CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

FPGA emulation of polarization-mode dispersion in optical fibers

HONG ZHOU & HAORUI KAN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Master's thesis 2021

FPGA emulation of polarization-mode dispersion in optical fibers

HONG ZHOU & HAORUI KAN

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

FPGA emulation of polarization-mode dispersion in optical fibers
HONG ZHOU & HAORUI KAN

Supervisor: Per Larsson-Edefors, Department of Computer Science and Engineering
Examiner: Lena Peterson, Department of Computer Science and Engineering

Master's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2021

FPGA emulation of polarization-mode dispersion in optical fibers
HONG ZHOU & HAORUI KAN
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Improving the transmission capacity through polarization-division mode (PDM) technology has given rise to polarization-mode dispersion (PMD) effect in optical fiber communication system. Research and analysis of how various parameters on optical fiber PMD will impact the system performance can help us reducing or equalizing PMD more efficiently. But the parameters cannot be easily changed in the optical fiber experiment to observe the impact on PMD. Therefore, we have built a real-time digital model of PMD based on field-programmable gate array (FPGA), which can change the parameters during operation and capture the output signal after adding PMD. At the same time, a general equalizer digital model was built to handle the PMD impairment. We build a two-polarization optical fiber emulating system which includes transmitter, additive white Gaussian noise (AWGN), PMD emulator, equalizer and receiver. This system allowed us to observe the impact of changing the various parameters of the PMD on the equalizer.

Keywords: Optical communication, PMD emulator, CMA equalizer, FPGA, real-time, fractional delay filter

Acknowledgements

First of all, we would like to thank Prof. Per Larsson-Edefors for giving us the opportunity of this master's graduation project. As our thesis supervisor, he gave us many suggestions and guidance. We are also grateful to Prof. Magnus Karlsson for enlightening our understanding of fiber-optical communication and giving suggestions during the M.Sc. project so that we can have enough communication knowledge to complete it. At the same time, thanks to Lena Peterson for the guidance of our M.Sc. thesis.

In addition, we would like to thank Erik Börjeson for providing us with the hardware foundation of the fiber-optical communication system, and at the same time providing us with lots of system optimization suggestions, so that our M.Sc. design can proceed smoothly. We would also like to thank Jochen Schröder, Christian Häger and Jinxiang Song for their suggestions on fiber-optical communication systems.

Hong Zhou & Haorui Kan, Gothenburg, June 2021

Contents

Abbreviations

ADC	Analog-to-digital converter
AWGN	Additive white Gaussian noise
BER	Bit error rate
BPSK	Binary phase-shift keying
CMA	Constant modulus algorithm
DGD	Differential group delay
DSP	Digital signal processing
FFT	Fast Fourier transform
FIR	Finite impulse response
FPGA	Field-programmable gate array
IFFT	Inverse fast Fourier transform
ILA	Integrated logic analyzer
ISI	Intersymbol interference
LMS	Least mean squares
LUT	Lookup table
MCMA	Modified constant modulus algorithm
MIMO	Multi-input and multi-output
PBS	Polarization beam splitter
PDL	Polarization-dependent loss
PDM	Polarization division multiplexing
PMD	Polarization-mode dispersion
PSP	Principal states of polarization
QAM	Quadrature amplitude modulation
QPSK	Quadrature phase shift keying
RLS	Recursive least squares
RNG	Random number generation

RRC	Root-raised-cosine
RSOP	Rotation of state of polarization
RTL	Register-transfer level
VHDL	VHSIC hardware description language

1
Introduction

Polarization division multiplexing (PDM) technology is widely used in optical-fiber communications. PDM technology uses the polarization characteristics of light when transmitted in a single-mode fiber, and uses two independent orthogonal polarization states of the wavelength as independent channels to transmit two signals respectively. This has the effect that PDM doubles the transmission capacity of optical fibers without adding additional bandwidth resources [1]–[3], thus significantly improving the spectrum efficiency of the system. However, the application of PDM technology leads to problems caused by interactions between the two polarizations. Impairments such as polarization-mode dispersion (PMD) and polarization-dependent loss (PDL) seriously affect the transmission quality of the system and reduce the spectrum utilization and transmission efficiency [4].

Digital signal processing (DSP) has become essential to compensate for different fiber-optical transmission impairments. But as the capacity and throughput have increased, the power dissipation of DSP circuits has increased significantly. Since power dissipation depends on physical properties of a circuit as well as the logic signals applied to the circuit, power analysis of DSP circuits requires circuits to be implemented in the context of a system that can provide meaningful input signals to the DSP circuit implementation. The problem is that building a physical-level optical fiber communication system to evaluate the DSP implementation requires complicated experimental setups [5]. In addition, since the prevalent approach of optical experiments to evaluate DSP is to only analyze snapshots of transmitted data, real-time aspects of DSP cannot be analyzed. Therefore, a digital model of optical-fiber communication built inside an field-programmable gate array (FPGA) can be used to simulate the impairments on a two-polarization system and the performance of the DSP-based equalizer or de-multiplexer in real time.

1.1 Related work

The study of PMD in optical fiber originated from the study of the polarization state of light in coherent optical communication systems in the 1980s. In recent years, with the increase of bit rate per channel, researchers found that PMD can significantly damage the transmission performance and limit the transmission distance of the system [6]–[8]. To correctly evaluate the signal distortion caused by PMD theoretically, simulation models have been extensively studied [9]–[11]. PMD

can be optically emulated in optical communication systems by a combination of components such as polarization maintaining fibers or birefringent crystals and was introduced in detail in [6]. The waveplate model [12] which was put forward by Curti et al. in 1989 is the most popular mathematical PMD simulation model and will be used in our project. This model has been implemented in many software-based simulation systems such as [13] and [14]. However, no matter if using optical components or software-based simulator, it is difficult to emulate the change of PMD and how it influences the later compensation in real time. FPGA has not been used for this purpose before and we believe that it can make a difference, because this allows us to change the PMD by changing the factors that affect the PMD, such as the delay and rotation angle of each fiber section, to analyze the impact of different factors on the compensation capability and power consumption of equalizer or de-multiplexer.

1.2 Thesis goals

In order to facilitate long, real-time emulations of fiber communication systems with QPSK and 16QAM modulation format on an FPGA, the overall goal of the project is to develop register-transfer level (RTL) for synthetic models of a transmitter, a fiber channel with two different polarizations, and a receiver with a polarization equalizer or de-multiplexer DSP. With the addition of two polarizations to the CHOICE environment, which is a single-polarization FPGA-based model developed for emulating the fiber impairments in real time [15], we aim to digitally capture the noise of the 2-polarization system in the fiber and compensate the PMD by equalizer or de-multiplexer.

The top-level research questions in this thesis are

- How can we develop a digital PMD and equalizer or de-multiplexer model that are both faithfully capturing PMD effects in real fibers and feasible to implement in the CHOICE environment?
- How do we convert the PMD and equalizer or de-multiplexer model from MATLAB to FPGA?
- Running real-time FPGA experiments with our PMD model and an equalizer, what is the impact of different parameters of PMD on equalizer convergence speed and power dissipation?

1.3 Thesis outline

The main content of each chapter of the thesis is arranged as follows: Chapter 2 is the basic introduction to the optical fiber communication system, including the modulation format, RRC filter, PMD generation and compensation, and the introduction of the CHOICE system. Chapter 3 describes the implemented methodology. In chapter 4 we describe the implementation process of the MATLAB and VHDL of the PMD emulator and equalizer. Chapter 5 introduces the VHDL results of PMD and CMA equalizer under different parameter settings, including simulation

results and FPGA running results. In Chapter 6, we discussed the accuracy and resource consumption of the output data under different data wordlength, as well as the resource consumption of each component under different modulation format. Chapter 7 summarizes the results achieved in this project and the future prospects.

2
Technical background

In this chapter, the basic concepts of optical fiber communication systems are introduced, including modulation format, causes of PMD and compensation methods and a brief introduction to the CHOICE system [15]. In addition, mathematical models and principles for implementing PMD are introduced.

Fig. 2.1 shows the general structure of an optical fiber communication system, including transmitter, up-sampling, fiber, and receiver. The transmitter converts and modulates the electrical signals of both channels into two optical signals which contain I and Q components. After being pulse shaped by an RRC filter, the signals are sent into the optical fiber, in which PMD and other impairments degrade the signal. The optical signals are converted back into the electrical domain in the receiver, whose equalizer compensates for different linear impairments.

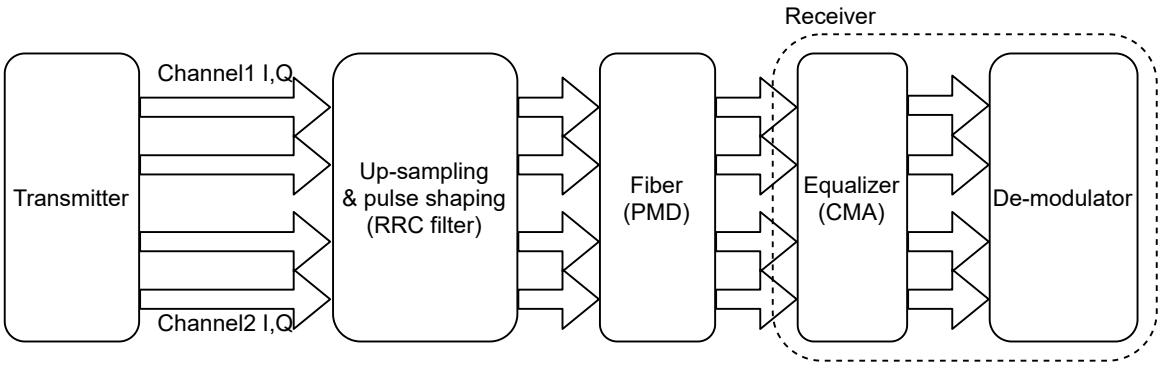


Figure 2.1: Simplified diagram of optical fiber communication system

2.1 Modulation formats

Quadrature phase-shift keying (QPSK) and 16 quadrature amplitude modulation (16QAM) are common modulation methods in optical communication systems. In this project, while we focus on QPSK and 16QAM, we will work to ensure the PMD emulator should work also for higher-order modulation formats, such as 64QAM, without requiring large changes.

QPSK is a kind of quaternary phase modulation, which has good anti-noise characteristic and frequency band utilization. It is widely used in satellite links, digital clusters and other communication services [16]. The sinusoidal carrier of a QPSK

signal has four possible discrete phase states, and each carrier phase carries the information of two bits. 16QAM is an extension to QPSK, and its symbols are distinguished by both phase and amplitude and each state has four binary symbols [17]. The constellation diagrams of both QPSK and 16QAM are shown in Fig. 2.2.

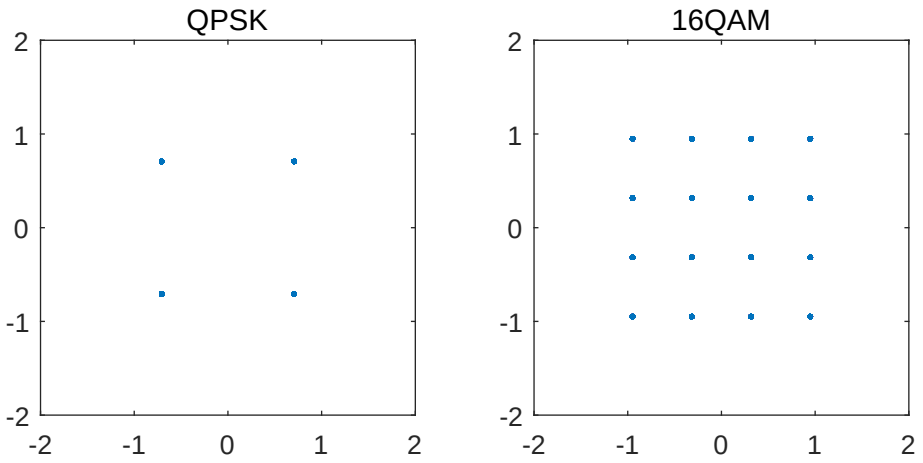


Figure 2.2: Constellation diagrams of QPSK and 16QAM

2.2 RRC filter

After the digital baseband is modulated, it must be transmitted after pulse shaping. Because an ideal rectangular pulse has an infinite bandwidth in the frequency domain, after passing through the low-pass filter, it becomes narrower in the frequency domain, resulting in a wider time domain. Therefore, adjacent pulse signals will interfere with each other [18], the ideal impulse without intersymbol interference (ISI) cannot be achieved. The root raised cosine (RRC) filter is known to reduce ISI, and it is used at the transmitter and receiver end to ensure that there is no ISI influence at the sampling time [18]. The transfer function of RRC filter is shown below:

$$H_{RRC}(\omega) = \begin{cases} \sqrt{T} & |\omega| \leq \omega_1 \\ \sqrt{\frac{T}{2}}\sqrt{1+\cos(\pi\frac{|\omega|-\omega_1}{r\omega_c})} & \omega_1 \leq |\omega| \leq \omega_2 \\ 0 & |\omega| > \omega_2 \end{cases} \tag{2.1}$$

where $\omega_1 = \frac{1-r}{2}\omega_c$ and $\omega_2 = \frac{1+r}{2}\omega_c$ [19].

2.3 PMD

The geometry of the ideal optical fiber is uniform and stress-free, so the two polarization states of the light wave travel at exactly the same speed without any delay at the other end of the fiber. However, in practice, the circular symmetry of the fiber is usually destroyed by various factors, such as asymmetry in production, residual stress, applied stress and so on. Due to the existence of optical birefringence,

two states with different polarizations have different group velocities. Polarization-mode dispersion effect is a kind of linear electromagnetic propagation phenomenon in single-mode fibers [20]. In high–baud-rate transmitting systems, the resulting PMD phenomena would lead to significant pulse distortion and system impairments which will limit the system performance [20].

2.3.1 Differential group delay

PMD is divided into first-order and higher-order PMD. We first realize first-order PMD. For the first-order PMD in a single-mode fiber, the two polarization states have different delays during transmission. The delay between them is the differential group delay (DGD), resulting in signal pulse broadening, as shown in Fig. 2.3.

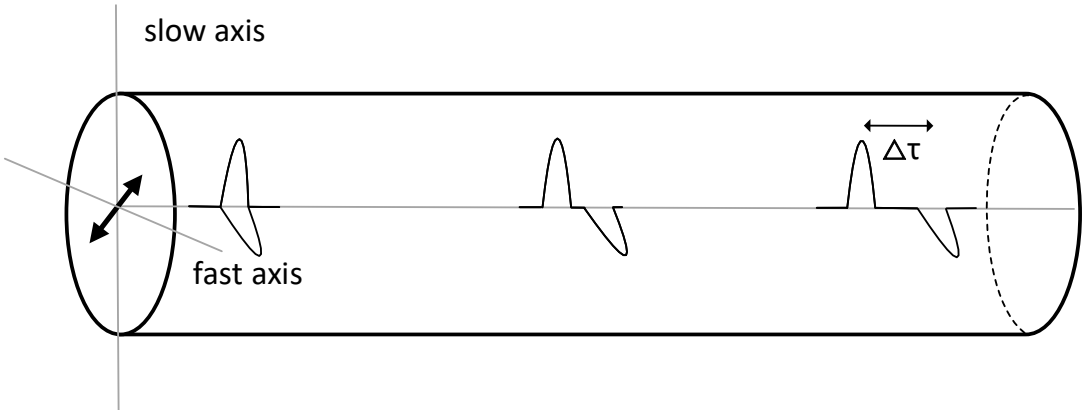


Figure 2.3: Differential group delay in an optical fiber

Through principal states of polarization (PSPs), the probability distribution of DGD($\triangle\tau$) in fibers obeys Maxwell's distribution [21]:

$$p(\triangle\tau) = \begin{cases} \sqrt{\frac{2}{\pi}}\frac{\triangle\tau^2}{\alpha^3}e^{\frac{-\triangle\tau^2}{2\alpha^2}} & \triangle\tau \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

where $\alpha = \frac{2\sqrt{2}}{\sqrt{3\pi}} \triangle \tau_{rms}$, and τ_{rms} is the root mean square(rms) value of the DGD.

$$\triangle\tau_{rms} = \left\langle \triangle\tau^2 \right\rangle = \left\langle \Omega_1^2 + \Omega_2^2 + \Omega_3^2 \right\rangle \tag{2.3}$$

where $\Omega_1, \Omega_2, \Omega_3$ are dispersion vectors with independent Gaussian distributions.

2.3.2 Waveplate model

Under the first-order PMD approximation, the PMD can be numerically simulated by the waveplate model which is shown in Fig. 2.4. In this model, the fiber is considered as a concatenation of many waveplates which have random birefringence and rotation angle. The optical fiber transmission in one section is represented by the Jones matrix [20].

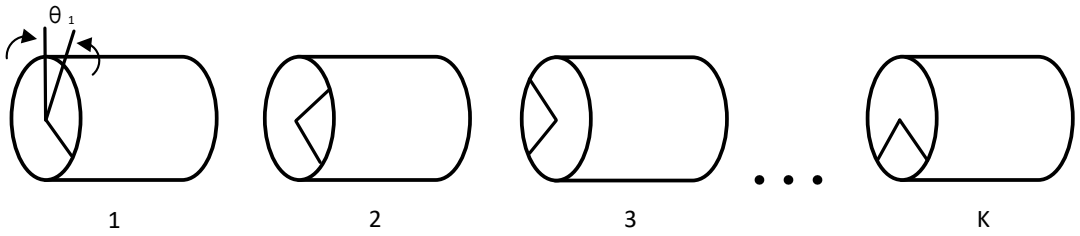


Figure 2.4: Waveplates with random birefringence and mode coupling for modeling a real optical fiber

$$N_k(\omega) = R_k M_k(\omega) \qquad k = 1, 2, ...K \tag{2.4}$$

where $N_k(\omega)$ is the fiber transmission matrix in the presence of birefringence.

$$M_k(\omega) = \begin{bmatrix} e^{j\omega\tau_k/2} & 0 \\ 0 & e^{-j\omega\tau_k/2} \end{bmatrix} \qquad k = 1, 2, ...K \tag{2.5}$$

where τ_k is the DGD of one section, and K is the number of fiber sections.

$$R_k = \begin{bmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{bmatrix} \qquad k = 1, 2, ...K \tag{2.6}$$

where R_k is the phase rotator. θ_k is a random variable which varies from $-\pi$ to π.

