FPGA IMPLEMENTATION OF OFDM BASED VLC

A Thesis

by

Gürol Sağlam

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Computer Science

Özyeğin University
August 2022

FPGA IMPLEMENTATION OF OFDM BASED VLC

Approved by:

Professor H. Fatih Uğurdağ, Advisor
Department of Electrical and Electronics Engineering
Özyeğin University

Professor Murat Uysal
Department of Electrical and Electronics Engineering
Özyeğin University

Professor Nizamettin Aydın
Department of Computer Engineering
Yıldız Technical University

Date Approved: 08/08/2022

To those I love and value

ABSTRACT

This thesis presents the Field Programmable Gate Array (FPGA) based implementation of Orthogonal Frequency Division Multiplexing (OFDM) for a Visible Light Communication (VLC) System. This happens to be the Single-Input Single Output (SISO) Direct Current Biased Optical OFDM (DCO-OFDM) version. The details of the transceiver and its parts are explained. VLC is a type of data communication that employs visible light from a light emitting diode (LED) as a short-range transmission technique. Communication occurs by modulating the optical signal, one such modulation technique is the DCO-OFDM employed in this thesis. Wang et. al. proposed a VLC system using a red LED, where DCO-OFDM is employed as the modulation technique [1]. However, even though they show the basic structure of their implementation as a block diagram, little detail is given into how the communication occurs. The devices used and the area of the blocks are not provided, the structure of the OFDM frames are not explained. Fuada et. al. presents a System-on-Chip (SoC) architecture for OFDM for Optical Wireless Communication (OWC) in different studies [2, 3]. They give some details on their implementation and mention the devices they used. However, in their OFDM frame structure they do not have a Channel Estimation Training Sequence (CETS) to recover the signal from the noise added by a dispersive channel, and the spectral shaping occurs outside of the FPGA, whereas this thesis implements this within the FPGA. In this thesis, assuming the same methodologies that Wang et. al. used were used for the implementation, this implementation is $3.22x$ faster, and assuming the methodologies from the latter group were used, this implementation is $2.92x$ faster.

ÖZETÇE

Bu tez, Görünür Işıkla Haberleşme (VLC) Sistemi için Ortogonal Frekans Bölmeli Çoğullama'nın (OFDM) Alanda Programlanabilir Kapı Dizisi (FPGA) tabanlı uygulamasını sunar. Bu, Tek Girişli Tek Çıkışlı (SISO) Doğru Akım Eklemeli Optik OFDM (DCO-OFDM) versiyonudur. Alıcı-vericinin ve parçalarının detayları açıklanmıştır. VLC, bir ışık yayan diyottan (LED) gelen görünür ışığı kısa menzilli iletim tekniği olarak kullanan bir tür veri iletimidir. İletişim, optik sinyalin modülasyonu ile elde edilir; bu modülasyon türlerinden biri bu tezde kullanılmış olan DCO-OFDM'dir. Wang ve arkadaşları modülasyon tekniği olarak DCO-OFDM'nin kullanıldığı kırmızı bir LED kullanan bir VLC sistemi önermiştir [1]. Bununla birlikte, tasarımlarının temel yapısını bir blok diyagram olarak gösterseler de, iletişimin nasıl gerçekleştiğine dair çok az ayrıntı verilmiştir. Kullanılan cihazlar ve blokların alanı belirtilmemiş, OFDM çerçevelerinin yapısı anlatılmamıştır. Fuada ve arkadaşları farklı çalışmalarda Optik Kablosuz İletişim (OWC) için OFDM için bir Yonga Üzerinde Sistem (SoC) mimarisi sunmuştur [2, 3]. Tasarımları hakkında bazı detaylar vermişlerdir ve kullandıkları cihazlardan bahsetmektelerdir. Bununla birlikte, OFDM çerçeve yapılarında, dağıtıcı bir kanal tarafından eklenen gürültüden sinyali geri kazanmak için bir Kanal Tahmini Eğitim Dizisi (CETS) yoktur ve spektral şekillendirme FPGA dışında gerçekleşir, oysa bu tez bunu FPGA içinde uygular. Bu tezde, Wang ve arkadaşlarının tasarım için kullandığı metodoloji kullanıldığı durumda bu tasarım $3.22x$ daha hızlıdır, ve ikinci grubun metodolojisinin kullanıldığı durumda bu tasarım $2.92x$ daha hızlıdır.

ACKNOWLEDGEMENTS

It has been a long and arduous journey; this one is only the prequel. I can only hope that the next one is as good as this one.

I must start by thanking my Professor, Prof. Dr. Hasan Fatih Uğurdağ, for helping me become this person. I met him at the end of my first year in Bachelor's, around June 2016, when he became an advisor for a project we wanted to do. After meeting him, he paid close attention to me and gave me advice when needed. He helped me direct myself in the right direction regarding my future career.

I want to thank Dr. Mohammed Elamassie, who helped me with my work when needed. He helped me understand the concepts in my work; answered every question I ever had, half of them before I asked them.

I must mention the nEMESysLab (an Embedded and MicroElectronic Systems Lab) and their valuable members for working with me and elevating one another.

I want to express my appreciation for all my friends I know in person and through social networks. Everyone contributed in some way to my life; some listened to me while I rambled on about something I am compassionate about, some offered me a piece of their mind, and some offered peace of mind. I should also especially thank Deniz Kurt for always putting up with me, guiding me in unfamiliar situations, and exchanging technical information with me. And also Helin Özel for pushing me to finish my work and encouraging me to push myself forward. I am grateful to everyone who gifted a piece of themselves and helped me grow as a person.

Last but not least, I would like to thank my parents for always believing in me. They followed every step I took and supported me in them. They helped me become the person I am today, and they will continue to do so for me to become the best version of me.

Contents

List of Tables

List of Figures

Chapter I

INTRODUCTION

This thesis introduces the hardware implementation of a Visible Light Communication (VLC) Modulation Technique called DC Biased Optical Orthogonal Frequency Division Multiplexing (DCO-OFDM).

1.1 Problem Statement

In recent years, the focus on VLC Systems has been increasing as a means of data transmission. Many works are being done on this topic, and many different modulation techniques are devised for these systems.

The most basic technique for a VLC System could be considered to be mapping the 1s (ones) and 0s (zeros) of data into the "light" from a light source such as a Light Emitting Diode (LED) or a Laser as the light source turned on and turned off, respectively. However, using this technique will be difficult or impossible at high transmission speeds because the switching time (the rise time and fall time) of the source limits the transmission speed. The source has to be given enough time for the data to be transmitted accurately. Therefore, other more complex algorithms were devised that modulate the data in some way that hides it within the light without turning the source off completely. These different modulation techniques will be discussed in the next chapter.

In the scope of this thesis, DCO-OFDM was chosen as the modulation technique to be implemented as a Register-Transfer Level (RTL) design.

1.2 Why DCO-OFDM?

DCO-OFDM is one specific technique of many modulation techniques available under the VLC Modulation Techniques umbrella. This section answers why this technique was implemented as an RTL design.

1.2.1 What is OFDM?

Orthogonal Frequency Division Multiplexing (OFDM) is a data transmission method devised generally for wireless data transmission. The technique embeds a data stream into several narrowband frequencies instead of one wideband frequency. It is mainly used in Radio Frequency communication.

The output of the method is complex and bipolar, making it incompatible with VLC Systems since the output of the light sources is real and positive. However, several different methods of OFDM modify the output to be real and positive, making those methods suitable for VLC Systems.

1.2.2 What is DCO-OFDM?

DCO-OFDM, or DC (Direct Current) Biased Optical Orthogonal Frequency Division Multiplexing, is a modulation technique developed for VLC Systems. Since in OFDM, the output is complex and bipolar, to be able to use it in VLC Systems, the output must be converted to real and positive.

In this modulation, the input is modulated with Quadrature Amplitude Modulation (QAM) which has complex and bipolar results. Then Hermitian Symmetry is applied to these results so that the output of the Inverse Fast Fourier Transform (IFFT) are real values [5, 6]. After the IFFT, the Cyclic Prefix (CP) is added, and then the Preamble and the Channel Estimation Training Sequence (CETS) are added. After the signal is upsampled and filtered, the process which is also called spectral shaping, the resulting signal becomes real, but bipolar.

In VLC Systems, the communication must not have a dimming effect on lighting, therefore there must be a static level of voltage supplied to the LED that carries the information. Hence, the output is then added to a DC Bias to ensure the signal is real and positive and has no dimming effect.

1.2.3 Why was DCO-OFDM chosen?

For VLC Systems, within all of the different existing modulation techniques, Optical OFDM is the most promising for communication in higher data rates. High

data rates are realized by utilizing multiple orthogonal subcarriers to transmit parallel data streams simultaneously, reducing Inter Symbol Interference (ISI) and the need for complex equalizers [7, 6]. The main advantages of Optical OFDM over the single-carrier techniques are:

- Optical OFDM has higher spectral efficiency [6, 8, 9].
- It is robust against channel dispersion (chromatic dispersion, PMD, mode dispersion) [6, 8, 9].
- It adapts to the time-varying channel conditions [9].
- It has straightforward channel estimation and equalization schemes [9].

However, OFDM shows a large peak-to-average power ratio (PAPR) and is sensitive to frequency offset and phase noise. Those are the drawbacks that need to be addressed in high speed application scenarios.

Optical OFDM is much more complex in terms of implementation with respect to the other techniques, however; the advantages outweigh the complexity of implementation.

1.3 Contribution of the Thesis

In this thesis, the implementation of a DC Biased Optical Orthogonal Frequency Division Multiplexing (DCO-OFDM) Transceiver is developed on the Intel Arria10 SoC Development Kit using Verilog HDL. The design includes the ethernet receiver and transmitter implementations and Digital-Analog Converter (DAC) and Analog-Digital Converter (ADC) controller implementations. The DCO-OFDM transmitter and receiver are both implemented on the same design, on the same FPGA.

The ethernet receiver part of the design receives the packets sent from the transmitting computer. After adding a “header” and “footer” in front and at the

end of the packet, it is sent to the DCO-OFDM transmitter. In turn the DCO-OFDM transmitter modulates the incoming signal and sends it to the DAC to be transmitted.

The ADC receives the signal, and then immediately sends the signal to the DCO-OFDM receiver to be demodulated. The DCO-OFDM receiver finds the OFDM frame within the signal and demodulates it, then sends it to the ethernet transmitter. The ethernet transmitter takes the header and footer apart from the actual ethernet packet, then sends the ethernet packet to the receiving computer.

For the thesis, two different Quadrature Amplitude Modulation (QAM) techniques were used with two different speeds of DAC and ADC. Due to the channel noise increasing the Bit Error Rate (BER), after analysing the received data, some subcarriers were cancelled in order to improve the BER. QPSK (4-QAM) results in 0% BER using the DAC and ADC with a speed of 125 MHz, resulting in a data rate of 1.875 Mb/s, and with the DAC and ADC being used with the speed set to 250 MHz, the implementation results in a doubled data rate, 3.75 Mb/s. The 256-QAM technique results in a higher BER while reaching higher data rate (9.6875 Mb/s if DAC/ADC are used with 125 MHz clock, and 19.375 Mb/s if used with 250 MHz) with respect to the 4-QAM design. The results will be discussed in detail in Chapter 4.

Chapter II

RELATED WORK

Visible Light Communication (VLC) is a broad topic with many branches called Modulation Techniques. The topic dates back to the end of 19th century, to Alexander Graham Bell, and has been the focus of many researchers in the last few decades. Since then, the topic has only broadened and branched out into its categories. In this chapter, the history of VLC is explained, and then some of the current Modulation Techniques in the literature will be discussed.

2.1 History of Visible Light Communication

The earliest known use of visible light wireless communication comes from Alexander Graham Bell, who developed a photophone in 1880, which transmitted voice data over 200m using beams of sunlight [10, 6, 11]. This was useful in theory, but in practice would have been an insufficient means of communication since communication from larger distances would be more difficult to achieve in free space transmission due to this free space being lit up with sunlight during the communication, introducing a large amount of noise, and also since it was using sunlight, this means of communication was restricted only to daytime. This method would require another type of light source that would make it independent of daytime that did not exist until much later.

After other methods of communication were invented, the topic of Visible Light Communication came to a halt, until 1999. Then, the concept of using fast switching LEDs as well as modulating visible light for communication was presented for the first time by Pang et al. in 1999 [12, 13, 6, 11]. In 2001, RONJA (Reasonable Optical Near Joint Access) used visible light beams to transmit data at 10 Mb/s over 1.4 km [14]. There are several types of commercial RONJA models and numerous installations all over the world. At Keio University, Japan,

Tanaka et al. led the way to indoor wireless communication utilizing White-LED in early 2000s [15, 16].

The Visible Light Communication Consortium (VLCC) was established in 2003 with major companies in Japan with the aim to publicize and standardize the VLC technology [17, 6, 11]. hOME Gigabit Access project (OMEGA) in Europe aiming to develop a user-friendly home access network that could deliver high-bandwidth services at gigabit data rates achieved a 100 Mbps video broadcast based on visible LED on ceiling lighting in 2011 [18, 19]. The project had 20 European partners from both industry and academia, including Siemens and France Telecom. In that same year, the first IEEE 802.15.7 VLC standard was published with definitions of a Physical (PHY) and a MAC layer for short-range optical wireless communications using visible light in optically transparent media [20, 21, 22].

There are currently various research groups from both the academia and the industry including, but not limited to: Center for Ubiquitous Communication by light (UC-Light) [23], Center on Optical Wireless Applications (COWA) [24], Smart Lighting Engineering Research Centre (Boston University) [25], University of Edinburgh [26] and Oxford University [27][6].

2.2 Modulation Techniques

Many different types of Modulation Techniques currently exist in the literature. They can mainly be separated into 4 categories: Single-Carrier Modulation (SCM), Multi-Carrier Modulation (MCM), Orthogonal Frequency Division Multiplexing (OFDM) and Colour Domain Modulation. OFDM technically should fall under MCM since it utilizes multiple carriers, however; there are so many various types of OFDM techniques that makes it into a category of its own.

Islim M.S. and Haas H. discuss several modulation techniques in their paper [4]. The techniques they consider within SCM are On-Off Keying (OOK), Pulse Width Modulation (PWM), M-ary Pulse Amplitude Modulation (M-PAM), M-ary Pulse Position Modulation (M-PPM), Discrete Fourier Transformation spread

OFDM (DFT-s-OFDM) and Carrier-less Amplitude Modulation (CAP). In Colour Domain Modulation, they talk about Colour Shift Keying (CSK), Colour Intensity Modulation (CIM) and Metameric Modulation (MM). MCM is comprised of several techniques including OFDM, however; it is its own category, hence the rest of the MCM techniques considered in their paper are Hadamard Coded Modulation (HCM), Wavelet Packet Division Multiplexing (WPDM) and Discrete Hartley Transform.

The category OFDM is then separated into sub-categories: DC biased Optical OFDM (DCO-OFDM), Inherent Unipolar, Superposition OFDM and Hybrid. Islim and Haas consider Asymmetrically Clipped Optical OFDM (ACO-OFDM), Pulse Amplitude Modulation Discrete Multitone (PAM-DMT) and Unipolar OFDM/ Flipped OFDM (U-OFDM/Flip-OFDM) within the Inherent Unipolar subcategory. In Superposition OFDM, there is Enhanced Unipolar OFDM (eU-OFDM), Enhanced ACO-OFDM (eACO-OFDM), Enhanced PAM-DMT (ePAM-DMT), Spectrally and Energy Efficient OFDM (SEE-OFDM), and Layered ACO-OFDM (LACO-OFDM). There are also several Hybrid OFDM Modulation Techniques, such as Polar-OFDM (P-OFDM), Asymmetrically Clipped DC Biased Optical OFDM (ADO-OFDM) and Hybrid Asymmetrically Clipped Optical OFDM (HACO-OFDM). The various modulation techniques in Islim and Haas' paper are shown in Figure 1.

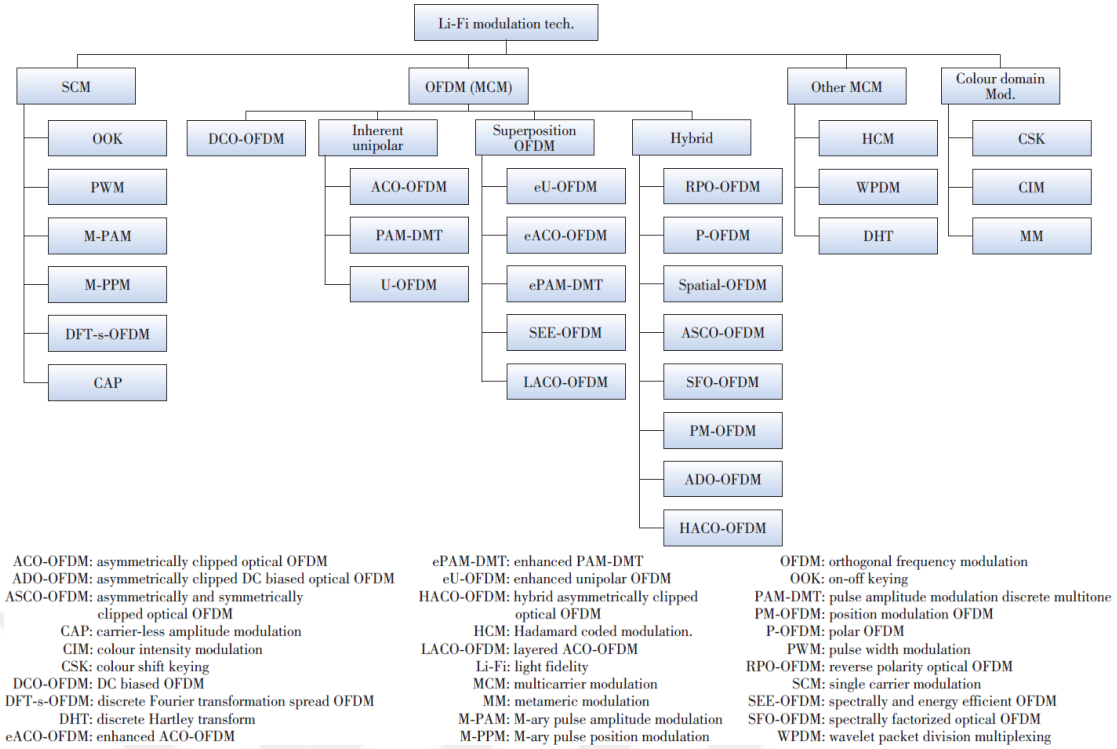

Figure 1: Li-Fi Modulation Techniques [4].

Some of these different modulation techniques will be shortly explained in this section. The first category will be Single-Carrier Modulation, since it is the most elementary category, then some Colour Domain Modulation techniques will be explained, and then some Multi-Carrier Modulation techniques will follow. I will conclude this section with OFDM. I have chosen to discuss them in this order so that I can talk about the different techniques before focusing on the OFDM Techniques, the category my thesis work is a part of.

2.2.1 Single-Carrier Modulation (SCM)

In SCM, much like traditional forms of transmission, one sinusoidal signal is used. This sinusoidal signal is added to the DC level, so that the light stays turned on, allowing illumination. However, the lighting fluctuates with the sinusoidal signal, where the digital data is encoded.

This modulation technique category is the most straightforward to implement for Li-Fi Systems, and preferable when the data rate requirements are low to

moderate [4]. The data is encoded into the light as “on” or “off” states. The light intensities of the “on” and “off” states are adjusted for illumination control.

In OOK, the method is to change the LED’s illumination level to “on” or “off” state according to the bits, 1 for “on” and 0 for “off”. These states are Manchester Coded in the 802.15.7 Standard, so that the period of positive pulses and negative pulses are the same. In this coding, however, the required bandwidth is doubled. Instead of Manchester Coding, Run Length Limited (RLL) might be used to achieve a higher bit rate.

The idea is to change the duty cycle of the signal according to the data in PWM. With a 1-bit PWM, there are 2 states to indicate 1s and 0s. That means there are 2 different duty cycles which must have a difference that must be wide enough to have a high noise margin, but also must be narrow enough so that there is no dimming effect on the lighting. In the implementation of a multi-bit PWM, there must be 2^{M} many states, M being the number of bits. The more bits introduced to the PWM, the more states; which decreases the maximum duty cycle difference it can have. With decreased duty cycle difference, due to noise, the error rate will increase [28].

