UNIVERSITAT POLITÈCNICA DE CATALUNYA - BARCELONATECH

BACHELOR'S THESIS

Real-Time MIMO receiver for mode-division multiplexing over coupled-mode optical fibers

---

Author:
Jie LUAN

Supervisors:
Dr. Sebastian RANDEL
Dr. Joan M. GENÉ BERNAUS

A thesis submitted in fulfilment of the requirements for the Bachelor Degree of Engineering Physics

in the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

and performed at

NOKIA Bell Labs

May 16, 2016

UNIVERSITAT POLITÈCNICA DE CATALUNYA - BARCELONATECH

Abstract

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Bachelor Degree of Engineering Physics

Real-Time MIMO receiver for mode-division multiplexing over coupled-mode optical fibers

by Jie LUAN

Today's demand for increasing information transmission capacity has led us to develop new technologies to beat the theoretical capacity limit. Among these, advances in digital fiber-optic communication have been a especially promising approach to achieve such target because of the high speed of light and its multiple physical dimensions to take advantage of.

This project aims to apply and expand our knowledge acquired in the first introductory course in signal theory to understand how real world digital communication systems work. In particular, we will focus on the multiple-input multiple-output (MIMO) digital signal processing (DSP) part of the digital fiber-optic communication.

Acknowledgements

Here I want to express my most sincere acknowledgements to the following people.

First I want to thank Dr. Joan M. Gené and Dr. Peter Winzer for giving me the amazing opportunity to participate in this internship program at Bell Laboratories, which constitutes an important period in my career.

I want to thank Dr. Sebastian Randel for his willingness to go through the algorithmic implementation step by step with me and for sharing his great ideas. I felt very fortunate to have such a patient supervisor and learned the way of analyzing complex problems from him.

I also want to thank my signal theory's teacher Dr. Antonio Pascual, for introducing me for the first time the intriguing problem of signal equalization in fiber-optic communication, which incited my interest towards the field.

And I want to thank Dr. Iñigo Molina for generously providing me some of the materials he used to deliver that lecture at the Optical Fibers Conference 2016 which helped me understand better some concepts of complex modulation I was working on.

Some people from the lab, I want to thank Borui Li and Haoshuo Chen with whom I had a good time having lunch and entertaining conversations together; and Christoph Füllner, who helped a lot on the experimental setup, without which we could not test our hardware performance.

The people who lived in my house have been very important to make my stay as enjoyable as it could be, therefore I want to thank my landlord's family the Yang's, and my roommates Nicholas Hu and Ivan Luo.

Thanks to express to my friends and teachers as well, especially the Yu's who gave me support during my stay and my visit to the OFC conference in Anaheim, and my high school teacher Rosa López for caring about my stay.

No need to say that my parents have always been a motor to me. Therefore this work would not have been possible either without their constant support and concerns.

Contents

List of Abbreviations

ADC	**A**nalog-to-**D**igital **C**onverter
AWGN	**A**dditive **W**hite **G**aussian **N**oise
BER	**B**it **E**rror **R**atio
CCF	**C**oupled-**C**ore **F**iber
CLB	**C**onfigurable **L**ogic **B**lock
CMA	**C**onstant-**M**odulus **A**lgorithm
CMOF	**C**oupled-**M**ode **O**ptical **F**iber
DC	**D**irect **C**urrent
DFT	**D**iscrete **F**ourier **T**ransform
DMGD	**D**ifferential **M**ode **G**roup **D**elay
DSP	**D**igital **S**ignal **P**rocessing
EDFA	**E**rbium-**D**oped **F**iber **A**mplifier
ECL	**E**xternal **C**avity **L**aser
FDAF	**F**requency-**D**omain **A**daptive **F**ilter
FFT	**F**ast **F**ourier **T**ransform
FMF	**F**ew-**M**ode **F**iber
FPGA	**F**ield-**P**rogramable **G**ate **A**rray
ICR	**I**ntegrated **C**oherent **R**eceiver
IQM	**I**nphase-**Q**uadrature **M**odulator
LMS	**L**east **M**ean-**S**quare
LO	**L**ocal **O**scillator
LPF	**L**ow-**P**as **F**ilter
MCF	**M**ulti**C**ore **F**iber
MDM	**M**ode-**D**ivision **M**ultiplexing
MIMO	**M**ultiple-**I**nput **M**ultiple-**O**utput
MSE	**M**ean **S**quare **E**rror
MZM	**M**ach-**Z**ehnder **M**odulator
PDM	**P**olarization-**D**ivision **M**ultiplexing
PL	**P**hotonic **L**antern
PPG	**P**rogrammable-**P**attern **G**enerator
QPSK	**Q**uadrature **P**hase-**S**hift **K**eying
SDM	**S**pace-**D**ivision **M**ultiplexing
SMF	**S**ingle-**M**ode **F**iber
SNR	**S**ignal-to-**N**oise **R**atio
VHDL	**V**HSIC **H**ardware **D**escription **L**anguage
VHSIC	**V**ery **H**igh **S**peed **I**ntegrated **C**ircuit
VOA	**V**ariable **O**ptical **A**ttenuator
WDM	**W**avelength-**D**ivision **M**ultiplexing

Chapter 1

Introduction

Just imagine we want to send a text message from the US to somebody in China. First you punch into your smartphone and then it goes to the nearest cell phone tower which receives the signal where electromagnetic wave is converted to light and goes to the nearest central office, where it is multiplexed and combined with millions of other text messages. Lots of these signals will terminate in American cities, but ours need to go into a fiber that goes across the Pacific Ocean. Then the signal reaches China, our text message is identified, it experiences the reverse process to what happened in the US until it reaches our friend. The trip takes 1/20-th of a second, that is the time it takes light to traverse 10000 km. This is happening millions and millions of times every hour across the globe and all of that is carried by the fiber-optic network that none of us can see but all of us will use.

So optical fibers have often been considered to offer large capacity to support the rapid traffic growth essential to our information society. However, as long-haul single-mode fiber (SMF) systems is approaching its capacity limit[1] (Fig. 1.1), new techniques must be developed to satisfy the increasing demand.

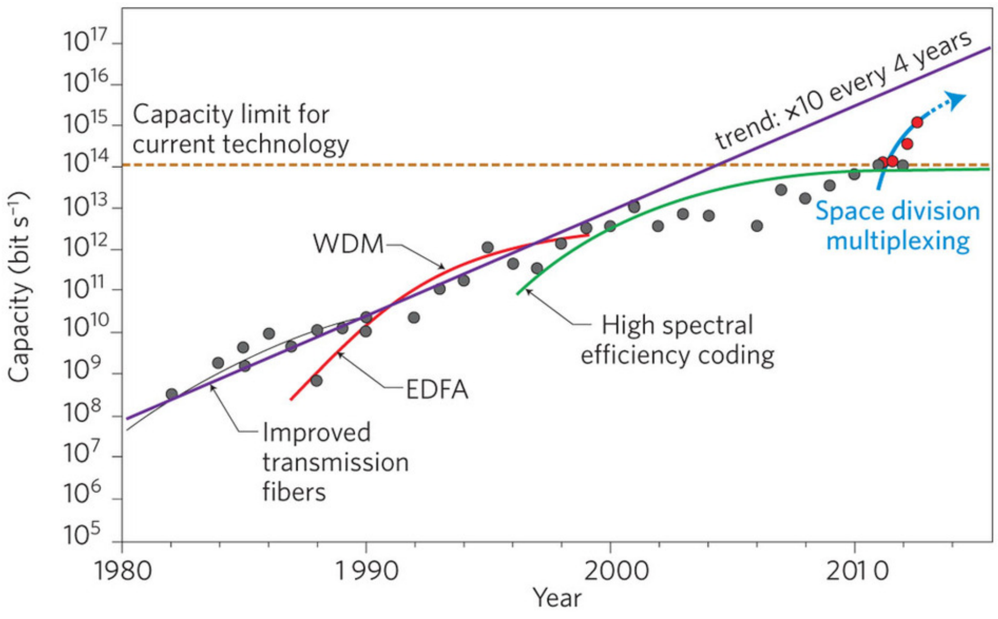


FIGURE 1.1: The capacity increase trend of approximately $\times 10$ bit/s every 4 years, as evidenced by different experiments using novel techniques over the last three decades.

[1]Such limit is derived from the Shannon information capacity limit as an extension for a nonlinear fiber channel, under quite broad assumptions. For more details see [7].

Fortunately there are still dimensions of lights to explore, as for instance the spatial, and research are being carried out across the globe. Chapter 3 will give us an idea of two of the schemes to manage parallel channels at the same time, the mode-division multiplexing and the space-division multiplexing.

In Chapter 2 we will walk through some concepts on digital communication theory which will help us gain a better understanding of our ultimate experiment, explained in Chapter 5, which is already an important step in opening more possibilities for optical transmission.

As for the MIMO DSP part, we will also need to study some of the adaptive filters algorithms. This will be done in Chapter 4

Chapter 2

Notions on Digital Communication

2.1 Description of the general problem

The general scheme for digital communication is shown as follows:

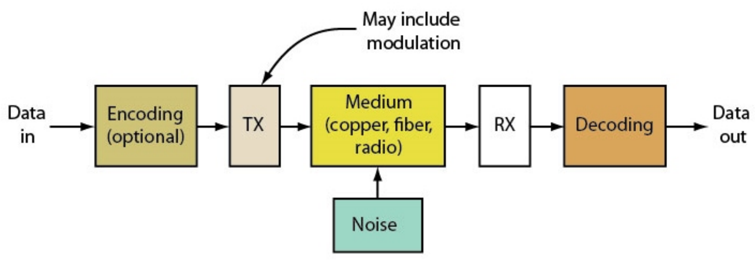


FIGURE 2.1: In fiber-optic communication, complex modulation is usually applied at the transmitter's terminal.

Maybe it seems obvious, but still it is worth saying that, as we live in a physical world, all signals with which we make communication possible are real. They are carried by real physical media such as copper cables and optical fibers in the form of current and light or, in general, a wave. Complex signals are just for making easy to understand the mathematics behind them, to carry out simple calculations.

2.2 Complex modulation for coherent communications

2.2.1 Complex envelope

Suppose we want to send and receive a real signal in time domain using a physical carrier over long distances. We know that real signals in time domain have hermitian symmetry in its frequency counterpart through Fourier transform. In other words, if $s(t) \in \mathbb{R}$, then $S(-\omega) = S^*(\omega)$. Therefore, without losing information, we may only work with the positive spectrum of $s(t)$, splitting it from the negative spectrum. Then, if the signal contains high-frequency components but concentrated around a certain frequency f_c, we may bring it down to around zero frequency by applying a frequency shift [1]. After performing these two steps, known as *downconversion*, let us call this new signal $\tilde{s}(t)$ or the *complex envelope* of $s(t)$.

$$\tilde{s}(t) = \frac{1}{\sqrt{2}}\Big(s(t) + j\hat{s}(t)\Big)e^{-j2\pi f_c t} \tag{2.1}$$

[1]Remember that zero frequency yields a DC signal, much more easier to work with.

where $\hat{s}(t)$ is called the Hilbert transform of $\tilde{s}(t)$, defined as the output of $\tilde{s}(t)$ through a filter with transfer function $H(f) = -j\text{sign}(f)$ [2]. It results from the even-odd parts decomposition of the filters transfer function, which is nothing but the Heaviside step function defined in the frequency domain:

$$U(f) = \frac{1}{2}(1 + jH(f)) = \begin{cases} 0, & f < 0 \\ 1, & f \geq 0 \end{cases} \tag{2.2}$$

The action becomes more clear if written in frequency domain:

$$\tilde{S}(f) = \sqrt{2}u(f + f_c)S(f + f_c) \tag{2.3}$$

It is clear from the definition of $\tilde{s}(t)$ that our original signal $s(t)$ can be retrieved by *upconverting* its complex envelope:

$$s(t) = \sqrt{2}\text{Re}\{\tilde{s}(t)e^{j2\pi f_c t}\} \tag{2.4}$$

Detection of complex modulated signals

Things become easier to understand by thinking complex signals as two parallel real signals carried by a physical medium using two of its independent properties. Usually such properties used are the amplitude, the frequency or the phase of a physical wave. First we couple the two real signals into one, thus forming a complex signal, or equivalently, a complex envelope. The information traveling through the medium is precisely the complex envelope, which is later decoded in the receiver's terminal to retrieve its two components, namely *in-phase* and *quadrature*, or I and Q.

In fact, mathematically speaking, the $I - Q$ components been sent are the real and the imaginary parts of the complex envelope , i.e.,

$$\tilde{s}(t) = s_I(t) + js_Q(t) \tag{2.5}$$

Therefore,

$$\begin{aligned} s(t) &= \sqrt{2}\text{Re}\{(s_I(t) + js_Q(t))e^{j2\pi f_c t}\} \qquad (2.6)\\ &= \sqrt{2}s_I(t)\cos(2\pi f_c t) - \sqrt{2}s_Q(t)\sin(2\pi f_c t) \qquad (2.7) \end{aligned}$$

Likewise, to obtain the $I - Q$ components in the transmitter's terminal, a similar scheme is implemented, plus the use of two LPF (low-pass filter), as we saw in the downconverting process.

A similar approach can be followed by taking the polar coordinates of the complex envelope. In this case, we can write

$$\tilde{s}(t) = e(t)e^{j\theta(t)} \tag{2.8}$$

being $e(t)$ and $\theta(t)$ the *envelope* and the *phase*. Thus,

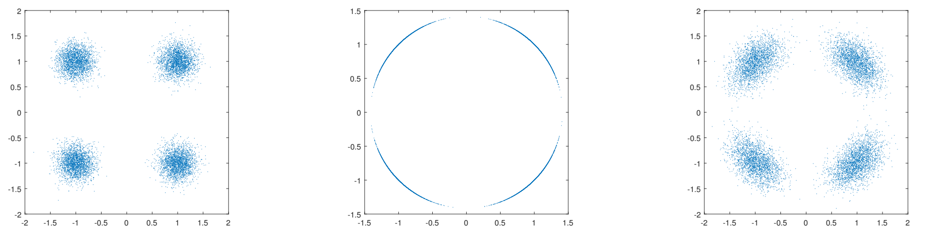
$$\begin{aligned} s(t) &= \sqrt{2}\text{Re}\{(e(t)e^{j\theta(t)}e^{j2\pi f_c t}\} \qquad (2.9)\\ &= e(t)\cos(2\pi f_c t + \theta(t)) \qquad (2.10) \end{aligned}$$

[2]The Hilbert transform performs a $-\pi$ phase shift at all frequencies for the input. Note that it is also real valued for real valued inputs because of its complex-conjugate symmetry.

2.2.2 Constellation diagram and Gray coding

A constellation diagram represents the possible symbols that may be selected by a given modulation scheme as points in the complex plane. Measured constellation diagrams can be used to recognize the type of distortions the signal may have suffered while propagating through the channel. Such distortions can be additive white Gaussian noise, phase noise, interference, or crosstalk in the case of coupled-core fibers, as we will see in Chapter 3. The MIMO DSP step carries out the recovery of the signals by selecting the most probable signal been transmitted, based on the Euclidean distance between the corrupted version of the received symbol and all the symbols of the constellation. The symbol closest to it will be the decision. Therefore, this is maximum likelihood detection, there is always a certain probability of wrong decision.

Oftentimes we can observe the effects that different types of distortions cause to the signals. For example, if the constellation points show a fuzzy pattern, then we can tell that Gaussian noise has been added, or a rotationally spreading points pattern on the diagram is a sign of phase noise, etc.



(A) (B) (C)

FIGURE 2.2: Effects on the received constellation diagram caused by (A) Gaussian noise, (B) phase noise, and (C) Gaussian plus phase noise.

Depending on the performance requirements of the transmission, whether more capacity for the channel or more robustness for the transmission's quality, one can choose among a large variety of modulating constellation schemes. Fig. 2.3 and 2.4 represent the mostly used ones in nowadays digital communication.

The symbols are coded with the so-called Gray coding scheme. A binary Gray code of length 2^N assigns to each of a contiguous set of 2^N symbols a binary string of length N such that two strings assigned to two adjacent symbols differ by exactly one bit. Gray codes are also known as single-distance codes, meaning that the Hamming distance[3] between adjacent symbols is always 1. Fig. 2.5 shows some Gray-coded constellation diagrams. Note that such code is not unique. In the case of QPSK, for instance, we can deduce that in case that the received symbol was corrupted, it would be more likely that it happened to be one of its reflections over the coordinate axis (one bit wrong) than its reflection over the origin of coordinates (both bits are wrong).

[3]The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. In another words, it measures the minimum number of substitutions required to change one string into the other, or the minimum number of errors that could have transformed one string into the other.

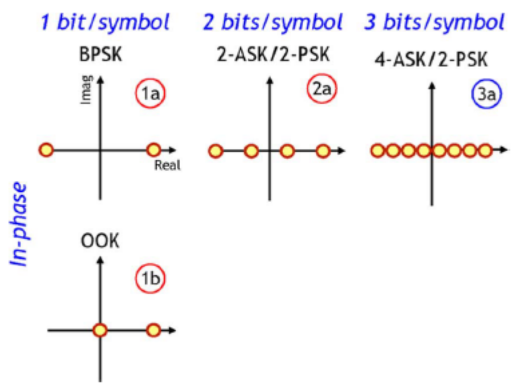


FIGURE 2.3: Examples of constellations using only one quadrature of the field (here the real part). The number of bits/symbol is given by $\log_2 M$ where M is the total number of symbols. The number $\log_2 M$ of symbols is used as the first digit of the format label.

