 1$, $\rho_S = 100$, $\lambda = 1.55$ µm, $w_0 = 2.5$ cm)[10]

The scintillation index can be reduced by increasing the receiving lens diameter. However, this solution does not work well for a higher C_n^2 parameter value, as it can be seen at the figure 2.5.

2.3.3 BER in Function of Received Power

As we can see at the figure 2.6, the BER depends mainly on the $P_{opt}[dBm]$, ρ_S and the C_n^2 parameter.

The first one can be adjusted by increasing or decreasing the transmitting power. A system designer would want to keep the received power under the saturation level

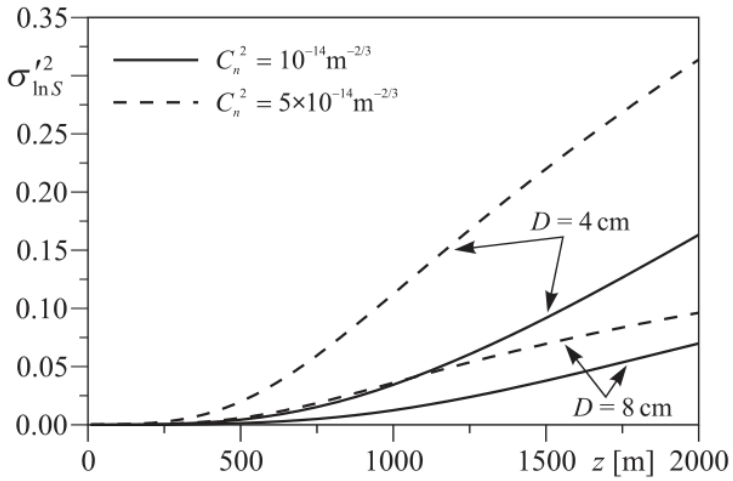

Fig. 2.5: Aperture averaged scintillation index as function of channel distance ($\hat{r} = 1$, $\rho_S = 100$, $\lambda = 1.55$ µm, $w_0 = 2.5$ cm) [10]

of the receiver's PIN diode, but also to keep it as high as possible to be more in the right side of the figure 2.6.

The second parameter can be changed by selecting an appropriate transmitting aperture. When we have $\rho_S \approx 10$, the receiver aperture averaging effect helps to reduce the scintillations and signal fades. Also selecting a greater receiver lens diameter can help to reduce the scintillations.

The last parameter is determined by current state of the atmosphere and can have nearly no influence on the transmission during certain periods of the day time (from the evening until the morning, when it is cloudy etc.), but the same parameter can on the other hand have the strongest influence on the received signal - during a hot day for example. In the following chapter, common techniques to solve this problem will be resumed, and also the techniques planned to be used in the system will be announced and their benefits will be described.

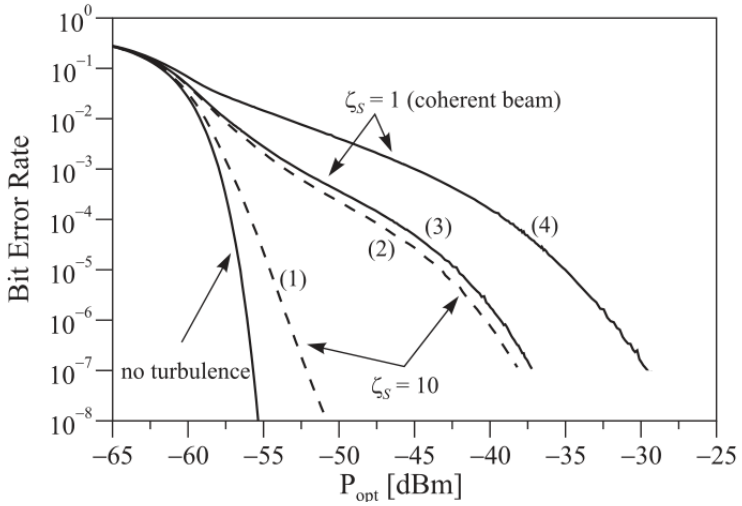

Fig. 2.6: Effect of optical turbulence on BER as a function of received optical power for collimated ($\hat{r} = 1$) coherent ($\rho_S = 1$) and partially coherent ($\rho_S = 10$) beams ($\lambda = 0.785\ \mu\text{m},\ w_0 = 2.5\,\text{cm},\ z = 2000\,\text{m},\ D = 10\,\text{cm}$). Left to right (1) $\rho_S = 10, C_n^2 = 10^{-15}\,\text{m}^{-2/3}$, (2) $\rho_S = 10, C_n^2 = 1.210^{-14}\,\text{m}^{-2/3}$, (3) $\rho_S = 1, C_n^2 = 10^{-15}\,\text{m}^{-2/3}$, (4) $\rho_S = 1, C_n^2 = 1.210^{-14}\,\text{m}^{-2/3}$ [10]

3 APPROACHES TO IMPROVE BER IN FSO

Based on previous chapters, several BER improving methods will be presented here and those which will be consequently used in the implementation of the system will be announced.

3.1 Multiple Input, Multiple Output (MIMO)

MIMO is a technique, which is used mostly in RF link designs to exploit the phonomenon of multipath propagation to increase the spectral efficiency of the communication. In case of FSO, the situation is however a little bit different - the main task is to mitigate the atmospheric turbulence.

MIMO principle is based on using M transmitters and N detectors. The signals from the N detectors are then summed either using equal gain for each branch or weighted gains. [27] If the number of receivers approaches infinity a gain of up to 25 dB over a single-input, single-output configuration can be attained. [12]

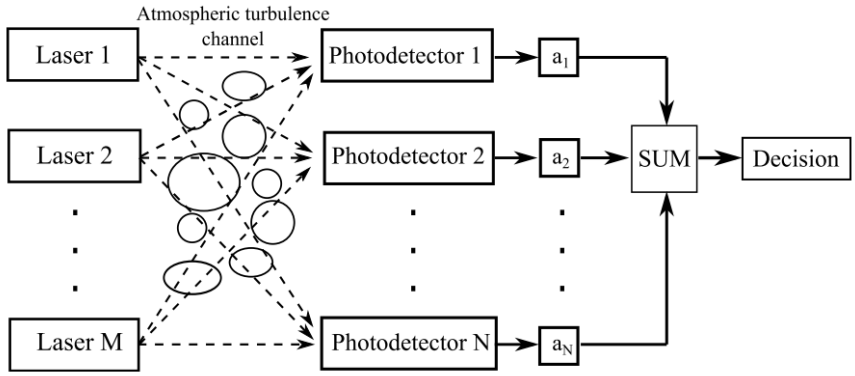

Fig. 3.1: Block diagram of M x N MIMO FSO system over atmospheric turbulence channel [3]

In [4], more realistic figures are obtained: for a given C_n^2 and link distance, a gain of 5 dB at ASER of 10^{-5} is obtained when upgrading from SISO configuration to 2x2 MIMO or from 2x2 MIMO to 4x4 MIMO. What's more, the average spectral efficiency is improved by 2 b/s/Hz in case of the upgrade. These results were obtained using the Monte Carlo simulation of a turbulent channel, results are given in 3.2.

The subject of this thesis is however not to exploit the MIMO technique, but it can be combined with the methods, which will be used to cooperatively improve the BER.

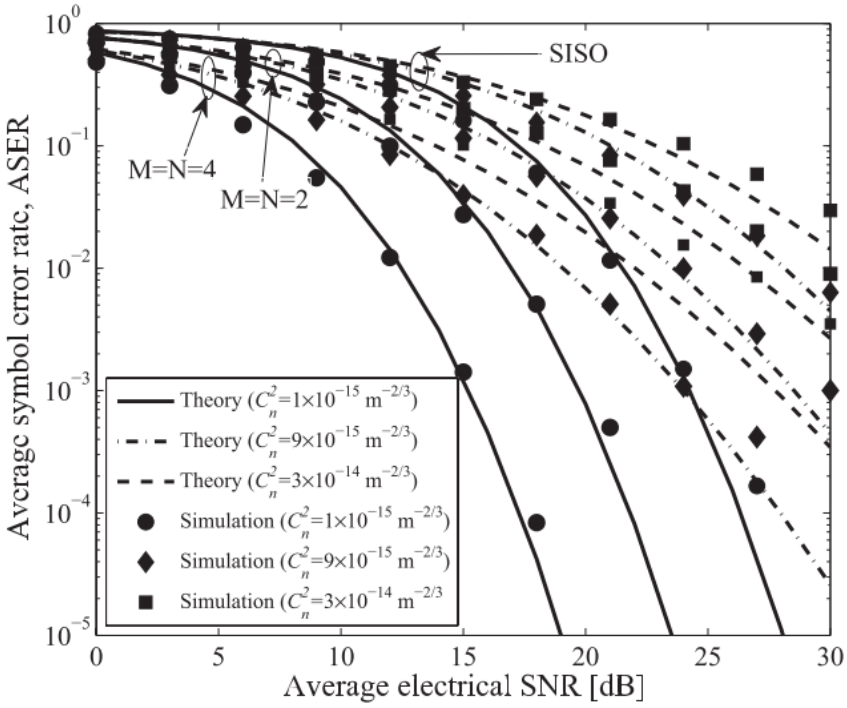

Fig. 3.2: ASER versus the average SNR of SISO, 2x2 MIMO and 4x4 MIMO FSO systems for various C_n^2. Link distance is $L = 4000$ m. [4]

3.2 The Aperture Averaging

For weak to moderate level of atmospheric turbulence, increasing the diameter of the receiver lens can reduce the scintillation index $\sigma^2_{\ln S}$ as seen in 2.5 and 1.1. This technique however does not work well for strong level of turbulence and is not cost effective. It is however one of the most obvious techniques.

3.3 Adaptive Optics

Another technique to reduce the impact of atmospheric turbulence is the adaptive optics. This method consists of compensating the distortion of the beam wave-front adaptively according to the current beam wave-front. The realization of this can be made in the optical domain as well as in the electrical domain according to the picture 3.3. The wave-front corrector can be implemented by a 132-actuator MEMS piston-type deformable mirror controlled by a VLSI system as in [25]. This method is however not aimed in this thesis.

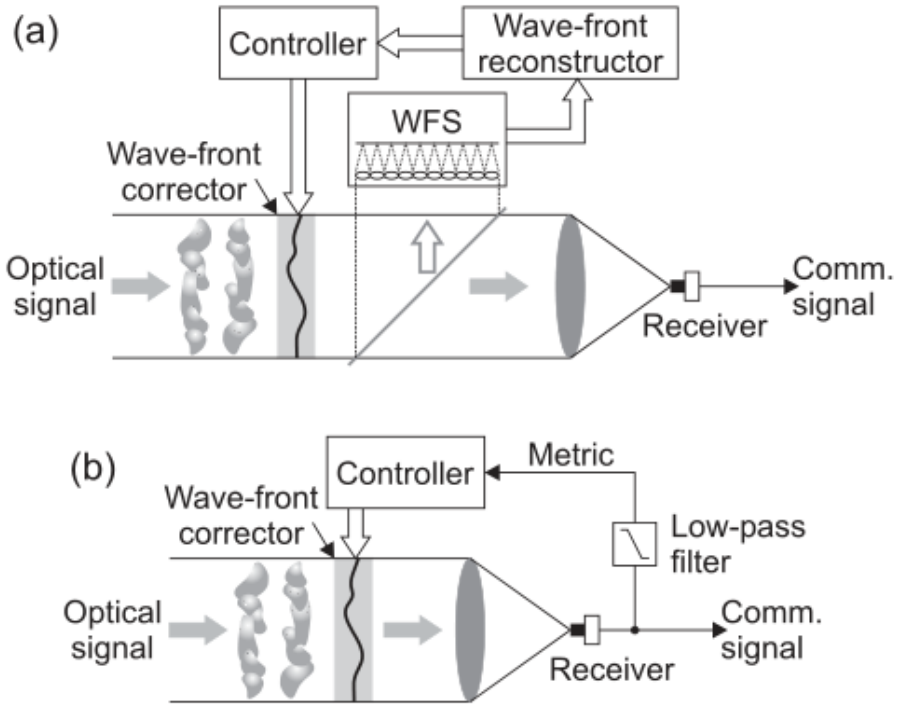

Fig. 3.3: Schematics of adaptive optics approaches for FSO communication. (a) Conventional adaptive optics uses a wave-front sensor (WFS) and wave-front reconstruction for control of the wave-front corrector. (b) Wave-front distortion compensation based on blind optimization of a system performance metric may use the received signal strength (determined from the communication signal after low-pass filtering) for the feedback [25]

3.4 Channel Coding

Channel coding also known as forward error correction (FEC) is computionally one of the most complex techniques control the errors in data transmission. This approach was first applied by mathematician Richard Hamming in 1940s - the implementation is the well known Hamming(7, 4) linear eror-correction code. [14] The principle resides in adding a redundant information in the data stream, which can be consequently used to recover the lost data.

3.4.1 Reed-Solomon Coding

The Reed-Solomon error correction coding was first released to public in 1960 and was defined by Irving S. Reed and Gustave Solomon. This code can be classified as

non-binary cyclic error-correcting code. Even if Reed and Solomon did not cooperate with creators of BCH code, RS code is a subset of BCH code.

By adding $2t$ check symbols to the data, RS code is capable of detecting $2t$ erroneous symbols or to correct up to t symbols.

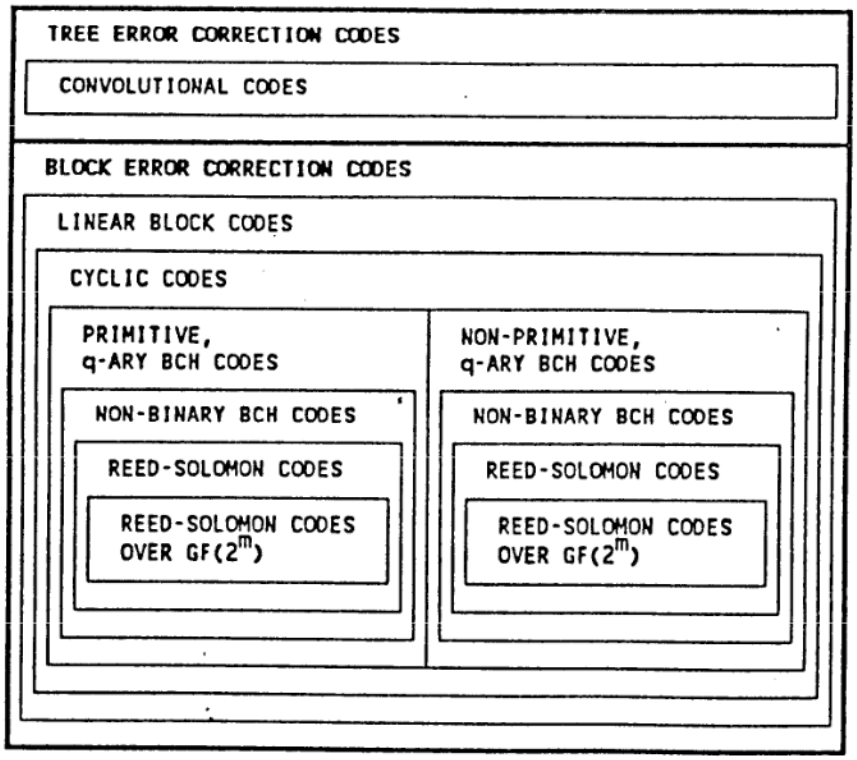

Fig. 3.4: Venn diagram to classify the RS code [5]

If we have for example a message consisting of k payload symbols and we want it to be transferred over an error-prone channel allowing up to t errors to occur (up to t erroneous symbols), we would need to send n symbols over the channel as in equation 3.1.

$$n = k + 2t \tag{3.1}$$

Where: [7]

n is the number of symbols, which will be transmitted to the channel

k is the number of payload symbols, which we want to transfer reliably

t is the number of errors we can fix in the RS decoder

As the RS codes are systematic block codes, the payload or let's say information bits are placed in the most significant byte positions of the resulting codeword and the remaining LSBs are computed by the encoder. As the RS codes support both errors and erasures to occur, some information symbols in the codeword can be erased before the actual transmission to reduce the coding rate. This way the number of tolerable errors is however reduced accordingly. Having MSBs be sent first to the medium, the figure 3.5 shows a conceptual schematic of a RS encoder. After all payload symbols are sent to the encoder, the switch changes its state and starts outputting the parity symbols. [19]

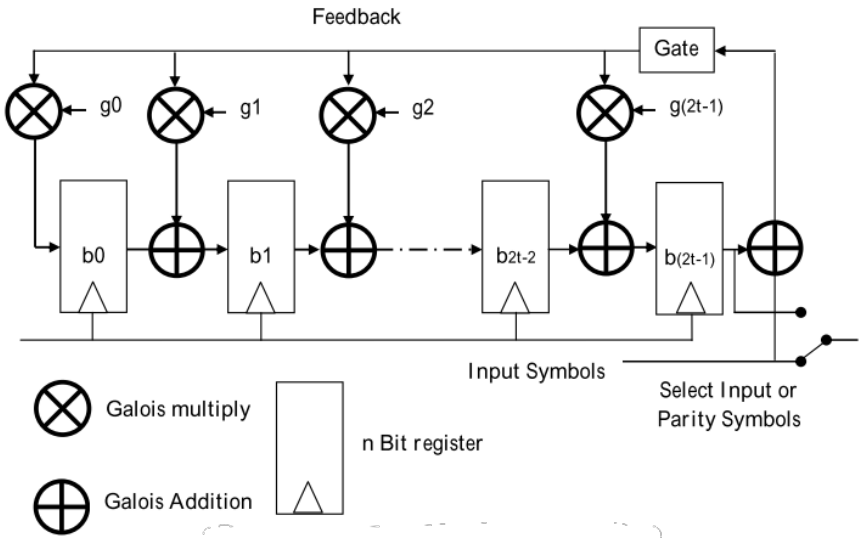

Fig. 3.5: Block diagram of RS encoder [19]

On the other side of the channel, at the receiver, the decoding occurs in five steps:

- Syndrome calculation
- Determine error-location polynomial
- Solving the error locator polynomial - Chien search
- Calculating the error magnitude
- Error correction

Each step is further discussed in [19]. The figure 3.6 shows the flow of results in the decoder. The decoder is the most complex part of the RS code implementation and will be in detail discussed in the implementation section of this thesis.