2.2.2 Colour Domain Modulation

In Colour Domain Modulation, Colour Shift Keying (CSK), Colour Intensity Modulation (CIM) and Metameric Modulation (MM) are considered, in the work of Islim M.S. and Haas H. [4]. In this section, CSK will be explained shortly.

CSK as a modulation technique is proposed by the IEEE802.15.7 standard [4, 21]. The received bits are mapped using a constellation of colours from the chromatic CIE 1931 colour space [4, 29]. Using an RGB LED, the instantaneous colour of the LED is changed, meanwhile keeping the overall intensity of the output colour. This allows constant illumination and eliminates the dimming effects. CSK is advantageous in that it overcomes the defects of PWM, and the encryption of the bit patterns are done in wavelength sets [8, 30]. Although it can overcome the PWM defects, CSK cannot be employed in VLC systems that use pc-LED as the

source [6]. The implementation of this modulation technique is also complex [6].

2.2.3 Multi-Carrier Modulation (MCM)

Although OFDM is technically a modulation technique that should be under this category, since it is a large umbrella encompassing multiple different techniques, it is a category of its own. Other than OFDM; HCM, WPDM and DHT are considered under this category.

HCM was proposed for VLC systems as a solution to the limitations of OFDM at higher illumination levels [4]. As an alternative to FFT used in OFDM, HCM employs the fast Walsh-Hadamard transformation (FWHT). HCM has higher performance gains than ACO-OFDM or DCO-OFDM at higher illumination levels. The performance improvement over RPO-OFDM is modest, however.

2.2.4 OFDM (MCM)

The category of OFDM splits into subcategories of its own, as mentioned before. While other subcategories have several different techniques, DCO-OFDM has none.

ACO-OFDM was proposed to solve the optical power efficiency problem, found in DCO-OFDM. Due to the DCO-OFDM technique having a DC-bias which does not carry information and is just used so that there is no dimming effects, it raises the power efficiency problem. Much like in DCO-OFDM, to turn the OFDM data into real values, Hermitian Symmetry is used. However, there is no DC bias in ACO-OFDM, and instead only the odd subcarriers are used. This results in halved spectral efficiency for ACO-OFDM compared to DCO-OFDM [31].

In ACO-OFDM, using the odd subcarriers eliminates the bipolarity of the signal. However, in U-OFDM, also known as Flip-OFDM, the bipolar is turned into unipolar with a different methodology. The OFDM symbols, when they are generated and being put into the OFDM frames, they are basically separated into two parts. These OFDM subframes, each carry either the positive values of the OFDM symbol, or the negative. This is done by clipping the negative values at

zero for the positive subframe, and clipping the positive values at zero for the negative subframe [32, 33].

DCO-OFDM, however, uses the DC biasing method to turn the bipolar signal into unipolar, so that it can be used in the VLC systems. DC biasing is done so that the negative values in the OFDM frame are turned into positive values, by adding a DC value to all the signal points. This results in lower optical power efficiency. However, the trade-off is that although it has lower power efficiency, DCO-OFDM has higher spectral efficiency and lower complexity than the other techniques [31].

There are many studies done on DCO-OFDM that focus on different parts of the technique. There are also several studies implementing DCO-OFDM in the literature. Most of the studies existing in the literature implement the modulation technique in Software Defined Radios (SDR). There are also some works that implement DCO-OFDM in FPGAs.

Wang et. al. propose a VLC system using a red LED, where DCO-OFDM is employed as the modulation technique [1]. The system includes an ethernet interface, a DAC and an ADC. The proposed system uses 16-QAM as the modulation mode, has a CP ratio of 1/16, has 1024 subcarriers. They claim that their design results in a 24 Mbit/s PHY (physical layer). The design was tested with a red LED, and due to the power of the LED being small, the transmission was done with a distance of 15 cm. However, even though they show the simple block diagram of their DCO-OFDM design, the design is not given in enough detail, it is unknown what devices (FPGAs, DAC and ADC) are used.

Fuada et. al. presents a System-on-Chip (SoC) architecture for OFDM for optical wireless communication (OWC) in different studies [2, 3]. The first study is implemented using the Arty Z-20 board which includes the XC7Z020 FPGA device of the Zynq-7000 variants. The second study, to continue the development on Zynq-7000 variants, is carried out using the ZYBO Development Board. In both studies, the Advances eXtensible Interface (AXI) bus is used extensively

to handle the communication between the FPGA devices and the PCs, and the communication between the subblocks of the OFDM system. Thus, their system includes a UART interface as well as an ethernet interface, a DAC and an ADC. The system implements different modulation modes, namely Binary Phase-Shift Keying (BPSK), Quadrature Phase-Shift Keying (QPSK), 16-QAM and 64-QAM. There are 64 subcarriers, the CP ratio is 1/8. The OFDM frame consists of 1 Preamble sequence and 1 OFDM symbol, both of which have $N + N_{CP} = 64 + 8$ data points. They use an 8-bit DAC and 12-bit ADC, with a sampling speed of 1 MSPS. Because of the low sampling speed, they find their resulting data rate low; the results of their experiment are that the data rates using an Analog-Front-End (AFE) are 14.4 kb/s, 33.6 kb/s and 71.9 kb/s for BPSK, QPSK, and 16-QAM modulation modes, respectively. Without the AFE, they can reach data rates of 198.39 kb/s, 396.76 kb/s, and 793.43 kb/s for BPSK, QPSK, and 16-QAM modulation. They claim according to their simulations that their design can achieve maximum throughputs of 6.67 Mb/s, 13.31 Mb/s, 26.51 Mb/s for BPSK, QPSK and 16-QAM, respectively [3].

Chapter III

IMPLEMENTATION

In this chapter, the implementation of the different parts of the DCO-OFDM Transceiver will be discussed. The design is comprised of 3 main parts: Ethernet implementation, DCO-OFDM implementation, DAC/ADC implementation. The Ethernet implementation has an Intellectual Property (IP) core that handles the communication with the computer, but there are two main implementations, referred to as Ethernet Receiver and Ethernet Transmitter, that handle the communication between the Ethernet IP and the DCO-OFDM implementation. There are several blocks within the DCO-OFDM implementation that carry out the different parts of the technique. Finally, the DAC/ADC implementation, which communicates with several IP cores, and connects the DCO-OFDM implementation and the DAC/ADC IP cores, and hence, the DAC and the ADC.

The DCO-OFDM modulation technique implemented in this thesis will be explained in Section 3.1. Then, in Section 3.2, the Ethernet implementation for this design will be explained; the Ethernet Receiver part and the Ethernet Transmitter part will be explained in their respective subsections. The DAC/ADC implementation is described in Section 3.3, with the DAC Controller and ADC Controller being described in their subsections.

In Section 3.4, the implementation of the DCO-OFDM Transmitter will be clearly described. This section is divided into its subsections where the different blocks of the technique are disclosed in detail. The DCO-OFDM Receiver implementation will be described in Section 3.5. The details of the blocks of this implementation are given in their respective subsections.

3.1 Design Overview

The DCO-OFDM implementation in this thesis is comprised of several blocks, each with their own task. This section explains the DCO-OFDM technique implemented, clearly defines the outlines of the blocks, shows the outline of the dataflow within the transceiver and explains the task of each of the blocks shortly.

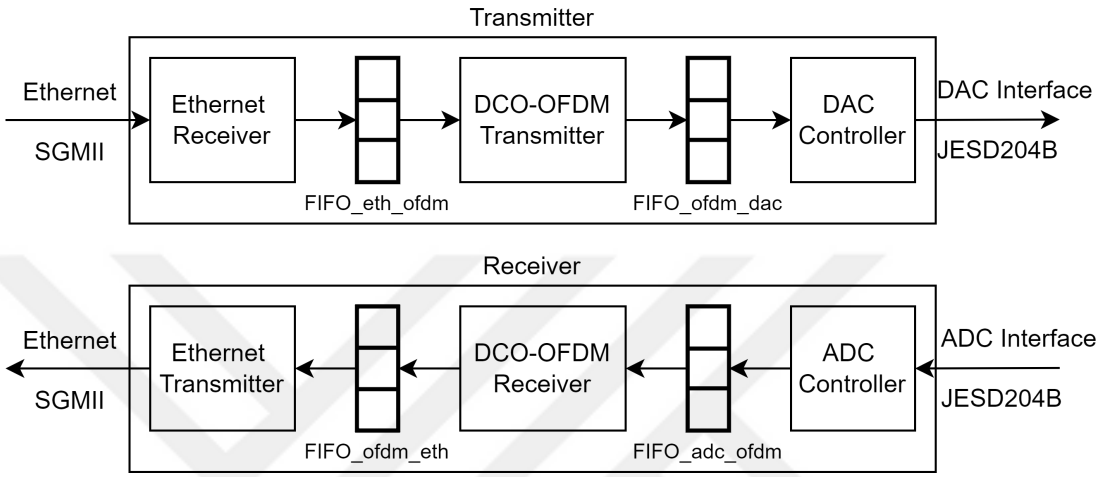

Figure 2: Block Diagram showing the top blocks of the Transceiver.

In Figure 2, the top blocks of the design can be seen. Both the Transmitter side and the Receiver side is placed on the same FPGA board, the Intel Arria10 SoC Development Kit, hence both designs can fit on the board. The Transmitter side is comprised of 3 top blocks: Ethernet Receiver, DCO-OFDM Transmitter and DAC Controller. The Receiver side is also divided into 3 blocks which are: ADC Controller, DCO-OFDM Receiver and Ethernet Transmitter. Each of the blocks here are connected to each other with a FIFO design between them, to make sure the data is not lost when the following block is not ready to read or the data is read correctly in the different clock domains, since different speed clocks can be used and are used in the designs.

3.1.1 Ethernet Overview

The Ethernet Receiver and Transmitter implementations are quite straightforward. The Ethernet Receiver receives the data from the “10/100/1000 Ethernet

MAC with 1000BASE-X/SGMII PCS" IP core. The implementation then adds a header in front of the packet and a footer at the end of the packet, and then writes it to a First-In-First-Out (FIFO) memory design, called FIFO_eth_ofdm, which is read by the DCO-OFDM Transmitter whenever data is available. The Ethernet Transmitter side is slightly more tricky as it needs to find out where the header and footer lies within the incoming signal. And to make sure that the header and footer can be found even when the Bit Error Rate (BER) is high within the system due to noise, there has to be a threshold of tolerance for the comparison between the actual header or footer and the received signal.

3.1.2 DAC/ADC Overview

The DAC and ADC Controller implementations communicate with several IP Cores that are found in the example design given by the DAC/ADC board provider. The DAC/ADC board used in this thesis is the FMCDAQ2-EBZ board. The DAC Controller is again straightforward in a way; when an OFDM frame is ready to be sent, meaning when every sample of the OFDM frame is calculated and written to the FIFO before the DAC Controller, called FIFO_ofdm_dac, the DAC Controller starts reading from the FIFO and sending the signal through the JESD204B Interface to the DAC. In the same way, the ADC Controller takes the ADC samples from the JESD204B Interface and then writes the samples to the FIFO before the DCO-OFDM Receiver reads the data.

3.1.3 DCO-OFDM Overview

The thesis implements both the DCO-OFDM Transmitter and the Receiver. The DCO-OFDM Transmitter and Receiver algorithms are explained in this subsection, before going into the implementations in Section 3.4 and Section 3.5.

In Algorithm 1, the simplest version of the algorithm can be seen. Each line corresponds to one block within the DCO-OFDM implementation. After the data coming through the ethernet connection is written to FIFO_eth_ofdm, the DCO-OFDM Transmitter checks whether the FIFO is empty or not, when it is empty,

it waits for data to arrive; when it is not empty, it starts reading the data and modulates it. First, the data is modulated with Quadrature Amplitude Modulation (QAM) in the desired modulation mode. The implementation already has QPSK (or 4-QAM), 16-QAM and 256-QAM ready to be used if required. Then, the modulated data is assigned to the subcarriers with Hermitian Symmetry, so that the output of the IFFT is real. Then, this assigned data is given to the IFFT. Cyclic Prefix (CP) is added to the data before the OFDM Preamble and the Channel Estimation Training Sequence (CETS) is added. Then that packet of data is upsampled and filtered with a predetermined Pulse Shape (PS), and if the FIFO between DCO-OFDM Transmitter and DAC Controller is not full, the signal is written to the FIFO. In the DAC Controller side, when the frame is fully written to the FIFO, the signal is then output from the DAC.

Algorithm 1 DCO-OFDM Transmitter

```
1: while 1 do
2:     if ¬FIFO_eth_ofdm.empty then
3:         symbols ← modulateQAM(inputs)
4:         assignedData ← assignData2Subcarriers(symbols)
5:         IFFTedData ← IFFT(assignedData)
6:         cpInsertedData ← insertCP(IFFTedData)
7:         OFDMFrame ← addPreambleAndCETS(cpInsertedData)
8:         upsampledFrame ← upsampleNFilter(OFDMFrame)
9:         if (upsampledFrame.isCalculated())&(¬FIFO_ofdm_dac.full) then
10:            data2Transmit ← upsampledFrame
```

For the modulation method, Quadrature Amplitude Modulation (QAM) was chosen. For the design; QPSK (4-QAM), 16-QAM and 256-QAM were implemented in Verilog HDL. This modulation is straightforward. Left half of the bits of the inputs are coded in the real part of the symbols and the right half of the bits are coded in the imaginary part in a Cartesian map with the constellation points. Then, these symbols are divided by the square root of the average energy of the symbols (E_{QAM}) to normalize the generated symbols. The QPSK (4-QAM) modulation algorithm can be seen in Algorithm 2.

Finding the constellation points for QPSK, 16-QAM, and 256-QAM are quite

simple. The indices of the points are varied so that the indices of the two adjacent points have only one bit difference. The constellation maps in QPSK and 16-QAM are provided in Figure 3 and Figure 4.

Algorithm 2 Modulator (4-QAM)

```
1:  procedure MODULATEQAM(inputs)
2:      if inputs[1] = 0 then                    ▷ Modulation of the real part
3:          symbolReal ← -1
4:      else
5:          symbolReal ← 1
6:      if inputs[0] = 0 then                    ▷ Modulation of the imaginary part
7:          symbolImag ← -1
8:      else
9:          symbolImag ← 1
10:     energyOfSymbols ← calculateEnegry()
11:     sqrtEnergy ← sqrt(energyOfSymbols)
12:     symbols ← [symbolReal/sqrtEnergy, symbolImag/sqrtEnergy]
```

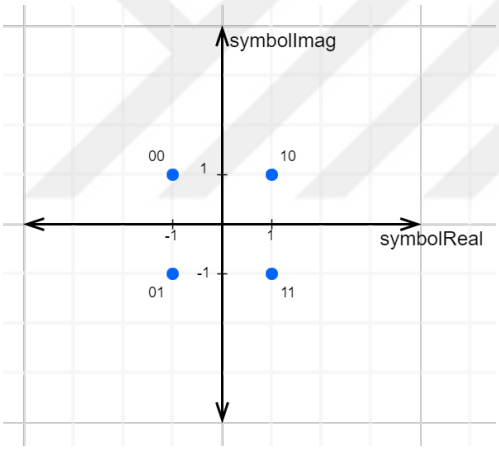

Figure 3: Digital QPSK (4-QAM) showing the indices of constellation points.

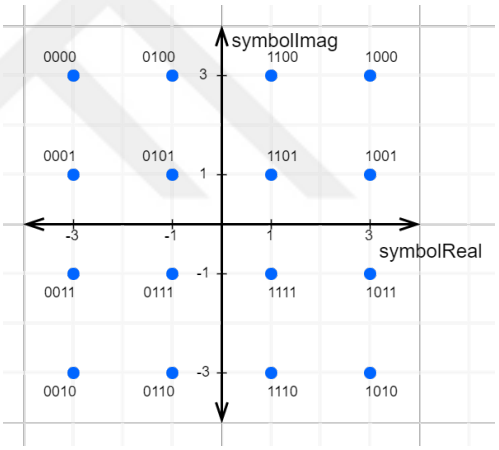

Figure 4: Digital 16-QAM showing the indices of constellation points.

The modulated data is then assigned to the subcarriers. Since this OFDM design is to be used in a VLC System, the result of the IFFT must be real. To achieve this, when assigning data to subcarriers, Hermitian Symmetry is applied to the data. The assigned data with Hermitian Symmetry can be seen in Equation (1). The "S" in the equation refers to the modulated data coming from the Modulator, and the "N" refers to the number of subcarriers for this system.

$$assignedData = [0, S_1, S_2, ..., S_{N/2-1}, 0, S^*_{N/2-1}, ..., S^*_2, S^*_1][32] \quad (1)$$

This equation is also illustrated as an algorithm in Algorithm 3. The chosen number of subcarriers in this design is 64. The first subcarrier gets 0, the rest of the first half of the subcarriers get (N/2)-1 symbols, which is 31 symbols. The second half starts with 0, the rest are the complex conjugates of the same symbols from the first half, but in reverse order.

Algorithm 3 Data Assigner

```
1: procedure assignData2Subcarriers(symbols)
2:     assignedData[0] ← 0
3:     assignedData[(N/2)-1:1] ← symbols[(N/2)-2:0]
4:     assignedData[N/2] ← 0
5:     assignedData[N-1:(N/2)+1] ← complexConj(symbols[0:(N/2)-2])
6:                                  ▷ N = Number of Subcarriers = 64
```

After the Hermitian Symmetry is applied, the resulting data is then supplied to the IFFT. To achieve this with a low latency and low area usage, an IP Core is used in the implementation.

A Cyclic Prefix (CP) must be added to the data to prevent interference between OFDM symbols and to ensure the circular convolution with the channel [32]. The decided length of CP, which is referred to as N_{cp} in Equation (2), is 16. The data from the last 16 subcarriers are copied and appended to the start of the data. The resulting signal is given by the following equation.

$$cpInsertedData = [X_{N-N_{cp}}, ..., X_{N-1}, X_0, X_1, ..., X_{N-1}][32] \quad (2)$$

After the Cyclic Prefix is added to the OFDM symbols, the symbols are appended at the end of the Preamble and the CETS. For this thesis, 2 OFDM symbols were chosen. The Preamble is a Pseudo Random Binary Sequence (PRBS) that is used for Frame Detection on the receiver side. It allows the Receiver to find the start of the frame so that it can decipher the data. It was chosen to have a length equal to one OFDM Symbol. The CETS is the following sequence after the

Preamble that is used to calculate the noise from, and to correct the data using this calculated noise. Before the 2 OFDM symbols are added to the Preamble and the CETS, the data is multiplied with the square root of average electrical transmit power.

Then, this packet is upsampled by a factor of 20, and filtered with Root Raised Cosine Pulse Shape. This procedure is also called Pulse Shaping or Spectral Shaping in the literature. The Pulse Shape that was used in this implementation can be seen in Figure 5.

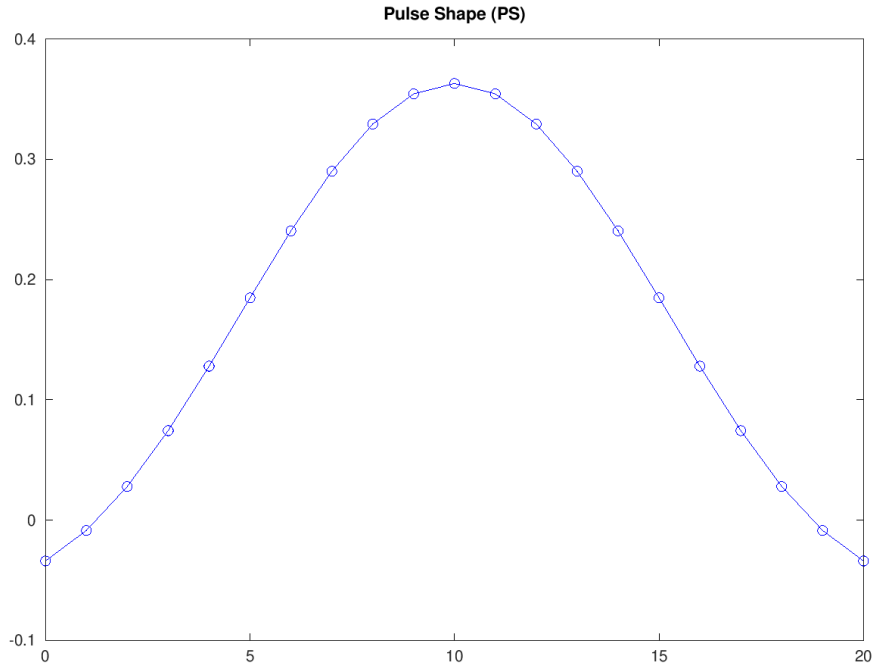

Figure 5: The Root Raised Cosine Pulse Shape used in this design.