2.4 Lagrange fractional delay filter

As its name implies, a fractional delay filter can delay a signal by a non-integer time, which could be used to introduce DGD in our system. The main implementation methods of a fractional delay filter are the windowing function method, Lagrange interpolation method and Farrow filter method [22]. Since only additions and multiplications are needed in the Lagrange interpolation, it's more hardware friendly compared with other methods. The filter coefficients required for the Lagrange interpolation method [22] are shown below:

$$h(n) = \prod_{k=0, k\neq n}^{N-1} \frac{D-k}{n-k}, n = 0, 1, 2, 3, \ldots, N-1 \tag{2.7}$$

$$D = D_0 + u. \tag{2.8}$$

Here N is the tap count of the finite-impulse response (FIR) filter and D is the group delay which contains the constant delay D_0 and the fractional delay u. For a filter with odd number of taps, the constant delay is $D_0 = (N-1)/2$, so $D = (N-1)/2+u$.

2.5 Polarization de-multiplexing and equalization

Polarization de-multiplexing and equalization can be used to cancel the effects of PMD. And the equalizer can also be used for polarization de-multiplexing. In this subsection their basic principles are introduced.

2.5.1 Polarization de-multiplexing

The polarization state of the optical signal transmitted by the PDM system will randomly change, resulting in the rotation of state of polarization (RSOP) effect [23]. The two polarization signals cannot be completely separated at the receiver, so it is necessary to split the orthogonally multiplexed optical signal at the receiver using a polarization de-multiplexer to obtain two independent signals.

There are two methods for polarization de-multiplexing and we will be concerned with the second type of de-multiplexing:

(1) Polarization de-multiplexing based on direct detection, by tracking and adjusting the polarization states of the optical signals in the two transmission channels, and then separating the two orthogonal polarization signal by polarization beam splitter (PBS).

(2) Polarization de-multiplexing based on DSP and coherent detection technology. At the receiver, a local oscillator with the same frequency and phase as the optical signal at the transmitter is mixed with the received optical signal in 90 degree optical hybrid, converted from analog to digital in an ADCs [24], and then processed by a DSP algorithm to compensate PMD, PDL and so on.

2.5.2 MIMO technique and CMA Equalizer

Multiple-input and multiple-output(MIMO) is widely used in telecommunication systems for multiplying the capacity by taking advantage of the spatial dimension [25]. When applied to the optical communication field, different polarizations or other spatial dimensions of orthogonal fiber are used for multi-channel communication [26]. However, due to some defects in the design and manufacturing of optical fiber, cross-talk happens among signals carried in different orthogonal dimensions, such as PMD which is introduced above [20], [26]. Therefore, the equalizer in digital receiver also needs to apply the corresponding MIMO digital signal processing algorithm to restore the original signals [27] and de-multiplex them.

Assuming a MIMO system with M transmitters and N receivers, an M by N channel matrix $H_{m\times n}$ like Eq. (2.9) will be formed between the transceiver and receiver.

$$H_{m\times n} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{m1} & h_{m2} & \cdots & h_{mn} \end{bmatrix} \tag{2.9}$$

Because cross-talk seriously affects the performance of the system, it needs to be

balanced or equalized at the receiving end. In general, the channel transmission matrix $H_{m\times n}$ is a linear unitary matrix, so the original signal can be restored by finding the inverse matrix of $H_{m\times n}$. In the frequency domain, it is assumed that the channel response is $H(f)$ and the response function of the equalization system is $W(f)$, then the final system response $G(f)$ can be expressed as $G(f) = H(f)\times W(f)$. For MIMO, we expect the channel transmission system response to be a linear system whose impulse response is an identity matrix, which means that we expect $G(f)$ to be an identity matrix. A classical two-input-two-output MIMO equalizer is shown in Fig. 2.5, which can compensate PMD and perform polarization de-multiplexing.

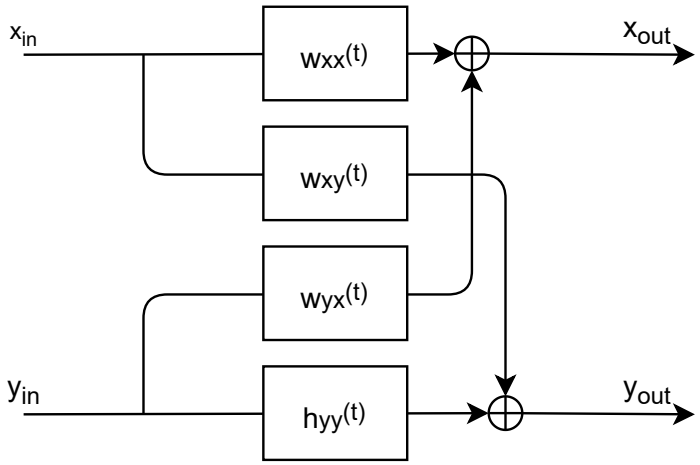


Figure 2.5: Structure of 2x2 MIMO polarization de-multiplexing Equalizer

Common MIMO equalization algorithms can be divided into non-training-sequence and training-sequence algorithms. The non-training-sequence equalization algorithms, also known as blind equalization algorithms, use the characteristics of the received signal itself without the help of training sequence. The advantages of blind equalization method are that no training sequence is needed and bandwidth resources are saved. The disadvantages are that the convergence speed is slower and the convergence accuracy is lower while its performance is less stable than that of the equalization algorithm with training sequence. Constant modulus blind equalization algorithm (CMA) is the most common used blind equalization algorithm [28]. In the equalization algorithm with training sequence, the training sequence is used as the reference signal comparing with the received signal to update the filter coefficients. The advantages are the fast convergence speed and good real-time performance, while the disadvantages are the need to send a certain length of training sequence in a circular manner, which occupies a certain bandwidth resource. Commonly used training sequence equalization algorithms include the least mean square algorithm (LMS) and the recursive least squares algorithm (RLS). In the equalizer part, we only focus on compensating the generated QPSK PMD signal, so we focus on the CMA eqaulizer.

Fig. 2.6 shows the flow graph of CMA algorithm [29].

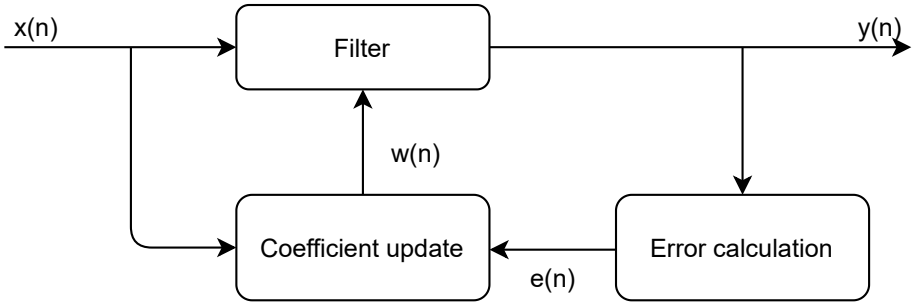


Figure 2.6: Structure of CMA equalizer

The error $e(n)$ is defined as Eq. (2.10).

$$e(n) = y(n) \times (R^2 - |y(n)|^2) \tag{2.10}$$

Here R^2 is a constant which is defined by Eq. (2.11) and determined by modulation format and signal amplitude of transmitter.

$$R^2 = \frac{E[|y(n)|^4]}{E[|y(n)|^2]} \tag{2.11}$$

The coefficients update can be explained by Eq. (2.12).

$$w(n+1) = w(n) + \mu e(n) x^*(n) \tag{2.12}$$

2.6 Filter Structure

Because different FIR structures can lead to different hardware resource consumption and critical path delays, different FIR structures need to be considered to meet the design requirements.

2.6.1 Direct form FIR

FIR means finite impulse response, that is, the number of FIR accumulations at each sampling moment is limited. FIR filters are known to have good linear characteristics. For an FIR filter with N taps:

$$y[n] = x[n] * h[n] = \sum_{k=0}^{N-1} x[n]h[n-k] = \sum_{k=0}^{N-1} x[n-k]h[k] \tag{2.13}$$

Here $*$ is the convolution symbol, $x[n]$ is the input signal and $y[n]$ is the output signal, $h[n]$ is the coefficient of the filter.

Expanding Eq. (2.13) to get Eq. (2.14) as follows:

$$y[n] = x[n]h[0] + x[n-1]h[1] + \cdots + x[n-N+1]h[N-1] \tag{2.14}$$

We can convert Eq. (2.14) to a circuit structure to get the structure of direct FIR, as shown in Fig. 2.7.

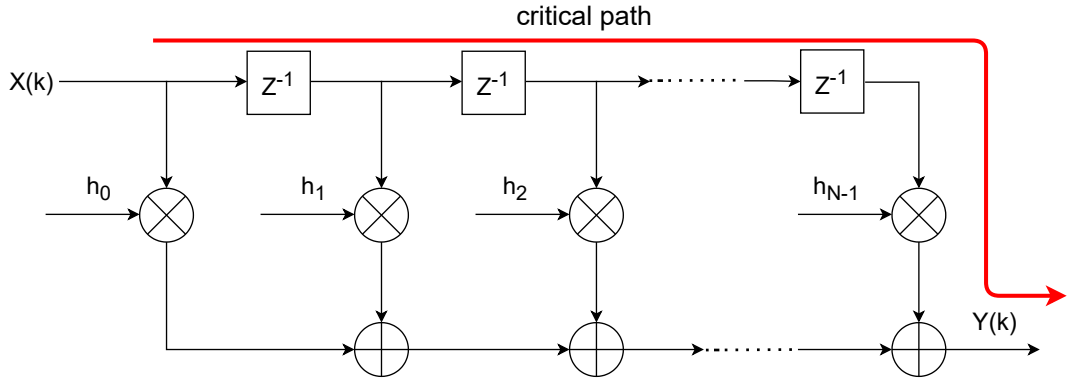


Figure 2.7: Structure of direct FIR

Here z^{-1} represents a delay unit, $\otimes$ represents a multiplier and $\oplus$ represents an adder. The red line in Fig. 2.7 shows the critical path, if the delay of a multiplier is τ_{mul} and the delay of an adder is τ_{add}. The critical path $\tau_{cp} = \tau_{mul} + N \cdot \tau_{add}$

2.6.2 Transposed FIR

When the order of the filter increases, the critical path of the direct form filter will also increase. Therefore, we can use the transposed FIR to reduce the critical path. We reverse the input and output positions of the direct form FIR and the flow direction of the branches to obtain the structure of the transposed FIR as shown in Fig. 2.8. The critical path is shown by the red line, which is $\tau_{cp} = \tau_{mul} + \tau_{add}$.

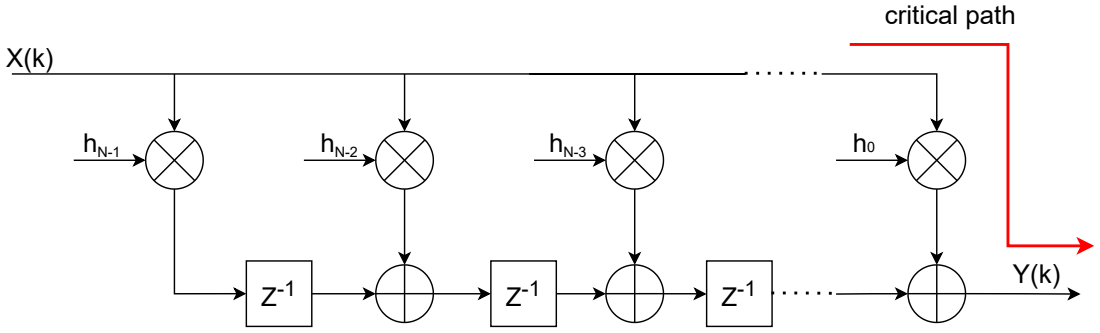


Figure 2.8: Structure of transposed FIR

The order of the filter coefficients of the transposed FIR is reversed, and the input arrives at all multipliers at the same time, so the multipliers of the transposed FIR can process in parallel, and the critical path delay is greatly reduced. However, this will lead to limitations in the input fan-out of the transposed FIR structure. When designing a filter with a large number of taps, the input fan-out must be considered. Considering the number of registers and critical path delay, the transposed FIR structure can be selected, but the direct form does not have the fan-out limitation of the transposed structure.

2.6.3 Symmetric transposed FIR

When the coefficients of the filter are symmetrical, such as an RRC filter, a symmetrical structure can be used. For symmetric FIR, only half of the coefficients will be calculated, that is, half of the multipliers resources can be saved. Converting the transposed structure shown in Fig. 2.8 to a symmetric transposed structure [30], Fig. 2.9 shows the symmetric transposed FIR structure with odd-order coefficients.

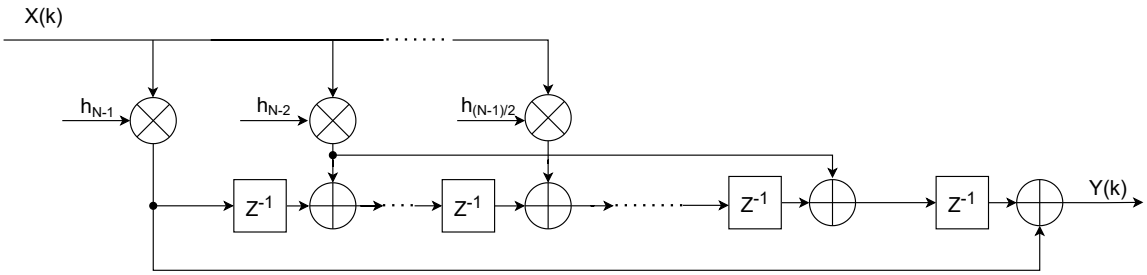


Figure 2.9: Structure of symmetric transposed FIR

2.6.4 Parallel symmetric transposed FIR

In recent years, in high-speed communication systems, the demand for operating speed is increasing. Therefore, in order to improve the throughput and operating speed of filters, parallel design is widely used. When the parallel coefficient is 2 and the taps of FIR is 5, the structure of Fig. 2.9 is converted to the structure shown in Fig. 2.10 [31]. The blue line and the green line respectively represent one output signal.

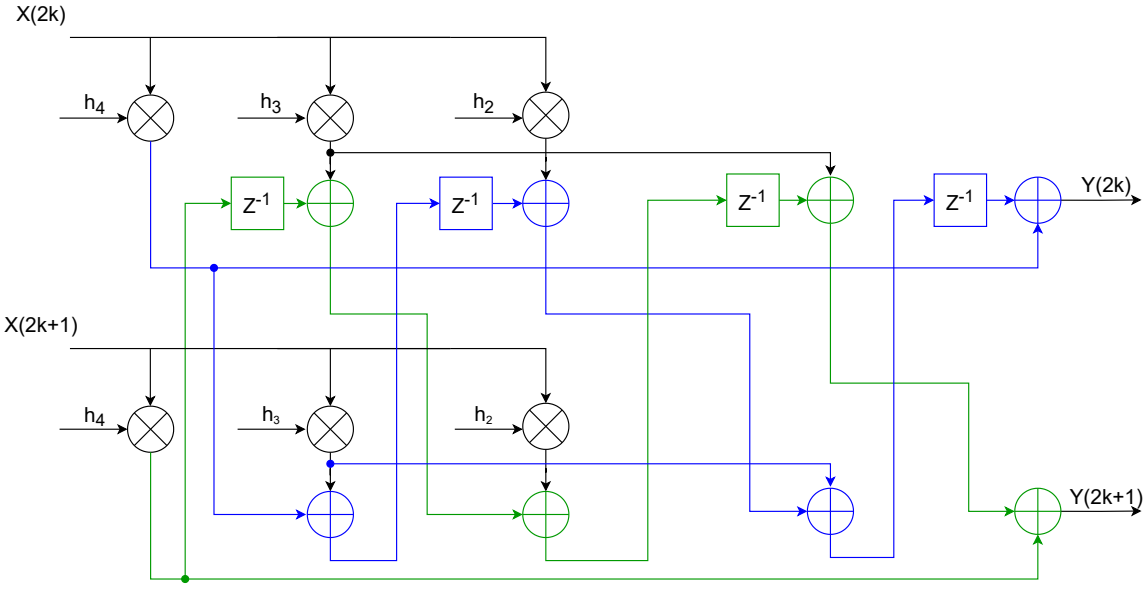


Figure 2.10: Structure of parallel symmetric transposed FIR

2.7 CHOICE

The Chalmers optical fiber channel emulator (CHOICE) is a design-and-analysis environment developed for emulating the fiber impairments such as AWGN and phase noise [15]. CHOICE is developed based on commonly used MATLAB-HDL co-simulation method. The input data are generated in MATLAB and simulated in a hardware model, then send back to MATLAB. This method is accurate but slow. The FPGA based CHOICE environment can significantly reduce the runtime of low-BER simulations compared to MATLAB-VHDL co-simulations. It implements a pseudo-random number generator for signal source, a modulator, AWGN and phase-noise generators, a de-modulator and an error counter. However, the current version of CHOICE can only emulate single-polarization systems.

3
Methodology

The development flow of this project can be divided into two parts: MATLAB design and VHDL development. This chapter explains the work flow in detail. As shown in Fig. 3.1, in the beginning we implement PMD and equalizer or de-multiplexer model in MATLAB. After successful implementation, we convert the MATLAB model of PMD to VHDL model, and perform joint simulation with the equalizer or de-multiplexer MATLAB model. Finally, we convert the equalizer or de-multiplexer model to a VHDL model, integrate this with PMD VHDL model to complete the final digital model, and analyze the influence of different PMD factors on the equalizer or de-multiplexer compensation capability and power consumption.