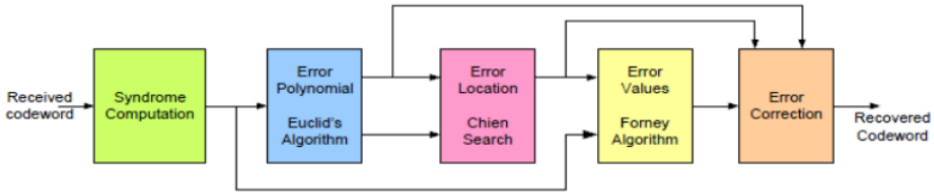

Fig. 3.6: Architecture and data flow in an RS decoder [19]

3.4.2 Convolutional Coding

Convolutional codes have an advantage over the block codes such as the mentioned Reed Solomon codes - the payload does not have to be framed in blocks. This implies multiple practical advantages:

- The data does not have to be aligned to the block boundary - random length data can be send without additional logic processing it
- The frame/block synchronization does not need to be present

We have the advantage to use the 8B10B coding in our application, so the data is already extracted from the channel and message boundaries are recovered. In this case, the only disadvantage of the block codes is a padding at the end of the last block forming the decoded application frame. If this disadvantage would need to be further mitigated, a frame-merging logic can be implemented so only in case of an interruption of the data stream a padding would need to be added.

The realization of convolutional encoder is quite simple - the input data is fed in a shift register, whose outputs are then wired to modulo two adders (XOR gates), which creates k output data bits from n input data bits. The resulting coding rate is therefore n/k. If n is desired to be different from n in the coding rate, the output data stream needs to be punctured - output data stream bits are erased following an erasure patern vector.

The decoding is again the most complex process and cannot be described in a single paragraph as the encoder. The most commonly used algorithm for decoding convolutional code is the Viterbi algorithm 3.8, which is the most efficient one. Another alternative is the Fano algorithm. The principle consists in searching for the most likely binary sequence, which is further described in following subsection.

3.4.3 MLSE

MLSE stands for Maximum Likelihood Sequence Estimation and it is a method capable of reconstructing a signal with erroneous bits in it. This task can be accomplished by using the Viterbi decoder.

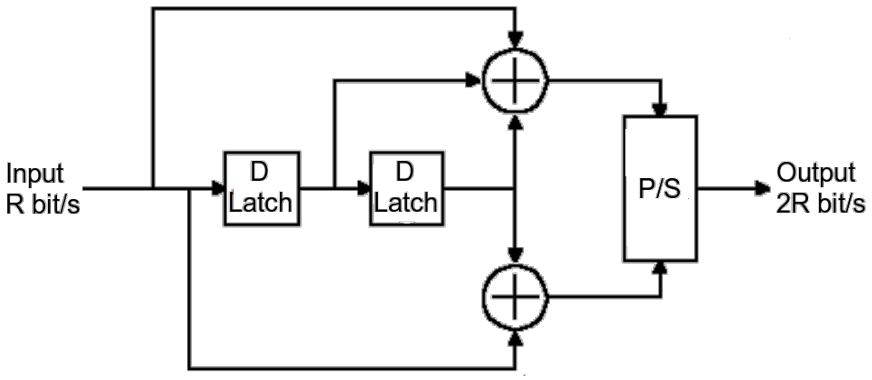

Fig. 3.7: Example of convolutional encoder with $k = 1/2$

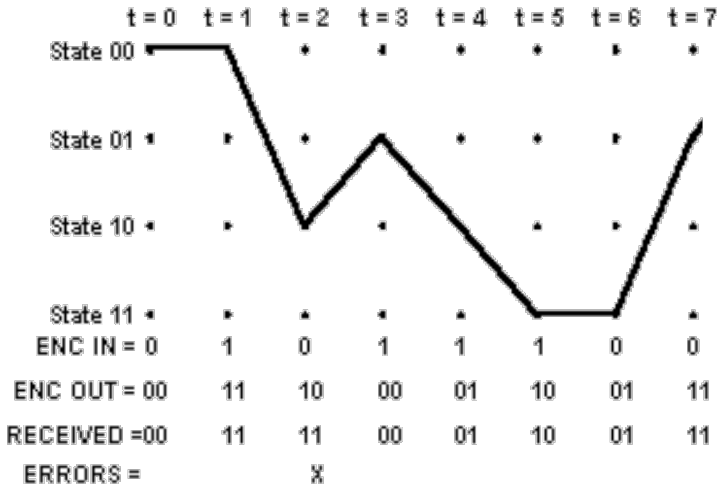

Fig. 3.8: Example of Viterbi decoder decoding process [6]

Let's explain how Viterbi decoder works following the example from the previous subsection. (3.8 and 3.7) The chosen convolutional code will create the output binary stream according to table 3.1. As you can see, only one half of the transitions is allowed (due to 1/2 coding rate).

The Viterbi decoder evaluates at each tap the metrics of any possible path to chose based on the received dibit and memorizes up to K previous taps' metrics. An accumulated path metrics is calculated and at the end, the Trellis diagram path with the best (smallest) accumulated metric is selected, which gives a sequence of states the decoder FSM went through. This sequence translates directly to the most probable binary sequence which was transmitted by the other communicating side. To be complete, the accumulated metrics is computed as a sum of metrics at each tap. It is obvious, that the decoder cannot memorize an infinite number of previous

	Next State	
Current State	Input = 0	Input = 1
00	00	10
01	00	10
10	01	11
11	01	11

Tab. 3.1: Transition table

states' metric and so only K previous taps is considered. In practice, K has a value going from 5 to 7. K is called the "constraint length".

3.4.4 LDPC Coding [18]

LDPC codes are regular block parity check codes, which have a special parity check matrix H. 3.2.

$$s = Hy^T \tag{3.2}$$

Where:

H is the parity check matrix

y^T is a collumn vector of received message bits

s is the syndrome of the received message y

Even though H defines an LDPC code, if we want to actually encode our payload, we need to first calculate a generator matrix from it, which we will do using equation 3.3.

$$G = GaussJordan(H) \tag{3.3}$$

Where:

H is the parity check matrix

G is the codeword generator matrix

$GaussJordan()$ is the Gauss-Jordan elimination

Having calculated the G matrix, we can then calculate our codeword using the relation 3.4.

$$c = uG \tag{3.4}$$

Where:

c is the resulting codeword to be send

G is the codeword generator matrix

u is the input payload bits

The performance of an LDPC code is entirely defined by properties of its H matrix. The required properties are:

- H has to be sparse - the density of ones in the matrix must be low. (It must be nearly all zeroes matrix)
- The number of ones on a row has to be constant.

As you can see, the requirements are not really strict and there are therefore multiple approaches of how to design the H matrix. These are well discussed in [18]. Nevertheless, these requirements ensure that the decoding complexity increases only linearly with the code length as well as the minimum distance increases linearly with the code length.

The major difference between LDPC codes and generic block codes is the way LDPC is decoded, which is based on the sparseness of their matrix H. A general name of the class of decoding algorithms used to decode LDPC codes is "message-passing", among these we can cite:

- Bit-flipping decoding - hard decision algorithm
- Sum-product decoding - soft decision algorithm

Example

To better illustrate how the encoder of an LDPC can be realized, in figure 3.9 an example of the encoder circuit is given of a 108 bits length codeword, 3/4 rate LDPC code.

The parity check matrix of such code is given in 3.5.

$$H = [A_1, A_2, A_3, A_4] \tag{3.5}$$

$$\begin{aligned} a_1(x) &= 1 + x^3 + x^{16} \\ a_2(x) &= x^2 + x^6 + x^8 \\ a_3(x) &= x + x^8 + x^9 \\ a_4(x) &= x + x^{13} + x^{23} \end{aligned}$$

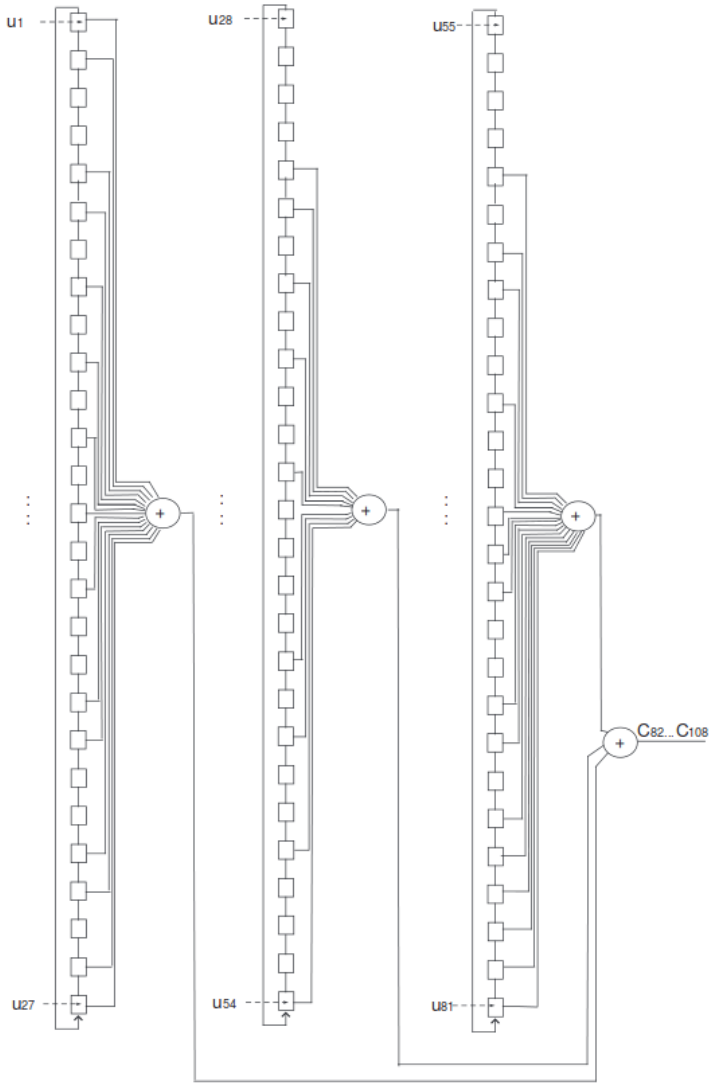

Fig. 3.9: LDPC encoder example, $k = 3/4$ and $L = 108bits$ [18]

The data are fed in the D latches in parallel and in following 28 clock cycles, the parity bits are shifted out of the structure of the encoder. This way a 108 bit long codeword is generated and can be consequently sent to the channel.

3.5 Leveraging Channel Reciprocity

BER reduction method presented in the previous section used the channel coding, which will work both in simplex and full-duplex communication schemes. In our case, however, we want to implement a full-duplex FSO bridge, which can give us a certain advantage over the simplex communication system. Let's leverage this.

3.5.1 Comparison with Radio-frequency Channels

In FSO, in case of a full-duplex link, the transmitting an the receiving signals travel through exactly the same environment and effects like multipath propagation does not apply, which allows us to say, that we can apply the principle of reciprocity to it. In case of RF links, this is not possible in a real environment.

3.5.2 Proposed BER Reduction Technique

The proposed technique consists in monitoring the incoming optical signal from the other peer and not to forward data from the ethernet to the channel when the signal from the other peer is not present (Figure 3.10). This approach was already mentioned in article [33].

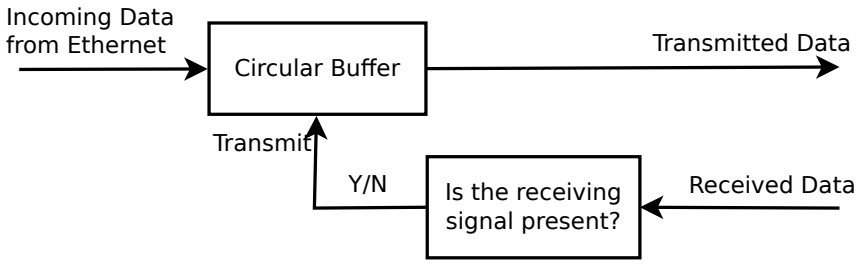

Fig. 3.10: Proposed BER/FER reduction method illustration

3.5.3 Example of Usage of Presented Technique

In our design, we will use an SFP optical module with SGMII communication interface and MDIO management interface. The SFP has however another signaling interface - an LOS (Loss Of Signal) signal, which can be used to detect whether the

other side is receiving our signal or not (by knowing whether we can see the signal from the peer or not). The question is whether the latency of the LOS signal will be small enough to implement well this technique. Another solution is to use the MDIO interface and to read out the RSSI reading from the built-in registers in the SFP module and not to transmit when the RSSI level starts to drop or when it goes under a certain threshold level. The actual implementation will be considered in the last chapter of this thesis.

4 FSO TRANSCEIVER ON FPGA

In this chapter, the hardware and software platform used to implement the system will be briefly introduced.

4.1 Why FPGA?

Today's Field Programmable Gate Array integrated circuits are the best choice when developing a prototype of a high speed communication systems for multiple reasons:

- Hardware serializers and deserializers are present on the chip, which is absolutely necessary for doing a signal processing of a signal at data rate in order of Gbps.
- The hardware is configurable in software, which allows a custom hardware implementation
- The signal processing can be efficiently parallelized
- Many hardware IP blocks are available

Any other alternative to process such a high-speed signal would be a digital signal processor, which does not offer the same level of hardware customization (To illustrate this statement - a DSP can be synthesized on the FPGA, if desired).

4.2 General Architecture

As announced in previous chapters, in this thesis the coding along with the method proposed in chapter 3.5.2 to reduce the BER/FER will be used. The block diagram 4.1 shows the overall system design implemented on Virtex-5 FPGA from Xilinx. The blocks coloured in gray show parts of the system implemented as a part of this thesis. Other blocks are implemented in [9].

4.2.1 Used IP Cores

In order to implement the whole transceiver, these IP cores were so far used:

- Virtex-5 Embedded Tri-Mode Ethernet MAC Wrapper (TEMAC):
  this IP core realizes a user friendly wrapper for the embedded Ethernet MAC controller.
- FIFO memory:
  this IP core was used several times in different configurations to accomplish both different clock domain transitioning as well as frame assembling and internal frame signalling.

- Reed-Solomon encoder and decoder v7.1:
  this IP core was used for channel coding. (239,255) configuration was used, so upto 8 erroneous bytes can be corrected in a block of 239 bytes.
- Memory Interface Generator:
  this IP core from Xilinx was used to implement the interface with the on board DDR2 dynamic RAM. It allows the user to use a FIFO-based API signals to write and read from the DDR2 memory.

4.2.2 Modularity and Interfaceability

One of the aims of this thesis is to create a platform, where multiple channel codes can be applied and their performance can be measured in real communication. Thus the encoder/decoder block is made to contain multiple encoders and decoders (Reed-Solomon, LDPC, convolutional or others). Different coding rates and coding algorithms are to be evaluated over time. The current codec is to be selectable via a web-based interface running on the Microblaze microcontroller, which can access data about current status of the system and setup its parameters. Additionally, the registers of the systems are accessible via the I^2C inteface, so that an external microcontroller could access it.

4.3 ML505 Development Board

The ML505 development board contains the Virtex-5 FPGA from Xilinx, 88E1111 Marvell 1000BASE-TX PHY chip with SGMII interface, SFP interface and DDR2 interface (and others, but these are important for the implementation). VHDL is used to implement discussed design along with the ISE IDE for Xilinx FPGAs.