The OFDM frame that is upsampled and pulse shaped is then written to FIFO_ofdm_dac, from which this data is read and sent to the DAC board via the JESD204B Interface. The structure of the OFDM frame before the upsamling and pulse shaping can be seen in Figure 6.

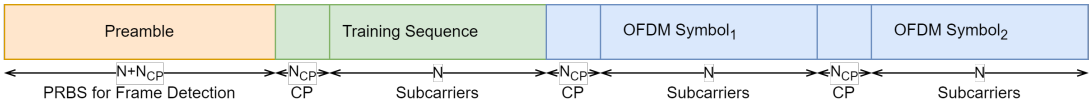

Figure 6: The frame structure of the transmitted data (before upsampling and pulse shaping).

On the Receiver Side, the algorithm is more complex. The simplest version can be seen in Algorithm 4. The signal read from the channel by the ADC arrives at the DCO-OFDM Receiver, then the first block finds the peaks and valleys of the signal and downsamples it. This downsampled data may have an OFDM frame, but the receiver does not yet know if there is one. To find the Frame, Frame Detection is carried out. When the Frame is detected, after removing the Preamble, the rest of the OFDM frame is sent for the CP to be removed. CP is removed from both the CETS and the OFDM symbols. The resulting data is then taken through FFT. Afterwards, the data is reconstructed with respect to the CETS. The reconstructed data is demodulated using the constellation points from Quadrature Amplitude Modulation (QAM), with Maximum Likelihood (ML) decoding.

Algorithm 4 DCO-OFDM Receiver

```
1: while 1 do
2:     downsampledData ← findPeakNDownsample(adcdata)
3:     ofdmSymbols ← detectFrame(downsampledData)
4:     cpRemovedData ← removeCP(ofdmSymbols)
5:     FFTedData ← FFT(cpRemovedData)
6:     reconstructedData ← reconstructData(FFTedData)
7:     demodulatedData ← demodulateQAM(reconstructedData)
8:     if (¬FIFO_ofdm_eth.full) then
9:         data2Transmit ← demodulatedData
```

In Peak Finding and Downsampling, since the peaks are supposed to occur every 20 data points, which is the upsampling factor (USF), 20 different sums are calculated, each of them are the sum of the squares of every other 20^{th} data point. That is, 1^{st}, 21^{st}, 41^{st} and so on are squared and added to sum1; 2^{nd}, 22^{nd}, 42^{nd} and so on are squared and added to sum2, and so on. Then, these 20 sums

Algorithm 5 Peak Finder and Downsampler

```
1: procedure findPeakNDownsample(adcdata)
2:     datalen ← length of adcdata
3:     for i = 0; i < datalen; i++ do
4:         sum[i % USF] ← sum[i % USF] + adcdata[i] * adcdata[i]
5:     [max, maxIndex] ← max(sum)
6:     downsampledData ← adcdata[maxIndex:USF:datalen]
7:                                          ▷ USF = Upsampling Factor = 20
```

are compared and the largest one is found. The index of the largest sum is the starting index of the peaks, each one 20 data points apart. The algorithm for this part can be found in Algorithm 5.

Frame Detection can be difficult when the Signal-to-Noise Ratio (SNR) is low due to a dispersive channel. Therefore, a good method for Frame Detection must be present. For this part, Cross-correlation method which is given in Equation (3) is used. Cross-correlation is commonly used in searching for a shorter, known signal in a longer signal. To achieve this, one specific property of Cross-correlation is used. Like in Convolution Theorem, Cross-correlation satisfies Equation (4).

$$(f \star g)(n) = \sum_{m=-\infty}^{\infty} \overline{f[m]} * g[m+n] \tag{3}$$

$$\mathcal{F}\{f \star g\} = \overline{\mathcal{F}\{f\}} \cdot \mathcal{F}\{g\} \tag{4}$$

In Equation (4), the complex conjugate of the Fourier Transform of the signal must be calculated. This can be done easily, using the Conjugation property of Fourier Transform shown in Equation (5).

$$\overline{\mathcal{F}\{f(t)\}} = \mathcal{F}\{\overline{f(-t)}\} \tag{5}$$

In the Frame Detection Algorithm, to find an array of numbers which represent the magnitude of correlation, this property is used. The FFT of the Preamble is calculated, the FFT of the incoming signal is calculated, and then the element-wise product of these two results is calculated. This result is equivalent to the FFT of the Cross-correlation of the Preamble and the signal. Hence, the IFFT of

the element-wise product is calculated, to get the Cross-correlation results. The maximum of this array of numbers is then found, the index of which is the last index of the Preamble. Using this, the rest of the OFDM frame is output so that the CP can be removed. The algorithm for the Frame Detection can also be seen in Algorithm 6. CP Removal is a straightforward process in which the first $N_{\text{CP}} = 16$ of the data points in every $N + N_{\text{CP}} = 80$ is discarded, and the rest of the data are sent to the FFT part. The FFT of the data is then calculated and CETS symbol and the 2 OFDM symbols in the packet are sent to the Data Reconstructor.

Algorithm 6 Frame Detector

1: **procedure** DETECTFRAME(downsampledData)
2: *FFTofPreamble* ← FFT(*preamble*)
3: *FFTofSignal* ← FFT(*downsampledData*)
4: *FFTofCorrelations* ← ElementwiseProduct(*FFTofPreamble*, *FFTofSignal*)
5: *correlations* ← IFFT(*FFTofCorrelation*)
6: *[max, index]* ← max(*correlations*)
7: *ofdmSymbols* ← *downsampledData[index+1:index+(*N_{OFDM}*+1)*(N+*N_{CP}*)]*
8: ▷ N_{OFDM} = Number of OFDM Symbols = 2
9: ▷ N = Number of Subcarriers = 64
10: ▷ N_{CP} = Length of CP = 16

In Data Reconstruction, the CETS received from the channel is crucial. Using the received CETS and the actual CETS, the Estimated Effective Channel Coefficient (EECC) is calculated. Then, the data is reconstructed using this coefficient. The received OFDM symbol is divided by the EECC, since the result of this still has the data from all 64 subcarriers, the two half can be used to calculate the reconstructed data more accurately. The reverse of the second half is taken and the complex conjugate is calculated, then the average of the first half and the second half is calculated. Then, after multiplying this result with the square root of the energy of the symbols (E_{QAM}) in the constellation map, the data is sent to the Demodulator. The algorithm for the Data Reconstruction part is also shown in Algorithm 7.

Algorithm 7 Data Reconstructor

```
1: procedure reconstructData(FFTedData)
2:     receivedTS ← FFTedData[0:N])
3:     estEffChCoeff ← receivedTS/realTS          ▷ Estimated Effective Channel Coefficient (EECC)
4:     for i = 1; i < (N_OFDM + 1); i++ do
5:         recOFDMSymbol ← FFTedData[i*N:(i+1)*N])
6:         recSubcarrs ← recOFDMSymbol/estEffChCoeff
7:         recSymbols ← recSubcarrs[1:(N/2)-1]
8:         recSymbols ← (recSymbols + conj(recSubcarrs[N:-1:(N/2)+1]))/2
9:         energyOfSymbols ← calculateEnegry()
10:        recSymbols ← recSymbols * sqrt(energyOfSymbols)
11:        reconstructedData ← [reconstructedData, recSymbols]
12:                                   ▷ N_OFDM = Number of OFDM Symbols = 2
13:                                   ▷ N = Number of Subcarriers = 64
```

The data is demodulated using this constellation map, using Maximum Likelihood (ML) coding. The resulting data is then written to FIFO_ofdm_eth if the FIFO is not full. The Maximum Likelihood coding is carried out by finding the closest actual constellation point in the constellation map to the received data. In Figure 7, some received data can be seen as an example. The red squares represent the area for each of the constellation points, data within which are equivalent to that constellation point. The binary values in the corners of each red square represent what the output of demodulation should be when the incoming data is within that square.

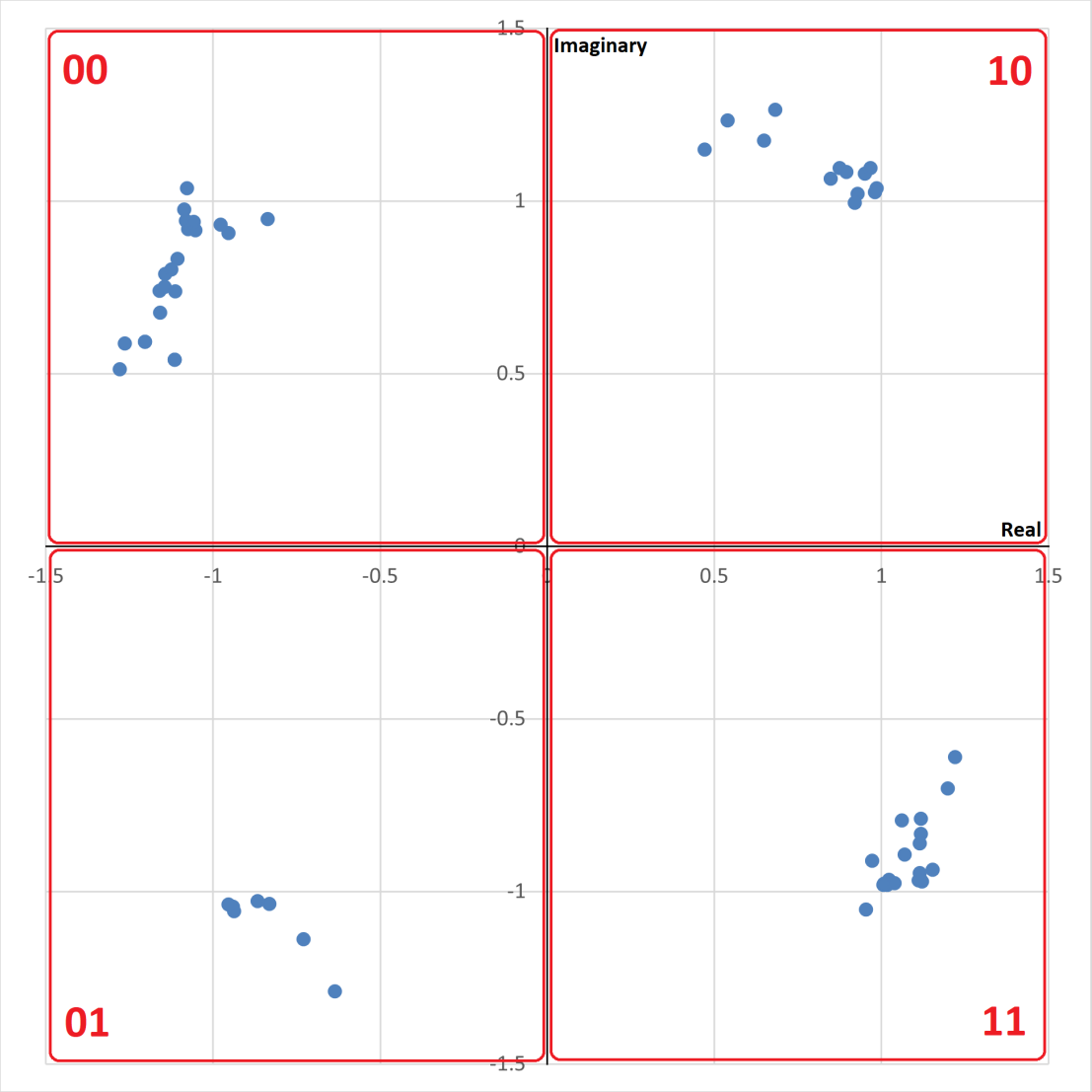

Figure 7: Example of received data before demodulation on QPSK (4-QAM).

In the next sections, the implementations of the different parts of the DCO-OFDM Transceiver will be described in detail.

3.2 Ethernet Implementation

In this section, The Ethernet Implementations, namely the Ethernet Receiver which receives the ethernet packets from the transmitting PC and the Ethernet Transmitter which transmits the ethernet packets to the receiving PC will be discussed.

3.2.1 Ethernet Receiver

The transmitting and the receiving PCs are connected to the FPGA via ethernet cables. The 10/100/1000 Ethernet MAC with 1000BASE-X/SGMII PCS IP core enables the communication between the FPGA and the PC. However, the IP core must be controlled so that the ethernet packets are sent in the format that can be modulated by the DCO-OFDM Transmitter, and that the packets can be found within the data resolved by the DCO-OFDM Receiver.

The Ethernet Receiver receives the ethernet packets from the PC using the Ethernet IP core. After the DCO-OFDM Receiver, the received and resolved data must be converted back into the ethernet packets. However, in a continuous stream, it can be difficult to find the starting point of the ethernet packet. To be able to find the packet, at the start of each packet, 24 bytes are added as a header. And to be able to find the end of the packet, 12 bytes are added. In this module, when there is an ethernet packet incoming, the 24 bytes are sent to the FIFO. Then, the incoming ethernet packet is sent to the same FIFO, after which the 12-byte-long footer is sent.

The secondary control logic then connects this FIFO with the DCO-OFDM Transmitter. When there is data in the FIFO, it tells that to the Transmitter. When the Transmitter requests data, the logic reads the data from the FIFO, and according to the modulation (QPSK, 16-QAM, 256-QAM), divides the data into smaller parts. For 256-QAM the Modulator requires 8-bit inputs, whereas for QPSK, the inputs are 2 bits. The 8 bits from the FIFO are divided into 2 bits for QPSK, and then sent to the Transmitter one by one. After all 8 bits are sent, the next data is read from the FIFO, until there is none left.

3.2.2 Ethernet Transmitter

At the Receiver side of the Transceiver, the Ethernet Transmitter Implementation carries out the task of transmitting the received DCO-OFDM transmission to the receiving PC. As mentioned in the previous section, the FPGA and the receiving

PC are connected via an ethernet cable. The IP core enables the communication. The Ethernet Transmitter Implementation enables the control of the IP core so that the received transmission can be sent to the receiving PC. The implementation is comprised of two parts: Packet Finder and Packet Storer. The Packet Finder module is a more complex version of a pattern matcher that finds the approximate matches within the resolved OFDM transmission so that it can find the approximate header and the approximate footer to send the ethernet packet. This module must be able to handle approximate matches due to the possible bit errors caused by the channel. The Packet Storer module is, as its name suggests, stores the ethernet packet data until the footer is found by the Packet Finder, and sends the packet to the receiving PC.

Packet Finder module, as a complex pattern matcher, must be able to find both the header and the footer. The header is 24 bytes, and the footer is 12 bytes. The ethernet data are 8-bit values coming from the DCO-OFDM Receiver, which are compared against the header and then the footer, however; due to the bit error(s) caused by a dispersive channel, the simplistic approach of a pattern matcher that attempts to find the exact matches within the resolved OFDM transmission is ineffective. With only 1-bit error within the header or the footer, the Packet Finder module would be unsuccessful. Thus, an approximating pattern matcher must be employed. Since the header is 24 bytes and the footer is 12 bytes, pattern matching by 12 bytes would be useful to optimize the module, using fewer registers. The Packet Finder module takes 12-byte inputs, then moves on to calculating the correlation between the received 12 bytes and the first 12 bytes of the header. The correlation is done by calculating the bitwise XOR of each byte. Then, the number of 1s, which are the number of different bits, are found, after which the module moves on to attempt at finding the second part of the header if the number of different bits is less than the number of bits specified as a threshold. The threshold is decided to be 20 bits within the 12 byte frame, which is approximately 20%. If the number of different bits is more, then it is decided that this part of the

resolved OFDM transmission is not the first part of the header. When the first part is found, then the second part of the header is attempted to be found, in the same way. Afterwards, all incoming data could be the footer, therefore the attempt at finding the footer begins. As long as the footer is not found, meaning that the number of different bits is more than the threshold, then the received data is sent to the Packet Storer for safekeeping. When the footer is found, the signal that dictates the packet has ended is sent to the Packet Storer, so that it can send the packet to the receiving PC.

Packet Storer module, as mentioned, stores the resolved OFDM transmission which are determined to be the ethernet data. This is done using a FIFO within the module. The FIFO has 8-bit wordsize and 1024 locations. When the Packet Finder sends the ethernet data, the module saves them to the FIFO, and after the Packet Finder finds the footer, the data saved to the FIFO is sent to the Ethernet IP core. In the case that the Packet Finder is not able to find the footer and the FIFO within the Packet Storer is full, then the Packet Storer empties the existing ethernet data to the Ethernet IP core.

3.3 DAC/ADC Controller Implementation

In this section, The DAC Controller and the ADC Controller Implementations will be discussed.

3.3.1 DAC Controller

DAC Controller logic connects the DCO-OFDM Transmitter to the DAC, enabling the OFDM frames to be sent through the channel to the DCO-OFDM Receiver. The DAC used in this thesis is part of the AD-FMCDAQ2-EBZ Evaluation Board. The board has a quad DAC with 16-bit resolution and supports quad or dual DAC mode up to 2.8 GSPS (Gigasamples per second).

After the DCO-OFDM Transmitter receives ethernet data and generates an OFDM frame, the frame must be sent through the channel. However, the output of the DCO-OFDM Transmitter is in the IEEE-754 Floating Point format, which

are 32-bit values. Since the DAC has a resolution of 16 bits, this format must be changed. Hence, a Floating Point to Fixed Point Converter is added to the design. The Converter takes each 32-bit values that are in the IEEE-754 Floating Point format, and converts them to 16-bit Fixed Point values. The leftmost bit is the sign bit, and the rest are the magnitude. After plotting the output of the DCO-OFDM Transmission, it is seen that all points are less than 0.5, which means that all magnitude bits can be set to represent the fraction part. And since all values are less than 0.5 and more than -0.5, even the leftmost bit of the fraction can be eliminated, so that the magnitude part can represent values from 0 to $\approx$0.49999. The output of the DCO-OFDM Transmitter is directly connected to the converter, the output of which is then saved to a FIFO. This FIFO fills up as the OFDM frame is transmitted from the DCO-OFDM Transmitter.

The OFDM frame is made up of the Preamble, the CETS, and 2 OFDM symbols. The Preamble, the CETS and each OFDM symbol has a length of 80, which results in 320 data points. In Upsampling and Filtering, the length is multiplied by 20, which is the upsampling factor. This results in an OFDM frame of length $320 * 20 = 6400$. Before this amount of data points are saved to the FIFO, the DAC Controller does nothing. When all 6400 points are written to the FIFO, the data points are read one by one, and they are sent to the DAC. Every time that an OFDM frame is ready to be sent, meaning that when all 6400 points of an OFDM frame is ready, the frame is sent to the DAC.

The DAC and ADC Controllers connect to the DAC using a large collection of IP cores. The DAC Controller sends the data to the DAC over the JESD204B Interface. JESD204B Interface is a standardized serial interface that was created through the JEDEC Committee to reduce the number of data inputs/outputs between high-speed data converters and other devices, such as FPGAs (Field-Programmable Gate Arrays) [34]. The JESD204B Interface enables the user to send multiple data to the DAC and the ADC. The Interface receives 4 sample data at the same time at a slower clock rate, and then sends them one by one

with a faster clock rate, using the clock rate of the DAC. For example when the DAC has a clock frequency of 1 GHz, the JESD204B Interface receives the data with a clock rate of 250 MHz. The clock rate of the DAC and the ADC can be set to several different values using the initialization script for the board. Some examples to change the script to get the DAC and ADC to work at different speed rates are given in [35].

