

NetXPTO - LinkPlanner

3 de Janeiro de 2018

Conteúdo

1	Introduction	4
2	Simulator Structure	5
2.1	System	5
2.2	Blocks	5
2.3	Signals	5
3	Development Cycle	6
4	Visualizer	7
5	Case Studies	8
5.1	QPSK Transmitter	8
5.2	BPSK Transmission System	10
5.2.1	Theoretical Analysis	10
5.2.2	Simulation Analysis	11
5.2.3	Comparative Analysis	15
5.3	M-QAM Transmission System	18
5.3.1	Theoretical Analysis	18
5.3.2	Simulation Analysis	20
5.3.3	Comparative Analysis	21
5.4	Kramers-Kronig Transceiver with Stokes PolDemux	25
5.4.1	Theoretical Analysis	25
5.4.2	Simulation Analysis	31
5.4.3	Experimental Analysis	35
5.4.4	Comparative Analysis	36
5.4.5	Know Problems	36
5.5	Radio Over Fiber Transmission System	42
5.5.1	Theoretical Analysis	43
5.5.2	Experimental	45
5.6	Optical Detection	46

Conteúdo	2
5.6.1 Theoretical Analysis	46
5.6.2 Simulation Analysis	56
5.6.3 Experimental Analysis	62
5.6.4 Comparative analysis	66
5.6.5 Known problems	66
5.7 Continuous Variable QKD Transmission System	67
5.7.1 Theoretical Analysis	67
5.7.2 Simulation Analysis	69
5.7.3 Experimental Analysis	71
5.7.4 Comparative Analysis	75
5.8 BB84 with Discrete Variables	77
5.8.1 Theoretical Description	77
5.8.2 Simulation Analysis	85
5.8.3 Open Issues	90
5.9 Quantum Oblivious Key Distribution with Discrete Variables	91
5.9.1 Theoretical Description	91
5.9.2 Simulation Analysis	102
5.9.3 Experimental Setup	106
5.9.4 Comparative Analysis	108
6 Library	110
6.1 Add	111
6.2 Bit Error Rate	112
6.3 Binary Source	115
6.4 Bit Decider	119
6.5 Clock	120
6.6 Clock_20171219	122
6.7 Coupler 2 by 2	125
6.8 Decoder	126
6.9 Discrete To Continuous Time	128
6.10 Fork	130
6.11 MQAM Receiver	131
6.12 IQ Modulator	135
6.13 Local Oscillator	137
6.14 MQAM Mapper	139
6.15 MQAM Transmitter	142
6.16 Alice QKD	146
6.17 Polarizer	148
6.18 Bob QKD	149
6.19 Eve QKD	150
6.20 Rotator Linear Polarizer	151
6.21 Optical Switch	153
6.22 Optical Hybrid	154

<i>Conteúdo</i>	3
6.23 Photodiode pair	156
6.24 Pulse Shaper	159
6.25 Sampler	161
6.26 Sink	163
7 Mathlab Tools	164
7.1 Generation of AWG Compatible Signals	165
7.1.1 sgnToWfm_20171121	165
7.1.2 Loading a signal to the Tektronix AWG70002A	167
8 Algorithms	169
8.1 Overlap-Save Method	170
8.1.1 Frequency Response of Filter	171
8.2 FFT	175

Capítulo 1

Introduction

LinkPlanner is devoted to the simulation of point-to-point links.

Capítulo 2

Simulator Structure

LinkPlanner is a signals open-source simulator.

The major entity is the system.

A system comprises a set of blocks.

The blocks interact with each other through signals.

2.1 System

2.2 Blocks

2.3 Signals

List of available signals:

- Signal

Capítulo 3

Development Cycle

The NetXPTO-LinkPlanner has been developed by several people using git as a version control system. The NetXPTO-LinkPlanner repository is located in the GitHub site <http://github.com/netxpto/linkplanner>. The more updated functional version of the software is in the branch master. Master should be considered a functional beta version of the software. Periodically new releases are delivered from the master branch under the branch name Release<Year><Month><Day>. The integration of the work of all people is performed by Armando Nolasco Pinto in the branch Develop. Each developer has his/her own branch with his/her name.

Capítulo 4

Visualizer

visualizer

Capítulo 5

Case Studies

5.1 QPSK Transmitter

2017-08-25, Review, Armando Nolasco Pinto

This system simulates a QPSK transmitter. A schematic representation of this system is shown in figure 5.1.

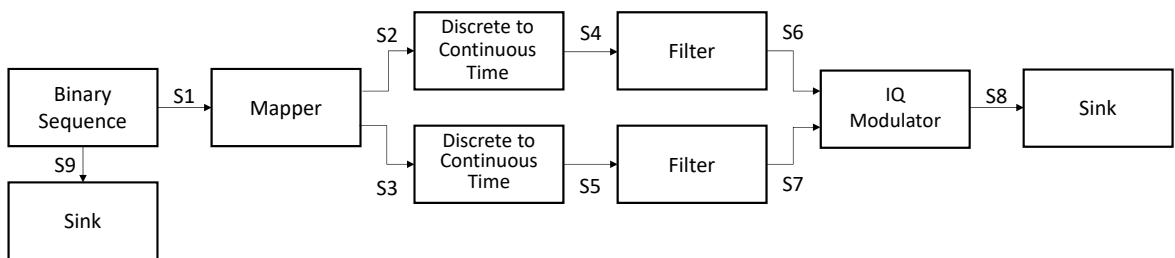


Figura 5.1: QPSK transmitter block diagram.

System Input Parameters

Parameter: *sourceMode*

Description: Specifies the operation mode of the binary source.

Accepted Values: PseudoRandom, Random, DeterministicAppendZeros, DeterministicCyclic.

Parameter: *patternLength*

Description: Specifies the pattern length used by the source in the PseudoRandom mode.

Accepted Values: Integer between 1 and 32.

Parameter: *bitStream*

Description: Specifies the bit stream generated by the source in the DeterministicCyclic and DeterministicAppendZeros mode.

Accepted Values: "XXX..", where X is 0 or 1.

Parameter: *bitPeriod*

Description: Specifies the bit period, i.e. the inverse of the bit-rate.

Accepted Values: Any positive real value.

Parameter: *iqAmplitudes*

Description: Specifies the IQ amplitudes.

Accepted Values: Any four pair of real values, for instance { { 1,1 },{ -1,1 },{ -1,-1 },{ 1,-1 } }, the first value correspond to the "00", the second to the "01", the third to the "10" and the forth to the "11".

Parameter: *numberOfBits*

Description: Specifies the number of bits generated by the binary source.

Accepted Values: Any positive integer value.

Parameter: *numberOfSamplesPerSymbol*

Description: Specifies the number of samples per symbol.

Accepted Values: Any positive integer value.

Parameter: *rollOffFactor*

Description: Specifies the roll off factor in the raised-cosine filter.

Accepted Values: A real value between 0 and 1.

Parameter: *impulseResponseTimeLength*

Description: Specifies the impulse response window time width in symbol periods.

Accepted Values: Any positive integer value.

>>> Romil

5.2 BPSK Transmission System

Student Name	:	Daniel Pereira (2017/09/01 - 2017/11/16)
Goal	:	Estimate the BER in a Binary Phase Shift Keying optical transmission system with additive white Gaussian noise. Comparison with theoretical results.
Directory	:	sdf/bpsk_system

Binary Phase Shift Keying (BPSK) is the simplest form of Phase Shift Keying (PSK), in which binary information is encoded into a two state constellation with the states being separated by a phase shift of π (see Figure 5.2).

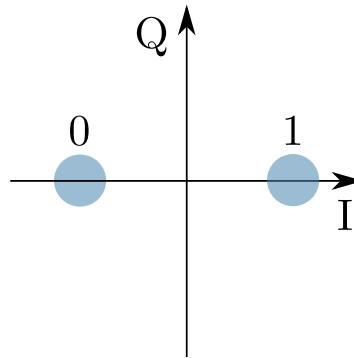


Figura 5.2: BPSK symbol constellation.

White noise is a random signal with equal intensity at all frequencies, having a constant power spectral density. White noise is said to be Gaussian (WGN) if its samples follow a normal distribution with zero mean and a certain variance σ^2 . For WGN its spectral density equals its variance. For the purpose of this work, additive WGN is used to model thermal noise at the receivers.

The purpose of this system is to simulate BPSK transmission in back-to-back configuration with additive WGN at the receiver and to perform an accurate estimation of the BER and validate the estimation using theoretical values.

5.2.1 Theoretical Analysis

The output of the system with added gaussian noise follows a normal distribution, whose first probabilistic moment can be readily obtained by knowledge of the optical power of the received signal and local oscillator,

$$m_i = 2\sqrt{P_L P_S G_{ele}} \cos(\Delta\theta_i), \quad (5.1)$$

where P_L and P_S are the optical powers, in watts, of the local oscillator and signal, respectively, G_{ele} is the gain of the trans-impedance amplifier in the coherent receiver and

$\Delta\theta_i$ is the phase difference between the local oscillator and the signal, for BPSK this takes the values π and 0, in which case (5.1) can be reduced to,

$$m_i = (-1)^{i+1} 2 \sqrt{P_L P_S} G_{ele}, \quad i = 0, 1. \quad (5.2)$$

The second moment is directly chosen by inputting the spectral density of the noise σ^2 , and thus is known *a priori*.

Both probabilist moments being known, the probability distribution of measurement results is given by a simple normal distribution,

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-m_0)^2}{2\sigma^2}}. \quad (5.3)$$

The BER is calculated in the following manner,

$$BER = \frac{1}{2} \int_0^{+\infty} f(x|\Delta\theta = \pi) dx + \frac{1}{2} \int_{-\infty}^0 f(x|\Delta\theta = 0) dx, \quad (5.4)$$

given the symmetry of the system, this can be simplified to,

$$BER = \int_0^{+\infty} f(x|\Delta\theta = \pi) dx = \frac{1}{2} \operatorname{erfc} \left(\frac{-m_0}{\sqrt{2}\sigma} \right) \quad (5.5)$$

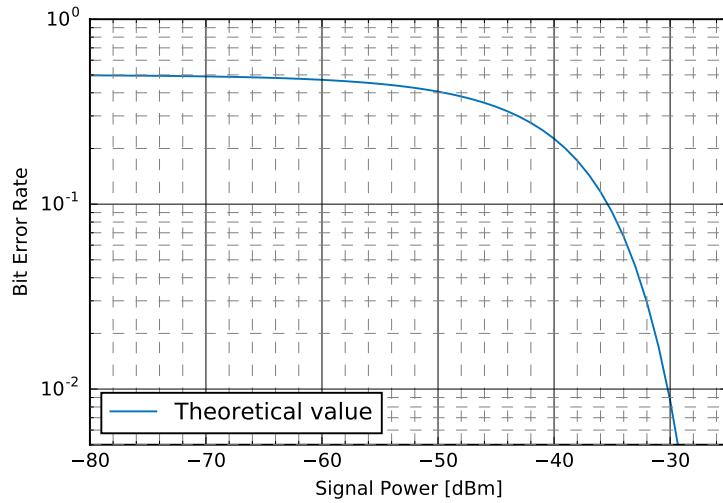


Figura 5.3: Bit Error Rate in function of the signal power in dBm at a constant local oscillator power level of 0 dBm.

5.2.2 Simulation Analysis

A diagram of the system being simulated is presented in the Figure 5.4. A random binary sequence is generated and encoded in an optical signal using BPSK modulation. The decoding of the optical signal is accomplished by an homodyne receiver, which combines the

signal with a local oscillator. The received binary signal is compared with the transmitted binary signal in order to estimate the Bit Error Rate (BER). The simulation is repeated for multiple signal power levels, each corresponding BER is recorded and plotted against the expectation value.

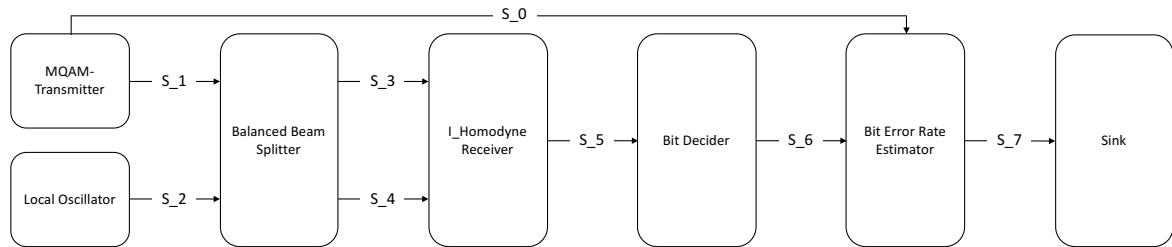


Figura 5.4: Overview of the BPSK system being simulated.

Required files

Header Files		
File	Comments	Status
add.h		✓
balanced_beam_splitter.h		✓
binary_source.h		✓
bit_decider.h		✓
bit_error_rate.h		✓
discrete_to_continuous_time.h		✓
netxpto.h		✓
m_qam_mapper.h		✓
m_qam_transmitter.h		✓
local_oscillator.h		✓
i_homodyne_reciever.h		✓
ideal_amplifier.h		✓
iq_modulator.h		✓
photodiode.h		✓
pulse_shaper.h		✓
sampler.h		✓
sink.h		✓
super_block_interface.h		✓
white_noise.h		✓

Source Files		
File	Comments	Status
add.cpp		✓
balanced_beam_splitter.cpp		✓
binary_source.cpp		✓
bit_decider.cpp		✓
bit_error_rate.cpp		✓
discrete_to_continuous_time.cpp		✓
netxpto.cpp		✓
m_qam_mapper.cpp		✓
m_qam_transmitter.cpp		✓
local_oscillator.cpp		✓
i_homodyne_reciever.cpp		✓
ideal_amplifier.cpp		✓
iq_modulator.cpp		✓
photodiode.cpp		✓
pulse_shaper.cpp		✓
sampler.cpp		✓
sink.cpp		✓
super_block_interface.cpp		✓
white_noise.cpp		✓

System Input Parameters

This system takes into account the following input parameters:

System Input Parameters		
Parameter	Default Value	Comments
numberOfBitsGenerated	40000	
bitPeriod	20×10^{-12}	
samplesPerSymbol	16	
pLength	5	
iqAmplitudesValues	$\{ \{-1, 0\}, \{1, 0\} \}$	
outOpticalPower_dBm	Variable	Value varied from -75 dBm to -25 dBm with intervals of 5 dBm
loOutOpticalPower_dBm	0	
localOscillatorPhase	0	
transferMatrix	$\{ \{ \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}} \} \}$	
responsivity	1	
amplification	10^3	
noiseSpectralDensity	$5 \times 10^{-4} \sqrt{2} \text{ V}^2$	
confidence	0.95	
midReportSize	0	

Inputs

This system takes no inputs.

Outputs

This system outputs the following objects:

Parameter: Signals:

Description: Initial Binary String; (S_0)

Description: Optical Signal with coded Binary String; (S_1)

Description: Local Oscillator Optical Signal; (S_2)

Description: Beam Splitter Outputs; (S_3, S_4)

Description: Homodyne Detector Electrical Output; (S_5)

Description: Decoded Binary String; (S_6)

Description: BER result String; (S_7)

Parameter: Other:

Description: Bit Error Rate report in the form of a .txt file. (BER.txt)

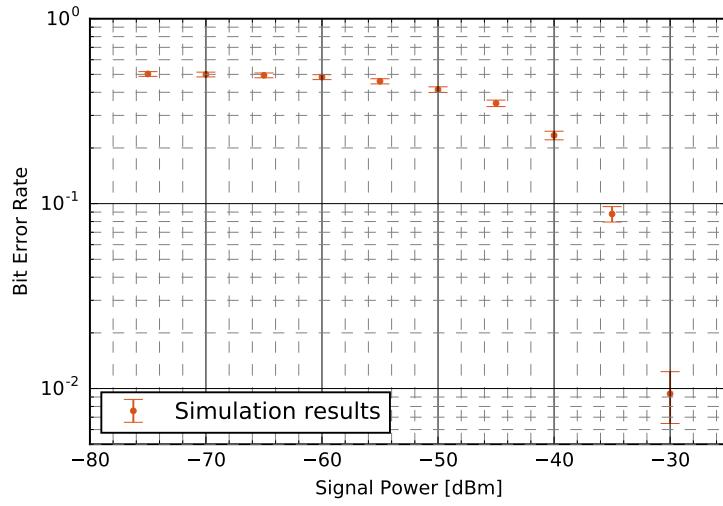


Figura 5.5: Bit Error Rate in function of the signal power in dBm at a constant local oscillator power level of 0 dBm.

5.2.3 Comparative Analysis

The following results show the dependence of the error rate with the signal power assuming a constant Local Oscillator power of 0 dBm, the signal power was evaluated at levels between -70 and -25 dBm, in steps of 5 dBm between each. The simulation results are presented in orange with the computed lower and upper bounds, while the expected value, obtained from (5.5), is presented as a full blue line. A close agreement is observed between the simulation results and the expected value. The noise spectral density was set at $5 \times 10^{-4}\sqrt{2} \text{ V}^2$ [1].

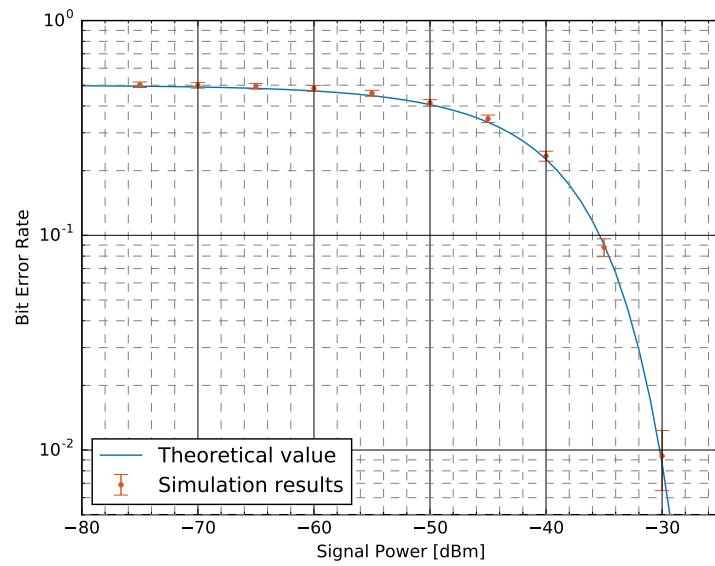


Figura 5.6: Bit Error Rate in function of the signal power in dBm at a constant local oscillator power level of 0 dBm. Theoretical values are presented as a full blue line while the simulated results are presented as a errorbar plot in orange, with the upper and lower bound computed in accordance with the method described in 6.2

Bibliografia

- [1] Thorlabs. *Thorlabs Balance Amplified Photodetectors: PDB4xx Series Operation Manual*, 2014.
- [2] Xinping Huang, Mario Caron, and Dan Hindson. A recursive Gram-Schmidt orthonormalization procedure and its application to communications. In *Wireless Communications, 2001.(SPAWC'01). 2001 IEEE Third Workshop on Signal Processing Advances in*, pages 340–343. IEEE, 2001.
- [3] Tie-Ming Liu, Lie-hui Jiang, Hong-qi He, Ji-zhong Li, and Xian Yu. *Researching on Cryptographic Algorithm Recognition Based on Static Characteristic-Code*, pages 140–147. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [4] Gilles Brassard and Louis Salvail. *Secret-Key Reconciliation by Public Discussion*, pages 410–423. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.
- [5] Min Xu, Run-hua Shi, Zhen-yu Luo, and Zhen-wan Peng. Nearest private query based on quantum oblivious key distribution. *Quantum Information Processing*, 16(12):286, Oct 2017.
- [6] Álvaro J Almeida, Nelson J Muga, Nuno A Silva, João M Prata, Paulo S André, and Armando N Pinto. Continuous control of random polarization rotations for quantum communications. *Journal of Lightwave Technology*, 34(16):3914–3922, 2016.

5.3 M-QAM Transmission System

Student Name	:	Ana Luisa Carvalho
Goal	:	M-QAM system implementation with BER measurement and comparison with theoretical values.
Directory	:	sdf/m_qam_system

The goal of this project is to simulate a Quadrature Amplitude Modulation transmission system with M points in the constellation diagram (M-QAM) and to perform a Bit Error Rate (BER) measurement that can be compared with theoretical values.

M-QAM systems can encode $\log_2 M$ bits per symbol which means they can transmit higher data rates keeping the same bandwidth when compared, for example, to PSK systems. However, because the states are closer together, these systems can be more susceptible to noise.

The Bit Error Rate (BER) is a measurement of how a bit stream is altered by a transmission system due to noise (among other factors). To study this effect we introduced Additive White Gaussian Noise (AWGN) to model thermal noise at the receiver.

For $M = 4$ the M-QAM system reduces to a Quadrature Phase Shift Keying system (QPSK) system that uses four equispaced points in the constellation diagram (see figure 5.7).

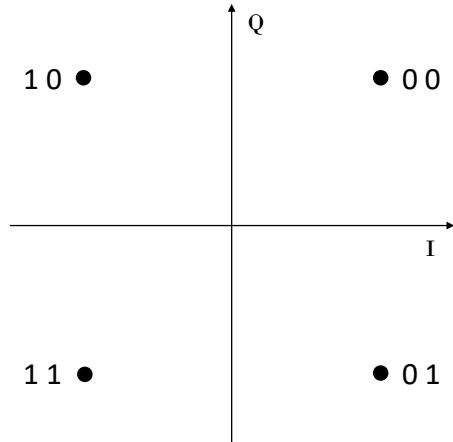


Figura 5.7: 4-QAM constellation points.

5.3.1 Theoretical Analysis

M-QAM is a modulation scheme that takes advantage of two carriers (usually sinusoidal waves) with a phase difference of $\frac{\pi}{2}$. The resultant output consists of a signal with both amplitude and phase variations. The two carriers, referred to as I (In-phase) and Q (Quadrature), can be represented as

$$I = A \cos(\phi) \quad (5.6)$$

$$Q = A \sin(\phi) \quad (5.7)$$

which means that any sinusoidal wave can be decomposed in its I and Q components:

$$A \cos(\omega t + \phi) = A (\cos(\omega t) \cos(\phi) - \sin(\omega t) \sin(\phi)) \quad (5.8)$$

$$= I \cos(\omega t) - Q \sin(\omega t), \quad (5.9)$$

where we have used the expression for the cosine of a sum and the definitions of I and Q.

The probability of symbol error, P_s , in coherent M-PSK demodulation with AWGN is given by

$$P_s = 2 Q \left(\sqrt{2 \log_2 M \left(\frac{E_b}{n_0} \right) \sin^2 \frac{\pi}{M}} \right) \quad (5.10)$$

where E_b is the energy of one bit, n_0 is the noise power and the function Q is defined as

$$Q(x) = \frac{1}{2} \operatorname{erfc} \left(\frac{x}{\sqrt{2}} \right). \quad (5.11)$$

The probability of bit errors, P_b is related to P_s by

$$P_b = \frac{1}{\log_2 M} P_s. \quad (5.12)$$

For QPSK we get, using $M = 4$ in equations 5.11 and 5.12,

$$P_b = \frac{1}{2} \operatorname{erfc} \left(\sqrt{\frac{2 E_b}{n_0}} \right). \quad (5.13)$$

This function is plotted in figure 5.8 for $n_0 = 10^{-6}$.

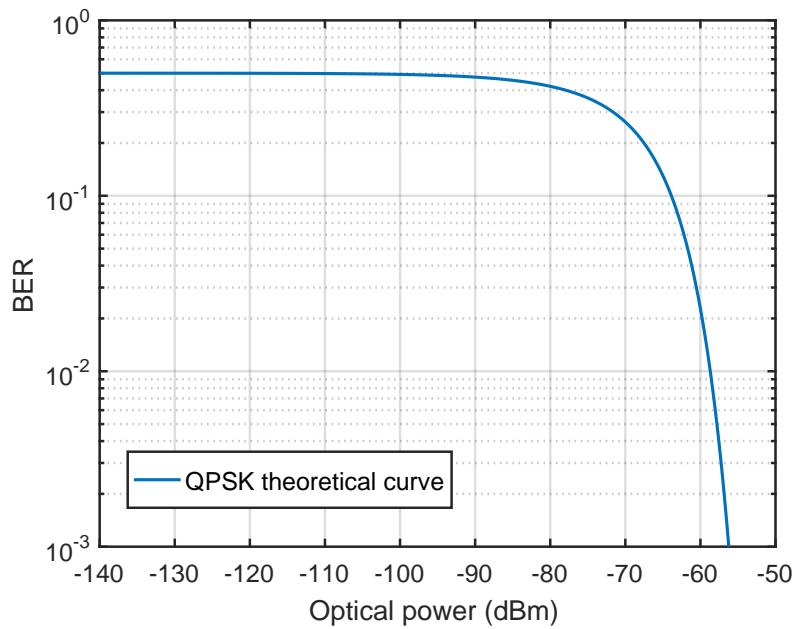


Figura 5.8: QPSK theoretical BER values as a function of the output optical power in dBm.

5.3.2 Simulation Analysis

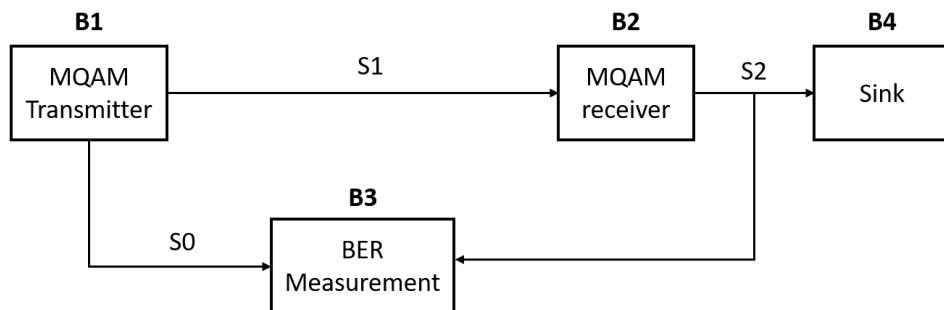


Figura 5.9: Schematic representation of the MQAM system.

The M-QAM transmission system is a complex block of code that simulates the modulation, transmission and demodulation of an optical signal using M-QAM modulation. It is composed of four blocks: a transmitter, a receiver, a sink and a block that performs a Bit Error Rate (BER) measurement. The schematic representation of the system is presented in figure 5.9.

Current state: The system currently being implemented is a QPSK system ($M=4$).

Future work: Extend this block to include other values of M .

Tabela 5.1: Main system files

System blocks	Source file	Header file	Status
Main	m_qam_system_sdf.cpp	—	✓
M-QAM transmitter	m_qam_transmitter.cpp	m_qam_transmitter.h	✓
M-QAM receiver	homodyne_receiver.cpp	homodyne_receiver.h	
Sink	sink.cpp	sink.h	✓
BER estimator	bit_error_rate.cpp	bit_error_rate.h	

Functional description

A complete description of the M-QAM transmitter and M-QAM homodyne receiver blocks can be found in the *Library* chapter of this document as well as a detailed description of the independent blocks that compose these blocks.

The M-QAM transmitter generates one or two optical signals by encoding a binary string using M-QAM modulation. It also outputs a binary signal that is used to perform the BER measurement.

The M-QAM homodyne receiver accepts one input optical signal and outputs a binary signal. It performs the M-QAM demodulation of the input signal by combining the optical signal with a local oscillator.

The demodulated optical signal is compared to the one produced by the transmitter in order to estimate the Bit Error Rate (BER).

The files corresponding to each of the system's blocks are summarized in table 5.1. Along with the library and corresponding source files these allow for the full operation of the M-QAM system.

Required Files

The required header and source files needed to run this system are summarized in table 5.2.

Input Parameters

The system accepts several input parameters that can be defined by the user. These are described in table 5.3.

Simulation results

In this section we show the eye diagrams for the S1 signals for two different values of the output optical power.

5.3.3 Comparative Analysis

In this section we show the simulation results and compared them with the theoretical predictions. Figures 5.11 shows the variation of the BER with the power of the signal, using

Tabela 5.2: Required files

Header file	Source file	Description	Status
add.h	add.cpp	Adds two signals.	✓
binary_source.h	binary_source.cpp	Produces a binary sequence.	✓
bit_error_rate.h	bit_error_rate.cpp	Computes the BER and writes it to a text file.	✓
discrete_to_continuous_time.h	discrete_to_continuous_time.cpp	Converts a signal from discrete in time to continuous in time.	✓
homodyne_receiver.h	m_qam_homodyne_receiver.cpp		
ideal_amplifier.h	ideal_amplifier.cpp	Amplifies the signal.	✓
iq_modulator.h	iq_modulator.cpp	Divides the signal in its quadrature and in phase components	✓
local_oscillator.h	local_oscillator.cpp		
m_qam_mapper.h	m_qam_mapper.cpp	Maps the signal using the defined constellation	✓
m_qam_transmitter.h	m_qam_transmitter.cpp		✓
netxpto.h	netxpto.cpp	General class that contains definition from signals and buffers.	✓
optical_hybrid.h	optical_hybrid.cpp	Implements an optical hybrid.	✓
photodiode_old.h	photodiode_old.cpp	Pair of photodiodes and current subtraction.	✓
pulse_shaper.h	pulse_shaper.cpp	Electrical filter.	✓
sampler_20171119.h	sampler_20171119.cpp	Samples the signal.	✓
sink.h	sink.cpp	Deletes signal.	✓
super_block_interface.h	super_block_interface.cpp		✓
white_noise.h	white_noise.cpp	Generates white gaussian noise.	✓

4000 bits and a pseudorandom binary sequence with pattern length 2^7 . To produce this plots we considered a noise amplitude of 10^{-6} and an amplification of 10^3 .

Tabela 5.3: Input parameters

Parameter	Type	Description
numberOfBitsGenerated	t_integer	Determines the number of bits to be generated by the binary source
samplesPerSymbol	t_integer	Number of samples per symbol
prbsPatternLength	int	Determines the length of the pseudorandom sequence pattern (used only when the binary source is operated in <i>PseudoRandom</i> mode)
bitPeriod	t_real	Temporal interval occupied by one bit
rollOffFactor	t_real	Parameter of the raised cosine filter
signalOutputPower_dBm	t_real	Determines the power of the output optical signal in dBm
numberOfBitsReceived	int	Determines when the simulation should stop. If -1 then it only stops when there is no more bits to be sent
iqAmplitudeValues	vector<t_iqValues>	Determines the constellation used to encode the signal in IQ space
symbolPeriod	double	Given by bitPeriod / samplesPerSymbol
localOscillatorPower_dBm	t_real	Power of the local oscillator
responsivity	t_real	Responsivity of the photodiodes (1 corresponds to having all optical power transformed into electrical current)
amplification	t_real	Amplification provided by the ideal amplifier
noiseAmplitude	t_real	Amplitude of the white noise
samplesToSkip	t_integer	Number of samples to be skipped by the <i>sampler</i> block
confidence	t_real	Determines the confidence limits for the BER estimation
midReportSize	t_integer	
bufferLength	t_integer	Corresponds to the number of samples that can be processed in each run of the system

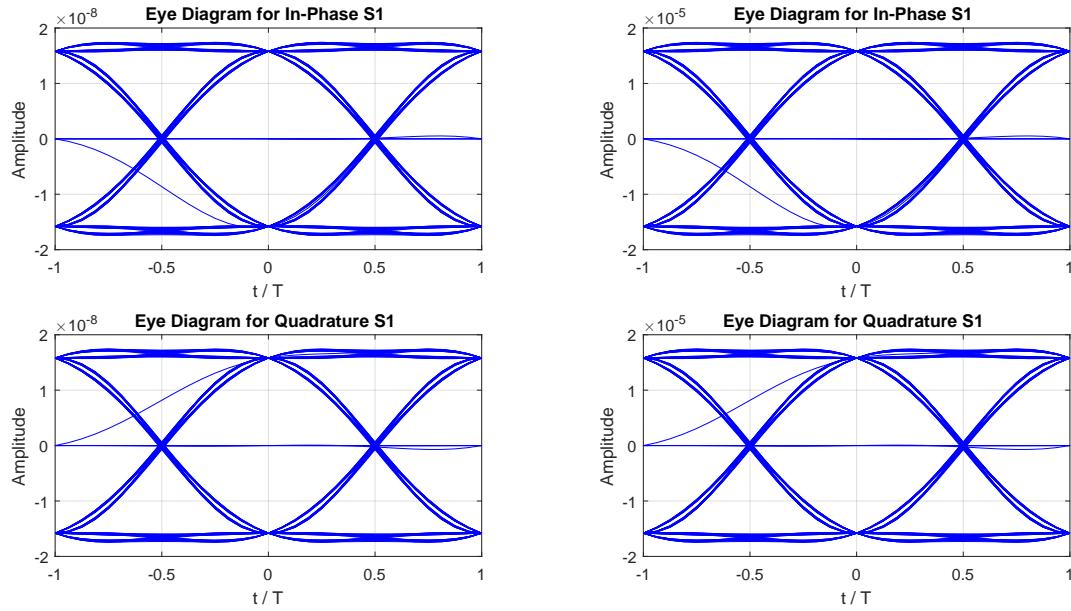


Figura 5.10: Eye diagrams for the S1 bandpass signal with an output optical power of -120dBm (left) and -60dBm (right). Note differente scales on y axis.