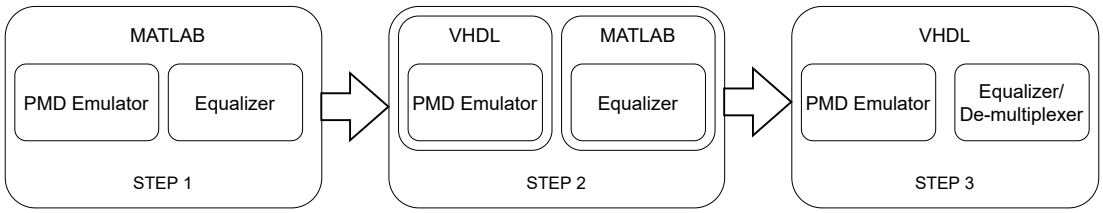


Figure 3.1: Work flow

3.1 MATLAB simulation system set up

Since MATLAB is always a productive way for developing and simulating systems, we choose to start with building a system model in MATLAB. We first implement the two-channel transmitter by expanding the MATLAB model of CHOICE [15] provided by Erik Börjeson, and generate two-channel modulated signals, and then implement the PMD digital model in MATLAB through the waveplate mathematical model in section 2.3.2. The trade-off of conciseness and reality needs to be carefully considered since the MATLAB algorithmic model of PMD will be developed into VHDL codes.

After the waveplate model is successfully implemented, we then simplify the PMD mathematical model. By dividing the PMD model into a delay part and a rotation part, the PMD model is moved from the frequency domain to the time domain. The delay part is implemented by Lagrange fractional delay filter mathematical model which is introduced in section 2.4. After successfully implementing the PMD model in the time domain, we explore different scenarios where we change the number of fiber sections, the fractional delay (which is the same for all sections) and the rota-

tion angle (which may be different for all sections). The results we obtain from our implementation of a PMD model are then compared with the results we got from using the reference waveplate model.

Finally, the PMD data with different delays and different rotation angles are directly passed through the demodulation function, and then compared with the data generated by the transmitter to obtain the bit error rate(BER). We also add the equalizer to compensate PMD before demodulation, and compare the corresponding BER with the BER without equalizer, we can see the compensation effect of equalizer. The MATLAB design part ends up with a complete verification of the 2-polarization system including PMD emulator and equalizer.

3.2 MATLAB-VHDL codesign of PMD emulator

Since the main target platforms are FPGA, we need to eventually convert the whole system from MATLAB to VHDL. For this purpose we use MATLAB-HDL cosimulation in this and the following sections. Vivado, which is a software suite developed by Xilinx for VHDL compilation, simulation and FPGA implementation, is used for FPGA designs in this project. We first focus on developing the PMD emulator since this module is our main focus.

The first thing that needs to be realized is the fiber optic transmitter which generates two-channel QPSK or 16QAM encoded data. Then the input signal is upsampled and pulse-shaped by the RRC filter, and the rotation and delay are added to the output signals of the two channels in sequence. The more the number of fiber sections, the more rotation and delay are added.

Finally, the data generated by VHDL simulation is output to MATLAB, and the difference between the ideal data generated by MATLAB and the fixed point data generated by VHDL is compared. The focus of this stage is to implement the VHDL model of the Lagrange fractional delay filter. After the corresponding FIR coefficients are generated, the FIR VHDL code is written to finally realize the delay module.

When the basic PMD VHDL model is implemented, for a more reasonable use of resources, we need to optimize this VHDL model, such as reduce the FPGA resource utilization by reducing the number of data bits, reducing the use of LUTs and reduce critical paths by parallellizing the FIR filter. After the PMD basic model is implemented, we can move to the next stage to implement the equalizer or de-multiplexer's VHDL model.

3.3 2-polarization channel with equalizer completion

Once the hardware transmitter and PMD emulator works well, we start to realize the conversion of equalizer from MATLAB to VHDL. Because we have implemented equalizer MATLAB model, when the VHDL model is implemented, we can use the same method: import the data generated by VHDL into MATLAB, and then compare it with the ideal data generated in MATLAB to get the corresponding difference, which is the accuracy of the VHDL model.

After the equalizer model is successfully implemented, the previous transmitter and PMD emulator modules are integrated to optimize the code. Finally, we analyze the performance and convergence time of equalizer under different delays and different rotation angles. Because of fixed point representations and approximations, the BER of VHDL simulations, which is calculated by dividing the amounts of errors by the amounts of transmitted bits, will be higher than for ideal MATLAB ones and this means that the algorithm performance will decrease. It's hard to avoid this in the equalizer since DSP has limited resolution, but we work to minimize the errors introduced by other parts, such as the PMD emulator.

4

PMD emulator and CMA equalizer implementation

This chapter first describes in detail the implementation and optimization of the PMD emulator in MATLAB, and how we convert the software implementation of MATLAB to the hardware implementation of Vivado. Then we show results from an evaluation of how well our hardware implementations in Vivado correspond to the ideal models in MATLAB. After that, the implementation of CMA equalizer in Vivado is introduced.

4.1 PMD emulator MATLAB design

4.1.1 Original waveplate model setup

First, we implement in MATLAB the waveplate model which is introduced in section 2.3.2, using the structure shown in Fig. 4.1. Since the waveplate model is in the frequency domain, the data streams are converted to frequency domain by applying FFT first. After a multiplication with rotation matrix and birefringence transmission matrix, the data streams are converted back to the time domain by applying IFFT.

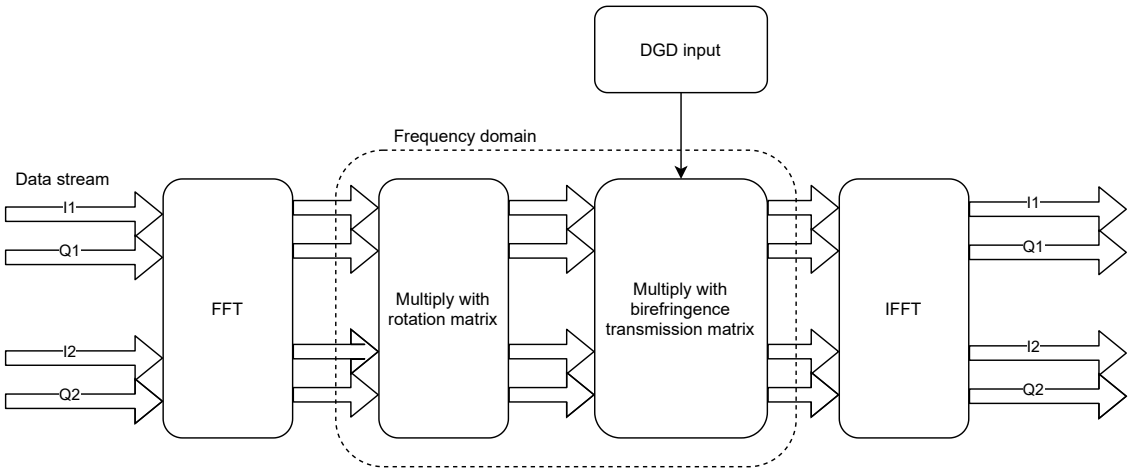


Figure 4.1: Structure of original waveplate model

The most important benefit of this model is that the waveplate model of different sections could be easily combined by using a simple matrix multiplication in time domain. The resulting matrix is the transmission matrix of the whole fiber. However,

this model is unsuitable to be applied to our system for several reasons, one of which is that the implementation of Fourier transformations in real-time hardware will increase resource utilization and system complexity. Also, the multiplication of transformed input data in frequency domain assumes the data stream is the cyclic extension of input data. Although it is acceptable in theoretical analysis, the assumption will introduce significant errors in a practical real-time system.

4.1.2 Optimized waveplate model setup

Because of the drawbacks of the original waveplate model, we want to keep the data stream in the time domain to make the system suitable for real-time implementation. One way of doing this is to, as Fig. 4.2 shows, move the rotation matrix to the time domain and use the IFFT results of birefringence transmission matrix as coefficients of the FIR filter. This method moves the frequency-domain part out of the critical data stream path and, in addition, allows the coefficients to be generated offline. To verify our optimizations, we carry out a MATLAB simulation where we compare our optimized waveplate implementation with the original one which was introduced in section 4.1.1.

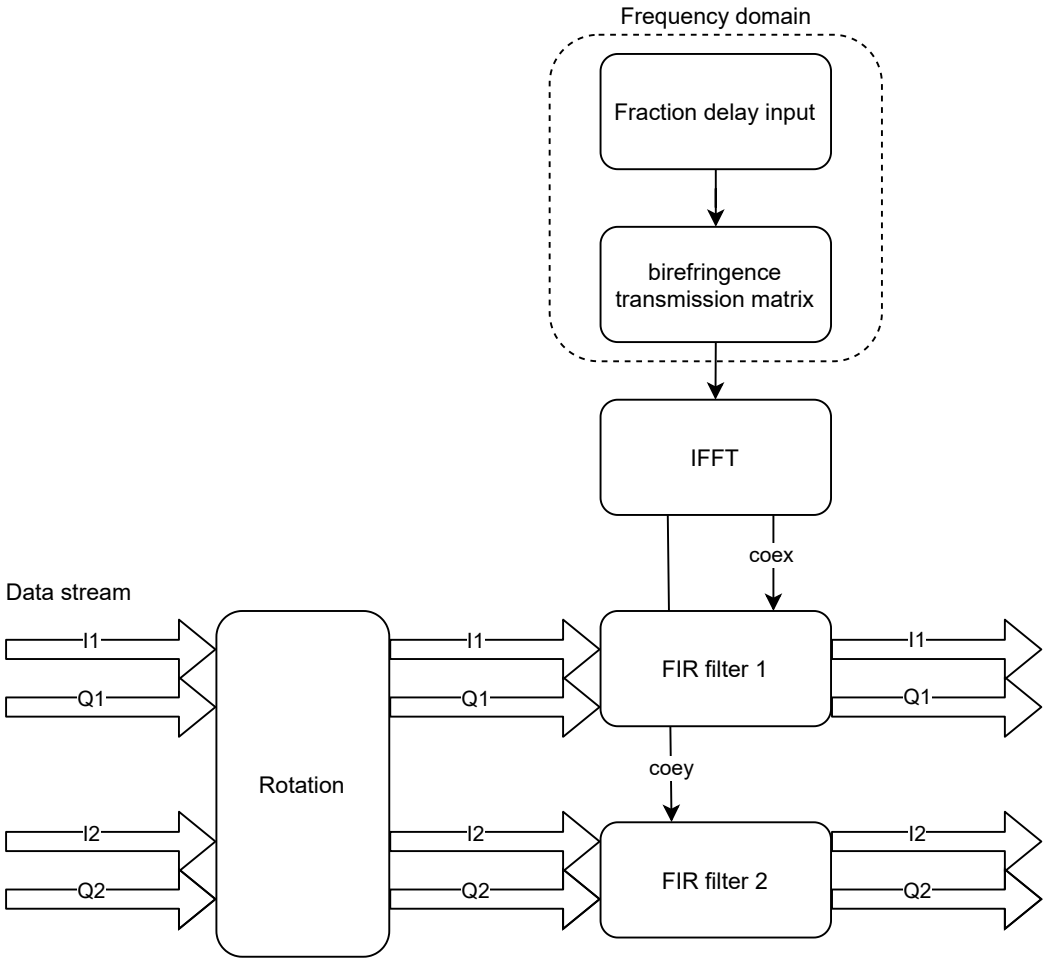


Figure 4.2: Structure of optimized waveplate model

The simulation results are shown in Fig. 4.3, which is consistent with our expected. The system symbol rate is 10Gbaud. We apply -10 and 10 ps DGD to different polarization signals respectively so DGD is equal to 20 ps, which is 1/5 of the sample period. The rotation angle is 0.4π.

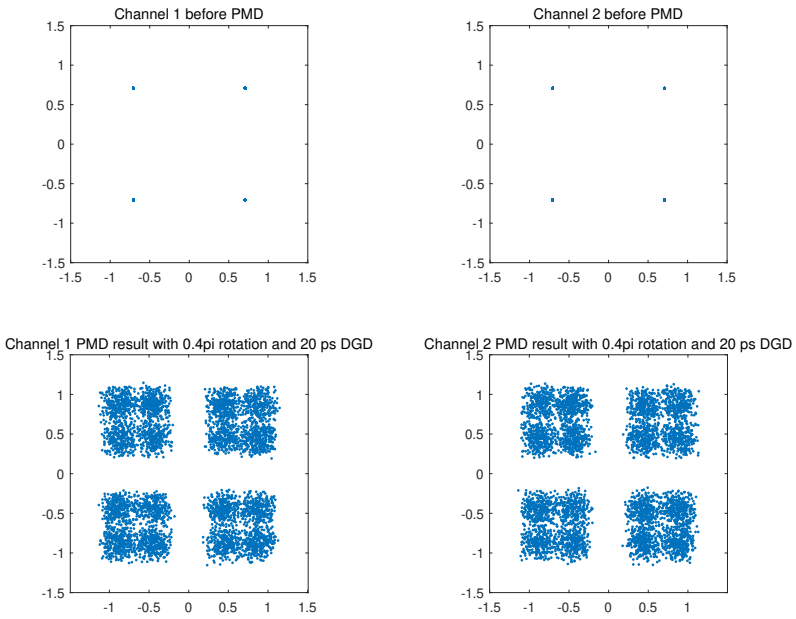


Figure 4.3: Simulation results of optimized waveplate model

4.1.3 Fractional delay filter based waveplate model setup

From the previous sections 4.1.1 and 4.1.2 we know that the essence of waveplate model is a combination of rotation and delay. The delay should be a fraction of the sample period. So fractional delay filter is a perfect substitute of original frequency-domain birefringence transmission matrix. The fractional delay filter could delay the signal by a fraction of the sample period. In this case, the coefficient generating equation is transformed to Eq. (4.1) [22].

$$h(n) = \prod_{k=0,k\neq n}^{N-1} \frac{D_0 + u - k}{n - k}, n = 0, 1, 2, 3, \ldots, N-1 \tag{4.1}$$

Here, u is calculated from the target DGD using Eq. (4.2), where τ_k is the DGD of one section and T_s is the sample period of the system.

$$u = \frac{\tau_k/2}{T_s} \tag{4.2}$$

The structure of this model is shown in Fig. 4.4.

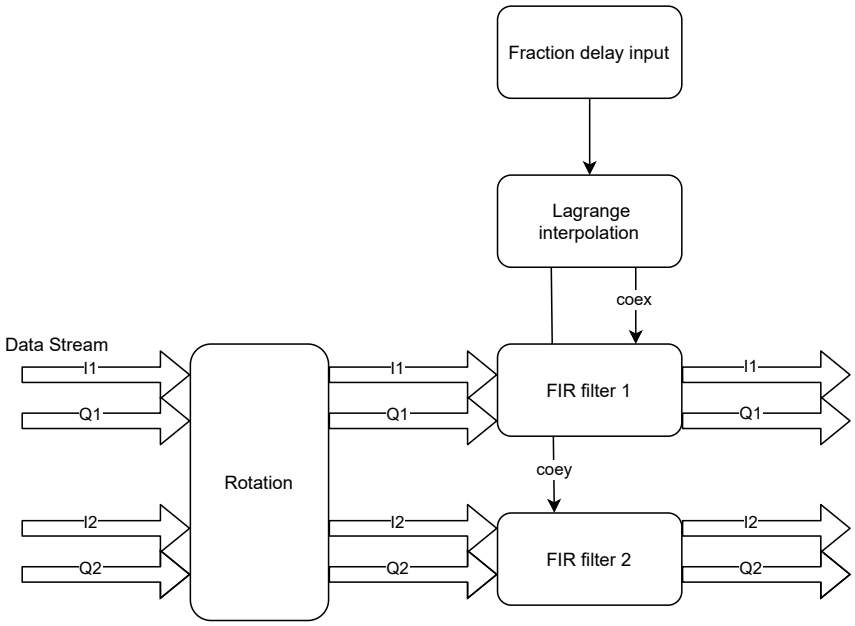


Figure 4.4: Structure of fractional delay based waveplate model

The simulation results are shown in Fig. 4.5. The simulation was ran with the same parameters as the previous section, which means DGD is set to 20 ps and rotation angle is 0.4π. Compared with previous models, the coefficients generation part is moved from frequency domain, which means that the whole system is kept in time domain now, which is friendly to our real-time system. Also, fractional delay filter still has high accuracy with low tap numbers, which will decrease the calculation amount and ease resource utilization in hardware implementation.

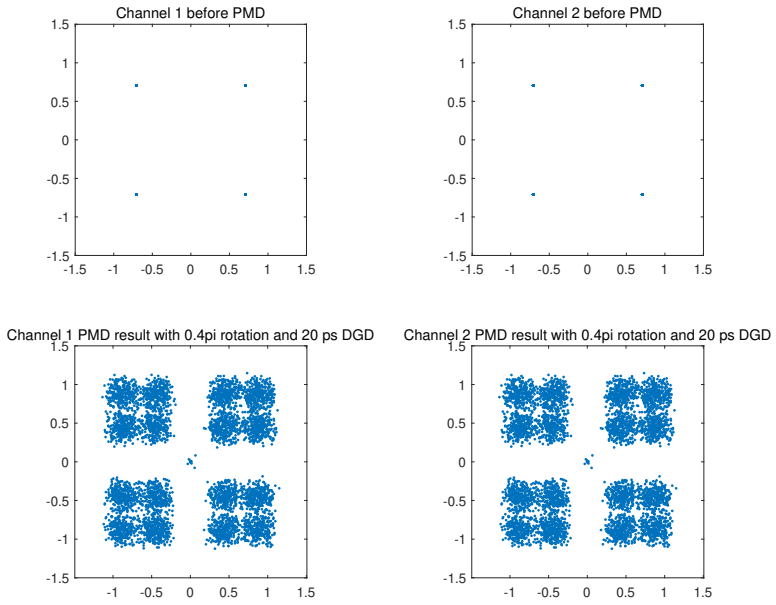


Figure 4.5: Simulation results of fractional delay based waveplate model

4.2 PMD emulator VHDL design

After performing software functional verification for PMD in MATLAB, we begin to implement the hardware design of the PMD emulator and the two-channel transmitter required to support two different polarizations. The hardware structure diagram of the PMD emulator is shown in Fig. 4.6.