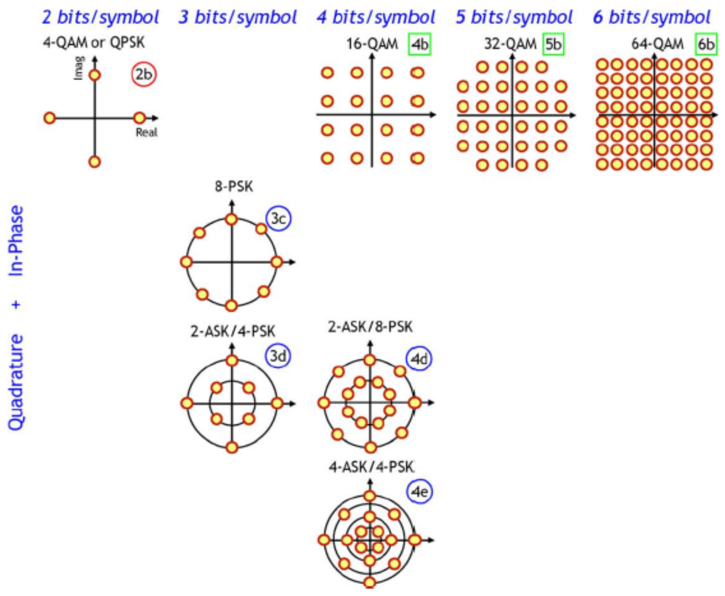


FIGURE 2.4: Examples of constellations that use both quadratures of the field.

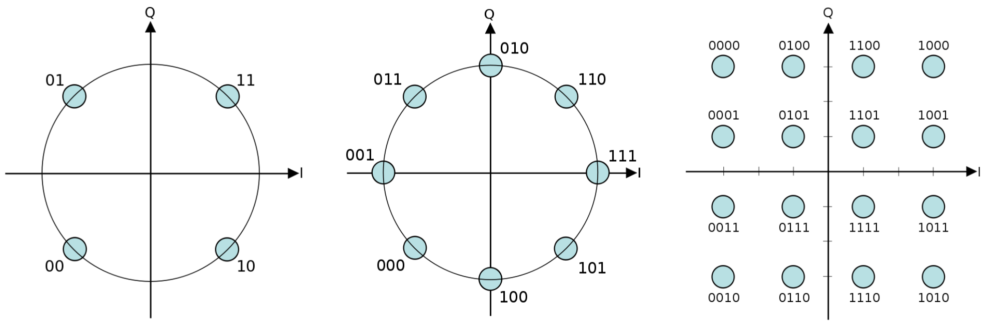


FIGURE 2.5: Gray-coded constellation diagram for QPSK, 8-PSK, and rectangular 16-QAM.

2.3 SNR in AWGN channels

Recall that an AWGN channel is a channel that is affect by a noise which is

- Additive: it is added to any noise that might be intrinsic to the channel.
- White: it has uniform power across the frequency band for the channel.
- Gaussian: it has a normal distribution in the time domain with an average time domain value of zero, i.e., $\sim N(0, \sigma^2)$.

Given the outputs of our adaptive filter algorithm, we want to estimate the SNR.

Formulation of the problem

The output can be modeled with the following random variable:

$$\boldsymbol{Y} = (\sqrt{P_S} e^{j\boldsymbol{\phi}} + \sqrt{\frac{P_N}{2}} \boldsymbol{n}_0 + j\sqrt{\frac{P_N}{2}} \boldsymbol{n}_1) \cdot e^{j\boldsymbol{\theta}} \tag{2.11}$$

where $\sqrt{\frac{P_N}{2}} \boldsymbol{n}_0 + j\sqrt{\frac{P_N}{2}} \boldsymbol{n}_1$ and $e^{j\boldsymbol{\theta}}$ are the modulus and phase noises, respectively. For convenience, we reexpress $\boldsymbol{Y}$ as the following, by defining $\alpha = \sqrt{P_S}$ and $\beta = \sqrt{P_N/2}$:

$$\boldsymbol{Y} = (\alpha e^{j\boldsymbol{\phi}} + \beta \boldsymbol{n}_0 + j\beta \boldsymbol{n}_1) \cdot e^{j\boldsymbol{\theta}} \tag{2.12}$$

We assume the random variables $\boldsymbol{\phi}$, $\boldsymbol{n_i}$ and $\boldsymbol{\theta}$ have the following values, all uniformly distributed within their intervals.
$\boldsymbol{\phi} \in \{\phi_0, \phi_1, \phi_2, \phi_3\}$ (In general it can be any finite set $\{\phi_i\}_{0 \le i < N}$)
$\boldsymbol{n_i}_{i=0,1} \sim N(0,1)$
$\boldsymbol{\theta} \in [0, 2\pi)$

Let us define $\boldsymbol{Z}$ to be $|\boldsymbol{Y}|^2$, hence

$$\begin{aligned}
\boldsymbol{Z} &= |\boldsymbol{Y}|^2 = \boldsymbol{Y}^\dagger \cdot \boldsymbol{Y} && (2.13)\\
&= (\alpha e^{j\boldsymbol{\phi}} + \beta(\boldsymbol{n}_0 + j\boldsymbol{n}_1)) \cdot (\alpha e^{-j\boldsymbol{\phi}} + \beta(\boldsymbol{n}_0 - j\boldsymbol{n}_1)) && (2.14)\\
&= \alpha^2 + \alpha\beta \Big[e^{j\boldsymbol{\phi}}(\boldsymbol{n}_0 - j\boldsymbol{n}_1) + e^{-j\boldsymbol{\phi}}(\boldsymbol{n}_0 + j\boldsymbol{n}_1) \Big] + \beta^2(\boldsymbol{n}_0^2 + \boldsymbol{n}_1^2) && (2.15)\\
&= \alpha^2 + \alpha\beta \Big[\boldsymbol{n}_0(e^{j\boldsymbol{\phi}} + e^{-j\boldsymbol{\phi}}) - j\boldsymbol{n}_1(e^{j\boldsymbol{\phi}} - e^{-j\boldsymbol{\phi}}) \Big] + \beta^2 && (2.16)\\
&= \alpha^2 + 2\alpha\beta \boldsymbol{n}_0 \cos \boldsymbol{\phi} - 2\alpha\beta \boldsymbol{n}_1 \sin \boldsymbol{\phi} + \beta^2 \boldsymbol{n}_0^2 + \beta^2 \boldsymbol{n}_1^2 && (2.17)
\end{aligned}$$

In the case that $\{\phi_0, \phi_1, \phi_2, \phi_3\} = \{0, \pi/2, \pi, 3\pi/4\}$, we will have four different possible values for $\boldsymbol{Z}$:

$$\begin{aligned}
\boldsymbol{Z_0} &= \alpha^2 + 2\alpha\beta \boldsymbol{n}_0 + \beta^2 \boldsymbol{n}_0^2 + \beta^2 \boldsymbol{n}_1^2 = (\alpha + \beta \boldsymbol{n}_0)^2 + (\beta \boldsymbol{n}_1)^2 && (2.18)\\
\boldsymbol{Z_1} &= \alpha^2 - 2\alpha\beta \boldsymbol{n}_1 + \beta^2 \boldsymbol{n}_0^2 + \beta^2 \boldsymbol{n}_1^2 = (\alpha - \beta \boldsymbol{n}_1)^2 + (\beta \boldsymbol{n}_0)^2 && (2.19)\\
\boldsymbol{Z_2} &= \alpha^2 - 2\alpha\beta \boldsymbol{n}_0 + \beta^2 \boldsymbol{n}_0^2 + \beta^2 \boldsymbol{n}_1^2 = (\alpha - \beta \boldsymbol{n}_0)^2 + (\beta \boldsymbol{n}_1)^2 && (2.20)\\
\boldsymbol{Z_3} &= \alpha^2 + 2\alpha\beta \boldsymbol{n}_1 + \beta^2 \boldsymbol{n}_0^2 + \beta^2 \boldsymbol{n}_1^2 = (\alpha + \beta \boldsymbol{n}_1)^2 + (\beta \boldsymbol{n}_0)^2 && (2.21)
\end{aligned}$$

We notice that indeed all the random variables in the parenthesis are normally distributed, and therefore $\boldsymbol{Z}$ can be compactly written as

$$\boldsymbol{Z} = \sum_{p=0}^{1} \boldsymbol{m}_p^2 \tag{2.22}$$

where $\boldsymbol{m}_0 \sim N(\alpha, \beta^2)$ and $\boldsymbol{m}_1 \sim N(0, \beta^2)$. This is the same as saying that

$$\boldsymbol{Z}/\beta^2 = \sum_{p=0}^{1} \boldsymbol{m}_p^2 \tag{2.23}$$

where $\boldsymbol{m}_0 \sim N(\alpha/\beta, 1)$ and $\boldsymbol{m}_1 \sim N(0, 1)$

So $\boldsymbol{Z}/\beta^2$ follows a noncentral χ^2 distribution with two degrees of freedom ($r = 2$) with noncentrality parameter $\lambda = \alpha^2/\beta^2$. Then,

$$\mu_{\boldsymbol{Z}/\beta^2} = \lambda + r = \alpha^2/\beta^2 + 2 \tag{2.24}$$

$$\sigma^2_{\boldsymbol{Z}/\beta^2} = 2(2\lambda + r) = 4(\alpha^2/\beta^2 + 1) \tag{2.25}$$

which implies

$$\mu_{\boldsymbol{Z}} = \beta^2 \cdot \mu_{\boldsymbol{Z}/\beta^2} = \alpha^2 + 2\beta^2 = P_S + P_N \tag{2.26}$$

$$\sigma^2_{\boldsymbol{Z}} = (\beta^2)^2 \cdot \sigma^2_{\boldsymbol{Z}/\beta^2} = 2\beta^2(2\alpha^2 + 2\beta^2) = (2P_S + P_N)P_N \tag{2.27}$$

Estimation of the SNR

In our formulation the signal and the noise are:

$$\boldsymbol{S} = \sqrt{P_S} e^{j\boldsymbol{\phi}} \cdot e^{j\boldsymbol{\theta}} \tag{2.28}$$

$$\boldsymbol{N} = \left(\sqrt{\frac{P_N}{2}} \boldsymbol{n}_0 + j \sqrt{\frac{P_N}{2}} \boldsymbol{n}_1 \right) \cdot e^{j\boldsymbol{\theta}} \tag{2.29}$$

Their power are obtained as:

$$Pow(\boldsymbol{S}) = |\boldsymbol{S}|^2 = \boldsymbol{S}^\dagger \cdot \boldsymbol{S} = P_S \tag{2.30}$$

$$Pow(\boldsymbol{N}) = |\boldsymbol{N}|^2 = \boldsymbol{N}^\dagger \cdot \boldsymbol{N} = \frac{P_N}{2}(\boldsymbol{n}_0^2 + \boldsymbol{n}_1^2) = P_N \tag{2.31}$$

So the SNR can be estimated as the ratio between these two values, i.e., P_S/P_N. So far we have the system formed by equations 2.26 and 2.27:

$$\begin{cases} \mu_{\boldsymbol{Z}} = P_S + P_N \\ \sigma^2_{\boldsymbol{Z}} = (2P_S + P_N)P_N \end{cases}$$

which has a unique solution, considering P_S and P_N to be positive:

$$\begin{cases} P_S = \sqrt{\mu^2_{\boldsymbol{Z}} - \sigma^2_{\boldsymbol{Z}}} \\ P_N = \mu_{\boldsymbol{Z}} - \sqrt{\mu^2_{\boldsymbol{Z}} - \sigma^2_{\boldsymbol{Z}}} \end{cases}$$

Hence,

$$SNR = \frac{P_S}{P_N} \tag{2.32}$$

$$= \frac{\sqrt{\mu_{\boldsymbol{Z}}^2 - \sigma_{\boldsymbol{Z}}^2}}{\mu_{\boldsymbol{Z}} - \sqrt{\mu_{\boldsymbol{Z}}^2 - \sigma_{\boldsymbol{Z}}^2}} \tag{2.33}$$

$$= \frac{1}{\sqrt{\frac{1}{1-\frac{\sigma_{\boldsymbol{Z}}^2}{\mu_{\boldsymbol{Z}}^2}}} - 1} \tag{2.34}$$

$$\tag{2.35}$$

which can be estimated using the statistical mean and variance for the theoretical ones.

Chapter 3

Multiplexing Techniques for Fiber-Optic Communication

Optical communication technology has been advancing rapidly for several decades, supporting our increasingly information-driven society and economy. Much of this progress has been in finding innovative ways to increase the data-carrying capacity of a single optical fiber [5]. However, as demand has grown and technology has developed, we have begun to realize that there is a fundamental limit to fiber capacity of about 100 Tb/s per fiber for systems based on conventional single-core single-mode optical fiber as the transmission medium.

Nevertheless, as light wave is defined by more parameters than just amplitude, we have more possibilities to encode information by using all of the light wave's degrees of freedom. Fig. 3.1 shows the mathematical description of the electric field of an electromagnetic wave with two polarization components E_x and E_y. These orthogonal components are used in polarization-division multiplexing (PDM) as two different channels to transfer independent signals. In wavelength-division multiplexing (WDM), different frequencies ω are applied as different channels for independent data transfer at these frequencies/wavelengths. For complex modulation schemes both the amplitude E and the phase ϕ of a light wave can be modulated for defining the above described symbols.

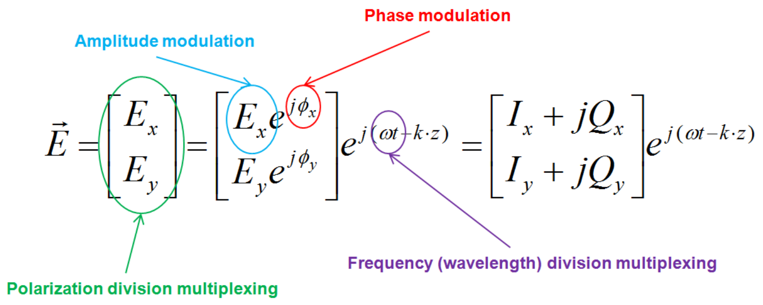

Phase modulation
Amplitude modulation
$\vec{E} = \begin{bmatrix} E_x \\ E_y \end{bmatrix} = \begin{bmatrix} E_x e^{j\phi_x} \\ E_y e^{j\phi_y} \end{bmatrix} e^{j(\omega t - k\cdot z)} = \begin{bmatrix} I_x + jQ_x \\ I_y + jQ_y \end{bmatrix} e^{j(\omega t - k\cdot z)}$
Frequency (wavelength) division multiplexing
Polarization division multiplexing

FIGURE 3.1: Mathematical description of an electromagnetic wave (electric field), where we see the I-Q components of each polarization.

Thus each of these dimensions can be used as parallel channels for transmitting independent signals, which constitutes the idea of *multiplexing*[1]. In this chapter we are going to introduce some of the basic multiplexing schemes.

[1]Sometimes it is shortened as *muxing*, which explains the origin of the *MUX* symbol that appears in block diagrams. The inverse process, i.e., the separation of a single channel into multiple parallel ones, is thus called *demultiplexing*.

3.1 Mode-division multiplexing (MDM)

An important concept and also a problem to deal with in MDM is called mode coupling. Although ideally the normal modes[2] of light propagation are independent of each other, in real systems such as the one inside the multi-mode fiber (MMF) there is some perturbation that causes different modes to interact between them, which is translated into a energy transfer and therefore mixture of information. Also there is a phenomenon called differential mode group delay (DMGD), which exhibits the different propagation speeds of different modes. The MIMO DSP part of our experiment precisely has to uncouple the received information and solve the DMGD issue.

Multi-mode fibers

MDM in MMFs was already proposed in 1982 [16], but subsequent experimental results were limited in bit rate and transmission distance, mainly because of the large number of supported modes and the large DMGD present in the fibers under study. Moreover, in MMFs, the distinguishable light paths have significant spatial overlap, and consequently signals are susceptible to couple randomly among the modes during propagation. Crosstalk occurs when light is coupled from one mode to another and remains in that mode on detection. To mitigate these impairments, MIMO equalization has to be implemented at the receiver, the complexity of which scales quadratically with the number of modes, making this approach unsuitable for long-haul transmission.

Recent advances have led to the development of fibers supporting a small number of modes, the so-called few-mode fibers (FMFs, Fig. 3.5 (c)), that have a low DMGD. The most significant demonstrations have so far concentrated on the simplest FMF, which supports a total of six polarization- and spatial modes. The three spatial modes are approximated by the linearly polarized (LP[3]) and the twofold degenerate pseudo-mode (Fig. 3.2).

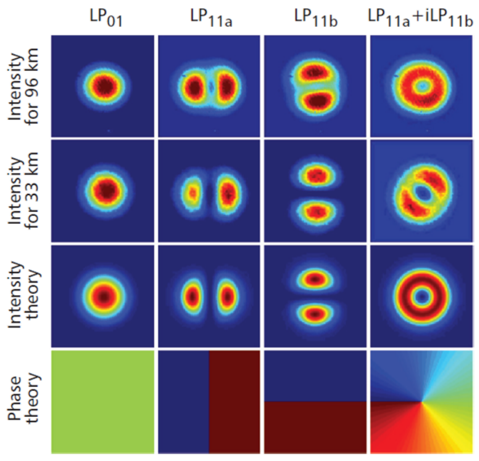


FIGURE 3.2: Theoretical and realistic intensity profiles of some linear combinations of LP_{01} and LP_{11} modes. The realistic ones are measured after 96-km and 33-km 6-mode FMF.

[2]Sometimes also known as eigenmodes, hence the name of *eigenlight* that appears in our experiment setup (see Sec. 5.1).

[3]A simple way to remember the shape of a LP mode is that a LP_{xy} mode has x diagonal cuts and y rings.

3.2 Space-division multiplexing (SDM)

Continued traffic growth has motivated the exploration of the last degree of freedom of light which we did not mention before: the spatial domain. Central to this effort has been the development of brand new fibers for space-division multiplexing (SDM) and mode-division multiplexing (MDM) [11].

The general scheme for realizing SDM transmission in optical fibers with N parallel paths is depicted in Fig. 3.3. In it we can distinguish four main steps, namely the generation of the signals by N parallel transmitters, the transmission through fiber of the multiplexed signal, the detection of the N parallel demultiplexed signals by N coherent receivers, and MIMO DSP of the received signals. We have to bear in mind that, whatever the type of fibers used, SDM always have to overcome the problem of crosstalk, which is any phenomenon by which a signal transmitted on one channel creates an undesired effect in another channel. The reason is simply the close proximity of the paths, which is inevitable if we want to achieve high core densities. Therefore the MIMO DSP part is designed to compensate any crosstalk that may have been introduced by the optical transmission system. This way, a capacity gain of a factor N can be ultimately achieved.