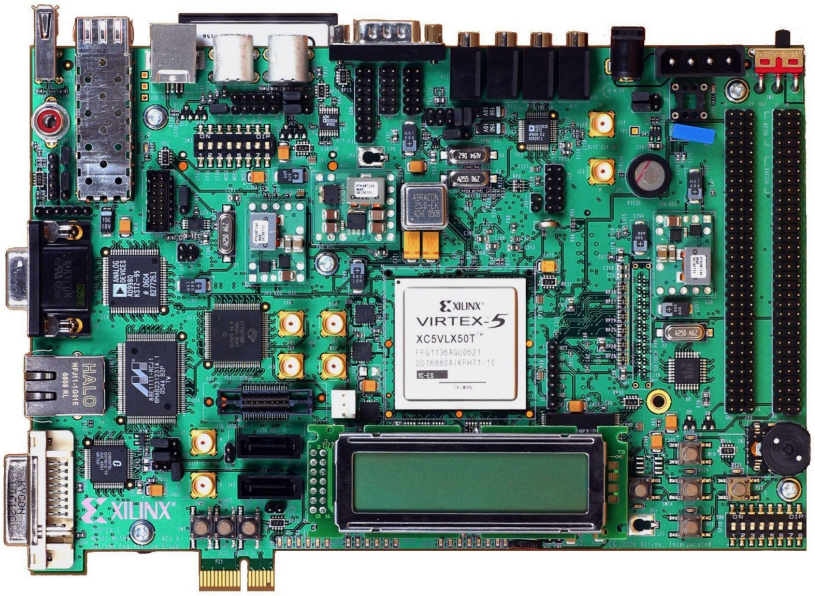

Fig. 4.2: ML505 Development Board, top view

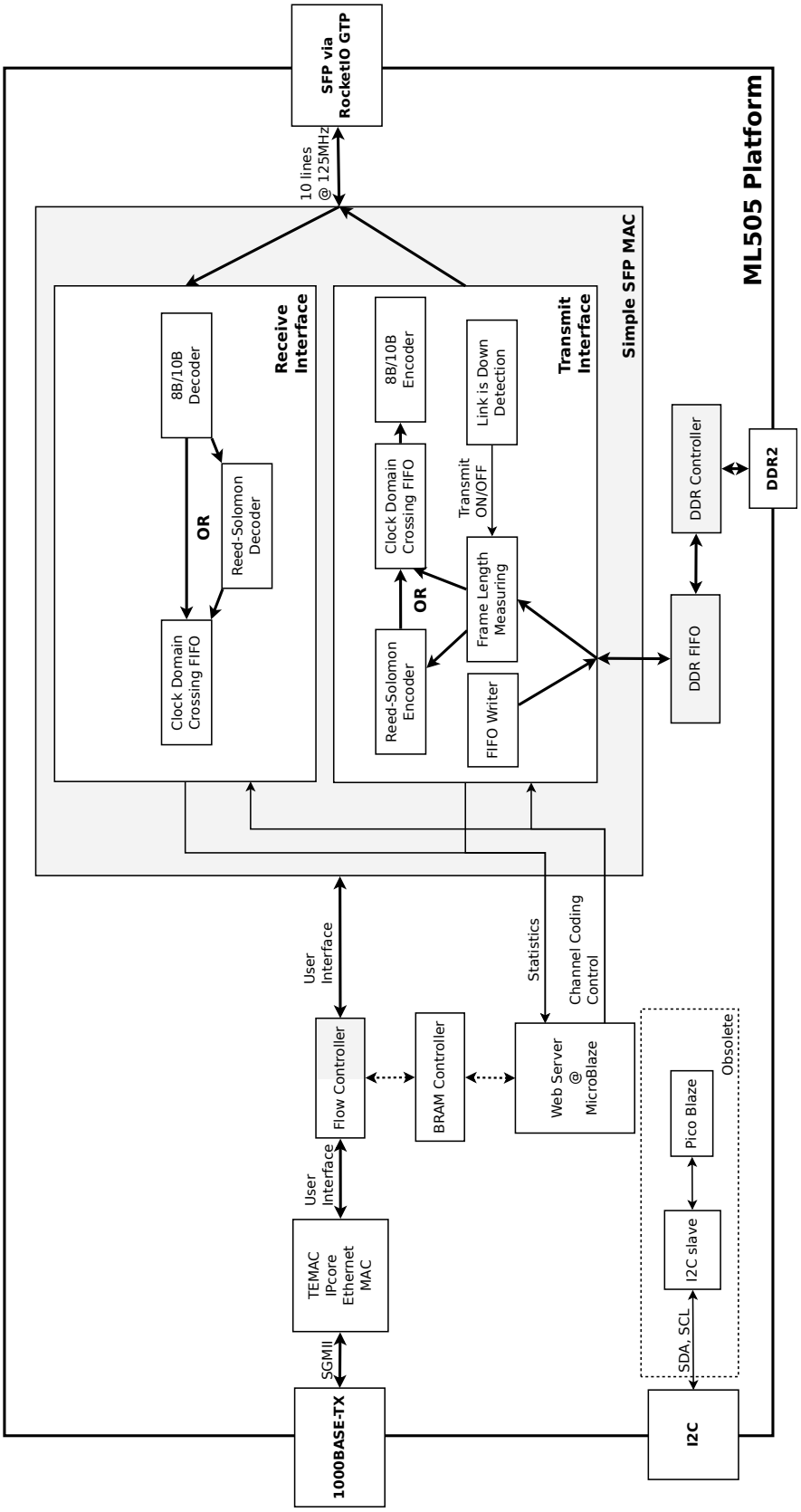

Fig. 4.1: Overall block diagram of the FPGA system

5 IMPLEMENTATION ON ML505

In this chapter, the role of each HDL module will be described as well as its principle of operation. In Figure 5.1, the hierarchy of the design is shown. The grayed modules are mainly targeted by this chapter.

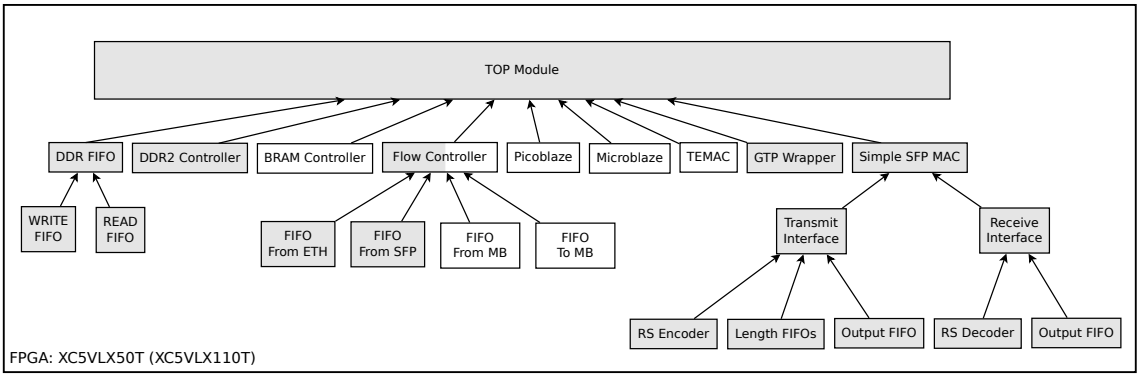

Fig. 5.1: Hierarchy of HDL modules

The design is symmetric and contains two major data paths - Ethernet and SFP. SFP stands for Small Form Pluggable, which is the name of the modular component allowing to change any time the wavelength used for the transmission by changing the transmission and reception diodes. The Ethernet data path was easy to realize, since the TEMAC IP Core is available in the ISE IDE. This core has a simple API based on Good Frame / Bad Frame (GF/BF) signals, Data Valid signal, Acknowledge signal (ACK) and Data Out and Data In signals. [32]

5.1 GTP Wrapper

This module is part of the physical layer (PHY) of the SFP data path. It takes care both of the transmission and reception of the high speed communication signal. It contains several optional hard-wired components, which can be instantiated using the Virtex-5 FPGA RocketIO GTP Transceiver Wizard from Xilinx.

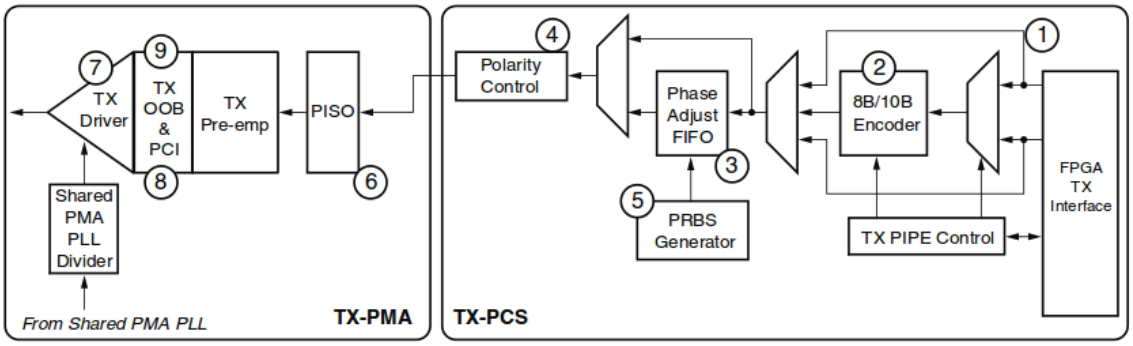

Fig. 5.2: GTP TX Block Diagram[28]

In Figure 5.2, all the optional parts of the transmit chain are shown. In the design, only Polarity Control, Paralel-In Serial-Out (PISO), TX Driver and Shared PMA PLL Divider modules are actively used. The 8B/10B Encoder is not used, since it is implemented in the Simple SFP MAC module, described in next section. One of the most important parts of the transmit chain is the PISO, since it allows us to do all the signal processing in a symbol-based manner. The symbol rate is ten times lower than the actual serialized transmit frequency. (In considered design 125 MHz vs. 1.25 GHz)

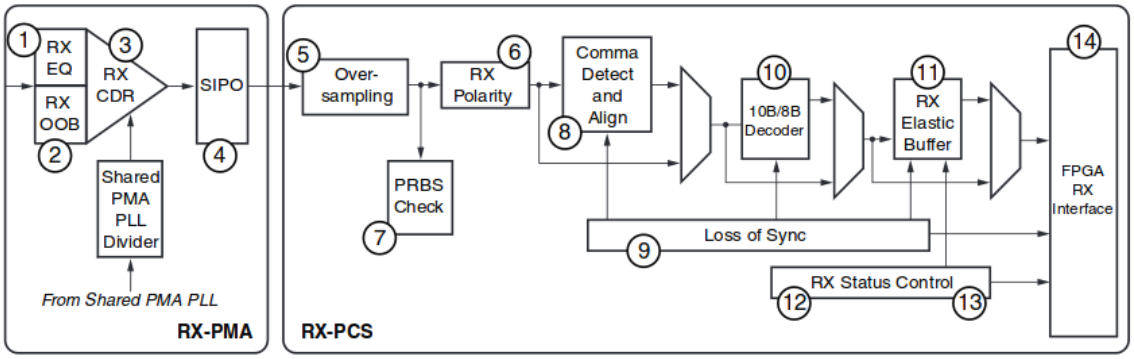

Fig. 5.3: GTP RX Block Diagram[28]

Figure 5.3 shows the receive chain. Only RX Clock Data Recovery (CDR), Serial-In Paralel-Out (SIPO), RX Polarity and Shared PMA PLL Divider modules are used in the design receive path. SIPO is analogical to PISO in the transmitter. Another interesting part of this block is the RX CDR, which recovers the clock from the received data. A basic view at this block is depicted in Figure 5.4. All receive logic until the Output FIFO of the receive part of the Simple SFP MAC module is clocked by this recovered clock signal. The FIFO in the MAC takes care of the

clock domain crossing. CDR provides also an information about a successful phase alignment of the on-chip PLL with the clock recovered from the received data. If the received signal disappears (The receiving diode can get saturated or the received signal is too weak), CDR drives high a signal, which is used further in the Simple MAC for statistics and for stopping output of the transmit data to prevent data losses. If the synchronization is lost, a finite state machine (FSM) restarts the alignments process.

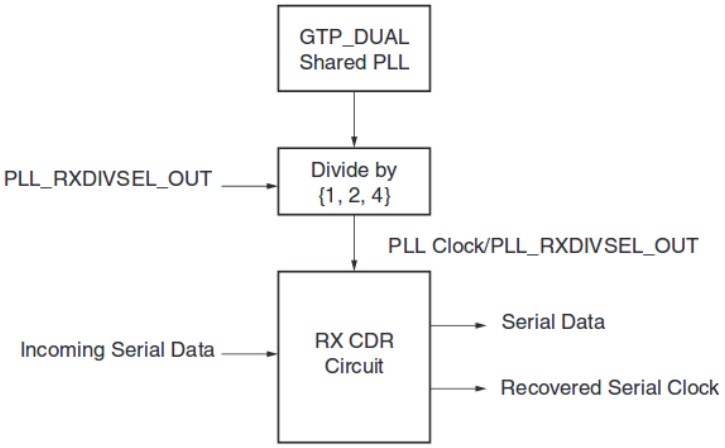

Fig. 5.4: Clock Data Recovery[28]

5.2 Simple SFP MAC Module

Unlike the GTP Wrapper, which is a wrapper realizing instantiation of a semi-analog (CDR, PLL) block, Simple SFP MAC is completely synthesized digital circuit. Its behavior is simulated by the *SimpleMACandFlowController.vhd* test bench along with the Flow Controller, which is a simple Ethernet switch implementation. (Available in the GIT repository)

The Simple SFP MAC has the same interface signals for frame reception and transmission as the TEMAC Ethernet MAC module, so that it can be easily interfaced by users used to this common interface.

5.2.1 Transmission Path

Figure 5.5 shows how the transmit interface operates. First, the Data Valid (DV_TX) signal is driven high and the transmit logic waits for the Acknowledge (ACK_TX) signal to go high. Then on each subsequent rising edge of the clock signal, a byte of

the payload is fed in the Data Out ($DOUT\_TX$) port. When the DV_TX signal goes low, the last byte of the frame was sent to the MAC.

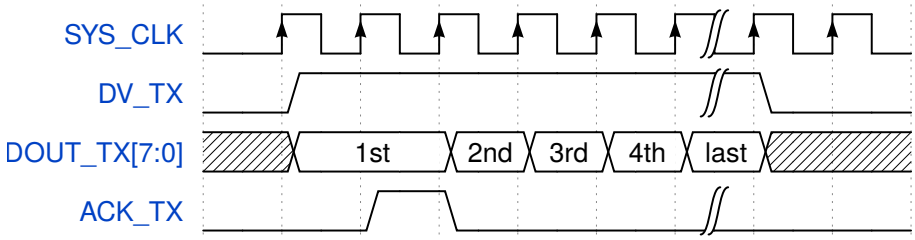

Fig. 5.5: Transmit interface, TEMAC compatible

When the data is clocked to the transmit interface, it gets written to the DDR FIFO 9 bit interface. This is done by a transmit interface process which transforms the 8 bit stream delimited by the DV_TX signal into a 9 bit stream with the nineth bit used for Start-Of-Frame (SOF) and the End-Of-Frame (EOF) signalling. When the frame propagates through the DDR FIFO (described in next section), it is either directly written to the input of a BRAM-based asynchronous output FIFO realizing clock domain crossing, or written to the input of a FIFO measuring the length of each frame. This is used later in the flow to write the frame into the Reed-Solomon encoder IP core. This behavior can be controlled by an interface signal. Figure 5.7 shows the situation when the channel coding is enabled, and Figure 5.6 shows the behavior when it is disabled. Inactive blocks are grayed.

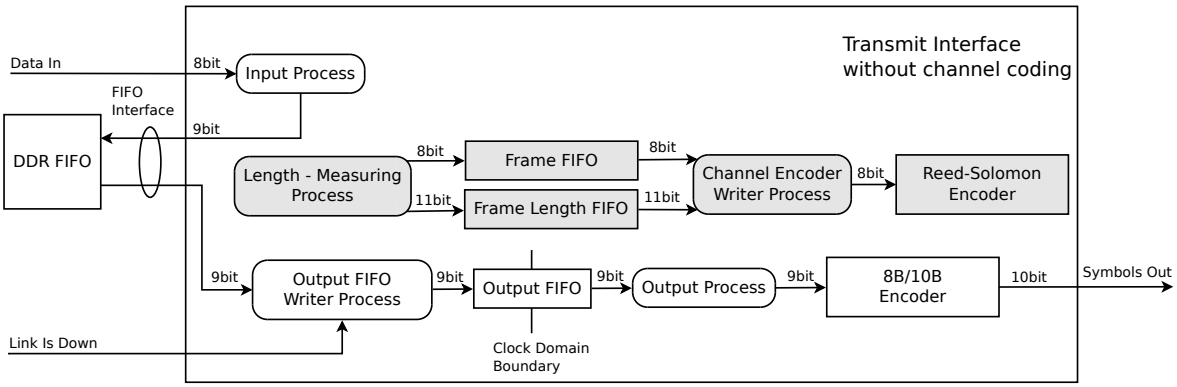

Fig. 5.6: Transmit unit of the MAC, channel coding off

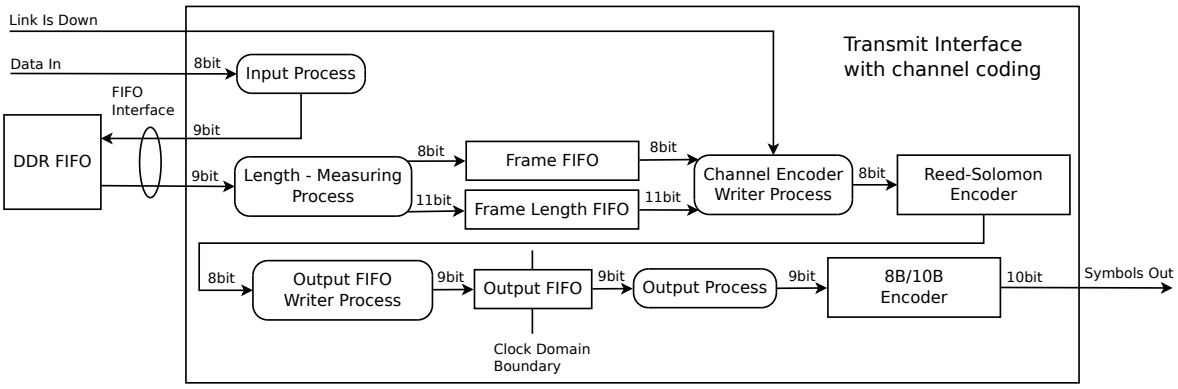

Fig. 5.7: Transmit unit of the MAC, channel coding on

With the channel coding enabled, the situation is more complicated. Reed-Solomon coding is a block coding, so the length of the input frames is first measured and then the frames and their lengths are written to the input of the Reed-Solomon encoder. This is depicted in Figure 5.8. Another process is observing the Data Valid encoder output signal showing that an encoded Reed-Solomon block is appearing at the output. When there is a pause greater than three clock cycles at the output of the encoder, an EOF symbol is written to the input of the FIFO realizing clock domain crossing mentioned in the previous paragraph. This way a super-frame is created sending several encoded frames in a single burst of encoded blocks. This is described in Figure 5.9.

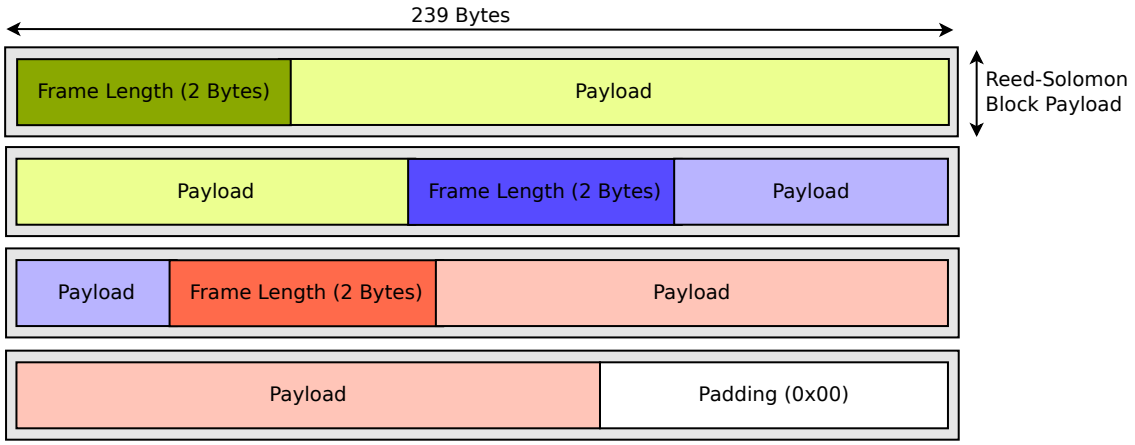

Fig. 5.8: Data input to the channel coder, three frames

The FIFO realizing clock domain crossing is kept as small as possible, since when the *LINK_IS_DOWN* signal goes high, the transmission stops, but the content of this FIFO is "flushed out" to the 8B/10B encoder and transmitted. Indeed, there

is no use in keeping a part of the frame untransmitted, when the other part of it has been either lost in the unavailable channel. When the *LINK_IS_DOWN* is too sensitive, the flushed frame can be corrected by the channel coding (in our case, when there is less than 8 erroneous symbols for 255 symbols transmitted).