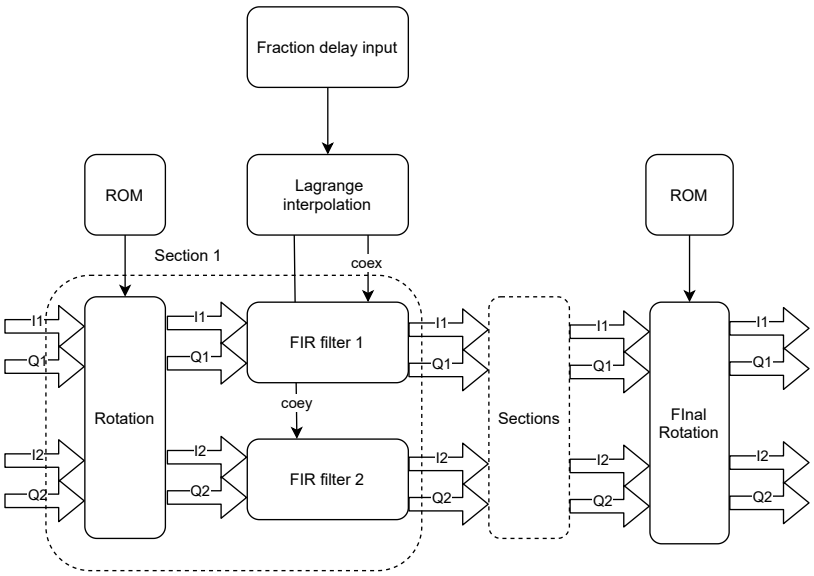


Figure 4.6: PMD emulator structure in VHDL

4.2.1 Transmitter

The structure of the transmitter is shown in Fig. 4.7. Here, the signal output of two channels is generated by expanding the CHOICE system [15]. The RNG component generates the pseudo-random bit sequence [32] corresponding to the modulation format. Then the modulator component is used to do QPSK or 16QAM signal modulation. Finally output the real part and imaginary part of the two optical fiber signals and the valid signal.

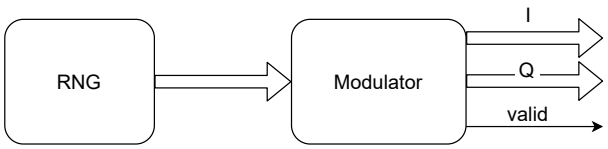


Figure 4.7: Transmitter structure in VHDL

4.2.2 Upsampling and pulse shaping

The structure of upsampling component is shown in Fig. 4.8. After the transmitter outputs valid signals and valid I and Q signals, upsampling component implements upsampling and RRC filtering of the modulated signal through the parallel symmetric transposed RRC FIR introduced in section 2.6.4. When the input signal is 12-bit

data, because of the parallel structure, the output data is 24 bits. The upper 12 bits are valid data, and the lower 12-bit data is the data generated by RRC filtering after interpolation of 0.

Figure 4.8: Upsampling structure in VHDL

4.2.3 Rotation

After the data has passed through the upsampling and RRC filter, it needs to experience the rotation introduced in Eq. (2.6). The structure of rotation component is shown in Fig. 4.9, parallel input and parallel output. The pre-calculated sine and cosine values are stored through Vivado's ROM IP. Then we use the input *rotation angle* as the ROM address to get the corresponding value output. The step size of the *rotation angle* is 1°, and the total rotation range is 0 to 45°. When there are N fiber sections, there are corresponding $N + 1$ rotation components. Each fiber section has a rotation component, and there is an additional rotation component at the end of the fiber section.

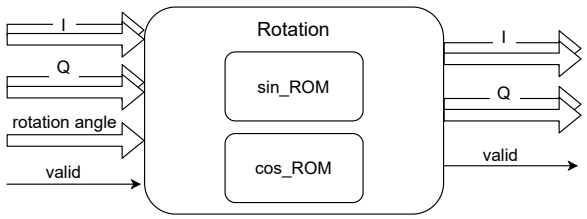


Figure 4.9: Rotation structure in VHDL

When there are 10 fiber sections, the rotation setting is as shown in the Fig. 4.10. Here, we set the first, center and last section to variable rotation, with rotation frequency f_1, f_2 and f_3 respectively, and set the other rotation sections to fixed rotation angles. The first *rotation angle* is related to the transmitter, the center *rotation angle* is related to *DGD*, and the last *rotation angle* is related to the receiver. Therefore, we pay more attention to the *rotation angle* in the center section.

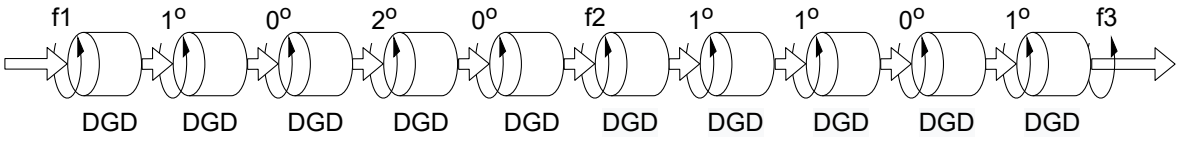


Figure 4.10: Rotation with 10 fiber sections

4.2.4 Fractional delay filter

After adding rotation to the two channels, we need to add DGD delay to each section, which is realized by the fractional delay filter, whose structure is shown in Fig. 4.11. The filter coefficients of the two channels are obtained through the Lagrange interpolation component and the fractional delay filter uses the parallel direct FIR structure. Each FIR filter can process the signal of the real and imaginary parts of one channel, so each fiber section needs two FIR filters.

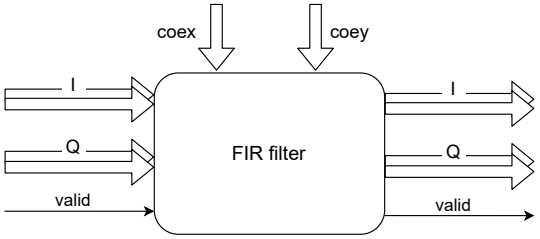


Figure 4.11: Fractional delay FIR structure in VHDL

4.2.5 Lagrange interpolation

The hardware implementation of Lagrange interpolation is realized by converting Eq. (2.7) into VHDL language, as shown in Fig. 4.12. We calculate the fifth-order filter coefficients *coex* and *coey* corresponding to the two channels by inputting the value of fractional delay. The range of the input *FDdelay* is greater than 0 and less than 1. The finish signal represents the completion of the coefficient calculation. The generated 5-tap FIR filter coefficients are stored in registers. When the value of the fractional delay changes, the output coefficients will also change, and finally output to the FIR component to filter the input signals. When each section has the same delay, only one Lagrange interpolation needs to be used to generate the filter coefficients of the two channels. And when the delay of each section is different, multiple Lagrange interpolation components need to be used. Normally, all sections have same DGD value.

Figure 4.12: Lagrange interpolation structure in VHDL

4.3 CMA equalizer implementation

We first introduce the basic flow of CMA algorithm. Firstly we set the tap number and step size, also determine the R^2 value which is introduced in Eq. (2.11) according to the specific modulation signal type. Then we initialize the coefficients of filters. After that the errors are calculated and the coefficients are adjusted dynamically according to errors and input signals.

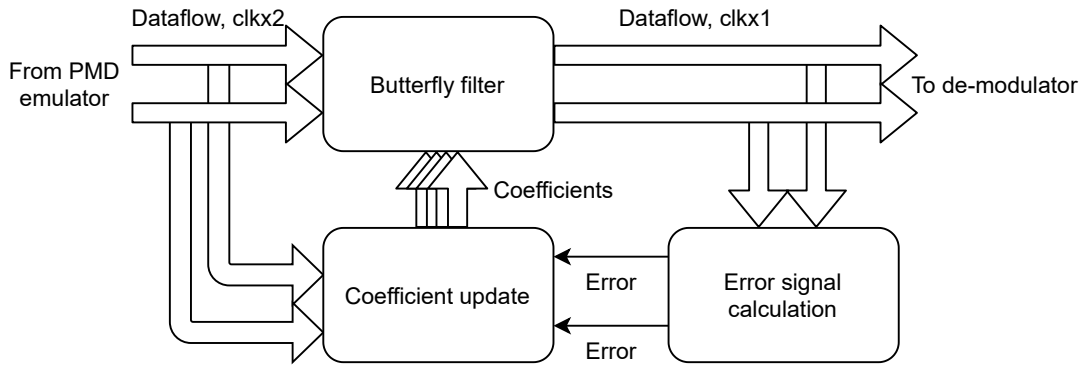


Figure 4.13: Equalizer structure in VHDL

Fig. 4.13 shows the hardware structure of CMA equalizer; more details on the implementation considerations for this type of equalizer can be found in [33]. The signals of two channels coming from PMD emulator are used as the input of butterfly filter and are stored as input of coefficient update modules. The structure of butterfly filter is shown in Fig. 4.14. It contains four complex number parallel transposed FIR filters and the signal is downsampled after it. After butterfly filter, the errors of both channels are calculated by error calculation module by comparing the expected modulus and the amplitudes of output signals. The errors are used for updating the coefficients used in butterfly filter by coefficient update module. The coefficient update modules delay the input signals for synchronizing the the signals to corresponding errors since delay exists in filter and error calculation modules. Then they update the coefficients used by butterfly filter dynamically. Since the core idea of CMA is updating tap coefficients according to the errors between the amplitudes of signals and the ideal modulus, the basic CMA equalizers are only efficient to phase-shift keying formats such as binary phase-shift keying (BPSK) and QPSK. Since our system is mainly tested in QPSK format in this project, we implement a CMA equalizer in VHDL. Note that for 16QAM, we design a modified constant modulus algorithm (MCMA) equalizer in MATLAB for system simulations. In MCMA, the coordinates of received 16QAM signals are transformed for calculating errors for using CMA [34]. The MCMA will not be further discussed in this report.

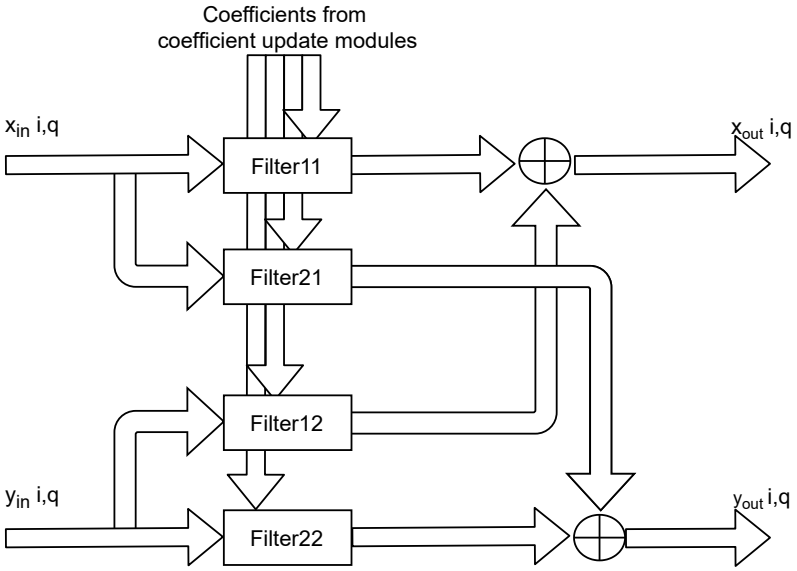


Figure 4.14: Butterfly filter structure in VHDL

Fig. 4.15 shows that the equalizer can significantly reduce the BER. The convergence speed mainly depends on the severity of signal impairment. With larger PMD the system converges slower. The convergence speed doesn't change much when equalizer taps changes, but with more taps the equalizing performance increases.

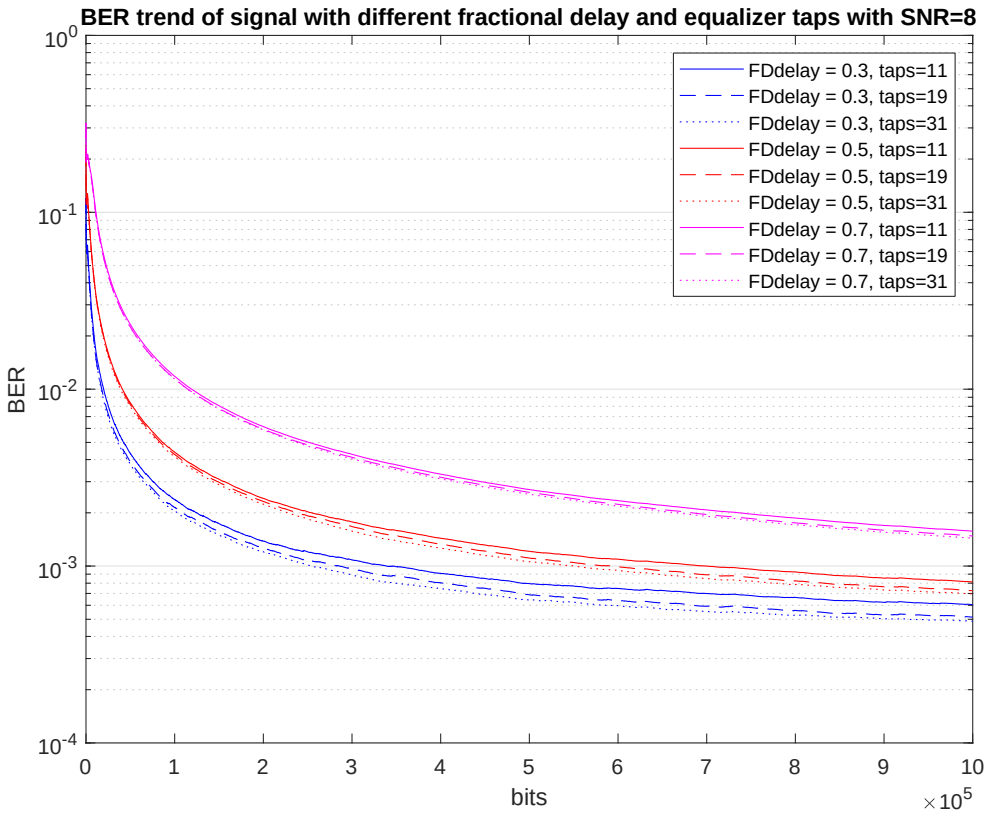


Figure 4.15: BER comparison with different taps and different fractional delay

Table 4.1 shows the resource utilization of equalizer with different taps after implementation. The target FPGA platform is Xilinx VC709 FPGA. We can see that the taps of equalizer have huge impact on resource utilization, especially DSP utilization. The reason for this is that the increase of tap number will directly increase the DSP needed in coefficient update modules, which are calculation intensive.

Table 4.1: Resource utilization of equalizer for different number of taps

Equalizer taps	LUT utilization	DSP utilization	Slice registers
11	2116/0.49%	380/10.56%	7455/0.86%
19	3588/0.83%	652/18.11%	12940/1.49%
31	5796/4.93%	1060/27.80%	20799/2.47%

5

Results

This chapter first describes the simulation and emulation results of PMD emulator. Then we will describe the system including PMD emulator and CMA equalizer. Finally we will give simulation results of system in which we use parameters that are inspired by practical fiber-optic communication systems.

5.1 PMD Vivado simulation results

The complete structure of the system we used to test the PMD emulator is shown in Fig. 5.1. Since we want to show the impairments introduced by PMD directly, the equalizer is not included in this system, but the RRC downsampling and de-modulator components are added after the PMD emulator.

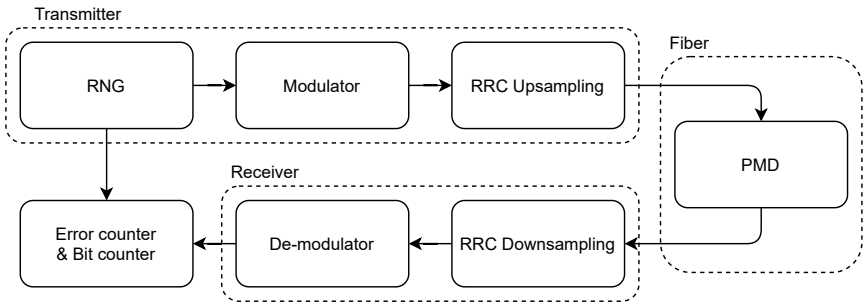


Figure 5.1: Structure of PMD test system with demodulator and error counter

In order to compare the accuracy of the VHDL model with MATLAB, the output signals of RRC downsample component and the modulated signals are output as a .vec file through Vivado, and then read through MATLAB to compare the simulation results of MATLAB and VHDL. The simulation process is shown in Fig. 5.2. The modulated data are processed in MATLAB the same way as in Vivado, and then compared with the simulation results of Vivado.