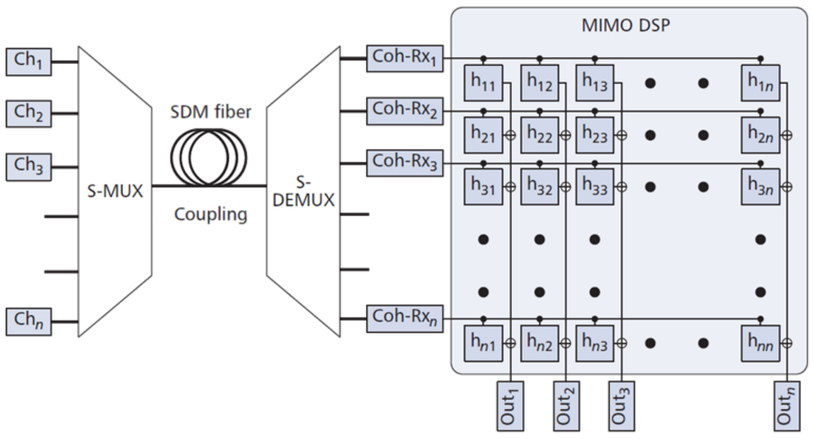


FIGURE 3.3: $N \times N$ SDM transmission system based on coherent MIMO DSP.

Next we are going to explore some of the approaches to realize SDM in a feasible way, by using different types of fibers.

Multicore fibers

A direct way of establishing distinguishable light paths would be through bundles of single-mode fibers (Fig. 3.5 (a)). However, it cannot achieve higher core densities, which makes them undesireble in order to use space efficiently.

A multicore fiber (MCF, Fig. 3.5 (b)), however, has distinct single-mode cores embedded directly into it, and therefore use them as independent light paths, at the same time achieving a higher core density. To limit crosstalk, one has to ensure that the fiber cores are well separated.

Nevertheless, these fibers are susceptible to fracture, meaning that MCF diameters greater than $\tilde{2}00$ μm are not considered practical, which imposes a fairly rigid limit on the number of cores that can be incorporated in MCFs for long-haul transmission. Most fibers to date have a hexagonal arrangement of seven cores. In this configuration, the central core has the highest level of crosstalk as it has six nearest neighbors, whereas the outer cores have only three nearest neighbors.

Coupled-core fibers

Although it may sound contradictory, studies have shown that strong mode coupling can potentially reduce the MIMO DSP complexity [15]. Even more surprising is the fact that mode coupling can be beneficial when applied to MCFs, by bringing the cores closer together to ensure strong linear mode coupling (Fig. 3.5 (d)), all establishing some supermodes[4] (Fig. 3.4) defined by the array of cores, which can then be used to provide spatial information channels for MDM to which MIMO can be applied. This enables higher spatial channel densities for MCFs than can be obtained using isolated cores designs. Indeed, in this case the MCF is essentially equivalent to a MMF.

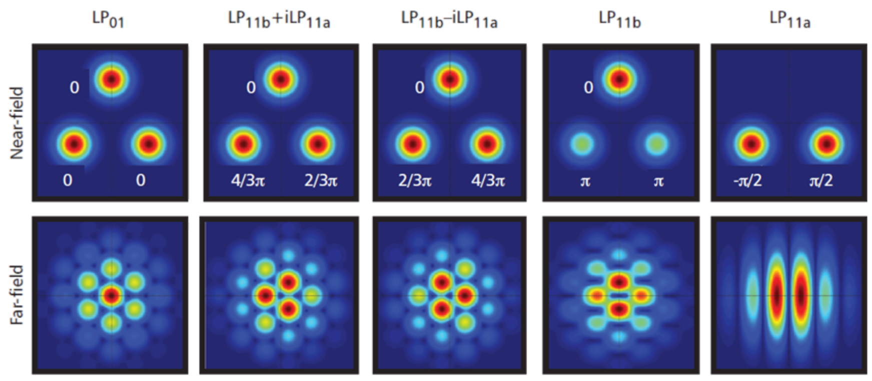


FIGURE 3.4: Linearly polarized super-modes and corresponding far-fields of the 3-core CCF.

[4]The presence of such strong coupling between cores suggests that the modes of the fiber can no longer be considered simple superpositions of the individual core modes, but the modes of the whole structure, the *supermodes*, have to be considered.

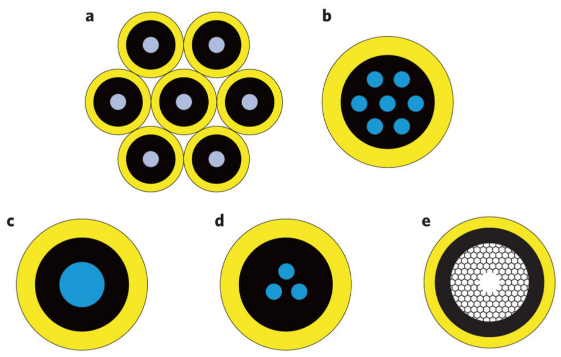


FIGURE 3.5: Different approaches for realizing MDM/SDM. Here are shown the cross section of (a) fiber bundles, (b) multicore fibers, (c) few-mode fibers, (d) coupled-core fibers, and (e) photonic bandgap fibers. But still, reducing cost and power consumption are formidable challenges.

3.3 Combining different techniques

As long as we exploit orthogonality in one or more of the physical dimensions shown in Fig. 3.6, multiplexing can be performed, i.e., independent bit streams can share a common transmission medium. Remember that two signals are orthogonal if messages sent in these two dimensions can be uniquely separated from one another at the receiver without impacting each other's detection performance. The amount of individual bit streams that can be packed onto a single transmission medium determines a system's aggregate capacity.

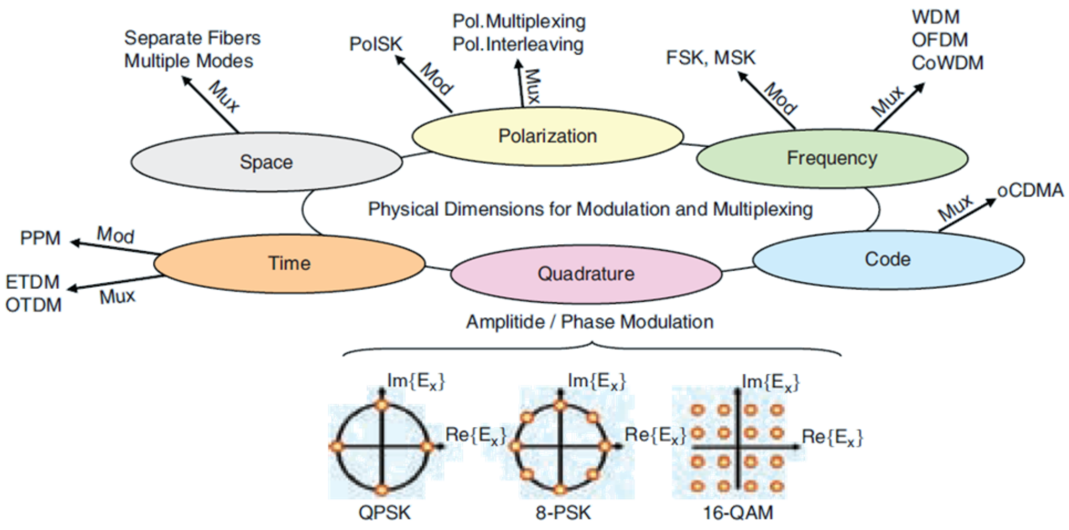


FIGURE 3.6: Different modulation and multiplexing schemes depending on the dimension used to convey parallel information signals.

So far we have seen that we can combine MDM and SDM by transmitting light waves of different modes while at the same time using the spatial dimension, that was the case of CCFs. But there are many other possible combination schemes. Fig. 3.7 gives another example of wavelength-division multiplexing combined with polarization-division multiplexing. We see that now each independent wavelength channel is extended in the polarization dimension, which is orthogonal to the wavelength dimension.

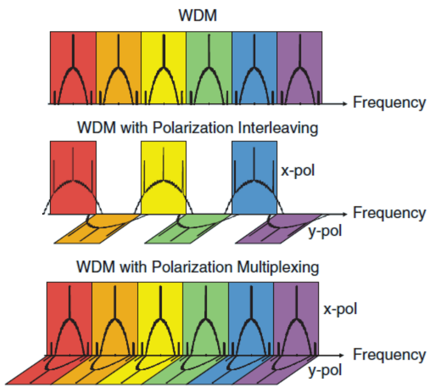

WDM
Frequency
WDM with Polarization Interleaving
x-pol
Frequency
y-pol
WDM with Polarization Multiplexing
x-pol
Frequency
y-pol

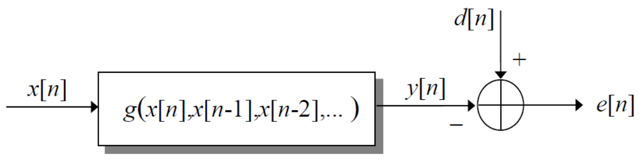
FIGURE 3.7: WDM can be combined with PDM.

Chapter 4

Adaptive Filters Algorithms

4.1 State of the problem

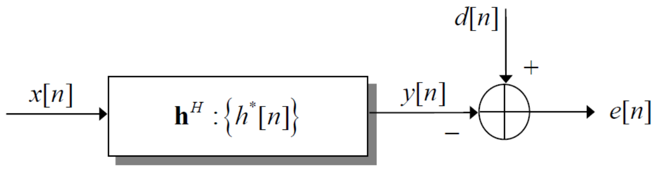
Consider the following scheme:



The target of our problem is to design a filter (a function $g(\mathbf{x})$) that minimizes a certain function of the error. A complete description of the variables for our problem is provided as follows.

- $x[n]$: Input to the system. For the sake of the analysis that follow, we define the input signal vector $\mathbf{x}[n] := (x[n], x[n-1], ..., x[n-N+1])^T$.
- $y[n]$: Output of the system, estimation of $d[n]$.
- $d[n]$: Desired signal, also called reference.
- $e[n]$: Estimation error.
- $\xi(g(\mathbf{x}[n]))$: Cost function we want to minimize.

In case that $\mathbf{x}$ and d are jointly distributed Gaussian processes, it can be shown that the optimum function $g(\mathbf{x})$ is linear. Therefore, the previous scheme is reduced to the following:



In other words, what we want to implement now is a linear filter $\mathbf{h} := (h[0], h[1], ..., h[N-1])^T$) and the problem is reduced to find the N coefficients of $\mathbf{h}$. Note that y can be expressed as $y = \mathbf{h}^H\mathbf{x}$.

4.2 The least mean-square (LMS) algorithm

Given the filter (or estimator) **h**, the mean square error (MSE) of **h** with respect to the reference d is defined as

$$\begin{aligned}
\xi(\mathbf{h}) :&= E\big[|e|^2\big] = E\big[|d - y|^2\big] = E\big[|d - \mathbf{h}^H\mathbf{x}|^2\big] && (4.1)\\
&= E\big[|(d - \mathbf{h}^H\mathbf{x})^H(d - \mathbf{h}^H\mathbf{x})|^2\big] && (4.2)\\
&= E\big[|d|^2\big] - E\big[d^*\mathbf{h}^H\mathbf{x}\big] - E\big[d(\mathbf{h}^H\mathbf{x})^H\big] + E\big[|\mathbf{h}^H\mathbf{x}|^2\big] && (4.3)\\
&= E\big[|d|^2\big] - \mathbf{h}^H E\big[d^*\mathbf{x}\big] - E\big[d\mathbf{x}^H\big]\mathbf{h} + \mathbf{h}^H E\big[\mathbf{x}\mathbf{x}^H\big]\mathbf{h} && (4.4)
\end{aligned}$$

The least mean-square (LMS) algorithm aims at finding the coefficients of the filter **h** that minimizes the MSE. However, this requires full knowledge of the expectation values in the expression we have just derived, for instance, the auto-correlation matrix $E\big[\mathbf{x}\mathbf{x}^H\big]$. Therefore we propose to minimize a slightly modified MSE, without the expectation operator:

$$\begin{aligned}
\tilde{\xi}(\mathbf{h}) :&= |e|^2 = |d - \mathbf{h}^H\mathbf{x}|^2 = |(d - \mathbf{h}^H\mathbf{x})^H(d - \mathbf{h}^H\mathbf{x})|^2 && (4.5)\\
&= |d|^2 - d^*\mathbf{h}^H\mathbf{x} - d(\mathbf{h}^H\mathbf{x})^H + |\mathbf{h}^H\mathbf{x}|^2 && (4.6)\\
&= |d|^2 - d^*\mathbf{h}^H\mathbf{x} - d\mathbf{x}^H\mathbf{h} + \mathbf{x}^H\mathbf{h}\mathbf{h}^H\mathbf{x} && (4.7)
\end{aligned}$$

which is the same expression just without the expectation operator, because the operator is linear.

Now, applying the rules of complex multivariate derivation, the gradient with respect to $\mathbf{h}^*$ is

$$\begin{aligned}
\nabla_{\mathbf{h}^*}\tilde{\xi}(\mathbf{h}) &= 0 - d^*\mathbf{x} - 0 + \mathbf{x}^H\mathbf{h}\mathbf{x} && (4.8)\\
&= (-d^* + \mathbf{x}^H\mathbf{h})\mathbf{x} = (-d^* + y^*)\mathbf{x} = -e^*\mathbf{x} && (4.9)
\end{aligned}$$

Therefore, using the steepest gradient approach to find the optimum **h**, the filter would be updated according to

$$\mathbf{h}_{n+1} = \mathbf{h}_n - \mu\nabla_{\mathbf{h}^*}\tilde{\xi}(\mathbf{h}_n) = \mathbf{h}_n + \mu e^*\mathbf{x} \tag{4.10}$$

4.3 Blind adaptation techniques

The design of the adaptive optimal Wiener filter using the LMS algorithm requires training a previously known sequence of signals before each adaptation. Although being accurate in case of convergence, this algorithm makes the adaptation slow if the channel is highly varying. An alternative would be a class of algorithms that do not require such a training set, however by losing accuracy due to the no availability of reference signals to train, so to assure the filter's optimality. Hence there is a trade off between high speed adaptation and degree of its accuracy.

4.3.1 The constant-modulus algorithm (CMA)

Among this class of algorithms, the so-called blind adaptations, there is the constant modulus algorithm (CMA), whose cost function to be minimized is defined as

$$J_{CMA} := E[(R_2 - |y|^2)^2], \qquad R_2 = \frac{E[|x[n]|^2]}{E[|x[n]|]} \tag{4.11}$$

The name of the algorithm comes from the fact that given a real number R_2, which is the *modulus* to be compared with, the cost function penalizes any values y that are far from R_2, bringing them closer to it when the algorithm is reaching to the minimum through steepest gradient approach. Thus, at convergence, the distance between the y's and the *modulus* remains more or less *constant*.

In a more general way, we could have defined our cost function to be[1]

$$J(\mathbf{h}) := J(y(\mathbf{h})) = E[|R_p - |y|^p|^q] \tag{4.12}$$

Sometimes, because of lack of knowledge about the expectations, we replace expectation by instantaneous value, just as we did with the LMS algorithm, by defining a modified MSE. Then the gradient is

$$\nabla_{\mathbf{h}^*} J(\mathbf{h}) = \nabla_{\mathbf{h}^*}\left[(R_p - |y|^p)^2\right]^{q/2} \tag{4.13}$$
$$= \frac{q}{2}\cdot\left[(R_p - |y|^p)^2\right]^{q/2-1}\cdot 2(R_p - |y|^p)\cdot\nabla_{\mathbf{h}^*}(-|y|^p) \tag{4.14}$$
$$\tag{4.15}$$

As

$$|y|^p = (y^* y)^{p/2} \tag{4.16}$$

and

$$\nabla_{\mathbf{h}^*}(y^* y) = \nabla_{\mathbf{h}^*}(\mathbf{x}^H \mathbf{h}\mathbf{h}^H \mathbf{x}) = \mathbf{x}^H \mathbf{h}\mathbf{x} = y^* \mathbf{x} \tag{4.17}$$

the previous calculation yields

$$\nabla_{\mathbf{h}^*} J(\mathbf{h}) = \frac{q}{2}\cdot\left[(R_p - |y|^p)^2\right]^{q/2-1}\cdot 2(R_p - |y|^p)\cdot\frac{-p}{2}|y|^{2(p/2-1)}\cdot y^*\mathbf{x} \tag{4.18}$$
$$= -\frac{pq}{2}\cdot(R_p - |y|^p)^{q-1}\cdot|y|^{p-2}\cdot y^*\mathbf{x} \tag{4.19}$$

Taking $p = 2$ and $q = 2$ which reduces to the case of constant modulus algorithm, the gradient is

$$\nabla_{\mathbf{h}^*} J(\mathbf{h}) = -2(R - |y|)\cdot y^*\cdot\mathbf{x} \tag{4.20}$$

and therefore the adaptation scheme for our filter is

$$\mathbf{h}_{n+1} = \mathbf{h}_n - \mu\nabla_{\mathbf{h}^*}\tilde{\xi}(\mathbf{h}_n) = \mathbf{h}_n + \mu(R - |y|)\cdot y^*\cdot\mathbf{x} \tag{4.21}$$

where we have absorbed the factor 2 in μ.

[1]The case for $q = 2$ is referred as Godard algorithm in the literature.

4.4 Frequency-domain adaptive filters (FDAF)

4.4.1 Block adaptive filtering

So far we have seen that in order to converge to the optimum linear filter its N coefficients are updated every time a new input is inserted, because the gradients depend explicitly on the input vector $\mathbf{x}$. What if such update occurs only after a block of data has been accumulated? This is the essence of the so-called block adaptive filtering algorithms, which reduce the complexity of their corresponding non-block counterparts by a factor proportional to the block's size.