DAC and ADC clock rates must be the same so that the samples match. This means that the JESD204B clock rates connecting to the DAC Controller and the ADC Controller must be the same. However, because of the downsampling in the DCO-OFDM Receiver, the JESD204B must have a lower frequency than the one the Downsampler uses, otherwise the FIFO between them overflows. For the Downsampler, a clock frequency of 270 MHz is found within the Arria10 SoC Development Kit. Hence, a clock frequency of 250 MHz is used.

The JESD204B can receive 4 samples at the same time, as mentioned above. However, the Upsampler and Filter outputs the data serially, and the data can be read from the FIFO serially. Hence, the same data point from the OFDM frame is sent to the JESD204B Interface 4 times, in parallel.

3.3.2 ADC Controller

ADC Controller logic connects the ADC to the DCO-OFDM Receiver. The samples coming from the ADC are sent to the DCO-OFDM Receiver to be resolved and sent to the receiving PC. The ADC used in this thesis is part of the AD-FMCDAQ2-EBZ Evaluation Board, as mentioned in the previous subsection. The board has a dual ADC, with 14-bit resolution and supports sampling up to 1.0 GSPS.

The data arrives to the ADC Controller through the JESD204B Interface, as mentioned in the previous subsection. The JESD204B Interface is chosen to have a clock frequency of 250 MHz, and the ADC data comes to the ADC Controller in parallel, 4 data at each clock period. However, at the transmitter side, the same data is sent parallelly, so the 4 data sampled here are equivalent to one another.

They may differ slightly due to channel noise, but since all 4 of them are essentially the same values, 3 of them can be discarded and the design can work with only one. Thus, one of each 4 ADC values received are used.

The received values are in the 16-bit Fixed Point format, however, since the ADC has a resolution of 14 bits, the leftmost two bits must be discarded. Then, using a Fixed Point to Floating Point Converter, the 14 bits are converted to the IEEE-754 Floating Point format. This is done because the DCO-OFDM takes 32-bit Floating Point inputs. After the conversion is done, the output is then saved to a FIFO which precedes the DCO-OFDM Receiver. The output of the FIFO is connected directly to the DCO-OFDM Receiver, which is connected to the Downsampler within the Receiver. The Downsampler reads the data from the FIFO and after the downsampling, sends the results to the Frame Detector. However, during the calculation phase of the downsampling, the received ADC data are not being taken as input to the DCO-OFDM Receiver. This is the main reason that the FIFO exists between the ADC Controller and the DCO-OFDM Receiver.

3.4 DCO-OFDM Transmitter Implementation

In this section, DCO-OFDM Transmitter Implementation of this thesis will be described in detail. In the following subsections, the different modules within this implementation will be explained thoroughly.

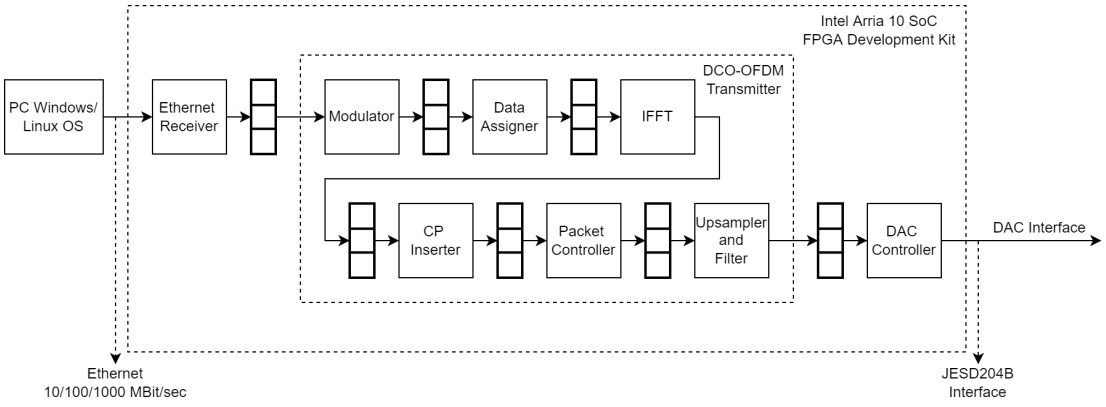

Figure 8: The block diagram of the DCO-OFDM Transmitter Implementation.

In Figure 8, the block diagram of the DCO-OFDM Transmitter can be seen. The Ethernet Receiver and DAC Controller are also in the block diagram so that the dataflow can be observed more clearly. Between each module, there are FIFO designs, so that no data is lost when one module is outputting signals and the other is not yet ready to receive them. This design introduces more simplicity and independence for each module. Each module can work without being dependent on the other modules, as long as there is data within the preceding FIFO and there is enough space in the following FIFO. Adding all of these FIFO designs also let the modules work with different clock domains, meaning every module can work at different clock rates.

3.4.1 Modulation

The implementation of the Modulator is straightforward since it is a simple mapping procedure. The Modulator is designed in a simplistic way such that it reads data and returns the result of the mapping immediately. In Algorithm 2, it can be seen that the mapping has multiple steps. The input is taken, the constellation point is located within the map and the real and imaginary value pair is found. Then, the average energy of the symbols (E_{QAM}) for the modulation type is calculated. Then, after taking the square root of this value, both the real and the imaginary part are divided by it. The pair is then returned.

The design could do these operations individually, but it can be optimized. For example, the E_{QAM} is only different for different types of modulation, QPSK, 16-QAM, 256-QAM and so on. With a chosen modulation, this value is a constant. The square root of this is also a constant. Hence, this constant value could be introduced to the design precalculated, so two operations can be omitted. The only two operations left would be the mapping and the division. However, the divisor is the constant that is precalculated; and there are only 2*sqrt(M) many different values the real and imaginary parts can be, M being the modulation type (QPSK=4, 16-QAM=16, 256-QAM=256). Since the divisor is a constant, there are the same number of different quotients as the number of dividends.

Therefore, it is more logical to map the inputs of the Modulator to the quotients instead of the actual constellation point pairs. This eliminates the need for the division. The Modulator becomes a simple if-elseif-else statement that outputs a value conditioned by the input.

This simple logic is a combinational one, and does not need a clock. As long as there is an input, there can be an output. There is only a need for a signal to ask to read from the preceding FIFO, and another to write the result to the following FIFO. By checking if the preceding FIFO is not empty and the following FIFO is not full, the Modulator can read from the preceding FIFO and write to the following one. The schematic for this module is shown in Figure 9.

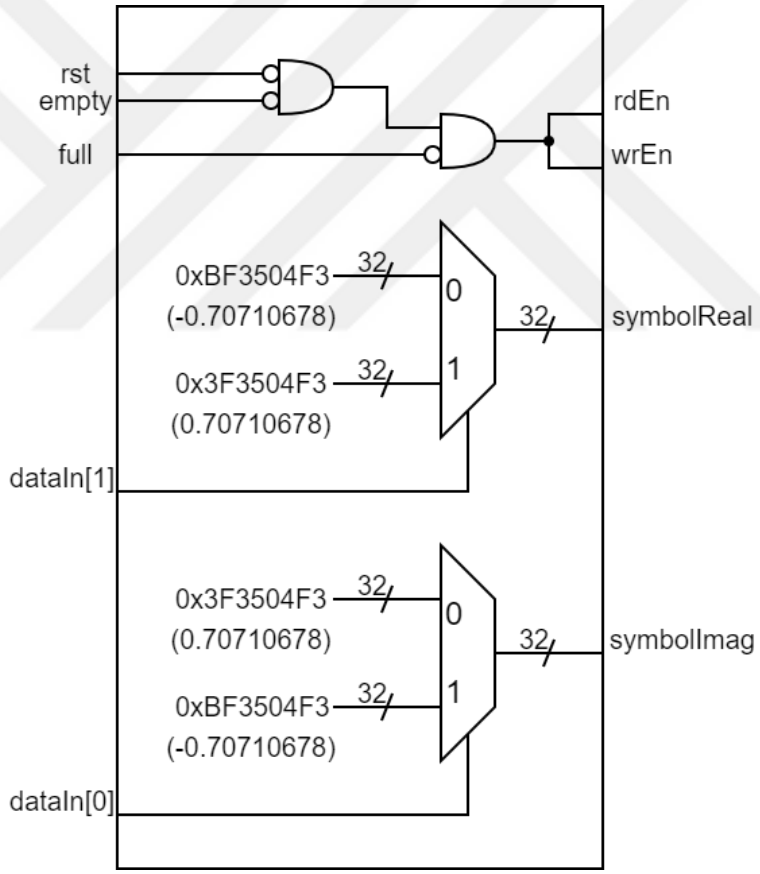

Figure 9: The schematic of the Modulator (QPSK).

3.4.2 Assigning Data to Subcarriers

Data Assigner module, as its name suggests, assigns the incoming data to the subcarriers. For this design, 64 subcarriers were chosen, therefore at most 64

modulated symbols can be placed to the subcarriers for one OFDM symbol.

That is generally what is done in non-VLC OFDM Implementations. However, since this is a VLC System, the output of the OFDM Transmitter must be real and positive. To be able to achieve that, DC Bias must be added to the output signal for the signal to be positive. To make it real, the output of the IFFT module must be made real. To achieve this, Hermitian Symmetry is used. Hermitian Symmetry is explained in Section 3.1.3. In Equation (1) and in Algorithm 3, the format of the signal is shown.

Since the incoming data is being read by this module from a FIFO, and the data is being read serially, one modulated symbol comes every clock cycle. Then, it is logical that the output is also serial, so that the following modules can also start their procedures. For 64 subcarriers with Hermitian Symmetry, there are (N-2)/2 inputs to the Data Assigner. The Data Assigner must take 31 modulated symbols, assign them to the subcarriers and output them. The first and (N/2)th subcarriers have "0" assigned, the subcarriers from the second to ((N/2)-1)th subcarriers have the same data as the input, and from ((N/2)+1)th to Nth subcarriers have the complex conjugate of the input data, in reversed order. The schematic for this module can be seen in Figure 10. The rdEn and wrEn signals are omitted for a better understanding of the module.

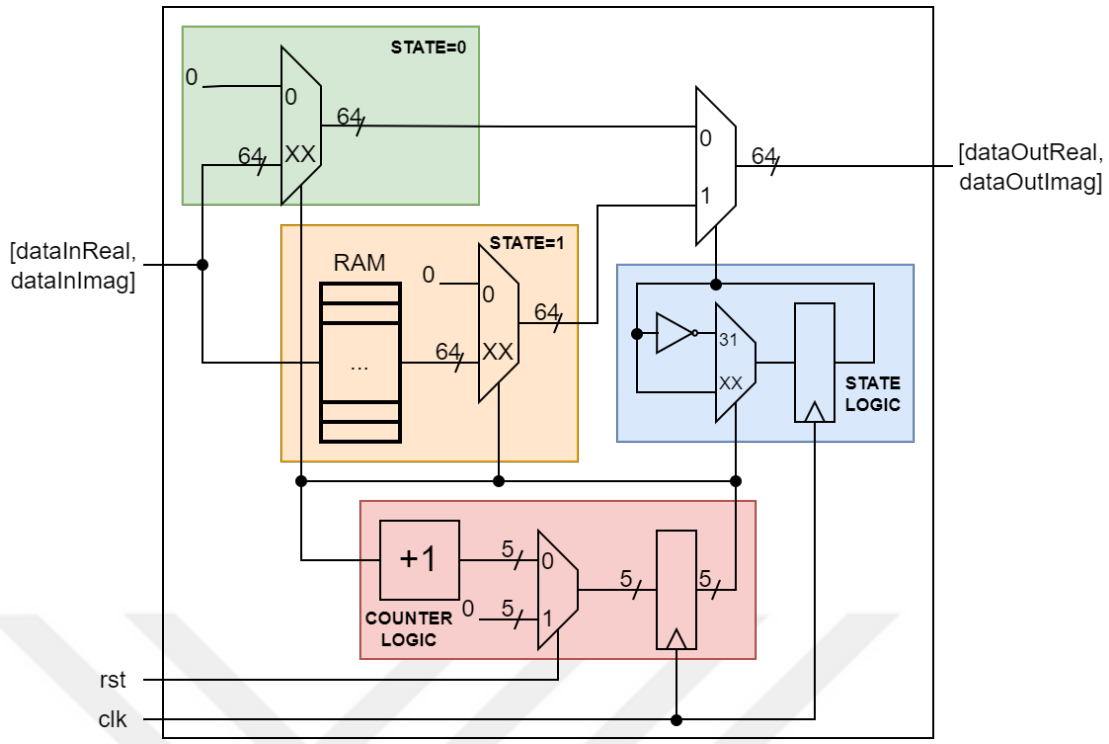

Figure 10: The schematic of the Data Assigner.

3.4.3 IFFT

This module uses the FFT IP core from the DSP/Transforms category that can be found in the IP Catalog. The FFT IP has both FFT and Inverse FFT that can be generated. Since there are 64 subcarriers, IFFT length must be 64. The inputs are in 32-bit IEEE-754 Floating Point format. There are 3 options for the direction of the FFT IP core, forward, reverse and bi-directional. Reverse is chosen to use the Inverse FFT and eliminate resources that would not be used in bi-directional. This option is only valid for Variable Streaming Data Flow and Single Floating Point representation with hard floating point blocks disabled. Variable Streaming Data Flow must have the data input order in Digit Reverse order and the data output order in Natural order. Digit Reverse order is where the data is scrambled so that the index of the input data is Digit Reversed, where each 2 bit is reversed. So, in an IFFT with 64 inputs, since there must be 6 bits to index those inputs, then $index_{\text{IFFT}} = \{index_{\text{data}}[1:0], index_{\text{data}}[3:2], index_{\text{data}}[5:4]\}$.

The IFFT IP core therefore requires a top module which can control the inputs

and outputs. Thus, the IFFT top module has the IFFT IP core and the controller logic. Within this module, when there is data, it is read and saved to a memory of size 64 since there are 64 subcarriers. This memory is a 1 port read and 1 port write Random Access Memory (RAM). When all 64 data are saved, the data is then sent to the IFFT IP core with the digits reversed. During this part, if there is still data in the preceding FIFO, there is a second memory that the incoming data is saved to. The IFFT function in MATLAB and the IFFT IP core in the design differ slightly, the difference is that the output of the IFFT core is equivalent to the output of the function multiplied by 64. However, in the MATLAB code, the result of the IFFT is multiplied by the square root of $N = 64$, and the top module can just output the data that is equivalent to this result. Hence, instead of dividing the output of the IFFT IP core by 64, it is divided by $64/sqrt(N) = 8$. This division can be done simply with subtraction, because of the IEEE-754 Floating Point format. The 32 bit value has 3 parts; the 31st bit is the sign bit, from 30th to 23rd bit is the exponent and from 22nd to the 0th bit is the mantissa. The format is simply $sign*2^{exponent}*mantissa$. To divide the values by 8, all that needs to be done is simply subtracting $log_2 8 = 3$ from the exponent. Because the IFFT IP core has the output dataflow as "natural", the output is directly connected to the output of the IFFT top module after it is divided by 8, which is connected to the following FIFO. The schematic for the IFFT top module can be seen in Figure 11.

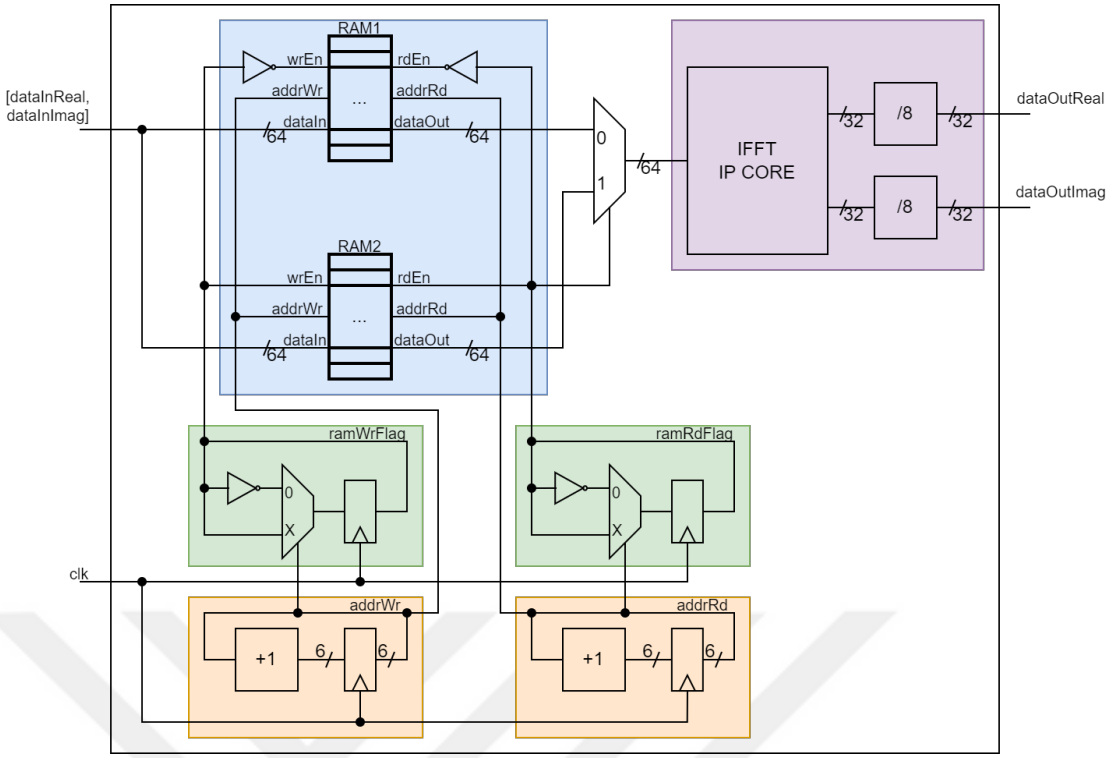

Figure 11: The schematic of the IFFT top module.

3.4.4 Cyclic Prefix (CP) Insertion

Cyclic Prefix (CP) is chosen to have a length of $N_{\text{CP}} = 16$. The data within the CP is the last N_{CP} data from the subcarriers. Hence, the input to this module is of length N, and the output is of length $N+N_{\text{CP}}$. This module takes N many inputs and saves it to a memory. Then, it outputs the last N_{CP} data within the memory, and then goes back to the start of the memory and outputs all N many data. Then the module goes back to the initial state where it takes N many inputs again. The schematic for this module is shown in Figure 12.

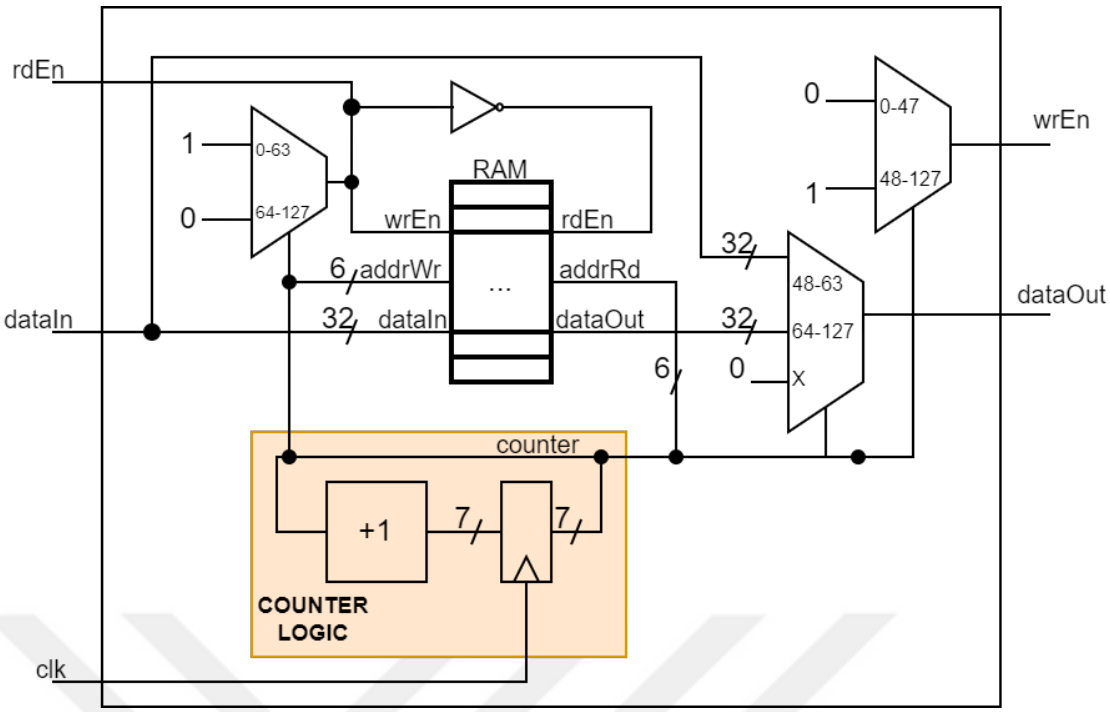

Figure 12: The schematic of the CP Inserter module.