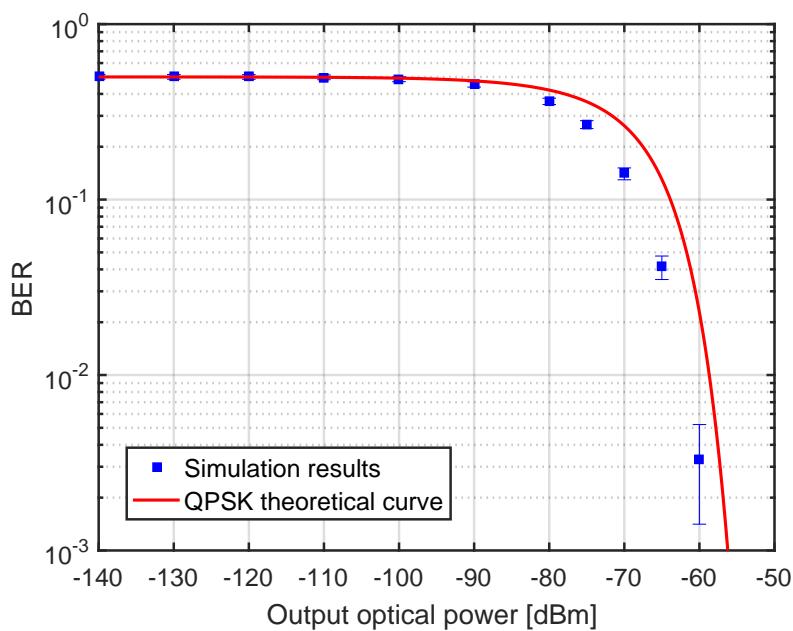


Figura 5.11: Simulation result for a random binary sequence with 4000 bits, a noise amplitude of 10^{-6} and an amplification of 10^3 .

5.4 Kramers-Kronig Transceiver with Stokes PolDemux

Student Name	:	Romil Patel
Starting Date	:	August 16, 2017
Goal	:	Develop a simplified structure (low cost) for a coherent transceiver, that can be used in coherent PON, inter-data center connections, or metropolitan networks (optical path lengths < 100 km). We are going to explore a Kramers-Kronig transceiver with Stokes based PolDemux.
Directory	:	LinkPlanner\doc\tex\sdf\simplified_coherent_receiver

Coherent optical transmission schemes are spectrally efficient since they allow the encoding of information in both quadrature of sinusoid signal. However, the cost of coherent receiver becomes a major obstacle in the case of short-reach links applications like PON, inter-data-center communications and metropolitan network. In order to make the transceiver applicable in short-reach links, an architecture has been proposed which combines the advantages of coherent transmission and cost effectiveness of direct detection. The working principle of the proposed transceiver is based on the Kramers-Kronig (KK) relationship. The KK transceiver scheme allows digital compensation of propagation impairment because both amplitude and phase of the electrical field can be retrieved at the receiver.

5.4.1 Theoretical Analysis

The Kramers-Kronig relations are bidirectional mathematical relations, connecting the real and imaginary parts of any complex function that is analytic in the upper half-plane. For instance, a signal $x(t) = x_r(t) + ix_i(t)$ satisfies the Kramers-Kronig relationship if,

$$x_r(t) = -\frac{1}{\pi} p.v. \int_{-\infty}^{\infty} \frac{x_i(t')}{t - t'} dt'$$

$$x_i(t) = \frac{1}{\pi} p.v. \int_{-\infty}^{\infty} \frac{x_r(t')}{t - t'} dt'$$

This relationship imposes that the real and the imaginary parts of the signal are related to each other though Hilbert transform. Therefore, if we have the real part of the signal then the imaginary part can be calculated by its Hilbert transform.

For a signal that satisfies the Kramers-Kronig relationship, the real and imaginary part can be obtained only from the module. The following questions would give a comprehensive overview of Kramers-Kroning relation and the detailed mathematical calculation which depicts how phase can be extracted from the amplitude information.

1. What is Hilbert transform?

If we consider a filter $H(\omega)$, described in Figure 5.12, that has a unity magnitude response for all frequencies and the phase response is $-\pi/2$ for all positive frequencies and $\pi/2$ for negative frequencies. The transfer function of this filter is given by

$$H(\omega) = -isgn(\omega) \quad (5.14)$$

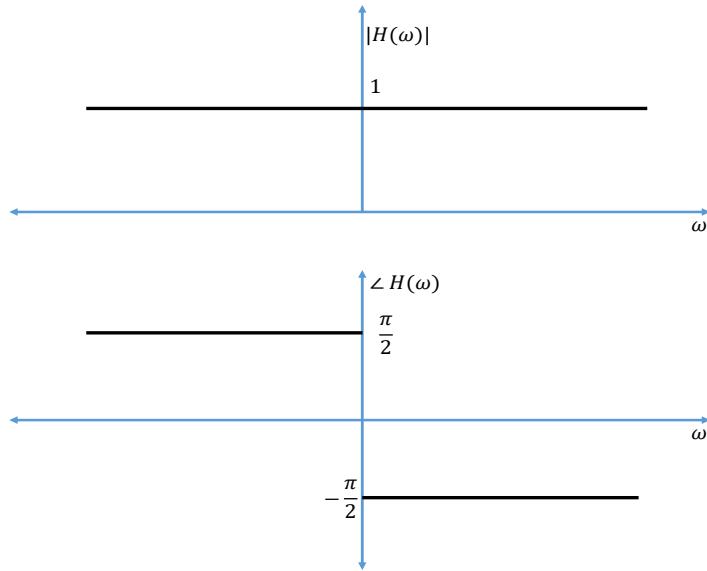


Figura 5.12: Magnitude and phase of Hilbert transform filter

The impulse response of this filter can be given as,

$$\begin{aligned} h(t) &= \mathcal{F}^{-1}[H(i\omega)] \\ &= -i\mathcal{F}^{-1}[sgn(\omega)] \\ &= -i\left(\frac{i}{\pi t}\right) \\ &= \frac{1}{\pi t} \end{aligned} \quad (5.15)$$

When this filter driven by an arbitrary signal $s(t)$, the filter produces the output as,

$$\begin{aligned} \hat{s}(t) &= s(t) * h(t) \\ &= \int_{-\infty}^{\infty} \frac{s(u)}{\pi(t-u)} du \end{aligned} \quad (5.16)$$

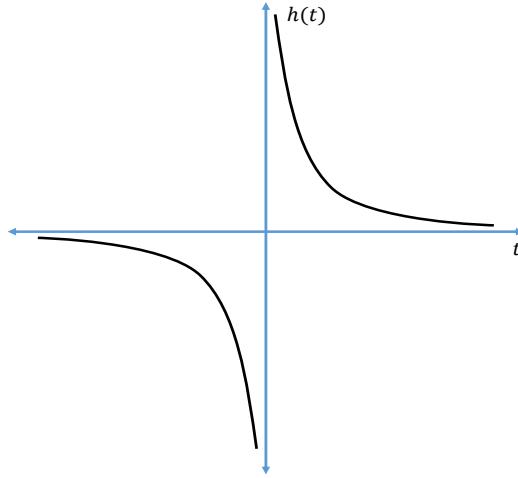


Figura 5.13: Impulse response $h(t)$ of Hilbert transform filter

The function $\hat{s}(t)$ is called the Hilbert transform if $s(t)$. Note that

$$\mathcal{F}[\hat{s}(t)] = H(\omega)S(\omega) = -isgn(\omega)S(\omega) \quad (5.17)$$

In conclusion, if we convolve any time domain signal with $\frac{1}{\pi t}$ then it will give us Hilbert transformed signal in time domain. Similarly, from the convolution property of the Fourier transform, if we multiply $-isgn(\omega)$ with any frequency domain signal $S(\omega)$ then it'll give us Hilbert transformed signal in frequency domain.

2. What is analytical signal?

An analytic signal is a complex-valued signal that has no negative frequency components, and its real and imaginary parts are related to each other by the Hilbert transform.

$$s_a(t) = s(t) + i\hat{s}(t) \quad (5.18)$$

where, $s_a(t)$ is an analytical signal and $\hat{s}(t)$ is the Hilbert transform of the signal $s(t)$. Such analytical signal can be used to generate Single Sideband Signal (SSB) signal.

3. What is a SSB signal and how it can be generated?

By definition, the SSB is the signal which contains either upper sideband or lower sideband and hence it reduces the spectral occupancy by half.

This section will represent the brief idea of generating SSB signal using Hilbert transform method. To understand this, we may express signal $s(t)$ as a summation of the two complex-valued functions.

$$s(t) = \frac{1}{2}[s(t) + i\hat{s}(t)] + \frac{1}{2}[s(t) - i\hat{s}(t)] \quad (5.19)$$

From Equation 5.18,

$$s(t) = s_a(t) + i s_a^*(t) \quad (5.20)$$

Such representation of $s_a(t)$ and $s_a^*(t)$ divide the signal into non-negative frequency component and non-positive frequency component respectively. Considering only non-negative frequency $s_a(t)$ part, we can write it as

$$\frac{1}{2} S_a(f) = \begin{cases} S(f) & \text{for } f > 0 \\ 0 & \text{for } f < 0 \end{cases} \quad (5.21)$$

where $S_a(f)$ and $S(f)$ are the Fourier transform of $t_a(t)$ and $s(t)$ respectively. The frequency translated version of $S_a(f - f_0)$ contains only one side (positive) of $S(f)$ and hence it is called single sideband signal $s_{ssb}(t)$,

$$F^{-1}\{S_a(f - f_0)\} = s_a(t)e^{i2\pi f_0 t} = s_{ssb}(t) + i\hat{s}_{ssb}(t) \quad (5.22)$$

Therefore, from the Euler's formula,

$$\begin{aligned} s_{ssb}(t) &= Re\{s_a(t)e^{i2\pi f_0 t}\} \\ &= Re\{[s(t) + i\hat{s}(t)][\cos(2\pi f_0 t) + i\sin(2\pi f_0 t)]\} \\ &= s(t)\cos(2\pi f_0 t) - \hat{s}(t)\sin(2\pi f_0 t) \end{aligned} \quad (5.23)$$

This Equation 5.23 displays the mathematical modeling of the upper sideband SSB signal. Similarly, we can generate lower sideband SSB signal by,

$$s_{ssb}(t) = s(t)\cos(2\pi f_0 t) + \hat{s}(t)\sin(2\pi f_0 t) \quad (5.24)$$

4. What is minimum phase signal?

A necessary and sufficient condition for a complex signal $A(t)$ to be minimum phase is that the curve described in a complex plane by $A(t)$ when $t \rightarrow -\infty$ to $t \rightarrow \infty$ **does not encircle the origin**. A minimum-phase signal has an useful property that the natural logarithm of the magnitude of the frequency response is related to the phase angle of the frequency response by the Hilbert transform.

For instance, if we consider a complex data-carrying signal whose spectrum is contained between $-B/2$ and $B/2$, and consider a SSB signal of the form,

$$x(t) = A + A_s(t)\exp(-i\pi Bt) \quad (5.25)$$

where A is a constant. Here, $x(t)$ is minimum phase if and only if the winding number of its trajectory into complex plane is zero. The condition $|A| > |A_s(t)|$ is sufficient for guaranteeing minimum phase property [3].

5. How we can use these signals and profit from them?

This section represents the justification that why we need to use these signals into our proposed transceiver system.

Analytical Signal:

If we denote an analytic signal $A_s(t)$ as,

$$A_s(t) = A_{s,r}(t) + iA_{s,i}(t) \quad (5.26)$$

then in the equation 5.26, the real and imaginary parts $A_{s,r}(t)$ and $A_{s,i}(t)$ are related through the Kramers-Kronig relation with each other. An intuitive way to analyze the relation is based on expressing its Fourier transform $A_s(\omega)$ as follows,

$$A_s(\omega) = \frac{1}{2}[1 + sgn(\omega)]A_s(\omega) \quad (5.27)$$

The equation 5.27 follows the SSB signal condition $A_s(\omega) = 0$ for $\omega < 0$. Further, simplification0n of the signal can be summarized as follows:

$$\begin{aligned} A_s(\omega) &= \frac{1}{2}[1 + sgn(\omega)]A_s(\omega) \\ &= \frac{1}{2}A_s(\omega) + \frac{1}{2}sgn(\omega)A_s(\omega) \end{aligned} \quad (5.28)$$

Taking inverse Fourier transform of the equation 5.28,

$$\begin{aligned} A_s(t) &= IFT\{A_s(\omega)\} \\ &= \underline{\frac{1}{2}A_s(t) + \frac{1}{2}[IFT\{sgn(\omega)\} \otimes A_s(t)]} \end{aligned} \quad (5.29)$$

The underlined term in Equation 5.29 displays that multiplication in frequency domain converted into the convolution in the time domain. Further, IFT of the function $sgn(\omega)$ given as $(-i/\pi t)$. As a consequences, we can further simplify our equation as,

$$\begin{aligned} A_s(t) &= \frac{1}{2}A_s(t) + \frac{1}{2}\left[\frac{i}{\pi t} \otimes A_s(t)\right] \\ \frac{A_s(t)}{2} &= \frac{1}{2}\left[\frac{i}{\pi t} \otimes A_s(t)\right] \\ A_s(t) &= i\left[\frac{1}{\pi t} \otimes A_s(t)\right] \\ A_s(t) &= \frac{i}{\pi} p.v. \int_{-\infty}^{\infty} \frac{A_s(t')}{t - t'} dt' \end{aligned} \quad (5.30)$$

Using Equation 5.26 into Equation 5.30,

$$A_{s,r}(t) + iA_{s,i}(t) = \frac{i}{\pi} p.v. \int_{-\infty}^{\infty} \frac{A_s(t')}{t - t'} dt' \quad (5.31)$$

Therefore,

$$\begin{aligned} A_{s,r}(t) + iA_{s,i}(t) &= \frac{i}{\pi} p.v. \int_{-\infty}^{\infty} \frac{A_{s,r}(t') + iA_{s,i}(t')}{t - t'} dt' \\ A_{s,r}(t) + iA_{s,i}(t) &= -\frac{1}{\pi} p.v. \int_{-\infty}^{\infty} \frac{A_{s,i}(t')}{t - t'} dt' + \frac{i}{\pi} p.v. \int_{-\infty}^{\infty} \frac{A_{s,r}(t')}{t - t'} dt' \end{aligned} \quad (5.32)$$

which leads to,

$$\begin{aligned} A_{s,r}(t) &= -\frac{1}{\pi} p.v. \int_{-\infty}^{\infty} \frac{A_{s,i}(t')}{t - t'} dt' \\ A_{s,i}(t) &= \frac{1}{\pi} p.v. \int_{-\infty}^{\infty} \frac{A_{s,r}(t')}{t - t'} dt' \end{aligned} \quad (5.33)$$

Minimum Phase signal:

Given function $A(t) = A_s(t) + \bar{A}$ never encircles the origin for $t \in (-\infty, \infty)$. if we define,

$$G(t) = \log \left[\frac{A(t)}{\bar{A}} \right] \quad (5.34)$$

then $G(\omega)$, the spectrum of $G(t)$, is such that $G(\omega) = 0$ for $\omega < 0$. Under the hypothesis stated by Equation 5.31, we can write $G(t)$ as,

$$G(t) = \frac{i}{\pi} p.v. \int_{-\infty}^{\infty} \frac{G(t')}{t - t'} dt' \quad (5.35)$$

From the equations 5.34 and 5.35,

$$\log|A(t)| - \log|\bar{A}| + i[\phi(t) - \bar{\phi}] = \frac{i}{\pi} p.v. \int_{-\infty}^{\infty} \frac{\log|A(t)| - \log|\bar{A}|}{t - t'} dt' + \frac{1}{\pi} p.v. \int_{-\infty}^{\infty} \frac{\phi(t) - \bar{\phi}}{t - t'} dt' \quad (5.36)$$

Comparing Imaginary part of Equation 5.36,

$$\phi(t) - \bar{\phi} = +\frac{1}{\pi} p.v. \int_{-\infty}^{\infty} \frac{\log|A(t)| - \log|\bar{A}|}{t - t'} dt' \quad (5.37)$$

In equation 5.37, $\frac{1}{\pi} p.v. \int_{-\infty}^{\infty} \frac{\log|\bar{A}|}{t - t'} dt' = 0$ which leads to,

$$\begin{aligned} \phi(t) &= \bar{\phi} + \frac{1}{\pi} p.v. \int_{-\infty}^{\infty} \frac{\log|A(t)|}{t - t'} dt' \\ \phi(t) &= \bar{\phi} + \frac{1}{2\pi} p.v. \int_{-\infty}^{\infty} \frac{\log|A(t)|^2}{t - t'} dt' \end{aligned} \quad (5.38)$$

5.4.2 Simulation Analysis

Transmitter setup

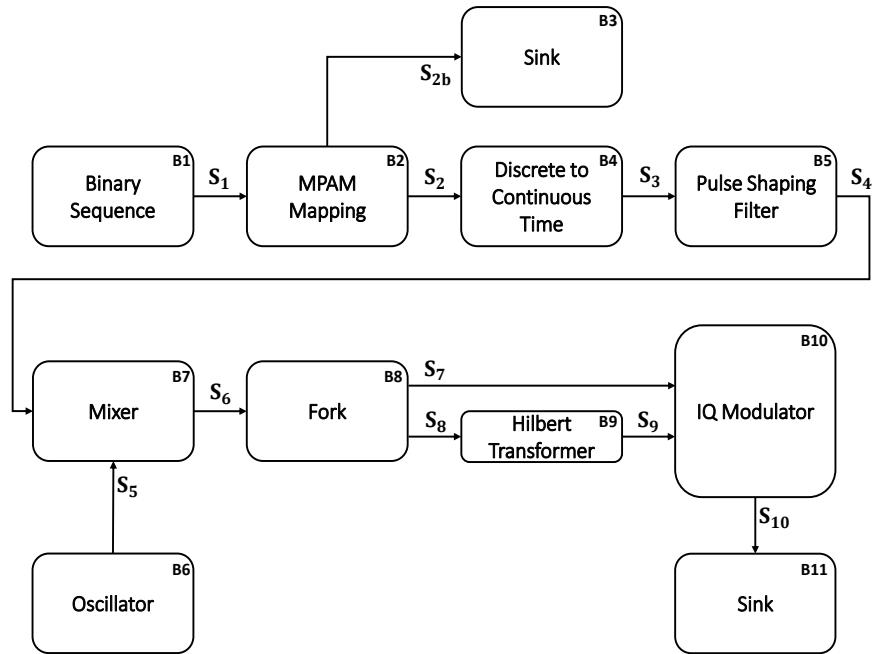


Figura 5.14: Transmitter simulation setup

System input parameters:

Parameter	Default Value	Description
sourceMode	PseudoRandom	
patternLength	5	
bitPeriod	1/1.25e9	
iqAmplitudes	{{0,0},{1,0},{2,0},{3,0}}	
numberOfBits	1000	
numberOfSamplesPerSymbol	8	
filterType	RaisedCosine	
rollOffFactor	0.3	
impulseResponseTimeLength	16	
outputOpticalPower	1e-3	

Transmitter setup description:

Header Files		
File name	Comments	Status
binary_source.h		✓
m_qam_mapper.h	DONE!	✓
discrete_to_continuous_time.h		✓
pulse_shaper.h		✓
RF_Oscillator.h	DONE!	✓
mixer.h	DONE!	✓
fork.h	DONE!	✓
hilbert_transform.h	DONE!	✓
iq_modulator.h		✓
sink.h		✓
netxpto.h		✓

Source Files		
File name	Comments	Status
binary_source.cpp		✓
m_qam_mapper.cpp	DONE!	✓
discrete_to_continuous_time.cpp		✓
pulse_shaper.cpp		✓
RF_Oscillator.cpp	DONE!	✓
mixer.cpp	DONE!	✓
fork.cpp	DONE!	✓
hilbert_transform.cpp		
iq_modulator.cpp		✓
sink.cpp		✓
netxpto.cpp		✓
kramers_kronig_transceiver_sdf.cpp		✓

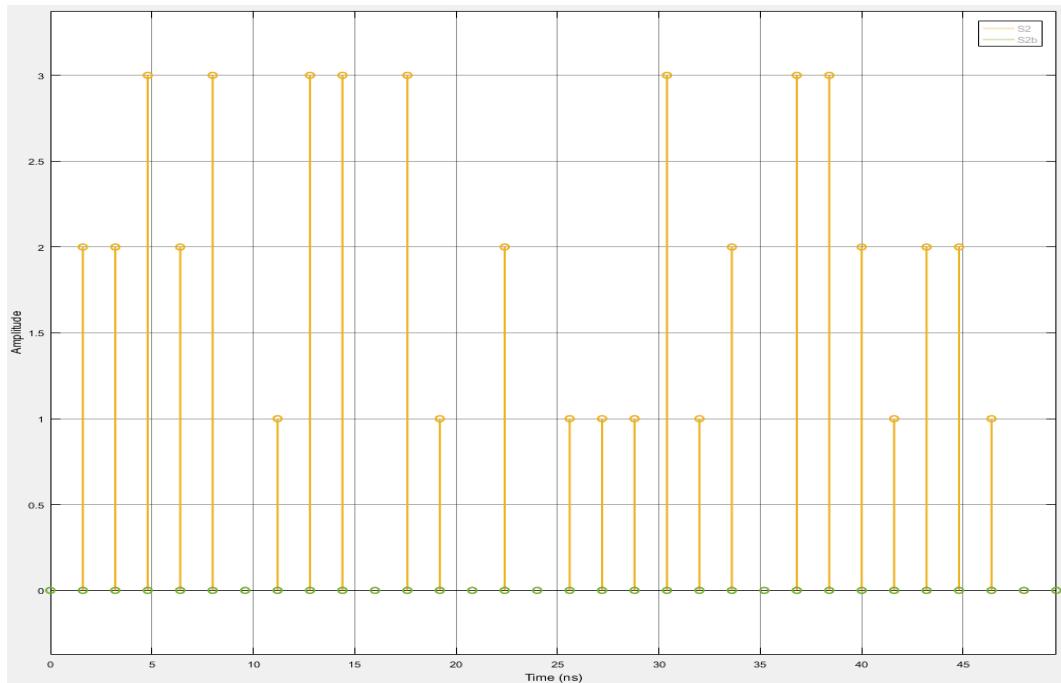
Simulation results:

Figura 5.15: S2 and S2b

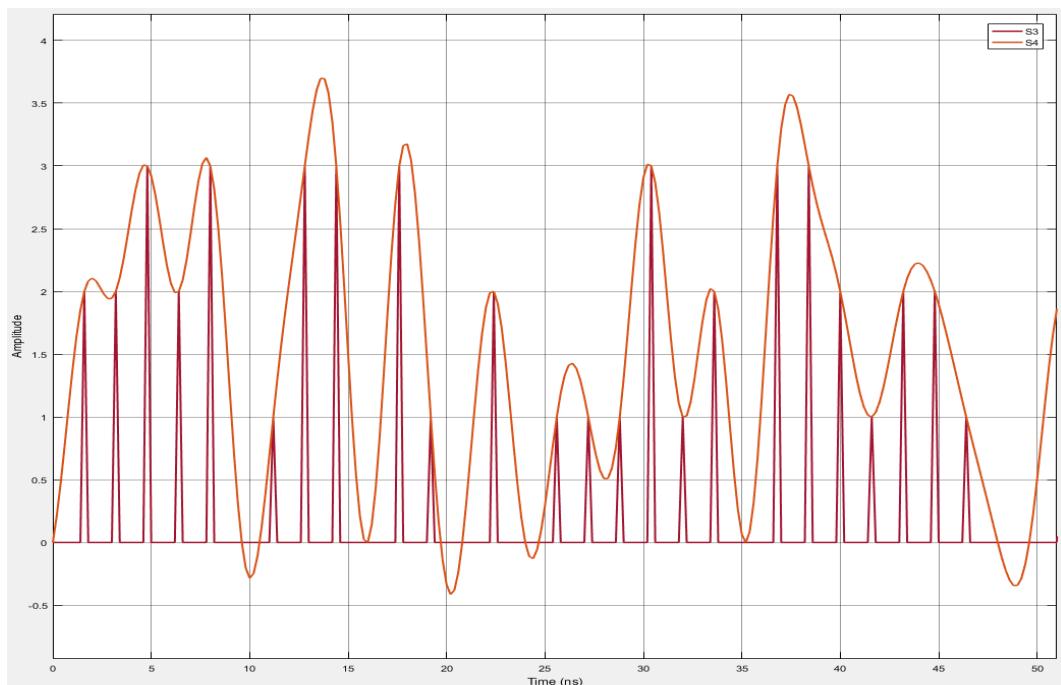


Figura 5.16: S3 and S4

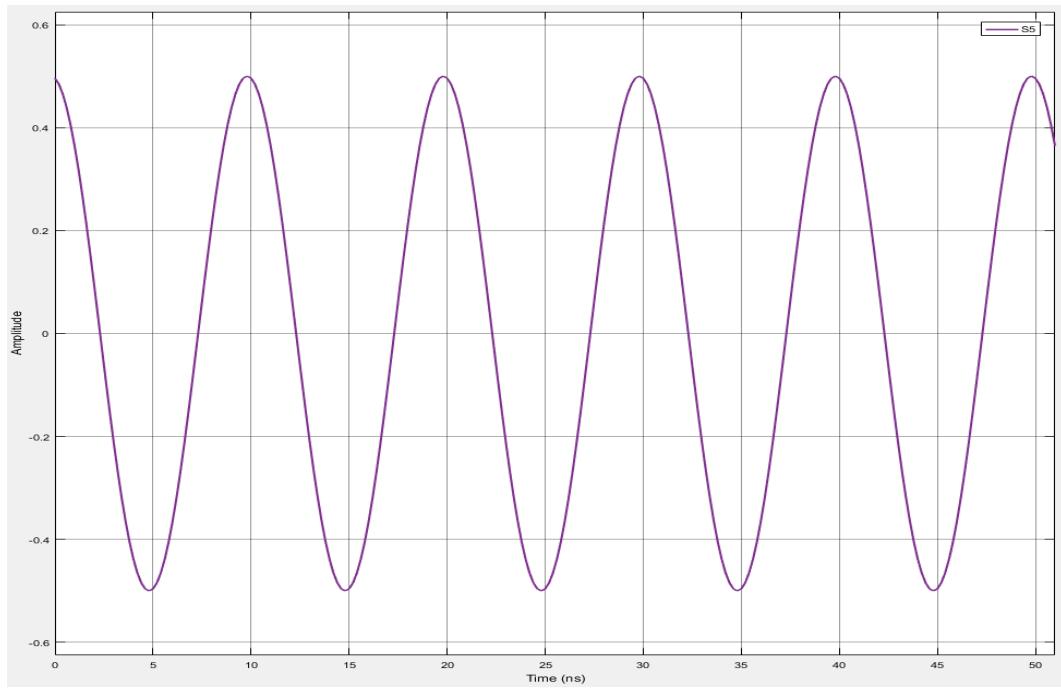


Figura 5.17: S5

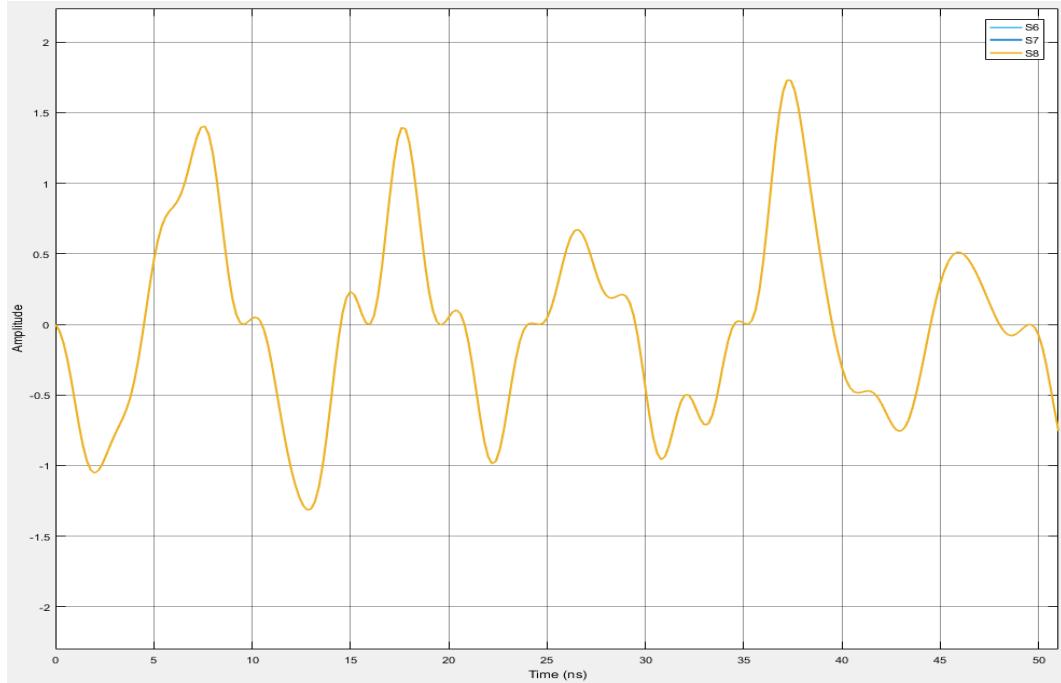


Figura 5.18: S6, S7 and S8

Receiver setup

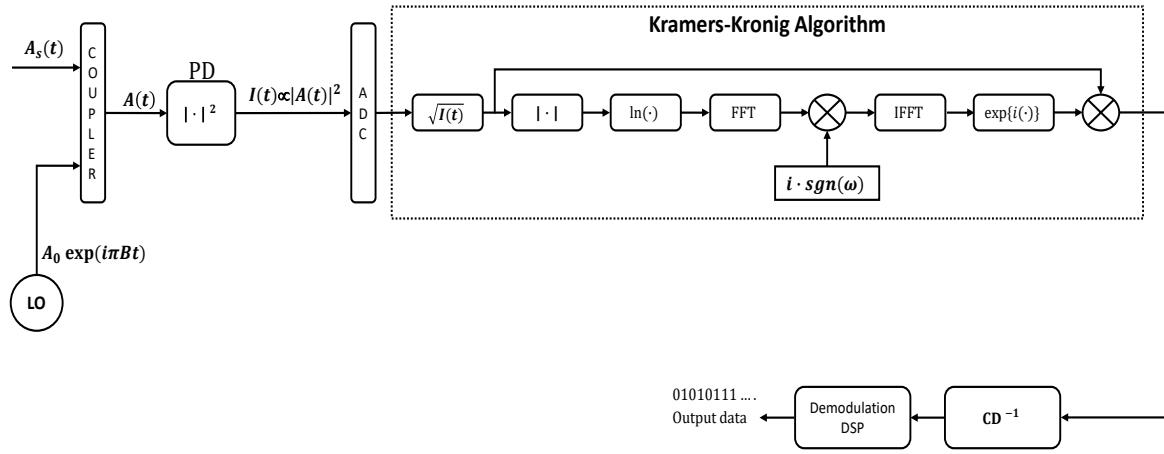


Figura 5.19: Receiver simulation setup

5.4.3 Experimental Analysis

As shown in the Figure 5.20, at the transmitter end, analytical signal generated with the help of netxpto and applied to the AWG. Waveform generated by AWG applied dual polarization IQ modulator which generates SSB signal in optical domain. SSB optical signal generated by both the IQ modulator is then combined using polarization beam combiner (PBC) and launched into the optical fiber.