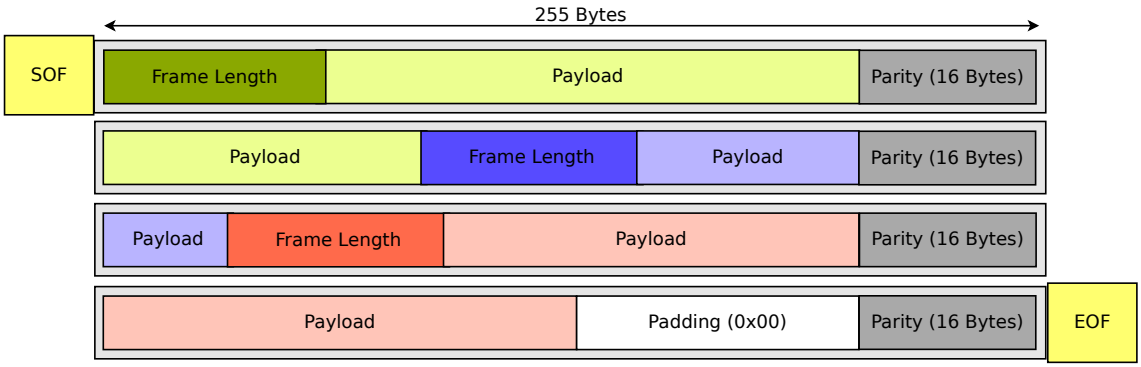

Fig. 5.9: Super-frame, output of the channel encoder with the SOF and the EOF

The selected Reed-Solomon channel coding has an overhead of 16 bytes per each 255 bytes long coding block. There is an additional overhead due to the padding, which needs to be inserted after the last frame as shown in Figure 5.8. Indeed, the padding formed of bytes with value 0x00 is inserted only when the length limit of a super-frame is reached (currently set to 2 kB) or when there is no more Ethernet frame to be forwarded through the FSO link. In the latter case, the overhead is not a problem, since there is no payload to send. In the first case, some overhead is intentionally added due to the length limit of a super-frame. This was introduced to make the Start-Of-Frame (SOF) and End-Of-Frame (EOF) be transmitted at least once in 2 kB of payload. This enables the receiver side of the link to renew the frame synchronization when it was lost after a link outage.

When no channel coding is used, the BER reduction technique described in previous chapters (specifically in 3.5.2) is still active, but realized by the Output FIFO Writer Process (Figure 5.6). When the channel coding is active, the technique is realized in the Channel Encoder Writer Process (Figure 5.7), since there is no advantage in stopping the transmission when the Output FIFO Writer Process has a portion of encoded block to be read from the Reed-Solomon Encoder.

For completeness, Figure 5.10 shows how the data are presented to the 8B/10B Encoder input in case there is no channel coding enabled. The advantage of disabling the channel coding resides in higher throughput and a more frequent apparition of SOF and EOF symbols in the channel, which enables a faster renewal of frame synchronization when it is lost.

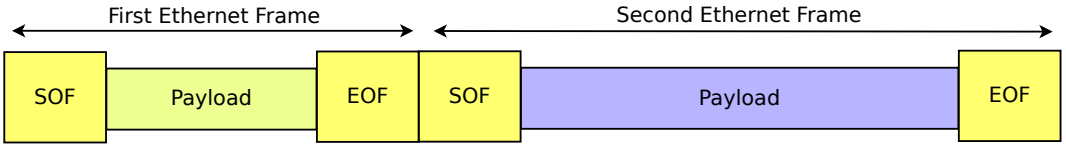

Fig. 5.10: Frame with SOF and EOF, case with no channel coding

All unencoded or encoded frames or super-frames are input at the end of the transmit chain to the 8B/10B Encoder, which maps each 9 bit value (8 bit data and 1 bit K character signal) to a 10 bit symbol. This is required especially by the CDR circuit in the GTP transceiver to be able to recover the clock signal from the received data. Another advantage of higher density of edges in the signal is a less rich spectrum, which facilitates the printed circuit board design. Indeed, from the SFP module to the FPGA chip, the signal is sent by a differential backplane copper pair. 8B/10B Encoder can also detect a bad symbol, since not all 10 bit combinations are used - if an invalid combination is received, it is detected. This is also used by the receiver to detect the link going unavailable.

5.2.2 Reception Path

Figure 5.11 shows how the receive interface operates. First, the Data Valid (DV_RX) signal is driven high simultaneously with the first valid byte of the frame being output at the Data Out ($DOUT\_RX$) port. When DV_RX goes low, the last byte of the frame is received. In the next clock cycle, either Good Frame (GF_RX) or Bad Frame (BF_RX) signal goes high for one clock cycle to show, whether a valid or invalid frame was just received. This way the Flow Controller which interfaces with this MAC module can either drop or forward the received frame.

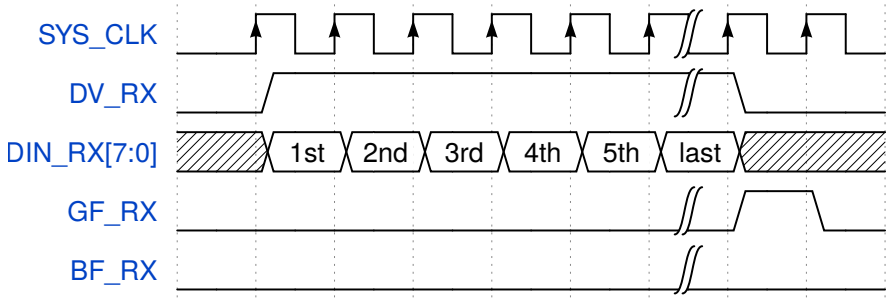

Fig. 5.11: Receive interface, TEMAC compatible

For data to be available at the Data Out port, it has to go through all the receiver module logic. First, a raw 10 bit symbol is received at the Symbols In

port (Figure 5.12). The symbol is then forwarded to the 8B/10B Decoder only if the receiver is aligned to symbols' boundary. This alignment is done by receiving comma symbols, which are reserved 10 bit symbols for this purpose. The Symbol Boundary Alignment Process keeps a history of two 10 bit symbols and looks for the comma sequence in it. When the comma is found, the bit offset from the start of the first received 10 bit symbol is saved. Next four symbols have to be correctly recognized as comma symbols for the Symbol Boundary Alignment Process to put the Data Valid signal high. The process is active only when a frame or a super-frame is not being received (not in between SOF and EOF symbols). Sending the comma symbols even if the transmitter has no data frames to send is required for this technique to work. For recovery from outage, there is also a minimum number of comma symbols (five) inserted between each EOF and SOF.

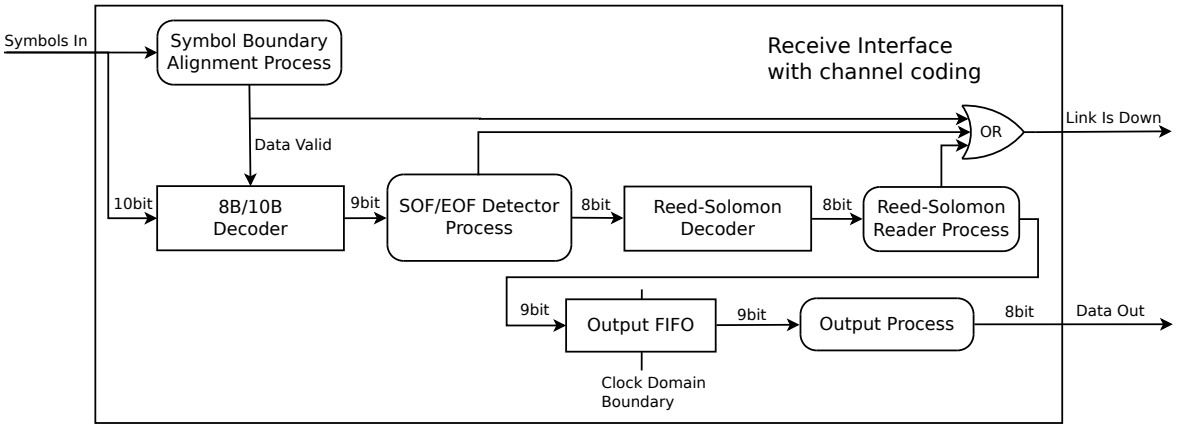

Fig. 5.12: Receive unit of the MAC, channel coding on

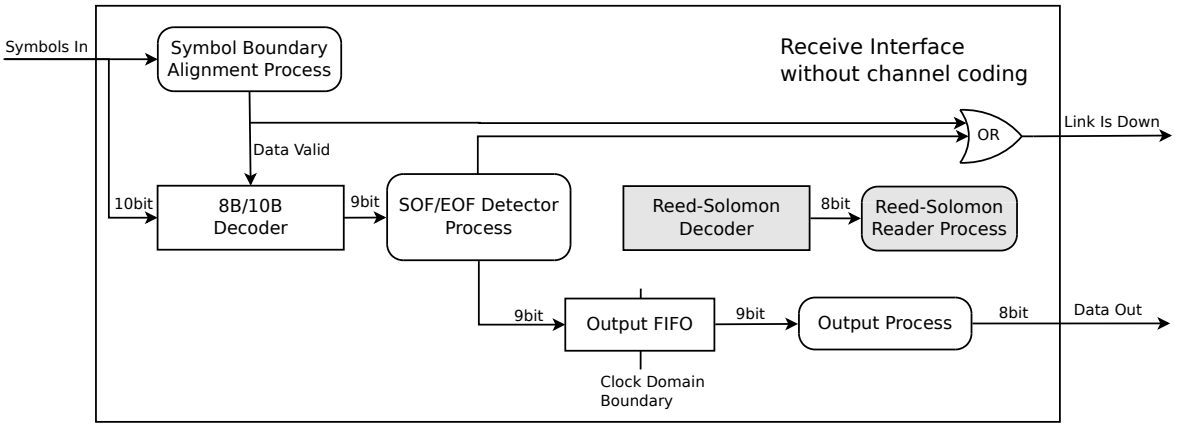

Fig. 5.13: Receive unit of the MAC, channel coding off

The rest of the receiver is different with channel coding enabled or disabled. When the channel coding is enabled (Figure 5.12), the output of the 8B/10B Decoder is read by the SOF/EOF Detector Process which generates a synchronization signal for the Reed-Solomon Decoder. Thanks to this, the Reed-Solomon Decoder is block aligned. The output of the decoder is read by the Reed-Solomon Reader Process which parses the length field of each frame contained in received super-frame. The length field is 16 bit long (Figure 5.14): 4 bits are used for cyclic redundancy check of the length and only 11 bits are used for the actual length value. One bit is used to differentiate between the first byte of padding which has value of 0x00 and the first byte (LSB) of the length field. When an invalid length field is found, Link Is Down is asserted high and the rest of the super-frame is dropped.

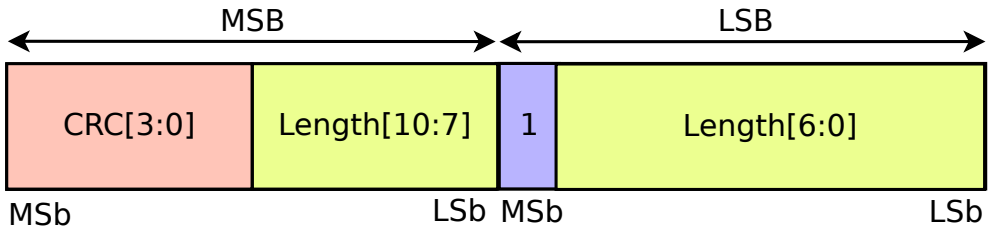

Fig. 5.14: Length field, preceding each frame in a super-frame

The Output FIFO is written to by the Reed-Solomon Reader Process in case the channel coding is enabled. If it is not, the SOF/EOF Detector Process writes to this FIFO directly bypassing the Reed-Solomon Decoder and length field parsing (Figure 5.13). The Output FIFO has another important role - it is used for different clock domain crossing. Up to this component, the CDR-recovered clock is used and from Output FIFO read interface on, the system clock provided as part of the user interface of the receiver is used (SYS_CLK in Figure 5.11).

The Link Is Down signal is asserted high when the Symbol Boundary Alignment Process has not yet finished the bit alignment, when the SOF/EOF Detector Process detects an invalid output from the 8B/10B Decoder (invalid 10 bit symbol received) or when the Reed-Solomon Reader Process detects an unrecoverable number of errors in the received encoded block. The last signal is not available, when the channel coding is not enabled. This signal is consequently used by the transmitter to enable or disable the transmission of its own data - when the link is down, there is no use in transmitting data. The transmitter should send comma symbols until the Link Is Down cool-down counter reaches zero.

5.2.3 Optical Path Statistics Signals

The Simple SFP MAC Module provides statistic signals which can be used when evaluating the properties of a given real configuration of the link. Two different kind of statistic output is available - counters of events and FIFO-based output of time stamped link status transitions. Following table sums up the event counters:

- RX_FRAMES_RECEIVED[63:0]:
  This counter gives the total count of frames received by the user at the client receive interface of the Simple SFP MAC Module.
- RX_FRAMES_BAD[63:0]:
  This counter gives the number of corrupted frames received by the user of this module.
- RX_BYTES_CORRECTED[63:0]:
  This counter gives the number of bytes recovered successfully thanks to the channel coding.
- RX_BYTES_RECEIVED[63:0]:
  This counter gives the total number of bytes received by the receiver.
- TX_FRAMES_SENT[63:0]:
  This counter gives the number of frames sent into the transmit interface of this module
- TX_FRAMES_TRUNCATED[63:0]:
  This counter gives the number of frames, which had to be truncated (prematurely terminated). This happens at the transmit interface, when the DDR FIFO becomes full in the middle of a frame. The frame is marked as bad and dropped consequently.
- TX_BYTES_SENT[63:0]:
  This counter gives the number of bytes sent out to the transmit interface.
- TX_BYTES_DROPPED[63:0]:
  This counter gives the number of bytes, which were dropped because the DDR FIFO was already full and the link was still down.

Apart of this, the module has also a FIFO output, which records upto 512 link event frames. Each time a transition occurs, a new value is saved in a FIFO in the format described in Figure 5.15. The MSB is the new value of the *LINK_IS_DOWN* signal, followed by bits signalling a framing error, that an unrecoverable Reed-Solomon block was received, that there is an invalid symbol at the input of the 8B10B Decoder, an invalid bit alignment and finally the lower 59 bits are the value of a 125 MHz counter at the moment of the signal transition. The counter is reset when the MAC module is reset and can run without overflowing for approximately 150 years, so no handling of the overflow is really needed. The *STAT_COUNT[9:0]*

signal gives the total count of transition events saved in the internal FIFO. The *STAT_DOUT* signal is the description of the first event in the queue. For fetching the next event, user needs to put high the *STAT_RD_EN* signal, just like reading a regular first-word fall-through FIFO. Thanks to this statistical output, the user can monitor the frequency of the outages, their duration and its correlation with other parameters of the system (i.e. receiving power, humidity, temperature, time of the day, etc.). The output is to be read by the on-chip Microblaze-based microprocessor system and presented via Ethernet interface to the user of the device.

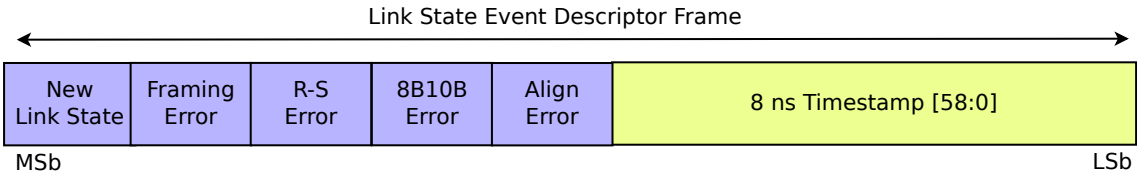

Fig. 5.15: *LINK_IS_DOWN* event frame format

5.2.4 Channel Coding Control

As described in the previous sections, the channel coding used by this module can be optionally turned off. When turning off the Reed-Solomon channel coding, the user of the device can evaluate the impact of this action on the frame error rate. The error rate can be extracted from the statistics output of the Simple SFP MAC Module or externally, i.e. by counting the number of successfully received UDP packets versus the total number of sent packets.

The enabling or disabling of the channel coding is done by the on-chip microprocessor, which has to go through steps described in Figure 5.16.

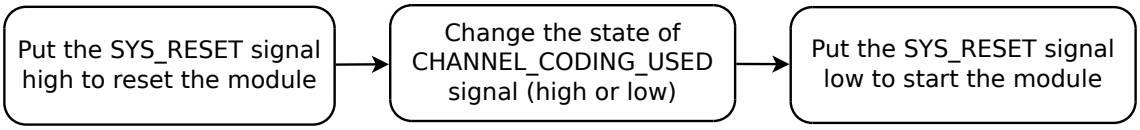

Fig. 5.16: Channel coding on/off sequence

5.3 DDR-based FIFO

In Figure 5.7 and 5.6, a DDR FIFO was used. This component was created to store the output data frames when the channel is not available [33]. Indeed, if the drop-out duration to be covered is up to 1 s and the rate of the Ethernet is 1 Gbps, then a FIFO of at least 125 MB is needed. This cannot be realized with on-chip resources of any commercially available FPGA.

5.3.1 Principle of Operation

The implementation itself uses the memory interface generator [29] (MIG) IP core from Xilinx and interfaces this with DDR Reader and Writer. This module uses the half-duplex communication interface with DDR2 SODIMM module and implements a circular buffer in it. Thanks to the bus width of 128 bits, the module has enough time to keep the Input FIFO empty and the Output FIFO full during non-interrupted data transmissions. 128 bit wide communication bus was selected, since it is the maximum bus width supported by the ML505 Evaluation Platform used in testing of the design and it gives enough time margin for the processing of input and output FIFOs.