3.4.5 Adding Preamble and Channel Estimation Training Sequence (CETS)

The number of OFDM symbols within one OFDM frame is decided as $N_{\text{OFDM}} = 2$. The Preamble has the same length as one OFDM symbol, and so does the CETS. Preamble is a Pseudo Random Binary Sequence (PRBS) that is pre-generated so that at the receiver side, the start of the OFDM frame can be successfully found. CETS is a pre-generated random sequence that is generated as if it is already CP inserted, with all the previous steps also applied. Since for every OFDM frame, both the Preamble and the CETS are pre-generated and constant values, these values can be statically added to the design, where they could reside in a Read Only Memory (ROM), and can be read from the memory for the beginning of every OFDM frame. Hence, after taking $N_{\text{OFDM}}*(N+N_{\text{CP}}) = 160$ data as inputs to this module, $(1+1+N_{\text{OFDM}})*(N+N_{\text{CP}}) = 320$ data must be output. Before the OFDM symbols are added to the OFDM frame, they also go through multiplication where each data point is multiplied by the square root of the average electrical transmit power, which is equal to 0.1584893167. In Figure 13, the schematic of the Packet

Controller module is given.

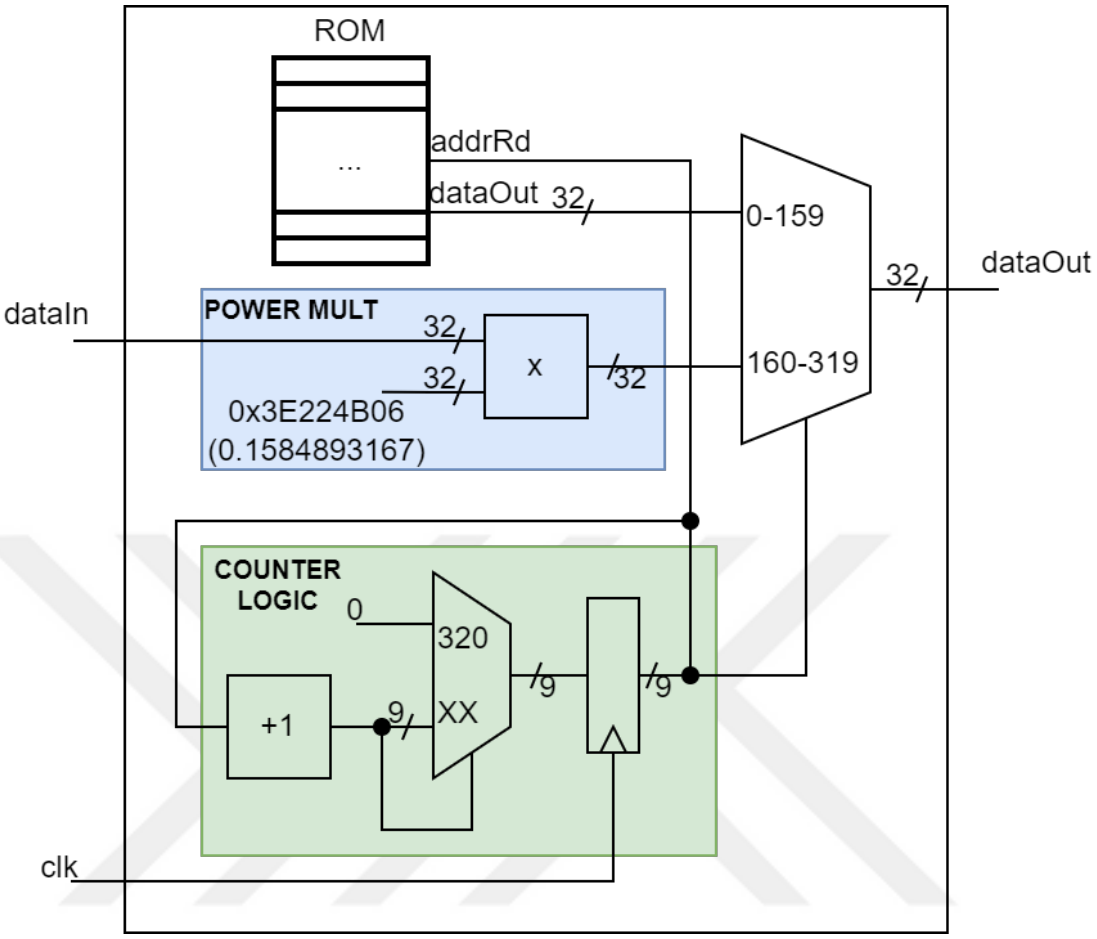

Figure 13: The schematic of the Packet Controller.

3.4.6 Upsampling and Filtering (Spectral/Pulse Shaping)

The OFDM frame is a sequence of discrete data points. To have a signal that is continuous, they must be upsampled and filtered. This process is also called Spectral Shaping or Pulse Shaping. The Upsampling Factor (USF) that was chosen for this transceiver is 20. Hence, the Pulse Shape (PS) must be of size 21. The chosen PS (Root Raised Cosine Pulse Shape) is shown in Figure 5. The module takes inputs one by one, and multiplies each input with the 21 points in the PS. It starts by outputting the 20 results, takes another input, calculates the multiplications for the new input, adds the 21st result from the previous calculation to the first result of this new input, and outputs these first 20 results, and does the same thing again and again. If the number of outputs is $USF * 320 = 6400$

(320 coming from the Packet Controller), then it restarts from the initial state. The schematic of this module is shown in Figure 14. The schematic is simplified to show the parallel-to-serial conversion after the filtering. The subblock named "MULT" within the figure contains 21 Floating Point multipliers and 1 adder, 21 multipliers are used to multiply the filter with the incoming data, and the adder is used to add the last data from the last multiplier with the next data from the first multiplier.

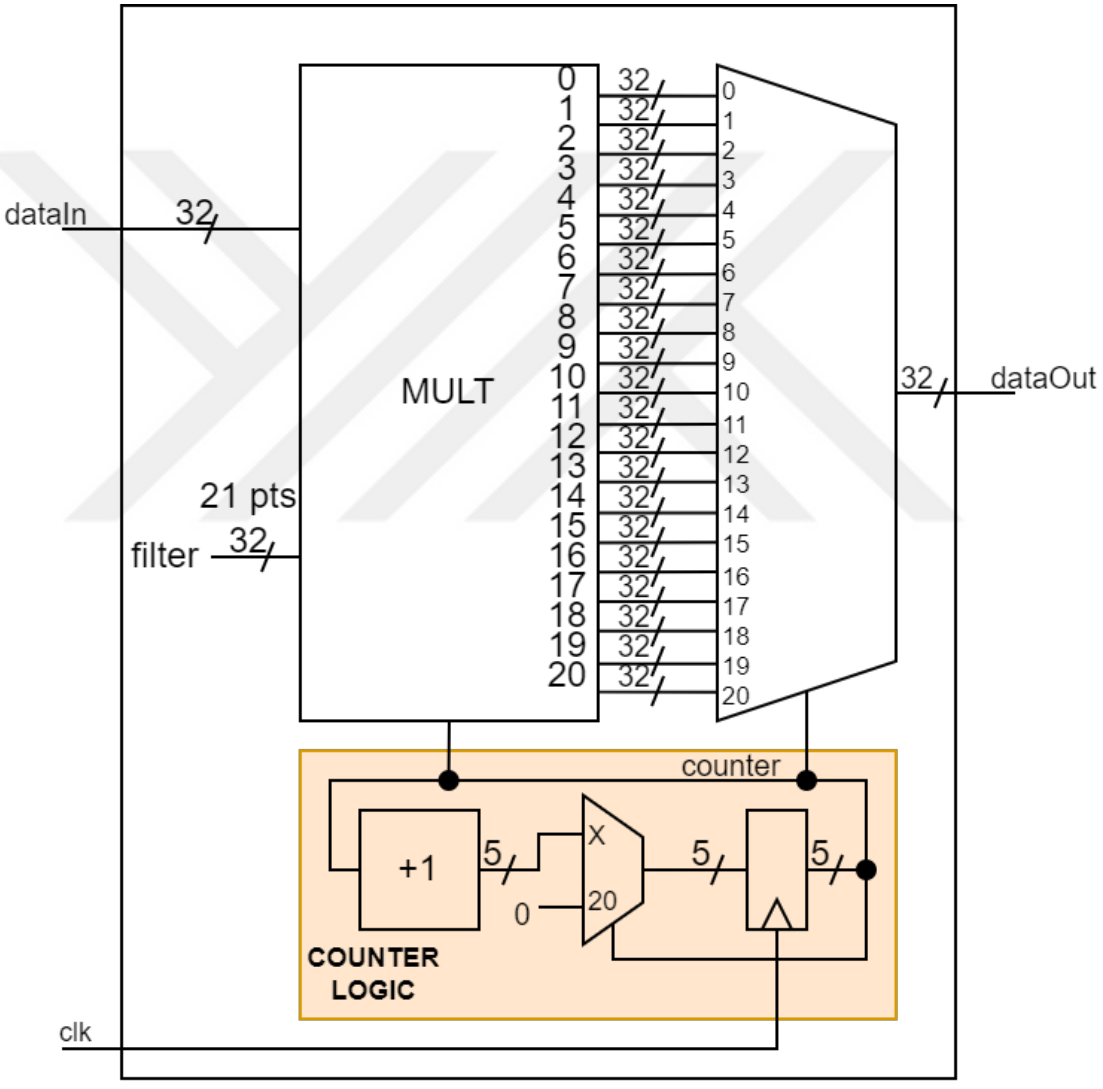

Figure 14: The schematic of the Upsampler.

3.5 DCO-OFDM Receiver Implementation

DCO-OFDM Receiver Implementation that was designed for this thesis will be described in this section. The different modules within the implementation will be explained in detail in the following subsections.

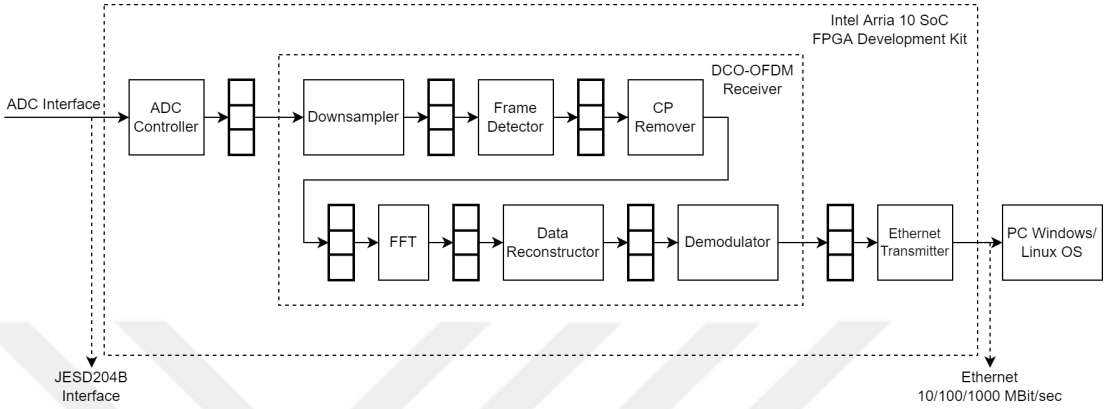

Figure 15: The block diagram of the DCO-OFDM Receiver Implementation.

The block diagram for the DCO-OFDM Receiver Implementation is shown in Figure 15. Similar to the diagram of the DCO-OFDM Transmitter Implementation, ADC Controller and Ethernet Transmitter are also included in the figure. There are FIFO designs between each module, as it was done in the transmitter side, not to lose any data. The introduced FIFO designs let the receiver modules work independently, like it was mentioned in Section 3.4.

3.5.1 Peak Finding and Downsampling

In Peak Finding and Downsampling module, named the Downsampler although it does both procedures, the inputs come from the FIFO after the ADC Controller. The mentioned inputs are spectrally shaped, which are continuous signals. The Pulse Shape (PS) creates peaks and valleys, residing at the center of each pulse, which are essentially the discrete data points from the input side of the Upsampler, except the data points are multiplied by the value at the center of the PS ($PS_{10} \approxeq 0.36304766$), with added noise caused by the channel. In this module, these peaks (and valleys) are found and they are sent through to the next module, with a

FIFO as a bridge. Sending only the peaks essentially acts as the downsampling process.

The Peak Finding process is simply accomplished by summation of the squares of data points. The USF in the Upsampler was decided to be 20, therefore the peaks and valleys only occur every other 20th data point within the signal. Thus, every other 20th data point in a specific window must be added. The square of the 0th point should be summed with the squares of 20th, 40th and so on; the square of the 1st point should be summed with the squares of 21st, 41st and so on, until all 20 sum of squares is found. Then, the greatest of these sums must be found, index of which should be where the peaks reside. Using this index, the peaks can be sent to the following FIFO by starting from the index and sending every other 20th data point.

This process is done within MATLAB very easily, since all points of the signal is ready ahead of time and does not arrive to the downsampling part one by one. The MATLAB algorithm simply reshapes the signal into a matrix with 20 rows, then squares the matrix element-wise. Then finds the index of the maximum sum after summing each row. However, in a real time Transceiver with an incoming signal, this is more complicated.

First, the inputs are squared as they arrive, and each square is added to the sum value which is kept as a register within the module. There are 20 registers for the sum values. Meanwhile, the same incoming data are kept in the memory blocks within the module. There are 20 memory blocks, keeping the data points of each sum value. This memory implementation is equivalent to the reshaping the data into a matrix that is done in MATLAB. By dividing the data points with respect to their sum values, after finding the greatest sum, the data points can easily be sent only by sending every data within that memory block. However, as there are 20 memory blocks of yet undetermined size, each row keeping 32 bit IEEE-754 Floating point data, this module becomes larger as the size of the window for the input is increased. With a large window size, the peaks can be

more accurately found, however the design is much larger. With a small window size, the design becomes much smaller, but the accuracy could be affected, when the peaks or valleys are significantly close to the other data points within that window. For this module, a window of size 6000 was chosen as the input, which means that each memory block within the module has 300 rows. This value was chosen specifically so that it could have a significantly large difference between the largest and the second largest sum values within the window, but yet could be small enough that the memory blocks are smaller to improve the area of the module. When all 6000 data points are taken as input to the module and all the sums are calculated, the sums are compared to find the largest one. After finding the largest value, the index of that value is determined as the starting index of the peaks. Thus, the contents of the memory block of the corresponding index is sent as the output. The schematic for this module can be seen below in Figure 16.

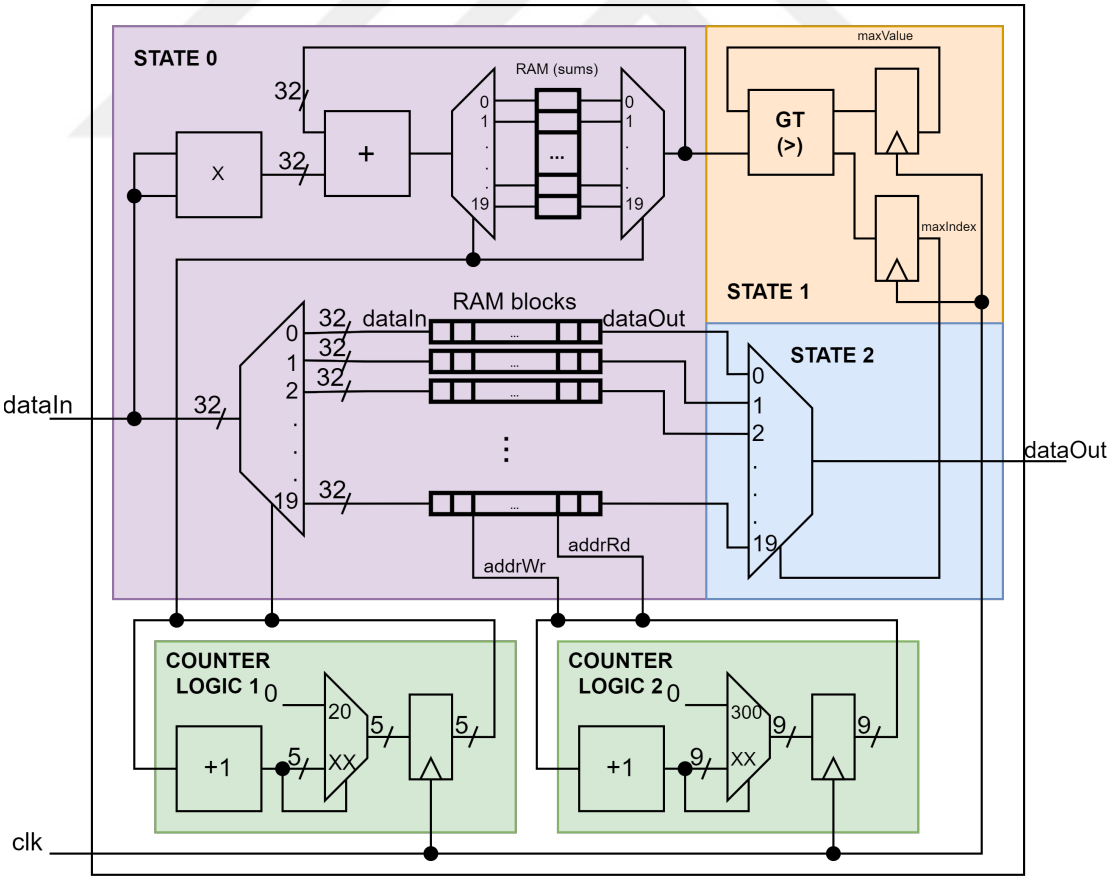

Figure 16: The schematic of the Downsampler.

3.5.2 Frame Detection (Cross-Correlation)

For a continuous incoming stream, to be able to get the transmitted data, first the location of the data must be found within the stream. The data can be found with a sequence that precedes the data itself. Hence, in the transmitter side, a preamble was added in front of the data, which is part of every OFDM frame. The Frame Detection module carries out this process by comparing the incoming data with the pre-generated Preamble. It finds the correlation between them, then compares the correlation values, and if the found maximum value is above a threshold value, then the module decides that it has found the OFDM frame. By discarding the received Preamble, the rest of the OFDM frame is sent to the following FIFO for CP Removal. The Frame Detector module is separated into several submodules to control the different parts of the algorithm.

For preamble detection, Cross-correlation is used. The submodule for this part is called X-Correlator. The Cross-correlation equation is given in eq. (3). The Cross-correlation calculation is done in the time domain, however, using eq. (4), the calculation can be done in the frequency domain. The FFT of both the Preamble and the incoming signal must be calculated. Then, after multiplying the FFT results element-wise, the IFFT of the result is calculated. The results of the IFFT are the correlation values. This process is done using the FFT IP core.