At the receiver end, the PDM received signal first spitted by a polarization beam splitter (PBS). Each polarization is combined with an LO tapped from the transmit laser. The laser's wavelength and power are set to ensure that the received signal should satisfy the minimum phase condition. After direct-detection, Kramers-Kronig algorithm is performed on each polarization separately to recover full complex signal. After compensation of the chromatic dispersion, stokes parameter based poldemux algorithm can be applied to recover PDM signals.

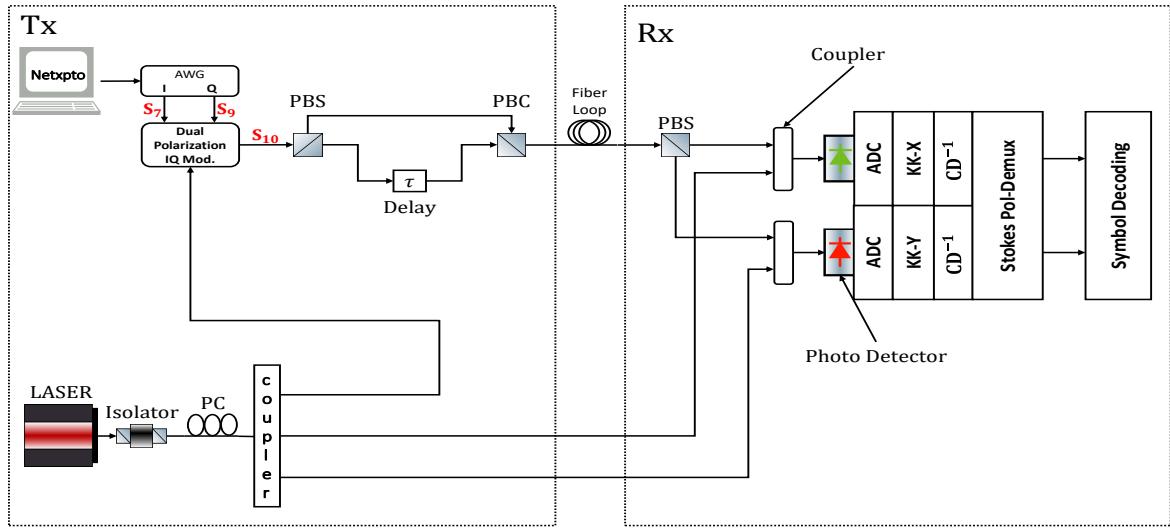


Figura 5.20: PDM Kramers-Kronig receiver experimental setup

Status of equipment

Equipment name	Description	Status
LASER		✓
PC		✓
Coupler		✓
AWG		✓
Dual polarization IQ mod		✓
PBS		✓
PLC		✓
Delay		✓
Fiber loop		✓
Single photodetector	APD+TIA Optical Receiver: - Maximum bit rate: 10 Gb/s - Sensitivity: -26 dBm	✓

5.4.4 Comparative Analysis

5.4.5 Known Problems

Problem type	Description	Note
Simulator	Require a generalized block for FFT and IFFT	

Bibliografia

- [1] Antonio Mecozzi, Cristian Antonelli, and Mark Shtaif. *Kramers-Kronig Coherent Receiver*. Optica, vol.3, no.11, 2016, p.1220., doi:10.1364/optica.3.001220.
- [2] Antonio Mecozzi. *Retrieving the full optical response from amplitude data by Hilbert transform*. Opt. Comm. 282, 4183-4187.
- [3] Antonio Mecozzi. *A necessary and sufficient condition for minimum phase and implication of phase retrieval*. arXiv:1606.04861.

APPENDICES

Appendix A : SSB with graphical explanation

This section describes the SSB signal generation using Hilbert transformation method (Phase Shift Method). Consider a message signal $m(t)$ with its frequency domain spectrum $M(F)$ as shown in Figure 5.21. From the Figure 5.21, we can see that both the side are scaled by factor '1' which means it represents the original signal.

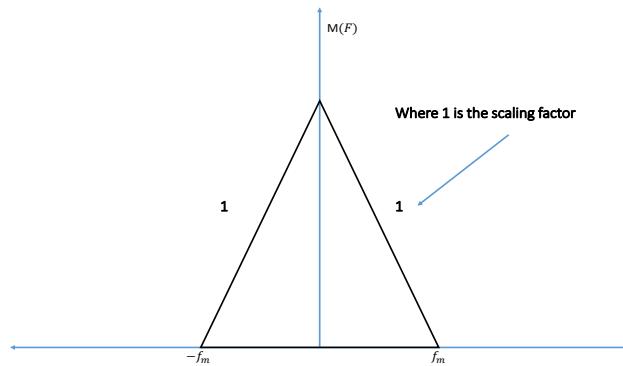


Figura 5.21: Original baseband signal

Now let's consider the modulated signal $x(t)$ given as,

$$x(t) = m(t)\cos(2\pi f_c t) \quad (5.39)$$

Frequency domain representation of the equation 5.39 can be given as,

$$X(F) = \frac{1}{2}M(f - f_c) + \frac{1}{2}M(f + f_c) \quad (5.40)$$

Here in equation 5.40, we can observe that each side band are scaled by $\frac{1}{2}$ on the frequency spectrum. Figure displays the frequency domain representation of the modulated signal $X(F)$.

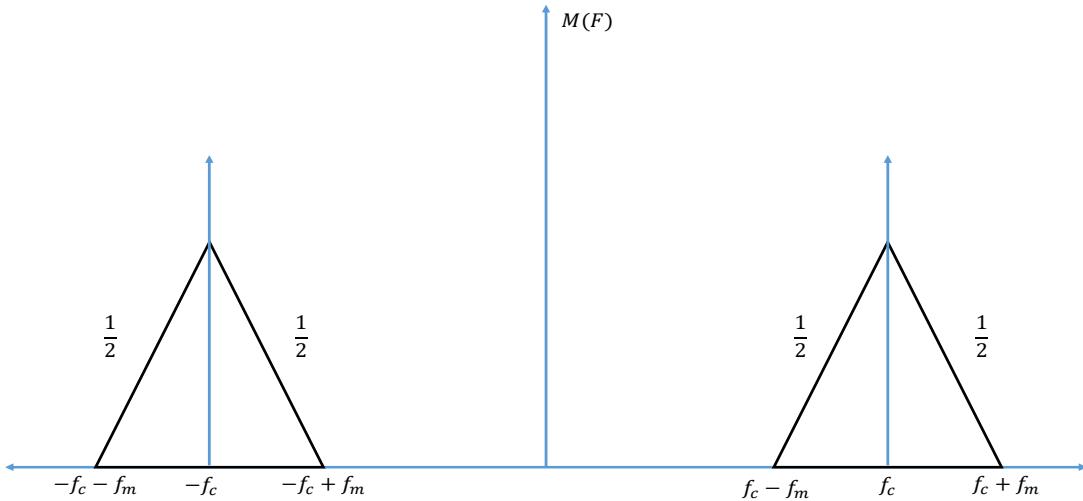


Figura 5.22: Original modulated signal

Next, we will discuss something more interesting which is called as Hilbert transform of the original message signal $m(t)$. As we discussed earlier, in the frequency domain, the Hilbert transformed signal $\hat{M}(f)$ can be achieved by multiplying the Fourier transformed signal $M(F)$ with $[-isgn(F)]$. Suppose we modulate the Hilbert transformed message signal $\hat{m}(t)$

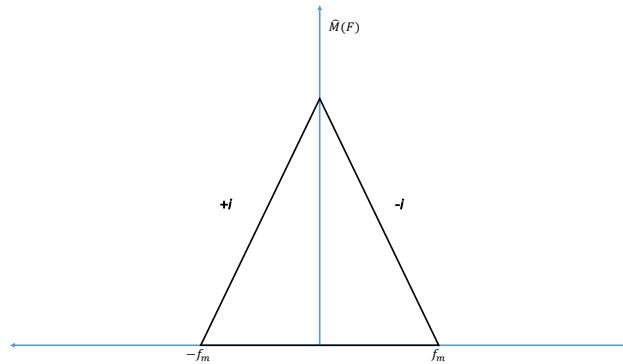


Figura 5.23: Hilbert transformed modulated signal

with the $\sin(2\pi f_c t)$ (quadrature phase carrier), then we get the following results:

$$\begin{aligned}
 \hat{m}(t) \sin(2\pi f_c t) &= \hat{m}(t) \frac{e^{i2\pi f_c t} - e^{-i2\pi f_c t}}{2} \\
 &= \hat{m}(t) \frac{e^{i2\pi f_c t}}{2} - \hat{m}(t) \frac{e^{-i2\pi f_c t}}{2} \\
 &= \frac{\hat{M}(f - f_c)}{2i} - \frac{\hat{M}(f + f_c)}{2i} \\
 &= \frac{-i}{2} \hat{M}(f - f_c) + \frac{-i}{2} \hat{M}(f + f_c)
 \end{aligned} \tag{5.41}$$

The detailed explanation of the equation 5.41 has been given in the Figure 5.24 and 5.25. Figure 5.24 displays the spectrum of the $\hat{M}(f + f_c)$ and $\hat{M}(f - f_c)$ for the positive and negative frequencies respectively. The final equation resolution of equation displays that both positive and negative side of the spectrum multiplied with $\frac{i}{2}$ and $-\frac{i}{2}$ respectively. Finally the spectrum of the signal $\hat{m}(t) \sin(2\pi f_c t)$ can be given as Figure 5.25.

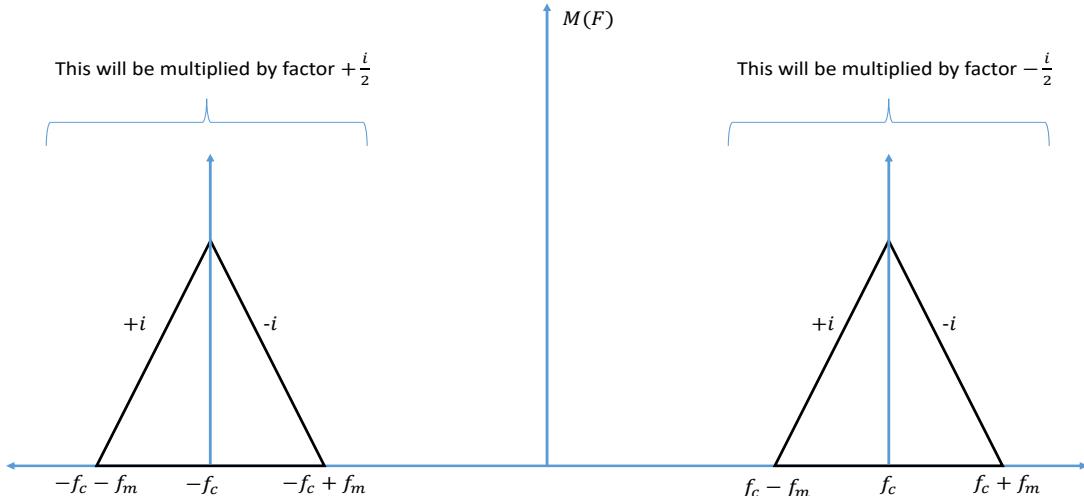


Figura 5.24: Hilbert transformed modulated signal

Further, summation of the two signals $m(t) \cos(2\pi f_c t)$ and $\hat{m}(t) \sin(2\pi f_c t)$ will generate the upper sideband SSB signal as follows,

$$u(t) = m(t) \cos(2\pi f_c t) - \hat{m}(t) \sin(2\pi f_c t) \tag{5.42}$$

From the above discussion, the spectrum of the Equation 5.42 can be given by the Figure 5.26. Similarly, for the lower sideband SSB can be generated by Equation,

$$u(t) = m(t) \cos(2\pi f_c t) + \hat{m}(t) \sin(2\pi f_c t) \tag{5.43}$$

Appendix B : Kramers-Kronig scheme

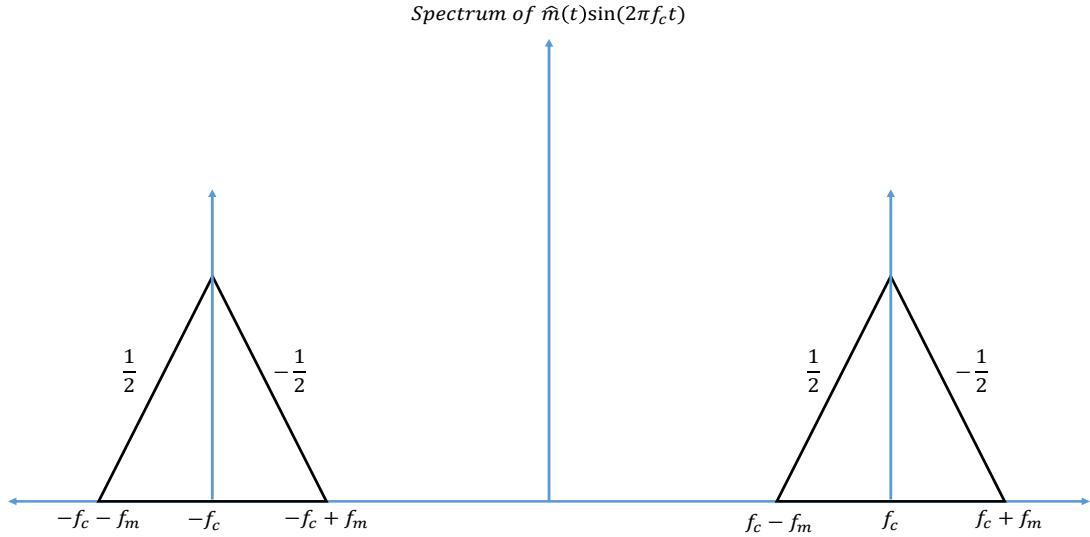


Figura 5.25: Hilbert transformed modulated signal

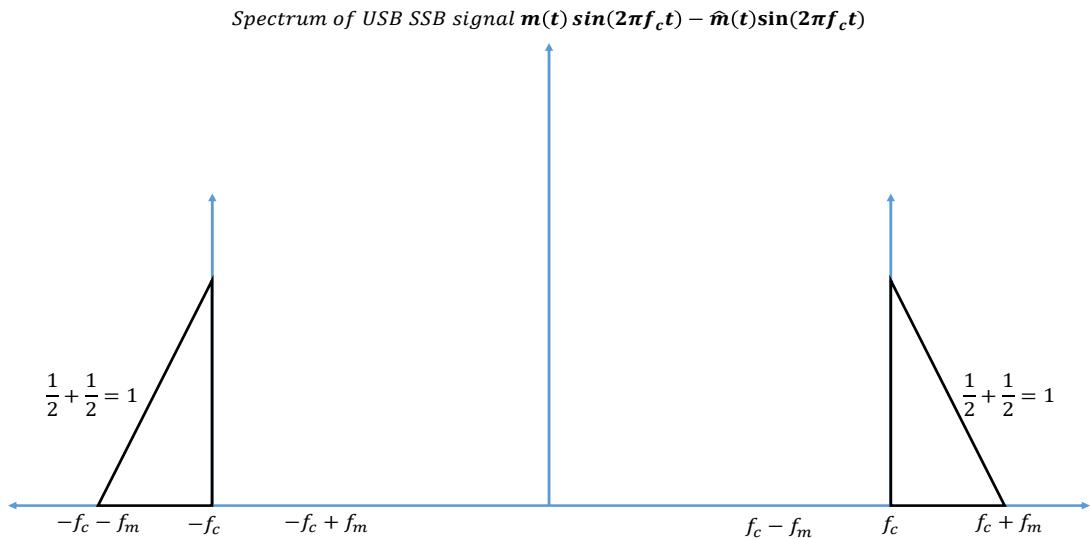


Figura 5.26: SSB signal spectrum

If we consider the complex envelope of the incoming electric field by $A_s(t)$ confined within the optical bandwidth denoted by B . The LO assumed to be a continuous wave (CW) signal whose amplitude is A_0 whose frequency coincides with the left edge of the information-carrying signal spectrum. Here, we assumed that A_0 is real-valued and positive, which is equivalent to referring all phase value to that of LO.

The complex envelope of the field striking upon the photo-diode can be given as,

$$A(t) = A_s(t) + A_0 \exp(i\pi Bt) \quad (5.44)$$

The photo current I produced by the photo-diode is proportional to the field intensity $I = |A(t)|^2$, here proportionality constant considered as 1 for the sake of simplicity. If A_0 is large enough to ensure that the signal $A(t)\exp(-i\pi Bt) = A_0 + A_s(t)\exp(-i\pi Bt)$ is minimum phase. The discussed hypothesis can be used to reconstruct the signal $E_s(t)$ as follows[1]:

$$A_s(t) = \{\sqrt{I(t)}\exp[i\phi_E(t)] - A_0\}\exp(i\pi Bt) \quad (5.45)$$

$$\phi_A(t) = \frac{1}{2\pi} p.v. \int_{-\infty}^{\infty} dt' \frac{\log[|I(t')|]}{t - t'} \quad (5.46)$$

5.5 Radio Over Fiber Transmission System

Student Name	:	Celestino Martins
Starting Date	:	September 25, 2017
Goal	:	Simulation of Radio over fiber Transmission considering the uplink of base station cooperation systems.

Radio over fiber (RoF) technology comprises the transmission over fiber technology, where radio signal is modulated onto optical carrier and transmitted over an optical fiber link to provide a simple antenna front ends with increased capacity and broadband wireless services. In this network a central processing units (CPU) is connected to numerous base stations (BSs) via optic fibers. That means, RoF networks use optic fiber links to distribute radio frequency (RF) signals between the CPU and BSs. The downlink RF signals are distributed from a CPU to many BSs through the fibres, while the uplink signals received at BSs are sent back to the CPU for any signal processing. Figure 5.27 shows a general RoF architecture, where the wireless signals are transported over the optical fiber between a CPU and a set of base stations before being radiated through the air. RoF transmission systems are usually classified into two main categories, depending on the frequency range of the radio signal to be transported: i) RF-over-Fiber; ii) intermediate frequency (IF)-over-Fiber.

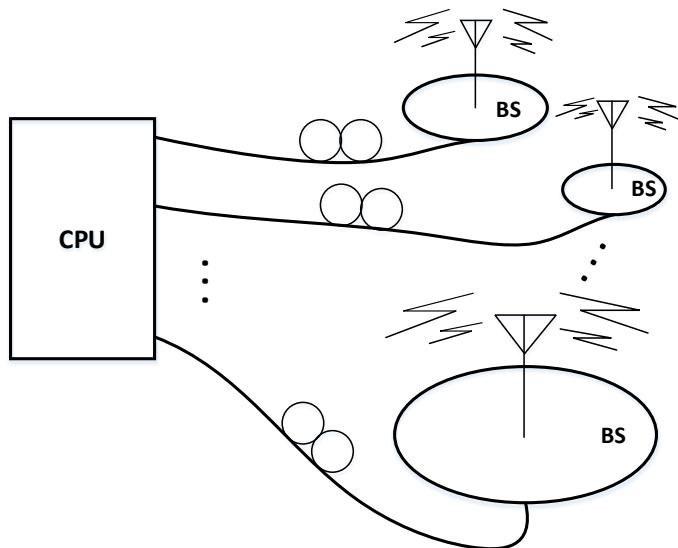


Figura 5.27: Schematic showing the concept of a centralized CPU architecture for future integrated optical wireless networks based on RoF.

- (i) In RF-over-Fiber architecture, a data-carrying RF (Radio Frequency) signal with a high frequency (usually greater than 10 GHz) is imposed on a lightwave signal before being transported over the optical link. Therefore, wireless signals are optically distributed to base stations directly at high frequencies and converted from the optical to electrical

domain at the base stations before being amplified and radiated by an antenna. As a result, no frequency up/down conversion is required at the various base stations, thereby resulting in simple and rather cost-effective implementation is enabled at the base stations.

- (ii) In IF-over-Fiber architecture, an IF (Intermediate Frequency) radio signal with a lower frequency (less than 10 GHz) is used for modulating light before being transported over the optical link. Therefore, before radiation through the air, the signal must be up-converted to RF at the base station.

In addition, the RoF technology can be implemented as analog RoF or digital RoF:

- (i) In analog RoF technology, the analog signal is transmitted over the optical fiber, being either RF signal, IF signal or baseband BB signal. In the optical transmitter, the RF/IF/BB signal is modulated onto the optical carrier by either using direct or external modulation of the laser. In this case, the signal distribution through RoF has the advantage of simplified BS design, however it is susceptible to fiber chromatic dispersion and nonlinearity generated by optical devices.
- (ii) In the digitized RoF the wireless carrier RF signal is first digitized prior to transport over the optical link. The digitalization of an RF signal produces a sampled digital signal in a serial form that can be directly modulated on an optical carrier, transmitted over the fiber optic link, and then detected like any other digital information. Modulation of the digital signal onto an optical carrier minimizes the nonlinear effects originating from the optical-to-electrical conversion function presented on analog RoF. In order to use not so high sample rates at the ADC/DAC components generally the bandpass sampling technique is applied to the RF signal.

5.5.1 Theoretical Analysis

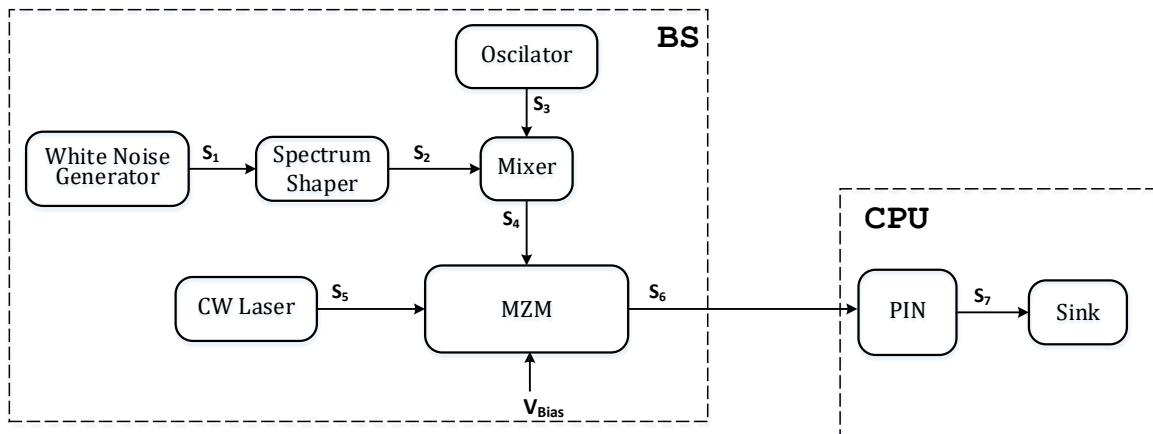


Figura 5.28: Simulation setup for the uplink of RoF transmission system;

Figure 5.28 depicts the simulation setup for an uplink of RoF, providing the connection between BS to CPU. At BS, we model the RF signal received from a mobile terminal, as zero mean complex gaussian (CG) signal with a given bandwidth imposed by the low pass filter, Filter-1. The generated baseband signal is then up-converted to RF carrier frequency by utilizing an oscillator and a mixer. In this simulation we consider the RF carrier frequency between 2 to 5 GHz, according to the 5G technologies specifications. The generated RF passband signal modulates optical carrier by utilizing a laser and a single driver mach-zehnder modulator (MZM). The optical signal is then transmitted to CPU using optical fiber. In CPU, the optical signal is detected by a PIN, amplified and followed with an electrical filter. After these operations, digital signal processing techniques can be applied to recover the transmitted signal.

Tabela 5.4: System Input Parameters.

Parameter	Default Value	Description
sourceMode		
symbolPeriod		
samplePeriod		
numberOfSamplesPerSymbol		
filterType1		
rollOffFactor		
filterType2		
outputOpticalPower		
outputOpticalWavelength		
rfCenterFrequency		
fiberAttenuation		

Tabela 5.5: Header Files for RoF Transmission System.

File Name	Description	Status
complex_gaussian_signal.h		
pulse_shaper.h		✓
local_oscillator.h		✓
mixer.h		
cw_laser.h		
iq_modulator.h		✓
fiber.h		
pin.h		
amplifier.h		
filter_rx.h		
sink.h		✓
netxpto.h		✓

Tabela 5.6: Source Files for RoF Transmission System.

File Name	Description	Status
complex_gaussian_signal.cpp		
pulse_shaper.cpp		✓
local_oscillator.cpp		✓
mixer.cpp		
cw_laser.cpp		
iq_modulator.cpp		✓
fiber.cpp		
pin.cpp		
amplifier.cpp		
filter_rx.cpp		
sink.cpp		✓
netxpto.cpp		✓

5.5.2 Experimental

5.6 Optical Detection

Contributors	:	Nelson Muga, (2017-12-21 - ...)
	:	Diamantino Silva, (2017-08-18 - ...)
	:	Armando Pinto (2017-08-15 - ...)
Goal	:	Analise of various optical detection schemes.

The detection of light is a fundamental stage in every optical communication system, bridging the optical domain into the electrical domain. This section will review various theoretical and practical aspects of light detection, as well a series of implementations and schemes. The objective of this work is to develop numerical models for the various detections schemes and to validate these numerical models with experimental results.

5.6.1 Theoretical Analysis

Contributors	:	Nelson Muga (2017-12-20 -)
	:	Diamantino Silva (2017-08-18 - ...)
	:	Armando Pinto (2017-08-15 - ...)
Goal	:	Theoretical description of various optical detection schemes.

The aim of this section is to calculate the signal to noise ratio at the input of the decision circuit for various detection schemes. For each detection scheme a classical and a quantum description is going to be developed and a comparative analise is going to performed.

Classical Description

Contributors	:	Nelson Muga (2017-12-20 -)
	:	Diamantino Silva (2017-08-18 - ...)
	:	Armando Pinto (2017-08-15 - ...)
Goal	:	Develop a classical description of various optical detection schemes.

Direct Detection

One of the most simple detection methods is the direct detection of light with a single detector and the analise of the resulting photocurrent.

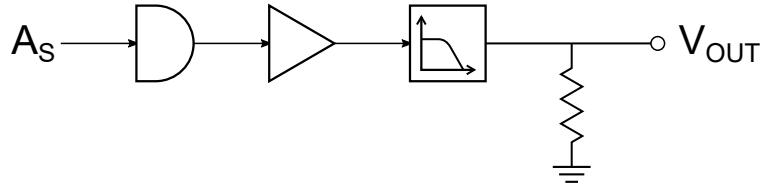


Figura 5.29: Direct detection.

Given an electric field generated by an ideal monochromatic single mode laser, it can be modulated in amplitude and phase by a signal $A(t)$ resulting in the function

$$E_R(t) = \sqrt{2}|A(t)|\cos(-\omega t + \theta) \quad \sqrt{W} \quad (5.47)$$

with $A(t) = |A(t)|e^{-i\theta(t)}$.

We will consider that the detector has a bandwidth B , greater than the signal $A(t)$, but much smaller than $2\omega_0$. The calculation of power incident in the photodiode is given by the expected value of the square of the amplitude during a time interval $\Delta t = 2\pi/\omega$

Measurable optical power, assuming that the detector bandwidth, B , is greater than the signal, $A(t)$, bandwidth but much small than $2\omega_0$

$$\begin{aligned} P(t) &= \overline{E_R^2(t)} \\ &= \overline{|A(t)|^2} + \overline{|A(t)|^2 \cos(-2\omega t + 2\theta(t))} \\ &= |A(t)|^2 \quad W \end{aligned} \quad (5.48)$$

To simplify calculations, the electric field can be expressed the complex notation

$$E(t) = A(t)e^{-i\omega_0 t} \quad (5.49)$$

The physically measurable quantities are obtained by taking the real part of the complex wave. Using this notation, the beam power, $P(t)$, is obtained by multiplying the electric field's conjugate by itself

$$\begin{aligned} P(t) &= E^*(t)E(t) \\ &= |A(t)|^2 \end{aligned} \quad (5.50)$$

$$i(t) = \eta q \frac{P(t)}{\hbar\omega_0} \quad (5.51)$$

in which η is the photodiode's responsivity, q is the unit charge and $P(t)/\hbar\omega_0$ is the number of removed electrons.

The signal is ...

$$E(t) = A(t)e^{-i\omega_0} \quad (5.52)$$

Using the definition of electric power of a complex electric field representation, we will get

$$P(t) = |A(t)|^2 \quad (5.53)$$

recovering the result of the real representation. The photocurrent can be rewritten as a function of the signal $A(t)$

$$i(t) = \eta q \frac{|A(t)|^2}{\hbar\omega_0} \quad (5.54)$$

which will use to express the second moment of the photocurrent as

$$\langle i^2(t) \rangle = \eta^2 q^2 \frac{\langle |A(t)|^4 \rangle}{\hbar^2 \omega_0^2} \quad (5.55)$$

Assuming a fase modulation, in which the amplitude is constant, the signal is simplified to

$$A(t) = |A|e^{i\theta} \quad (5.56)$$

Therefore, the current becomes constant

$$i(t) = I_0 = \eta q \frac{A_s^2}{\hbar\omega_0} \quad (5.57)$$

and it's second moment becomes simply

$$\langle i^2(t) \rangle = I_0^2 \quad (5.58)$$

Shot noise in photodiodes

$$\langle i_n^2(t) \rangle = 2qBI_0 \quad (5.59)$$

The signal to noise ratio is obtained by the relation between the second moment of the sinal to the second moment of the noise

$$\begin{aligned} \frac{S}{N} &= \frac{\langle i^2(t) \rangle}{\langle i_n^2(t) \rangle} \\ &= \frac{I_0}{2qB} \\ &= \eta \frac{|A|^2}{\hbar\omega_0 B} \end{aligned} \quad (5.60)$$

Homodyne Detection

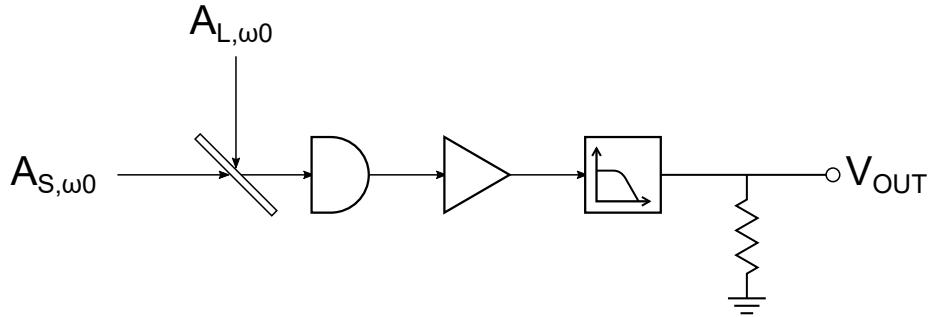


Figura 5.30: Homodyne detection.