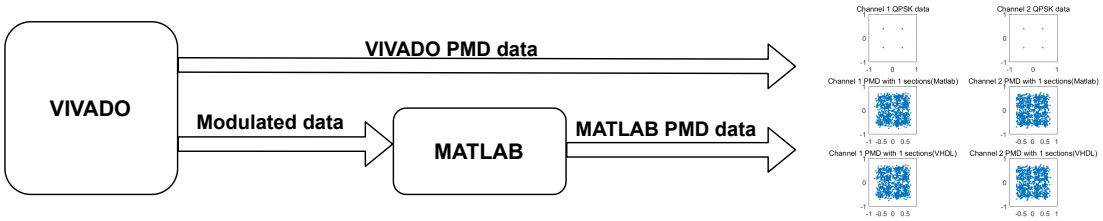
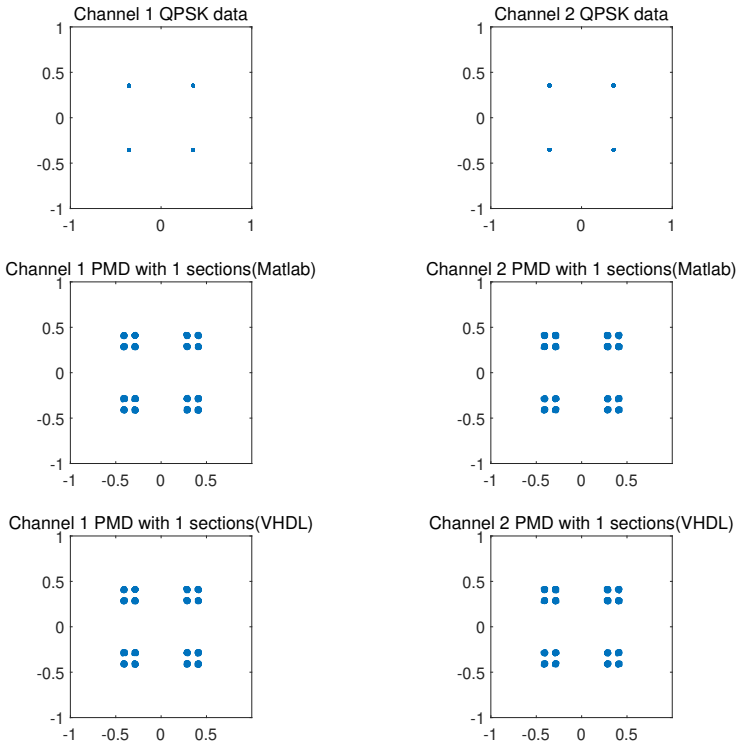
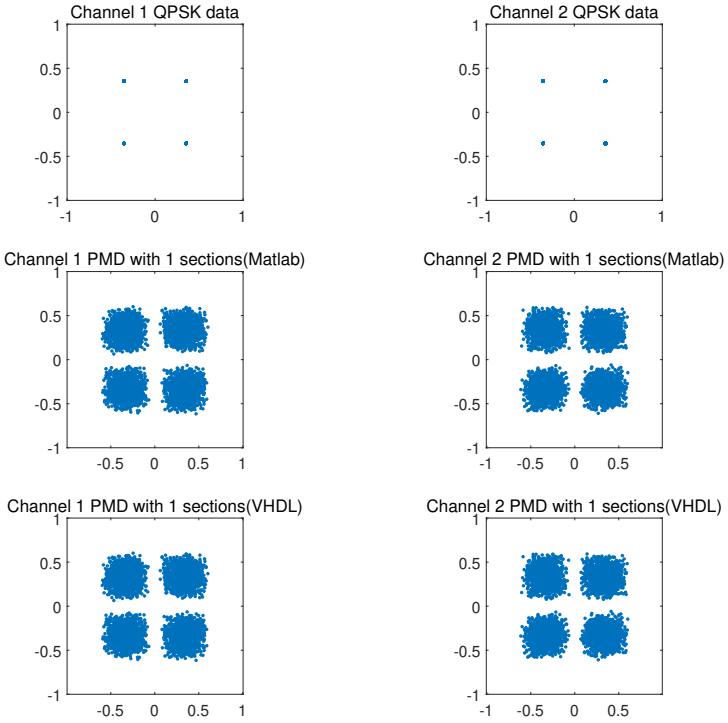


Figure 5.2: Simulation process between MATLAB and Vivado

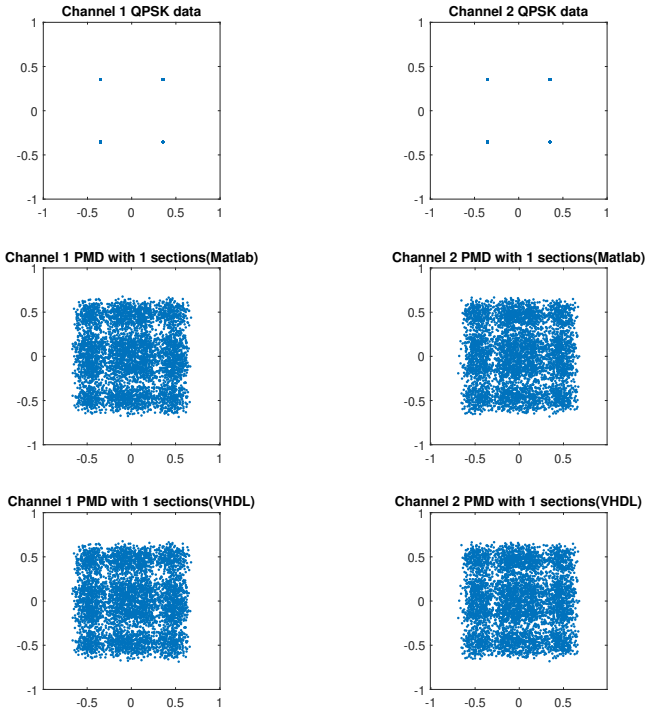
In Fig. 5.3, the first row of the picture is original modulated data generated by transmitter component, the second row is the simulation result of MATLAB, and the third row is the simulation result of VHDL. First, we test the PMD VHDL model with only one fiber section, and collect several results by changing different parameters. When the *FDdelay* or *rotation angle* increase, the influence of PMD on the input signal increases as Fig. 5.3 shows. The difference between MATLAB and VHDL as shown in Fig. 5.4 is less than 1.5‰.



(a) *FDdelay*= 0.03, total *rotation angle* = 10°



(b) *FDdelay* = 0.3, total *rotation angle* = 10°



(c) *FDdelay* = 0.3, total *rotation angle* = 30°

Figure 5.3: QPSK simulation results of MATLAB and Vivado with one section

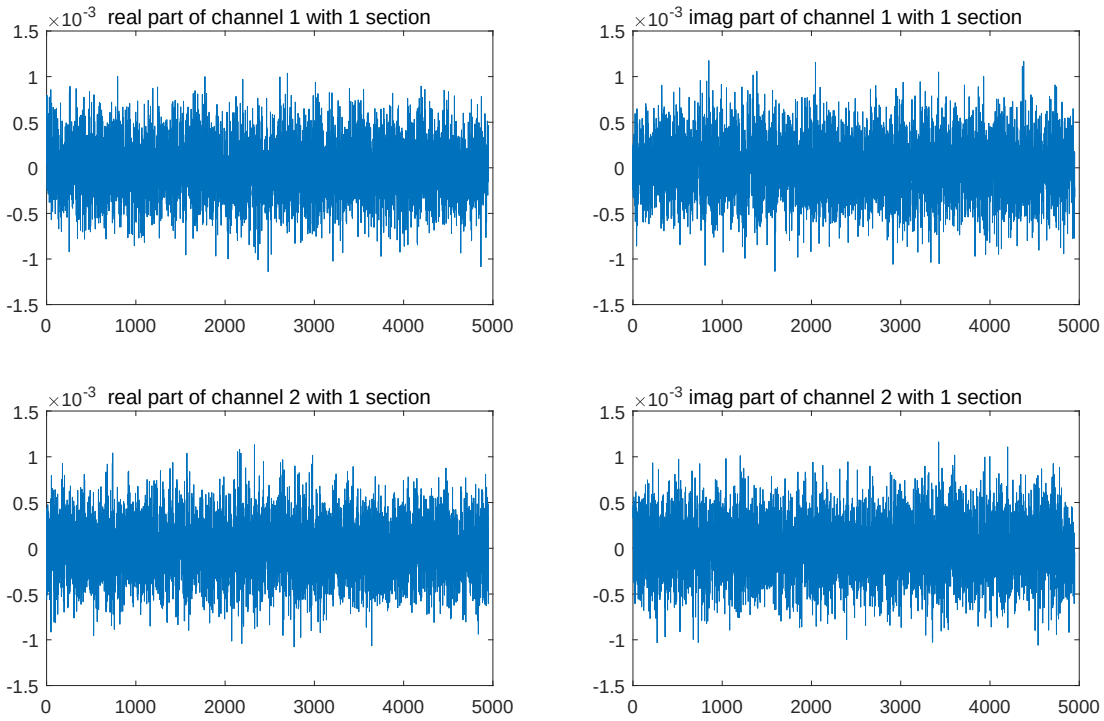
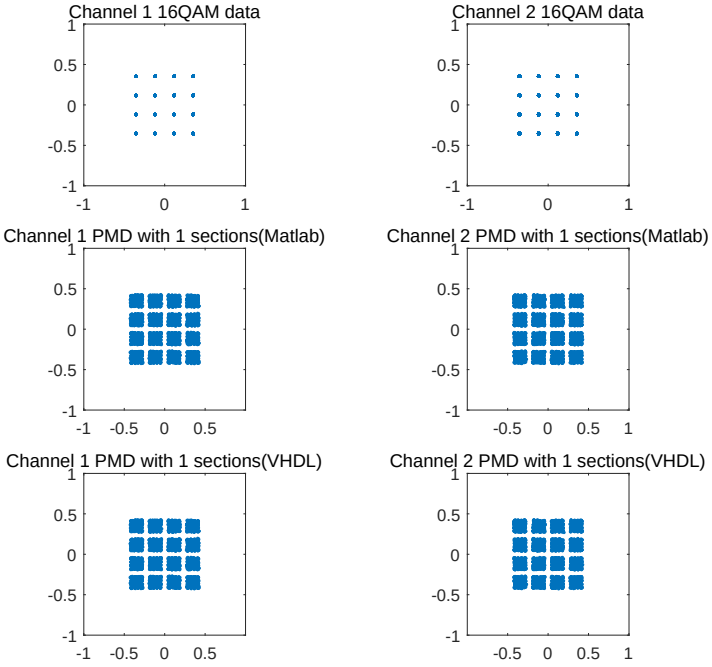
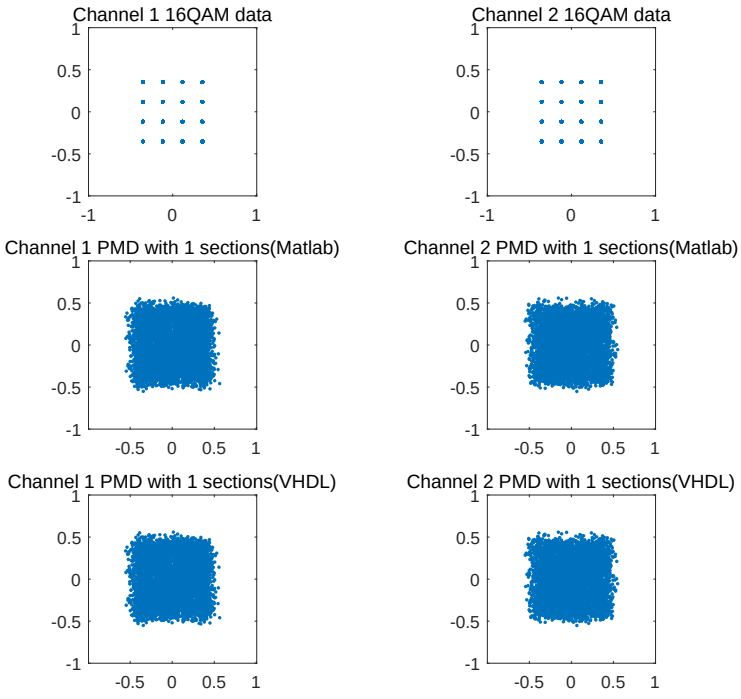


Figure 5.4: Difference between MATLAB and Vivado with one section of QPSK modulation

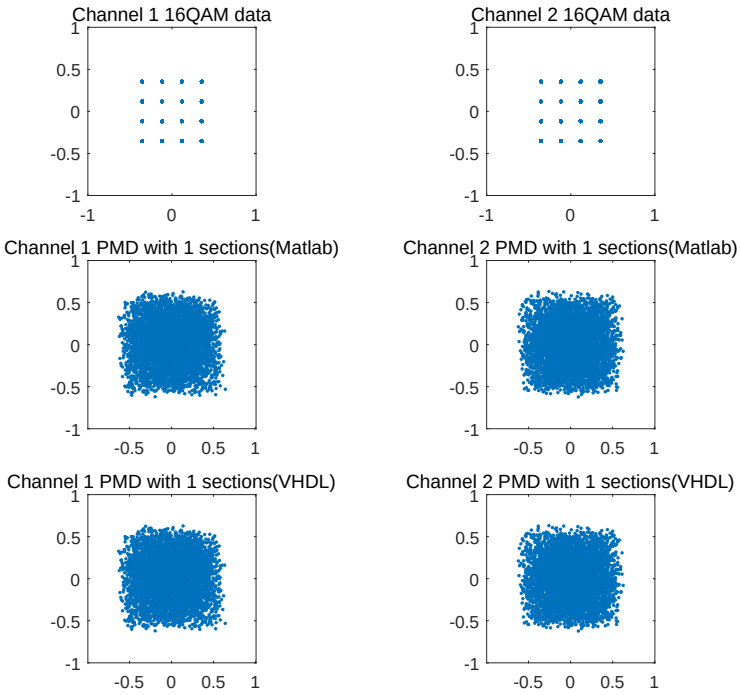
Next we change the modulation format from QPSK to 16QAM, the parameter settings are the same as for the QPSK. The simulation result is shown in Fig. 5.5. Compared with the simulation results of QPSK, PMD has a greater impact on 16QAM. The difference between MATLAB and VHDL simulation results is shown in Fig. 5.6, which is similar to the QPSK format, less than 1.5‰. The simulation results of 10 sections QPSK and 16QAM are in appendix A.



(a) *FDdelay* = 0.03, total *rotation angle* = 10°



(b) *FDdelay* = 0.3, total *rotation angle* = 10°



(c) *FDdelay* = 0.3, total *rotation angle* = 30°

Figure 5.5: 16QAM simulation results of MATLAB and Vivado with one section

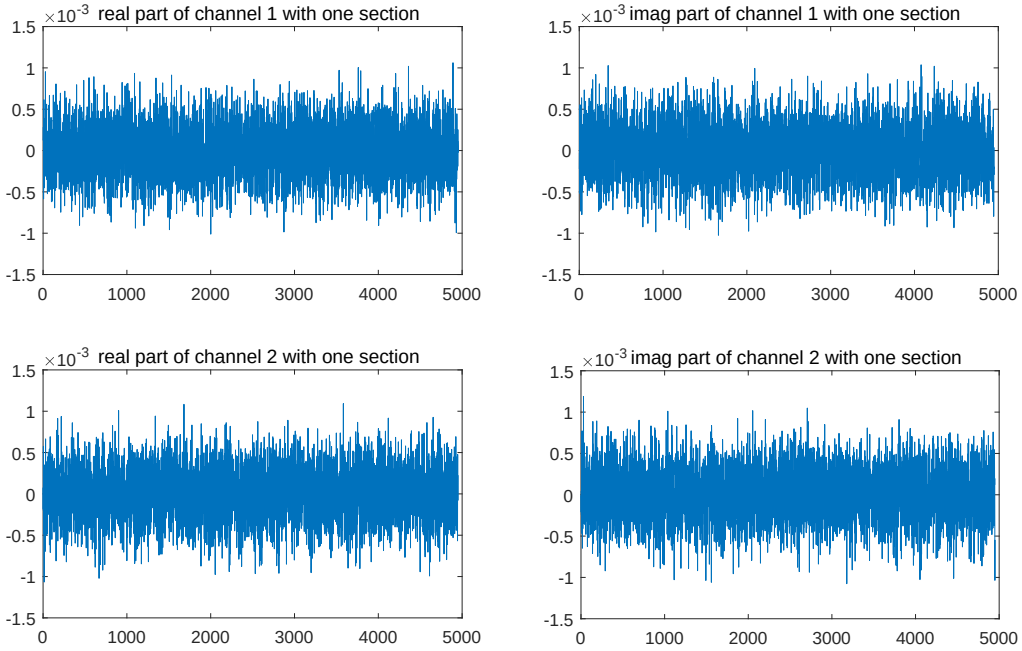


Figure 5.6: 16QAM difference between MATLAB and Vivado with one section

In order to see the influence of PMD on the input signal more intuitively, the demodulator and error counter components are added after the RRC downsampling

component. The current VHDL structure is shown in Fig. 5.1. By delaying the output of the original data generated by RNG and comparing it with the demodulated PMD signal, the number of errors can be calculated.

Using one fiber section, we set *FDdelay* to 0.03, and the *rotation angle* increases from 0 to 45° according to a certain frequency. The simulation result is shown in Fig. 5.7. The signals *bit_cnt*1 and *bit_cnt*2 are the counters which count the amount of bits of both channels have been transmitted since beginning. The signals *error_cnt*1 and *error_cnt*2 are the counters count the amount of errors of both channels that have been detected since beginning. The signal *angle* stands for the current *rotation angle*. When the *rotation angle* increases to 41°, an error occurs. When the *rotation angle* is 45°, more errors are generated in the same time. After the *rotation angle* becomes 0° again, the number of errors no longer increases. This is because when the *rotation angle* is small, the polarization effect between the two channels is small.

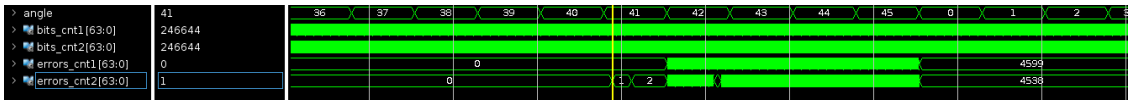


Figure 5.7: QPSK simulation result with one section and $FDdelay = 0.03$

Then we change the *FDdelay* to 0.3, and change the *rotation angle* gradually again. The Vivado simulation result is shown in Fig. 5.8. When the *rotation angle* is 13°, an error occurs. This is because when *FDdelay* increases, the influence of PMD on the signal increases, so the error will start to occur at a smaller *rotation angle*.