Recall the LMS algorithm, with the coefficients update given by 4.21:

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \mu e^*[n]\mathbf{x}[n] \tag{4.22}$$

Each time a new input $x[n]$ is received, it causes the input vector $\mathbf{x}[n]$ to change and the recursion to be done. This is why before we did not write out explicitly indexes for e and $\mathbf{x}$ indicating the number of iteration, it was clear that the recursion uses the most recent updated e and $\mathbf{x}$. But now we can also perform the recursion, i.e., update the coefficients, after L input samples $x[n], x[n+1], ..., x[n+L-1]$ are received:

$$\mathbf{h}_{n+L} = \mathbf{h}_{n+L-1} + \mu e^*[n+L-1]\mathbf{x}[n+L-1] \tag{4.23}$$

From 4.21 we know the expression of $\mathbf{h}_{n+L-1}$, then

$$\mathbf{h}_{n+L} = \mathbf{h}_{n+L-2} + \mu e^*[n+L-2]\mathbf{x}[n+L-2] + \mu e^*[n+L-1]\mathbf{x}[n+L-1] \tag{4.24}$$

We keep substituting until we reach the last updated value we know, which is $\mathbf{h}_n$. The result is

$$\mathbf{h}_{n+L} = \mathbf{h}_n + \mu \sum_{m=0}^{L-1} e^*[n+m]\mathbf{x}[n+m] \tag{4.25}$$

Note that for the output $y[n]$, it is updated using the same $\mathbf{h}_n$ until L data samples are accumulated, according to

$$y[n+m] = \mathbf{h}_n^H \mathbf{x}[n+m] \tag{4.26}$$

Since 4.25 is a block update that operates at a lower sampling rate than that of the incoming data, it will be convenient to define a new time index k where one increment corresponds to L increments of the original index n. Without loss of generality, we can substitute $n = kL$ where n is an integer multiple of k. By factoring the argument $kL+L$ as $(k+1)L$ on the left-hand side of the equation and dropping the explicit dependence of $\mathbf{h}$ on L, we have the following equivalent block update:

$$\mathbf{h}_{k+1} = \mathbf{h}_k + \mu \sum_{m=0}^{L-1} e^*[n+m]\mathbf{x}[n+m] \tag{4.27}$$

which can also be rewritten as

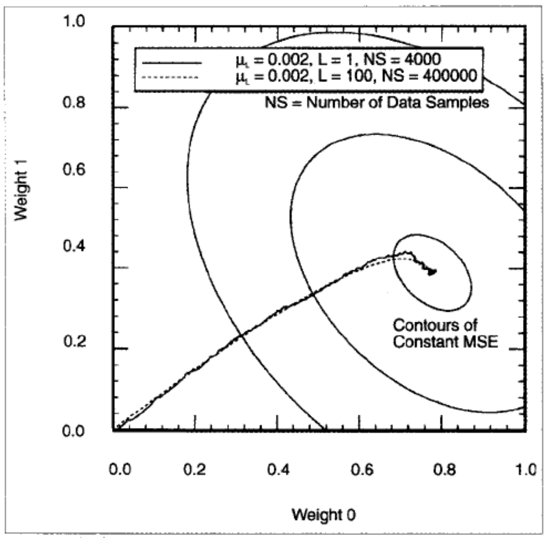


FIGURE 4.1: The averaged gradient estimate is a more accurate estimate of the true block gradient. We also see a smoother convergence.

$$\mathbf{h}_{k+1} = \mathbf{h}_k + \mu L \left(\frac{1}{L} \sum_{m=0}^{L-1} e^*[n+m]\mathbf{x}[n+m] \right) \tag{4.28}$$

$$= \mathbf{h}_k + \mu_L \frac{1}{L} \hat{\nabla}_k \tag{4.29}$$

Sometimes the summation term $\sum_{m=0}^{L-1} e^*[n+m]\mathbf{x}[n+m]$, denoted with the nabla symbol (as for gradient) with a hat (as for estimations), is called the block gradient estimate because it takes the place where before was the gradient. In the rewritten form, $1/L \cdot \hat{\nabla}_k$ is therefore called the averaged gradient estimate, which is a more accurate estimate of the true block gradient, as shown in Fig. 4.1. However, because now the *effective step size* μ_L is μ divided by L, its range of convergence is shrunk by a factor of L, meaning that the convergence rate is slower. Moreover, in order to achieve a similar convergence time compared to the non-block approach, in terms of number of iterations, the block LMS requires more data because of the lower convergence rate.

It is reported that $L = N$ is the most efficient value for the FFT algorithms.

Note that 4.28 is a linear correlation[2], hence it can be efficiently computed using FFT algorithms presented in Sec. 5.2.1, similarly with when computing a convolution, which turns out to be a product in frequency domain[3].

Furthermore, 4.26 and 4.28 can be compactly written in matrix forms:

$$\mathbf{y}[k+1] = \mathbf{h}_k^H \underset{\sim}{\mathbf{x}}[k] \tag{4.30}$$

$$\mathbf{h}_{k+1} = \mathbf{h}_k + \mu \underset{\sim}{\mathbf{x}}[k] \cdot \mathbf{e}^H[k] \tag{4.31}$$

[2]The linear correlation of two signal vectors $\mathbf{x}[n]$ and $\mathbf{y}[n]$ is defined as $z_{corr}[n] = \sum_{m=0}^{N} x[m]y[n+m]$, analogous to the convolution $z_{conv}[n] = \sum_{m=0}^{N} x[n]y[n-m]$.

[3]The procedure for computation of a convolution is therefore by first performing DFT, then a product of the signal vectors in frequency domain, and finally convert the result back to time domain by inverse DFT.

where we have defined the vectors

$$\mathbf{y}[k] = \big(y[kL] \quad y[kL+1] \quad \cdots \quad y[(k+1)L-1]\big)$$
$$\mathbf{e}[k] = \big(e[kL] \quad e[kL+1] \quad \cdots \quad e[(k+1)L-1]\big)$$

and the matrix

$$\underset{\sim}{\mathbf{x}}[k] = \begin{pmatrix} \vdots & \vdots & & \vdots \\ \mathbf{x}[kL] & \mathbf{x}[kL+1] & \cdots & \mathbf{x}[(k+1)L-1] \\ \vdots & \vdots & & \vdots \end{pmatrix}$$

Also bear in mind that

$$\mathbf{h}_k = \big(h_k[0] \quad h_k[1] \quad \cdots \quad h_k[L-1]\big)^T$$

4.4.2 The overlap-save algorithm

For convenience, in this section let us introduce another set of notations for our signals. As before, we will use bold letter to denote vectors. Brackets will be used to numerate samples, while parenthesis will specify the number of iteration. Therefore, the output can be expressed as[4]

$$\begin{aligned} y(n) &= \mathbf{x}(n)^T \mathbf{w}(n) && (4.32)\\ &= [x[n], x[n-1], ..., x[n-N+1]] \cdot [w_0(n), w_1(n), ..., w_{N-1}(n)]^T && (4.33)\\ &= \sum_{m=0}^{N-1} w_m(n) x[n-m] && (4.34)\end{aligned}$$

This means that the output vector **y** is a convolution between **x** and the filter **w**. As we know, this operation can be done faster in frequency domain by simply multiplying the transformed signals, according to the circular convolution theorem:

$$y_k(n) = \text{DFT}^{-1}(\text{DFT}(\mathbf{x}_k(n)) \cdot \text{DFT}(\mathbf{w}(n)) \qquad (4.35)$$

where we have defined $\mathbf{x}_k(n)$ with its components

$$x_k[n] \overset{\text{def}}{=} \begin{cases} x[n+kL] & 1 \le n \le L+M-1 \\ 0 & \text{otherwise.} \end{cases} \qquad (4.36)$$

and

$$y_k(n) \overset{\text{def}}{=} \mathbf{x}_k(n) * \mathbf{w}(n) \qquad (4.37)$$

Hence, what we have recovered is the circular convolution, not the linear convolution which is the one we are interested in. Fortunately, the latter can be performed using the *overlap-save* algorithm[5]. In order to understand how this algorithm works,

[4]Just to be consistent with our implementation, in which we use the letter w instead of h for the impulse response of the filter, and there is no transpose conjugate.

[5]In fact there is another algorithm for computing the linear convolution. It is called *overlap-add*. But due to its not so high usage, we are not going to present it in this work.

let us first illustrate the difference between linear and circular convolutions with the example in Fig. 4.2, with the two finite length discrete time signals been convoluted $\mathbf{x}_1[n] = [-1, 2, 3, -2, 0, 1]$ and $\mathbf{x}_2 = [5, 4, 3, 2, 1]$. Recall the definitions of the two types of convolutions, where N is the maximum of the lengths of the signals and $M \geq N$:

$$\mathbf{y}_{conv}[n] = \mathbf{x}_1[n] * \mathbf{x}_2[n] = \sum_{m=0}^{N-1} x_1[m]x_2[n-m] \tag{4.38}$$

$$\mathbf{y}_{circ}[n] = \mathbf{x}_1[n] \circledast \mathbf{x}_2[n] = \sum_{m=0}^{M-1} x_1[m]x_2[(n-m) \mod M] \tag{4.39}$$

Thus, graphically $\mathbf{y}_{conv}[n]$ can be performed by reflecting $\mathbf{x}_2$ and right shifting it n times, then lining it up vertically with the samples of $\mathbf{x}_1$, multiplying sample by sample and summing the products together. On the other hand, $\mathbf{y}_{circ}[n]$ can be performed graphically in a very similar manner, by using circular reflection and shifts of $\mathbf{x}_2$[6]. The result is

$$\mathbf{y}_{conv}[n] = \{-5 \quad 6 \quad 20 \quad 6 \quad 4 \quad 7 \quad 3 \quad 1 \quad 2 \quad 1\} \tag{4.40}$$

$$\mathbf{y}_{circ}[n]_{M=6} = \{-2 \quad 7 \quad 22 \quad 7 \quad 4 \quad 7\} \tag{4.41}$$

$$\mathbf{y}_{circ}[n]_{M=7} = \{-4 \quad 8 \quad 21 \quad 6 \quad 4 \quad 7 \quad 3\} \tag{4.42}$$

$$\mathbf{y}_{circ}[n]_{M=8} = \{-3 \quad 7 \quad 20 \quad 6 \quad 4 \quad 7 \quad 3 \quad 1\} \tag{4.43}$$

$$\mathbf{y}_{circ}[n]_{M=9} = \{-4 \quad 6 \quad 20 \quad 6 \quad 4 \quad 7 \quad 3 \quad 1 \quad 2\} \tag{4.44}$$

$$\mathbf{y}_{circ}[n]_{M=10} = \{-5 \quad 6 \quad 20 \quad 6 \quad 4 \quad 7 \quad 3 \quad 1 \quad 2 \quad 1\} = \mathbf{y}_{conv}[n] \tag{4.45}$$

$$\tag{4.46}$$

For $M > 10$, $\mathbf{y}_{circ}[n]$ is just $\mathbf{y}_{conv}[n]$ padded with $M - 10$ zeros to the right. We marked in red the subsets that match part of the linear convolution.

In general, the circular convolution will be the same as linear convolution (disregarding the padded zeros) if $M \geq L_1 + L_2 - 1$, i.e., the sum of the input signals' lengths minus 1. Otherwise only a portion of it will correspond to a subset of the linear convolution, as shown in the previous example. It is also true that if the circular convolution size is L_2 (5 in our example), then the last $L_1 - L_2 + 1$ samples of the circular convolution correspond to a linear convolution, that is 2 samples in our case.

The same conclusions can be generalized if instead of samples we have blocks. Imagine that we want to compute the convolutions of the following signals of size $2N - 1$ formed by concatenating blocks of size N:

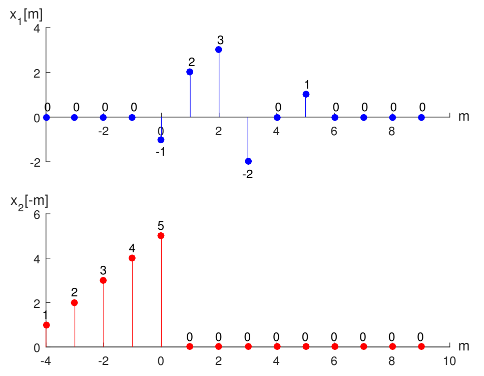
$$\tilde{\mathbf{x}}(k) = [x[kN-N], \cdots, x[kN-1], x[kN], \cdots, x[kN+N-1]] = [\mathbf{x}(k-1) \quad \mathbf{x}(k)] \tag{4.47}$$

$$\tilde{\mathbf{w}}(k) = [w_0[k], \cdots, w_{N-1}[k], 0, \cdots, 0] = [\mathbf{w}(k) \quad \mathbf{0}] \tag{4.48}$$

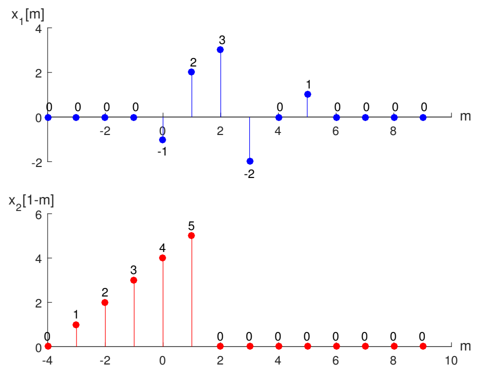
$$\tag{4.49}$$

Then it is true that the last $L_1 - L_2 + 1 = 1 - 1 + 1 = 1$ block of the circular convolution correspond to the linear convolution, being that precisely the one between the blocks

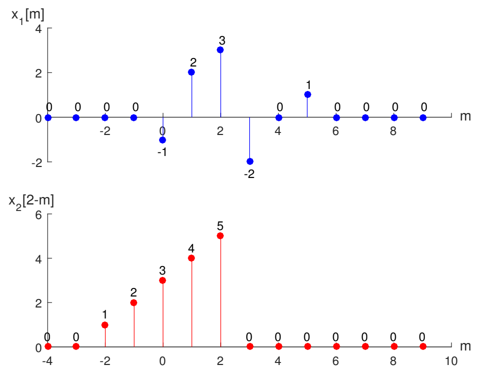
[6]As both convolutions are commutative, we could have performed them using the same graphical scheme just explained but instead by maintaining $\mathbf{x}_2$ and reflecting and shifting $\mathbf{x}_1$



$y_{conv}[0] = -1 \cdot 5 = -5 \qquad y_{circ}[0] = -1 \cdot 5 + 2 \cdot 0 + 3 \cdot 1 + (-2) \cdot 2 + 0 \cdot 3 + 1 \cdot 4 = -2$

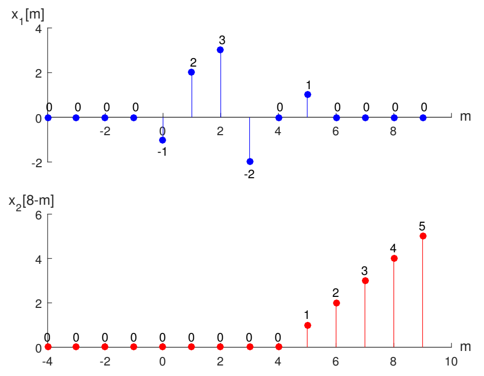


$y_{conv}[1] = -1 \cdot 4 + 2 \cdot 5 = 6 \qquad y_{circ}[1] = -1 \cdot 4 + 2 \cdot 5 + 3 \cdot 0 + (-2) \cdot 1 + 0 \cdot 2 + 1 \cdot 3 = 7$



$y_{conv}[2] = -1 \cdot 3 + 2 \cdot 4 + 3 \cdot 5 = 20 \qquad y_{circ}[2] = -1 \cdot 3 + 2 \cdot 4 + 3 \cdot 5 + (-2) \cdot 0 + 0 \cdot 1 + 1 \cdot 2 = 22$

$\vdots$



$y_{conv}[9] = 1 \cdot 1 = 1 \qquad y_{circ}[9] = y_{circ}[3] = -1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 + (-2) \cdot 5 + 0 \cdot 0 + 1 \cdot 1 = 7$

FIGURE 4.2: Linear and circular convolutions between two discrete time signals with $M = N = 6$.

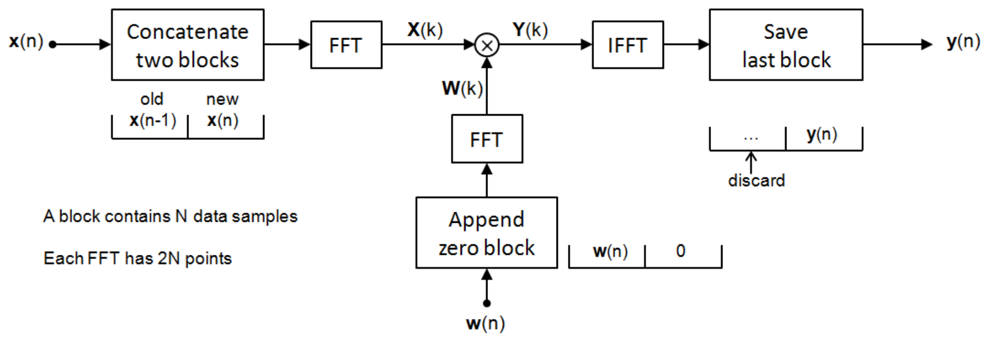


FIGURE 4.3: The overlap-save sectioning algorithm.

$\mathbf{x}(k)$ and $\mathbf{w}(k)$. Notice that now the lengths refer to the number of blocks instead of samples.

Fig. 4.3 shows the block diagram of the overlap-save algorithm, which is the one we have implemented. The name of the algorithm comes from the fact that only N new samples are introduced before the $2N$ sized FFT is computed, which means a 50% *overlap*, while for the output we *save*[7] half of the result.

4.4.3 General scheme of the FDAF algorithms

In order to compute the output $y(k)$ more efficiently, we have seen that we can work in frequency domain by taking advantage of the circular convolution theorem and the overlap-save algorithm. Therefore the adaptive filter's coefficients, $W(k)$, must be updated in frequency domain as well. Let us think now about the error term, which is the key component that closes our feedback loop. We can say that there are two possible ways to implement a FDAF, depending on whether the error computation is carried out in time domain or in frequency domain. Fig. 4.4 shows the block diagrams of the two approaches.