The homodyne detection scheme uses an auxiliary local oscillator, which is combined in a beamsplitter with the signal beam. After this step, it is similar to the direct detection. As we will see this has some implications in the phase detection???

Given a splitter with intensity transmission ϵ , the resulting field incident to the photodetector is [?]

$$E(t) = \sqrt{\epsilon}E_S(t) + \sqrt{1-\epsilon}E_{LO}(t) \quad (5.61)$$

in which $E_{LO} = e^{i\omega_0 t}$. Given a local oscillator with a much larger power than the signal, then, the incident power in the photodiode is

$$P(t) = \eta \left[(1 - \epsilon)P_{LO}(t) + 2\sqrt{\epsilon(1 - \epsilon)}\text{Re}[E_S(t)E_{LO}^*(t)] \right] \quad (5.62)$$

$$= \eta \left[(1 - \epsilon)P_{LO}(t) + 2\sqrt{\epsilon(1 - \epsilon)}|E_S(t)||E_{LO}(t)|\cos(\phi) \right] \quad (5.63)$$

$$(5.64)$$

Balanced Homodyne Detection

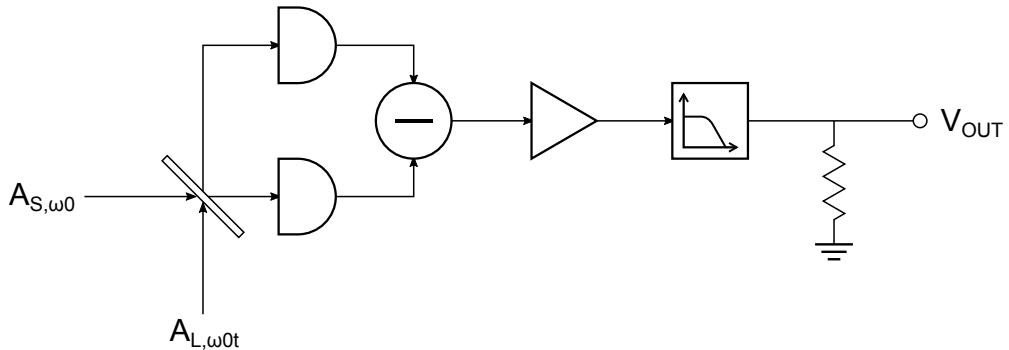


Figura 5.31: Balanced homodyne detection.

"In a balanced homodyne detector (BHD), the signal to be measured is mixed with a local oscillator (LO) at a beam splitter. The interference signals from the two output ports of the beam splitter are sent to two photodiodes followed by a subtraction operation, and then, amplification may be applied. The output of a BHD can be made to be proportional to either the amplitude quadrature or the phase quadrature of the input signal depending on the relative phase between the signal and the LO".

IQ Homodyne Balanced Detection

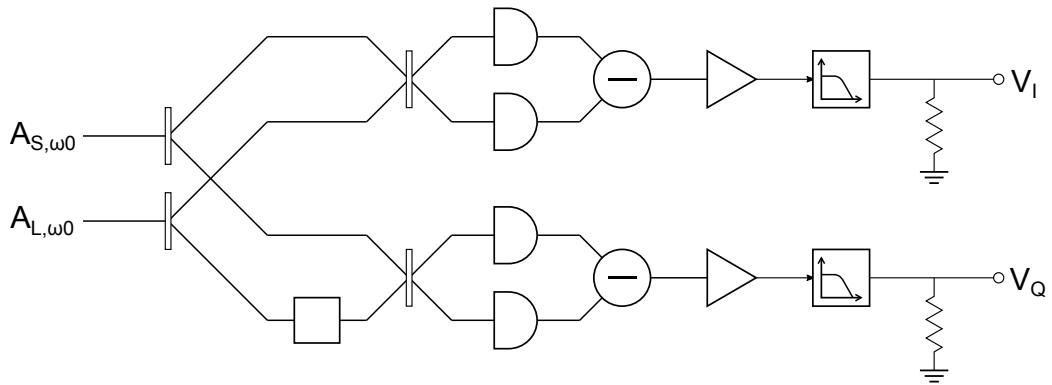


Figura 5.32: IQ balanced homodyne detection.

Semiclassical model

Quantum model

Heterodyne Detection

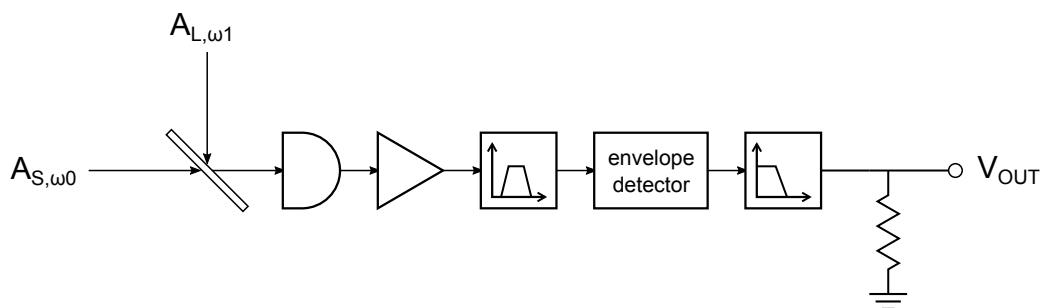


Figura 5.33: Heterodyne detection.

In contrast with the homodyne detection, in which the frequency of the signal carrier is equal to the frequency of the local oscillator, in the heterodyne detection, these frequencies are different.

Because of this, the inference will result in a new signal with an intermediate frequency at...

Balanced Heterodyne Detection

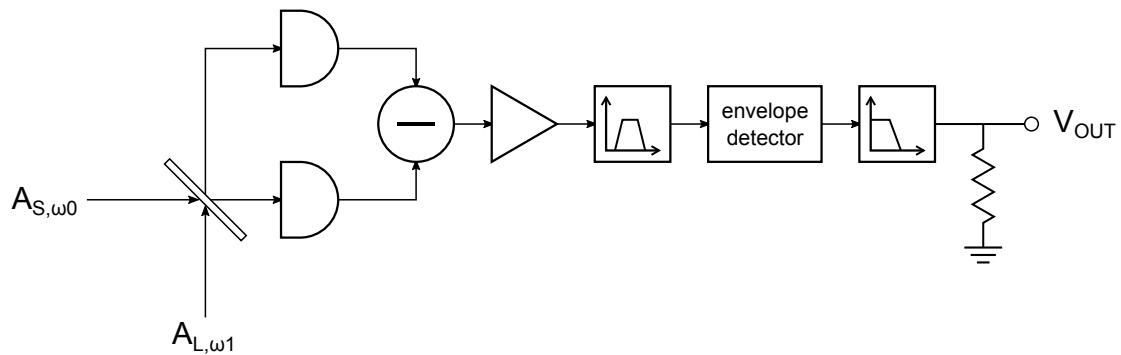


Figura 5.34: Balanced heterodyne detection.

Semiclassical model

Quantum model

IQ Heterodyne Balanced Detection

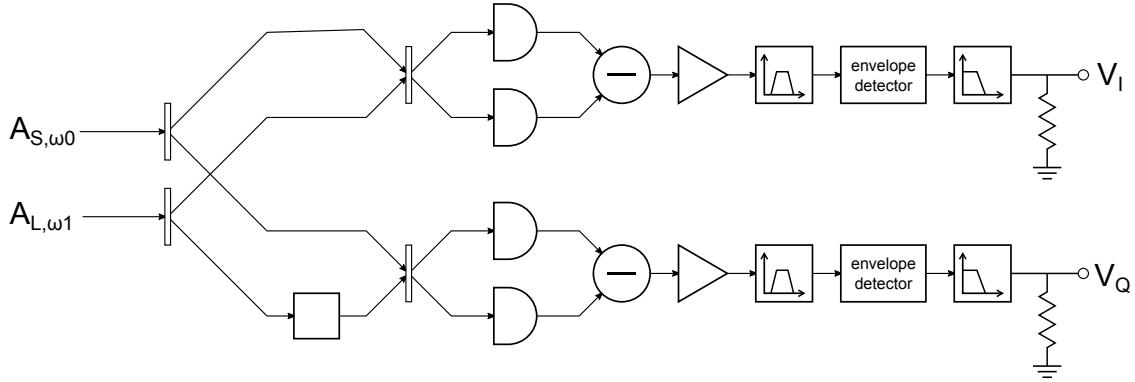


Figura 5.35: IQ balanced heterodyne detection.

Thermal noise

Thermal noise is generated by electrons in response to temperature. Its contribution to the resulting current can be described by the following equation [?]

$$\langle (\Delta i_T)^2 \rangle = 4K_B T_0 B / R_L \quad (5.65)$$

in which K_B it's Boltzmann's constant, T_0 is the absolute temperature, B is the bandwidth and R_L is the receiver load impedance. The B value is imposed by default or chosen when the measurements are made, but the R_L value is dependent in the internal setup of the various components of the detection system. Nevertheless, for simulation purposes, we can just introduce an experimental value.

Quantum Description

Contributors	:	Diamantino Silva (2017-08-18 - ...)
	:	Armando Pinto (2017-08-15 - ...)
Goal	:	Develop a quantum description of various optical detection schemes, and compare with the classical description.

We start by defining number states $|n\rangle$ (or Fock states), which correspond to states with perfectly fixed number of photons [?]. Associated to those states are two operators, the creation \hat{a}^\dagger and annihilation \hat{a} operators, which in a simple way, remove or add one photon from a given number state [?]. Their action is defined as

$$\hat{a}|n\rangle = \sqrt{n}|n-1\rangle \quad (5.66), \quad \hat{a}^\dagger|n\rangle = \sqrt{n+1}|n+1\rangle \quad (5.67), \quad \hat{n}|n\rangle = n|n\rangle \quad (5.68)$$

in which $\hat{n} = \hat{a}^\dagger \hat{a}$ is the number operator. Therefore, number states are eigenvectors of the number operator.

Coherent states have properties that closely resemble classical electromagnetic waves, and are generated by single-mode lasers well above the threshold. [?] We can define them, using number states in the following manner

$$|\alpha\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \quad (5.69)$$

in which the complex number α is the sole parameter that characterizes it. In fact, if we calculate the expected number of photons with $\langle \alpha | \hat{n} | \alpha \rangle$ we will obtain $|\alpha|^2$. The coherent state is an eigenstate of the annihilation operator, $\hat{a} |\alpha\rangle = \alpha |\alpha\rangle$.

Using the creation and annihilation operators, we can define two quadrature operators [?]

$$\hat{X} = \frac{1}{2} (\hat{a}^\dagger + \hat{a}) \quad (5.70), \quad \hat{Y} = \frac{i}{2} (\hat{a}^\dagger - \hat{a}) \quad (5.71)$$

The expected value of these two operators, using a coherent state $|\alpha\rangle$ are

$$\langle \hat{X} \rangle = \text{Re}(\alpha) \quad (5.72), \quad \langle \hat{Y} \rangle = \text{Im}(\alpha) \quad (5.73)$$

We see that the expected value of these operators give us the real and imaginary part of α . Now, we can obtain the uncertainty of these operators, using:

$$\text{Var}(\hat{X}) = \langle \hat{X}^2 \rangle - \langle \hat{X} \rangle^2 \quad (5.74)$$

For each of these quadrature operators the variance will be

$$\text{Var}(\hat{X}) = \text{Var}(\hat{Y}) = \frac{1}{4} \quad (5.75)$$

This result shows us that for both quadratures, the variance of measurement is the same and independent of the value of α .

Homodyne detection

The measurement of a quadrature of an input signal (S) is made by using the balanced homodyne detection technique, which measures the phase difference between the input signal and a local oscillator (LO). The measurement of quadrature are made relative to a reference phase of the LO, such that if the measurement is made in-phase with this reference, the value will be proportional to the \hat{X} quadrature of the signal. If the phase of the LO has an offset of $\pi/2$ relative to the reference, the output will be proportional to the \hat{Y} quadrature of the signal.

Experimentally, the balanced homodyne detection requires a local oscillator with the same frequency as the input signal, but with a much larger amplitude. These two signals

are combined using a 50:50 beam splitter, from where two beams emerge, which are then converted to currents using photodiodes. Finally, the two currents are subtracted, resulting in an output current proportional to a quadrature of the input signal [?].

A phase of the local oscillator can be defined as the reference phase. A phase offset equal to 0 or $\pi/2$ will give an output proportional to the signal's in-phase component or to the quadrature component, respectively. Therefore, the \hat{X} operator will correspond to the in-phase component and \hat{Y} operator correspond to quadrature component

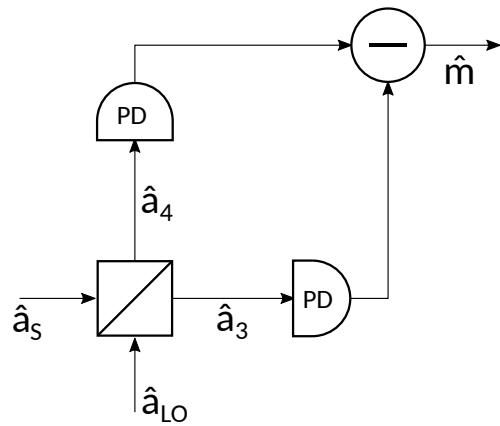


Figura 5.36: Balanced homodyne detection.

In the lab and in our simulations, a more complex system is used, the double balanced homodyne detection, which allows the simultaneous measurement of the \hat{X} and \hat{Y} components. The signal is divided in two beam with half the power of the original. One of the beams is used in a balanced homodyne detection with a local oscillator. The other beam is used in another balanced homodyne detection, but using a local oscillator with a phase difference $\pi/2$ relative to the first one.

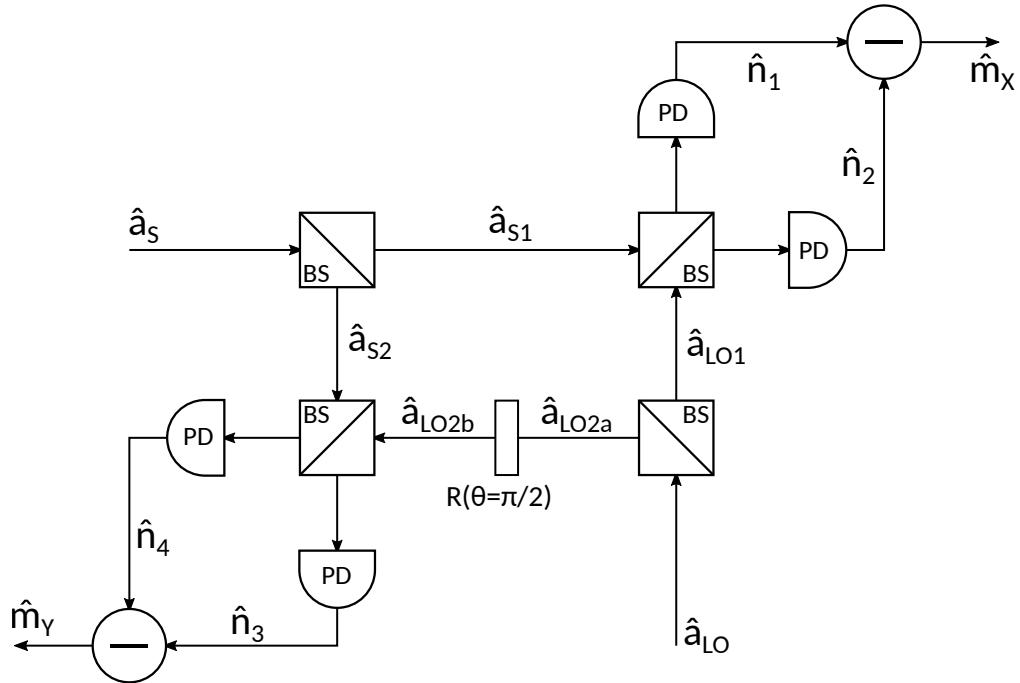


Figura 5.37: Balanced double homodyne detection.

Noise sources in homodyne detection

The detection of light using photodiodes is subjected to various sources of noise. One of these sources is the electrical field itself. The interaction of the signal with the vacuum field adds quantum noise to the detection. Another source of noise comes from the detection system, such as photodiodes and other electrical circuits, originating various kinds of noise, such as thermal noise, dark noise and amplifier noise [?]. In the following sections, we will focus on two noise sources, quantum noise and thermal noise.

Quantum Noise

In order to grasp this effect, the quantum mechanical description of balanced homodyne detection will be used, employing quantum operators to describe the effect of each component in the system (fig. ??). We start with the operators \hat{a}_S and \hat{a}_{LO} corresponding to the annihilation operator for the signal and local oscillator, which are the inputs in a beam divisor. The outputs will be \hat{a}_3 and \hat{a}_4 . Using a balanced beam splitter, we can write the output as

$$\hat{a}_3 = \frac{1}{\sqrt{2}} (\hat{a}_S + \hat{a}_{LO}) \quad (5.76), \quad \hat{a}_4 = \frac{1}{\sqrt{2}} (\hat{a}_S - \hat{a}_{LO}) \quad (5.77)$$

The final output of a homodyne measurement will be proportional to the difference between the photocurrents in arm 3 and 4. Then

$$I_{34} = I_3 - I_4 \sim \langle \hat{n}_3 - \hat{n}_4 \rangle \quad (5.78)$$

We can define an operator that describes the difference of number of photons in arm 3 and arm 4:

$$\hat{m} = \hat{a}_3^\dagger \hat{a}_3 - \hat{a}_4^\dagger \hat{a}_4 \quad (5.79)$$

If we assume that the local oscillator produces the the coherent state $|\beta\rangle$, then the expected value of this measurement will be

$$\langle m \rangle = 2|\alpha||\beta| \cos(\theta_\alpha - \theta_\beta) \quad (5.80), \quad \text{Var}(m) = |\alpha|^2 + |\beta|^2 \quad (5.81)$$

The local oscillator normally has a greater power than the signal , then $|\alpha| \ll |\beta|$. If we use as unit, $2|\beta|$, then these two quantities can be simplified to

$$\langle m \rangle = |\alpha| \cos(\theta_\alpha - \theta_\beta) \quad (5.82), \quad \text{Var}(m) \approx \frac{1}{4} \quad (5.83)$$

[?]

Has we have seen previously, in order to measure two quadratures simultaneously, we can use double balanced homodyne detection. For each quadrature, the input signal now has half the power, so $|\alpha| \rightarrow |\alpha/\sqrt{2}|$. If we use a local oscillator that produces states $|\beta\rangle$, then we can divide it in two beams in state $|\beta/\sqrt{2}\rangle$ and $|i\beta/\sqrt{2}\rangle$ which will be used in each homodyne detection. In this setting, the expected values for each quadrature, X and Y , (in normalized values of $\sqrt{2}|\beta|$) are

$$\langle m_X \rangle = \left| \frac{\alpha}{\sqrt{2}} \right| \cos(\theta_\alpha - \theta_\beta) \quad (5.84), \quad \text{Var}(m_X) \approx \frac{1}{4} \quad (5.85)$$

$$\langle m_Y \rangle = \left| \frac{\alpha}{\sqrt{2}} \right| \sin(\theta_\alpha - \theta_\beta) \quad (5.86), \quad \text{Var}(m_Y) \approx \frac{1}{4} \quad (5.87)$$

Therefore the measurement of each quadrature will have half the amplitude, but the same variance.

5.6.2 Simulation Analysis

Contributors	:	Diamantino Silva, (2017-08-18 - ...)
Goal	:	Simulation of various optical detection schemes.
Directory	:	sdf/optical_detection

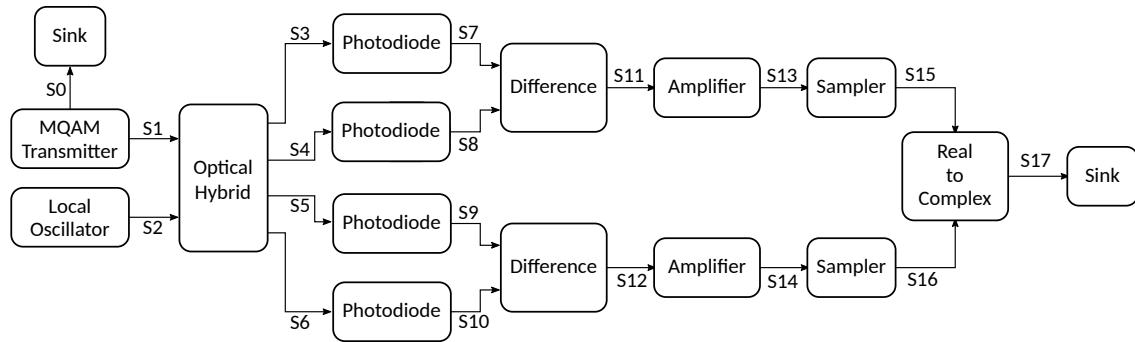


Figura 5.38: Overview of the simulated optical system.

List of signals used in the simulation:

Signal name	Signal type	Status
S0	Binary	check
S1	OpticalSignal	check
S2	OpticalSignal	check
S3	OpticalSignal	check
S4	OpticalSignal	check
S5	OpticalSignal	check
S6	OpticalSignal	check
S7	TimeContinuousAmplitudeContinuousReal	check
S8	TimeContinuousAmplitudeContinuousReal	check
S9	TimeContinuousAmplitudeContinuousReal	check
S10	TimeContinuousAmplitudeContinuousReal	check
S11	TimeContinuousAmplitudeContinuousReal	check
S12	TimeContinuousAmplitudeContinuousReal	check
S13	TimeContinuousAmplitudeContinuousReal	check
S14	TimeContinuousAmplitudeContinuousReal	check
S15	TimeDiscreteAmplitudeContinuousReal	check
S16	TimeDiscreteAmplitudeContinuousReal	check
S17	OpticalSignal	check

This system takes into account the following input parameters:

System Parameters	Default value	Description
localOscillatorPower1	2.0505×10^{-8} W	Sets the optical power, in units of W, of the local oscillator inside the MQAM
localOscillatorPower2	2.0505×10^{-8} W	Sets the optical power, in units of W, of the local oscillator used for Bob's measurements
localOscillatorPhase	0 rad	Sets the initial phase of the local oscillator used in the detection
responsivity	1 A/W	Sets the responsivity of the photodiodes used in the homodyne detectors
iqAmplitudeValues	$\{\{1, 1\}, \{-1, 1\}, \{-1, -1\}, \{1, -1\}\}$	Sets the amplitude of the states used in the MQAM

The simulation setup is represented in figure 5.38. The starting point is the MQAM, which generates random states from the constellation given by the variable iqAmplitudeValues. The output from the generator is received in the Optical Hybrid where it is mixed with a local oscillator, outputting two optical signal pairs. Each pair is converted to currents by two photodiodes, and the same currents are subtracted from each other, originating another current proportional to one of the quadratures of the input state. The other pair suffers the same process, but the resulting subtraction current will be proportional to another quadrature, dephased by $\pi/2$ relative to the other quadrature.

Required files

Header Files

File	Description	Status
netxpto.h	Generic purpose simulator definitions.	check
m_qam_transmitter.h	Outputs a QPSK modulated optical signal.	check
local_oscillator.h	Generates continuous coherent signal.	check
optical_hybrid.h	Mixes the two input signals into four outputs.	check
photodiode.h	Converts an optical signal to a current.	check
difference.h	Ouputs the difference between two input signals.	check
ideal_amplifier.h	Performs a perfect amplification of the input sinal	check
sampler.h	Samples the input signal.	check
real_to_complex.h	Combines two real input signals into a complex signal	check
sink.h	Closes any unused signals.	check

Source Files

File	Description	Status
netxpto.cpp	Generic purpose simulator definitions.	check
m_qam_transmitter.cpp	Outputs a QPSK modulated optical signal.	check
local_oscillator.cpp	Generates continuous coherent signal.	check
optical_hybrid.cpp	Mixes the two input signals into four outputs.	check
photodiode.h	Converts an optical signal to a current.	check
difference.h	Ouputs the difference between two input signals.	check
ideal_amplifier.h	Performs a perfect amplification of the input sinal	check
sampler.cpp	Samples the input signal.	check
real_to_complex.cpp	Combines two real input signals into a complex signal	check
sink.cpp	Empties the signal buffer.	check

Simulation Results

To test the simulated implementation, a series of states $\{|\phi_i\rangle\}$ were generated and detected, resulting in a series of measurements $\{(x_i, y_i)\}$. The simulation result is presented in figure 5.39:

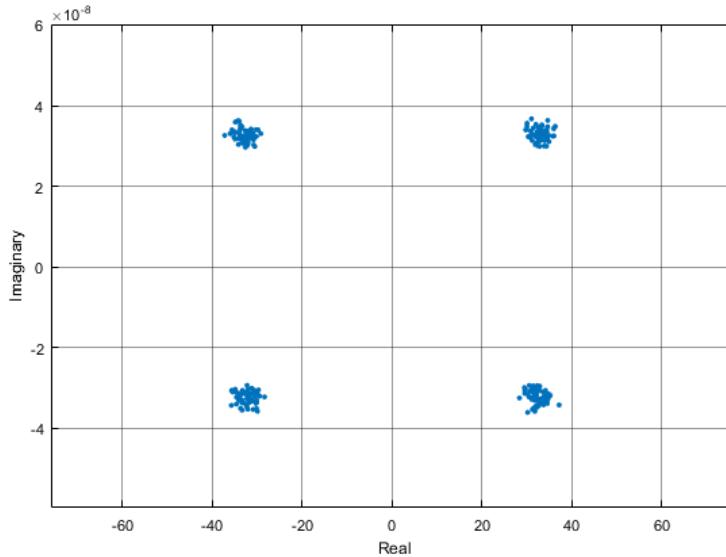


Figura 5.39: Simulation of a constelation of 4 states ($n = 100$)

We see that the measurements made groups in certain regions. Each of this groups is centered in the expected value $(\langle X \rangle, \langle Y \rangle)$ of one the generated states. Also, they show some variance, which was tested for various expected number of photons, $\langle n \rangle$, resulting in figure 5.40:

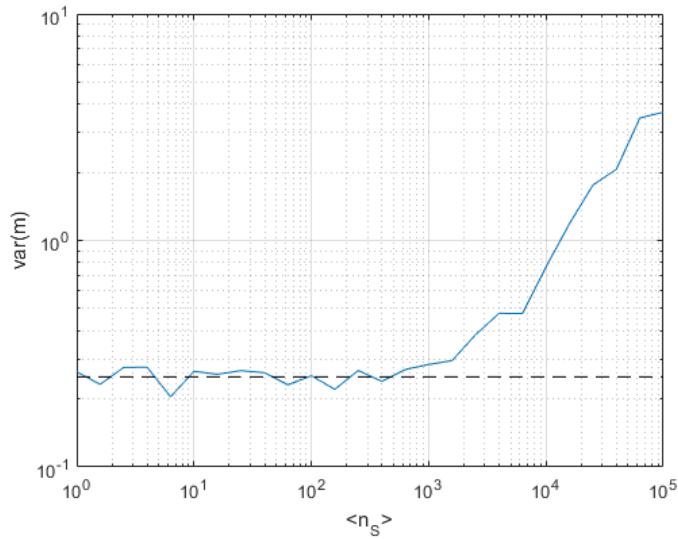


Figura 5.40: Simulation of the variance of m .
Local oscillator expected number of photons: 10^4

It was expected that the variance should independent of the input's signal number of photons. Plot 5.40 shows that for low values of n_S , the simulation is in accordance with the theoretical prevision, with $\text{Var}(X) = \text{Var}(Y) = \frac{1}{4}$. For large values of n_S , when the number of photons is about the same has the local oscillator, the quantum noise variance starts to grow proportionally to n_S , in accordance with the non approximated calculation of quantum noise (eq. ??).

Noise Variance with LO power Simulation

The following plot shows the behavior of current noise variance $\langle(\Delta i)^2\rangle$ with local oscilator power, P_{LO} :

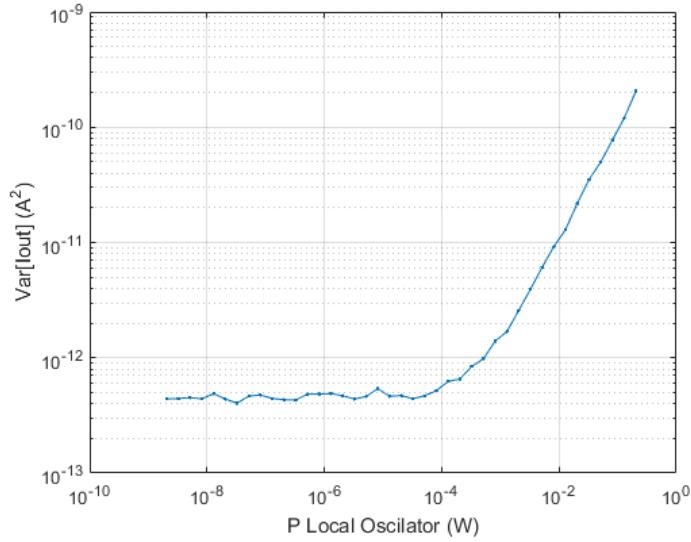


Figura 5.41: Output current variance in function of LO power.

We see that for low LO power, the dominant noise is the thermal contribution, but for higher power, quantum noise dominates, growing proportionally to P_{LO} . This in accordance with equation 5.82

5.6.3 Experimental Analysis

To confirm the theoretical results obtained in section 5.6.1, two experimental setups will be created. In the first experiment, we will study quantum noise in the single homodyne detection. The experimental setup will be based on the paper [?]. In the second experiment, we will study quantum noise in the double homodyne detection setup, which will be basically an extension of the single homodyne detection setup.

Single homodyne detection

To keep the experiment simple and avoid extra sources of noise, we will avoid using black boxes which have complicated inner workings, having a preference in using simple components, as shown in fig. 5.42:

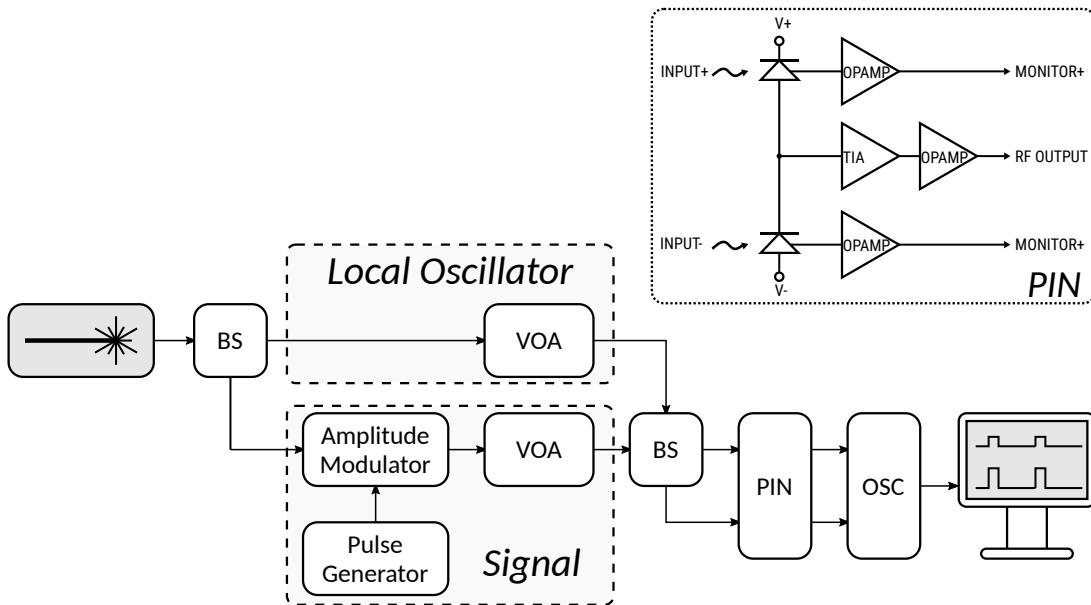


Figura 5.42: Experimental setup

Material list

Device	Description
Local Oscillator	Yenista OSICS Band C/AG
BS	Beam Splitter
Pulse Generator	HP 8116A Pulse Generator
Amplitude Modulator	Mach Zehnder SDL OC 48
VOA	Eigenlicht Power Meter 420
VOA	Thorlabs VOA 45-APC
PIN	Thorlabs PDB 450C
ADC	Picoscope 6403D

A single laser is splitted and used as the source for the signal (S) and the local oscillator (LO). The signal beam is pulsed and highly attenuated. The local oscillator is also attenuated, but not pulsed. The signal and local oscillator interfere in a Beam Splitter originating two beams which are then converted to voltages in the PIN. These voltages are read in the Digital Oscilator (OSC) and collect in the computer. In the post processing phase, the quantum noise is measured by applying a difference between the two beams and measuring it's variance.