FFT and IFFT are computation intensive functions. They are essentially matrix multiplication where $C = A * B$ is carried out, and A is the FFT or IFFT matrix and B is the input. Depending on how large the matrices are, the latency of the core increases. The size of the matrices is decided on the data being correlated. The incoming data does not particularly have a length due to it being a continuous transmission. However, the Preamble has a decided length, which is the length of 1 OFDM symbol, namely $N + N_{\text{CP}} = 80$. The FFT and IFFT IP cores are only generated with sizes that are powers of 2, the size can be 8, 16, 32, 64 and so on. Since the needed size must be at least 80, The size for both IP cores are chosen as 128. Because of this, the Preamble has following zeros to fill

all 128 data points, and the incoming data will be taken as 128 data. Meanwhile, in Equation (4), the complex conjugate of the Fourier Transform of one of the signals must be calculated. Instead of having to calculate this, using Equation (5), the signal can be reversed and conjugated, and then the Fourier Transform can be calculated. The incoming data comes to the Frame Detector serially, which means receiving all 128 data, and then providing them to the FFT by reversing and conjugating it would increase the latency of the module. Thus, the Preamble is provided to the FFT, reversed and conjugated. This allows the input to be provided to the FFT as soon as it arrives, eliminating the possible added latency.

After the FFT of both signals are calculated, they are multiplied using 4 Floating Point Multiplication IP cores. 4 cores are used here since both data coming from the FFT are complex numbers, and the multiplication must be done using $(a + bi) * (c + di) = (ac - bd) + (bc + ad) * i$. After the multiplication, the result is saved to a RAM until all 128 data is ready to be provided to the IFFT IP core, which then returns the correlation values. The module also waits until all data are output from the IFFT IP core, and then sends the correlation data for processing.

According to the simulations, the latency of the FFT and IFFT are 215 and 213 cycles. The multiplication arithmetic takes 18 cycles as a whole, and since the output from the FFT and the IFFT is saved to a RAM until all the data is ready to be processed, the overall latency of X-Correlator becomes 704 cycles. This latency is a large value which cannot be reduced, which means another architecture must be implemented for the Frame Detector.

Hence, in the Frame Detector, 3 different X-Correlators are used. All 3 correlators can run independently from each other and correlate different incoming data with the Preamble, so running all 3 in parallel adds a large latency gain for the Frame Detector. Every X-Correlator looks up and correlates 128 incoming data, with each data window 40 data points apart. So, for every frame detection procedure, $128 + 40 + 40 = 208$ data are taken as input, the first 128 data (from indices 0 to 127) are sent to the first X-Correlator, from indices 40 to 167 are sent

to the second X-Correlator, and from 80 to 207 are sent to the last one.

After the data is sent to the X-Correlators and the correlation results arrive to the Frame Detector module, the correlation values are compared. The results of each X-correlator are compared within themselves and the maximum value is found, and then compared against the threshold value. If the value is above this threshold, then this maximum value and its index is saved. When all three maximum values are found, they are compared against each other. Here, due to the desired frequency, the Floating Point Comparison IP cores take 2 cycles to find which one is the largest, and thus a small FIFO of size 128 is inserted into the design that saves the correlation values. The module keeps comparing the values in the meantime, so that only a negligible amount of latency is introduced.

After the 3 maximum correlation values are compared and the largest is found, The index of the largest value is then established as the start of the OFDM symbols within that frame. By just starting from that index, the rest of the data is sent to the following FIFO. The number of data sent to the FIFO is counted, and if the number is less than $(1 + NOFDM) * (N + N_{\text{CP}} = (1 + 2) * (64 + 16) = 240$, then the rest of the OFDM frame is sent directly to the FIFO by reading from the preceding FIFO.

In the case that the maximum correlation values are all smaller than the threshold, it is determined that no complete Preamble lies within that window that was taken as input. The keyword here is "no complete Preamble", which could potentially mean that at the end of the window, within the last 79 data points, the beginning of the Preamble might exist. Because of this, to not miss the OFDM frame, the last 79 data points are shifted to the beginning of the Frame Detector's input window. Thus, instead of 208 data points being taken as input from the preceding FIFO, $208 - 79 = 129$ are taken. The schematic for the Frame Detector can be seen in Figure 17.

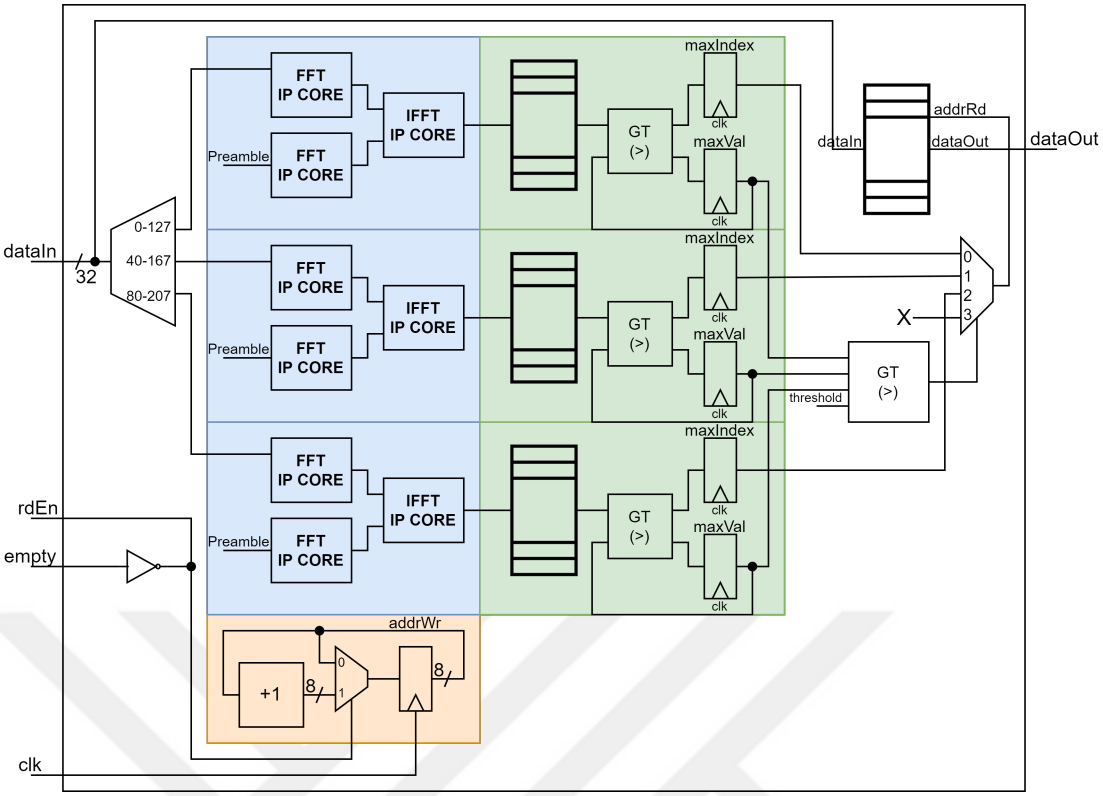

Figure 17: The schematic of the Frame Detector.

3.5.3 Cyclic Prefix (CP) Removal

In the Transmitter part, CP was added to each OFDM symbol including the CETS, so that interference between OFDM symbols is prevented, and circular convolution with the channel is ensured. In the Receiver part, CP must be removed to continue resolving the data hidden in the OFDM frame. This process can be done in a straightforward manner. Since CP is added as a prefix before the actual data, it can simply be discarded. Hence, this module takes N_{CP} many inputs, discard them, then takes N many more inputs, all of which are directly output through this module to the next FIFO. The schematic for this module is shown in Figure 18.

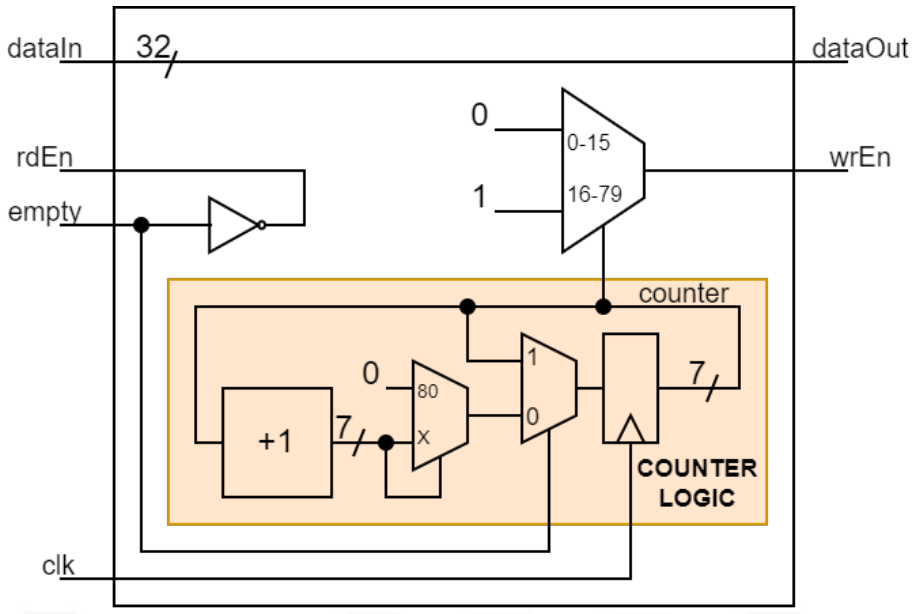

Figure 18: The schematic of the CP Remover.

3.5.4 FFT

Similar to the IFFT top module in the DCO-Transmitter implementation, this module uses the FFT IP core from the DSP/Transforms category that can be found in IP Catalog. This time, however, the IP core is used as the Forward FFT. There are 64 subcarriers per OFDM symbol in this DCO-OFDM Transceiver system, hence the FFT IP core in this module has a length of 64. The inputs are in 32-bit IEEE-754 Floating Point format. Forward direction, Variable Streaming and Single Floating Point representation is selected for this IP core. The input order and output orders both can be selected as natural, which sends the data serially, in order. Hence, the FFT top module is more straightforward.

The FFT top module has only the FFT IP core and the controller logic. The controller logic checks if the preceding FIFO has 64 or more data, then the data is read serially and directly sent to the FFT IP core. After the first OFDM symbol is sent to the FFT IP core, if there are more OFDM symbols ready, they are also sent. In the MATLAB code in the Transmitter part, the result of the IFFT was multiplied by $sqrt(N) = 8$, and in the Receiver part, the result of the IFFT is divided by $sqrt(N) = 8$. In the implementation of the IFFT top module, the

IFFT function in MATLAB and the IFFT IP core had different results, which is why the results of the IFFT IP core was divided by $sqrt(N) = 8$. But in this case, the results of the FFT function in MATLAB and the FFT IP core are the same. Hence the division can be done in the same way. This division is done with subtraction, using the property of the IEEE-754 Floating Point format, like it was done in the IFFT top module. Since the output of the FFT IP core is in natural order, the result of the division can be directly sent to the next FIFO from this module. The schematic for the IFFT top module can be seen in Figure 19.

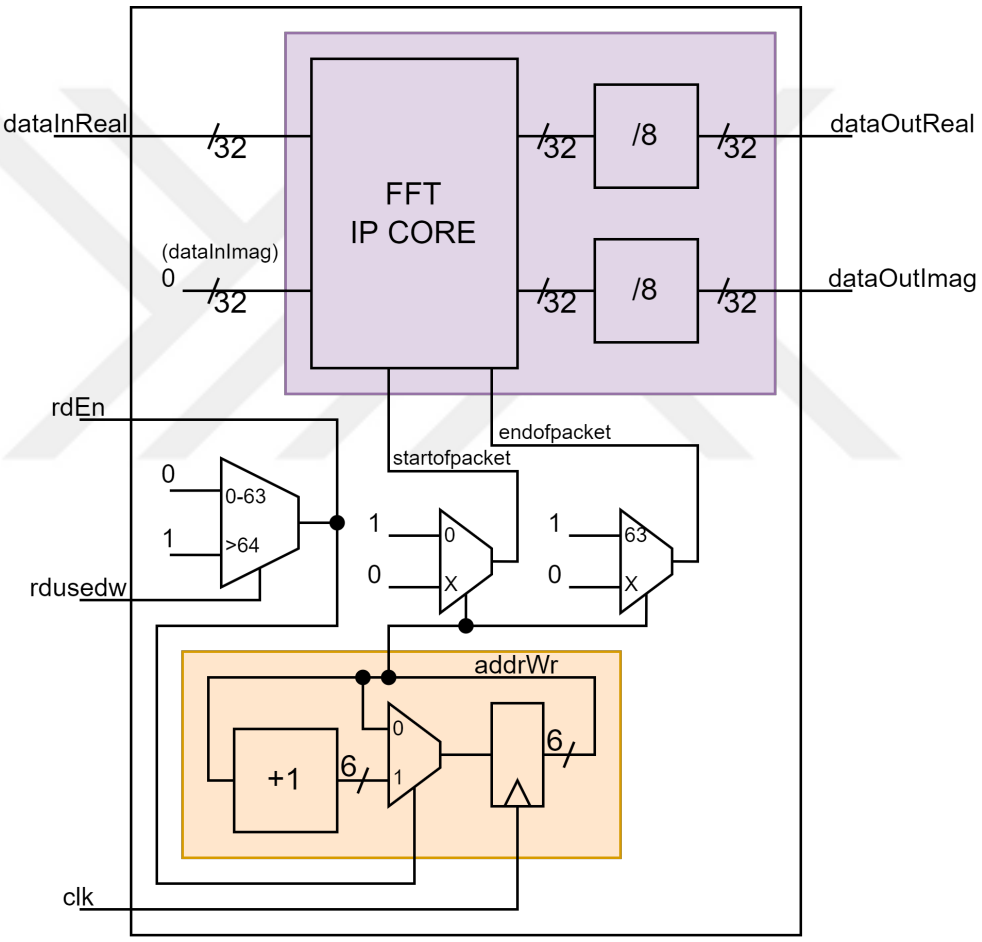

Figure 19: The schematic of the FFT top module.

3.5.5 Channel Estimation and Data Reconstruction

After the FFT of the signal is calculated, the result is equivalent to the data at the end of the Data Assigner, which are the data that are assigned to the subcarriers. However, the two data do not match exactly as there is noise added in the channel

during transmission. The amplitude may also be scaled down, affected by the proximity. Hence, to get the data to match to the one that was sent, the effects of the channel must be found, and that effect must be eliminated from the incoming data.

This process is done using the Channel Estimation Training Sequence (CETS). This is a pre-generated sequence that is known by both the transmitter and the receiver. In the MATLAB code, in the transmitter part, CETS also goes through modulation, data assigning, IFFT and CP insertion; however, since it is a pre-generated constant sequence, in the transmitter side, it is directly added to the OFDM frame post-CP-insertion. In the receiver side, however, the CETS used is only the version that is assigned to the subcarriers, because that sequence is the reference point from which the OFDM symbols are corrected, and the data that arrives at the Data Reconstructor are the equivalent to the pre-IFFT data from the transmitter side. Using the original CETS and the received CETS, the Estimated Effective Channel Coefficient (EECC) is calculated. It is simply calculated by dividing the received CETS by the original CETS, element-wise.

The CETS is pre-generated within MATLAB as mentioned previously. The generation is done using Algorithm 8.

Algorithm 8 CETS Generation

```
1: procedure GENERATECETS
2:     realPart ← 2*randi([0 1],1,N/2-1)-1          ▷ generate modulated CETS
3:     imagPart ← 2*randi([0 1],1,N/2-1)-1      ▷ real and imag both -1s and 1s
4:     symbolCETS ← realPart+1i*imagPart
5:     modulatedCETS ← symbolCETS/sqrt(2)
6:                 ▷ divided by sqrt(2) since the data is modulated in QPSK
7:                                   ▷ N = Number of Subcarriers = 64
```

In Data Reconstructor, the incoming CETS is divided by the original CETS to find the EECC. Then, the incoming OFDM symbol is divided by the EECC to get the reconstructed data that was assigned to the subcarriers. Then, this data is used to get back the modulated data. Since the modulated data was assigned to the subcarriers using the Hermitian Symmetry, every data is duplicated within the

OFDM symbol. Every Hermitian pair is added (the complex conjugate of the data from the second half of the subcarriers are added to the data from the first half of the subcarriers) and halved, so that data is recovered more accurately. Then, this data is multiplied by the square root of the average energy of the symbols (E_{QAM}). In Equation (6), Equation (7) and Equation (8), the equations for the different parts of this algorithm can be seen.

$$EECC(t) = \frac{CETS_{\text{received}}(t)}{CETS_{\text{assigned}}(t)} \tag{6}$$

$$\hat{X}(t) = \frac{Symbol_{\text{received}}(t)}{EECC(t)} \tag{7}$$

$$data_{\text{modulated}}(n) = \frac{\hat{X}(n) + \hat{X}^*(64-n)}{2} * sqrt(E_{\text{QAM}}) \tag{8}$$

$$\frac{a+bi}{c+di} = \frac{(a+bi)*(c-di)}{(c+di)*(c-di)} = \frac{(a*c+b*d)+(b*c-a*d)*i}{(c^2+d^2)} \tag{9}$$

In calculation of both Equation (6) and Equation (7), the numerator and the denominator are complex numbers. In the division of complex numbers, Equation (9) method is used. During the generation of the CETS, a random sequence of -1s or 1s (both for the real part and the imaginary part) are generated as the mapped CETS, then the data was divided by $sqrt(2)$, where 2 is the average energy of the symbols in QPSK. This division results in $c^2 + d^2 = 1$ in Equation (9). Thus, only the multiplication part needs to be done to calculate the EECC. Since the CETS is of length $N = 64$, the module needs to read 64 data points from the preceding FIFO. As it reads the data, it multiplies the incoming CETS and the original CETS. The multiplication is done with 4 Floating-Point Multiplication IP cores and 2 Addition IP cores. The result is then saved to a memory for the second part of data reconstruction.

For Equation (7), an OFDM symbol must be read. The module reads $N = 64$ data from the FIFO, and divides this data by the data in the EECC in the

corresponding index. This division is done using Equation (9). Then, when this part is completed, since the result is the data that are assigned to the subcarriers, the Hermitian Symmetry must be undone. Each Hermitian pair (pair in indices n and (64-n), $n = 1, ..., (N-2)/2, N = 64$) must be added and divided by 2 to get their average. Then, by multiplying this result with the square root of E_{QAM}, the mapped data is found. Thus, the Hermitian pairs are added, and then, since the remaining operations are done with constant values, the added pairs are multiplied by $sqrt(E_{\mathrm{QAM}})/2$. The result is then sent to the following FIFO for demodulation. The schematic of this module can be seen in Figure 20.

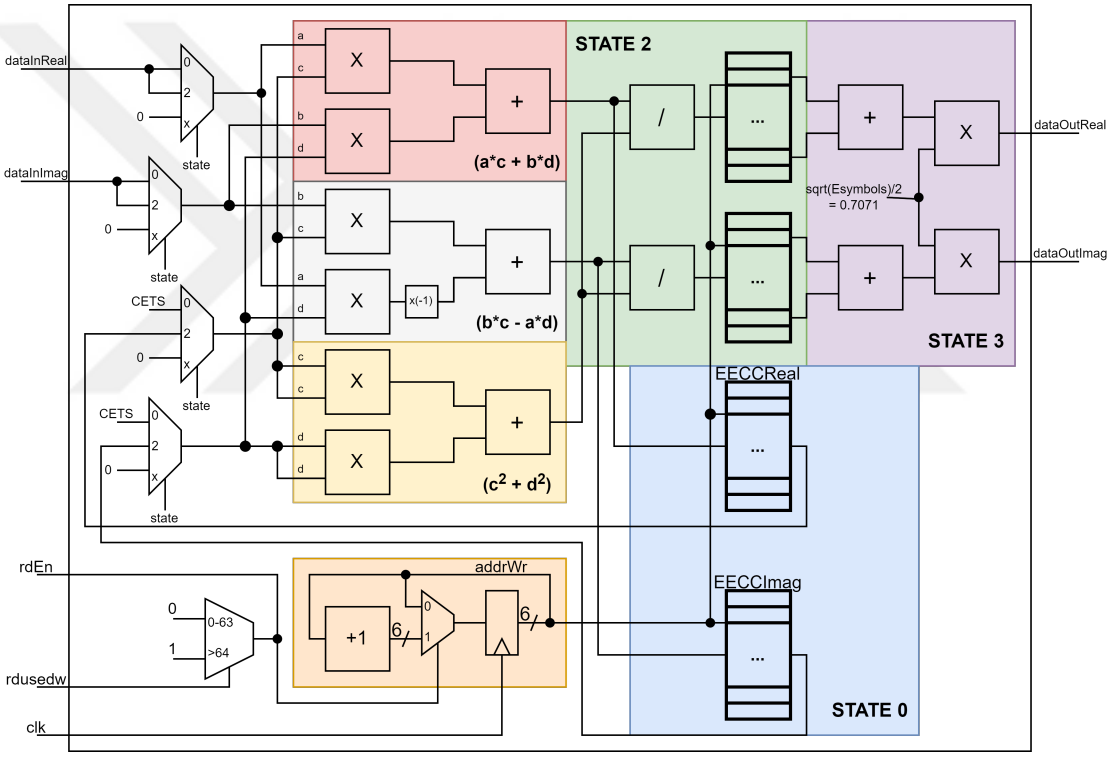

Figure 20: The schematic of the Data Reconstructor.