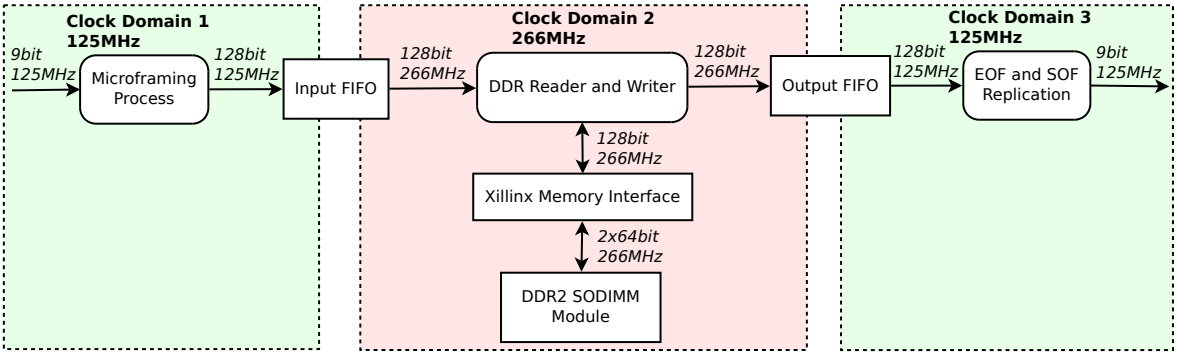

Fig. 5.17: DDR FIFO internal data flow

Data frames are stored in the DDR FIFO in a microframe memory scheme. The scheme was developed to separate individual data frames. It enables the output module to replicate the Start-of-Frame (SOF) flag at the beginning and the End-of-Frame (EOF) flag at the end of each data frame. The scheme uses 128 bit data units called microframes. The first fifteen bytes of microframe contain data payload and the last byte is a control character. It indicates whether the SOF or EOF flags should be generated and the size of data payload in each microframe. The indication of the data size in the last microframe is crucial as the incoming frames are not 15 B

aligned and 15 B alignment is needed for the microframe memory scheme (Figure 5.18).

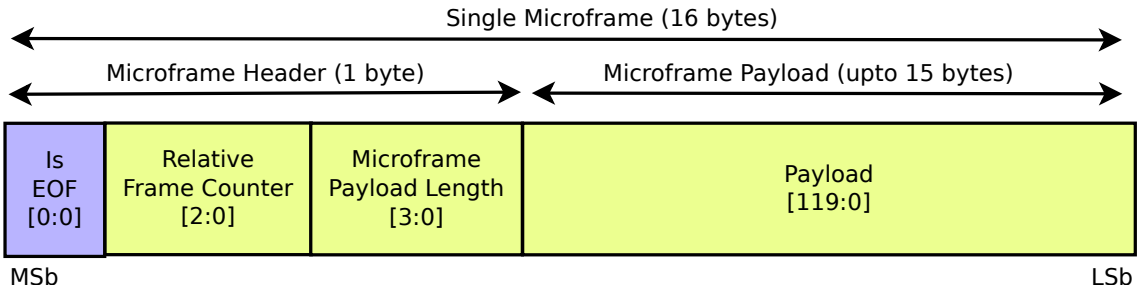

Fig. 5.18: Microframe structure

5.3.2 DDR FIFO Interface

For an ease of use and to allow a third party design to benefit also from this component, the interface of the DDR FIFO is kept as close to the interface of a regular FIFO as possible. The DDR FIFO replaced a former regular FIFO, which was present in the design. Those who want to implement the design, but do not have a DDR available in the system, they can simply replace the DDR FIFO by a regular BRAM-based FIFO. The mechanism will still work, but only very short outages will be covered. Due to the microframing and SOF/EOF reconstruction, the DDR FIFO cannot be used in a completely generic way. Nevertheless anywhere, where a storage of data frames is needed, this DDR FIFO can be used. The interface is shown in Listing 1. The interface towards the DDR controller generated by the Memory Interface Generator (MIG) from Xillinx is not shown.

5.4 Reed-Solomon IP Block

In order to implement the channel coding, Reed-Solomon IP cores generated in Xilinx ISE were used. In the Core Generator, the user can select a predefined code specification, which defines the field polynomial to use, number of symbols per block and number of data symbols per block. Another option is to define these parameters manually. In this project, the first option was used and the specification per ITU-T recommendation G.709 was used. This recommendation defines the code as shown in Table 5.1. As described in previous chapters, the G.709 specified RS code can correct up to 16 erroneous symbols.

As part of further experimenting, the size of the block and the number of data symbols per block can be changed to see, whether a lower coding rate or smaller

```
ENTITY DDRfifo IS
GENERIC (
    APPDATA_WIDTH : INTEGER := 128;
    -- # of user read/write data bus bits.
    FIFO_SIZE_POW2 : INTEGER := 25
    -- length of the FIFO is 2^25 bytes.
);
PORT (
    -- FIFO compatible interface, parsing SOF and EOF
    rst : IN STD_LOGIC := '0';
    wr_clk : IN STD_LOGIC := '0';
    rd_clk : IN STD_LOGIC := '0';
    din : IN STD_LOGIC_VECTOR(8 DOWNTO 0) := (OTHERS => '0');
    wr_en : IN STD_LOGIC := '0';
    rd_en : IN STD_LOGIC := '0';
    dout : OUT STD_LOGIC_VECTOR(8 DOWNTO 0) := (OTHERS => '0');
    full : OUT STD_LOGIC := '1';
    empty : OUT STD_LOGIC := '1';
    empty_prog : OUT STD_LOGIC := '1';

    -- DDR application interface
    ...
);
END DDRfifo;
```

Listing 1: DDR FIFO Interface

block can help to reduce better the BER. For this thesis, the G.709 specification was chosen, so number of data symbols per block or any other parameter cannot be manually changed.

5.4.1 Encoder IP Core

Figure 5.19 show the interface of the generated encoder module. The Core Generator enables the user to select to generate or not some optional interface signals. Only signals, which were selected to be used are shown.

The encoder input process is the following: First, the user waits for *rffd* signal to go high, which signals that the module is ready to accept a new pulse at the start signal input. Second, a pulse is generated at the start input and simultaneously first

Parameter	Value
Field Polynomial	285 (or $x^8 + x^4 + x^3 + x^2 + 1$)
Number of Symbols per Block	255
Number of Data Symbols per Block	239
Symbol Width [bits]	8

Tab. 5.1: Specification of Reed-Solomon code per G.709 recommendation

valid input data is presented at *data_in* input. All signals are sampled on the rising edge of the *clk* signal. The user can input other input symbols on next rising edges of the *clk* signal until the *rfd* signal goes low. Bypass signal is used to temporarily interconnect *data_in* and *data_out*, which makes the data at *data_in* port not to influence the resulting parity bytes. For reseting the core, the active high reset at *sclr* is introduced.

The output interface is extremely simple: When the *rdy* signal goes high, a first encoded symbol is present at the output of the encoder at *data_out* port. The *rdy* remains high as long as there is valid encoded data at *data_out*. Info output signal is not actively used in the design and it shows, when there are informational (payload) symbols at *data_out*. This information is not important, since all encoded symbols are output to the channel, including the non-informational ones. The latency of the encoder is 3 clock cycles and depends on the selected specification during IP core generation. [30]

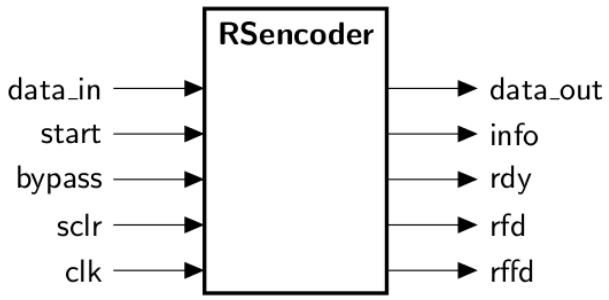

Fig. 5.19: Reed-Solomon Encoder Interface

5.4.2 Decoder IP Core

Figure 5.20 shows the interface of the decoder which resembles the encoder interface, but has some important status output ports.

Before starting to write a new received encoded frame to *data_in*, the user has to wait for the *rffd* signal to go high. After that, a pulse is generated at the *sync* signal and first symbol of the encoded block is input at *data_in*. The rest of the block is input on following rising edges of the *clk* signal, as long as the ready signal is high. In the design, the *mark_in* input is also used. This input is shifted through the decoder module synchronously with the decoded data, so it is used to mark a start of a new decoded block at the output. After the processing, the *mark_in* appears at *mark_out* synchronously with the first decoded symbol appearing at *data_out* port.

The output interface of this module is more complex. The start of a new decoded block is signalled by a pulse at *blk_strt* signal. Pulse at *blk_end* is generated with the last valid byte of the decoded block being output and a pulse is generated at *info_end* when the last informational decoded symbol appears at *data_out*. When fail signal is high, the decoder failed to correct the block and it should be considered as invalid. When fail is low, the number of corrected symbols appears at *err_cnt* and when *err_cnt* is not zero, the *err_found* signal is high. The processing delay of the decoder is 204 clock cycles and the latency is 466 clock cycles. When the processing delay is smaller than the latency, the output of the decoder is not interrupted when two or more consequent blocks are received. [31]

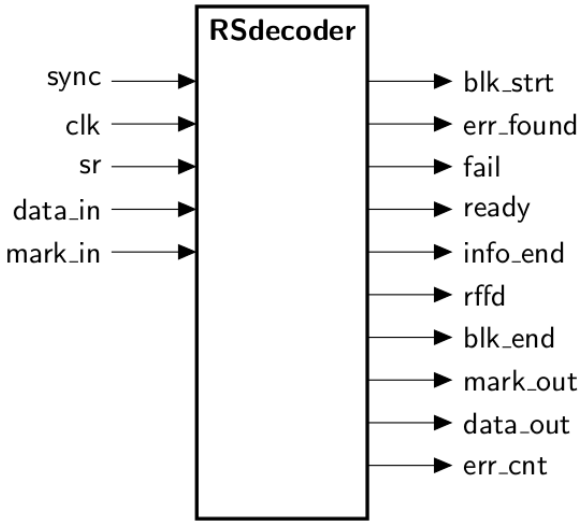

Fig. 5.20: Reed-Solomon Decoder Interface

6 SIMULATION OF THE SYSTEM

The system was simulated in two steps: First in ISIM simulation software from Xilinx for both evaluating the performance of the system and debugging it. Second, the performance of the system was tested by creating a physical loopback interconnecting the output of the SFP module with its input using an optical fiber. There are multiple reasons to simulate the behavior of the design not only in ISIM, but also in hardware - the simulation of 25 ms of run takes approximately eight hours in ISIM at 2.2 GHz quadcore Intel Core i7 processor with enough RAM available. In real implementation, a minute of simulation takes precisely one minute, not more. The evaluation of the system synthesizing it into the FPGA is not however suitable for debugging. There are tools like Chipscope, which partly allow it, but this approach is extremely time-consuming. Nevertheless when there is a bad timing or erroneously interconnected clock domains, the error is sometimes simpler to study in Chipscope. Therefore both approaches are required.

6.1 Simulation in ISIM

In order to debug the implementation of the Flow Controller and the Simple SFP MAC Module, a test case was created. This test case (*SimpleMACandFlowController.vhd*) instantiates the Flow Controller as well as the MAC module and connects the output of the transmitter to the input of the receiver creating thus a loopback. By generating randomly long frames and sending them to the transmitter after a randomly selected delay, all corner cases of the design could be debugged and they are now correctly handled.

Figure 6.1 shows, how the Simple SFP MAC Module, the Flow Controller and the DDR FIFO components are instantiated and interconnected. The stimulus is done via the receive Ethernet interface of the Flow Controller and the frame structure is monitored at the transmit Ethernet interface of the same module. The BER is simulated by an embedded random number generator and a requested compare value, which is set by the stimulus logic. (The BER simulation will be described later on in this chapter) The statistics showing how many frames were lost or dropped, how many frames were actually sent and how many bytes were corrected by the channel coding can be observed by the validation logic.

The stimulus is done by the stim_proc process and the stimulation loop is shown in Listing 2. The validation is not more complicated - the frame bytes' values are checked to start from 0x00 and to increment with each next byte. This way, when an erroneous frame is forwarded by the Flow Controller, it is detected and a debug message is printed to the console of ISIM simulator. The validation code is in

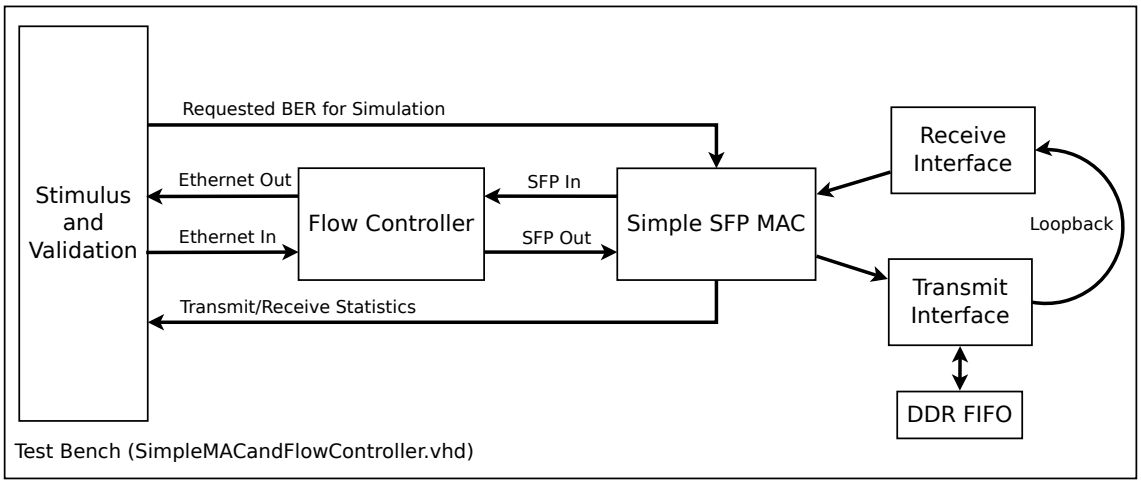

Fig. 6.1: Test Bench Block Diagram

Listing 3. Even though the validation and testing processes are quite simple, they are sufficient to test any fault in the design. Further asserts are included in the sythesizable VHDL code, so some critical logic constraints are self checking - if a bug is introduced in a future modification, it will be discovered just by running the SimpleMACandFlowController.vhd test bench.

For statistical study, the requested BER can be set by the BER_SIMULATION_COMP 32 bit signal and the resulting BER is given by Formula 6.1. The values of the counters can be sampled after a certain time (or after a certain number of sent frames) and a real BER can be calculated or the statistics can be output by the std.textio VHDL library for further processing.

$$BER = \frac{BER\_SIMULATION\_COMP}{4294967295} \tag{6.1}$$

Where BER is the resulting bit error rate and BER_SIMULATION_COMP is the value of the 32-bit comparation register, which can be set externally.

6.2 Simulation in FPGA

In order to simulate low receiving power, an attenuator was introduced in the physical fiber loopback optical path (Figure 6.8). This however does not simulate real channel, where phenomena like beam wander appear. To simulate these phenomena from a statistical point of view, a special on-chip module was developed, described in following subsection.

```
-- Send 100000 frames
for MASTERLOOP in 0 to 100000 loop
        -- Read random length from
        -- the random number generator
        this_len <= rand_len;
        -- Send the Frame
        ETH_DV_IN <= '1';
        ETH_DATA_IN <= X"00";
        wait for TCYC_SYS_0;
        for I in 0 to this_len loop
            ETH_DATA_IN <= ETH_DATA_IN + X"01";
            wait for TCYC_SYS_0;
        end loop;
        ETH_DV_IN <= '0';
        wait for TCYC_SYS_0;
        ETH_GF_IN <= '1';
        wait for TCYC_SYS_0;
        ETH_GF_IN <= '0';
        -- Insert a randomly long pause
        wait for rand_num*TCYC_SYS_0;
end loop;
-- Simulation is done
assert false report "Simulation Finished" severity failure;
```

Listing 2: Stimulus Loop

6.2.1 Embedded Channel Simulator

The Simple SFP MAC module contains an embedded channel simulator, which is based on a random number generator. When looking for a good synthesizable random number generator, the Linear Feedback Shift Register (LFSR) and Cellular Automaton Shift Register (CASR) architectures were considered. By visualizing the outputs of each of these, a partly deterministic pattern can be observed. Therefore the outputs of this two architectures were combined by a non-linear function (in our case XOR) to created a less deterministic pattern (Figure 6.2). [24]

Figure 6.3 shows the block diagram of the embedded channel simulator. After generating a random number, as described in previous paragraph, this number is compared to a modifiable value. When the result of the comparison is true, a bit is inverted in the symbol, which is being received. As the random number is 32 bit,

```
-- Validation process
process(SYS_CLK)
begin
    IF rising_edge(SYS_CLK) THEN
        IF ETH_DV_OUT = '1' THEN
            CHECK_CNTR <= CHECK_CNTR + "1";
            IF CHECK_CNTR /= ETH_DATA_OUT THEN
                assert false report "Error in data" severity warning;
            END IF;
        ELSE
            CHECK_CNTR <= X"00";
        END IF;
    END IF;
end process;
```

Listing 3: Validation Loop

BER can be for example as low as 10^{-9} when setting the compare signal to 4(Formula 6.1). Values lower than 10^{-10} cannot be achieved by this simulator, the bit width of the random number would have to be increased. This is however not required for now.

6.2.2 Access to Results

In order to access the results of the tests, the Ethernet interface was used. The Flow Controller module allows the on-chip Microblaze to communicate via the Ethernet device, so a web server and a UDP transmitter was implemented in the Microblaze. [9]

The embedded UDP transmitter transmits to IP address 255.255.255.255 (broadcast) on UDP port 10000 any new event frame appearing in the event frame FIFO. A simple Python script was developed with a GUI visualizing long-term transitions of the signals showing outages of the channel. The data are simultaneously stored in a CSV file and can be loaded and plotted with the same script. The GUI is shown in Figure 6.4. The script is available in the */Scripts* directory of the GIT repository.

The embedded web server pages can be simply loaded with any web browser by typing the server's IP address in the address bar. The IP address can be either manually hard-coded in the Microblaze firmware or DHCP can be enabled. The web server shows the values of the statistical counters described in previous sections.