It turns out that for adaptive algorithms where the error is a linear function of the data (e.g., the LMS algorithm explained in Sec. 4.2), these two approaches may yield similar results. However, for algorithms that have linear error functions (e.g., the CMA algorithm of Sec. 4.3.1), the two structures can lead to very different results and only one may provide acceptable performance. In our case, we have implemented the CMA algorithm and it can be shown that it is preferable to use the first scheme. Since our algorithm is given by[8]

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \mu \cdot \mathbf{X}^*(k)(R^2 - |\mathbf{y}(k)|^2) \cdot \mathbf{y}^*(k) \tag{4.50}$$

the error term is

$$\mathbf{e}(k) = (R^2 - |\mathbf{y}(k)|^2) \cdot \mathbf{y}^*(k). \tag{4.51}$$

Hence, after plugging our adaptive algorithm into the scheme, the complete block diagram is as shown in Fig. 4.5. We use a reference modulus $R = \sqrt{2}$, which ideally is the modulus of the four undistorted QPSK points. This way we bring the received symbols close to one of the four points during convergence.

[7]Sometimes the name *overlap-discard* is given to the algorithm, for it *discards* half of the result.

[8]This result can be obtained by following the same arguments we did in Sec. 4.3.1, but with $y(n) = \mathbf{x}(n)^T\mathbf{w}(n)$.

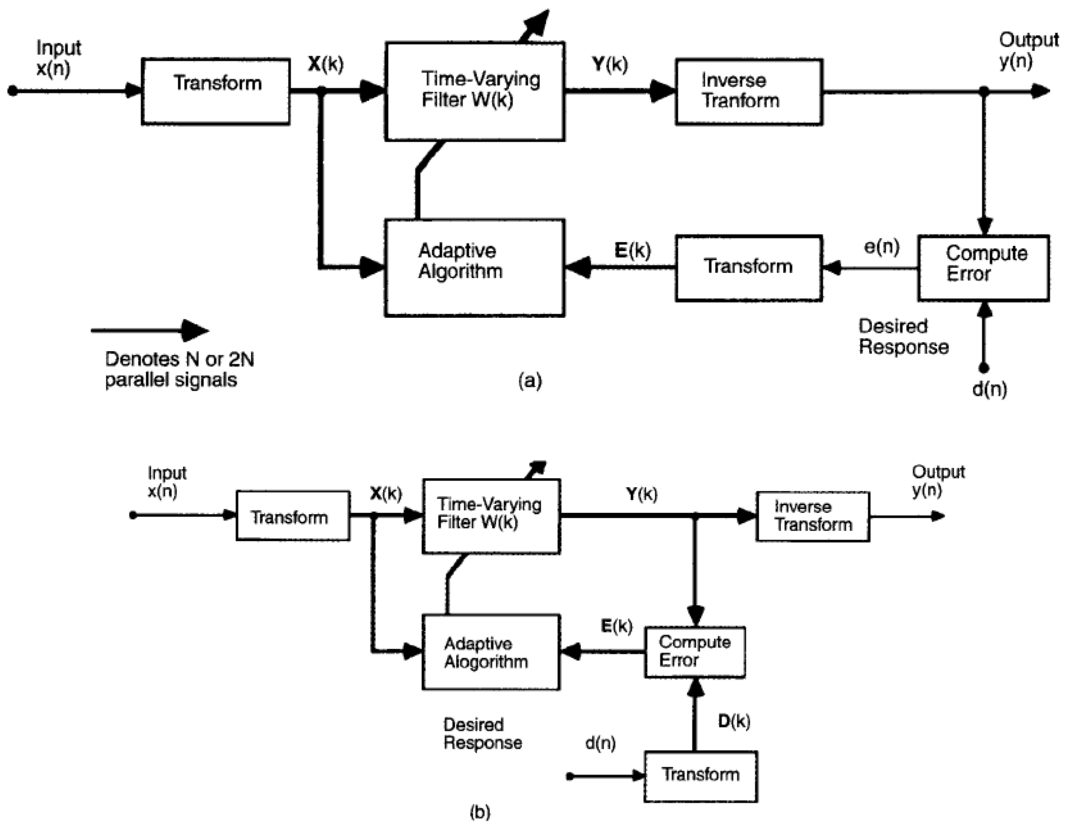


FIGURE 4.4: Two approaches for implementing a FDAF. In (a), error computation is performed in time domain, while in (b) it is done in frequency domain.

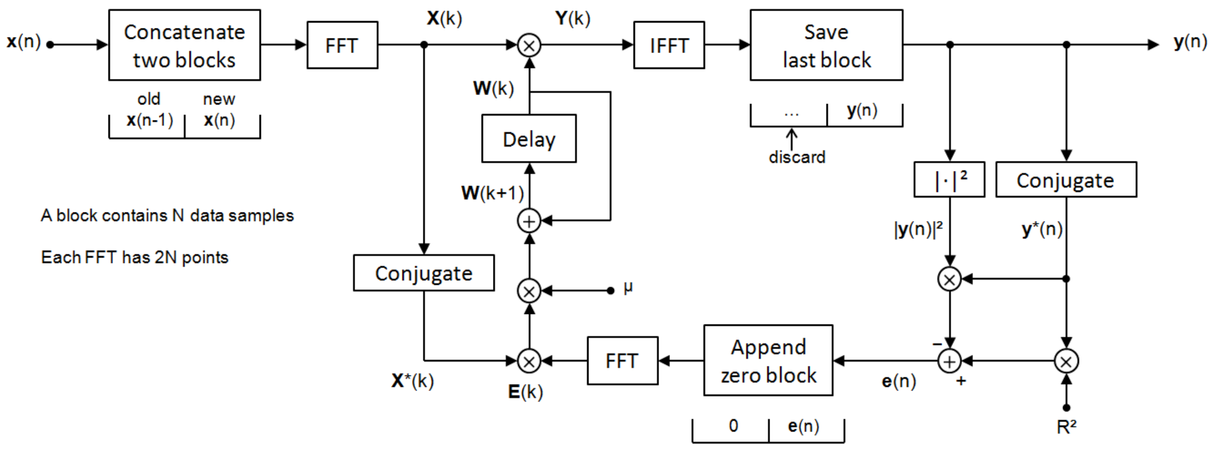


FIGURE 4.5: The overlap-save FDAF scheme using CMA as adaptive algorithm.

Chapter 5

The First Real-Time MIMO Receiver for MDM over CMOF

5.1 Setup of the experiment

The scheme of the coherent optical 6×6 MIMO experiment is shown in Fig. 5.1.

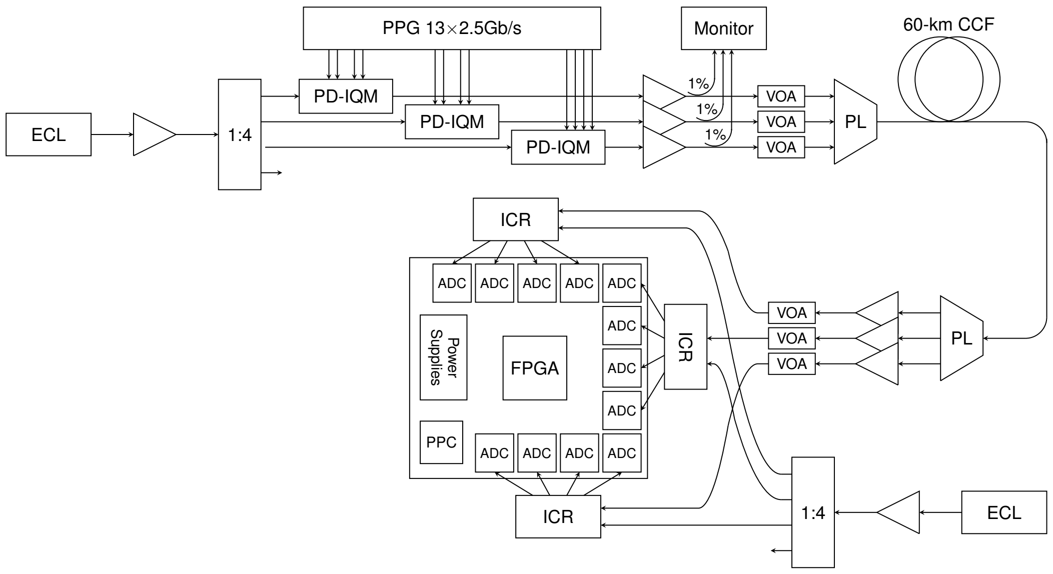


FIGURE 5.1: Experimental setup.

Next we are going to describe the different parts of the experiment with more details.

At the transmitter's terminal

- An external cavity laser (ECL) with a 1:4 beam splitter: With 100-kHz linewidth, it generates and split the optical beam into four beams, three to the Mach-Zehnder modulators and one for monitoring.
- A programmable-pattern generator (PPG): It generates twelve 2.5 Gb/s transmit signals modulated with 1/16-th pattern length delay-decorrelated copies of a pseudo-random binary sequence of length $2^{31} - 1$.
- Three polarization-diversity Lithium-Niobate inphase-quadrature modulators (PD-IQMs, the Mach-Zehnder modulators): They modulate the signals from the PPG onto an optical carrier at 1550 nm, generated from the ECL.

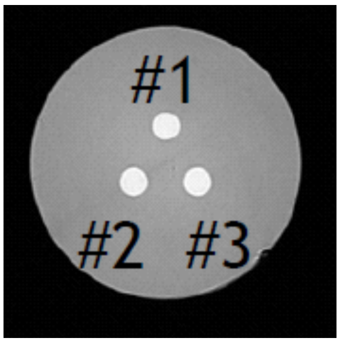


FIGURE 5.2: Cross section of the 60-km long CCF, in which we can distinguish the three cores for space-multiplex the signals.

- Three Erbium-doped fiber amplifiers (EDFAs): They amplify the polarization-division multiplexed QPSK (PDM-QPSK) signals from the output of the modulators.
- A monitor: 1% of the amplified signal is sent for monitoring, as a way to supervise the correct functionality of the components.
- Three variable optical attenuators (VOAs, also called eigenlights): They are used to adjust the desired input power of the signals launched into the PL.
- A photonic lantern (PL): It is used as a space-division multiplexer to orthogonally launch into the three cores of the CCF with a core pitch of 29 μm and a cross section shown in Fig. 5.2.

The physical channel through which the light travels is a 60-km CCF.

At the receiver's terminal

- A photonic lantern (PL): Using this second PL, demultiplexing is performed to separate the coupled signals from the three cores of the CCF.
- Three Erbium-doped fiber amplifiers (EDFAs): The received signals are strongly attenuated after been sent through the 60-km fiber, thus the need to amplify them.
- Three variable optical attenuators (VOAs): As before, the VOAs allow us to adjust the input power for the next detection step.
- A local oscillator ECL (with a 1:4 beam splitter): With the same characteristics of the ECL at the transmitter's terminal (100-kHz linewidth), it generates four beams, three for the ICRs and one for monitoring.
- Three polarization-diversity integrated coherent receivers (ICRs): They detect the signals using the scheme that will be explained in Sec. 5.1.2.
- The FPGA board (Fig. 5.3): The 28-layer printed circuit board implements the MIMO DSP in real-time.

Fig. 5.4 shows the block diagram of the FPGA board that performs the overlap-save algorithm implemented with VHDL.

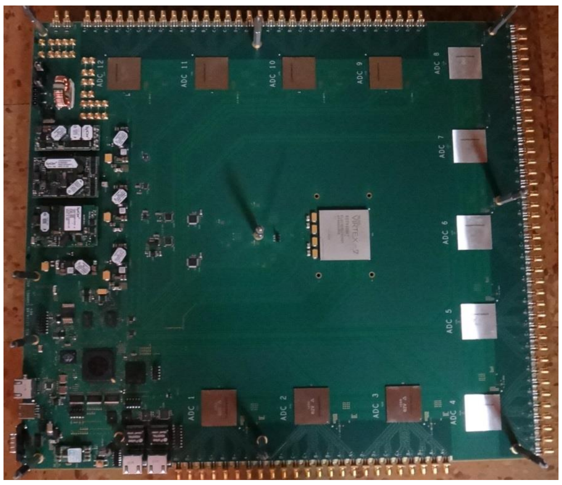


FIGURE 5.3: The FPGA board, in which we can see the ADCs arranged as in the experimental setup (fig 5.1).

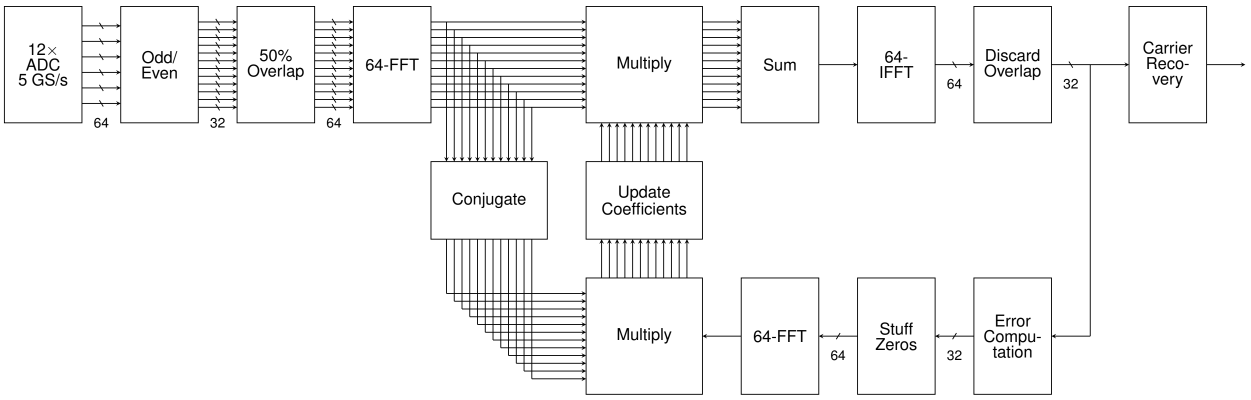


FIGURE 5.4: Real-time DSP block diagram.

5.1.1 The Mach-Zehnder modulator

It turns out that a constant or varying electric field can induce a change in the refractive index (birefringence) in an optical medium, proportionally to the strength of the field. This phenomenon is known as the linear electro-optic effect or Pockels effect in honor of its discoverer[1]. Hence, a light wave can be phase modulated, without change in polarization or intensity, using an electro-optic crystal and an input polarizer in the proper configuration (Fig. 5.5).

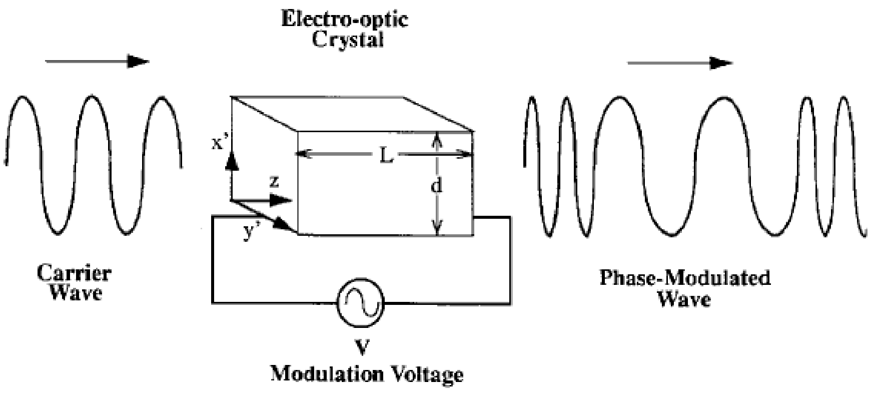


FIGURE 5.5: Only certain crystalline solids show the Pockels effect. The one employed in our experiment is lithium niobate ($LiNbO_3$).

A Mach-Zehnder modulator (MZM for short) is used for controlling the amplitude of an optical wave. The input waveguide is split up into two waveguide Mach-Zehnder interferometer arms. If a difference of voltages is applied across the arms, a phase shift is induced for the wave passing through the arms, according to the Pockels effect.

Mathematically speaking, if we denote E_{in} and E_{out} for the input and output waves, then it satisfies that

$$E_{out} = E_{in} \cdot ((1-\alpha)e^{-j\frac{V_1}{V_{ref}}} + \alpha e^{-j\frac{V_2}{V_{ref}}}) \tag{5.1}$$

where α is known as the interferometric splitting ratio, which is 1/2 in the ideal case (half the power for the arm having the wave $e^{-j\frac{V_1}{V_{ref}}}$ and half for the other). In our case we have the so-called *push-pull* configuration (Fig. 5.6), which means that $V_1 = -V_2$, i.e., the applied voltages on the two arms are inverted. In the ideal case ($\alpha = 1$) this implies that

$$E_{out} = E_{in} \cdot (\frac{1}{2}e^{j\theta} + \frac{1}{2}e^{-j\theta}) = E_{in} \cdot \cos\theta \tag{5.2}$$

where we have defined $\theta := -V_1/V_{ref}$. In other words, when the two arms are recombined, the phase difference between the two waves is converted to an amplitude modulation of the input wave.

By adjusting V_1 and V_2 we can phase modulate the two light waves in different ways and make them interfere through the Mach-Zehnder interferometer, resulting in the modulation scheme we are interested in. Fig. 5.7 shows that we can modulate with the QPSK scheme by applying a phase change of $\pi/2$.

[1] In some materials such change is proportional to the square of the field, in which case the phenomenon is called quadratic electro-optic effect or Kerr effect.