The second stage of the experiment will be very similar to the first one, in which the signal and local oscillator branches will be divided. One of the new branches of the local oscillator will suffer a phase delay of $\pi/2$, in order to measure the quadrature component of the incoming signal.

Thorlabs detector

The detector used in the laboratory is the Thorlabs PDB 450C. This detector consists of two well-matched photodiodes and a transimpedance amplifier that generates an output voltage (RF OUTPUT) proportional to the difference between the photocurrents of the photodiodes. Additionally, the unit has two monitor outputs(MONITOR+ and MONITOR-) to observe the optical input power level on each photodiode separately. [?]

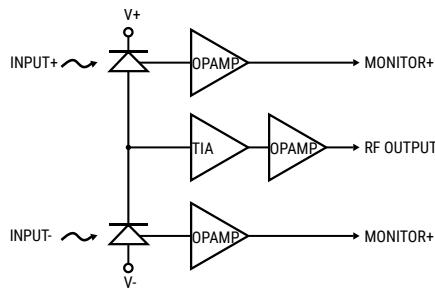


Figura 5.43: Thorlabs PDB 450 detection circuit.

The transimpedance amplifier (TIA) is an opamp-based current to voltage conversion amplifier.¹ It is used in this circuit, given that the photodiode's current has a linear response to incident light in contrast to it's non-linear voltage response.²

We have seen that the theoretical results obtained in section ?? all depend on various parameters, such as amplitude, bandwidth, amplification, frequency response and noise. Therefore, we need to characterize our photodetector in order that our experimental results can show that our theoretical results are indeed correct.

Table 5.7 lists various important characteristics to perform our measurements. These values were obtained from the device's manual [?]

Characteristic	Value
Bandwidth (monitor)	DC to 1 MHz
Voltage Gain (monitor)	10 V/mW @ peak responsivity
Voltage Noise (monitor)	<180 µV (RMS)
Max Responsivity (PIN)	1.0 A/W
RF OUTPUT Bandwidth(-3dB)	DC to 150 / 45 / 4 / 0.3 / 0.1 MHz
RF OUTPUT Transimpedance Gain	10^3 / 10^4 / 10^5 / 10^6 / 10^7 V/A
RF OUTPUT Conversion Gain	10^3 / 10^4 / 10^5 / 10^6 / 10^7 V/W

Tabela 5.7: Main characteristics of the Thorlabs PDB450C balanced detector

¹<http://www.cypress.com/file/131966/download>

²<http://www.ti.com/lit/an/sboa035/sboa035.pdf>, p.1

Gain response

The gain response is obtained by fixing an frequency ω_0 and generating optical signals with various amplitudes A_i , such as $V = A_i \sin(\omega_0 t)$ and reading the output voltage amplitude.³

Frequency response/Bandwidth

The frequency response is obtained by fixing an amplitude A_0 and generating optical signals with various frequencies ω_i , such as $V = A_0 \sin(\omega_i t)$ and reading the output voltage amplitude.⁴

The following plot shows the gain for various frequencies???

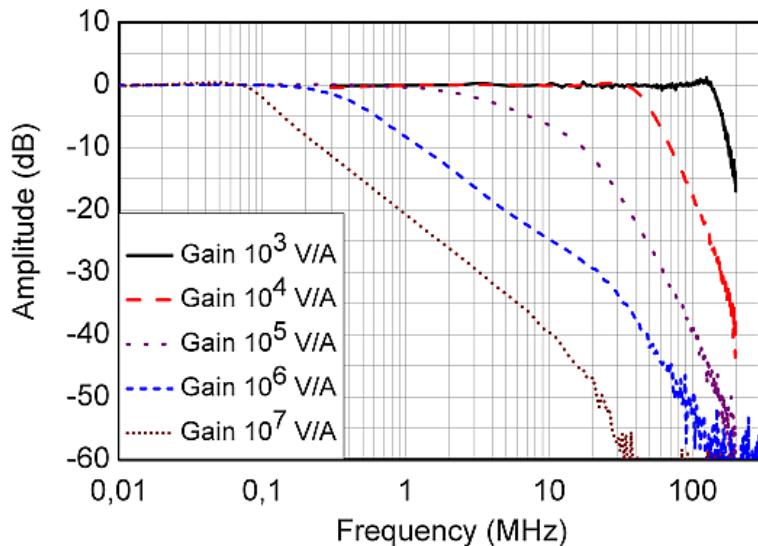


Figura 5.44: Thorlabs PDB 450 RF output amplitude with frequency, for various gain values. (Plots from the manual)

Data analysis

Using the data collected in the Digital Oscilloscope, we process each sample for each power level (which???). This data analysis will consist in the following steps: 1) Subtract the data coming from MONITOR- from the data from MONITOR+.

2) Calculate the average impulse and the variance of the impulse. (VALE A PENA ESPECIFICAR A EQUIVALÃŠNCIA DE FASE?)

3) The representative variance will be the variance of the pulse maximum.

For each combination of the expected number of photons present in the signal and the local oscillator, we will obtain the variance in the plateau of the pulse. The final variance is simply the mean of the variances of all pulses.

³REF?

⁴REF?

Double homodyne detection

The experimental setup for this experiment is a simple extension of the single homodyne detection. To implement this experiment, the signal beam will be divided in two beams, in which one of them will be detected in phase with local oscillator and the other in phase with a oscillator with a $\pi/2$ phase relative to the first one.

5.6.4 Comparative analysis

Given the theoretical, simulated and experimental frameworks, we will now compare the results obtained by each of them.

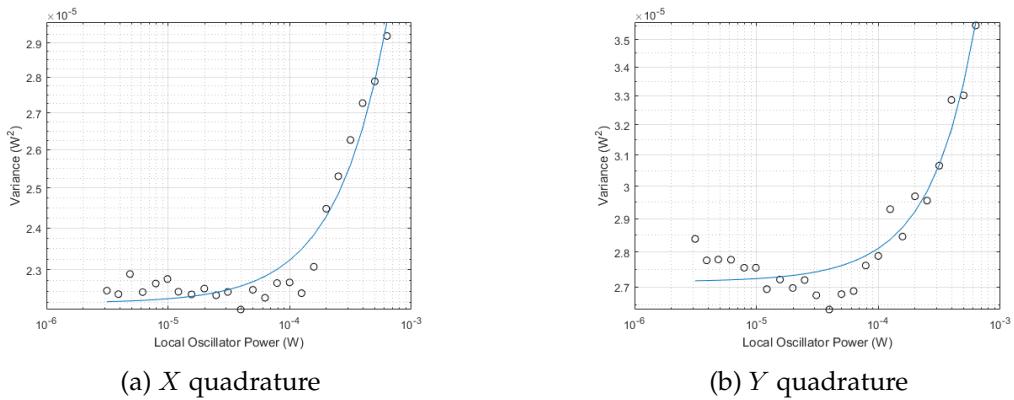


Figura 5.45: Noise variance dependency with local oscillator power for two different quadratures. Experimental vs fitted data.

Figures 5.45a and 5.45b show measurements of total noise for two different quadratures. For low power of LO, the noise variance fluctuates around a constant value. For high power of LO, ($P_{LO} > 10^{-4}W$), the variance of noise shows an increasing trend roughly proportional to P_{LO}^2 . The polynomial fittings confirm this trend, showing a degree 2 coefficient much larger than the degree 1 coefficient

$$\text{Var}_X = 2.22 \times 10^{-5} + 9.6 \times 10^{-3} P_{LO} + 3.40 P_{LO}^2 \quad (5.88)$$

$$\text{Var}_Y = 2.71 \times 10^{-5} + 8.9 \times 10^{-3} P_{LO} + 7.25 P_{LO}^2 \quad (5.89)$$

The expected growth should be proportional to P_{LO} , but the RIN noise, originated by the electric apparatus, which grows quadratically with the power, is dominating the noise amplitude for large P_{LO} .

We see that both the simulation and experimental data display a similar behaviour, but the quadratic growth of noise for large P_{LO} was not predicted in the simulations.

5.6.5 Known problems

5.7 Continuous Variable QKD Transmission System

Student Name	: Daniel Pereira (2017/05/01 -)
Goal	: Simulation and experimental validation of a CV-QKD transmission system.
Directory	: sdf/cv_system

The aim of Continuous Variable Quantum Key Distribution (CV-QKD) is to encode information in observables whose measurements take continuous values.

The purpose of this study is to analyse a CV-QKD transmission system in which the information is sent in the two orthogonal quadratures of a coherent state.

5.7.1 Theoretical Analysis

In this section we describe in depth the Continuous Variables Quantum Key Distribution (**CV-QKD**) system analysed here. The CV-QKD system enables two parties (Alice and Bob) to share a secret key to be employed in a symmetric encryption protocol. The following CV-QKD employs discrete modulation of coherent states, resulting in the Quadrature Phase Shift Keying (**QPSK**) constellation presented in Figure 5.46. The state to bit correspondence,

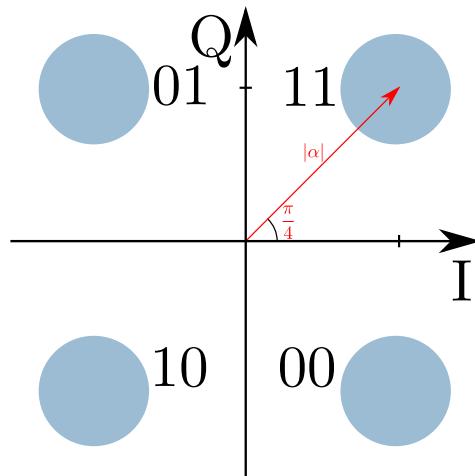


Figura 5.46: QPSK symbol constellation.

agreed by Alice and Bob beforehand, also presented in Figure 5.46, is as follows:

State	Bit
$ \alpha_0\rangle = \alpha e^{i\frac{\pi}{4}}\rangle$	11
$ \alpha_1\rangle = \alpha e^{i\frac{3\pi}{4}}\rangle$	01
$ \alpha_2\rangle = \alpha e^{i\frac{5\pi}{4}}\rangle$	10
$ \alpha_3\rangle = \alpha e^{i\frac{7\pi}{4}}\rangle$	00

A step by step description of the protocol now follows:

1. Alice generates a random bit string of length $2N$. Lets assume that $N = 8$ and that the following random bit string of length $2N = 16$ is generated:

$$B_A = \{1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0\},$$

2. Next, Alice sends to Bob a sequence of N quantum states based on the bit string B_A , according to the encoding presented before, resulting in the following state sequence:

$$S_A = \{|\alpha_0\rangle, |\alpha_1\rangle, |\alpha_2\rangle, |\alpha_1\rangle, |\alpha_0\rangle, |\alpha_1\rangle, |\alpha_3\rangle, |\alpha_2\rangle\}.$$

3. Bob measures both quadratures of the received states and obtains two results I_i and Q_i , $i \in 1, \dots, N$, corresponding to the in-phase and in-quadrature components, respectively, of the i -th coherent state. Bob's results b are related to Alice's originally encoded states a by:

$$b = ta + z,$$

which are normal linear models parametrized by $t = \sqrt{\frac{T}{2}}$ and where z are the total noise contribution following a normal distribution with null mean and variance σ^2 . Alice's sent states a are a Bernoulli random variable taking the values $\pm\sqrt{2}|\alpha|$.

4. Bob shares a subset of k measurement results with Alice, who uses the previous linear relations to estimate the transmission channel parameters as follows:

$$\tilde{t} = \frac{\sum_{i=1}^k b_i a_i}{\sum_{i=1}^k a_i^2}, \quad \tilde{\sigma}^2 = \frac{1}{k} \sum_{i=1}^k (b_i - \tilde{t} a_i)^2$$

These values are used to evaluate the key's security. If it is deemed not to be secure, the protocol is aborted, otherwise it continues as follows.

5. Bob attributes each measurement result to the closest corresponding possible state sent Alice, taking the considered QPSK constellation this corresponds to:

Measurement	Decoded state
$I_i > 0 \& Q_i > 0$	$ \alpha_0\rangle$
$I_i < 0 \& Q_i > 0$	$ \alpha_1\rangle$
$I_i < 0 \& Q_i < 0$	$ \alpha_2\rangle$
$I_i > 0 \& Q_i < 0$	$ \alpha_3\rangle$

and builds a bit string according to both the measurement results and the previously agreed upon state to bit correspondence. Lets assume that Bob builds the following decoded state sequence:

$$S_B = \{|\alpha_0\rangle, |\alpha_0\rangle, |\alpha_2\rangle, |\alpha_1\rangle, |\alpha_0\rangle, |\alpha_2\rangle, |\alpha_2\rangle, |\alpha_2\rangle\},$$

with the corresponding bit string:

$$B_B = \{1, 1, \underline{1}, 1, 1, 0, 0, 1, 1, \underline{1}, \underline{0}, \underline{1}, 0, 1, 0\}.$$

The mistakes in both the bit string and state sequence are underlined.

5.7.2 Simulation Analysis

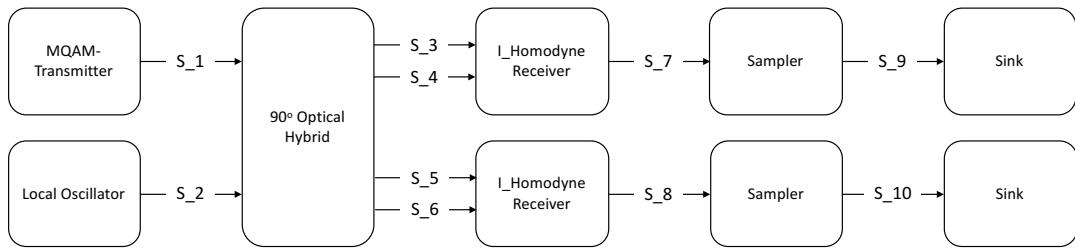


Figura 5.47: Overview of the CV-QKD system being simulated.

Required files

Header Files		
File	Comments	Status
add.h		✓
binary_source.h		✓
discrete_to_continuous_time.h		✓
i_homodyne_reciever.h		✓
ideal_amplifier.h		✓
iq_modulator.h		✓
local_oscillator.h		✓
m_qam_mapper.h		✓
m_qam_transmitter.h		✓
netxpto.h		✓
optical_hybrid.h		✓
photodiode.h		✓
pulse_shaper.h		✓
sampler.h		✓
sink.h		✓
super_block_interface.h		✓
white_noise.h		✓

Source Files		
File	Comments	Status
add.cpp		✓
binary_source.cpp		✓
discrete_to_continuous_time.cpp		✓
i_homodyne_reciever.cpp		✓
ideal_amplifier.cpp		✓
iq_modulator.cpp		✓
local_oscillator.cpp		✓
m_qam_mapper.cpp		✓
m_qam_transmitter.cpp		✓
netxpto.cpp		✓
optical_hybrid.cpp		✓
photodiode.cpp		✓
pulse_shaper.cpp		✓
sampler.cpp		✓
sink.cpp		✓
super_block_interface.cpp		✓
white_noise.cpp		✓

System Input Parameters

This system takes into account the following input parameters:

System Input Parameters		
Parameter	Default Value	Comments
numberOfBitsGenerated	40000	
bitPeriod	20×10^{-12}	
samplesPerSymbol	16	
pLength	5	
iqAmplitudesValues	{ {−1, 0}, {1, 0} }	
outOpticalPower_dBm	Variable	Value varied for presented study
loOutOpticalPower_dBm	0	
localOscillatorPhase	0	
transferMatrix	{ { $\frac{1}{\sqrt{2}}$, $\frac{1}{\sqrt{2}}$, $\frac{1}{\sqrt{2}}$, $\frac{-1}{\sqrt{2}}$ } }	
responsivity	1	
amplification	10^3	
noiseSpectralDensity	$5 \times 10^{-4} \sqrt{2} \text{ V}^2$	
confidence	0.95	
midReportSize	0	

Inputs

This system takes no inputs.

Outputs

This system outputs the following objects:

Parameter: Signals:

Description: Optical Signal with coded Binary String; (S_1)

Description: Local Oscillator Optical Signal; (S_2)

Description: 90° Optical Hybrid Outputs; (S_3, S_4, S_5, S_6)

Description: Homodyne Detectors' Electrical Output; (S_7, S_8)

Description: Sampled Signals; (S_9, S_{10})

5.7.3 Experimental Analysis

The main experimental setup used is presented in Figure 5.48.

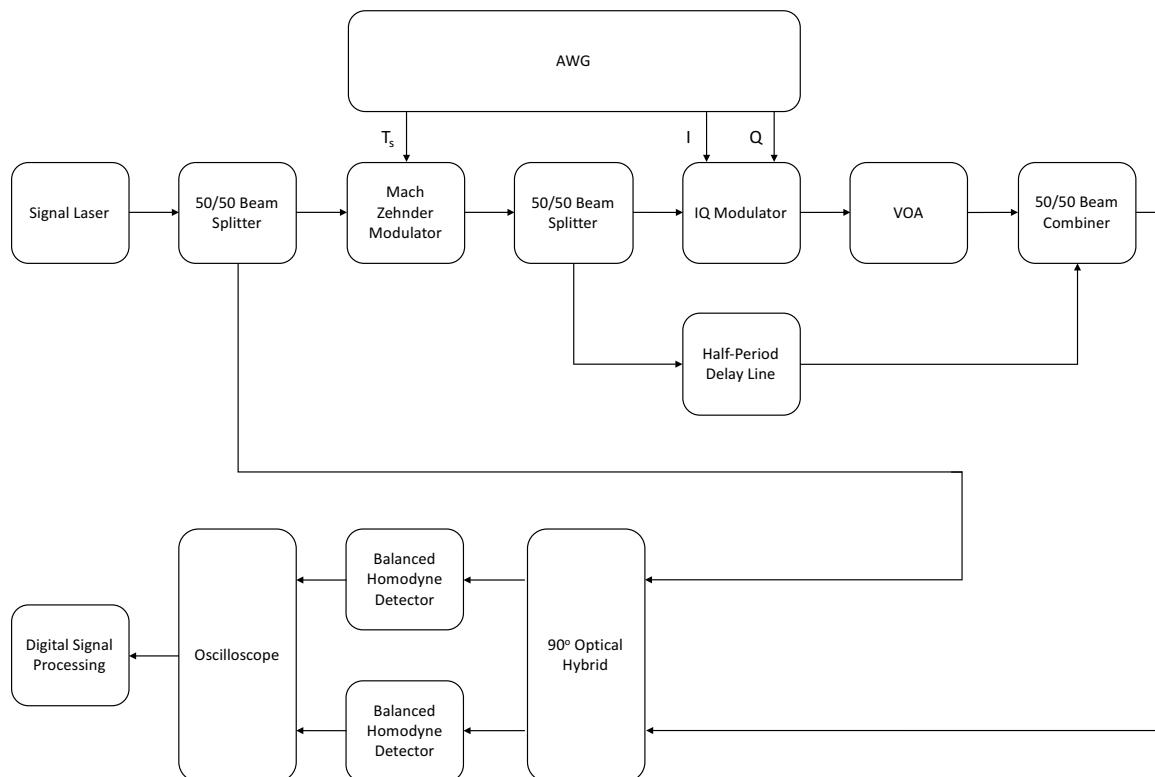


Figura 5.48: Diagram of full experimental setup.

The setup contained one Yenista OSICS Band C/AG TLS lasers, tuned to a 1550 nm wavelength, followed by a 50/50 beam-splitter, generating two different optical signals. A Mach-Zehnder Modulator, employed at one of the outputs of the beam-splitter and acting as an amplitude modulator, set the repetition rate at 1 MHz with a pulse time of 100 ns. This amplitude modulated light field is dubbed the Signal (**SI**) laser. The driving signal was generated by an Agilent Technologies BER Tester. A 50/50 beam-splitter was employed at the output of the Mach-Zehnder Modulator, one arm output is sent through an IQ Modulator while the other is sent through a fibre loop with length chosen such that the two arms have a relative delay of roughly 500 ns. The employed IQ Modulator was a u2t Photonics 32 GHz IQ Modulator, the driving signal being generated by a Tektronix AWG70002A Arbitrary Waveform Generator (**AWG**). A 15 dB attenuator followed by a Variable Optical Attenuator (**VOA**) was set at the output of the IQ modulator to allow a fine tuning of the phase modulated signal's optical power to the desired level. The two arms created by the first beam-splitter are combined by a 50/50 beam combiner.

The second output of the first beam-splitter in Figure 5.48 was sent through a VOA to a Kylia 90° Optical Hybrid, where it is mixed with the SI laser before being sent to two Thorlabs PDB450C Balanced Homodyne Detectors, set to an amplification factor of 10^5 , corresponding to a bandwidth of 4 MHz. This detection setup recovers both the in-phase and in-quadrature components of the incoming light field. The response of the receiver was recorded by a Tektronix DPO77002SX-R3 oscilloscope, with an acquisition frequency of 100 GHz for a period of 400 μ s.

The driving signal implemented on the AWG was generated by a short Matlab code that generated a Pseudo Random Bit Sequence (**PRBS**) with length 2^{17} preceded by a deterministic tram of length 32896 taking the form 1001110000(...).

Output Data Processing

A Matlab algorithm was developed to allow for the post-processing of the recorded experimental data. The code works as follows:

1. Loads the data into the Matlab workspace.
2. Generates a persistence figure isolating every information-reference pulse pair for easier post-processing (see Figure 5.49a), with the reference and signal pulses being visible, in the time ranges $\sim [0.1, 0.2] \times 10^{-9}$ s and $\sim [0.7, 0.8] \times 10^{-9}$ s respectively.
3. Removes low frequency noise by computing the average voltage value on an area outside the pulses for each persistence pair and subtracting that value from the full pair. The effect of this noise removal is noticeable in Figure 5.49b, notice the noise amplitude in the time range $\sim [0.3, 0.6] \times 10^{-9}$ s.
4. Selects the points inside the pulses, the following steps are performed only on these selected points.

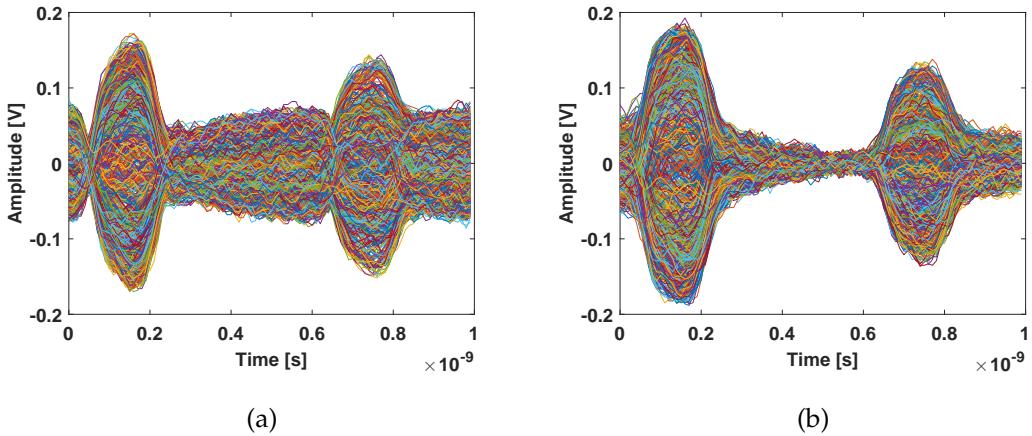


Figura 5.49: Persistence of 1000 pairs of information and reference pulses before (a) and after (b) low frequency noise removal.

5. Orthogonalizes the data by implementing a recursive Gram-Schmidt orthogonalization algorithm, adapted from the method presented in [2]. The correlation r_{IQ} between the in-phase and in-quadrature points of the reference pulses is calculated, this correlation is then used to weigh the amount of the in-phase contribution that is subtracted from the in-quadrature element for both the signal and reference pulses. For $i \geq 2$ this algorithm takes the form:

$$\begin{cases} r_{IQ}(i) = r_{IQ}(i-1) + p(\mathbf{Q}_r(i-1)\mathbf{I}_r(i-1) - r_{IQ}(i-1)) \\ \mathbf{Q}_{ro}(i) = \mathbf{Q}_r(i) - r_{IQ}(i)\mathbf{I}_r(i) \\ \mathbf{Q}_{so}(i) = \mathbf{Q}_s(i) - r_{IQ}(i)\mathbf{I}_s(i) \end{cases}, \quad (5.90)$$

with I_r/I_s being the in-phase components of the reference/signal pulses, which are maintained, Q_r/Q_s are the initial in-quadrature components of the reference/signal pulses, while Q_{ro}/Q_{so} are the in-quadrature components of the reference/signal pulses after removal of the in-phase contribution. For $i = 1$, r_{IQ} was set at 0 and p is a small number that we have chosen to be 10^{-3} . The effect of this orthogonalization procedure is presented graphically in Figure 5.50. This algorithm is applied on

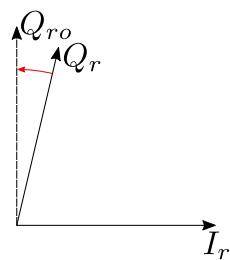


Figura 5.50: Visual representation of the employed axis orthogonalization.

the reference pulses. Before this step the constellation consists of an ellipse with a

small eccentricity, while after application of the algorithm the constellation becomes a circle, removing the detector imbalances. This algorithm is implemented only on the reference pulses because of the extremely low Signal to Noise Ratio (**SNR**) of the signal pulses while working at quantum levels. This algorithm has a short convergence time, rendering the first bits (numbering roughly 10^4) unusable.

6. Removes the phase difference between the signal and local oscillator lasers, see Figure 5.51, where it is clear that the constellation presented in Figure 5.51a is a dragging rotation of the constellation in Figure 5.51b. The compensation of the phase difference between the signal and local oscillator is accomplished by measuring the phase of the reference pulses in relation to the local oscillator, this phase difference is then subtracted from the phase measured from the signal pulses. This is accomplished by multiplying the signal constellation with the conjugate of the normalized reference constellation. It is visible that the constellation before the phase drift compensation is a dragging rotation of the QPSK constellation obtained afterwards.

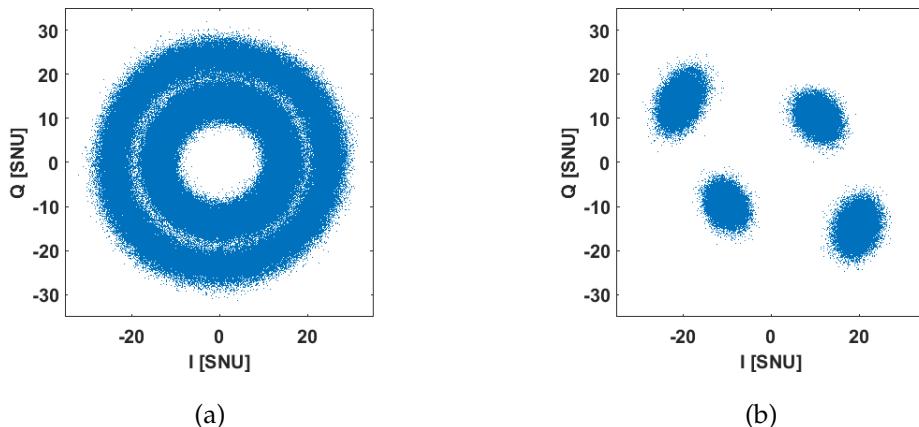


Figura 5.51: Signal constellation before (a) and after (b) phase difference compensation.

7. Ranges the start of the PRBS sequence via detection of the deterministic tram mentioned above.
8. Corrects rotations of the constellation via minimization of the error of the deterministic tram. Given the laser's central frequency fluctuations, this rotation should be compensated at regular intervals during transmission.
9. Converts the signal constellation to shot noise units.
10. Estimates the secret key rate obtained from the recovered bit string.

Figure 5.52 presents the recovered constellations for single and double laser setups at high levels of signal power (~ 200 photons per pulse on average). The expected output would be a *square* four state constellation, while the observed results show a clear skew on the recovered constellation. We conclude this is due to imbalances on the modulation

stage, this is the reason why this imbalance appears in both the single and double laser schemes. These imbalances are to be expected and in classical communications they would be compensated by applying an orthogonalization method on the data, this is not an option for our setup because at quantum levels the signal to noise ratio is so low that such methods are not efficient.

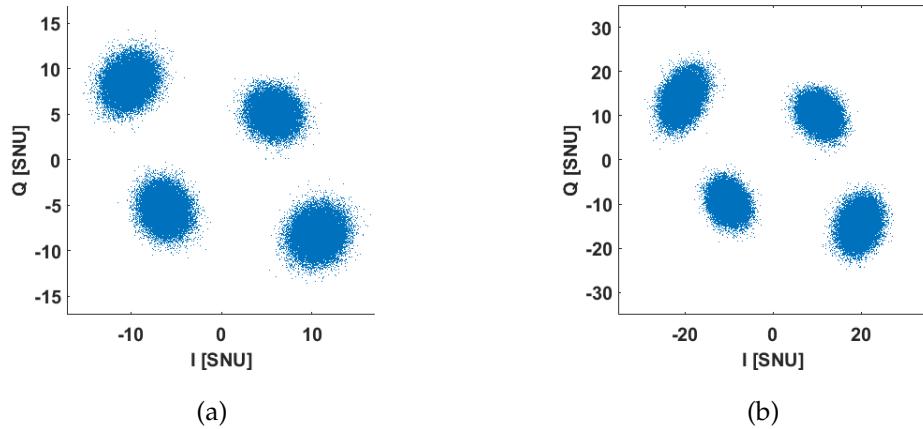


Figura 5.52: Final constellation for single (a) and double (b) laser setups.

5.7.4 Comparative Analysis

Bibliografia

- [1] Thorlabs. *Thorlabs Balance Amplified Photodetectors: PDB4xx Series Operation Manual*, 2014.
- [2] Xinping Huang, Mario Caron, and Dan Hindson. A recursive Gram-Schmidt orthonormalization procedure and its application to communications. In *Wireless Communications, 2001.(SPAWC'01). 2001 IEEE Third Workshop on Signal Processing Advances in*, pages 340–343. IEEE, 2001.
- [3] Tie-Ming Liu, Lie-hui Jiang, Hong-qi He, Ji-zhong Li, and Xian Yu. *Researching on Cryptographic Algorithm Recognition Based on Static Characteristic-Code*, pages 140–147. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [4] Gilles Brassard and Louis Salvail. *Secret-Key Reconciliation by Public Discussion*, pages 410–423. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.
- [5] Min Xu, Run-hua Shi, Zhen-yu Luo, and Zhen-wan Peng. Nearest private query based on quantum oblivious key distribution. *Quantum Information Processing*, 16(12):286, Oct 2017.
- [6] Álvaro J Almeida, Nelson J Muga, Nuno A Silva, João M Prata, Paulo S André, and Armando N Pinto. Continuous control of random polarization rotations for quantum communications. *Journal of Lightwave Technology*, 34(16):3914–3922, 2016.