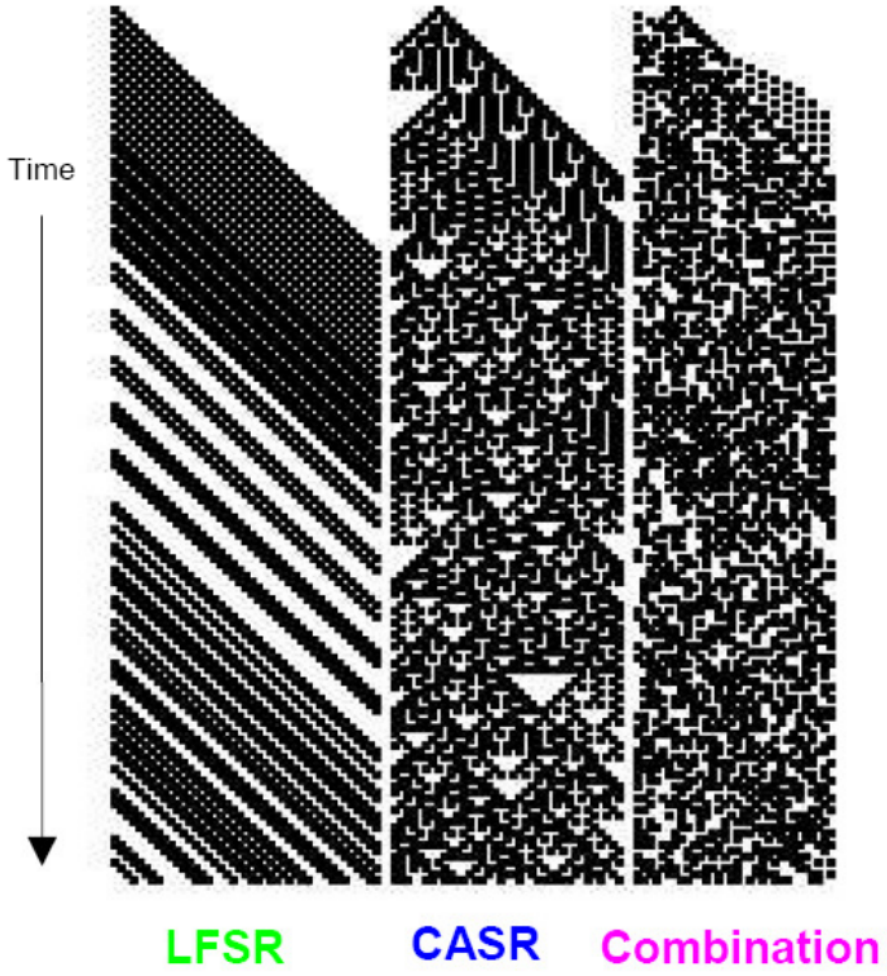

Fig. 6.2: Comparison of LFSR, CASR an its XORed combination [24]

Furthermore, it can be used to manually turn on or off the channel coding or to set the desired simulated BER. Figure 6.5 and 6.6 shows the server's web page.

The last method to evaluate the number of erroneous frames or to measure the network bandwidth is the Wireshark tool and the simpleFlood.py (available in */Scripts* directory). In the Python script, a UDP broadcast is sent with a given payload and in Wireshark, the traffic can be evaluated by filtering out all UDP frames and observing apparition of malformed frames. Another way, how to use Wireshark to evaluate the system is to open the Statistics/IO Graph window and to observe the evolution of the bandwidth over time. (Figure 6.7)

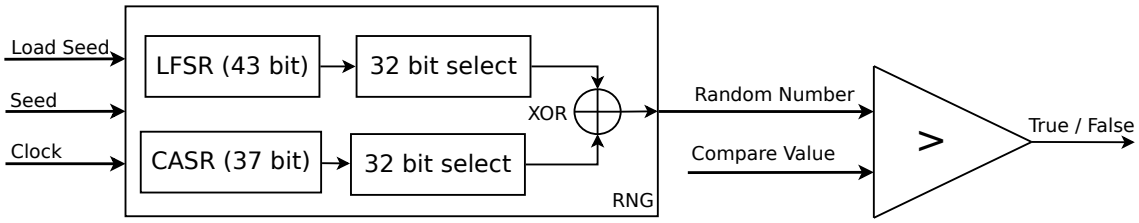

Fig. 6.3: Channel Simulator Principle

6.3 BER Measurements

This section describes measurements verifying the capabilities of implemented platform to measure the BER. It furthermore shows the coding gain attained by the Reed-Solomon channel coding.

6.3.1 Experimental Setup

The setup consists of two workstations running appropriate Python scripts. Figure 6.8 describes the situation - the sender runs script *SimpleFlood.py* (4) and the receiver runs script *EventFrameCatcher.py* (6.4). On the receiver side, the BER can be calculated using data from the statistics page in the web GUI (Figure 6.5). In the settings page, the BER to be generated at the input of the receiver can be selected, the channel coding can be enabled or disabled and the broadcast of event frames can be also disabled or enabled. Table 6.1 shows results of BER measurements.

6.4 Results

As can be observed in Table 6.1, the Reed-Solomon coding performs better when the size of the frame is 1182 B (line 1 vs. line 2 compared to line 3 vs. line 4). It can be explained by the fact that each encoded frame has the size which is a multiple of 255 B. Hence in case the size of the frame is smaller (100 B), the Reed-Solomon coder does not reduce the resulting FER, but increases it, since each such frame has to be padded to attain the required alignment (6.2). During the measurements, the transmitted power was kept constant and only the settings of the simulated BER was changing - the resulting FER (Frame Error Rate) shows the actual coding gain of the RS coder. As conclusion the coding gain of a block coder such as RS(255,239) highly depends on the implementation, the framing technique and the actual length of transmitted frames during the test.

```
import socket
UDP_IP = "192.168.2.13"
UDP_PORT = 5005
MESSAGE = "Change this to change the length of the test frames"
print("UDP target IP:", UDP_IP)
print("UDP target port:", UDP_PORT)
print("Message:", MESSAGE)
print("Length: %d" % len(MESSAGE))

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
#We can broadcast, if broadcast addressed used
sock.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
#Keep sending forever
while True:
    sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
```

Listing 4: Script SimpleFlood.py, used for traffic generation

$$\delta FER = \frac{FER_{RS\ Enabled}}{FER_{RS\ Disabled}} - 1 = \frac{2.9 \cdot 10^{-6}}{5.9 \cdot 10^{-6}} - 1 = -0.51 \tag{6.2}$$

Where δFER is a relative change in the frame error rate, $FER_{RS\ Enabled}$ is the FER measured with enabled Reed-Solomon coding and $FER_{RS\ Disabled}$ is the FER measured with RS coding disabled with all other parameters (transmit power, frame length) set to the same value as in case of $FER_{RS\ Enabled}$.

BER_{set}	$FrameLength$	$\#Frames$	$\#Bad$	FER_{meas}	**Coding**
10^{-8}	1182	218575335	634	$2.9 \cdot 10^{-6}$	8B10B+RS(255,239)
10^{-8}	1182	370548116	2184	$5.9 \cdot 10^{-6}$	8B10B
10^{-6}	100	15214736	1008	$6.6 \cdot 10^{-5}$	8B10B+RS(255,239)
10^{-6}	100	16770024	1026	$6.1 \cdot 10^{-5}$	8B10B

Tab. 6.1: Example Results - BER Measurements

When observing the measured FER, the experiments show that it is bound to the "Set BER" (BER which is introduced by the embedded channel simulator) by Formula 6.3. In case the Reed-Solomon encoding is enabled, the actual FER is influenced by the implementation of the fragmentation and framing strategy. Formula

6.4 illustrates this negative influence - any frame being sent must be padded up to the closest greater multiple of 239 B. To mitigate this influence, multiple Ethernet frames can be stored in a single RS block, which is why the length field was introduced. This approach has however an inconvenience - it introduces the constant LF in Formula 6.4. Generally, current implementation enabling the combination of multiple small Ethernet frames (< 239 B, due to RS(255,239) coding) into a single RS block is increasing the FER at the cost of reducing the overhead of the RS coder during the transmission for small sized frames.

$$FER_{RS\,off} \approx BER_{set} \cdot FrameLength \tag{6.3}$$

Where $FER_{RS\,off}$ is the frame error rate in case the RS coder is disabled, BER_{set} is the generated BER and FrameLength is the length of testing Ethernet frames in bytes.

$$FER_{RS\,on} \approx BER_{set} \cdot (ceil(\frac{FrameLength + LF}{PayloadLength}) \cdot BlockLength) \cdot CodingGain \tag{6.4}$$

Where $FER_{RS\,on}$ is the frame error rate in case the RS coder is enabled, BER_{set} is the generated BER, FrameLength is the length of testing Ethernet frames in bytes, LF is the length of the Length Field which equals to 2, PayloadLength is the length of the payload portion of a RS block, BlockLength is the total length of a RS block. In case of RS(255,239) PayloadLength equals to 239 and BlockLength equals to 255. CodingGain (ranging from 0 to 1) represents the influence of the coding gain of the RS coder.

In Figure 6.9, the measurements of real FER is shown. The figure illustrates above mentioned formulas and shows, how the RS coding gain depends on the length of the transported frames. Figure 6.10 provides a more readable plot of the same information, processed by equation 6.2.

In order to evaluate the performance of the link regarding the attainable bit rate, Iperf benchmarking tool was used to perform a series of measurements. Figure 6.11 shows how the link performs, when the Reed-Solomon coder is enabled and when it is disabled. Figure 6.12 shows an evolution of the difference between the bit rate when the coding is enabled and when it is disabled. The saturation for

smaller values of BER ($< 10^{-8}$) is caused by the limited performance of Ubuntu workstations, which were used during the test. It was not possible to produce more than 805 Mbps for UDP and 875 Mbps for TCP. The workstations were two laptops with Intel i7, 8 GB of physical memory, SSD and Ubuntu 14.04LTS installed. Iperf used TCP and UDP for the tests and at both receive sides of the link the level of the simulated BER was set to the same value (X axis). The TCP is influenced also by the erroneous backward channel - when the acknowledge arrives corrupted, it is lost and the effective throughput decreases. UDP is not influenced by the backward channel.