3.5.6 Demodulation

When there is data in the FIFO preceding the Demodulator, the data is read, and using the constellation map, the closest point to the received data is found. The output is then set to the index of this constellation point.

In this module, the received modulated data will be equivalent to the modulated data from the transmitter, but there will be slight differences due to the

channel noise. Because of this noise, the constellation points will not match perfectly with the received data. Hence, the actual constellation point of the received data is found using Maximum Likelihood (ML) coding. In the module, using the MSB within the IEEE-754 Floating Point format, an integer value from 0 to sqrt(QAM) is found for both the real value and the imaginary value. This is used to eliminate the decimal part of the values. In modulation, all points are separated by 2 units each. Therefore, in the constellation map, 2x2 area is selected to be belonging to the constellation point. If the actual constellation point is $(-3, 1)$ and the received data is $(-2.413, 0.870)$, the integer values will be $(-2, 1)$, which is 1 unit away from the $(-3, 1)$ point, and the rest of the points are much further. Hence, the received data must have been that closest point, which is $(-3, 1)$. After finding the constellation point, the index of that point is then sent as the output from the Demodulator, and then also directly from the DCO-OFDM Receiver. The schematic for the Demodulator is shown in Figure 21.

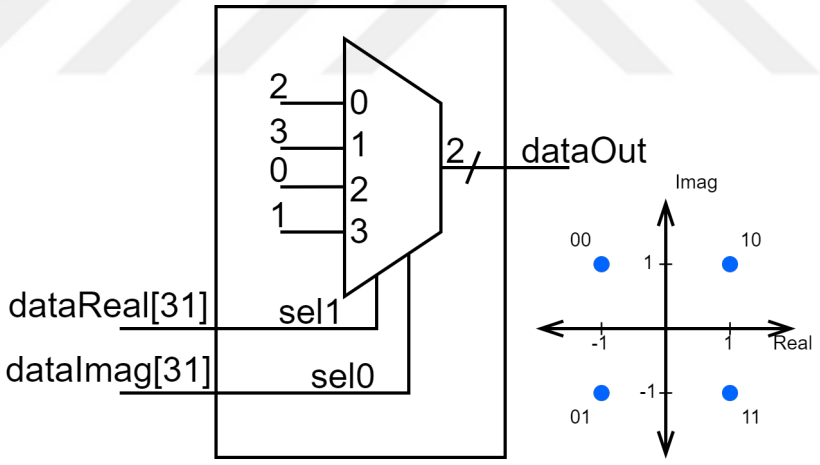

Figure 21: The schematic of the Demodulator.

Chapter IV

EVALUATION

The FPGA Implementation of the DCO-OFDM Technique for VLC Systems is evaluated in this chapter. The chapter is divided into sections to talk about various results. The first section discloses the Resource Statistics of the presented system, the second delivers the latency and throughput of the modules, while also the clock domains of the modules are explained. Then, the data rates of the tested systems and how they were calculated will be clarified in the third section. In the fourth section, the Bit Error Rates (BER) of the tested systems will be explained and interpreted. In the last section, The results of this study are compared with other works found in the literature.

4.1 Resource Statistics

In this thesis, the DCO-OFDM Transceiver implementation for VLC Systems is completed. The design is fitted into the Intel Arria10 SoC Development Kit. Ethernet Interface, DAC/ADC Controllers are included in the design. The modules are all implemented into the FPGA part of the kit while the CPU part is used only to initialize the DAC and the ADC.

The transceiver is implemented with two different Quadrature Amplitude Modulation (QAM) schemes, QPSK (4-QAM) and 256-QAM. The first table, Table 1, illustrates the resource statistics for each designed block for the implementation with the QPSK scheme. The design names are slightly indented to represent which modules are implemented within which modules. Ethernet top module and DAC-ADC top modules are implemented in the System top module, which are all given in bold to signify they are the topmost blocks encompassing the other modules below them. The IP cores such as the FIFOs or the Floating Point Arithmetic IP Cores are not shown explicitly in this table, however, they are included in the

Table 1: Resource Statistics for the DCO-OFDM Transceiver with QPSK scheme.

Design	*Combinational ALUTs*	*Dedicated LogicRegisters*	*BlockMem. Bits*	*DSP*
System top (Total)	250276 (3)	28898 (0)	21943264	494
Ethernet top	2969 (0)	3671 (0)	184288	0
Eth IP	2065 (0)	3160 (0)	167904	0
Packet Receiver	86 (70)	52 (36)	0	0
Packet Finder	490 (490)	255 (255)	0	0
Paket Storer	96 (41)	139 (25)	16384	0
DAC-ADC top	243812 (0)	273225 (0)	16384960	494
DAC Controller	15736 (363)	21949 (172)	532160	46
OFDM Transmitter	14798 (0)	20022 (0)	253632	46
Modulator	-	-	-	-
Data Assigner	229 (229)	42 (42)	1984	0
IFFT	9709 (147)	11754 (33)	14080	24
CP Inserter	113 (113)	81 (81)	8192	0
Packet Controller	334 (196)	313 (15)	0	1
Upsampler	4041 (395)	7227 (629)	0	21
ADC Controller	189222 (196)	192538 (480)	1383152	381
OFDM Receiver	188532 (0)	190639 (0)	838384	381
Downsampler	3673 (2960)	1482 (898)	192000	2
Frame Detector	163394 (19335)	166991 (7090)	393600	336
XCorrelator	47606 (256)	53129 (88)	127104	112
CP Remover	123 (123)	64 (64)	0	0
FFT	11686 (73)	12893 (7)	15216	24
Data Reconstructor	9214 (7554)	8561 (4237)	71680	19
Demodulator	1 (1)	3 (3)	0	0

results of the modules they are in. The DAC-ADC top module is comprised of not only the DAC Controller and the ADC Controller, but also a bundle of other IP cores that communicate with the FMCDAQ2-EBZ board, which is the DAC & ADC board, through the FMC Connector. These IP cores are also not given in this table, but are included in the results for the DAC-ADC top block.

In Table 1 and Table 2, the values in parentheses represent the number of that resource that this module uses alone. For example in Table 1, the Packet Receiver module uses 86 ALUTs, however only 70 of those ALUTs are specific to this module, and not used by the other modules. In the table, the Modulator row

is left empty. Since the results are provided by the Analysis & Synthesis stage, it can be inferred that due to the QPSK modulator being a simple mapping scheme, it is simply synthesized without using any more resources than already being used.

The resource statistics for the implementation with the 256-QAM scheme is depicted in Table 2. It can be seen that some values have been slightly modified in some modules, affected by the synthesis of the 256-QAM scheme, considering both tables. This is likely due to some rerouting made after the 256-QAM modulator is synthesized and the area is larger than that of the QPSK modulator.

Table 2: Resource Statistics for the DCO-OFDM Transceiver with 256-QAM scheme.

Design	*Combinational ALUTs*	*Dedicated LogicRegisters*	*BlockMem. Bits*	*DSP*
System top (Total)	254009 (3)	292400 (0)	22130912	496
Ethernet top	2963 (0)	3671 (0)	184288	0
Eth IP	2061 (0)	3160 (0)	167904	0
Packet Receiver	86 (70)	52 (36)	0	0
Packet Finder	491 (491)	255 (255)	0	0
Paket Storer	96 (41)	139 (25)	16384	0
DAC-ADC top	247551 (0)	277527 (0)	16572608	496
DAC Controller	15683 (268)	21933 (160)	532160	46
OFDM Transmitter	14841 (0)	20018 (0)	253632	46
Modulator	46 (0)	0 (0)	0	0
Data Assigner	228 (228)	42 (42)	1984	0
IFFT	9707 (147)	11750 (33)	14080	24
CP Inserter	113 (113)	81 (81)	8192	0
Packet Controller	334 (196)	313 (15)	0	1
Upsampler	4041 (395)	7227 (629)	0	21
ADC Controller	193057 (169)	196856 (466)	1570800	383
OFDM Receiver	192448 (0)	195085 (0)	1030128	383
Downsampler	3641 (2932)	1482 (898)	275200	2
Frame Detector	164223 (19356)	170963 (7090)	479616	336
XCorrelator	47285 (265)	54444 (97)	151680	112
CP Remover	124 (124)	64 (64)	0	0
FFT	11682 (72)	12893 (7)	15216	24
Data Reconstructor	12284 (10372)	9013 (4237)	71680	21
Demodulator	45 (1)	25 (25)	0	0

4.2 Timing

The latency and throughput of the modules are disclosed in this section. In Table 3, DCO-OFDM Transmitter Implementation and its submodules are provided with their latencies and throughputs. Modulator and Data Assigner both have a latency of 0 since they can output data as soon as there is incoming data. The IFFT part has 197 cycles latency due to the matrix multiplication. CP Inserter has a latency of 48 cycles due to the number of subcarriers (N) being 64 and the length of CP, namely N_{CP}, being 16. The Packet Controller module, since it has the pre-generated preamble and CETS, it can output those sequences as soon as the reset signal (rst) is turned off. Hence the latency could be considered as 0 cycles, however, from the first data input, the first output would have a latency of 6 cycles. The Upsampler also has 6 cycles latency due to the multiplication IP core. Since the output of the transmitter is directly connected to the Upsampler, the latency of the transmitter module becomes 6 cycles. The throughputs are given in either "OFDM frame/cycles" or "OFDM symbol/cycles" format to represent the percentage of the frame or the symbol being output per cycle.

Table 3: Table of Latency and Throughput for the DCO-OFDM Transmitter.

Design	*Latency (Cycles)*	*Throughput*
OFDM Transmitter	**6**	**1/6400 OFDM frame/cycles**
Modulator	0	1/31 OFDM symbol/cycles
Data Assigner	0	1/64 OFDM symbol/cycles
IFFT	197	1/64 OFDM symbol/cycles
CP Inserter	48	1/80 OFDM symbol/cycles
Packet Controller	6 (0 from rst)	1/330 OFDM packet/cycles
Upsampler	6	1/6400 OFDM frame/cycles

The latency of the submodules of the DCO-OFDM Receiver Implementation are reported in Table 4. The Downsampler has the largest latency due to the time it takes to receive a large window of samples. Since it takes 6000 data from the ADC to find the peaks and valleys and downsample, $6000/6016 = 0.997 = 99.7\%$

of the latency arises from the receiving input part. The calculations continue while the inputs are received, and in the last 16 cycles, the peaks are found. The 6000 data points are downsampled by 20, therefore 300 data are output from the Downsampler. This throughput of the Downsampler is 300/6336 downsampled points/cycles. The throughput is again limited by the number of inputs, and also the number of outputs. The Frame Detector, with 3 X-Correlators working independently, has a latency of 1188 cycles. This is largely due to the FFT and IFFT operations. The CP remover has 17 cycles, the FFT has 236 cycles, the Data Reconstructor has 209 cycles latency, whereas the Demodulator has 1 cycle latency. The latency of the DCO-OFDM Receiver is not reported, mainly due to the inputs being unpredictable. The inputs may or may not have an OFDM frame, which prohibits the calculation of latency correctly. The throughputs for the receiver submodules are also not reported due to this reason.

Table 4: Table of Latency for the DCO-OFDM Receiver.

Design	*Latency (Cycles)*
OFDM Receiver	**-**
Downsampler	6016
Frame Detector	1188
CP Remover	17
FFT	236
Data Reconstructor	209
Demodulator	1

The DAC and ADC within the FMCDAQ2-EBZ board is used in this thesis. The DAC and ADC both have up to 1GSPS sampling rate, which can be reduced with the correct configurations. The communication between the FPGA and the FMCDAQ2-EBZ board is done through the JESD204B interface, which works with 1/4th of the sampling rate of the DAC/ADC. The interface receives 4 data samples at the same time, and then sends them to the DAC and the ADC linearly. For this design, the same data is sent 4 times using the clock rate of the JESD204B,

therefore when the DAC/ADC are configured to have a sampling rate of 1GSPS, the DCO-OFDM Frame is sent 1/4th of that rate, 250MSPS. The clock frequency of the JESB204B interface is 250 MHz in this case.

The implementation is carried out with this clock frequency in mind. Because if the DAC is able to send data at 250 MHz, the DCO-OFDM Transmitter must be able to work at a frequency close to this rate, less or equal. If the DCO-OFDM Transmitter works faster, than the FIFO between the transmitter and the DAC would overflow, and the communication would fail. On the DCO-OFDM Receiver side, however, the opposite must be done. Since the ADC is working at 250 MHz, the DCO-OFDM Receiver must be able to work at a higher frequency. Using the Downsampler's latency and throughput, the minimum frequency for the DCO-OFDM Receiver can be calculated. However, considering the Upsampler and the Downsampler upsampling by 20 and downsampling by 20, these two modules can be separated from the other submodules within the DCO-OFDM Transmitter and the DCO-OFDM Receiver. Also, due to the high latency rate of the Frame Detector, this module can be considered with the Downsampler. Separating these modules from the rest enables the other modules to work at a lower frequency while these work at a higher one.

While the rest of the modules are configured to work with a clock frequency of 100 MHz, the Upsampler is configured with the same clock frequency as the JESD204B interface, which is 250 MHz. When the DAC/ADC clock frequencies are reconfigured, the Upsampler also works with the same clock frequency as the JESD204B interface, it is simply connected to the same clock signal.

The Downsampler and the Frame Detector however, cannot work with the same frequency as the ADC, due to the amount of data downsampled in the working period of the Downsampler. The Downsampler has a throughput of 300/6336 downsampled points/cycles, and it receives 6000 data within that frame to output 300 points. So, when the Downsampler can downsample 6000 data in 6336 cycles and starts receiving another 6000 data, the clock frequency must be increased

accordingly. The calculation for this is given in Equation (10). According to the calculations, the Downsampler and the Frame Detector must work with a clock frequency of at least 264 MHz.

$$XMHz = \frac{6336cycles}{6000cycles} * 250MHz = 264MHz \tag{10}$$

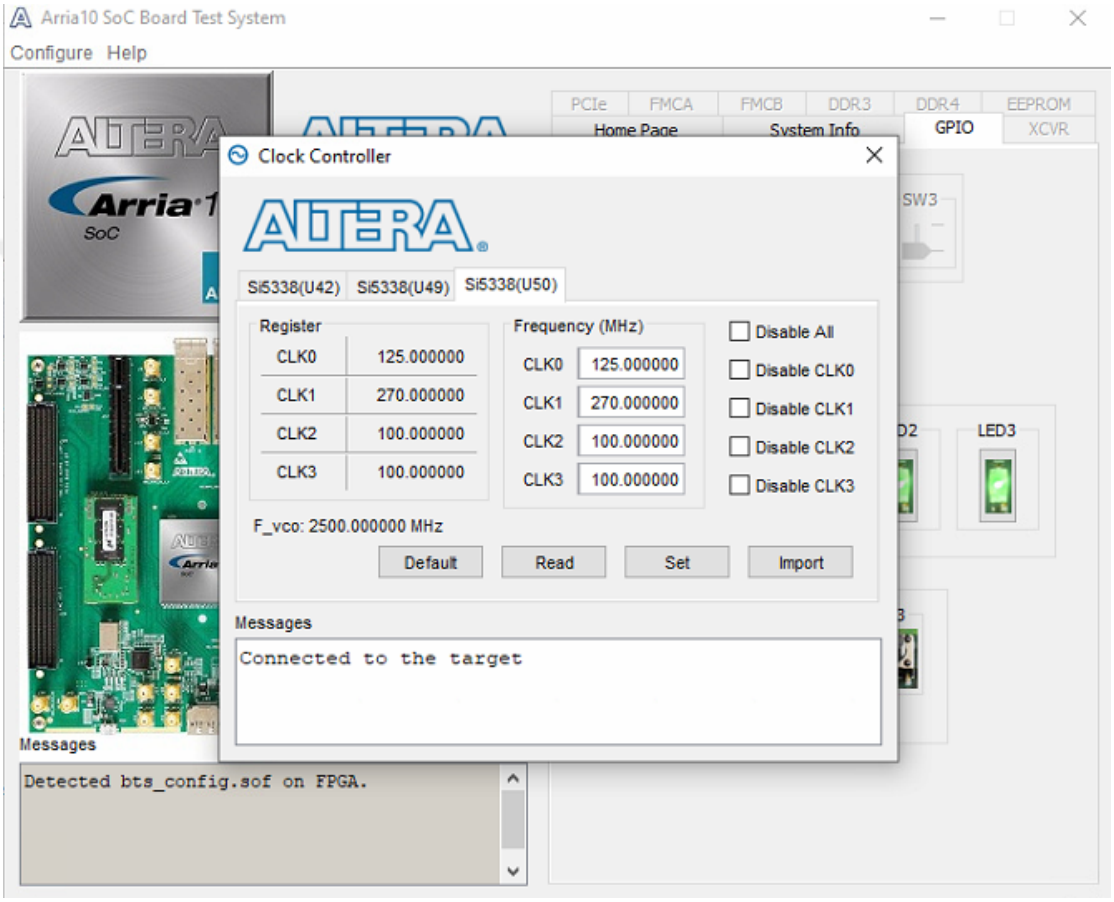

Figure 22: The configuration window showing the default frequencies of some of the reconfigurable clocks within the Intel Arria10 SoC Development Kit, from the Clock Controller program provided by Intel.

The Intel Arria10 SoC Development Kit has several clock domains. There are also reconfigurable clocks. In Figure 22, the default values of some of the reconfigurable clocks can be seen. Since CLK1 has a frequency of 270 MHz, this clock signal is used for the Downsampler and the Frame Detector. The summary of the modules and the clock frequency they are working with is provided in Table 5.

The Upsampler frequency is given as 250/125 MHz because the DAC and the ADC are used with both frequencies which will be explained in the rest of the chapter.

Table 5: Table of clock domains for the submodules.

Design	*ClockFrequency* (MHz)
OFDM Transmitter	
Modulator	100
Data Assigner	100
IFFT	100
CP Inserter	100
Packet Controller	100
Upsampler	250/125
OFDM Receiver	
Downsampler	270
Frame Detector	270
CP Remover	100
FFT	100
Data Reconstructor	100
Demodulator	100

4.3 How the Data Rate is Calculated

The DCO-OFDM Transmitter and the Receiver are implemented with the DAC and the ADC working frequencies kept in mind. DCO-OFDM Transmitter is able to output data as the data are being sent to the DAC, so the transmitter implementation does not slow down the transmission speed. The DCO-OFDM Receiver works faster than the ADC, which means that it also does not slow down the transmission. It can be inferred that the transmission relies on the sampling rate of the DAC and the ADC.

The calculation of the data rate must be done using the clock frequency of the DAC and the ADC. The data rate relies on how many bits are sent per OFDM frame, and how many OFDM frames can be sent per second. The OFDM frames in this thesis are comprised of the Preamble and CETS which are both 1 OFDM symbol long, and 2 OFDM symbols. One OFDM symbols has a length of

$N + N_{CP} = 64 + 16 = 80$ without the upsampling. the upsampling factor is 20. The number of bits sent in each OFDM frame depends on the QAM scheme used. Since there are 64 subcarriers and Hermitian Symmetry is applied, 31 different inputs can be sent per OFDM symbol. In QPSK, the inputs must be 2 bits, resulting in 62 bits for OFDM symbol. In 256-QAM, the inputs are 8 bits, which means that one OFDM symbol carries 248 bits, or 31 bytes.