5.8 BB84 with Discrete Variables

Students Name	:	Mariana Ramos and Kevin Filipe
Starting Date	:	November 7, 2017
Goal	:	BB84 implementation with discrete variables.

BB84 is a key distribution protocol which involves three parties, Alice, Bob and Eve. Alice and Bob exchange information between each other by using a quantum channel and a classical channel. The main goal is continuously build keys only known by Alice and Bob, and guarantee that eavesdropper, Eve, does not gain any information about the keys.

5.8.1 Theoretical Description

BB84 protocol was created by Charles Bennett and Gilles Brassard in 1984 [?]. This was the first created Quantum Key Distribution (QKD) protocol. A basic model is depicted in figure 5.53. It involves two parties sharing keys through a quantum channel to decipher the classical channel data. It is assumed that there is a eavesdropper, Eve to intercept the data.

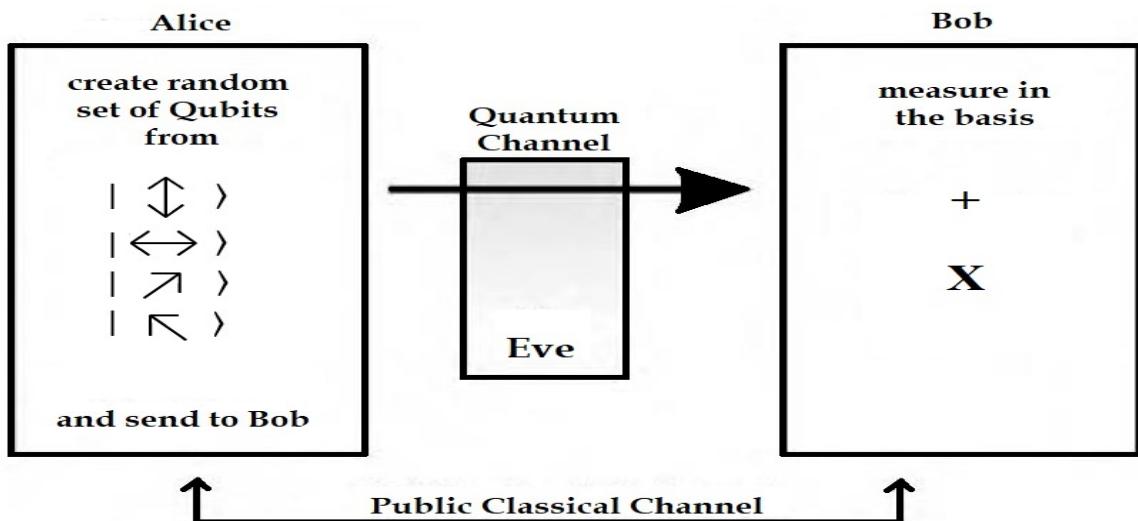


Figura 5.53: Basic QKD Model. Alice and Bob are connected by 2 communication channels, quantum and classical, with an eavesdropper, Eve, in the quantum communication channel. [?]

BB84 protocol uses bit encoding into photon state polarization. Two non-orthogonal basis are used to encoded the information, the rectilinear and diagonal basis. The following table shows this bit encoding and figure 5.54 shows a simple representation of it.

	<i>Rectilinear Basis</i>	<i>Diagonal Basis</i>
0	0	-45
1	90	45

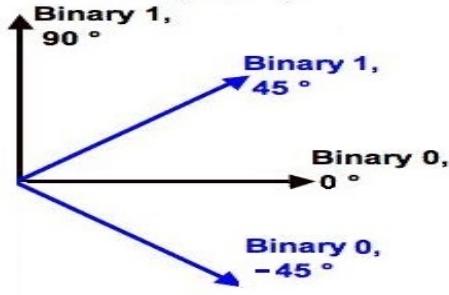


Figura 5.54: Simple representation of the bit encoding using the corresponding bases. [?]

The protocol is implemented with the following steps:

1. Alice generates two random bit strings. The random string R_{A1} , is the data to be encoded while R_{A2} is a string which 0 and 1 corresponds to rectilinear and diagonal basis B_A , respectively.

$$R_{A1} = \{0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1\}$$

$$R_{A2} = \{0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0\}$$

$$B_A = \{+, +, \times, +, \times, \times, \times, +, \times, \times, +, +, +, +, \times, +, \times, +\}$$

2. Alice transmits a train of photons, S_{AB} , in which, each photon represents a bit of the random string, R_{A1} , with the correspondent polarization regarding to the bit value.

$$S_{AB} = \{\rightarrow, \uparrow, \searrow, \rightarrow, \searrow, \nearrow, \nearrow, \uparrow, \searrow, \nearrow, \searrow, \uparrow, \searrow, \rightarrow, \rightarrow, \uparrow, \nearrow, \rightarrow, \nearrow, \uparrow\}.$$

3. Bob generates a random string, R_B , such as Alice, to receive the photon trains with the correspondent basis.

$$R_B = \{0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0\}$$

$$B_B = \{+, \times, \times, \times, +, \times, +, +, \times, \times, +, +, \times, +, +, \times, \times, +, +\}.$$

4. If Bob chooses a matched basis compared to the one encoded in the photon, then he can correctly deduce the right bit, otherwise deduced bit is randomly read. It is considered that the channel contains attenuation, this will not click the polarization detector

$$D_B = \{0, 1, 1, 0, -, 0, 1, 1, -, 0, 0, 1, -, 1, 0, 1, 0, -, 1, 1\}$$

Since less than half of the data is deduced, a second phase is needed so that Alice can announce to Bob which deduced bits are correct. This second phase, uses the classical communication channel:

1. Bob sends Alice about the no clicks, in which 1 represents a no click.

$$NC_{BA} = \{0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0\}.$$

2. Bob notifies Alice about what random basis he used to deduce each bit, R_B , and Alice also sends to Bob the basis R_{A1} , she used.
3. Both performs an XOR to deduce the correct sequence.

R_{A2}	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	1	0	1	0
R_B	0	1	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0
\oplus	0	1	0	1	1	0	1	0	0	0	1	0	0	1	0	0	0	1	1

4. By using the deduced bits and removing the no clicks, the final obtained sequence is K_{AB}

$$K_{AB} = \{0, 1, 0, 1, 0, 1, 0, 1, 0, 1\}.$$

The Quantum Bit Error Rate (QBER) is calculated by knowing the amount of no clicks and correct deduced bits by Bob.

$$QBER = \text{no_clicks_qty}/\text{correct_deduced_bits}$$

In the previous example, the QBER is 33%, since there are 4 no clicks.

The presence of a eavesdropper will carry the risk of changing the bits. This will produce disagreement between Bob and Alice in the bits they should agree. When Eve measures and retransmits a photon she can deduce correctly with a probability of 50%. So by learning the correct polarization of half of the photons, the induced error is 25%. Alice and Bob can detect Eve presence by sacrifice the secrecy of some bits in order to test.

Bibliografia

- [1] Bennett, C. H. and Brassard, G. Quantum Cryptography: Public key distribution and coin tossing. International Conference on Computers, Systems and Signal Processing, Bangalore, India, 10-12 December 1984, pp. 175-179.
- [2] Mart Haitjema, A Survey of the Prominent Quantum Key Distribution Protocols
- [3] Christopher Gerry, Peter Knight, "Introductory Quantum Optics" Cambridge University Press, 2005

Basis	
0	+
1	×

	Basis "+"
0	$\rightarrow (0^\circ)$
1	$\uparrow (90^\circ)$

	Basis "x"
0	$\searrow (-45^\circ)$
1	$\nearrow (45^\circ)$

1. Alice randomly generate a bit sequence with length ks being, in this case, $k = 2$ and $s = 4$ as it was defined at the beginning. Therefore, she must define two sets randomly: S_{A1} which contains the basis values; and S_{A2} , which contains the key values.

In that case, lets assume she gets the following sets S_{A1} and S_{A2} :

$$S_{A1} = \{0, 1, 1, 0, 0, 1, 0, 1\},$$

$$S_{A2} = \{1, 1, 0, 0, 0, 1, 0, 0\}.$$

2. Next, Alice sends to Bob throughout a quantum channel ks photons encrypted using the basis defined in S_{A1} and according to the keys defined in S_{A2} .

In the current example, Alice sends the photons, throughout a quantum channel, according to the following,

$$S_{AB} = \{\uparrow, \nearrow, \searrow, \rightarrow, \rightarrow, \nearrow, \rightarrow, \searrow\}.$$

$$S_{AB} = \{90^\circ, 45^\circ, -45^\circ, 0^\circ, 0^\circ, 45^\circ, 0^\circ, -45^\circ\}.$$

3. Bob also randomly generates ks bits, which are going to define his measurement basis, S_{B1} . He will measure the photons sent by Alice. Lets assume:

$$S_{B1} = \{0, 1, 0, 1, 0, 1, 1, 1\}.$$

When Bob receives photons from Alice, he measures them using the basis defined in S_{B1} . In the current example, S_{B1} corresponds to the following set:

$$\{+, \times, +, \times, +, \times, \times, \times\}.$$

Bob will get ks results:

$$S_{B1'} = \{1, 1, 0, 1, 0, 1, 1, 0\}.$$

4. Bob will send a *Hash Function* result HASH1 to Alice. This value will do Bob's commitment with the measurements done. In this case, this *Hash Function* is calculated from *SHA-256* algorithm for each pair (Basis from S_{B1} and measured value from $S_{B1'}$), i.e Bob sends to Alice sk pairs as his commitment. In this case, Bob sends eight pairs encoded using a *Hash Function* which is also send to Alice. From that moment on Bob cannot change his commitment neither the basis which he uses to measure the photons sent by Alice.
5. Once Alice has received the confirmation of measurement from Bob, she sends throughout a classical channel the basis which she has used to codify the photons, which in this case we assumed $S_{A1} = \{0, 1, 1, 0, 0, 1, 0, 1\}$.
6. In order to know which photons were measured correctly, Bob does the operation $S_{B2} = S_{B1} \oplus S_{A1}$. In the current example the operation will be:

$$\begin{array}{c|ccccccccc} S_{B1} & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ \hline S_{A1} & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ \hline \oplus & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}$$

In this way, Bob gets

$$S_{B2} = \{1, 1, 0, 0, 1, 1, 0, 1\}.$$

When Bob uses the right basis he gets the values correctly, when he uses the wrong basis he just guess the value. The values "1" correspond to the values he measured correctly and "0" to the values he just guessed.

Next, Bob sends to Alice, through a classical channel, information about the minimum number between "ones" and "zeros", i.e

$$n = \min(\#0, \#1) = 3,$$

where $\#0$ represents the number of zeros in S_{B2} and $\#1$ the number of ones in S_{B2} . At this time, Alice must be able to know if Bob is being honest or not. Therefore, she will open Bob's commitment from *step 4* and she verify if the number n sent by Bob is according with the commitment values sent by him. In other words, she opens a number of pairs committed by Bob which is known from the beginning.

7. If $n < s$, being s the message's size, Alice and Bob will repeat the steps from 1 to 7. In this case, $n = 3$ which is smaller than $s = 4$. Therefore, Alice and Bob repeat the steps from 1 to 7 in order to enlarge Bob's sets S_{B1} and S_{B2} as well as Alice's sets S_{A1} and S_{A2} .
8. Lets assume :

$$S_{B1} = \{1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1\}.$$

At Alice's side the new sets S_{A1} , which contains the basis values, and S_{A2} , which contains the key values, will be the following:

$$\begin{aligned} S_{A1} &= \{0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0\}, \\ S_{A2} &= \{1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1\}. \end{aligned}$$

Finally, for $S_{B2} = S_{B1} \oplus S_{A1}$ Bob gets the following sequence:

$$S_{B2} = \{1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1\}.$$

Note that the sets were enlarge in the second iteration.

9. At this time, Bob sends again to Alice, through a classical channel, the minimum number between "ones" and "zeros", $n = \min(\#0, \#1)$. In this case, n is equal to 7 which is the number of zeros.
10. Alice checks if $n > s$ and acknowledge to Bob that she already knows that $n > s$. In this case, $n = 7$ and $s = 4$ being $n > s$ a valid condition.
11. Next, Bob defines two new sub-sets, I_0 and I_1 . I_0 is a set of values with photons array positions which Bob just guessed the measurement since he did not measure them with the same basis as Alice, I_1 is a set of values with photons array positions which Bob measured correctly since he used the same basis as Alice used to encoded them.

In this example, Bob defines two sub-sets with size $s = 4$:

$$I_0 = \{3, 4, 7, 11\},$$

and

$$I_1 = \{2, 5, 6, 13\},$$

where I_0 is the sequence of positions in which Bob was wrong about basis measurement and I_1 is the sequence of positions in which Bob was right about basis measurement. Bob sends to Alice the set S_b

Thus, if Bob wants to know m_0 he must send to Alice throughout a classical channel the set $S_0 = \{I_1, I_0\}$, otherwise if he wants to know m_1 he must send to Alice throughout a classical channel the set $S_1 = \{I_0, I_1\}$.

12. With both the received set S_b and the hash function value HASH1, Alice must be able to prove that Bob has being honest.
13. Lets assume Bob sent $S_0 = \{I_1, I_0\}$. Alice defines two encryption keys K_0 and K_1 using the values in positions defined by Bob in the set sent by him. In this example, lets assume:

$$K_0 = \{1, 0, 1, 0\}$$

$$K_1 = \{0, 0, 0, 1\}.$$

Alice does the following operations:

$$m = \{m_0 \oplus K_0, m_1 \oplus K_1\}.$$

$$\begin{array}{c|cccc} m_0 & 0 & 0 & 1 & 1 \\ K_0 & 1 & 0 & 1 & 0 \\ \hline \oplus & 1 & 0 & 0 & 1 \end{array}$$

$$\begin{array}{c|cccc} m_1 & 0 & 0 & 0 & 1 \\ K_1 & 0 & 0 & 0 & 1 \\ \hline \oplus & 0 & 0 & 0 & 0 \end{array}$$

Adding the two results, m will be:

$$m = \{1, 0, 0, 1, 0, 0, 0, 0\}.$$

After that, Alice sends to Bob the encrypted message m through a classical channel.

14. When Bob receives the message m , in the same way as Alice, Bob uses S_{B1} , values of positions given by I_1 and I_0 and does the decrypted operation. In this case, he does following operation:

$$\begin{array}{c|cccccccc} m & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ \hline \oplus & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{array}$$

The first four bits corresponds to message 1 and he received $\{0, 0, 1, 1\}$, which is the right message m_0 and $\{0, 1, 1, 0\}$ which is a wrong message for m_1 .

5.8.2 Simulation Analysis

Students Name	:	Mariana Ramos
Starting Date	:	November 7, 2017
Goal	:	Perform a simulation of the setup presented below in order to implement BB84 communication protocol.

In this sub section the simulation setup implementation will be described in order to implement the BB84 protocol. In figure 5.64 a top level diagram is presented. Then it will be presented the block diagram of the transmitter block (Alice) in figure 5.56, the receiver block (Bob) in figure 5.57 and finally the eavesdropper block (Eve) in figure 5.58.

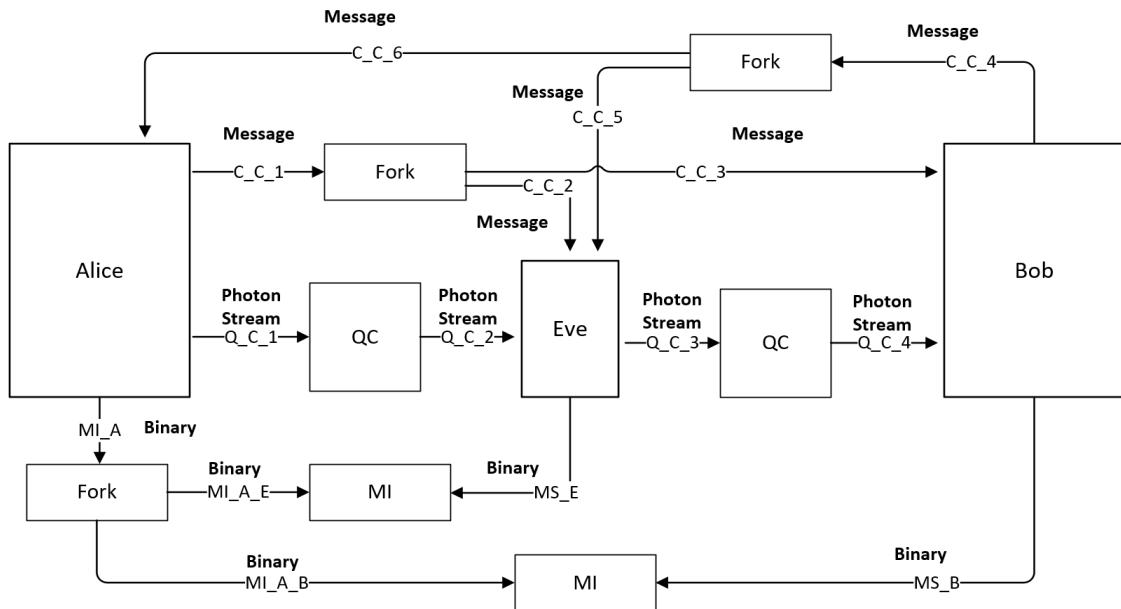


Figura 5.55: Simulation diagram at a top level

Figure 5.64 presents the top level diagram of our simulation. The setup contains three parties, Alice, Eve and Bob where the communication between them is done throughout two classical and one quantum channel. In the middle of the classical channel there is a Fork's diagram which has one input and two outputs. In the case of the classical channel C_C_4 which has the information sent by Bob, the fork's block enables Alice and Eve have access to it. In the quantum communication, the information sent by Alice can be intercepted by Eve and changed by her, or can follow directly to Bob as we can see later in figure 5.58. Furthermore, for mutual information calculation there must be two blocks MI, one to calculate the mutual information between Alice and Eve, and other to calculate the mutual information between Alice and Bob.

In figure 5.56 one can observe a block diagram of the simulation at Alice's side. As it

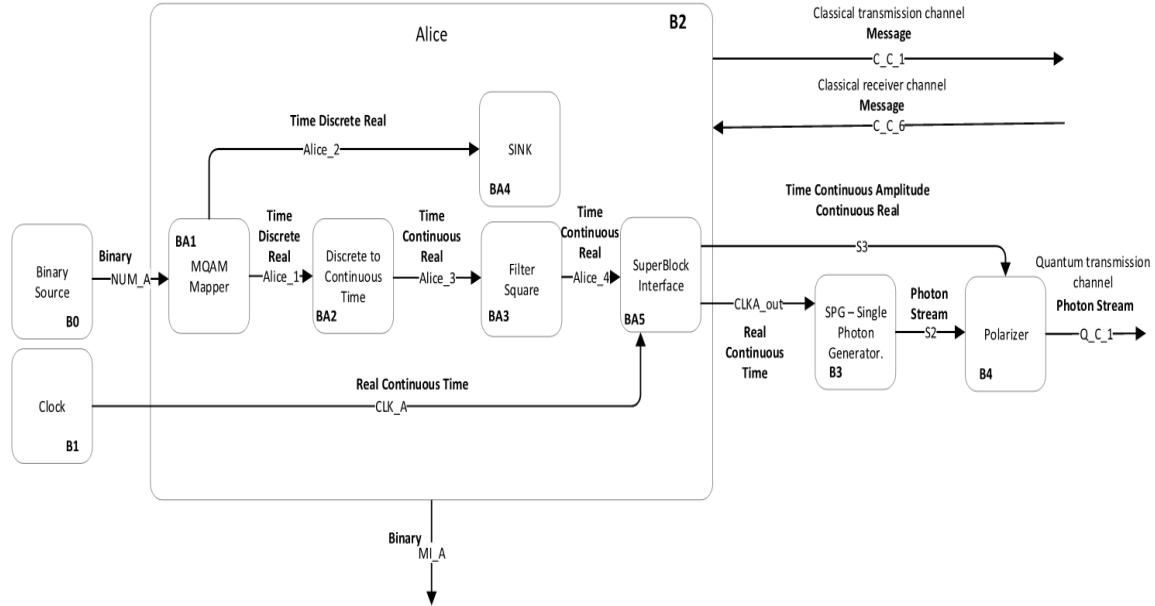


Figura 5.56: Simulation diagram at Alice's side

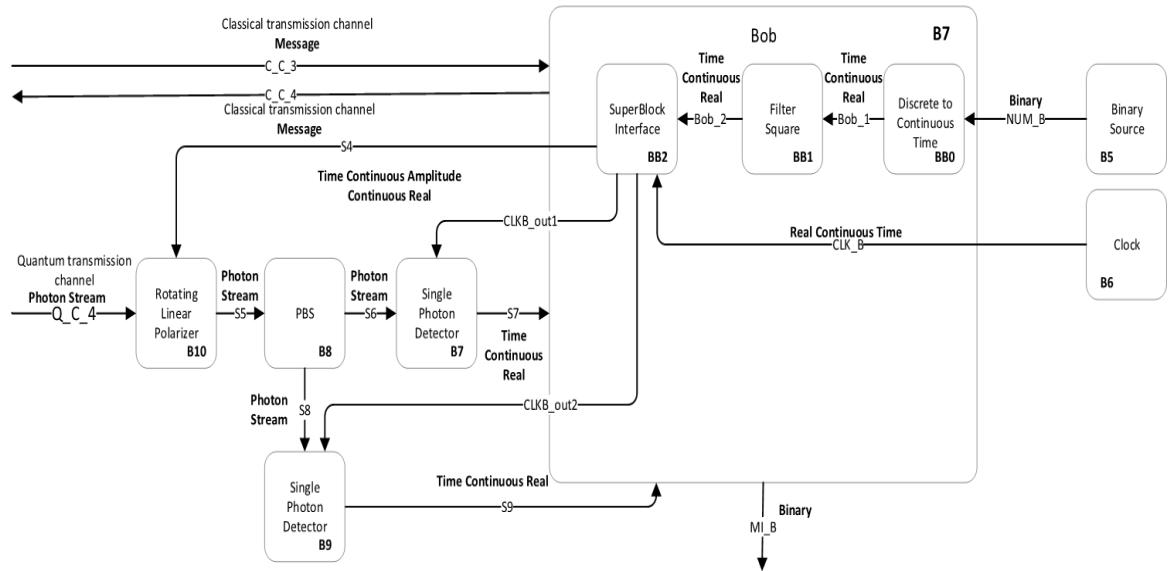


Figura 5.57: Simulation diagram at Bob's side

is shown in the figure, Alice must have one block for random number generation which is responsible for basis generation to polarize the photons, and for key random generation in order to have a random state to encode each photon. Furthermore, she has a Processor

block for all logical operations: array analysis, random number generation requests, and others. This block also receives the information from Bob after it has passed through a fork's block. In addition, it is responsible for set the initial length l of the first array of photons which will send to Bob. This block also must be responsible for send classical information to Bob. Finally, Processor block will also send a real continuous time signal to single photon generator, in order to generate photons according to this signal, and finally this block also sends to the polarizer a real discrete signal in order to inform the polarizer which basis it should use. Therefore, she has two more blocks for quantum tasks: the single photon generator and the polarizer block which is responsible to encode the photons generated from the previous block and send them throughout a quantum channel from Alice to Bob.

Finally, Alice's processor has an output to Mutual Information top level block, Ms_A .

In figure 5.57 one can observe a block diagram of the simulation at Bob's side. From this side, Bob has one block for Random Number Generation which is responsible for randomly generate basis values which Bob will use to measure the photons sent by Alice throughout the quantum channel. Like Alice, Bob has a Processor block responsible for all logical tasks, analysing functions, requests for random number generator block, etc. Additionally, it receives information from Alice throughout a classical channel after passed through a fork's block and a quantum channel. However, Bob only sends information to Alice throughout a classical channel. Furthermore, Bob has one more block for single photon detection which receives from processor block a real discrete time signal, in order to obtain the basis it should use to measure the photons.

Finally, Bob's processor has an output to Mutual Information top level block, Ms_B .

Figure 5.58 presents the Eve's side diagram. Eve's processor has two receiver classical signals, one from Alice (**C_C_2**) and other from Bob (**C_C_5**). About quantum channel, Eve received a quantum message from Alice through the channel **Q_C_1** and depends on her decision the photon can follows directly to Bob or the photon's state can be changed by her. In this case, the photon is received by a block similar to Bob's diagram 5.57 and this block sends a message to Eve's processor in order to reveal the measurement result. After that, Eve's processor sends a message to Alice's diagram similar to figure 5.56 and this block is responsible for encode the photon in a new state. Now, the changed photon is sent to Bob.

In addition, Eve's diagram has one more output Ms_E which is a message sent to the mutual information block as an input parameter.

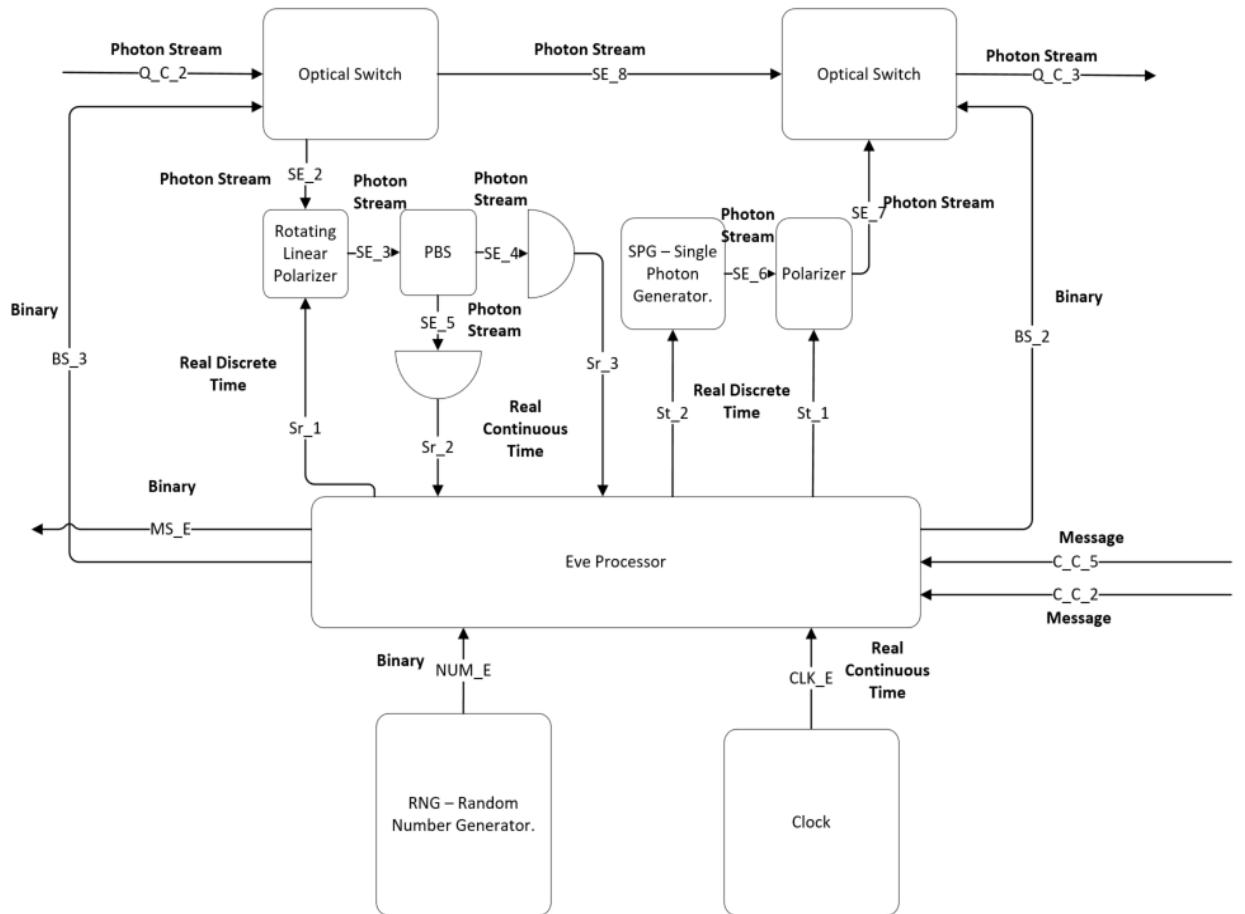


Figura 5.58: Simulation diagram at Eve's side

Tabela 5.8: System Signals

Signal name	Signal type	Status
NUM_A	Binary	
MI_A	Binary	
CLK_A	TimeContinuousAmplitudeContinuous	
CLK_A_1	TimeContinuousAmplitudeContinuous	
S2	PhotonStreamXY	
S3	TimeContinuousAmplitudeDiscreteReal	
C_C_1	Messages	
C_C_6	Messages	
Q_C_1	PhotonStreamXY	

Table 5.11 presents the system signals as well as them type.

Tabela 5.9: System Input Parameters

Parameter	Default Value	Description
RateOfPhotons	1K	
vector<t_iqValues> iqAmplitudeValues	{-45.0,0.0},{0.0,0.0},{45.0,0.0},{90.0,0.0}	

Tabela 5.10: Header Files

File name	Description	Status
netxpto.h		✓
alice_qkd.h		Working on
binary_source.h		✓
bob_qkd.h		Missing
clock_20171219.h		✓
discrete_to_continuous_time.h		✓
eve_qkd.h		Missing
m_qam_mapper.h		✓
optical_switch.h		Missing
polarization_beam_splitter.h		Working on
polarizer.h		Working on
pulse_shaper.h		✓
rotator_linear_polarizer.h		Working on
single_photon_detector.h		Missing
single_photon_source_20171218.h		✓
sink.h		✓
super_block_interface.h		✓

Tabela 5.11: Source Files

File name	Description	Status
netxpto.cpp		✓
bb84_with_discrete_variables.cpp		Working on
alice_qkd.cpp		Working on
binary_source.cpp		✓
bob_qkd.cpp		Missing
clock_20171219.cpp		✓
discrete_to_continuous_time.cpp		✓
eve_qkd.cpp		Missing
m_qam_mapper.cpp		✓
optical_switch.cpp		Missing
polarization_beam_splitter.cpp		Working on
polarizer.cpp		Working on
pulse_shaper.cpp		✓
rotator_linear_polarizer.cpp		Working on
single_photon_detector.cpp		Missing
single_photon_source_20171218.cpp		✓
sink.cpp		✓
super_block_interface.cpp		✓

5.8.3 Open Issues

There still are some open issues in simulation code.

One of them was detected in block `single_photon_source_20171218.cpp`. This block should assume each sample with 4 real values, since it writes two complex values each time the block runs, i.e each `bufferput()` should write an array of two complex values in `outputSignal, outputSignals[0]->bufferPut(valueXY)`, where `t_complex_xy valueXY = {valueX, valueY}` and `t_complex valueX = (realValue_1,realValue_2)`. This way, independently of the number of samples these four values should always be written. However, if we chose a number of samples which is not divisible by 4, the four numbers are not written in the last "sample" and the array data for X's values and Y's values have different sizes which is wrong. For example, if we chose 10 samples to acquire, the last values correspond to X's values instead of Y's values and the first array data is longer than the other.

5.9 Quantum Oblivious Key Distribution with Discrete Variables

Student Name	:	Mariana Ramos
Starting Date	:	September 18, 2017
Goal	:	Quantum oblivious key distribution (QOKD) implementation with discrete variables.
Directory	:	sdf/ot_with_discrete_variables.

Oblivious Transfer (OT) is a fundamental primitive in multi-party computation. The one-out-of-two OT consists in a communication protocol between Alice and Bob. At the beginning of the protocol Alice has two messages m_1 and m_2 and Bob wants to know one of them, m_b , without Alice knowing which one, i.e. without Alice knowing b , and Alice wants to keep the other message private, i.e. without Bob knowing $m_{\bar{b}}$. therefore two conditions must be fulfilled:

1. The protocol must be concealing, i.e at the beginning of the protocol Bob does not know nothing about Alice's messages, while at the end of the protocol Bob will learn the message m_b chosen by him.
2. The protocol is oblivious, i.e Alice cannot learn anything about Bob's choice, bit b , and Bob cannot learning nothing about the other message $m_{\bar{b}}$.