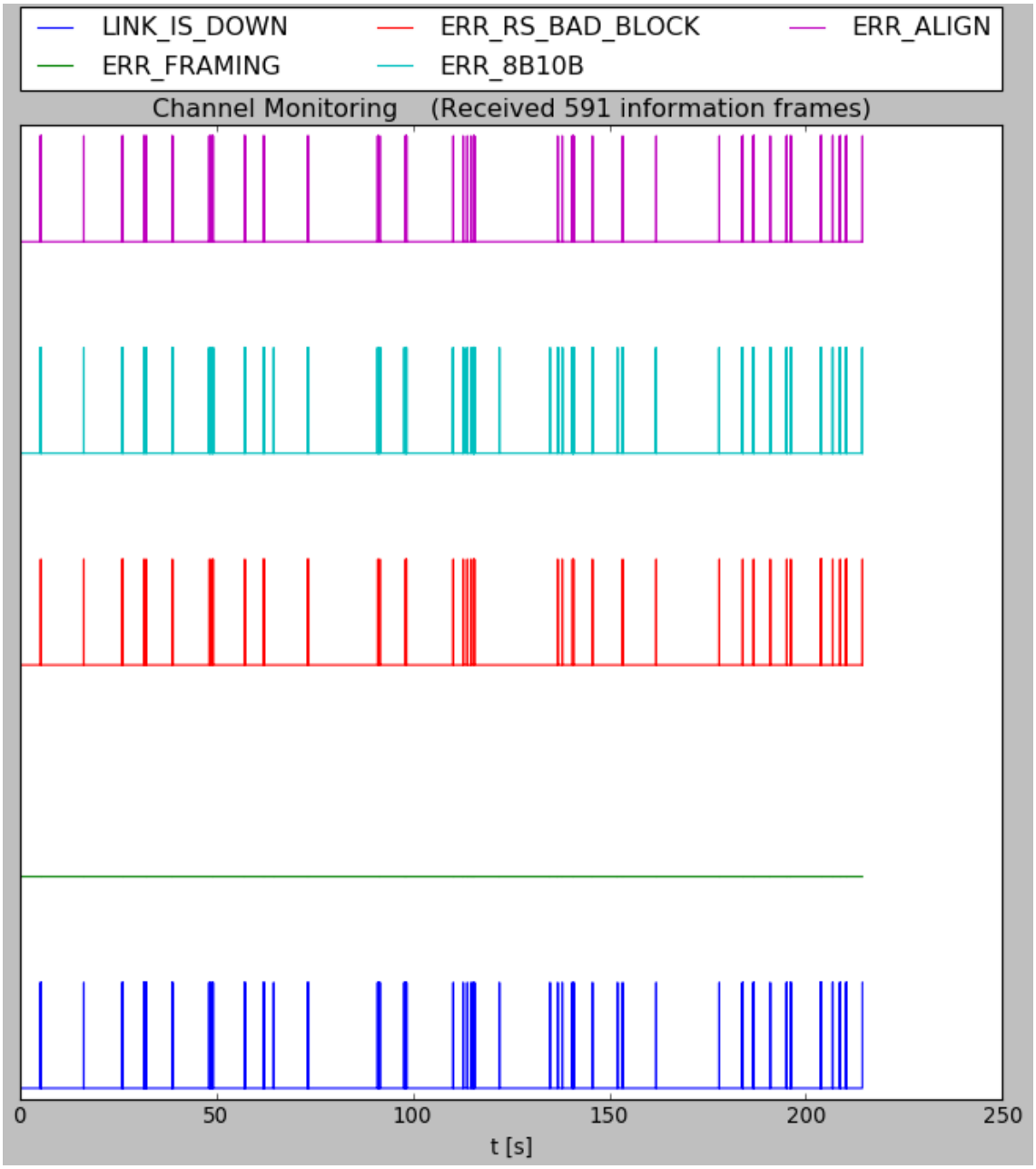

Fig. 6.4: Event Frame Catcher Python Script GUI

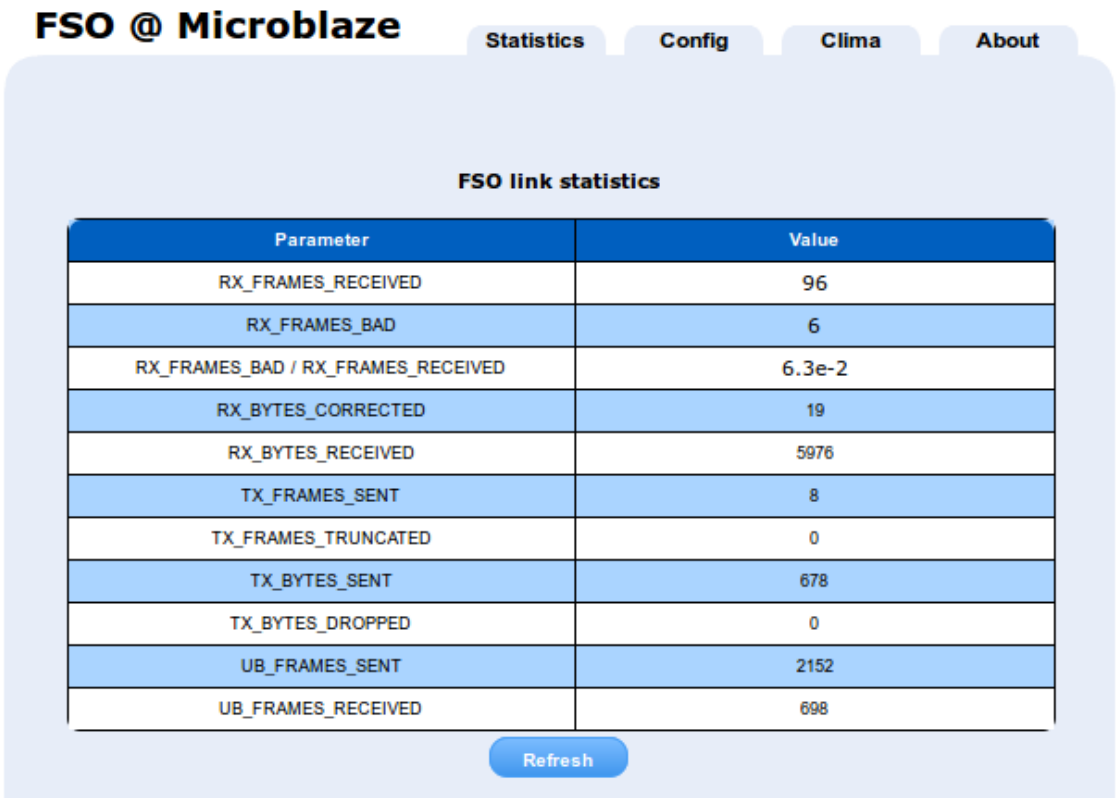

Fig. 6.5: Web Server Interface - Statistics

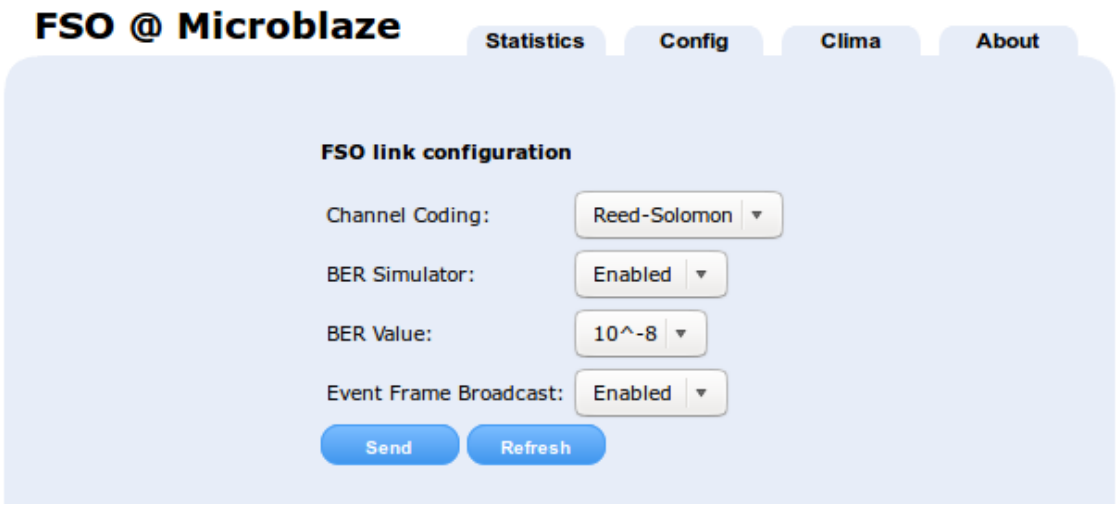

Fig. 6.6: Web Server Interface - Settings

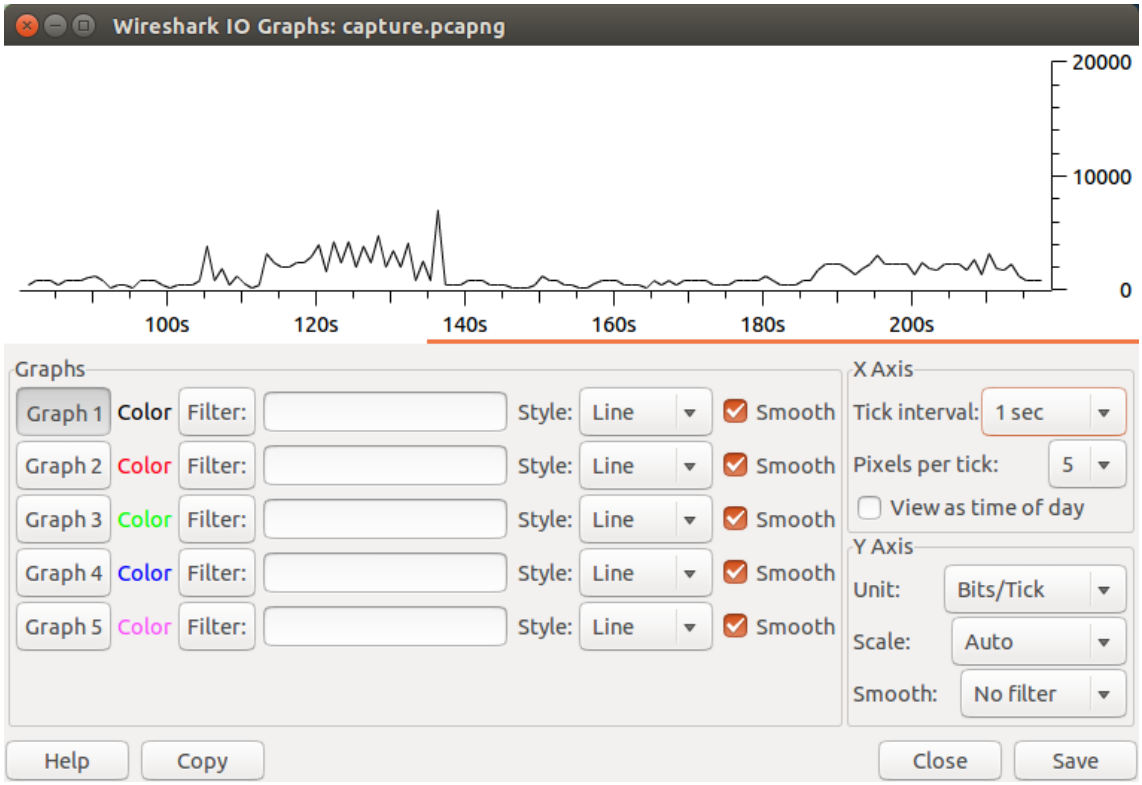

Fig. 6.7: Wireshark - IO Graph

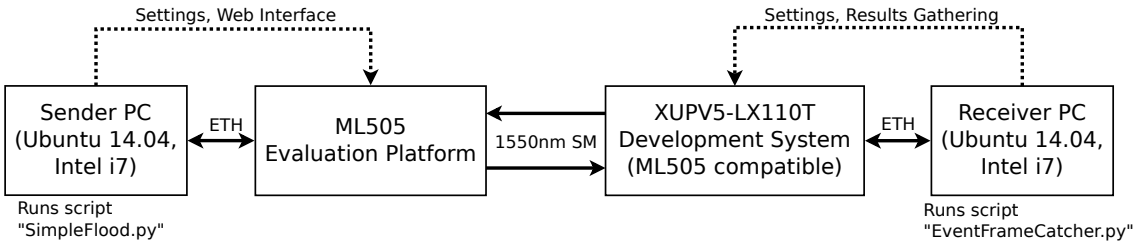

Fig. 6.8: Experimental Setup for BER Measurements

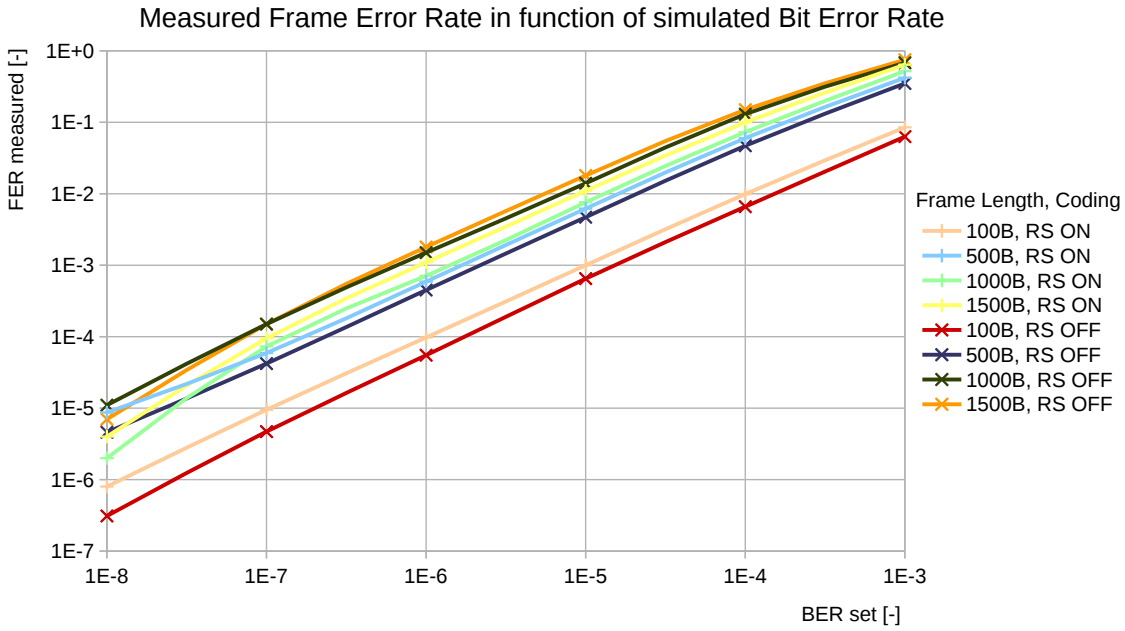

Fig. 6.9: Measured FER in Function of Simulated BER, Frame Length and Used Channel Coding as Parameters

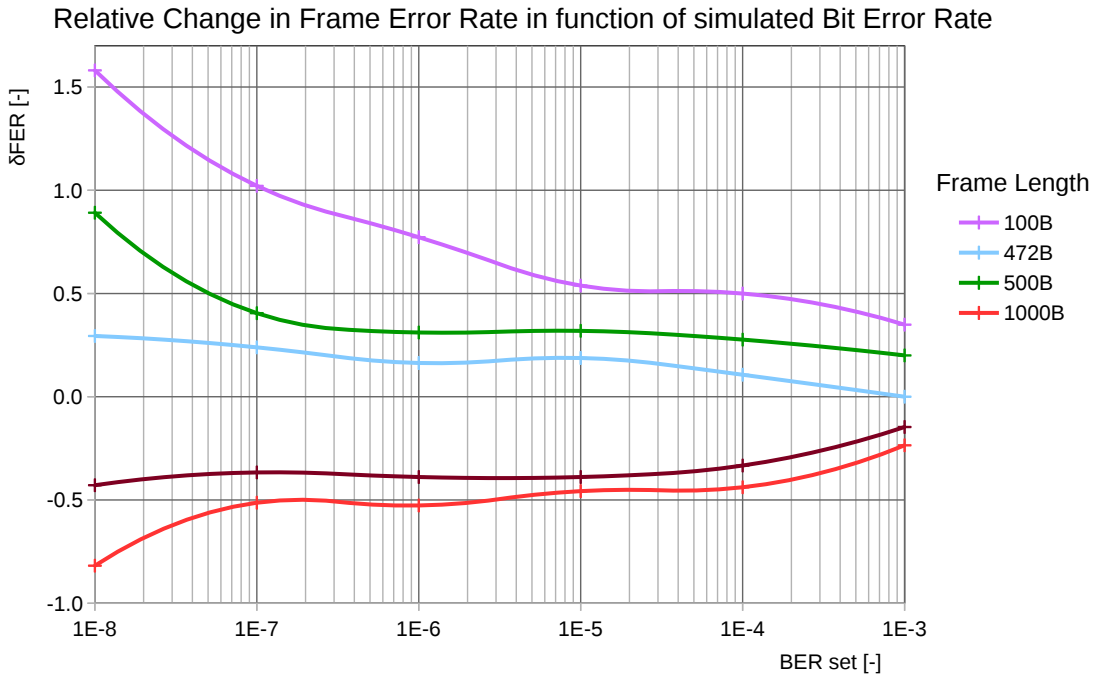

Fig. 6.10: FER Evolution After Enabling the RS Coder, δFER defined in 6.2

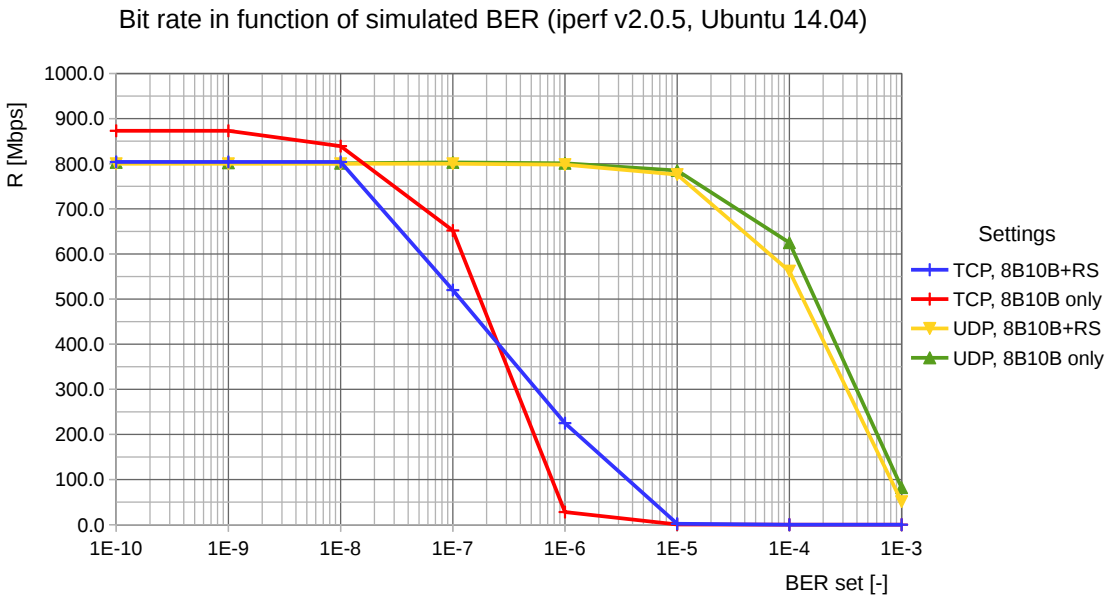

Fig. 6.11: Bit Rate In Function of Bit Error Rate of The Link

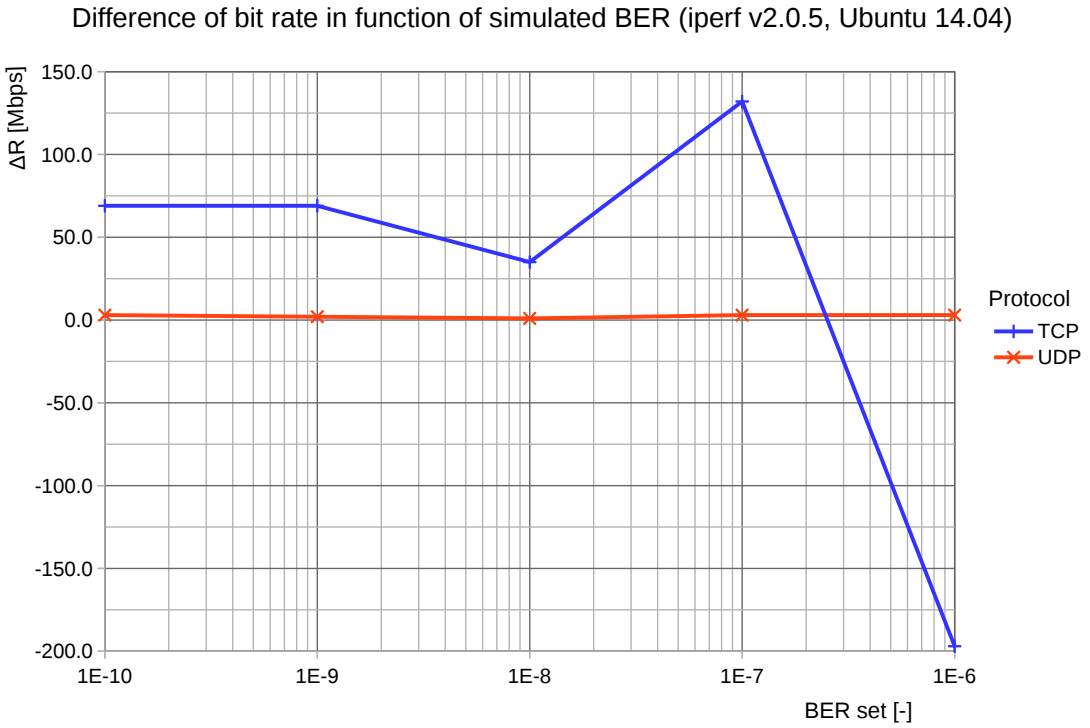

Fig. 6.12: Difference of Bit Rates In Function of Bit Error Rate

7 CONCLUSION

In the first part of the master thesis, different effects influencing the propagation of a laser beam have been studied in order to present different techniques of BER/FER reduction. From these techniques, this thesis focuses primarily on the channel coding combined with a new approach presented in 3.5.2. A project in ISE was realized as well as new reusable HDL modules such as SFP/GTP MAC interface or DDR-based FIFO. The last one is used to implement a new proposed technique (3.5.2) to reduce the BER. A selectable Reed-Solomon coder IP core handling was introduced in the SFP/GTP MAC interface. For validation of the BER measurements, a simple channel simulator was developed. The Reed-Solomon channel coder can be disabled or enabled at runtime, which allows to evaluate its influence to the resulting BER. Based on realized experiments, formulas describing FER in function of the frame length is presented for a situation when a RS coder is enabled and disabled. Impact of the RS coder on the bit rate performance is plotted as well as the evolution of FER in function of simulated BER. The whole project with all necessary documentation, test scripts and tools is available at `https://gitlab.com/OK2NMZ/thesis-fso.git` and can be used to test different channel coders (block or convolutional ones). This thesis implements a reusable research platform which is open to extensions and configurations via embedded microprocessor system.

BIBLIOGRAPHY

[1] Czaputa, Martin. *Durchsatzoptimierung optischer Freiraumübertragungssysteme durch geeignete Modulations- und Kodierungstechniken*, 2012.

[2] Deschamps, G. A. *Gaussian beam as a bundle of complex rays.* In Electronic Letters, p. 2. IET, 1971. doi:10.1049/el:19710467.

[3] Duy A. Luong; Truong C. Thang; Anh T. Pham;. *Average Capacity of MIMO/FSO Systems with Equal Gain Combining over Log-Normal Channels.* Tech. rep., The University of Aizu, Fukushima, 2013. doi: 10.1109/ICUFN.2013.6614831.
URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6614831`

[4] Duyen, Trung H A and Pham, Anh T. *Performance Analysis of MIMO / FSO Systems Using SC-QAM Signaling over Atmospheric Turbulence Channels.* (1):49–56, 2014.

[5] Geisel, William A. *Tutorial on Reed-Solomon Error Correction Coding (MSC-21834)*, 1996.

[6] Gerstlauer, Andreas. *Viterbi*, 2009.
URL `http://users.ece.utexas.edu/~gerstl/ee382v-ics_f09/lectures/Viterbi.pdf`

[7] Haiman, Mark. *Notes on reed-solomon codes.* pp. 1–14, 2003.

[8] Hecht, Eugene. Optics. Adisson-Wesley, 2002. ISBN 0-321-18878-0.

[9] Janík, Lukáš. *Support System for Administration and Control of FSO Transceiver*, 2016.

[10] Jennifer C. Ricklin, Stephen M. Hammel, Frank D. Eaton, Svetlana L. Lachinova. *Atmospheric channel effects on free-space laser communication.* Tech. rep., 2006. doi:10.1007/s10297-005-0056-y.

[11] Křivák, Petr. Long Range Free Space Optical Links. Ph.D. thesis, Brno University of Technology, 2009.

[12] Lee, E. J. *Part 1: optical communication over the clear turbulent atmospheric channel using diversity.* In IEEE Journal on Selected Areas in Communications, pp. 1896–1906. IEEE, 2004. doi:10.1109/JSAC.2004.835751.

[13] Majumdar, Arun K. Optical and Fiber Communication Reports: Free-Space Laser Communications. 2008. ISBN 978-0-387-28652-5.

[14] McGill School of Computer Science. *History of Hamming Codes*. URL http://biobio.loc.edu/chu/web/Courses/Cosi460/hamming_codes.htm

[15] M.E. Thomas, D.D. Duncan. The Infrared and Electro-Optical Systems Handbook. SPIE Press, 1993, vol. 2 ed.

[16] Nakamura, Shinki. *High-power and High Efficiency Yb:YAG Ceramic Laser at Room Temperature*. In Bishnu Pal, editor, Frontiers in Guided Wave Optics and Optoelectronics. 2010. ISBN 978-953-7619-82-4.

[17] Paschotta, Rudinger. *Gouy Phase Shift*, 2008. URL https://www.rp-photonics.com/gouy_phase_shift.html

[18] Sarah J. Johnson. *Introducing Low-Density Parity-Check Codes*.

[19] Subhashree Das. VHDL Implementation of Reed-Solomon Coding. Ph.D. thesis, National Institute of Technology Rourkela, 2011.