For 256-QAM, splitting the ethernet packets into 31 bytes of information is simple, however, for QPSK, splitting the packets into 62 bits result in ethernet bytes being split between OFDM frames. Since each OFDM frame carries 2 OFDM symbols, 124 bits can be carried per OFDM frame. In order to not split the bytes, 1 of the 31 inputs is sent as 0, and those inputs are removed after the DCO-OFDM Receiver resolves the OFDM frames, effectively cancelling the 2 subcarriers it uses. Hence, 120 bits, or 15 bytes are sent per OFDM frame.

The equation for the data rate calculation can be seen in Equation (11) for QPSK, and in Equation (12) for 256-QAM. NOFDM represents the number of OFDM symbols per OFDM frame, N represents the number of subcarriers, N_{CP} represents the length of the Cyclic Prefix, M_{QAM} represents the modulation scheme (QPSK=4, 256-QAM=256) and the USF represents the upsampling factor. The frequency of the DAC and the ADC are represented by $freq_{DAC}$.

$$X = \frac{NOFDM * ((N-2)/2) - 1) * sqrt(M_{QAM})bits}{USF * (1 + 1 + NOFDM) * (N + N_{CP}) * \frac{1}{freq_{DAC}}} \tag{11}$$

$$X = \frac{NOFDM * ((N-2)/2)) * sqrt(M_{QAM})bits}{USF * (1 + 1 + NOFDM) * (N + N_{CP}) * \frac{1}{freq_{DAC}}} \tag{12}$$

After calculating the data rate for the QPSK scheme with the DAC used with a frequency of 250 MHz, the result is found to be 4.6875 Mb/s. The data rate for 256-QAM with the DAC used with 250 MHz frequency, however, is 19.375 Mb/s.

4.4 Bit Error Rates (BER)

The DCO-OFDM Transceiver was first implemented with the 256-QAM modulation scheme. The implementation was tested with the DAC and the ADC configured first with 250 MHz frequency and then with 125 MHz. The tests were done using one FPGA board and one DAC/ADC board and a 1-meter-long coaxial cable with SMA connectors. The two ends of the cable was connected to the DAC output and the ADC input, creating a loopback to the same DAC/ADC board and to the same FPGA. The tests concluded that there is too much error in the system with such complexity. This implementation was tested by sending ethernet packets of 150 bytes through the ethernet cable from the PC. Since the Ethernet Receiver adds a header of 24 bytes and 12 bytes, the total transmitted data becomes 186 bytes. The OFDM frames with 256-QAM modulation carry 62 bytes of data, therefore 1 ethernet packet was sent through to the receiver using 3 OFDM frames. During the testing, 1000 packets were sent through. The first tests revealed that the header is sometimes not caught by the Ethernet Finder, resulting in missing the entire packet. This is likely due to the BER being high, since the Ethernet Finder has a tolerance of 20%. If the header has a higher BER, it is not found. The header is 24 bytes, however the same 12 bytes are repeated. To make sure that the packet is caught even if the first half of the header is not caught, the packets were modified to include the 12 bytes of the header so that even if the first half of the header is missed, the implementation will find the second half as the first, and the start of the ethernet packet as the second half of the header. Thus, the ethernet packet is found and although the first part of the packet is missing in the incoming ethernet packet, the data still arrives. After the data is sent and tested that way, with the DAC/ADC set to 125 MHz, it is found that 11.2% of the data within the 1000 packets sent are lost. Out of the received bytes, the BER is 15.5%. For the test with the DAC/ADC set to 250 MHz, the 8.4% of the data within the packets are lost, and the BER is 11.9%. With such high BER, transmission is virtually impossible.

The solution to reducing the error rate was found in reducing the modulation scheme. Thus, QPSK was implemented. Using QPSK, and sending 15 bytes per OFDM frame, the tests concluded that the system has very low error rate. With these tests, 174 bytes were sent on each ethernet packet, which a divided into 14 OFDM frames after adding the header and the footer. Applying the same method of testing, it is seen that with the DAC/ADC set to 125 MHz and 250 MHz frequency, 0.4% and 0.7% of the data are lost, respectively. Out of the received bytes, it can be seen that the BERs of these tests are 1.7% and 0.5%, respectively.

After changing the modulation scheme, it can be seen that the error rate decreases immensely. However, it can be reduced even further. After analyzing the BER per subcarrier, it is seen that most subcarriers have 0 BER. The subcarriers that have non-zero BER can be cancelled for to reduce the BER, with the cost of a lower data rate. It is seen that in the first OFDM symbol, subcarrier pairs with indices 1, 2, 3, 4, 5, 9 and 11 carry the bits that are received incorrectly. In the second OFDM symbol, the subcarrier pairs with indices 1, 2, 3, 4, 6 and 16 carry the bits that are received incorrectly. If these inputs are sent as zeros from the Ethernet Receiver implementation and are discarded in the Ethernet Transmitter part, then the BER must be improved. However, since there are 13 inputs that must be discarded, the Ethernet Receiver must split the bytes into OFDM frames. 1 more subcarrier pair must be cancelled so that the OFDM frame contains 12 bytes. This is done by cancelling the last input, the last subcarrier pair from the second OFDM symbol. When this configuration is applied and tested, with the DAC/ADC set to 125 MHz, 0.4% of the data within the packets are lost, and the BER is found to be 0%. With the DAC/ADC configured to 250 MHz, 0.6% of the data within the packets are lost, with 0% BER. These tests are done by sending 192 bytes per ethernet packet, which are split into 19 OFDM frames.

The minimum and maximum BER from the test results are reported in Table 6. These results are found by calculating the BER for each ethernet packet received. The BERs and the data rates for each configuration can be seen in Table 7. The

data rates are calculated using Equation (11) and Equation (12).

Table 6: Table of minimum and maximum BER results for each configuration.

$Modulation\ Scheme$	$Sentbytes\ (perframe)$	$DAC/ADC\ freq(MHz)$	$minBER(\%)\ inpackets$	$maxBER(\%)\ inpackets$
256-QAM	62	125	11.4	31.9
256-QAM	62	250	7.9	29.8
QPSK	15	125	0	4.9
QPSK	15	250	0	3.5
QPSK	12	125	0	0
QPSK	12	250	0	0

Table 7: Table of BER results and data rates for each configuration.

$Modulation\ Scheme$	$Sentbytes\ (perframe)$	$DAC/ADC\ freq(MHz)$	$BER(\%)$	$Datarate\ (Mb/s$
256-QAM	62	125	15.5	9.6875
256-QAM	62	250	11.9	19.375
QPSK	15	125	1.7	2.34375
QPSK	15	250	0.5	4.6875
QPSK	12	125	0	1.875
QPSK	12	250	0	3.75

The implementation with QPSK that sends 12 bytes per OFDM frame is then also tested with and LED and a Photodetector. The test was done with the DAC and the ADC set to 125 MHz, and the LED and the Photodetector have a distance of approximately 23 cm. The results indicate that the ethernet packets have a minimum BER of 7.6% and a maximum BER of 8.9%. The total BER of the test is 8.27%. This is most likely due to the channel noise.

4.5 Comparing with Other Works

In [1], Wang et. al. propose a VLC system employing DCO-OFDM with a data rate of 24 Mb/s. However, the paper does not report on the upsampling and filtering of the data, which suggests that there is no upsampling or filtering. They employed 16-QAM, using 1024 subcarriers with 336 of the subcarriers carrying

data. It is inferrable that due Hermitian Symmetry, $336/2 = 118$ inputs are taken for one OFDM symbol. The CP ratio of the study is also 1/8, which means that the CP has a length of 64. The study also does not explicitly mention how many OFDM symbols are within one OFDM frame, how long the Preamble and the CETS are, and the DAC/ADC sampling rate. Considering they used 16-QAM, their study resulted in a data rate of 24 Mb/s. If they had used QPSK, the data rate would be halved, meaning 12 Mb/s. Comparing to this study, considering the same methodology of implementation Wang et. al. used, if this study used 1024 subcarriers with 118 inputs, the CP length was 64 and the upsampling and filtering was not done, the calculations result in a data rate of 38.60 Mb/s. This theoretical data rate surpasses the 12 Mb/s that would be received from implementing their study with QPSK. Therefore, this study is theoretically $3.22x$ faster than the study done by Wang et. al.

The study done by Fuada et. al. proposes a system that also employs DCO-OFDM, in [2, 3]. They implemented the design in 3 different modulations: BPSK, QPSK, 16-QAM. In their study, however, the upsampling and filtering are not implemented in the FPGA. The filtering is most likely done in the Analog-Front-End (AFE). Their CP ratio is reported to be 1/8, They have 64 subcarriers, 56 of which carry the data in QPSK, meaning that $56/2 = 28$ inputs are taken for one OFDM symbol. There are no reported CETS. Their OFDM frame is made up of 1 preamble sequence that has a length of one OFDM symbol, and 1 OFDM symbol. They tested the system in simulations, using the FPGA + the DAC + the ADC (without Optical channel), using the FPGA + the DAC + the ADC + the AFE that they have designed. The simulations are done using a 100 MHz clock, the DAC and the ADC they have are reported to have a sampling rate of 1 MSPS, which suggests a clock rate of 1 MHz. Minimal information is given on the AFE. Their simulation results show that with QPSK modulation, they can achieve 13.31 Mb/s, which is bottlenecked by the transmitter. Their tests with the FPGA, the DAC and the ADC demonstrates that with a DAC/ADC frequency of 1 MHz, the

data rate achieved is 396.76 Kb/s. If, in this study, the implementation was done in the same methodology; using 64 subcarriers with 28 inputs, CP with a length of 8, without upsampling and filtering and with a clock frequency of 1 MHz for the DAC and the ADC, then the calculations show that this study would theoretically have a data rate of 388.89 Kb/s. This theoretical result is very close to the result that was received by Fuada et. al., with a ratio of $0.98x$. Therefore, although this design is slower, it is very close to their results. However, their design was implemented using a 100 MHz clock, with the transmitter bottlenecking the data rate. If this design was using the DAC and the ADC by configuring them to 100 MHz, and the system was implemented using their methodology, our design would theoretically result in a data rate of 38.89 Mb/s. In their simulations, since they are bottlenecked by the transmitter, they have a data rate of 13.31 Mb/s, as mentioned before. In this case, this design is theoretically $2.92x$ faster.

Chapter V

CONCLUSION

The DCO-OFDM modulation technique for VLC Systems was implemented on an FPGA with Verilog HDL in this study. The technique is implemented as a single-input single-output (SISO) transceiver. Both the transmitter and the receiver implementations are synthesized in a single FPGA using the Intel Arria10 SoC Development Kit. The design is complete with the ethernet interface and the DAC and ADC interface.

Multiple implementations were done using QPSK and 256-QAM modulation techniques, and using the DAC and the ADC as 125 MHz and 250 MHz. The reported BERs with the data rates are presented in Table 7. The tests concluded that with an OFDM frame carrying 2 OFDM symbols that have 12 bytes of information and using the DAC/ADC with the 250 MHz configuration, the BER is found to be 0. This test was done without the Optical Channel, but using a 1-meter-long coaxial cable with SMA connectors between the DAC and the ADC. The data rate for this system is 3.75 Mb/s.

Bibliography

[1] Y.-J. Wang, S.-L. You, Z.-H. Zhu, W.-T. Lin, C.-Y. Ho, C.-L. Hsu, C.-H. Lee, and H.-P. Ma, "A 24 Mbit/s Red LED-based Visible Light Communication System Employing DCO-OFDM Modulation," in *International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pp. 1–4, IEEE, 2020.

[2] S. Fuada, T. Adiono, A. P. Putra, and E. Setiawan, "Design of Reconfigurable System-on-Chip Architecture for Optical Wireless Communication.," *J. Commun.*, vol. 14, no. 10, pp. 965–970, 2019.

[3] S. Fuada, E. Setiawan, T. Adiono, and W. O. Popoola, "Design and Verification of SoC for OFDM-based Visible Light Communication Transceiver Systems and Integration with Off-the-Shelf Analog Front-End," *Optik*, vol. 258, p. 168867, 2022.

[4] M. S. Islim and H. Haas, "Modulation Techniques for Li-Fi," *ZTE Communications*, vol. 14, pp. 29–40, 04 2016.

[5] J. Armstrong, "OFDM for Optical Communications," *Journal of Lightwave Technology*, vol. 27, no. 3, pp. 189–204, 2009.

[6] D. Karunatilaka, F. Zafar, V. Kalavally, and R. Parthiban, "LED Based Indoor Visible Light Communications: State of the Art," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1649–1678, 2015.

[7] H. Elgala, R. Mesleh, H. Haas, and B. Pricope, "OFDM Visible Light Wireless Communication Based on White LEDs," in *IEEE 65th Vehicular Technology Conference - VTC2007-Spring*, pp. 2185–2189, 2007.

[8] S. N. Ismail and M. H. Salih, "A Review of Visible Light Communication (VLC) Technology," *AIP Conference Proceedings*, vol. 2213, no. 1, p. 020289, 2020.

[9] J. He, R. A. Norwood, M. Brandt-Pearce, I. B. Djordjevic, M. Cvijetic, S. Subramaniam, R. Himmelhuber, C. Reynolds, P. Blanche, B. Lynn, and N. Peyghambarian, "A Survey on Recent Advances in Optical Communications," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 216–240, 2014. 40th-year commemorative issue.

[10] A. G. Bell, W. Adams, W. Preece, *et al.*, "Discussion on "The Photophone and the Conversion of Radiant Energy into Sound"," *Journal of the Society of Telegraph Engineers*, vol. 9, no. 34, pp. 375–383, 1880.

[11] H. Chen, C. Wu, H. Li, X. Chen, Z. Gao, S. Cui, and Q. Wang, "Advances and Prospects in Visible Light Communications.," *Journal of Semiconductors*, vol. 37, no. 1, 2016.

[12] G. Pang, T. Kwan, H. Liu, and C.-H. Chan, "Optical Wireless Based on High Brightness Visible LEDs," in *Conference Record of the 1999 IEEE Industry Applications Conference. Thirty-Forth IAS Annual Meeting (Cat. No.99CH36370)*, vol. 3, pp. 1693–1699 vol.3, 1999.

[13] G. Pang, T. Kwan, C.-H. Chan, and H. Liu, "LED Traffic Light as a Communications Device," in *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No.99TH8383)*, pp. 788–793, 1999.

[14] K. Kulhavy, "Home, RONJA." `ronja.twibright.com` (accessed July 25, 2022).

[15] Y. Tanaka, S. Haruyama, and M. Nakagawa, "Wireless Optical Transmissions with White Colored LED for Wireless Home Links," in *11th IEEE International Symposium on Personal Indoor and Mobile Radio Communications. PIMRC 2000. Proceedings (Cat. No.00TH8525)*, vol. 2, pp. 1325–1329 vol.2, 2000.

[16] T. Komine and M. Nakagawa, "Integrated System of White LED Visible-Light Communication and Power-Line Communication," in *The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 4, pp. 1762–1766 vol.4, 2002.

[17] "Visible Light Communications Consortium (VLCC)." `https://web.archive.org/web/20160715181135/vlcc.net/modules/xpage1/?ml_lang=en` (accessed July 25, 2022).

[18] J. Grubor, S. Randel, K.-D. Langer, and J. W. Walewski, "Broadband Information Broadcasting Using LED-Based Interior Lighting," *Journal of Lightwave Technology*, vol. 26, no. 24, pp. 3883–3892, 2008.

[19] hOME Gigabit Access project (OMEGA). `https://web.archive.org/web/20160331190445/http://www.ict-omega.eu/` (accessed July 25, 2022).

[20] IEEE, "IEEE 802.15 WPAN Visible Light Communication Interest Group (IGvlc)." `https://ieee802.org/15/pub/IGvlc.html` (accessed July 25, 2022).

[21] "IEEE Standard for Local and Metropolitan Area Networks–Part 15.7: Short-Range Wireless Optical Communication Using Visible Light.," *IEEE Std 802.15.7-2011*, pp. 1 – 309, 2011.

[22] Y. Suh, C.-H. Ahn, and J. K. Kwon, "Dual-Codeword Allocation Scheme for Dimmable Visible Light Communications," *IEEE Photonics Technology Letters*, vol. 25, no. 13, pp. 1274–1277, 2013.

[23] U.-L. Center, "Center for Ubiquitous Communication by Light." `https://www.uclight.ucr.edu` (accessed July 25, 2022).

[24] S. Photonics for Communication and Illumination, "Photonics Consortium at Penn State." `https://photonics.psu.edu` (accessed July 25, 2022).

[25] S. L. E. R. Center, "Synthesizing Light for the Benefit of Humanity." `http://www.bu.edu/smartlighting` (accessed July 25, 2022).

[26] G. Povey, "D-Light Project Hits Target." https://web.archive.org/web/20190902100234/http://visiblelightcomm.com/d-light-project-hits-target (accessed July 25, 2022).

[27] O. W. C. Group, "Super Receivers for Visible light communications." https://eng.ox.ac.uk/optical-wireless-communications/projects/super-receivers-for-visible-light-communications/ (accessed July 25, 2022).

[28] A. Pradana, N. Ahmadi, and T. Adionos, "Design and Implementation of Visible Light Communication System Using Pulse Width Modulation.," *International Conference on Electrical Engineering & Informatics (ICEEI)*, pp. 25 – 30, 2015.

[29] T. I. C. on Illumination (CIE), "CIE." https://cie.co.at/ (accessed July 25, 2022).

[30] M. Akram, L. Aravinda, M. Munaweera, G. Godaliyadda, and M. Ekanayake, "Camera Based Visible Light Communication System for Underwater Applications," in *IEEE International Conference on Industrial and Information Systems (ICIIS)*, pp. 1–6, 2017.

[31] M. Mossaad, *Spatial Optical Orthogonal Frequency-Division Multiplexing for Indoor Visible-Light Communication Systems*. PhD thesis, 2021.

[32] B. Aly, M. Elamassie, M. Uysal, and E. Kınav, "Experimental Evaluation of Unipolar OFDM VLC System on Software Defined Platform," in *15th International Conference on Telecommunications (ConTEL)*, pp. 1–6, 2019.

[33] Q. Wang, T. Mao, Z. Wang, and J. Wang, "Index Modulation-Aided OFDM for Visible Light Communications," in *Visible Light Communications*, InTech Rijeka, 2017.

[34] A. Devices, "JESD204 Serial Interface and JEDEC Standard Data Converters." https://www.analog.com/en/applications/landing-pages/001/jesd204-serial-interface-jedec-standard-data-converters.html (accessed July 25, 2022).

[35] A. Ardelean, "AD-FMCDAQ2-EBZ Clocking." https://wiki.analog.com/resources/eval/user-guides/ad-fmcdaq2-ebz/clocking (accessed July 25, 2022).

VITA

Gürol SAĞLAM

Education

- Sep. 2019 to Aug. 2022: M.Sc. in Computer Science, Özyeğin University.
- Jan. 2018 to June 2019: Minor in Computer Science, Özyeğin University.
- Sep. 2015 to June 2019: B.Sc. in Electrical & Electronics Engineering, Özyeğin University.
- Sep. 2010 to June 2015: Sırrı Yırcalı Anatolian High School.

Publications

• **Fast and Efficient Implementation of Lightweight Crypto Algorithm PRESENT on FPGA through Processor Instruction Set Extension**

A.Varici, G.Saglam, S.Ipek, A.Yildiz, S.Goren, A.Aysu, D.Iskender, T.B.Aktemur, H.F.Ugurdag, ***IEEE Xplore, 2019***,

IEEE East-West Design & Test Symposium (EWDTS) 2019 (Presentation)

• **FPGA Based DCO-OFDM PHY Transceiver for VLC Systems**

V.E.Levent, G.Saglam, H.F.Ugurdag, N.F.R.Annafianto, F.Aydin, S.W.Tesfay, B.Aly, M.Elamassie, B.Kebapci, M.Uysal, ***IEEE Xplore, 2019***,

11th International Conf. on Electrical and Electronics Engineering (ELECO) 2019 (Presentation)

Honors

The top BS alumnus in the Electrical & Electronics Engineering Department in the class of 2019 at Özyeğin University with a CGPA of 3.29 out of 4.00.