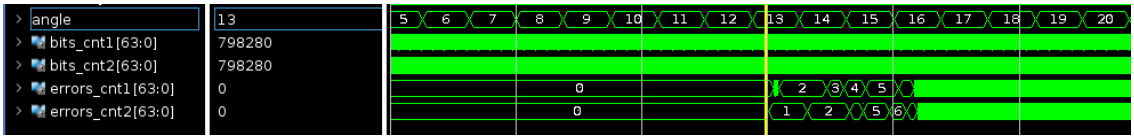
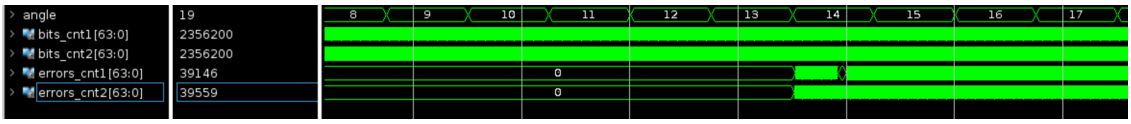


Figure 5.8: QPSK simulation result with one section and $FDdelay = 0.3$

We change the modulation format to 16QAM, set the *FDdelay* to 0.03 and 0.3 respectively, and then gradually increase the *rotation angle*. The simulation result is shown in Fig. 5.9. When *FDdelay* is 0.03, the *rotation angle* increases to 13 before an error occurs. When *FDdelay* increases to 0.3, an error occurs even if the *rotation angle* is 0. Clearly, PMD has a larger impact on 16QAM than on QPSK.



(a) *FDdelay* = 0.03

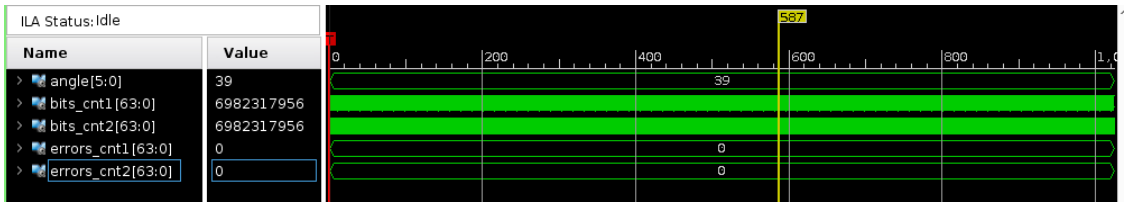
(b) *FDdelay* = 0.3

Figure 5.9: 16QAM simulation result with one section

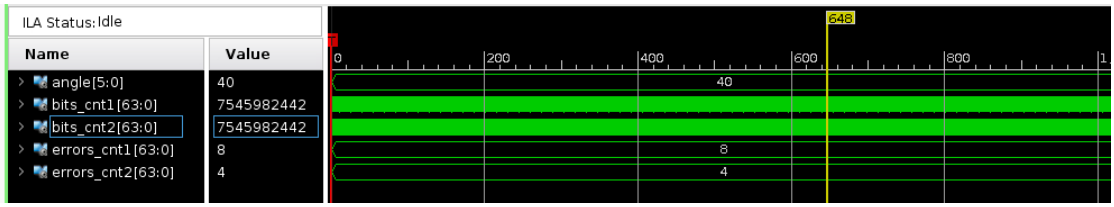
5.2 PMD real-time emulation results in FPGA

After we have successfully simulated our PMD model implementation in the Vivado simulator, Integrated Logic Analyzer(ILA) IP core, which is used to monitor the internal signal of FPGA, was added to capture the number of transmitted bits and the number of errors generated by error_counter component. We use QPSK as the modulation format of this real-time emulation and set *FDdelay* to 0.03. Then we generate bitstream, and download the code to run in FPGA.

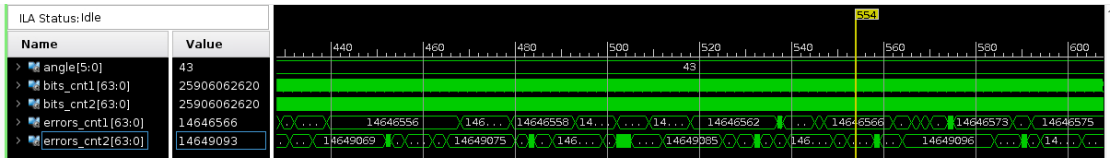
The results we observe through ILA are shown in Fig. 5.10. When the rotation angle is 39°, no errors occur; when the rotation angle is increased to 40°, some errors start to occur; but when the rotation angle is 43°, a large number of errors occur. Compared with Fig. 5.7, the error occurred in the FPGA operation earlier than the simulation result of Vivado. This is because the processing speed in FPGA is very fast, and hundreds of millions of data can be transmitted in a short period of time, which is impossible in simulation. Therefore, the simulation cannot contain all possible situations, so there will be a small difference between the simulation results and the actual results of the FPGA.



(a) *rotation angle* = 39°



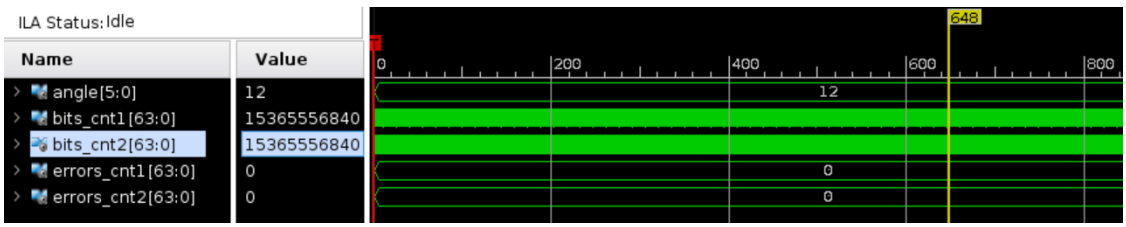
(b) *rotation angle* = 40°



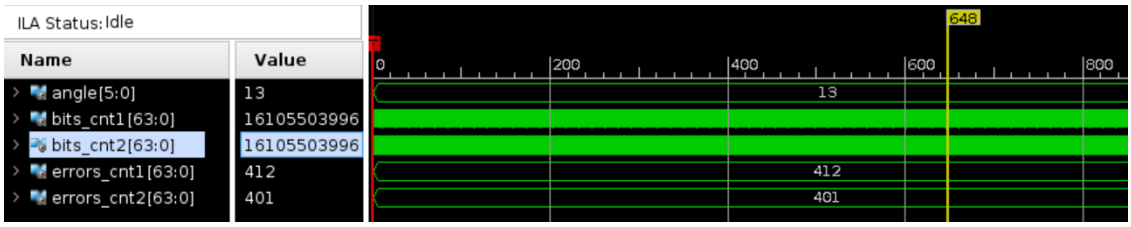
(c) *rotation angle* = 43°

Figure 5.10: QPSK ILA emulation result with one section and *FDdelay* = 0.03

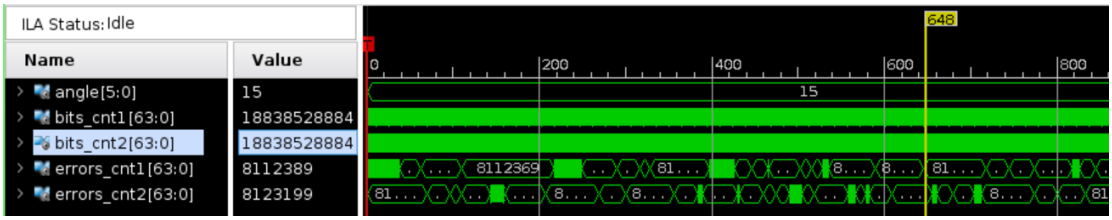
Then we changed the modulation format to 16QAM, and generated the bitstream file for 16QAM PMD emulator with one section. The emulation result for an *FDdelay* of 0.03 is shown in Fig. 5.11. The results of the 16QAM PMD emulator running in the FPGA are similar to the simulation results in Fig. 5.9. There is no error when the rotation angle is 12°, but some errors appear at 13°. Then, the larger the rotation angle is, the faster the error appears. The emulation results of QPSK and 16QAM with 0.3 *FDdelay* are shown in section A.2.



(a) *rotation angle* = 12°



(b) *rotation angle* = 13°



(c) *rotation angle* = 15°

Figure 5.11: 16QAM ILA emulation result with one section and *FDdelay* = 0.03

5.3 Results of complete system

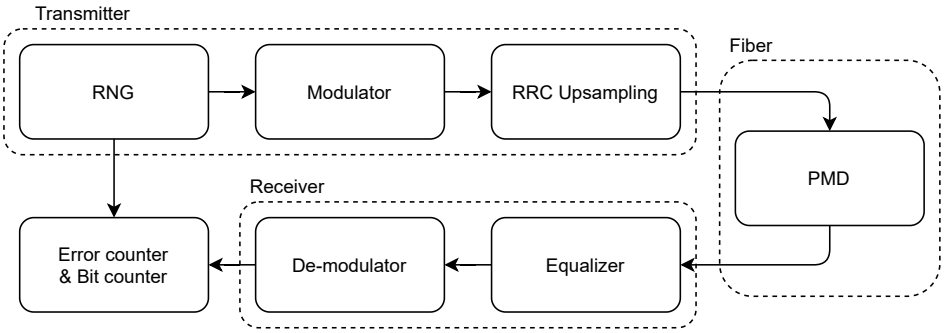


Figure 5.12: Structure of two-polarization fiber channel system including PMD emulator and equalizer

We replace the RRC downsampling module with our CMA equalizer to test if the CMA equalizer can efficiently equalize the PMD impairment introduced by our PMD module. The system structure is shown in Fig. 5.12. We start with a one-section PMD emulator whose *FDdelay* equals to 0.5 sample period and *rotation angle* equals to 35° to check if the equalizer can equalize the signal successfully. The result is shown in Fig. 5.13. From the figure we can see that the equalization errors in different channels increase at start up, then gradually reduce, and eventually become constant after the equalizer has converged. This means the impairments in this situation have been successfully equalized by equalizer.

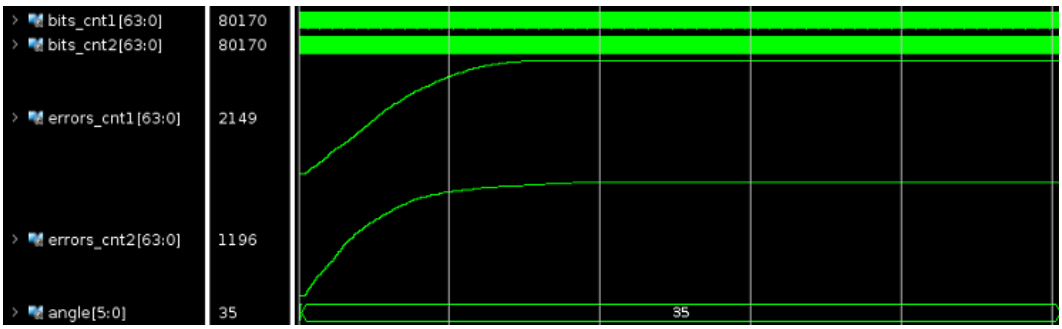


Figure 5.13: Simulation result of complete system with 0.5 *FDdelay* and 35° *rotation angle* PMD

The second scenario is that we use a one-section PMD emulator whose *FDdelay* equals to 0.3 sample period and the initial *rotation angle* equals to 0°. Then after some time we increase the *rotation angle* to 35° directly. The simulation result is shown in Fig. 5.14. We can see that the error counters start to increase when the *rotation angle* changes and increase slower as time goes by since equalizer begins to converge. The error become constant when the equalizer has equalized most of the impairments.

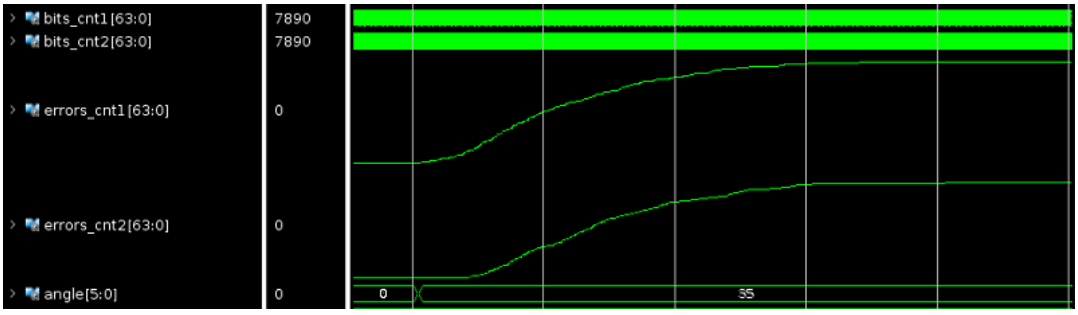


Figure 5.14: Simulation result of system with equalizer

Next we use ILA to capture the real-time signals of this system when running on FPGA. We set *FDdelay* to 0.3 and increment the *rotation angle* slowly. The amounts of errors of these two channels should start increasing after *rotation angle* reaches about 20° if no equalizer is implemented. However, as shown in Fig. 5.15, the error amounts stay zero when we have already increment *rotation angle* to 45°. This means that the equalizer can track small change of *rotation angle* perfectly.

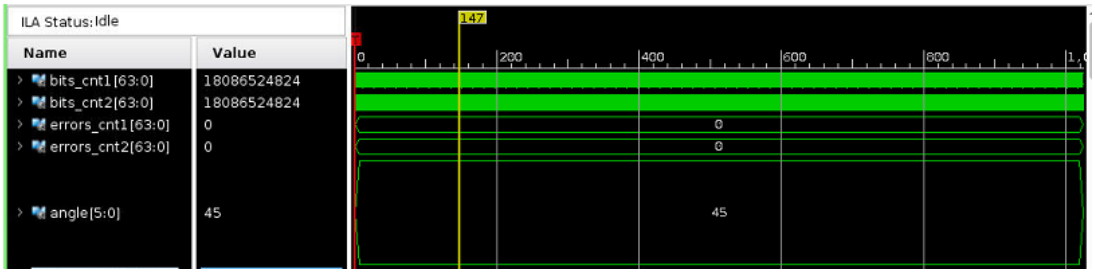


Figure 5.15: Emulation result of system with equalizer

5.4 Results of a more realistic system

To simulate and emulating a system that is similar to practical fiber channels, we set up a system whose structure is shown in Fig. 5.16. The AWGN component is added after the upsampling module for representing the white noise in fiber. The PMD emulator of this system has 10 sections, which have same *FDdelay* value but different rotation angles between them as introduced in section 4.2.4.

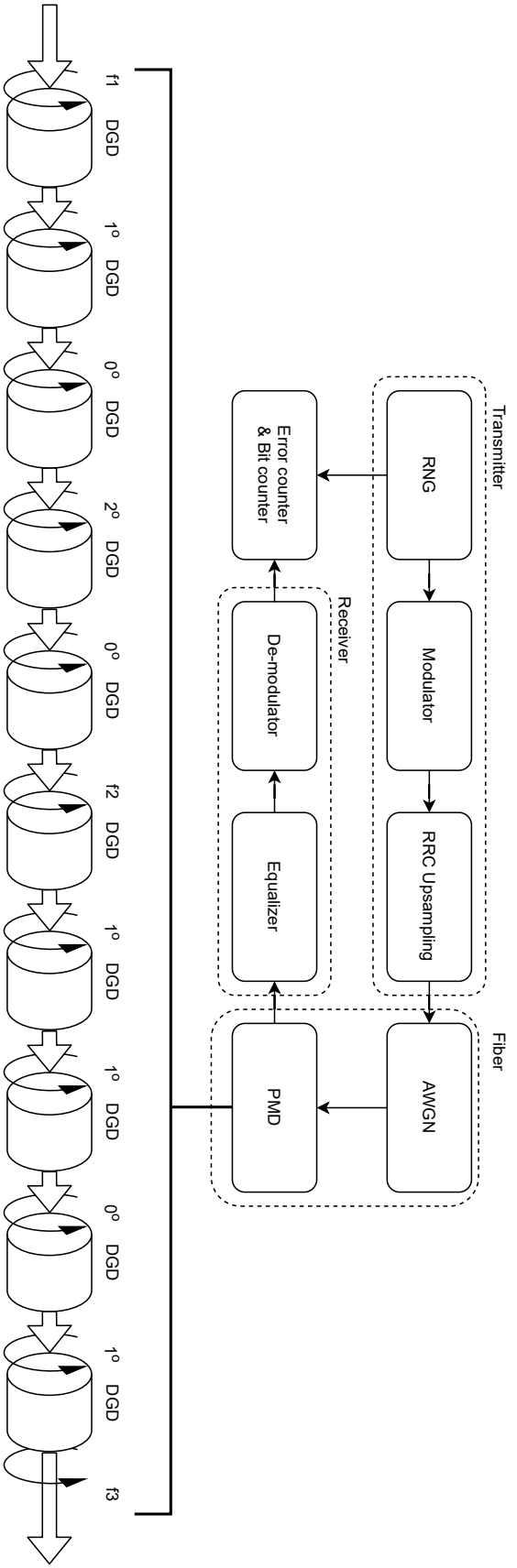


Figure 5.16: Detailed realistic system structure

Fig. 5.17 shows the simulation result of this system. We add AWGN to make the channels' SNR equal to 12 dB, and use a PMD emulator with 10 sections with same *FDdelay* equal to 0.03 sample period. The *rotation angle* of first, center and last section changes randomly. The *rotation angle* updating frequency of the center section is highest and then the last section then the first one, which are 10 kHz, 1 kHz and 100 Hz respectively. We use an 11-tap equalizer with a step size equal to 0.008 for increasing the convergence speed and make the result easy to observe. From Fig. 5.17, we can see that in the beginning, when the equalizer is converging the pulse shaping, the existence of AWGN has a negative influence on equalizing and introduce some errors. After that, the equalizer starts to compensate the impairments we introduce which is a dramatic change of the center section's *rotation angle* from 1° to 35°, and successfully equalize it after some time. Finally, the amount of errors stay relatively constant since the impairments have been mitigated.