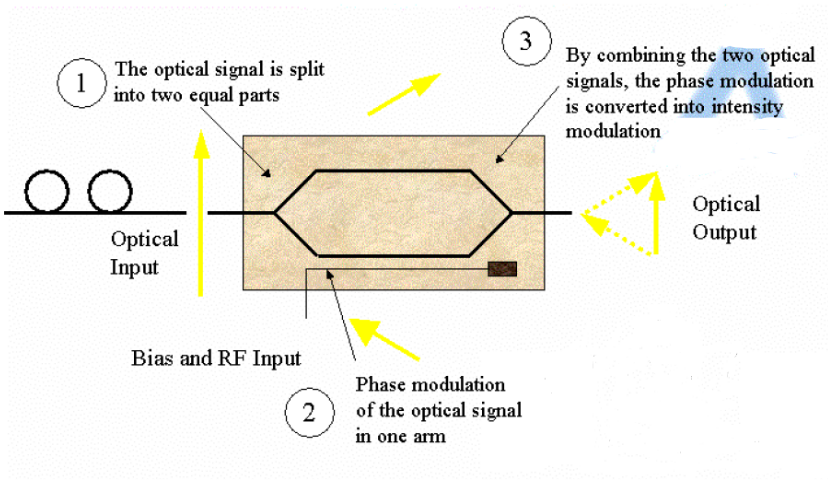

1
The optical signal is split into two equal parts
3
By combining the two optical signals, the phase modulation is converted into intensity modulation
Optical Input
Optical Output
Bias and RF Input
2
Phase modulation of the optical signal in one arm

FIGURE 5.6: The Mach-Zehnder modulator *push-pull* configuration.

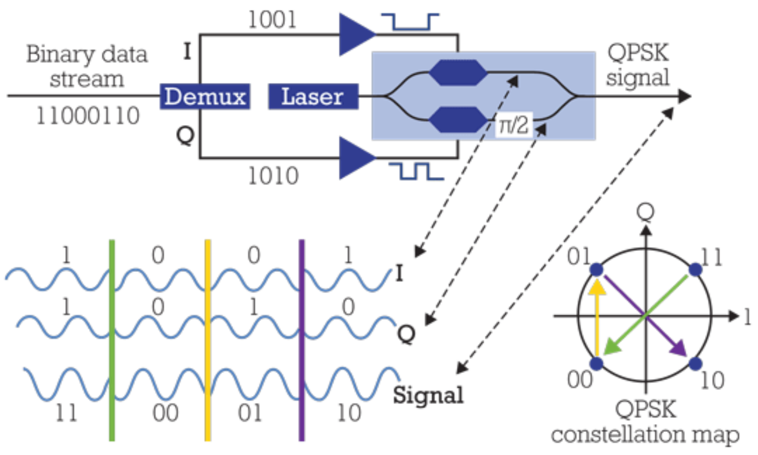


FIGURE 5.7: Transmitter setup for the modulation of a QPSK signal.

All the above analysis applies for one of the two polarizations a single core of the CCF carries. The output wave carries exactly the I-Q components of the corresponding polarization. To make the other polarization we need to implement the same scheme plus another phase rotation. The 10 Gb/s dual parallel nested MZM we use for our experiment is comprised of two matched, high-speed MZMs that are integrated in parallel inside an Mach-Zehnder super structure. Such super structure functions as a phase modulator.

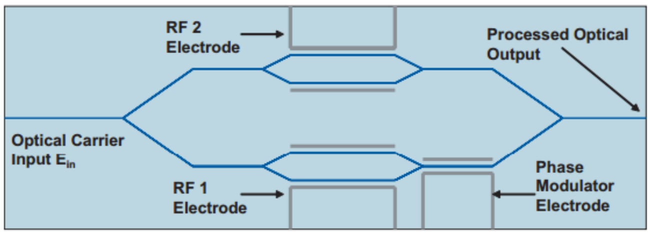


FIGURE 5.8: 10 Gb/s dual parallel Mach-Zehnder Modulator configuration.

5.1.2 The local oscillator

Upon reception, the optical signals are detected with a photodiode, which converts the optical power into an electrical current, sometimes called *photocurrent*. The photocurrent originating in the photodiode is directly proportional to the product of the optical signal and its complex conjugate, i.e., its modulus or amplitude (or power in a physical sense).

In order to successfully recover the in-phase and quadrature components of the signal, a local oscillator is needed at the receivers terminal, as depicted in Fig. 5.9.

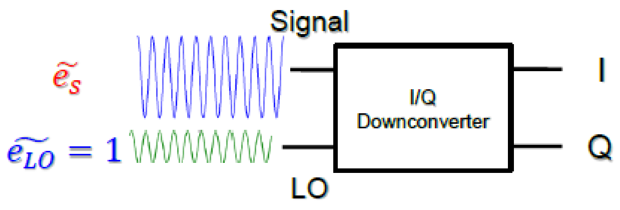


FIGURE 5.9: The local oscillator, which generate another signal that interacts with the optical one before being detected with the photodiode.

The interaction between the optical signal $\tilde{e}_s(t)$ and the LO's signal $\tilde{e}_{LO}(t)$ are performed through the mechanism shown in Fig. 5.10.

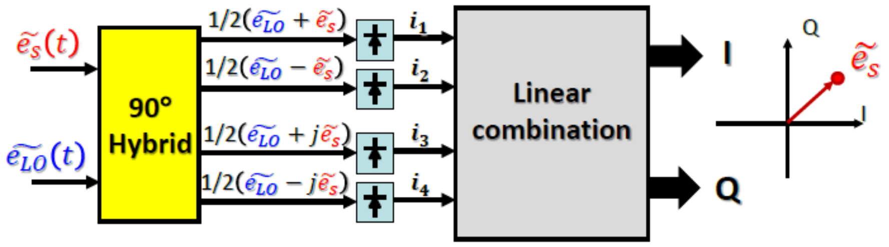

$\tilde{e}_s(t)$
$\tilde{e}_{LO}(t)$
90° Hybrid
$1/2(\tilde{e}_{LO} + \tilde{e}_s)$
$1/2(\tilde{e}_{LO} - \tilde{e}_s)$
$1/2(\tilde{e}_{LO} + j\tilde{e}_s)$
$1/2(\tilde{e}_{LO} - j\tilde{e}_s)$
i_1
i_2
i_3
i_4
Linear combination
I
Q
Q
I
$\tilde{e}_s$

FIGURE 5.10: Schematic of the downconverter.

Taking $e_{\tilde{L}O}(t) = 1$ as a reference for simplicity, we have that the photocurrents detected by the photodiodes are

$$i_1 = |1/2 \cdot (1 + \tilde{e}_s)|^2 = 1/4 + 1/4 \cdot |\tilde{e}_s|^2 + 1/2 \cdot Re\{\tilde{e}_s\} \tag{5.3}$$
$$i_1 = |1/2 \cdot (1 + \tilde{e}_s)|^2 = 1/4 + 1/4 \cdot |\tilde{e}_s|^2 - 1/2 \cdot Re\{\tilde{e}_s\} \tag{5.4}$$
$$i_1 = |1/2 \cdot (1 + \tilde{e}_s)|^2 = 1/4 + 1/4 \cdot |\tilde{e}_s|^2 - 1/2 \cdot Im\{\tilde{e}_s\} \tag{5.5}$$
$$i_1 = |1/2 \cdot (1 + \tilde{e}_s)|^2 = 1/4 + 1/4 \cdot |\tilde{e}_s|^2 + 1/2 \cdot Im\{\tilde{e}_s\} \tag{5.6}$$
$$\tag{5.7}$$

Then the $I - Q$ components are obtained by simple substraction of two photocurrent signals:

$$I = i_1 - i_2 = Re\{\tilde{e}_s\} \tag{5.8}$$
$$Q = i_4 - i_3 = Im\{\tilde{e}_s\} \tag{5.9}$$
$$\tag{5.10}$$

5.2 The DSP implementation

5.2.1 The fast Fourier transform (FFT) algorithm

As we have seen in Chapter 4, the FFT is a crucial block in our DSP implementation, so choosing a good scheme will help us build the DSP algorithm more efficient. Therefore we spent some time implementing and studying the complexity of this so-considered one of the top 10 algorithms of the twentieth century.

Suppose we want to calculate the discrete Fourier transform (DFT) of a finite-length sequence $x[n]$ ($0 \leq n \leq N-1$), defined as

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk} \tag{5.11}$$

Sometimes this is rewritten as $\sum_{n=0}^{N-1} x[n] W_N^{nk}$, where $W_N^n = e^{-j\frac{2\pi}{N}n}$, $0 \leq n \leq N-1$, are the $N-$th roots of unity.

Assuming that the factors W_N^{nk} are precomputed, the direct computation of the DFT would require $O(N^2)$ operations, that is, for each k, $O(N)$ operations for complex multiplications and additions, and there are N k's in total.

Nevertheless, it turns out that we can reuse some of the calculated results in order to lower the computation complexity. Starting with $N = 2$, the formula for DFT gives

$$X[0] = x[0] + x[1] \tag{5.12}$$
$$X[1] = x[0] - x[1] \tag{5.13}$$

The signal-flow graph connecting the inputs $x[n]$ to the outputs $X[k]$ can be represented by a so-called butterfly diagram, for its shape resembles a butterfly:

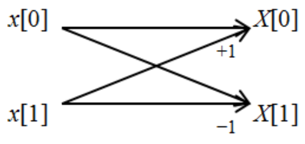


FIGURE 5.11: The butterfly diagram, in which we can see that $x[1]$ is multiplied by the weights on the arrows (± 1) before being added.

With $N = 4$ we get

$$X[0] = x[0] + x[1] + x[2] + x[3] \tag{5.14}$$
$$X[1] = x[0] - jx[1] - x[2] + jx[3] \tag{5.15}$$
$$X[2] = x[0] - x[1] + x[2] - x[3] \tag{5.16}$$
$$X[3] = x[0] + jx[1] - x[2] - jx[3] \tag{5.17}$$
$$\tag{5.18}$$

which can be rearranged into

$$X[0] = (x[0] + x[2]) + (x[1] + x[3]) \tag{5.19}$$
$$X[1] = (x[0] - x[2]) - j(x[1] - x[3]) \tag{5.20}$$
$$X[2] = (x[0] + x[2]) - (x[1] + x[3]) \tag{5.21}$$
$$X[3] = (x[0] - x[2]) + j(x[1] - x[3]) \tag{5.22}$$
$$\tag{5.23}$$

Note that we reduced the number of complex additions and multiplications by reusing some of the calculated values. In the signal-flow graph we observe a similar butterfly-like structure.

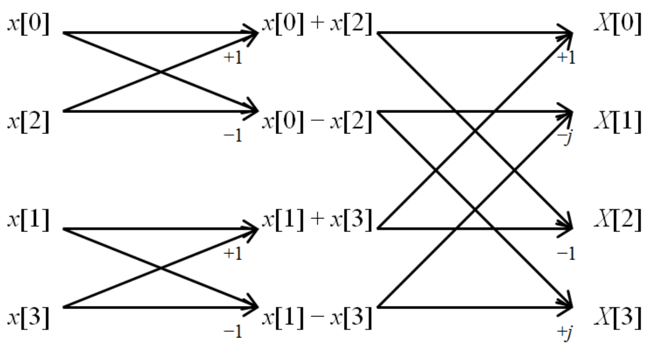


FIGURE 5.12: The butterfly diagram of a 4-point FFT.

This idea can be extended to any N power of 2. The diagram for $N = 16$ is shown in Fig. 5.13.

The radix-2 algorithm

Let us suppose now that N is even. Also let us define two subsequences of $x[n]$, namely $e[n]$ and $o[n]$ (for *even* and *odd*), as the even and odd samples of $x[n]$, i.e., $e[n] = x[2n], o[n] = x[2n+1], 0 \leq n \leq N/2 - 1$.

It turns out that the DFT of the original sequence, $X[k]$, can be calculated with the DFT's of the two subsequences, $E[k]$ and $O[k]$. Here is the proof.

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}nk} \tag{5.24}$$
$$= \sum_{m=0}^{N/2-1} x[2m]e^{-j\frac{2\pi}{N}2mk} + \sum_{m=0}^{N/2-1} x[2m+1]e^{-j\frac{2\pi}{N}(2m+1)k} \tag{5.25}$$
$$= \sum_{m=0}^{N/2-1} x[2m]e^{-j\frac{2\pi}{N/2}mk} + e^{-j\frac{2\pi}{N}k} \sum_{m=0}^{N/2-1} x[2m+1]e^{-j\frac{2\pi}{N/2}mk} \tag{5.26}$$
$$= E[k] + e^{-j\frac{2\pi}{N}k}O[k] \tag{5.27}$$

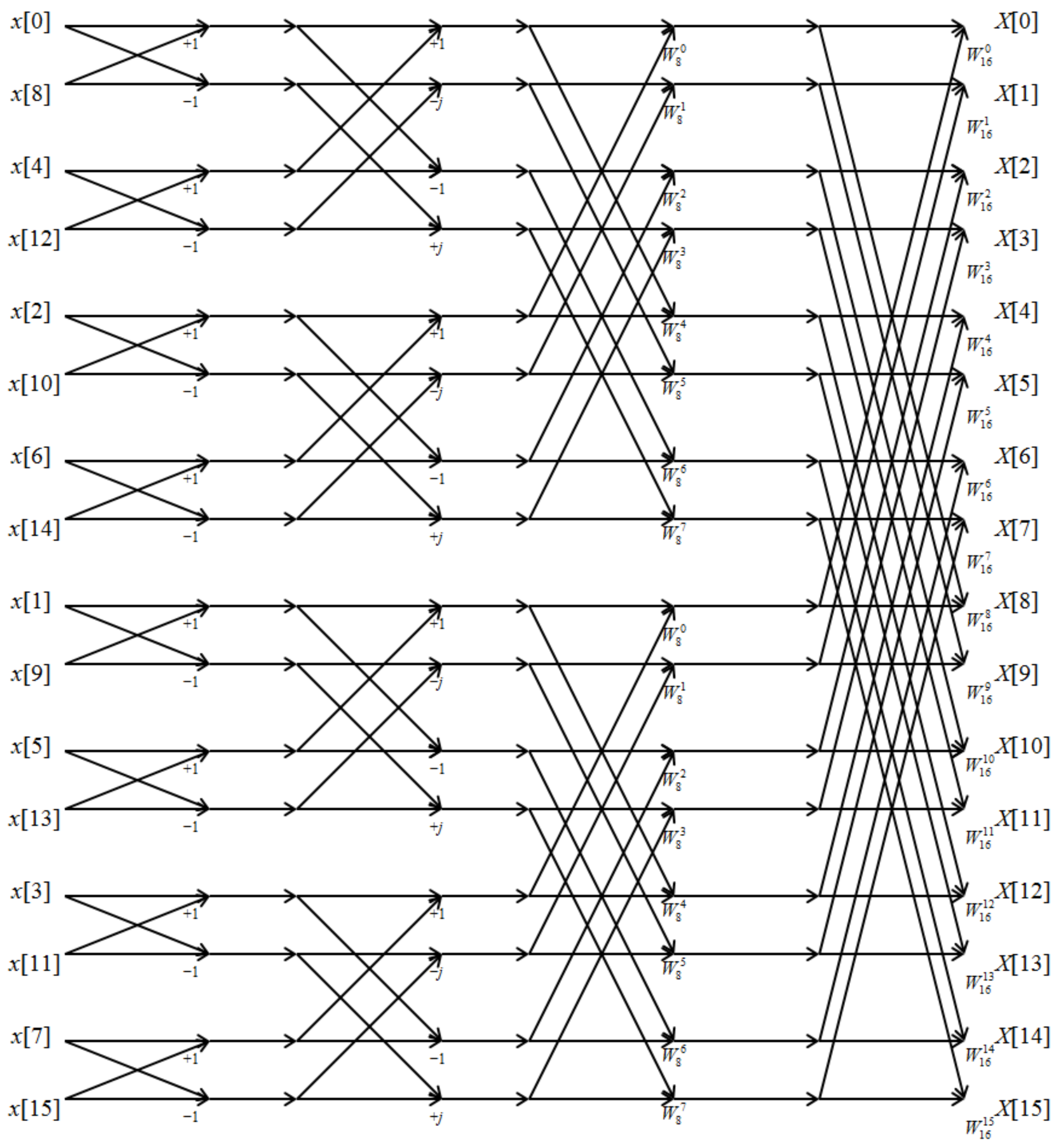


FIGURE 5.13: The butterfly diagram of a 16-point FFT.

We notice that due to the periodicity of the DFT's ($E[k]$ and $O[k]$) and that of the twiddle factors[2], we can write out

$$X[k+N/2] = \underbrace{E[k+N/2]}_{=E[k]} + e^{-j\frac{2\pi}{N}(k+N/2)}\underbrace{O[k+N/2]}_{=O[k]} \tag{5.28}$$

$$= E[k] - e^{-j\frac{2\pi}{N}k}O[k] \tag{5.29}$$

In this way, the last $N/2$ samples of the DFT $X[k]$ we want to calculate can be expressed in terms of the same $E[k]$ and $O[k]$, both previously calculated, yielding the radix-2 algorithm:

For $0 \leq k \leq N/2-1$,

$$\begin{cases} X[k] & = E[k] + e^{-j\frac{2\pi}{N}k}O[k], \\ X[k+N/2] & = E[k] - e^{-j\frac{2\pi}{N}k}O[k]. \end{cases}$$

Graphically this is the same scheme as the butterfly diagram we represented before.

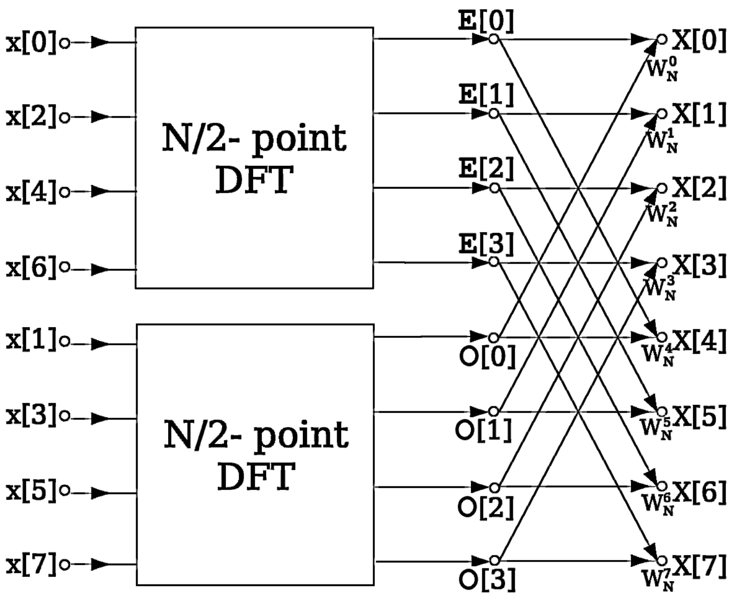


FIGURE 5.14: The butterfly diagram for the radix-2 algorithm.