In order to implement OT between two parties (Alice and Bob) they must be able to exchange continuously oblivious keys, i.e a QOKD system must exist between them.

5.9.1 Theoretical Description

Quantum Oblivious Key Distribution System (QOKD)

In this section we are going to describe the Quantum Oblivious Key Distribution system (QOKD). The QOKD system enables two parties (Alice and Bob) to share a set of keys. These keys have the particularity of being half right and half wrong. Only Bob knows which are right and wrong keys.

Considering a discrete variables implementation, both Alice and Bob agree with the following correspondence, where + corresponds to *Rectilinear Basis* and \times corresponds to *Diagonal Basis*,

<i>Basis</i>	
0	+
1	\times

Alice and Bob also agree with the bit correspondence for each direction for each basis. For *Rectilinear basis*, "+",

	Basis "+"
0	$\rightarrow (0^\circ)$
1	$\uparrow (90^\circ)$

and for *Diagonal Basis*, "x",

	Basis "x"
0	$\searrow (-45^\circ)$
1	$\nearrow (45^\circ)$

1. The first step is to establish for both Alice and Bob the block length l . In this case, lets assume $l = 16$. Alice randomly generate a bit sequence with length l . Therefore, she must define two sets randomly: S_{A1} which contains the basis values; and S_{A2} , which contains the key values.

In that case, lets assume she generates the following sets $S_{A1'}$ and $S_{A2'}$:

$$S_{A1'} = \{0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1\},$$

$$S_{A2'} = \{1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1\}.$$

2. Next, Alice sends to Bob throughout a quantum channel l photons encoded using the basis defined in $S_{A1'}$ and according to the key bits defined in $S_{A2'}$.

Therefore, in the current example, Alice sends the following photons,

$$\begin{aligned} S_{AB} &= \{\uparrow, \uparrow, \nearrow, \searrow, \downarrow, \rightarrow, \rightarrow, \searrow, \nearrow, \uparrow, \rightarrow, \searrow, \nearrow, \downarrow, \uparrow, \nearrow\} \\ &= \{90^\circ, 90^\circ, 45^\circ, -45^\circ, -45^\circ, 0^\circ, 0^\circ, -45^\circ, 45^\circ, 90^\circ, 0^\circ, -45^\circ, 45^\circ, -45^\circ, 90^\circ, 45^\circ\}. \end{aligned}$$

3. Bob also randomly generates $l = 16$ bits, which are going to define his measurement basis, $S_{B1'}$. Lets assume,

$$\begin{aligned} S_{B1'} &= \{0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1\} \\ &= \{+, \times, \times, +, +, \times, +, \times, \times, +, \times, +, +, \times\}. \end{aligned}$$

Bob will get l results:

$$S_{B2'} = \{1, -, \underline{0}, 0, -, 1, \underline{1}, -, 1, -, 1, 0, 1, 1, \underline{0}, 1\}.$$

The "−" corresponds to no clicks in Bob's detector, due to attenuation. The underlined values are bits which were measured with a correct basis but an error has occurred due to imperfections in the quantum communication system.

4. Bob is going to send a "-1" or a hash value to Alice for each measurement that he performed, thereby being "-1" the measurements which correspond to no clicks. In this case, we are going to assume that the hash value is calculated using the SHA-256 algorithm [3]. In detail, Bob has two sets $S_{B1'}$ and $S_{B2'}$ and he is going to generate the set S_{BH1} with l values (" -1 " or hash values calculated for each position of $S_{B1'}$ with the correspondent position of $S_{B2'}$). Therefore, Bob will send to Alice the following set:

$$S_{BH1} = \{S_1, -1, S_2, S_3, -1, S_4, S_5, -1, S_6, -1, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}\}.$$

5. Since Alice has received the confirmation of measurement from Bob, i.e after Alice has received S_{BH1} , she sends throughout a classical channel the basis which she has used to codify the photons updated with the information about the no received photons,

$$S_{A1'} = \{0, -1, 1, 1, -1, 0, 0, -1, 1, -1, 0, 1, 1, 1, 0, 1\}$$

Due to attenuation, the previous sets are reduced to the length 12 and they shall be replaced by the following:

$$S_{A1} = \{0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1\},$$

$$S_{A2} = \{1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1\},$$

$$S_{B1} = \{0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1\},$$

$$S_{B2} = \{1, \underline{0}, 0, 1, \underline{1}, 1, 1, 0, 1, 1, \underline{0}, 1\}$$

Note that S_{B2} still has errors.

6. In order to know which photons were measured correctly, Bob does the operation $S_{B3} = S_{B1} \oplus S_{A1}$. In the current example,

$$\begin{array}{c|cccccccccccc} S_{B1} & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ \hline S_{A1} & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ \oplus & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}$$

In this way, Bob gets

$$S_{B3} = \{1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1\}.$$

When Bob uses the right basis he gets the values correctly, apart from possible errors in transmission, when he uses the wrong basis he just guess the value. The values "1" correspond to the values he measured correctly and "0" to the values he just guessed. Thus, Bob is building two sets of keys, one with correct basis measurements values and other with the wrong basis measurement values that he just guessed.

Thus, Bob has two pair of sets, one for the right basis,

$$S_{B_{rp}} = \{1, 2, 5, 6, 8, 11, 12\},$$

$$S_{B_{rb}} = \{1, 0, 1, 1, 0, 0, 1\},$$

where $S_{B_{rp}}$ is the set of positions and $S_{B_{rb}}$ is the set of bit values he measured for each position. The other pair is for photons he measured with the wrong basis and then he just guessed the values,

$$S_{B_{wp}} = \{3, 4, 7, 9, 10\},$$

$$S_{B_{wb}} = \{0, 1, 1, 1, 1\},$$

where $S_{B_{wp}}$ is the set of positions and $S_{B_{wb}}$ is the set of bit values he measured for each position.

Nevertheless, due to errors in transmission, some bits in $S_{B_{rb}}$ may be not right.

At this point, in order to test Bob's honesty and to estimate the QBER of the channel, Alice is going to ask Bob to open some pairs of the Bob's sets. The definition of the protocol to test Bob's honesty is still an open issue. However, depending on the QBER estimated by her, Alice must have a parameter to set the number of right position she wants to open, i.e she must open a minimum number of right position in order to guarantee a minimum QBER. This will increase the security of the protocol. Alice chooses some positions to open and tells Bob which positions she wants to open. Bob sends to Alice the pairs she chose and then these pairs are eliminated from them sets. Lets assume she asked to open the positions 10, 11 and 12. If she concludes Bob is not being honest, she stops the protocol and they must start it again. Otherwise, the protocol continues. Lets assume Alice has verified these pairs using the hash function committed by Bob and concluded that he is being honest. Therefore, she sends to Bob the QBER estimated by her.

Now, Bob has the previous sets replaced by the following,

$$S_{B_{rp}} = \{1, 2, 5, 6, 8\}$$

$$S_{B_{rb}} = \{1, 0, 1, 1, 0\}$$

$$S_{B_{wp}} = \{3, 4, 7, 9\}$$

$$S_{B_{wb}} = \{0, 1, 1, 1\}$$

Bob is going to use a modified version of *Cascade algorithm* to correct the errors due transmission.

Modified version of Cascade Algorithm

The Cascade algorithm is often used with a key set where all values are supposed right. In this case, Bob has two pairs of sets, one with the position and bit values of photon he measured with the correct basis and other with position and bit values of photon

he measured with the wrong basis. He only needs to apply the Cascade algorithm in the set that he measured the photons correctly [4]. However, he must apply a modified version of the Cascade in the other set in order to keep in secret from Alice which set corresponds to right and which set corresponds to wrong measurements.

Bob randomly generates a bit value. If he gets 0, he will send to Alice the set $\{S_{B_{rp}}, S_{B_{wp}}\}$. Otherwise, if he gets 1 he will send the set $\{S_{B_{wp}}, S_{B_{rp}}\}$. This guarantee that Alice does not know which is the right or wrong set. Lets assume this random bit is "0" and he sends $\{S_{B_{rp}}, S_{B_{wp}}\}$.

- (a) Bob starts by applying the normal cascade to the set $S_{B_{rb}}$. After both know the error estimative Bob determine if the error rate is above the fail threshold. If it is truth they must start the procedure again. Lets assume the estimated error rate is acceptable. Bob and Alice use a random permutation which is represented in figure 5.59 for a larger number of bits (agreed at the beginning) by applying it to the shifted keys, in order to guarantee the spread out of the error bits randomly and to separate consecutive errors from each other.

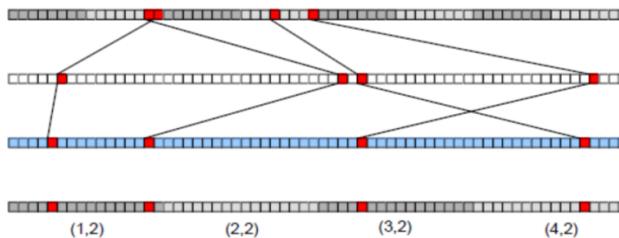


Figura 5.59: Cascade Algorithm - permutation

- (b) Bob and Alice divide all the shifted key bits into blocks of N bits depending on the estimated error rate in order to have one or no error per block. In general, the sets of keys are too large and it is easier to explain the algorithm based on a larger number of bits. Therefore, figure 5.60 represents the typical cascade initial steps. However, in this case, the set to be corrected only has five bits, therefore they divide the set in two sub-blocks, one with 3 bits and other with 2 bits.
- (c) They use a classical channel to compare the block parities. For blocks with different parities, an odd number of errors must exist, otherwise an even number of errors would mask each other. Thus, the block in which the parities disagree is divided in half into two smaller blocks of length $\frac{N}{2}$, and another parity check is performed on the first sub-block, as one can see in figure 5.61. As it was referred above, there is at least one error in one sub-block being the error location revealed by the parity of one sub-block. In other words, if the parity of the first sub-block passes, the error will be in the second sub-block. The sub-block with error will be sub-divided until the error is found.

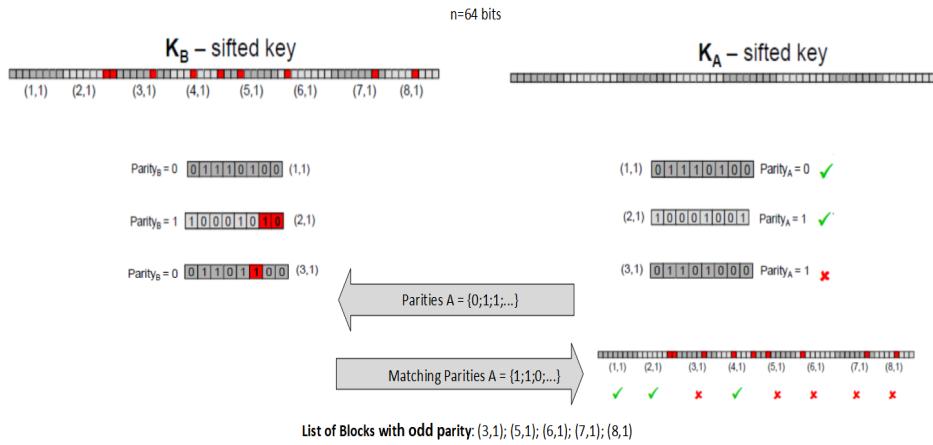


Figura 5.60: Cascade Algorithm

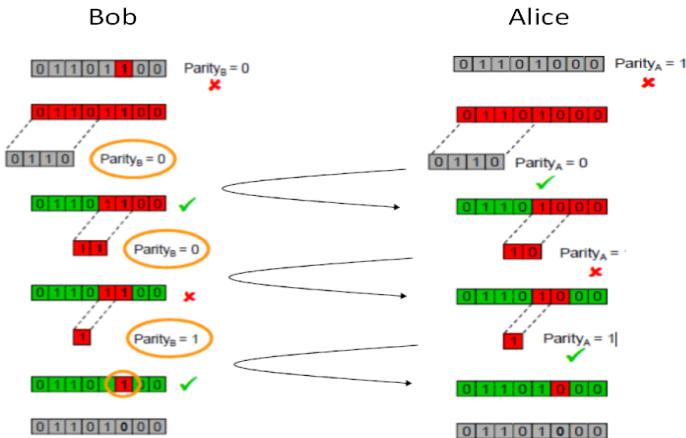


Figura 5.61: Cascade Algorithm - example of error correction

- (d) When the error is corrected, the last bit of the block is discarded in order to prevent the gain of additional information by Bob.

In this case, let's assume the set of right positions was corrected with the algorithm described above and it will be replaced by the following:

$$S_{B_{rp}} = \{1, 2, 5, 6, 8\}$$

$$S_{B_{rb}} = \{1, 1, 0, 1, 0\}$$

In order to test Alice's honesty, Bob must verify if the QBER sent by Alice is a realistic value. If it is not he stops the protocol and they must start again. Otherwise, the protocol continues.

After that, Bob needs to apply the Fake Cascade to the set $S_{B_{wb}}$. The main goal of this

step is to convince Alice she is performing the real Cascade but she is not.

- (a) First of all, based on the positions contained in S_{Bob} , Bob must build an array with the correspondent bits in a random order and informs Alice the order of positions. In order to best explain this version of the algorithm, lets assume a larger set of bits.

Bob sends to Alice throughout a classical channel the new positions order as if it were the permutation step represented in figure 5.59 in real Cascade algorithm.

- (b) Assuming each of them has a set with 32 bits randomly organized by Bob, they divide the supposed shifted keys in blocks with N bits according to the estimated error rate. As the QBER is the same as for real cascade, Bob will assume the same number of errors, even if he starts for this modified version he can know the number of errors from QBER estimated by Alice.
- (c) Bob and Alice use a classical channel to compare the block parities. Alice sends to Bob her parity list. Based on Alice's parity list, Bob sends a block list with odd parities, i.e the blocks position in which parity supposed disagree. This list is randomly built based on the number of errors considered by Bob, i.e if he considered five errors from QBER estimative, he will distributed them randomly and after that he will fill the remaining spaces with even parities. Bob sends to Alice the set with the list of odd parities, i.e the list of sub-sets he has different parities than Alice.
- (d) The blocks with errors will be consecutively divided until they found the supposed errors. Since we have assumed there were five errors, this is the number of errors that Alice must supposedly correct.

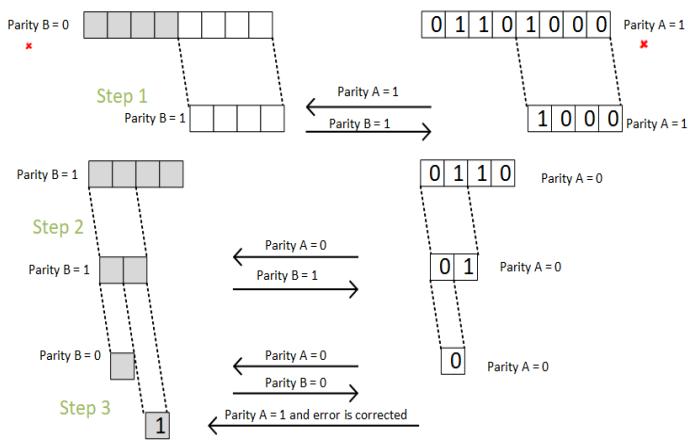


Figura 5.62: Fake cascade - example of error correction

Lets assume one of the blocks with error and analyse figure 5.62. Bob starts with a set filled with random bits, therefore we do not need to know which bits are. Alice starts by dividing her set in half with two blocks with N bits.

Step 1: Bob chooses one of the to blocks and informs Alice she must send the parity of this block. Lets assume he chose sub-block 2. She sends the parity and Bob is going to send his parity, which after know Alice's block parity he send the opposite parity. As referred in normal Cascade, there must be one error or no error in each block. Thus, since the parities disagree, the error must be in seconde block. They start the procedure to correct it.

Step 2: Bob divides again the sub-block in half with $\frac{N}{2}$ bits and asks Alice for the parity of the first sub-block. She sends her parity equals to 0 and Bob sends to her the opposite parity again.

Step 3: They divide the sub-block in half again and Bob asks Alice for the parity of the first bit. Alice sends to him the parity equals to her. As the error is not in the first be, it must be in the second, therefore Bob is able to correct this bit with the information sent by Alice.

Note that Bob make his choice of which half analyse first using a random bit generator result. If he got "0" he starts with the first half of the sub-block, otherwise, if he got "1", he starts with the second half. In addition, they must discard the last bit of each block and sub-block in which fake Cascade were applied in order to guarantee that Bob does not gain additional information.

In this case, after apply the fake Cascade to $S_{B_{wb}}$, lets assume,

$$\begin{aligned} S_{B_{wp}} &= \{3, 4, 7, 9\} \\ S_{B_{wb}} &= \{0, 1, 1, 0\} \end{aligned}$$

If Bob starts by applying the fake Cascade, he must test Alice's honesty at the beginning of the real Cascade application, based on the number of errors he has. If he thinks that the QBER sent by Alice is unrealistic, he stops the protocol at this point.

7. When Alice sends to Bob a photons set, they are building a set of pairs (array positions and bit values which correspond to measured photons at Bob's side and to the key bit with the photon was encoded at Alice's side). The main goal is to guarantee that Bob has the same number of right and wrong pairs. In addition, they must know information about t (represented in figure 5.63) which corresponds to the points where the previous condition is verified.

Since Bob has sent to Alice the information about the smallest set, in this example, Alice know that there are four pairs of wrong positions and five pairs of right positions. Alice must destroy one of the right pairs by asking Bob to open it. Therefore, at $t = 8$ both know that there are the same number of right and wrong pairs thereby being the main goal guaranteed.

As we can see in figure 5.63, unlike Bob, Alice does not know which positions corresponds to right or wrong measurements performed by Bob. They have been building these sets during all protocol.

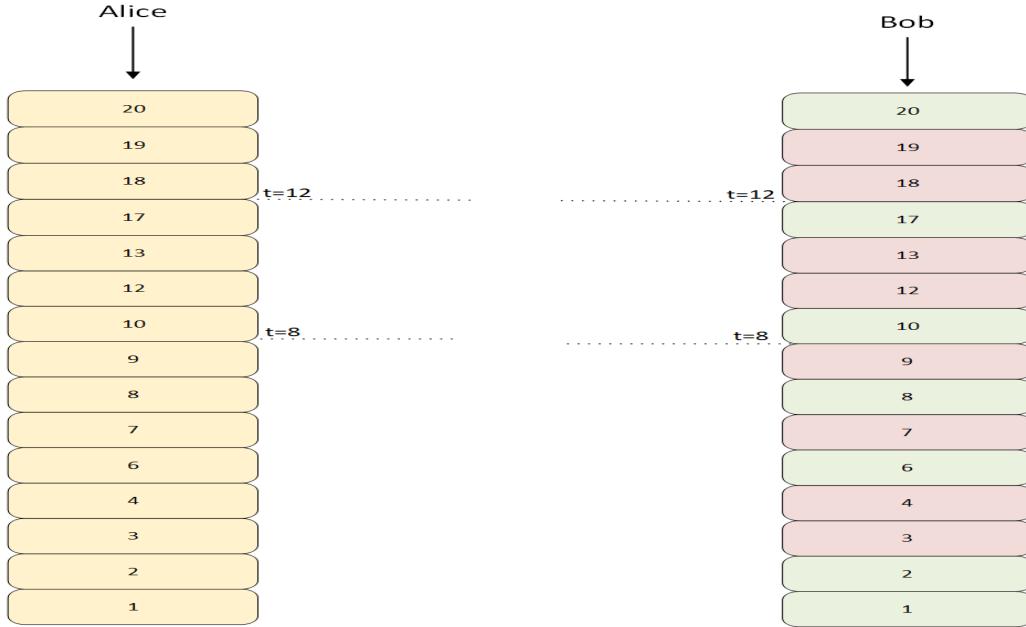


Figura 5.63: Alice and Bob key sets.

1-out-of-2 Oblivious Transfer Protocol with QOKD system

At this time, we are going to describe the oblivious transfer protocol with detail. As it was referred at the beginning, Alice sends two messages to Bob and he wants to know one of them. Alice does not know which message Bob wants and Bob only know the message he wants, i.e he does not know anything about the other message. Furthermore, only Alice knows information about messages m_0 and m_1 . In this case, lets assume the following two messages with size $s = 4$, $m_0 = \{0011\}$ and $m_1 = \{0001\}$. Alice must guarantee $t = s \times 2$. In order to do that, she must destroy the remaining pairs. In this case, there is no need to do that because they have a set for $t = 8$ with the same number of wrong and right pairs.

1. Bob defines two new sub-sets, I_0 and I_1 . I_0 is a set of values with photons array positions which Bob just guessed the measurement since he did not measure them with the same basis as Alice, I_1 is a set of values with photons array positions which Bob measured correctly since he used the same basis as Alice used to encoded them. The position of the pairs of each right and wrong message are in the keys sets that they have been building during the protocol.

In this example, the message size is 4. Since, at this time $t = 10$ and we have 5 right pairs and 5 wrong pairs, Alice ask to Bob to open one right pair and one wrong pair in order to both have exactly the message's size number of right and wrong pairs. Lets assume that Alice opened two pairs, position 15 which is a wrong measurement and position 10 which is a right measurement. We have now $t = 8$.

Next, Bob defines two sub-sets with size $s = 4$:

$$I_0 = \{3, 4, 7, 9\},$$

and

$$I_1 = \{1, 2, 6, 8\},$$

where I_0 is the sequence of positions in which Bob was wrong about basis measurement and I_1 is the sequence of positions in which Bob was right about basis measurement. Bob sends to Alice the set S_b

Thus, if Bob wants to know m_0 he must send to Alice throughout a classical channel the set $S_0 = \{I_1, I_0\}$, otherwise if he wants to know m_1 he must send to Alice throughout a classical channel the set $S_1 = \{I_0, I_1\}$.

2. Alice is sure about Bob's honesty, since she knows he only has 4 right basis to measure the photons. In addition, Alice cannot know which message Bob chose because she did not know the order that he sent the sets.
3. Lets assume Bob sent $S_0 = \{I_1, I_0\}$. Alice defines two encryption keys K_0 and K_1 using the values in positions defined by Bob in the set sent by him. In this example, lets assume:

$$K_0 = \{1, 1, 1, 0\}$$

$$K_1 = \{0, 0, 0, 1\}.$$

Alice does the following operations:

$$m = \{m_0 \oplus K_0, m_1 \oplus K_1\}.$$

$$\begin{array}{c|cccc} m_0 & 0 & 0 & 1 & 1 \\ \hline K_0 & 1 & 1 & 1 & 0 \\ \oplus & 1 & 1 & 0 & 1 \end{array}$$

$$\begin{array}{c|cccc} m_1 & 0 & 0 & 0 & 1 \\ \hline K_1 & 0 & 0 & 0 & 1 \\ \oplus & 0 & 0 & 0 & 0 \end{array}$$

Adding the two results, m will be:

$$m = \{1, 1, 0, 1, 0, 0, 0, 0\}.$$

After that, Alice sends to Bob the encrypted message m through a classical channel.

4. When Bob receives the message m , in the same way as Alice, Bob uses $S_{B1'}$ values of positions given by I_1 and I_0 and does the decrypted operation. In this case, he does following operation:

$$\begin{array}{c|cccccccc} m & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline \oplus & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{array}$$

The first four bits corresponds to message 1 and he received $\{0, 0, 1, 1\}$, which is the right message m_0 and $\{0, 1, 1, 0\}$ which is a wrong message for m_1 .

Nearest Private Query

The Nearest Private Query is another example of QOKD application [5]. In this case, there are also two parties: a user (who we called Bob) and a data owner (who we called Alice). Bob has an input secrete parameter x and Alice has a private data set A .

Lets assume Bob's secret input parameter $x = 8$ and Alice's data set $A = \{1, 2, 3, 6, 7, 10, 11, 14\}$.

Bob wants to know which element (x_i) is the closest to x in Alice data set A , without revealing his secrete x . Alice does not know which is the Bob's secret parameter, and Bob does not know any information about Alice's set except the closest element x_i to his x .

Step 1 Alice generates a new set with $N = 2^n$ elements, $D(j)$ for $j = 0, 1, \dots, N - 1$, in which $D(j) = x_l$ being $x(l)$ the closest element to j in A . n is the number of bits that Alice needs to represent each element of her data set.

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$D(j)$	1	1	2	3	3	6	6	7	7	10	10	11	11	14	14	14
	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	0	1	1	0	0	1	1	0	0	1	1	0	0	0

Step 2 Bob and Alice have a set of keys K_i^* and K_i , respectively, with 64 elements where 60 are bits resulted from wrong basis measurement and 4 resulted from correct basis measurement. This allows Alice to know that Bob is being honest. They start from QOKD with symmetric keys and then Alice is being asking Bob to destroy pairs of keys until they have a set with the characteristics above.

Step 3 Bob sends to Alice the set S with all positions referred random measurements except the position bits at $j = 8$.

Step 4 Alice encoded all bits with the corresponding bit keys at the positions sent by Bob, and then she sends the encoded message to Bob.

Step 5 Bob receives the encoded message and apply an operation XOR based on the correspondent bits to the positions he sent two Alice above.

Step 6 At the end, Bob gets the message he wanted, the closest element to 8, which corresponds to message 0, 1, 1, 1 or 7 and he remains know nothing about the other elements of Alice's data set. In the same way, Alice does not know which element Bob wants to know.

QKD from QOKD

All Quantum Key Distribution systems can be obtained from the QOKD system presented in this report. The Quantum Oblivious Key Distribution system allows to obtain symmetric secret keys and symmetric or asymmetric oblivious keys.

In fact, since Alice and Bob has the same set of keys and at some tab Alice knows that there are the same number of right and random bits, she can obtain any combination from this set by asking Bob to destroy some pairs.

QOKD has a big advantage over other QKD systems because of its versatility. However, the biggest disadvantage is related with a large number of photons consumption which can became the communication rate slower.

5.9.2 Simulation Analysis

First of all, the protocol will be simulated and then a experimental setup will be built in the laboratory.

The main goal of this simulation is to demonstrate that Bob was able to learn correctly message m_b and he does not know the message $m_{\bar{b}}$.

As one may see in figure 5.64 this simulation will have three top level blocks. Two of them are Alice and Bob and they are connected through two classical channels and one quantum channel. In addition, a third block will be performed in order to calculate the *Mutual Information*. The mutual information (MI) between Alice and Bob is defined in terms of their joint distribution.

1. In figure 5.65 one can observe a block diagram of the simulation at Alice's side. As it is shown in the figure, Alice must have one block for random number generation which is responsible for basis generation to polarize the photons, and for key random generation in order to have a random state to encode each photon. Furthermore, she has a Processor block for all logical operations: array analysis, hash function results validation, random number generation requests, and others. This block also receives the start information, i.e. message size s and messages m_0 and m_1 , as well as information from Bob, i.e sets I_0 and I_1 , hash function results, and others. In addition it is responsible for set the initial length l of the first array of photons which will send to Bob. This block also must be responsible for send classical information to Bob. Finally, Processor block will also send a real continuous time signal to single photon generator, in order to generate photons according to this signal, and finally this block

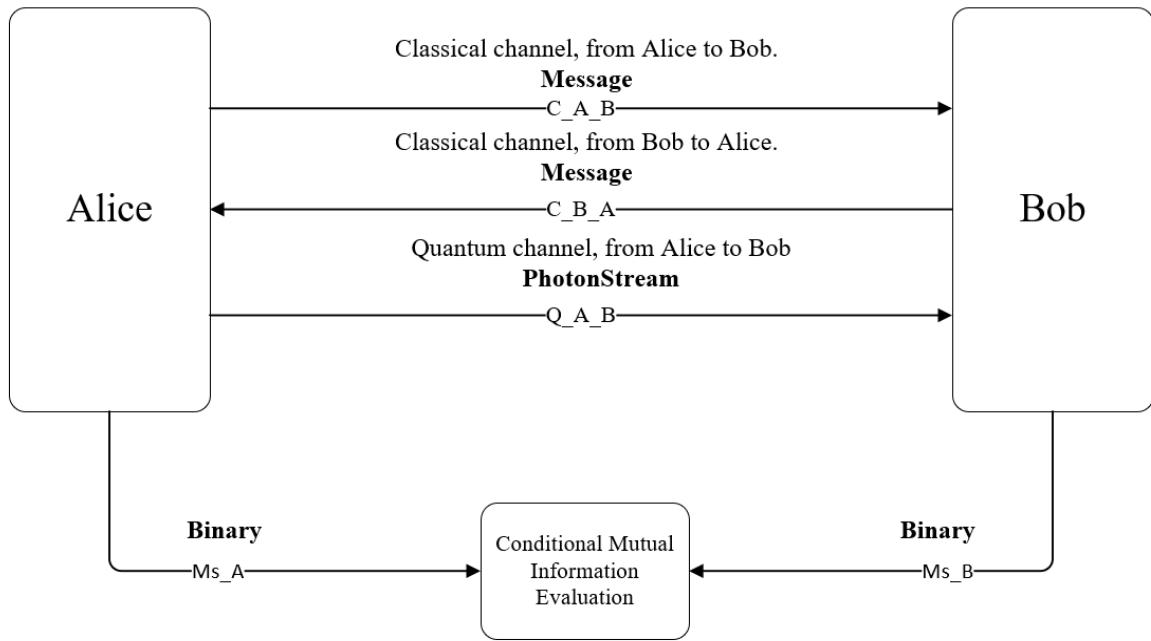


Figura 5.64: Simulation diagram at a top level

also sends to polarizer a real discrete signal in order to inform the polarizer which basis it should use. Therefore, she has two more blocks for quantum tasks: the single photon generator and the polarizer block which is responsible to encode the photons generated from the previous block and send them throughout a quantum channel from Alice to Bob.

Finally, Alice's processor has an output to Mutual Information top level block, Ms_A .

2. In figure 5.66 one can observe a block diagram of the simulation at Bob's side. From this side, Bob has one block for Random Number Generation which is responsible for randomly generate basis values which Bob will use to measure the photons sent by Alice throughout the quantum channel. Furthermore, this Block will generate the random bits that Bob needs in Modified Version of Cascade Algorithm. Like Alice, Bob has a Processor block responsible for all logical tasks, i.e Hash function generation, analysing functions, requests for random number generator block, etc. Additionally, it receives information from Alice throughout a classical channel and a quantum channel but it sends information to Alice only throughout a classical channel. Furthermore, Bob has one more block for single photon detection which receives from processor block a real discrete time signal, in order to obtain the basis it should use to measure the photons.

Finally, Bob's processor has an output to Mutual Information top level block, Ms_B .