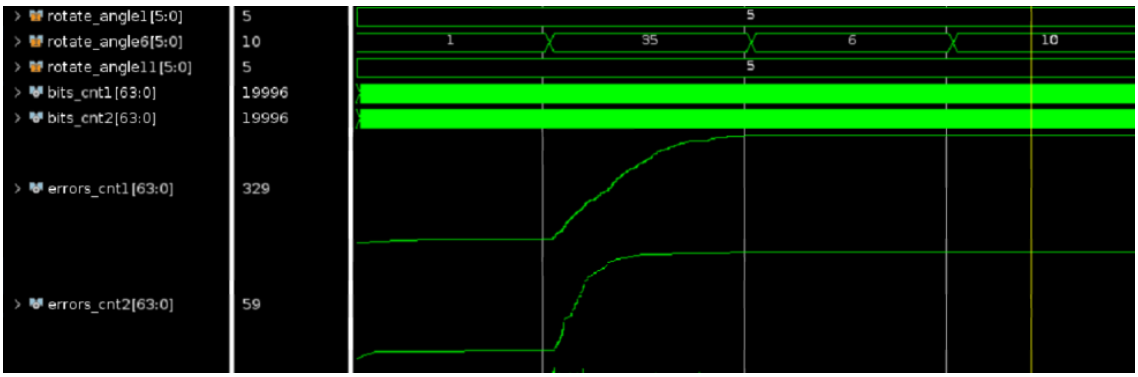


Figure 5.17: Simulation results of realistic system

Next, similar to the previous section, we use ILA to capture the real-time signals of this system when running on FPGA. In this emulation, we set the same AWGN and *FDdelay* as the previous simulation. The *rotation angle* of the first, center and last section changes randomly in the range from 0° to 15°. The updating frequencies of the *rotation angle* of those three sections are 0.3Hz, 1Hz and 0.1Hz to form a benign scenario. The step size of the equalizer is set to 0.0001 to make it stable. The emulation results are shown in Fig. 5.18. In this scenario the equalizer can converge the weak PMD easily. The amount of errors of both channels keep relatively constant, but will keep incrementing slowly due to the white noise.

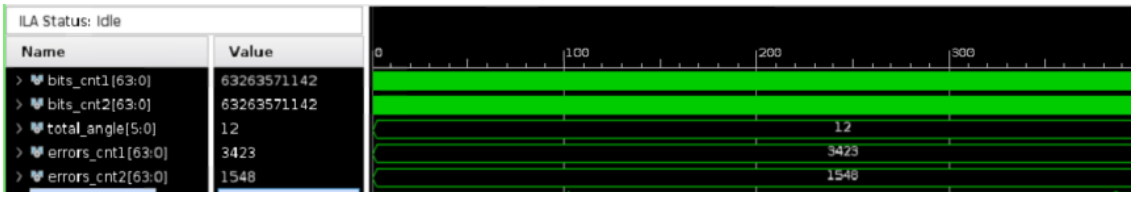


Figure 5.18: Emulation results of realistic system in a benign scenario

Finally, we create a harsh scenario for stressing the equalizer. The AWGN, *FDdelay* and the *rotation angle* range are same as the ones in previous scenario. However, the updating frequencies of the *rotation angle* of the first, center and the last sections

Table 5.1: Resource utilization

Module	LUT utilization	DSP utilization	Slice registers	BRAM
trans_com	306	-	191	-
up_rrc_com	6508	96	6899	-
awgn	1883	24	3279	-
fd_com	411	12	470	-
angle_com	102	-	119	1.5
pmd_com	11474	576	5818	11
equalizer	2116	380	7455	-
demodulator	24	-	10	-
synchro_com	8	-	33	-
error_counter	6	-	254	-
ila_comp	1386	-	2696	9
Total	25117/5.60%	1088/30.22%	28592/3.30%	21.5/1.46%

are 10 kHz, 1 kHz and 100 Hz respectively. The step size is set to 0.001 to make the equalizer sensitive. After some time, as Fig. 5.19 shows, one channel of the equalizer loses its convergence and the errors are increasing rapidly.

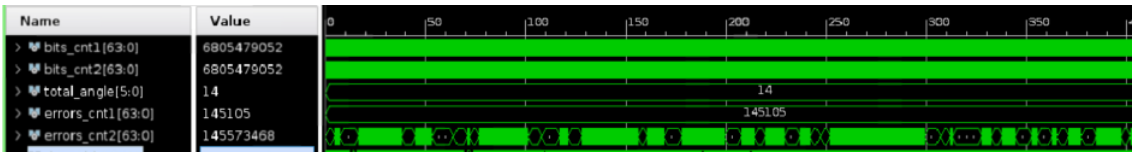


Figure 5.19: Emulation results of realistic system in a harsh scenario

The resource utilization for the realistic system is shown in Table 5.1, the percentages show how large portion of FPGA resources are utilized. The data signal is 12 bit wide. The RRC upsample filter has 51 taps with 16-bit coefficients. The PMD module has 10 sections and the filter of each section has 5 taps and the coefficient wordlength equal to 16. The equalizer has 11 taps with coefficient word-length equal to 16. The target FPGA platform is Xilinx Virtex-7 VC709 which is shown in Fig. A.7. The percentages show how large portion of FPGA resources are utilized.

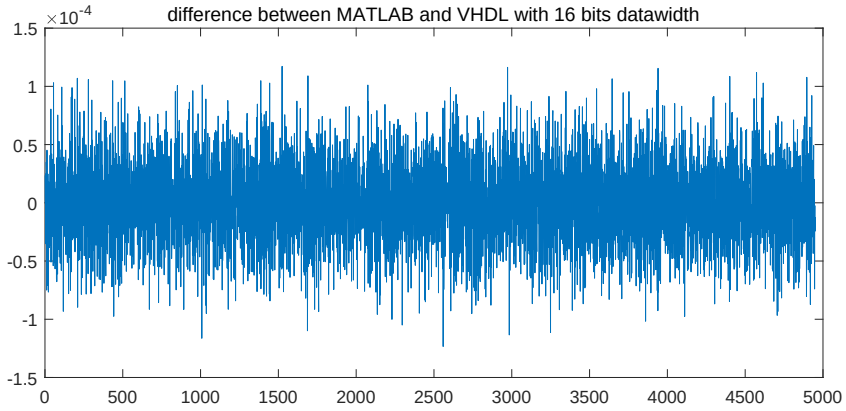
6

Discussion

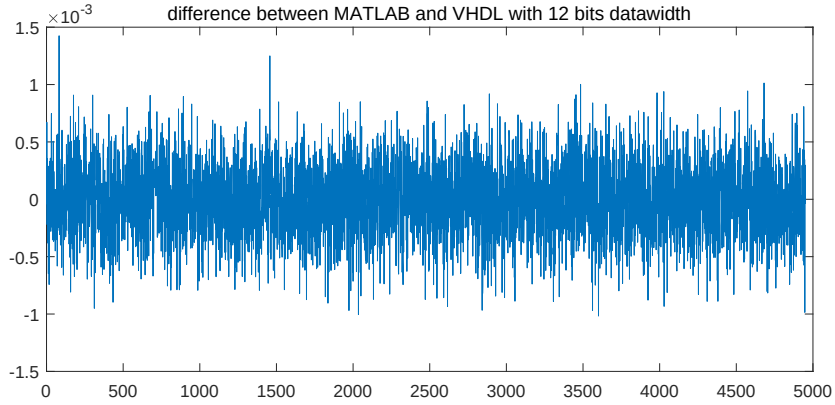
In this chapter we discuss the difference of PMD emulator between MATLAB and VHDL with different wordlength of the data, the resource consumption of the whole optical digital system under different wordlength of the data, and the resource consumption of different modulation formats.

6.1 Difference between PMD results with different wordlength of the data in MATLAB and VHDL

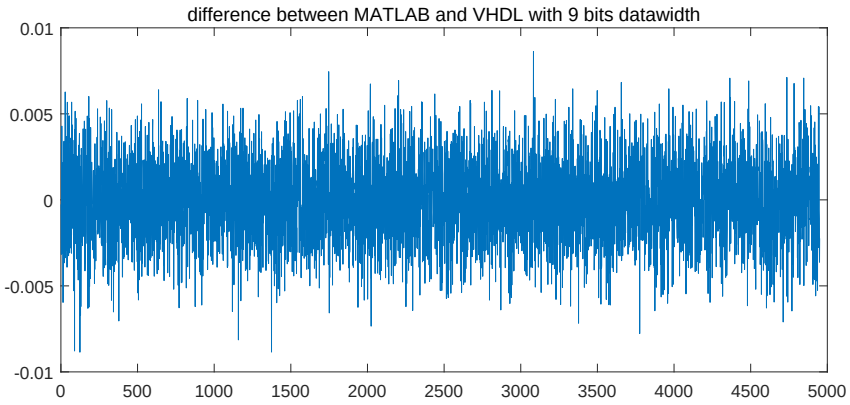
The performance difference between implementations of MATLAB and VHDL will decrease as we increase the wordlength of the data, however this will also lead to more resource being consumed. In order to study requirements on the data wordlength, we keep the other parameters of the PMD emulator unchanged, set the number of optical fiber section to 1 for the ease of observation, and only change the data wordlength, and then compare with the data generated by MATLAB. We choose the channel with *FDdelay* larger than zero as an example to show the difference. The difference between the outputs from MATLAB and VHDL is shown in Fig. 6.1. When the data wordlength is 16 bits, the difference is the smallest, and when the data wordlength is 9 bits, the difference is less than 0.01. Therefore, in order to reduce resource consumption and ensure high accuracy, we finally adopted 12 bits data wordlength.



(a) $FD = 0.03$, data wordlength = 16



(b) $FD = 0.03$, data wordlength = 12



(c) $FD = 0.03$, data wordlength = 9

Figure 6.1: QPSK difference with different data wordlength in MATLAB and Vivado

6.2 Resources of PMD and CMA equalizer with different data wordlength

Table 6.1 shows the total resources consumed by a digital system containing PMD and CMA equalizer with ten fiber section under different data wordlength, assuming the number of equalizer taps is 11. Compared with Table 5.1, when the tap count of equalizer increases, the resources consumed by equalizer increase significantly. When the data wordlength increases, the total resources consumed also increase except the DSP.

Table 6.1: Resource utilization with different data wordlength

Data wordlength	LUT utilization	DSP utilization	Slice registers	BRAM
12 bits	22857	1064	25174	21.5
16 bits	26093	1064	29642	21.5

6.3 Resources of PMD emulator with different modulation format

Different modulation format correspond to different RNG component output bits, so the resources consumed vary with the format. For 16QAM, 4 bits of RNG data are needed to generate a symbol, while QPSK only needs 2 bits of RNG data to generate a symbol, so 16QAM consumes more resources than QPSK. However, except for the increase in the resources of the transmitter and receiver, the other components remain basically unchanged, as shown in Table 6.2 and Table 6.3. The percentages show how large portion of FPGA resources are utilized. Therefore, changing the modulation format will not increase the complexity and resource utilization of the PMD emulator, but will only increase the resources of the transceiver.

Table 6.2: QPSK Resource utilization

Module	LUT utilization	DSP utilization	Slice registers	BRAM
trans_com	288	-	190	-
up_rrc_com	6524	96	6897	-
fd_com	425	12	470	-
angle_com	29	-	44	-
pmd_com	11474	576	5818	11
demodulator	4	-	9	-
synchro_com	8	-	35	-
error_counter	6	-	254	-
rrc_down_com	6096	96	3575	-
ila_comp	1033	-	2103	7.5
Total	25887/5.98%	780/21.67%	19395/2.23%	18.5/1.26%

Table 6.3: 16QAM Resource utilization

Module	LUT utilization	DSP utilization	Slice registers	BRAM
trans_com	317	-	211	-
up_rrc_com	6569	96	6927	-
fd_com	425	12	470	-
angle_com	29	-	44	-
pmd_com	11474	576	5818	11
demodulator	12	-	9	-
synchro_com	14	-	43	-
error_counter	8	-	126	-
rrc_down_com	6144	96	3587	-
ila_comp	1034	-	2103	7.5
Total	26026/6.00%	780/21.67%	19338/2.23%	18.5/1.26%

7
Conclusion

In this work, for the purpose of developing an FPGA-based digital model of optical-fiber communication, we developed a digital, practical and flexible PMD emulator and a basic CMA equalizer for equalizing the impairments. Finally, we implemented a more realistic fiber-optic system model based on the CHOICE platform. However, because the project never reached the ASIC implementation stage, the power dissipation of the equalizer mentioned in the project goals was never evaluated.

The modulation format of the system can be switched between QPSK and 16QAM by setting the generic *MOD_TYPE* in the system top module. For the PMD emulator, the DGD of each fiber section, which is used to control PMD, can be changed by setting the value of generic *FDdelay* between 0 and 1; the speed of rotation can be changed by setting the parameters $f1$, $f2$, and $f3$ of the rotation component; we can also set the number of fiber sections by changing the top-level generic *SECTIONS*.

By comparing the Vivado simulation results of the one-section and 10-section PMD emulator with the MATLAB simulation results, we find that the difference between the PMD VHDL model and the MATLAB model is very small. When the number of fiber sections increases, the difference between the PMD VHDL model and the MATLAB model will increase correspondingly. The real-time FPGA running data is captured and analyzed through ILA, and can be further analyzed using MATLAB.

Future extensions of the system will be parallelizing the system for increasing its data rate and better utilize the FPGA resources. The power dissipation of the equalizer is an interesting part of further research as well.

Bibliography

[1] M. Winter, D. Setti, and K. Petermann, "Cross-polarization modulation in polarization-division multiplex transmission," *IEEE Photonics Technology Letters*, vol. 22, no. 8, pp. 538–540, 2010.

[2] H. Wernz, S. Bayer, B. E. Olsson, M. Camera, H. Griesser, C. Furst, B. Koch, V. Mirvoda, A. Hidayat, and R. Noe, "112Gb/s PolMux RZ-DQPSK with fast polarization tracking based on interference control," in *2009 Conference on Optical Fiber Communication*, 2009, pp. 1–3.

[3] R. Noé, S. Hinz, and F. Sandel, D.and Wüst, "Crosstalk detection schemes for polarization division multiplex transmission." *Journal of Lightwave Technology*, vol. 19, no. 10, pp. 1469–1475, 2001.

[4] A. Galtarossa and C. R. Menyuk, *Polarization mode dispersion*. Springer Science & Business Media, 2006, vol. 1, pp. 113–124.

[5] T. Kupfer, A. Bisplinghof, T. Duthel, C. Fludger, and S. Langenbach, "Optimizing power consumption of a coherent DSP for metro and data center interconnects," in *2017 Optical Fiber Communication Conference and Exhibition(OFC)*. Optical Society of America, 2017, p. Th3G.2.

[6] L. Yan, X. S. Yao, M. Hauer, and A. Willner, "Practical solutions to polarization-mode-dispersion emulation and compensation," *Journal of lightwave technology*, vol. 24, no. 11, pp. 3992–4005, 2006.

[7] H. Kogelnik, R. M. Jopson, and L. E. Nelson, "Polarization-mode dispersion," in *Optical Fiber Telecommunications IV-B*. Elsevier, 2002, pp. 725–861.

[8] G. Foschini and C. Poole, "Statistical theory of polarization dispersion in single mode fibers," *Journal of Lightwave Technology*, vol. 9, no. 11, pp. 1439–1456, 1991.

[9] J. Damask, "A programmable polarization-mode dispersion emulator for systematic testing of 10 Gb/s PMD compensators," in *Optical Fiber Communication Conference*. Optical Society of America, 2000, p. ThB3.

[10] R. Khosravani, I. Lima, P. Ebrahimi, E. Ibragimov, A. Willner, and C. Menyuk, "Time and frequency domain characteristics of polarization-mode dispersion emulators," *IEEE Photonics Technology Letters*, vol. 13, no. 2, pp. 127–129, 2001.

[11] M. Hauer, Q. Yu, E. Lyons, C. Lin, A. Au, H. Lee, and A. E. Willner, "Electrically controllable all-fiber PMD emulator using a compact array of thin-film microheaters," *Journal of Lightwave Technology*, vol. 22, no. 4, p. 1059, 2004.

[12] F. Curti, B. Daino, Q. Mao, F. Matera, and C. Someda, "Concatenation of polarisation dispersion in single-mode fibres," *Electronics Letters*, vol. 25, pp. 290–292(2), February 1989.

[13] B. Lu, "Simulation of optical fiber communication system with polarization mode dispersion," Master's thesis, University of New Brunswick, Fredericton, NB, Canada, 2000.

[14] C. B. Czegledi, "Modeling and compensation of polarization effects in fiber-optic communication systems," Ph.D. dissertation, Chalmers University of Technology, Gothenborg, Sweden, 2018.

[15] E. Börjeson, C. Fougstedt, and P. Larsson-Edefors, "Towards FPGA emulation of fiber-optic channels for deep-BER evaluation of DSP implementations," in *OSA Advanced Photonics Congress (AP) 2019 (IPR, Networks, NOMA, SPPCom, PVLED)*. Optical Society of America, 2019, p. SpTh1E.4.

[16] G. Yeswanthi, M. Kavitha, P. S. Priyanka, and D. G. Kurup, "Programmable high data rate QPSK modulator for space applications," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 2276–2278.

[17] A. Das, *Digital Communication. [electronic resource] : Principles and System Modelling.*, ser. Signals and Communication Technology. Springer Berlin Heidelberg, 2010.

[18] K. Wesolowski, *Introduction to digital communication systems*. John Wiley & Sons, 2009.

[19] M. Joost, "Theory of root-raised cosine filter," *Research and Development*, vol. 47829, 2010.

[20] J. P. Gordon and H. Kogelnik, "PMD fundamentals: Polarization mode dispersion in optical fibers," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 97, no. 9, pp. 4541–4550, 2000.