To show the $O(N\log N)$ computational complexity of the algorithm, let us assume that N is a power of 2, i.e., $N = 2^m$. Now, if we define c_m as the number of complex multiplications when calculating a 2^m-points DFT, then the following recurrence relation holds:

$$c_m = 2c_{m-1} + 2^{m-1} \tag{5.30}$$

[2]This term is coined to the root-of-unity complex multiplicative constants in the butterfly operations of the general Cooley-Tukey FFT algorithm we will see later. The name "*twiddle*" is because the exponential actually performs a rotation.

This is because we need to compute the DFT's of the even and odd samples, each requiring c_{m-1} complex multiplications, plus $N/2 = 2^{m-1}$ complex multiplications of the twiddle factors with the DFT of the odd samples[3]. The base case would be $c_0 = 0$ (just the sample itself, so no complex multiplications).

We are not going to solve this recurrence relation by using the general approach for solving first-order homogeneous recurrence relations with variable coefficients, but rather by the following simple derivation:

$$c_m = 2c_{m-1} + 2^{m-1} \tag{5.31}$$

$$c_m - 2c_{m-1} = 2^{m-1} \tag{5.32}$$

$$\underbrace{\frac{c_m}{2^m}}_{=:A_m} - \underbrace{\frac{2c_{m-1}}{2^m}}_{=A_{m-1}} = \underbrace{\frac{2^{m-1}}{2^m}}_{=1/2} \tag{5.33}$$

$$\sum_{i=1}^{m} A_i - A_{i-1} = \sum_{i=1}^{m} \frac{1}{2} \tag{5.34}$$

$$A_m - A_0 = m \cdot \frac{1}{2} \tag{5.35}$$

$$A_m = A_0 + \frac{m}{2} \tag{5.36}$$

$$\frac{c_m}{2^m} = \frac{c_0}{2^0} + \frac{m}{2} = \frac{m}{2} \tag{5.37}$$

$$c_m = 2^{m-1} \cdot m \tag{5.38}$$

In other words, $c_m = 1/2 \cdot N\log_2 N$, which attains the computational complexity we wanted to show. The table below illustrates how efficient this algorithm can be compared with the traditional DFT computation by definition, in terms of the number of complex multiplications and with increasing N:

N	16	64	256	1024	4096	2^{20}
DFT by definition	256	4096	65536	1.0e6	1.7e7	1.1e12
radix-2 FFT	32	192	1024	5120	24576	1.0e7

The split-radix algorithm

Following the same approach of breaking the DFT into smaller pieces, the computational complexity of the radix-2 algorithm can even be lowered to $1/3 \cdot N\log_2 N$. The idea is based on the application of a radix-2 index map to the even-indexed samples and a radix-4 map to the odd-indexed samples. In other words, the algorithm performs three smaller sized DFT's: one of size $N/2$ (the even samples, indexes 0 or 2 modulo 4), and two of size $N/4$ (the "even subsamples" and the "odd subsamples" of the odd samples, which correspond to indexes 1 and 3 modulo 4, respectively).

We are not going to write down all the calculations, but just show here the final expressions the algorithm provides:

[3]We count special cases like when the twiddle factor equals 1 as multiplication by a complex number as well.

For $0 \le k \le N/4 - 1$,

$$\begin{cases} X[k] & = E[k] + (e^{-j\frac{2\pi}{N}k}O_e[k] + e^{-j\frac{2\pi}{N}3k}O_o[k]), \\ X[k+N/2] & = E[k] - (e^{-j\frac{2\pi}{N}k}O_e[k] + e^{-j\frac{2\pi}{N}3k}O_o[k]), \\ X[k+N/4] & = E[k+N/4] - j(e^{-j\frac{2\pi}{N}k}O_e[k] - e^{-j\frac{2\pi}{N}3k}O_o[k]), \\ X[k+3N/4] & = E[k+N/4] + j(e^{-j\frac{2\pi}{N}k}O_e[k] - e^{-j\frac{2\pi}{N}3k}O_o[k]). \end{cases}$$

We observe that the algorithm needs to take as many complex multiplications as it would need to compute one half-sized and two quarter-sized, plus the multiplications of the type "twiddle factor times $O_e[k]$ or $O_o[k]$", $N/2$ in total[4]. If now we define c_m as the number of complex multiplications when calculating a 2^m-points DFT, the previous argument leads to the following recurrence relations, with initial conditions $c_0 = c_1 = 0$:

$$c_m = c_{m-1} + 2c_{m-2} + 2^{m-1} \tag{5.39}$$

The solution given by Wolfram Alpha is

$$c_m = \frac{1}{9}\left(2^m(3m-2) + 2\cdot(-1)^m\right) \tag{5.40}$$

The leading term $1/9 \cdot 2^m \cdot 3m = 1/3 \cdot N\log_2 N$ reveals the improved computational complexity we wanted to verify.

The general Cooley-Tukey FFT algorithm

Throughout this section we have analyzed the different FFT algorithms assuming that N is a power of two. What if it is not? In general, if we have $N = N_1N_2$ (N_1 and N_2 not necessarily primes), the general Cooley-Tukey FFT algorithm rearranges the samples into a N_1 by N_2 matrix by redefining $k = N_2k_1 + k_2$ and $n = N_1n_2 + n_1$, with k_a, n_a run from 0 to $N_a - 1$, $a = 1, 2$. Hence the DFT can be written as

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}nk} \tag{5.41}$$

$$X[N_2k_1 + k_2] = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x[N_1n_2 + n_1]e^{-j\frac{2\pi}{N_1N_2}(N_1n_2+n_1)(N_2k_1+k_2)} \tag{5.42}$$

$$= \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x[N_1n_2 + n_1]\underbrace{e^{-j2\pi n_2k_1}}_{1}e^{-j\frac{2\pi}{N_2}n_2k_2}e^{-j\frac{2\pi}{N_1}n_1k_1}e^{-j\frac{2\pi}{N_1N_2}n_1k_2} \tag{5.43}$$

$$= \sum_{n_1=0}^{N_1-1}\left[e^{-j\frac{2\pi}{N}n_1k_2}\right]\left(\sum_{n_2=0}^{N_2-1} x[N_1n_2 + n_1]e^{-j\frac{2\pi}{N_2}n_2k_2}\right)e^{-j\frac{2\pi}{N_1}n_1k_1} \tag{5.44}$$

The formula itself tells us what Cooley-Tukey algorithm does. First it performs N_1 DFTs of size N_2 using a row of the N_1 by N_2 matrix, that is the expression inside the big parenthesis. Then each of the DFTs is multiplied by the corresponding twiddle factor, delimited with brackets. Finally it performs N_2 DFTs of size N_1. If either N_1 or

[4]We regard the multiplication by j as simply sign changes on real and imaginary parts.

N_2 can be written as a product of two smaller factors, then the algorithm is run again recursively.

Typically, either N_1 or N_2 is a small factor, called the radix. If N_1 is the radix, it is called a decimation in time (DIT) algorithm, whereas if N_2 is the radix, it is decimation in frequency (DIF). Therefore the radix-2 we presented before is a DIT algorithm.

Analysis of the computational complexity

The analysis of computational complexity for a general case is complex, due to the amount of possibilities the number N can be factored. Nevertheless it is well established that the algorithm still runs in $O(N \log N)$ time.

Nevertheless, we implemented and made a comparison between three of the commonly used schemes: the radix-2, the radix-4, and the split-radix. For each of them, we derived the recurrent relations for the number of real additions and multiplications needed, as well as the number of clock cycles (see Sec. 5.2.2) it has to be taken, according to our implementation. Then using online recurrence solvers we obtained the exact closed formulas (#ops are the total number of real operations, i.e., all the multiplications and additions):

$$\#\text{mults}_{radix2} = 2N\log_2 N - 7N + 12 \tag{5.45}$$
$$\#\text{mults}_{radix4} = \frac{3}{2}N\log_2 N - 5N + 8 \tag{5.46}$$
$$\#\text{mults}_{sradix} = \frac{4}{3}N\log_2 N - \frac{38}{9}N + 6 + \frac{2}{9}(-1)^{\log_2 N} \tag{5.47}$$
$$\#\text{adds}_{radix2} = 3N\log_2 N - 3N + 4 \tag{5.48}$$
$$\#\text{adds}_{radix4} = \frac{11}{4}N\log_2 N - \frac{13}{6}N + \frac{8}{3} \tag{5.49}$$
$$\#\text{adds}_{sradix} = \frac{8}{3}N\log_2 N - \frac{16}{9}N + 2 - \frac{2}{9}(-1)^{\log_2 N} \tag{5.50}$$
$$\#\text{ops}_{radix2} = 5N\log_2 N - 10N + 16 \tag{5.51}$$
$$\#\text{ops}_{radix4} = \frac{17}{4}N\log_2 N - \frac{43}{6}N + \frac{32}{3} \tag{5.52}$$
$$\#\text{ops}_{sradix} = 4N\log_2 N - 6N + 8 \tag{5.53}$$
$$\#\text{clks}_{radix2} = 7\log_2 N - 12 \tag{5.54}$$
$$\#\text{clks}_{radix4} = 4\log_2 N - 6 \tag{5.55}$$
$$\#\text{clks}_{sradix} = 4\log_2 N \tag{5.56}$$
$$\tag{5.57}$$

As shown in Fig. 5.15 and 5.16, the split-radix approach is always better than the other two, and that radix-4 is more efficient than radix-2. Note that the exact formulas for the number of real multiplications are totally consistent with the rough estimation of the number of complex multiplications we derived in Eq. 5.38 and 5.40, taking into account that we use 4 real multiplications for each complex one.

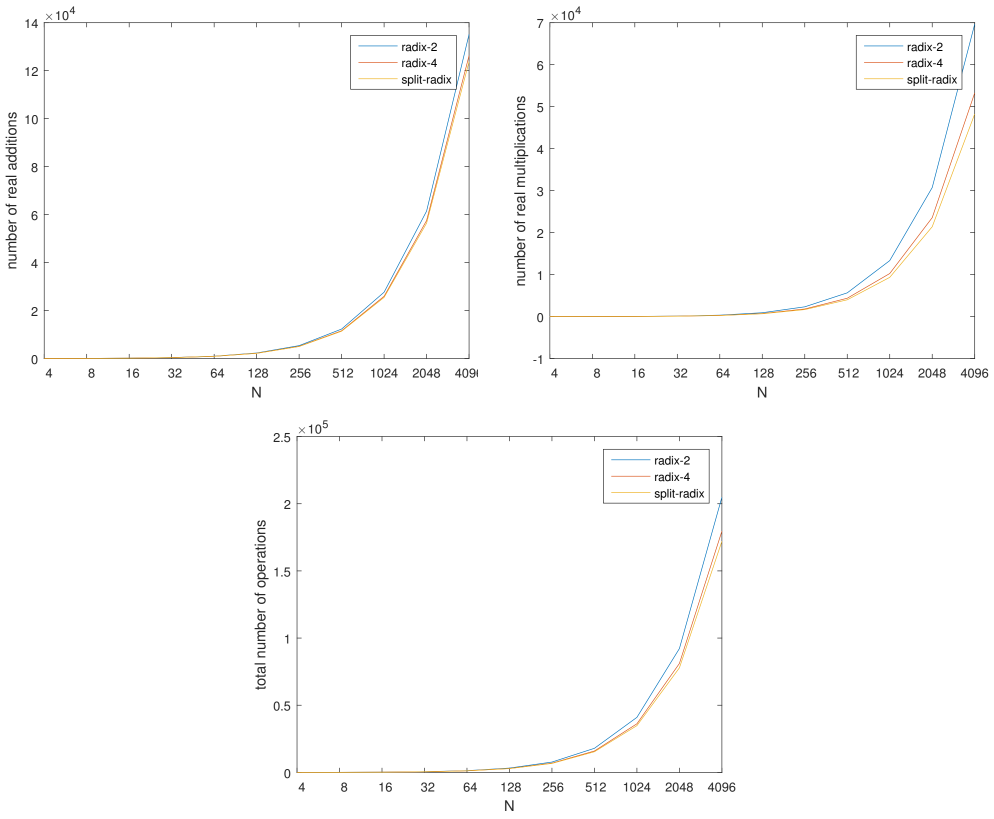


FIGURE 5.15: The number of operations scales with $N \log_2 N$.

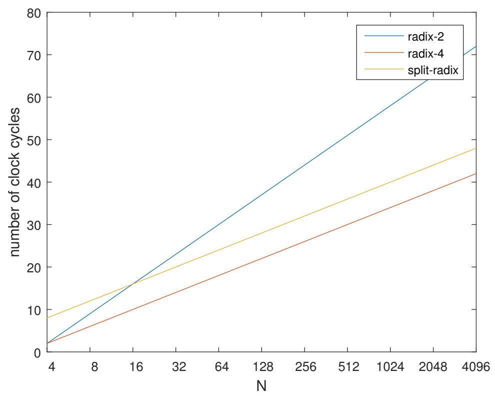


FIGURE 5.16: The number of clock cycles needed scales with $\log_2 N$.

5.2.2 Matlab vs. VHDL

The tools for DSP implementation and simulation used in the project are Matlab and ModelSim. The latter simulates systems behavior described by VHDL.

VHDL, which stands for VHSIC (very high speed integrated circuits) hardware description language, is a hardware description language used in electronic design to describe digital and mixed-signal systems such as FPGA (field-programmable gate arrays), ASIC (application-specific integrated circuit) and CPLD (complex programmable logic device). It was the first hardware description language standardized by the IEEE[5]. It allows circuit synthesis as well as circuit simulation. The former is the translation of a source code into a hardware structure that implements the intended functionality, while the latter is a testing procedure to ensure that such functionality is indeed achieved by the synthesized circuit. Therefore the key advantage of VHDL, when used for systems design, is that it allows the behavior of the required system to be described (modeled) and verified (simulated) before synthesis tools translate the design into real hardware (gates and wires).

VHDL is inherently concurrent, meaning that the commands are not executed step by step, like procedural computing (sequential) languages such as C, C++, Fortran or even Matlab. This fact has important consequences in our implementation, one of them is the introduction of latencies. Consider the following Matlab code that describes a function that computes a complex multiplication using three real multiplications[6]:

```matlab
function z = complex_mult(x,y)
    % initialization
    a = real(x);
    b = imag(x);
    c = real(y);
    d = imag(y);
    % define intermediate variables
    ac = a*c;
    bd = b*d;
    mu = (a+b)*(c+d);
    % calculate result
    f = ac - bd;
    g = mu - ac - bd;
    z = f + 1j*g;
end
```

Knowing the basics of VHDL's syntax, one would write the following VHDL code which might yield the same output if VHDL was sequential. Here is a brief explanation of the code. Lines 1-3 include all the necessary libraries and packages for the implementation. Next comes the entity (lines 5-14), which describes the inputs and the outputs of the program, along with their corresponding data type[7]. Followed by the most important part, the architecture, which contains all the instructions to be performed. The

[5]The Institute of Electrical and Electronics Engineers is the world's largest association of technical professionals of electrical and electronic engineering, telecommunications, computer engineering and allied disciplines.

[6]This is at the cost of calculating more additions, but with large complex numbers it is always preferable to calculate less real multiplications as its complexity is quadratic with respect to the size of the multiplicands.

[7]Unlike Matlab, VHDL does not have a predefined complex number type. Usually we work with other data types for inputs and outputs, especially one type called standard logic vectors (`std_logic_vector` written in VHDL), but for the sake of understanding better the mechanism of the code we prefer to use real numbers here.

architecture starts with some signal declarations. The symbol <= denotes assignment for signal, in contrast with the symbol := for variable assignment (lines 29-31).

```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3  use ieee.numeric_std.all;
 4
 5  entity complex_mult is
 6      port(
 7          clk    : in  std_logic;
 8          x_real : in  real;
 9          x_imag : in  real;
10          y_real : in  real;
11          y_imag : in  real;
12          z_real : out real;
13          z_imag : out real);
14  end entity;
15
16  architecture arch of complex_mult is
17      signal a, b, c, d, f, g : real;
18  begin
19      -- initialization
20      a <= x_real;
21      b <= x_imag;
22      c <= y_real;
23      d <= y_imag;
24      process(clk)
25          variable ac, bd, mu : real;
26      begin
27          if rising_edge(clk) then
28              -- define intermediate variables
29              ac := a*c;
30              bd := b*d;
31              mu := (a+b)*(c+d);
32              -- calculate result
33              f <= ac - bd;
34              g <= mu - ac - bd;
35          end if;
36      end process;
37      z_real <= f;
38      z_imag <= g;
39  end architecture;
```

Since everything we do will be synthesized onto hardware (e.g. FPGA), we must take into account the time dimension. In other words, the hardware needs some time to perform operations, whatever the kind is (addition, multiplication, or even a bit shift)[8]. Therefore we should give some time to the hardware for carrying out these operations. Our approach is to let the process be clocked (line 24), so all the additions and multiplications are done within a clock cycle, from any rising edge of the clock signal (line 27) depicted in Fig. 5.17 to the end of the corresponding clock's period, which is 10 nanoseconds in this case.

Nevertheless, we notice that because of the concurrent nature of VHDL, we do not obtain the corresponding outputs 37-38 at the same time the inputs are available, but rather have to wait until the clocked process 24-36 is completed. In this case, we would

[8]At the end every operation is decomposed by bitwise operations at the hardware level.

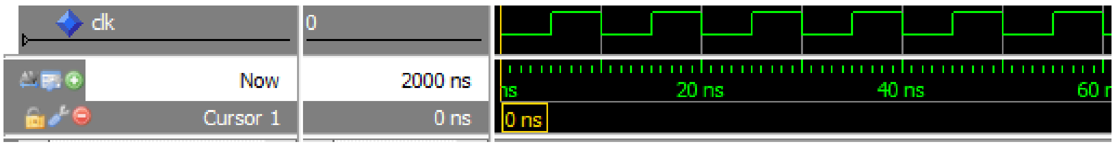


FIGURE 5.17: The clock signal for our clocked process, as simulated using ModelSim.

observe the corresponding outputs one clock cycle after the inputs have been introduced (Fig. 5.18), that is the concept of latency (or delay) we have to deal with constantly when writing our codes. We do not have this kind of problems with Matlab because it just carry out the instructions step by step as fast as it can, without having to worry about hardware limitations.