[20] Svetlo, Orazio. Principles of Lasers. Springer, 2010, fifth ed. ISBN 978-1-4419-1302-9. doi:10.1007/978-1-4419-1302-9. URL https://abmpk.files.wordpress.com/2014/02/svelto-o-principles-of-lasers-5ed-springer-2009isbn-1441913017o625s_peo__2.pdf

[21] the free encyclopedia, Wikipedia. *Gaussian beam*. URL https://en.wikipedia.org/wiki/Gaussian_beam

[22] Thomas Weyrauch; Mikhail A. Vorontsov. *Free-space laser communications with adaptive optics: Atmospheric compensation experiments*. p. 26. 2008. doi:10.1007/s10297-005-0033-5.

[23] Tikhonov, Nikolay. *Characterization of optical turbulence (Cn2) data measured at the ARL A_LOT facility*. Tech. rep., Army Research Laboratory, 2005.

[24] Tkacik, Thomas. *A Hardware Random Number Generator*. Tech. rep., 2002. URL http://www.chesworkshop.org/ches2002/presentations/Tkacik.pdf

[25] Weyrauch, Thomas and Vorontsov, Mikhail A. *Free-space laser communications with adaptive optics: Atmospheric compensation experiments.* 379:355–379, 2004. doi:10.1007/s10297-005-0033-5.

[26] Wilfert, Otakar. Optoelektronika.

[27] Wilson, Stephen G., Cao, Qianling, and Leveque, James H. *Free-Space Optical MIMO Transmission With Q-ary PPM.* 53(8):1402–1412, 2005.

[28] Xilinx. *Xilinx UG196 Virtex-5 FPGA RocketIO GTP Transceiver, User Guide.* Tech. rep., 2009.
URL `www.xilinx.com/support/documentation/user_guides/ug196.pdf`

[29] Xilinx. *UG086 Xilinx Memory Interface Generator (MIG), User Guide.* Tech. rep., 2010.
URL `http://www.xilinx.com/support/documentation/ip_documentation/ug086.pdf`

[30] Xilinx. *Xilinx DS251, LogiCORE IP Reed-Solomon Encoder v7.1 Data Sheet.* Tech. rep., 2011.
URL `http://www.xilinx.com/support/documentation/ip_documentation/rs_encoder_ds251.pdf`

[31] Xilinx. *Xilinx DS252, LogiCORE IP Reed-Solomon Decoder v7.1 Data Sheet.* p. 30, 2011.
URL `http://www.xilinx.com/support/documentation/ip_documentation/rs_encoder_ds252.pdf`

[32] Xilinx. *Xilinx UG194 Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC, User Guide.* Tech. rep., 2011.
URL `http://www.xilinx.com/support/documentation/user_guides/ug194.pdf`

[33] Zdenek Kolka; Viera Biolkova; Otakar Wilfert; Dalibor Biolek. *Simulation Model of Correlated FSO Channels.* p. 4. 2015. ISBN 9781479981212.

LIST OF SYMBOLS, PHYSICAL CONSTANTS AND ABBREVIATIONS

BER	Bit Error Rate
MLSE	Most Likely Sequence Estimation
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
FSO	Free Space Optics
LDPC	Low density parity-check
SFP	small form-factor pluggable: transceiver interface
MIMO	Multiple Input Multiple Output
RS	Reed-Solomon (encoder/decoder)
FER	Frame Error Rate
HDL	Hardware Description Language: VHDL, Verilog, etc.
SNR	Signal to Noise Ratio
SISO	Single Input Single Output (vs. MIMO)
ASER	Average Symbol Error Rate
AO	Adaptive Optics
VLSI	Very Large Scale of Integration
WFS	Wave Front Sensor
FEC	Forward Error Correction
BCH	Bose Chaudhuri and Hocquenghem (encoder/decoder)
MLSE	Maximum Likelihood Sequence Estimation
FSM	Finite State Machine
SGMII	Serial Gigabit Media-Independent Interface
RSSI	Received Signal Strength Indication

DSP	Digital Signal Processing / Digital Signal Processor
IP	Intellectual Property
FIFO	First In First Out (memory, buffer)
I^2C	Inter-Integrated Circuit: serial communication interface
IDE	Integrated Development Environment
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
SSD	Solid State Drive
DDR	Double Data Rate (memory technology)

LIST OF APPENDICES

A PIN SETTINGS AND EXPERIMENT SETUP

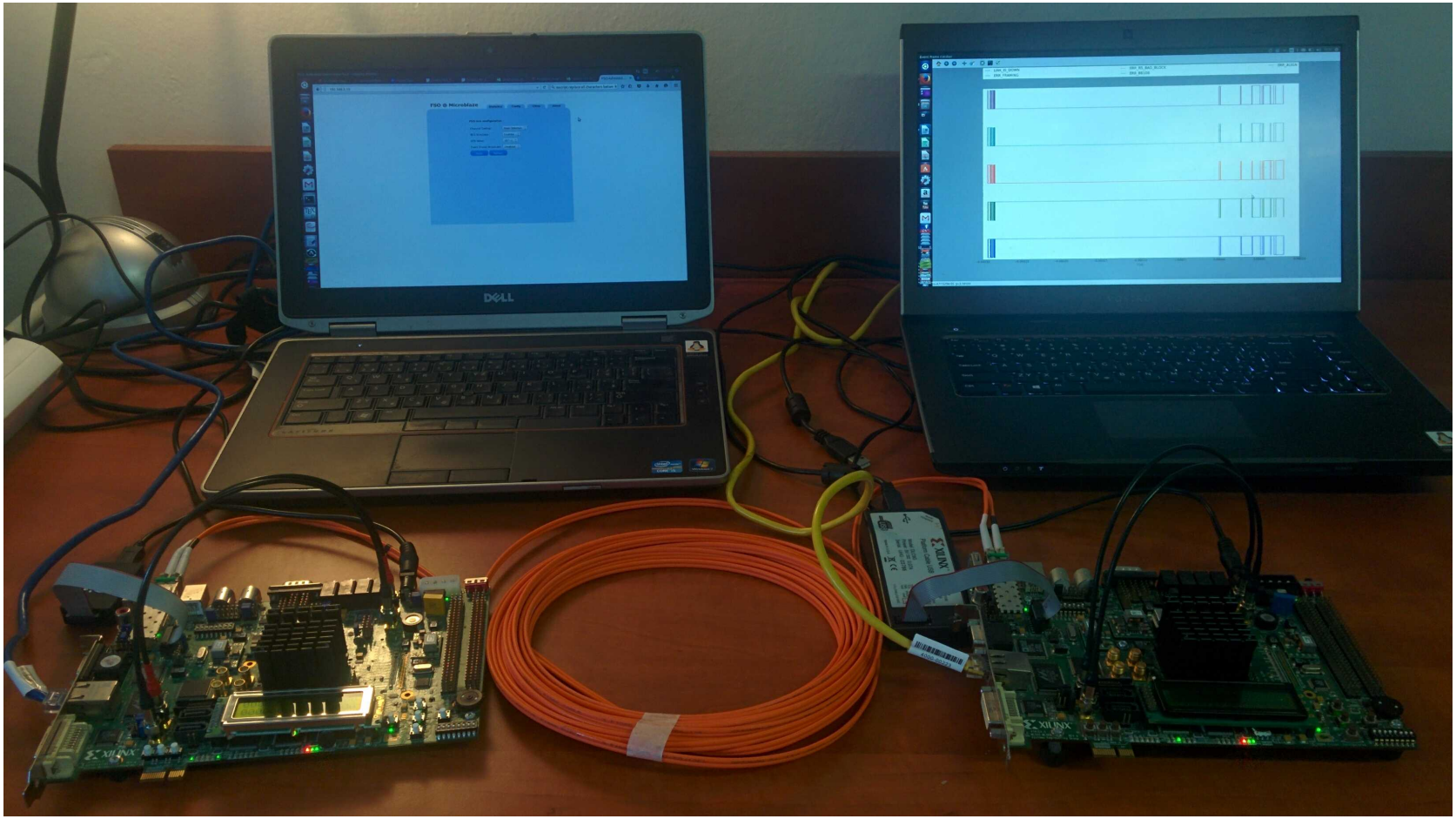

Fig. A.1: Measurements' Setup Used For Experiments

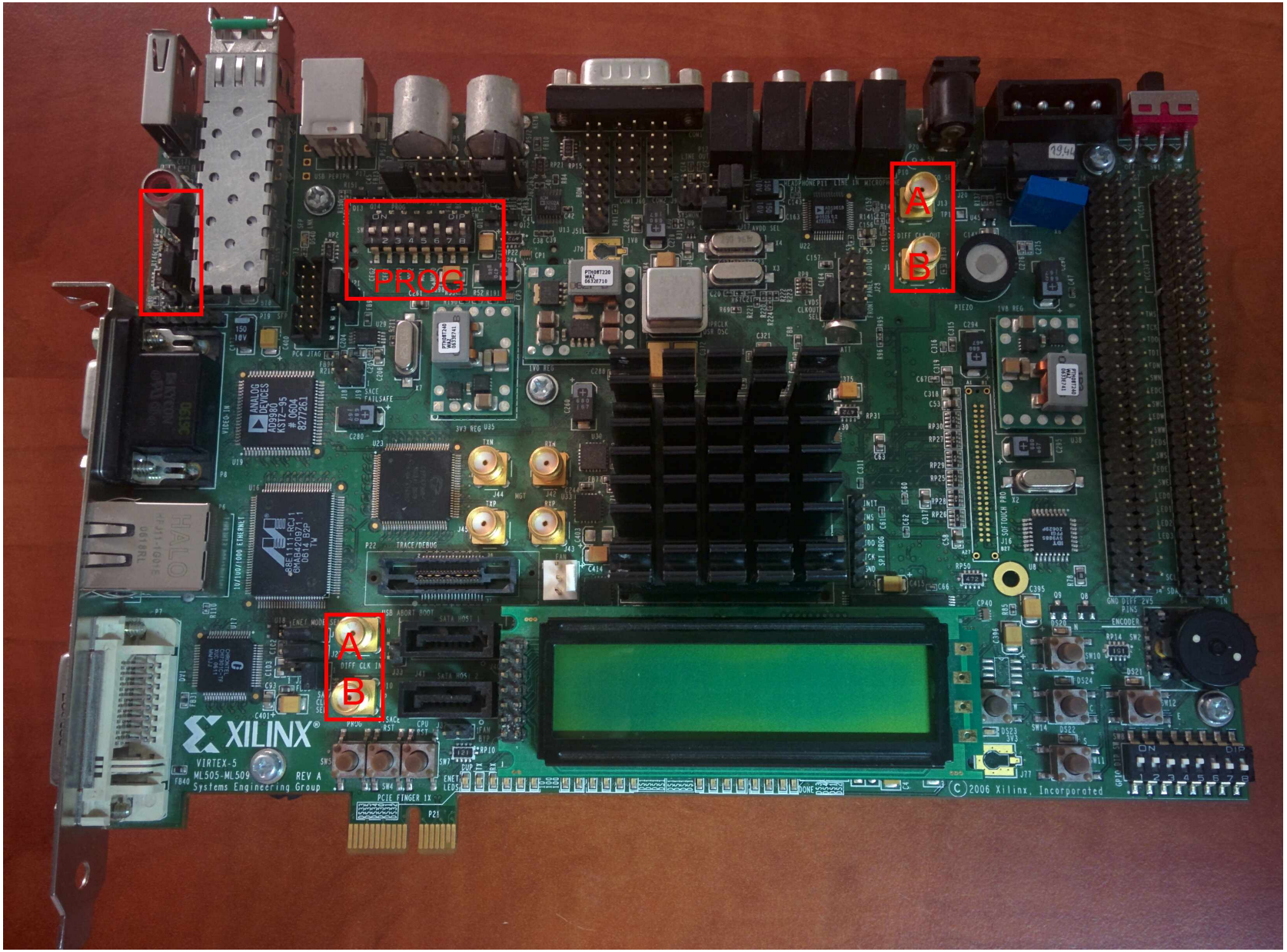

Fig. A.2: Pin Settings, Top View

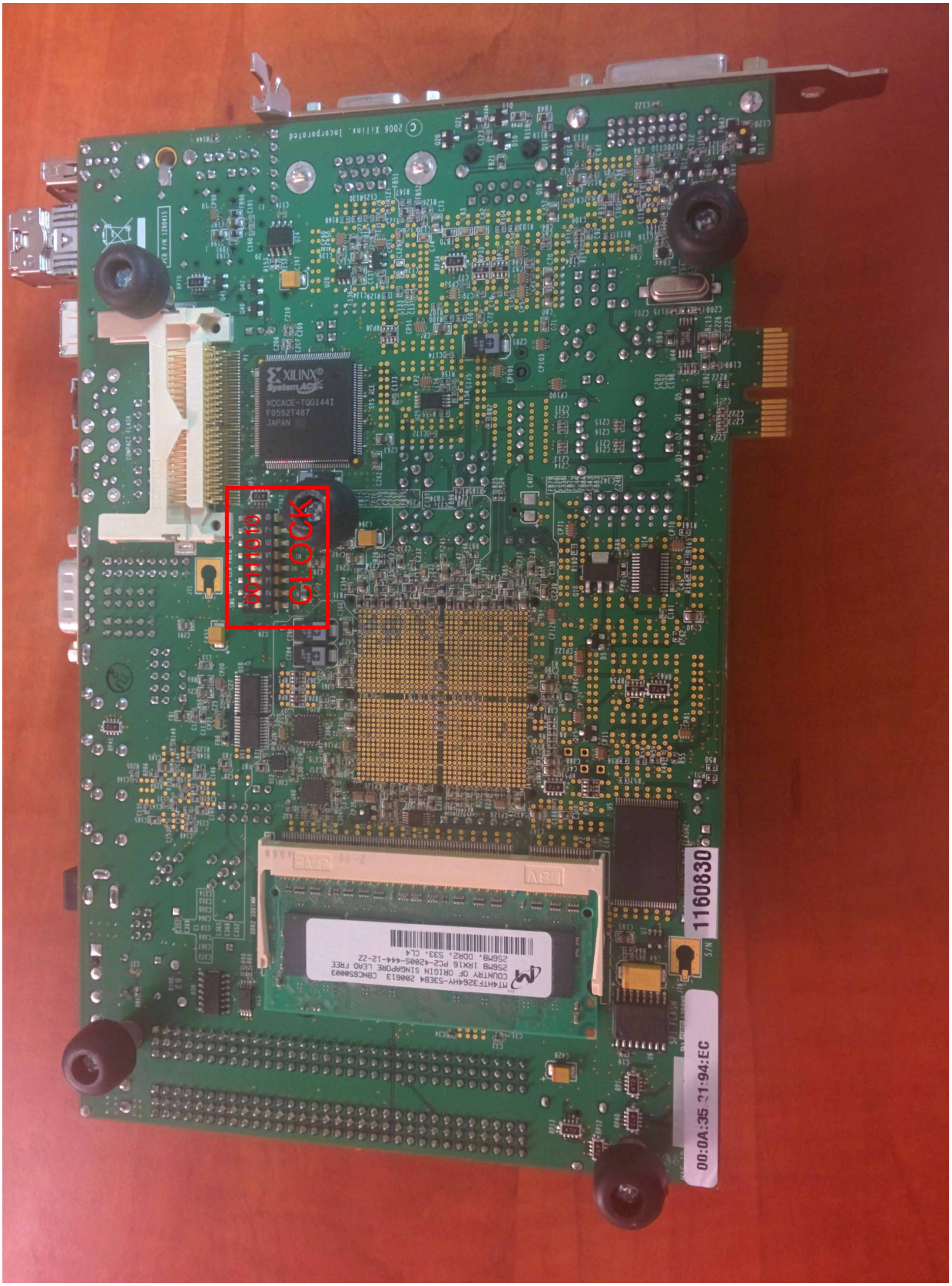

Fig. A.3: Pin Settings, Bottom View

B GRAPH DATA SOURCES

	FER									
	Channel Coding: 8B10B + RS(239,255)					Channel Coding: 8B10B				
BER	100B	472B	500B	1000B	1500B	100B	472B	500B	1000B	1500B
1E-3	8.5E-2	3.4E-1	4.2E-1	5.2E-1	6.4E-1	6.3E-2	3.4E-1	3.5E-1	6.8E-1	7.5E-1
1E-4	9.9E-3	5.2E-2	6.0E-2	7.3E-2	1.0E-1	6.6E-3	4.7E-2	4.7E-2	1.3E-1	1.5E-1
1E-5	1.0E-3	5.7E-3	6.2E-3	7.6E-3	1.1E-2	6.5E-4	4.8E-3	4.7E-3	1.4E-2	1.8E-2
1E-6	9.8E-5	5.7E-4	5.9E-4	7.1E-4	1.1E-3	5.5E-5	4.9E-4	4.5E-4	1.5E-3	1.8E-3
1E-7	9.5E-6	5.7E-5	5.9E-5	7.3E-5	9.5E-5	4.7E-6	4.6E-5	4.2E-5	1.5E-4	1.5E-4
1E-8	8.0E-7	6.6E-6	8.7E-6	2.0E-6	4.0E-6	3.1E-7	5.1E-6	4.6E-6	1.1E-5	7.0E-6

	(calculated) δFER [%]				
BER	100B	472B	500B	1000B	1500B
1E-3	35	0	20	-24	-15
1E-4	50	11	28	-44	-33
1E-5	54	19	32	-46	-39
1E-6	77	16	31	-53	-39
1E-7	102	24	40	-51	-37
1E-8	158	29	89	-82	-43

	R [Mbps]			
	8B10B + RS(239,255)		8B10B	
BER	TCP	UDP	TCP	UDP
1E-3	0	51	0	82
1E-4	0	562	0	625
1E-5	2	776	0.5	785
1E-6	225	798	28	801
1E-7	520	800	652	803
1E-8	804	800	839	801
1E-9	804	800	873	802
1E-10	804	800	873	803

	ΔR [Mbps]	
BER	TCP	UDP
1E-3	0	31
1E-4	0	63
1E-5	0	9
1E-6	-197	3
1E-7	132	3
1E-8	35	1
1E-9	69	2
1E-10	69	3

Tab. B.1: Source Data for Figures: 6.9, 6.10, 6.11 and 6.12