[21] F. Curti, B. Daino, G. De Marchis, and F. Matera, "Statistical treatment of the evolution of the principal states of polarization in single-mode fibers," *Journal of Lightwave Technology*, vol. 8, no. 8, pp. 1162–1166, 1990.

[22] V. Valimaki and T. I. Laakso, "Principles of fractional delay filters," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, vol. 6. IEEE, 2000, pp. 3870–3873.

[23] M. Xue, Y. Yanfu, X. Qian, C. Juntao, Z. Qun, and Y. Yong, "Joint blind equalization of PDL and RSOP using extended Kalman filter algorithm in stokes vector direct detection system," in *Eleventh International Conference*

on Information Optics and Photonics (CIOP 2019), vol. 11209. International Society for Optics and Photonics, 2019, p. 1120958.

[24] D. E. Crivelli, H. S. Carter, and M. R. Hueda, "Adaptive digital equalization in the presence of chromatic dispersion, PMD, and phase noise in coherent fiber optic systems," in *IEEE Global Telecommunications Conference, 2004. GLOBECOM '04.*, 2004, pp. 2545–2551.

[25] A. Kaye and D. George, "Transmission of multiplexed PAM signals over multiple channel and diversity systems," *IEEE Transactions on Communication Technology*, vol. 18, no. 5, pp. 520–526, 1970.

[26] A. Shah, R. Hsu, A. Tarighat, A. Sayed, and B. Jalali, "Coherent optical MIMO (COMIMO)," *Journal of Lightwave Technology*, vol. 23, no. 8, pp. 2410–2419, 2005.

[27] M. S.K. and R. A., "Equalization techniques in MIMO systems - an analysis," in *2014 International Conference on Communication and Signal Processing*, 2014, pp. 1082–1086.

[28] W. Rao, K. Yuan, Y. Guo, and C. Yang, "A simple constant modulus algorithm for blind equalization suitable for 16-QAM signal," in *2008 9th International Conference on Signal Processing*, 2008, pp. 1963–1966.

[29] S. J. Savory, "Digital coherent optical receivers: Algorithms and subsystems," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 16, no. 5, pp. 1164–1179, 2010.

[30] S. Dhabu, A. Ambede, K. Smitha, S. and, and A. Vinod, "Variable cutoff frequency FIR filters: a survey," *SN Applied Sciences*, 2020.

[31] B. Hou, Y. Yao, and M. Qin, "Design and FPGA implementation of high-speed parallel FIR filters," in *3rd International Conference on Mechatronics, Robotics and Automation*. Atlantis Press, 2015, pp. 975–979.

[32] G. Marsaglia, "Xorshift RNGs," *Journal of Statistical Software, Articles*, vol. 8, no. 14, pp. 1–6, 2003.

[33] C. Fougstedt, P. Johannisson, L. Svensson, and P. Larsson-Edefors, "Dynamic equalizer power dissipation optimization," in *Optical Fiber Communication Conference*, Mar. 2016, p. W4A.2.

[34] W. Rao, W. Tan, D. Li, G. Dai, F. Xia, L. Fan, J. Liu, and H. Xu, "Concurrent blind equalization suitable for 16-QAM signal," in *2009 International Conference on Wireless Communications Signal Processing*, 2009, pp. 1–5.

A

Appendix

A.1 Simulation results of 10 sections QPSK and 16QAM

We set the number of section to 10 and set the *FDdelay* and *rotation angle* of each section to 0.03 and 6° respectively, so now the total *FDdelay* is 0.3 and the total *rotation angle* is 66°. The simulation results and the difference of Vivado and MATLAB is shown in Fig. A.1 and Fig. A.2. When the number of sections increases from 1 to 10, the difference between MATLAB and VHDL results also increases. This is because there will be a loss of precision when converting MATLAB floating-point numbers to Vivado fixed-point numbers. Each section will have a certain error. When the number of sections increases, the error of each section will also accumulate.

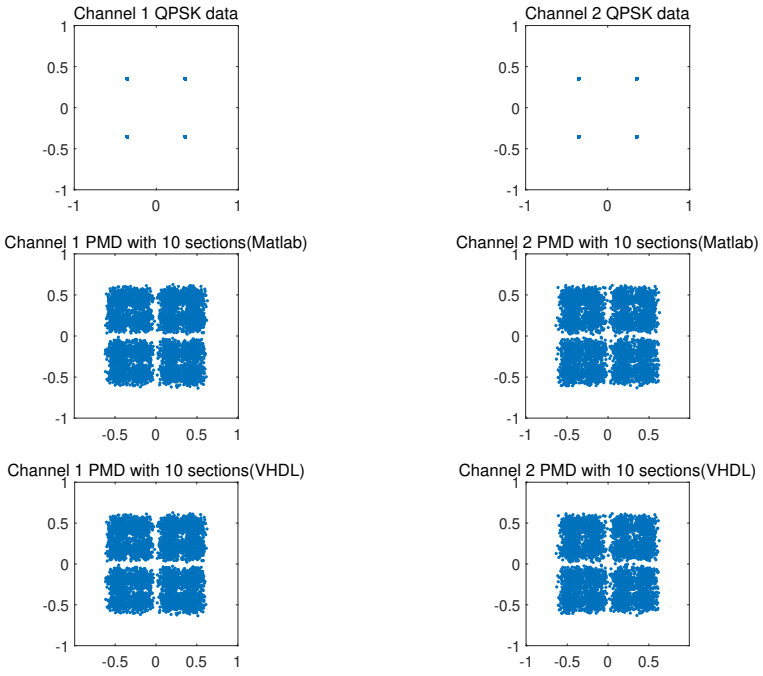


Figure A.1: QPSK simulation results of MATLAB and Vivado with 10-section

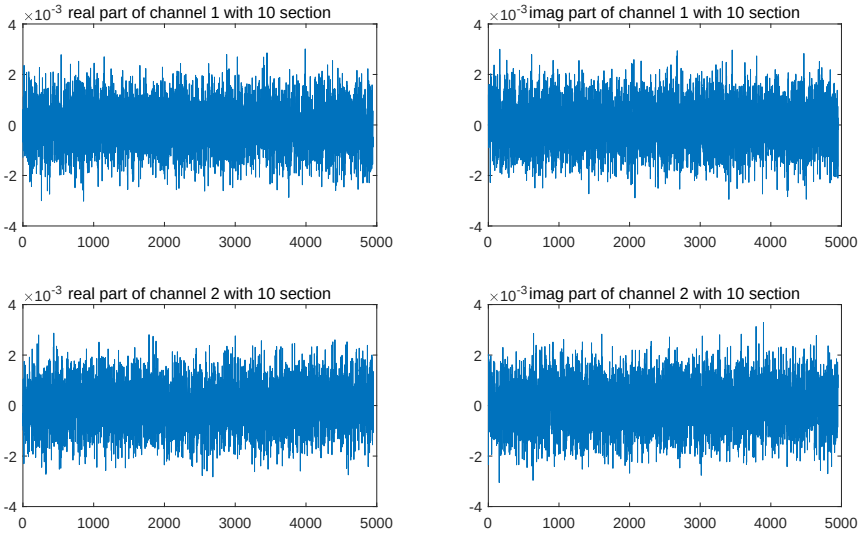


Figure A.2: Difference between MATLAB and Vivado with 10-section of QPSK modulation

The 16QAM fiber section is also changed from 1 to 10, and the other parameter settings are the same as QPSK with 10 sections. The simulation results are shown in Fig. A.3 and Fig. A.4, which is similar to QPSK, when the number of sections increases, the cumulative error between MATLAB and VHDL also increases.

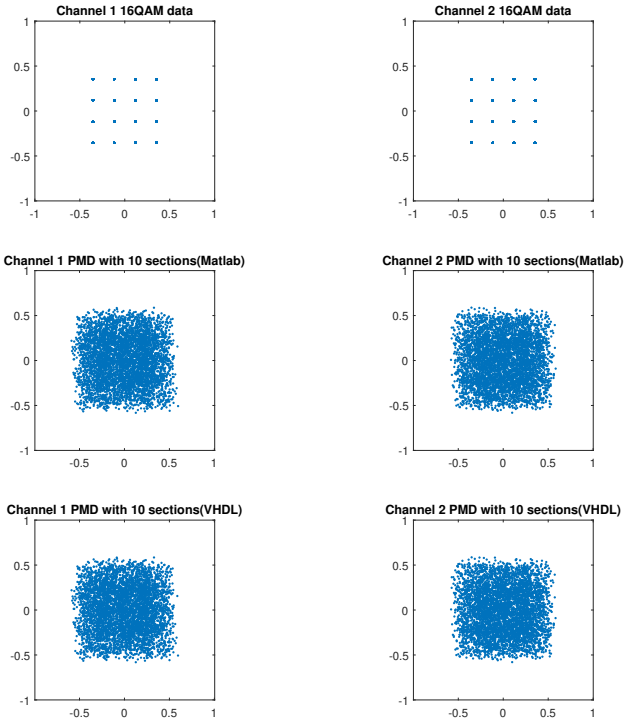


Figure A.3: 16QAM simulation results of MATLAB and Vivado with 10-section

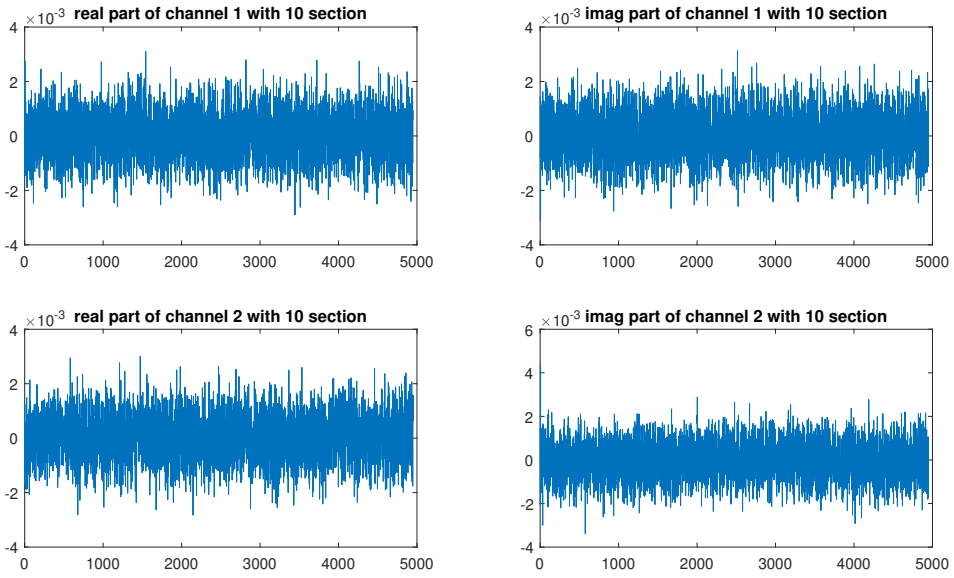
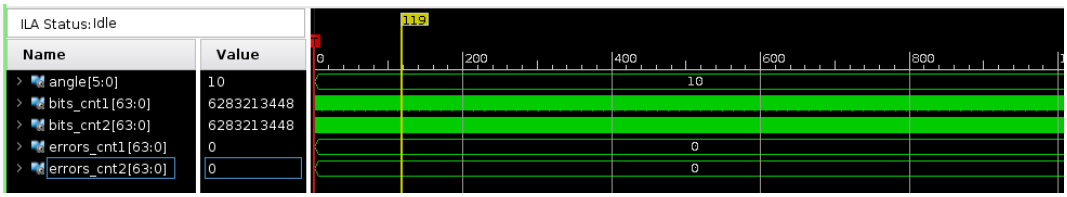


Figure A.4: 16QAM difference between MATLAB and Vivado with 10-section

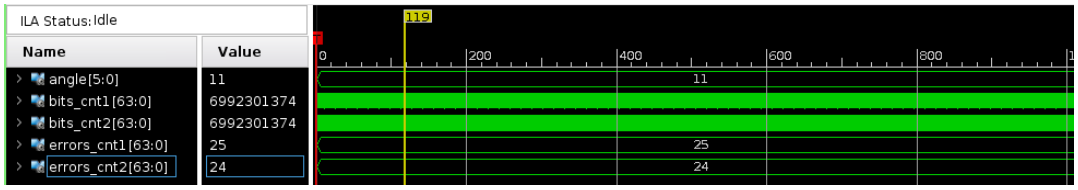
A.2 FPGA emulation results of 0.3 *FDdelay*

A.2.1 QPSK emulation results

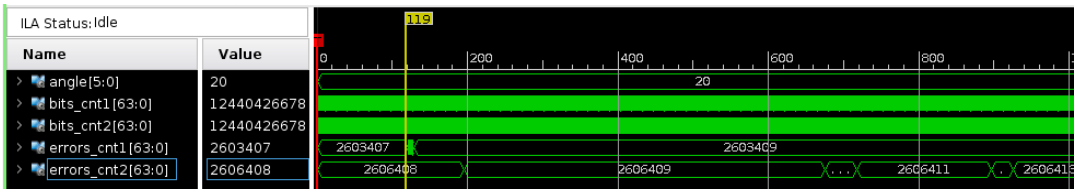
We change *FDdelay* to 0.3 and download it to the FPGA to run. The result is shown in Fig. A.5. When the *rotation angle* is 10°, no errors appear, when the *rotation angle* is increased to 11°, there are a few errors, and when the *rotation angle* is increased to 20°, there are a large number of errors, similar to the simulation results.



(a) *rotation angle* = 10°



(b) *rotation angle* = 11°



(c) *rotation angle* = 20°

Figure A.5: QPSK ILA emulation result with one section and *FDdelay* = 0.3

A.2.2 16QAM emulation results

Change the *FDdelay* to 0.3, the emulation result is shown in Fig. A.6, when the *rotation angle* is 0, the errors appear, which is consistent with the simulation results.

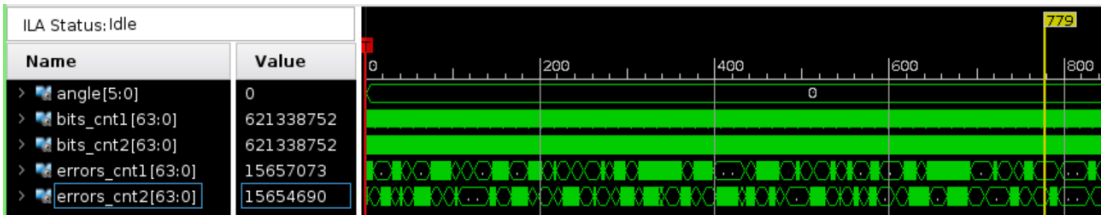


Figure A.6: 16QAM ILA emulation result with one section and *FDdelay* = 0.3

A.3 FPGA board

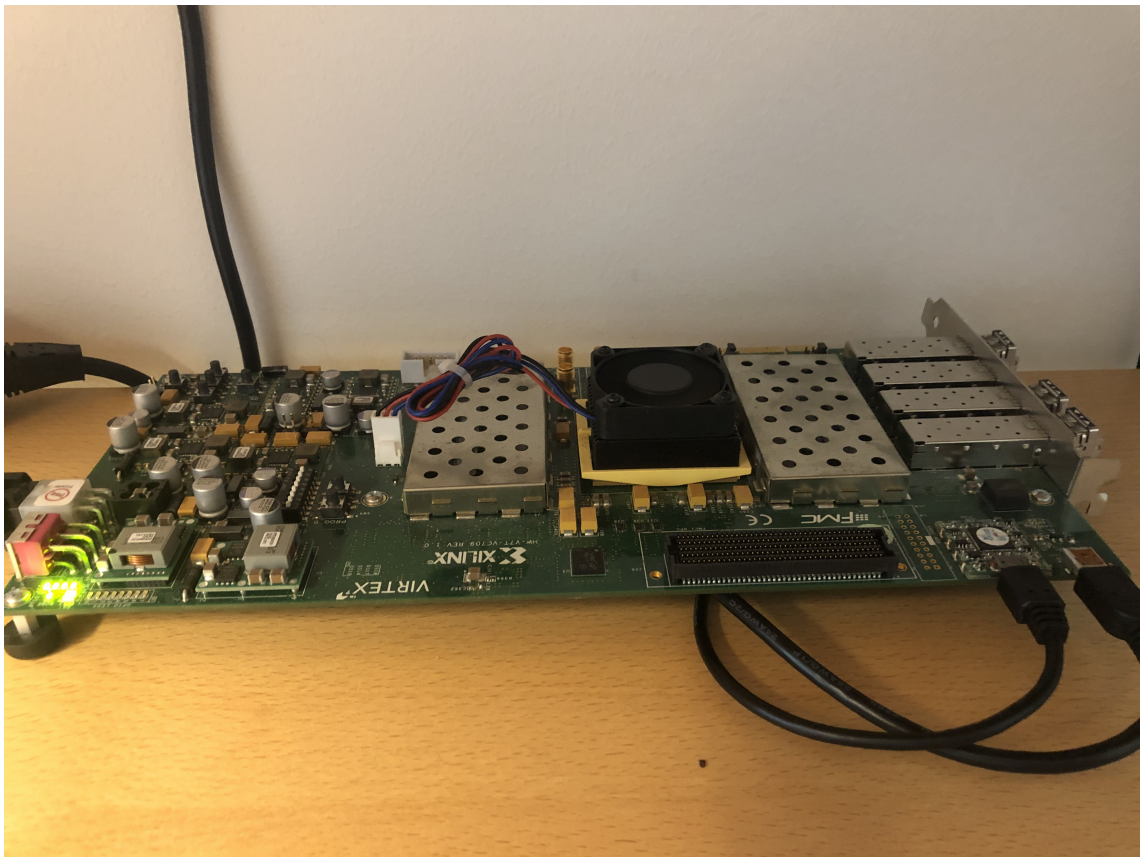


Figure A.7: XILINX Virtex-7 709 FPGA board