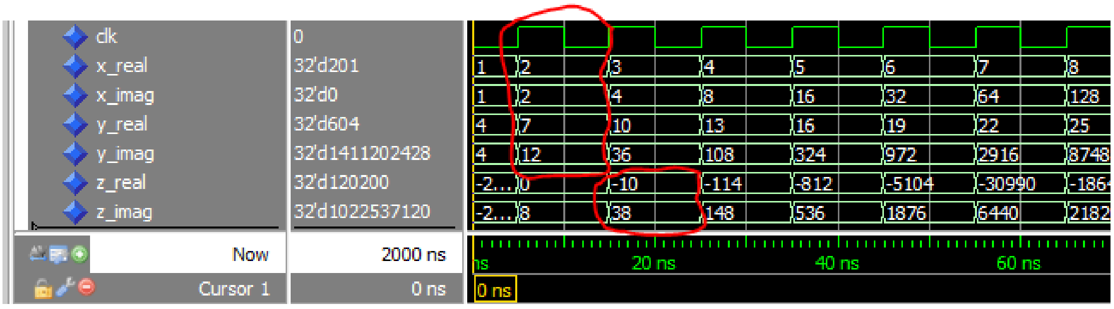


FIGURE 5.18: As seen in ModelSim, the outputs (e.g. -10+38i) are not aligned with their corresponding inputs (2+2i and 7+12i) because of the latency.

We have to take into account the latencies of the subprocesses whenever we have a feedback loop or we require several data streams to meet at the same time. As an example, the following implementation of the recursive split-radix FFT algorithm using Matlab would not be valid if translated directly into VHDL, because the $N/2$-sized FFT takes longer time to be completed than the two $N/4$-sized ones. Therefore, in VHDL, a delay has to be applied to the output of the two smaller FFTs so that they wait until adding to the correct output of the bigger FFT.

```
function [xx_real,xx_imag] = myfft_splitradix(N,isifft,x_real,x_imag)
  if N == 1
    xx_real = x_real;
    xx_imag = x_imag;
  elseif N == 2
    xx_real(1,:) = x_real(1,:)+x_real(2,:);
    xx_real(2,:) = x_real(1,:)-x_real(2,:);
    xx_imag(1,:) = x_imag(1,:)+x_imag(2,:);
    xx_imag(2,:) = x_imag(1,:)-x_imag(2,:);
  else
  % decimation in time
  a_real = x_real(1:2:N,:);
  a_imag = x_imag(1:2:N,:);
  c_real = x_real(2:4:N,:);
  c_imag = x_imag(2:4:N,:);
  d_real = x_real(4:4:N,:);
  d_imag = x_imag(4:4:N,:);
  % apply ffts recursively
  [aa_real,aa_imag] = myfft_splitradix(N/2,isifft,a_real,a_imag);
  [cc_real,cc_imag] = myfft_splitradix(N/4,isifft,c_real,c_imag);
  [dd_real,dd_imag] = myfft_splitradix(N/4,isifft,d_real,d_imag);
  % apply twiddle factors
  n = repmat((0:N/4-1)',1,size(x_real,2));
  if not(isifft)
    ee_real  = aa_real;
    ee_imag  = aa_imag;
    gg_real  = cc_imag.*sin(2*pi*1*n/N)+cc_real.*cos(2*pi*1*n/N);
    gg_imag  = cc_imag.*cos(2*pi*1*n/N)-cc_real.*sin(2*pi*1*n/N);
    hh_real  = dd_imag.*sin(2*pi*3*n/N)+dd_real.*cos(2*pi*3*n/N);
    hh_imag  = dd_imag.*cos(2*pi*3*n/N)-dd_real.*sin(2*pi*3*n/N);
  else
    ee_real  = aa_real;
    ee_imag  = aa_imag;
    gg_real  = cc_real.*cos(2*pi*1*n/N)-cc_imag.*sin(2*pi*1*n/N);
    gg_imag  = cc_real.*sin(2*pi*1*n/N)+cc_imag.*cos(2*pi*1*n/N);
    hh_real  = dd_real.*cos(2*pi*3*n/N)-dd_imag.*sin(2*pi*3*n/N);
    hh_imag  = dd_real.*sin(2*pi*3*n/N)+dd_imag.*cos(2*pi*3*n/N);
  end
  % addition stage 1
  if not(isifft)
    ii_real = ee_real;
    ii_imag = ee_imag;
    jj_real = [gg_real; gg_imag]+[hh_real;-hh_imag];
    jj_imag = [gg_imag;-gg_real]+[hh_imag; hh_real];
  else
    ii_real = ee_real;
    ii_imag = ee_imag;
    jj_real = [gg_real;-gg_imag]+[hh_real; hh_imag];
    jj_imag = [gg_imag; gg_real]+[hh_imag;-hh_real];
  end
  % addition stage 2
  xx_real = [ii_real;ii_real]+[jj_real;-jj_real];
  xx_imag = [ii_imag;ii_imag]+[jj_imag;-jj_imag];
end
```

After synthesis into circuits, our MIMO DSP implementation using VHDL results in the FPGA layout depicted in Fig. 5.19, where we can see its division into super logic regions (the vertical stripes). Each tiny dot is a configurable logic block (CLB) which can implement several types of logic gates and operations.

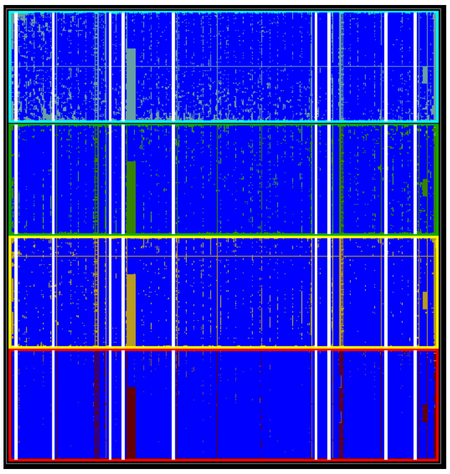


FIGURE 5.19: The FPGA layout.

5.3 Results

The results obtained in [19] are extended with the following observations.

Using a 2×1 equalizer implementation that maps two received polarizations onto one output signal we first characterized our transmitter and receiver performance in a single-mode polarization-division multiplexed back-to-back experiment. While the equalizer and the carrier recovery were running in real-time, the error counting was still carried out offline here. The resulting bit error ratio (BER) is plotted versus the optical-signal-to-noise ratio (OSNR) in Fig. 5.21 for all three transmitter and receiver pairs. In each case, we observe an implementation penalty of about 1.5 dB with respect to theory for differentially decoded PDM-QPSK.

Next we tested the performance of our real-time transmission system by determining the BER after differential decoding over a time frame of 30 minutes for each of the six spatial and polarization modes with a launch power into each core of 3 dBm. The results are shown in Fig. 5.20. We observe that the BERs are drifting between 8×10^{-4} and 7×10^{-3}.

This relatively high error-floor for the BER could be explained by several possible reasons. First, we observed that if the FPGA is running with a high resource utilization, glitches in the outputs of the ADCs appear, which we attribute to a clock drift in the communication between the ADCs and the FPGA. Second, due to the highly parallel implementation the feedback delay of the adaptive equalizer is in the order of 100 DSP clock cycles. This reduces the capability of the equalizer to track temporal variations of the channel.

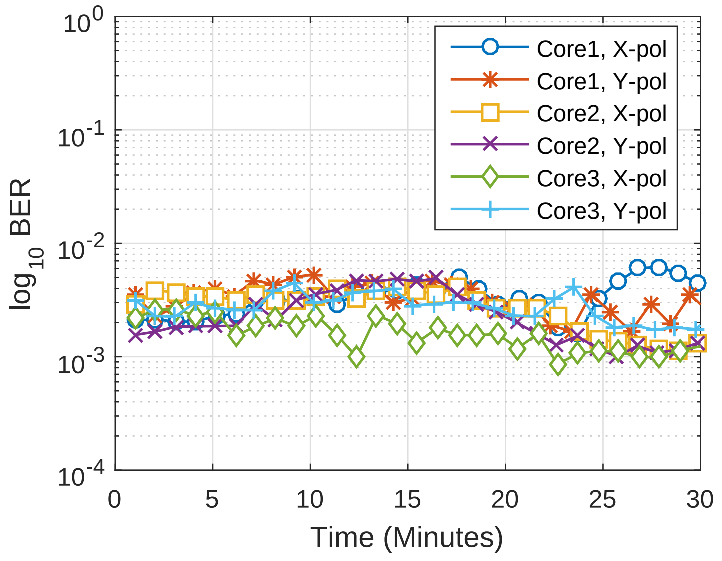


FIGURE 5.20: Real-time BER performance over 30 minutes for all six coupled spatial and polarization modes after 60-km CCF.

These drifts become stronger if the FPGA utilization is high, such as in the case where the full MIMO-DSP is running, and lead to sparse glitches in the QPSK constellations. If we configure the FPGA with a design that utilizes only a relatively small portion of its resources, such as if we use the FPGA board like an oscilloscope saving limited-length waveforms, the glitches disappear.

Finally, we configured the FPGA to demultiplex one out of six coupled spatial and polarization modes after the 60-km CCF using a 6×1 MIMO equalizer, i.e. one of the

6 independent equalizer branches of a full 6×6 MIMO-DSP matrix. In other words, we started convergence of our real-time equalizer based on a single transmit signal and gradually increased the number of transmit signals until all 6 spatial and polarization modes were switched on. A typical received constellation is shown in Fig. 5.22, demonstrating that the transmitted QPSK-signals can be recovered.

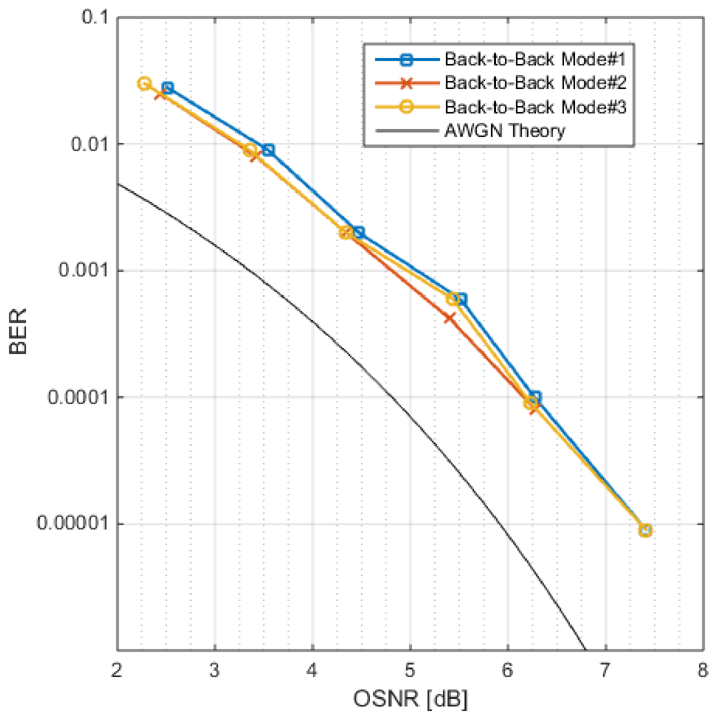


FIGURE 5.21: BER(OSNR) for the three transmitter and receiver pairs.

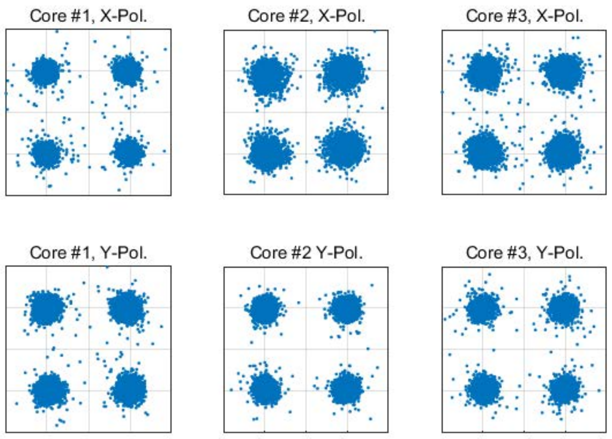


FIGURE 5.22: Received constellations after 6 × 6 MIMO channel (60 km CCF) and real-time MIMO-DSP.

Chapter 6

Conclusions

With this project the initial proposed goals are achieved: gaining a better understanding on how real world fiber-optic communication works, and testing a real experiment aimed to demonstrate the capabilities of MIMO DSP algorithms to reliably retrieve information. The resulting BERs were reasonable low to let us confirm that, using an adaptive MIMO equalizer, the first real-time transmission experiment do compensate mode coupling between spatial modes in an optical fiber.

However, we also encountered some results that were out of our expectation, like the glitches observed in the outputs of the analog-to-digital converters. If our hyothesis were correct, that it was due to clock drift in the communication between the ADCs and the FPGAs, further improvements will have to be investigated.

Therefore, we hope that in the future we could keep bettering this experiment, as a way to get closer to the transmission capacity limit.

Bibliography

[1] John G. Proakis, *Digital Communications*, Fourth Edition , McGraw-Hill, Aug. 2000

[2] John R. Barry, Edward A. Lee, David G. Messerschmitt, *Digital Communication*, Third Edition , Springer, Sep. 2003

[3] Alan V. Oppenheim, Ronald W. Schafer, John R. Buck, *Discrete-Time Signal Processing*, Second Edition , Prentice Hall, Jan. 1998

[4] Simon S. Haykin, *Adaptive Filter Theory*, Fourth Edition , Prentice Hall, Jan. 2001

[5] D. J. Richardson, J. M. Fini, and L. E. Nelson, "Space-division multiplexing in optical fibres", *Nature Photonics*, 7, 354–362, Apr. 2013.

[6] P. J. Winzer, "Modulation and multiplexing in optical communication systems", *IEEE-LEOS Newsletter*, Feb. 2009.

[7] R.-J. Essiambre, G. Kramer, P. J. Winzer, G. J. Foschini, and B. Goebel, "Capacity limits of optical fiber networks", *Journal of Lightwave Technology*, vol. 28, no. 4, pp. 662-701, Feb. 2010.

[8] P. J. Winzer and G. J. Foschini, "MIMO capacities and outage probabilities in spatially multiplexed optical transport systems", *Optics Express*, vol. 19, no. 17, Aug. 2011.

[9] C. Antonelli, A. Mecozzi, M. Shtaif, and P. J. Winzer, "Stokes-space analysis of modal dispersion in fibers with multiple mode transmission", *Opt. Express*, 20, 11718-11733, 2012.

[10] J. J. Shynk, "Frequency-domain and multirate adaptive filtering", *IEEE Signal Processing Magazine*, vol. 9, no. 1, pp. 14-37, Jan. 1992.

[11] T. Morioka, Y. Awaji, R. Ryf, P. Winzer, D. Richardson, and F. Poletti, "Enhancing optical communications with brand new fibers", *IEEE Commun. Mag.*, vol. 50, no. 2, pp. s31–s42, Feb. 2012.

[12] R. Ryf, *et al.*, "Coherent 1200-km 6×6 MIMO mode multiplexed transmission over 3-core microstructured fiber", *Proc. 37th Eur. Conf. Opt. Commun.*, 2012, pp. 1–3, paper Th.13.C.1.

[13] K. Imamura, H. Inaba, K. Mukasa, and R. Sugizaki, "Weakly coupled multi core fibers with optimum design to realize selectively excitation of individual supermodes", *Proc. OFC/NFOEC*, 2012, pp. 1–3, paper OM2D.6.

[14] R. Ryf, R.-J. Essiambre, S. Randel, M. A. Mestre, C. Schmidt, and P. Winzer, "Impulse response analysis of coupled-core 3-core fibers", *Proc. Eur. Conf. Exhibit. Opt. Commun.*, 2012, pp. 1–3, paper Mo.1.F.4.

[15] S. Mumtaz, R.-J. Essiambre, and G. P. Agrawal, "Nonlinear propagation in multimode and multicore fibers: generalization of the Manakov equations", *IEEE J. Lightwave Tech.*, 31, 398–406 (2013).

[16] S. Berdague and P. Facq, "Mode-Division Multiplexing in Optical Fibers", *Appl. Opt.*, vol. 21, no. 11, 1982, pp. 1950–55.

[17] S. Randel, R. Ryf, A. Sierra, P. J. Winzer, A. H. Gnauck, C. A. Bolle, R.-J. Essiambre, D. W. Peckham, A. McCurdy, and R. Lingle, "6$\times$56-Gb/s mode-division multiplexed transmission over 33-km few-mode fiber enabled by 6$\times$6 MIMO equalization", *Optics Express*, vol. 19, no. 17, pp. 16697, 2011.

[18] R. Ryf, N. K. Fontaine, M. A. Mestre, S. Randel, X. Palou, C. Bolle, A. H. Gnauck, S. Chandrasekhar, X. Liu, B. Guan, R. Essiambre, P. J. Winzer, S. Leon-Saval, J. Bland-Hawthorn, R. Delbue, P. Pupalaikis, A. Sureka, Y. Sun, L. Grüner-Nielsen, R. V. Jensen, and R. Lingle, "12 $\times$ 12 MIMO Transmission over 130-km Few-Mode Fiber," *Frontiers in Optics 2012/Laser Science XXVIII*, paper FW6C.4, 2012.

[19] S. Randel, S. Corteselli, D. Badini, D. Pilori, S. Caelles, S. Chandrasekhar, J. Gripp, H. Chen, N. K. Fontaine, R. Ryf, and P. J. Winzer, "First Real-Time Coherent MIMO-DSP for Six Coupled Mode Transmission", *IEEE Photonics Conference (IPC)*, 2015

[20] http://www.lightwaveonline.com/

[21] http://www.wikipedia.org/