3. Mutual Information calculation

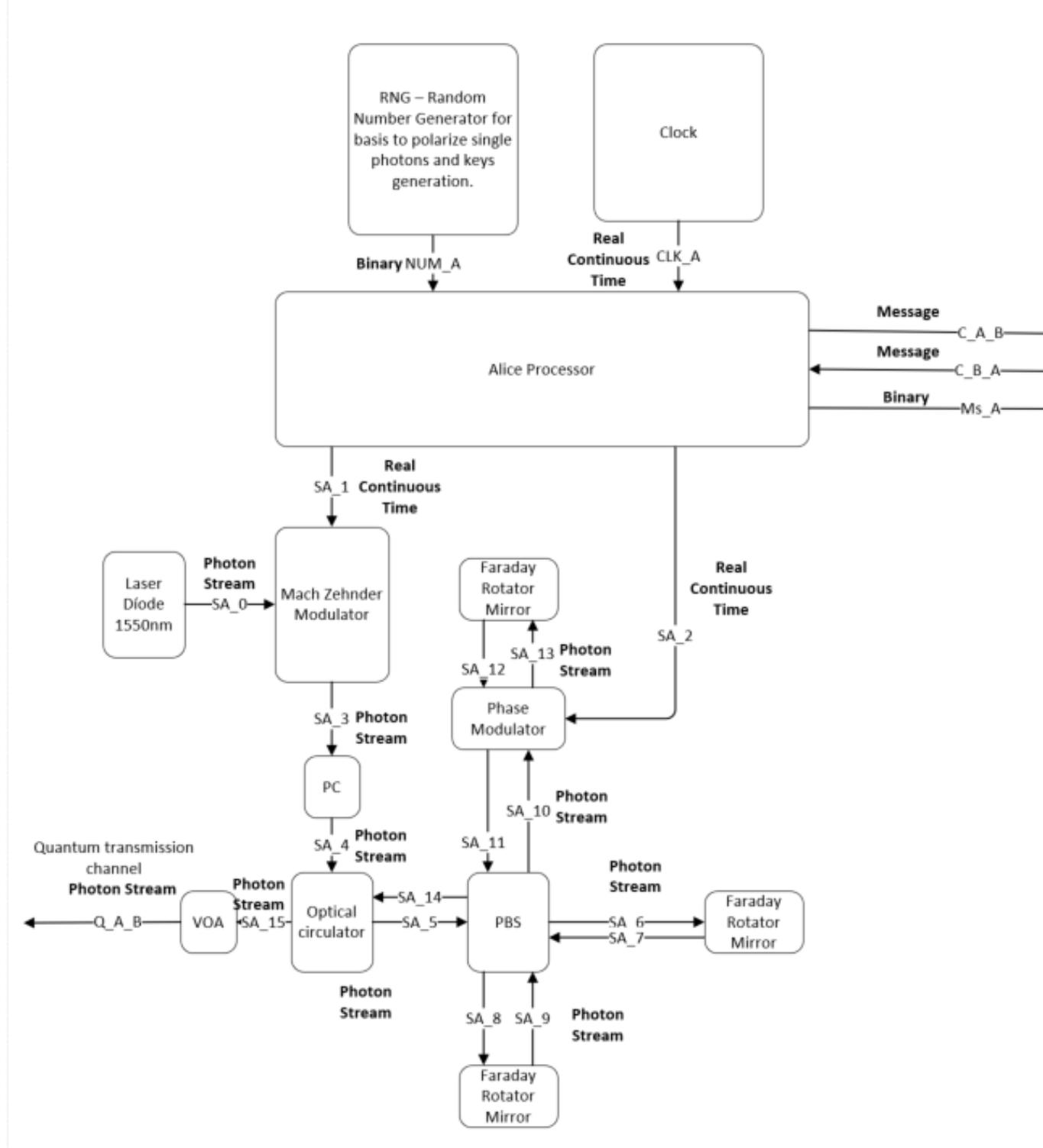


Figura 5.65: Simulation diagram - Alice's side

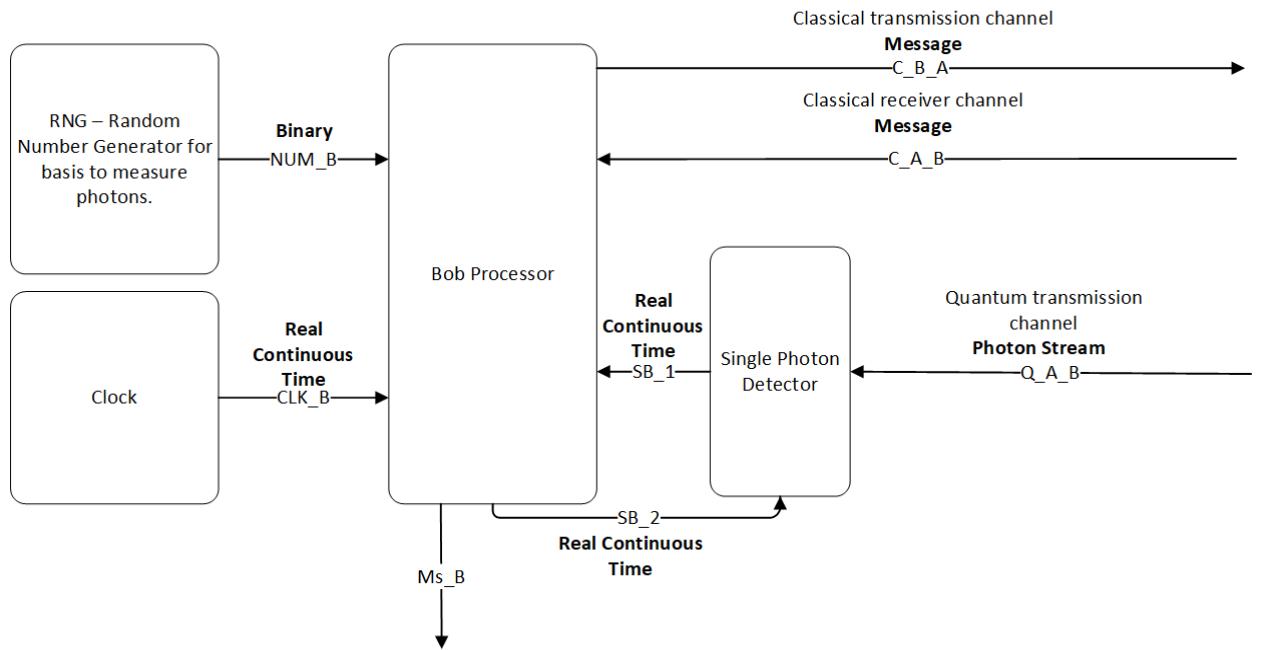


Figura 5.66: Simulation diagram - Bob's side

Tabela 5.12: System Signals

Signal name	Signal type	Status
NUM_A	Binary signal	
NUM_B	Binary signal	
CLK_A	Real continuous Time	
CLK_B	Real continuous Time	
C_A_B	Message	
C_B_A	Message	
SA_1	Real Continuous Time	
SA_2	Real Continuous Time	
SA_3	Real Discrete Time	
SA_3	Real Continuous Time	
Q_A_1	Photon Stream	
Q_A_2	Photon Stream	
Q_A_B	Photon Stream	
Ms_A	Binary	
Ms_B	Binary	
S_B1	Real continuous Time	
S_B2	Real continuous Time	

Tabela 5.13: System input parameters

Parameter	Default Value	Description
messageSize	4	Size of the message Alice must send to Bob.
blockLength	16	Block length.
symbolRate	100K	

Tabela 5.14: Header Files

File name	Description	Status
random_number_generator.h		
single_photons_generator.h		
single_photons_detector.h		
encorder.h		
decoder.h		
messageToSend.h		
message_toReceive.h		
mutual_information.h		
cascade_truth.h		
cascade_fake.h		
Sha256.h		

Tabela 5.15: Source Files

File name	Description	Status
random_number_generator.c		
single_photons_generator.c		
single_photons_detector.c		
encorder.c		
decoder.c		
messageToSend.c		
message_toReceive.c		
mutual_information.c		
QOKD_main.c		
cascade_truth.c		
cascade_fake.c		
Sha256.c		

5.9.3 Experimental Setup

In figure 5.67 are presented the experimental setup to be performed in the lab. The main goal is to build an experimental setup in which Alice and Bob communicate through two classical

channels and one quantum channel that will have only one direction (Alice to Bob).

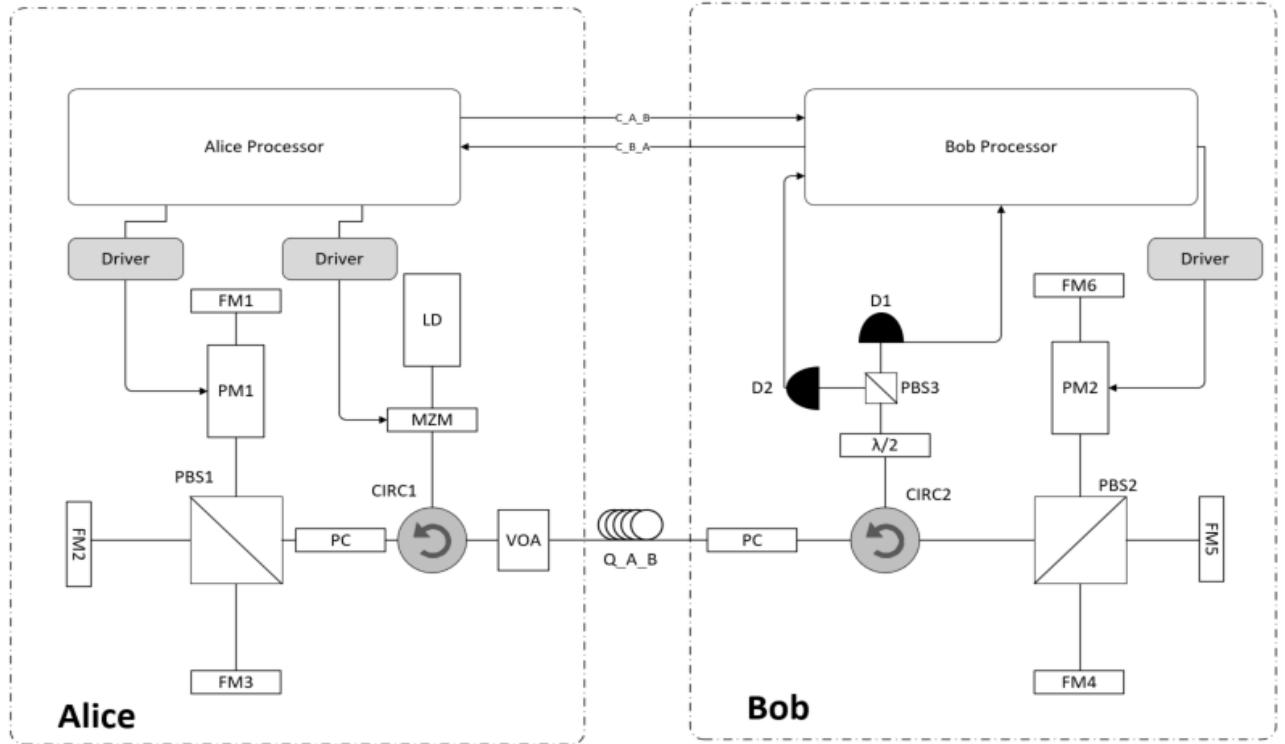


Figura 5.67: QOKD Experimental setup

The laser **LD** emits light with a wavelength of 1550 nm and then the light passes throughout a Mach-Zehnder Modulator (**MZM**) in order to have pulsed light. The light is polarized with a polarizer controller and gets a linear polarization of 45°. When the light reaches the polarization beam splitter, **PBS1**, single photon pulses in 45° linearly polarization state

$$|in\rangle = |45\rangle = \frac{\sqrt{2}}{2}(|H\rangle + |V\rangle)$$

and each pulse is divided in **PBS1** into two orthogonal components P_x and P_y . P_x is directly transmitted in **FM2** direction and it is reflected by it with its stated rotated, which means it has a new direction $|V\rangle$. This way, when it reaches again **PBS1** is reflected to **FM3** direction and it happens the same being the P_x component transmitted through the **PBS1** in **FM1** direction passing through the phase modulator, **PM**, that applies a phase shift to it, than it follows to the **FM1** and is reflected rotated by 90°, meaning that it follows to **FM3** and it is reflected at its original state. At the end, the two components recombine in **PSB1** and the single photon pulse follows its path with a polarization state defined by the phase shift applied in **PM**.

In table 5.16 is present the phases shifts that must be applied at Phase Modulator in order to get the polarized photons chose by Alice.

In table 5.17 are described all components needed to build the experimental setup.

Tabela 5.16: Different Alice's output polarization states based on shift phases applied in Phase Modulator.

Alice: (ϕ_1, ϕ_2)	Output Polarization
$(\frac{\pi}{2}, 0)$	Vertical
$(\frac{\pi}{2}, \frac{3\pi}{2})$	Horizontal
$(0, 0)$	-45°
$(0, \frac{\pi}{2})$	45°

Tabela 5.17: List of material

Material Name	Quantity	Status
Laser semiconductor 1550nm	1	✓
Manual polarization controller	5	
Faraday Mirror (FM)	6	
Mach-Zehnder Modulator	1	✓ 2.56GHz
Single Photon Detector	2	✓
Phase modulator	2	
Four-port polarization beam splitter	2	
Three-port polarization beam splitter	1	
Half-wave plate	1	
Optical circulator	2	
Variable Optical Attenuator	1	✓
Computer	1	

5.9.4 Comparative Analysis

Bibliografia

- [1] Thorlabs. *Thorlabs Balance Amplified Photodetectors: PDB4xx Series Operation Manual*, 2014.
- [2] Xinping Huang, Mario Caron, and Dan Hindson. A recursive Gram-Schmidt orthonormalization procedure and its application to communications. In *Wireless Communications, 2001.(SPAWC'01). 2001 IEEE Third Workshop on Signal Processing Advances in*, pages 340–343. IEEE, 2001.
- [3] Tie-Ming Liu, Lie-hui Jiang, Hong-qi He, Ji-zhong Li, and Xian Yu. *Researching on Cryptographic Algorithm Recognition Based on Static Characteristic-Code*, pages 140–147. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [4] Gilles Brassard and Louis Salvail. *Secret-Key Reconciliation by Public Discussion*, pages 410–423. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.
- [5] Min Xu, Run-hua Shi, Zhen-yu Luo, and Zhen-wan Peng. Nearest private query based on quantum oblivious key distribution. *Quantum Information Processing*, 16(12):286, Oct 2017.
- [6] Álvaro J Almeida, Nelson J Muga, Nuno A Silva, João M Prata, Paulo S André, and Armando N Pinto. Continuous control of random polarization rotations for quantum communications. *Journal of Lightwave Technology*, 34(16):3914–3922, 2016.

Capítulo 6

Library

6.1 Add

Input Parameters

This block takes no parameters.

Functional Description

This block accepts two signals and outputs one signal built from a sum of the two inputs. The input and output signals must be of the same type, if this is not the case the block returns an error.

Input Signals

Number: 2

Type: Real, Complex or Complex_XY signal (ContinuousTimeContinuousAmplitude)

Output Signals

Number: 1

Type: Real, Complex or Complex_XY signal (ContinuousTimeContinuousAmplitude)

6.2 Bit Error Rate

Header File	:	bit_error_rate.h
Source File	:	bit_error_rate.cpp
Version	:	20171810 (Responsible: Daniel Pereira)

Input Parameters

Name	Type	Default Value
Confidence	double	0.95
MidReportSize	integer	0
LowestMinorant	double	1×10^{-10}

Input Signals

Number: 2

Type: Binary (DiscreteTimeDiscreteAmplitude)

Output Signals

Number: 1

Type: Binary (DiscreteTimeDiscreteAmplitude)

Functional Description

This block accepts two binary strings and outputs a binary string, outputting a 1 if the two input samples are equal to each other and 0 if not. This block also outputs .txt files with a report of the estimated Bit Error Rate (BER), $\widehat{\text{BER}}$ as well as the estimated confidence bounds for a given probability α .

The block allows for mid-reports to be generated, the number of bits between reports is customizable, if it is set to 0 then the block will only output the final report.

Theoretical Description

The $\widehat{\text{BER}}$ is obtained by counting both the total number received bits, N_T , and the number of coincidences, K , and calculating their relative ratio:

$$\widehat{\text{BER}} = 1 - \frac{K}{N_T}. \quad (6.1)$$

The upper and lower bounds, BER_{UB} and BER_{LB} respectively, are calculated using the Clopper-Pearson confidence interval, which returns the following simplified expression for

$N_T > 40$ [6]:

$$\text{BER}_{\text{UB}} = \widehat{\text{BER}} + \frac{1}{\sqrt{N_T}} z_{\alpha/2} \sqrt{\widehat{\text{BER}}(1 - \widehat{\text{BER}})} + \frac{1}{3N_T} \left[2 \left(\frac{1}{2} - \widehat{\text{BER}} \right) z_{\alpha/2}^2 + (2 - \widehat{\text{BER}}) \right] \quad (6.2)$$

$$\text{BER}_{\text{LB}} = \widehat{\text{BER}} - \frac{1}{\sqrt{N_T}} z_{\alpha/2} \sqrt{\widehat{\text{BER}}(1 - \widehat{\text{BER}})} + \frac{1}{3N_T} \left[2 \left(\frac{1}{2} - \widehat{\text{BER}} \right) z_{\alpha/2}^2 - (1 + \widehat{\text{BER}}) \right], \quad (6.3)$$

where $z_{\alpha/2}$ is the $100(1 - \frac{\alpha}{2})$ th percentile of a standard normal distribution.

Bibliografia

- [1] Thorlabs. *Thorlabs Balance Amplified Photodetectors: PDB4xx Series Operation Manual*, 2014.
- [2] Xinping Huang, Mario Caron, and Dan Hindson. A recursive Gram-Schmidt orthonormalization procedure and its application to communications. In *Wireless Communications, 2001.(SPAWC'01). 2001 IEEE Third Workshop on Signal Processing Advances in*, pages 340–343. IEEE, 2001.
- [3] Tie-Ming Liu, Lie-hui Jiang, Hong-qi He, Ji-zhong Li, and Xian Yu. *Researching on Cryptographic Algorithm Recognition Based on Static Characteristic-Code*, pages 140–147. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [4] Gilles Brassard and Louis Salvail. *Secret-Key Reconciliation by Public Discussion*, pages 410–423. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.
- [5] Min Xu, Run-hua Shi, Zhen-yu Luo, and Zhen-wan Peng. Nearest private query based on quantum oblivious key distribution. *Quantum Information Processing*, 16(12):286, Oct 2017.
- [6] Álvaro J Almeida, Nelson J Muga, Nuno A Silva, João M Prata, Paulo S André, and Armando N Pinto. Continuous control of random polarization rotations for quantum communications. *Journal of Lightwave Technology*, 34(16):3914–3922, 2016.

6.3 Binary Source

Header File	:	binary_source.h
Source File	:	binary_source.cpp

This block generates a sequence of binary values (1 or 0) and it can work in four different modes:

- | | |
|-----------------|-----------------------------|
| 1. Random | 3. DeterministicCyclic |
| 2. PseudoRandom | 4. DeterministicAppendZeros |

This blocks doesn't accept any input signal. It produces any number of output signals.

Input Parameters

Parameter	Type	Values	Default
mode	string	Random, PseudoRandom, DeterministicCyclic, DeterministicAppendZeros	PseudoRandom
probabilityOfZero	real	$\in [0,1]$	0.5
patternLength	int	Any natural number	7
bitStream	string	sequence of 0's and 1's	0100011101010101
numberOfBits	long int	any	-1
bitPeriod	double	any	1.0/100e9

Tabela 6.1: Binary source input parameters

Methods

```
BinarySource(vector<Signal *> &InputSig, vector<Signal *> &OutputSig) :Block(InputSig, OutputSig){};
```

```
void initialize(void);
```

```
bool runBlock(void);
```

```
void setMode(BinarySourceMode m) BinarySourceMode const getMode(void)
```

```
void setProbabilityOfZero(double pZero)
```

```
double const getProbabilityOfZero(void)
```

```
void setBitStream(string bStream)
```

```

string const getBitStream(void)

void setNumberOfBits(long int nOfBits)

long int const getNumberOfBits(void)

void setPatternLength(int pLength)

int const getPatternLength(void)

void setBitPeriod(double bPeriod)

double const getBitPeriod(void)

```

Functional description

The *mode* parameter allows the user to select between one of the four operation modes of the binary source.

Random Mode Generates a 0 with probability *probabilityOfZero* and a 1 with probability $1 - \text{probabilityOfZero}$.

Pseudorandom Mode Generates a pseudorandom sequence with period $2^{\text{patternLength}} - 1$.

DeterministicCyclic Mode Generates the sequence of 0's and 1's specified by *bitStream* and then repeats it.

DeterministicAppendZeros Mode Generates the sequence of 0's and 1's specified by *bitStream* and then it fills the rest of the buffer space with zeros.

Input Signals

Number: 0

Type: Binary (DiscreteTimeDiscreteAmplitude)

Output Signals

Number: 1 or more

Type: Binary (DiscreteTimeDiscreteAmplitude)

Examples

Random Mode

PseudoRandom Mode As an example consider a pseudorandom sequence with *patternLength*=3 which contains a total of 7 ($2^3 - 1$) bits. In this sequence it is possible to find every combination of 0's and 1's that compose a 3 bit long subsequence with the exception of 000. For this example the possible subsequences are 010, 110, 101, 100, 111, 001 and 100 (they appear in figure 6.1 numbered in this order). Some of these require wrap.

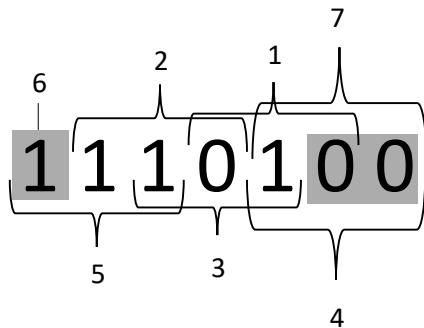


Figura 6.1: Example of a pseudorandom sequence with a pattern length equal to 3.

DeterministicCyclic Mode As an example take the *bit stream* '0100011101010101'. The generated binary signal is displayed in.

DeterministicAppendZeros Mode Take as an example the *bit stream* '0100011101010101'. The generated binary signal is displayed in 6.2.

Sugestions for future improvement

Implement an input signal that can work as trigger.

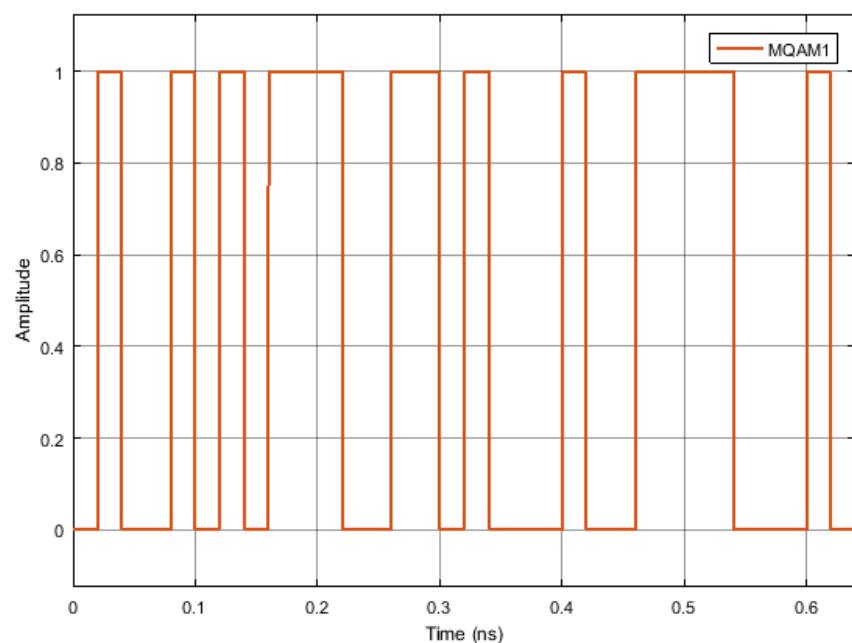


Figura 6.2: Binary signal generated by the block operating in the *Deterministic Append Zeros* mode with a binary sequence 01000...

6.4 Bit Decider

Input Parameters

Parameter: setPosReferenceValue

Parameter: setNegReferenceValue

Functional Description

This block accepts one real discrete signal and outputs a binary string, outputting a 1 if the input sample is above the predetermined reference level and 0 if it is below another reference value. The reference values are defined by the values of *PosReferenceValue* and *NegReferenceValue*.

Input Signals

Number: 1

Type: Real signal (DiscreteTimeContinuousAmplitude)

Output Signals

Number: 1

Type: Binary (DiscreteTimeDiscreteAmplitude)

6.5 Clock

Header File	:	clock.h
Source File	:	clock.cpp

This block doesn't accept any input signal. It outputs one signal that corresponds to a sequence of Dirac's delta functions with a user defined *period*.

Input Parameters

Parameter	Type	Values	Default
period	double	any	0.0
samplingPeriod	double	any	0.0

Tabela 6.2: Binary source input parameters

Methods

Clock()

```
Clock(vector<Signal *> &InputSig, vector<Signal *> &OutputSig) :Block(InputSig,
OutputSig)
```

```
void initialize(void)
```

```
bool runBlock(void)
```

```
void setClockPeriod(double per)
```

```
void setSamplingPeriod(double sPeriod)
```

Functional description

Input Signals

Number: 0

Output Signals

Number: 1

Type: Sequence of Dirac's delta functions.
(TimeContinuousAmplitudeContinuousReal)

Examples

Sugestions for future improvement

6.6 Clock_20171219

This block doesn't accept any input signal. It outputs one signal that corresponds to a sequence of Dirac's delta functions with a user defined *period*, *phase* and *sampling period*.

Input Parameters

Parameter: period{ 0.0 };

Parameter: samplingPeriod{ 0.0 };

Parameter: phase {0.0};

Methods

Clock()

```
Clock(vector<Signal *> &InputSig, vector<Signal *> &OutputSig) :Block(InputSig,  
OutputSig)
```

```
void initialize(void)
```

```
bool runBlock(void)
```

```
void setClockPeriod(double per) double getClockPeriod()
```

```
void setClockPhase(double pha) double getClockPhase()
```

```
void setSamplingPeriod(double sPeriod) double getSamplingPeriod()
```

Functional description

Input Signals

Number: 0

Output Signals

Number: 1

Type: Sequence of Dirac's delta functions.
(TimeContinuousAmplitudeContinuousReal)

Examples

Sugestions for future improvement

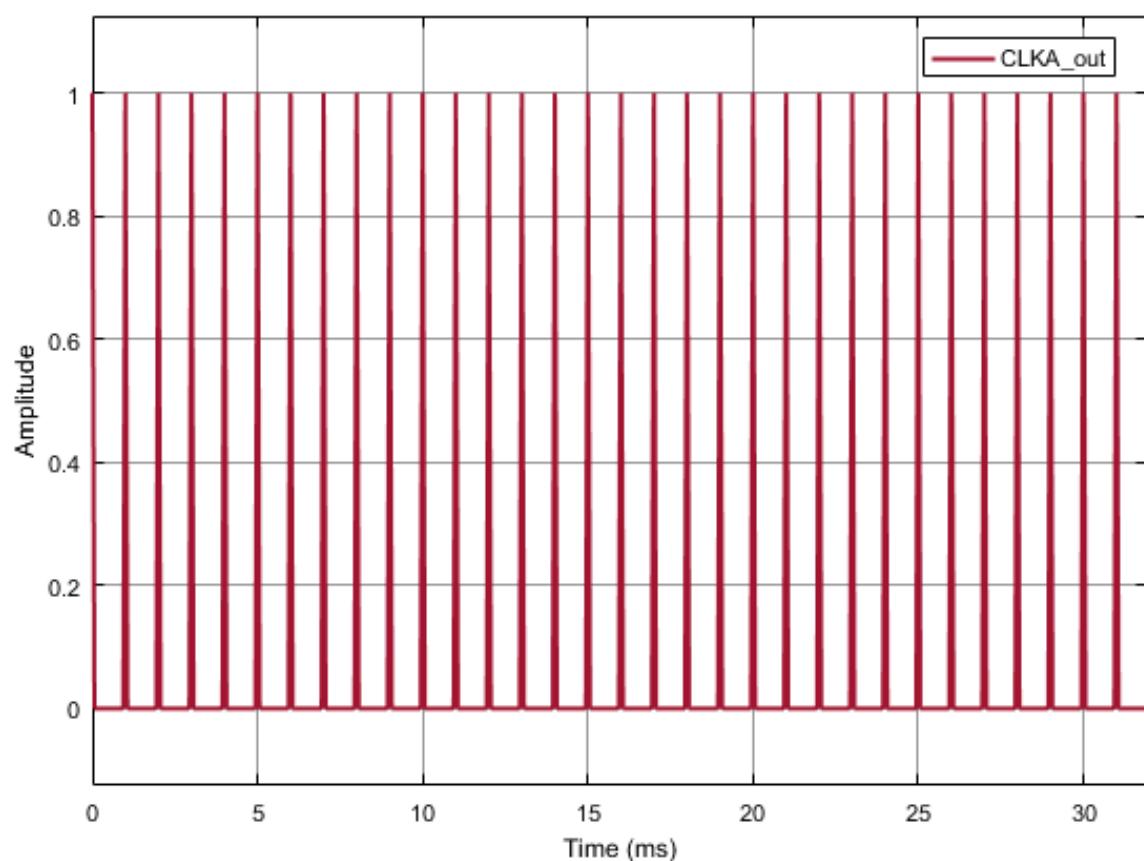


Figura 6.3: Example of the output signal of the clock without phase shift.

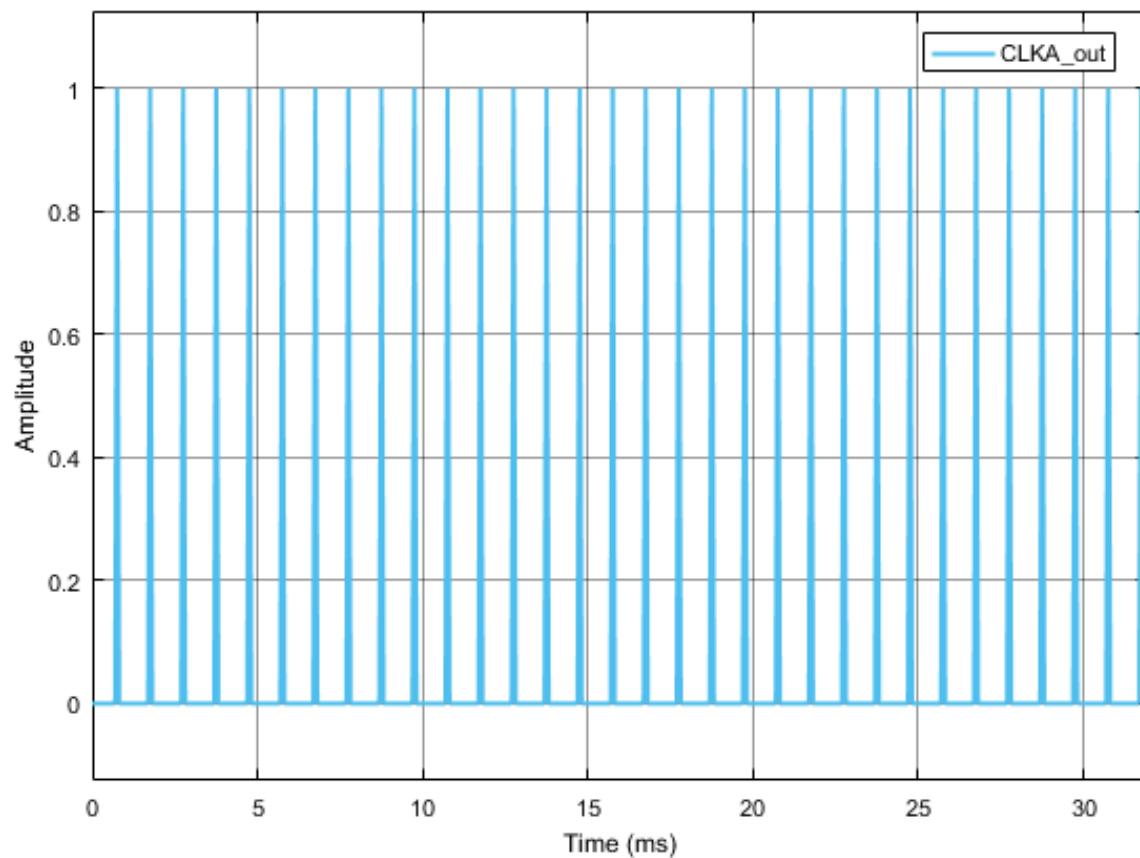


Figura 6.4: Example of the output signal of the clock with phase shift.

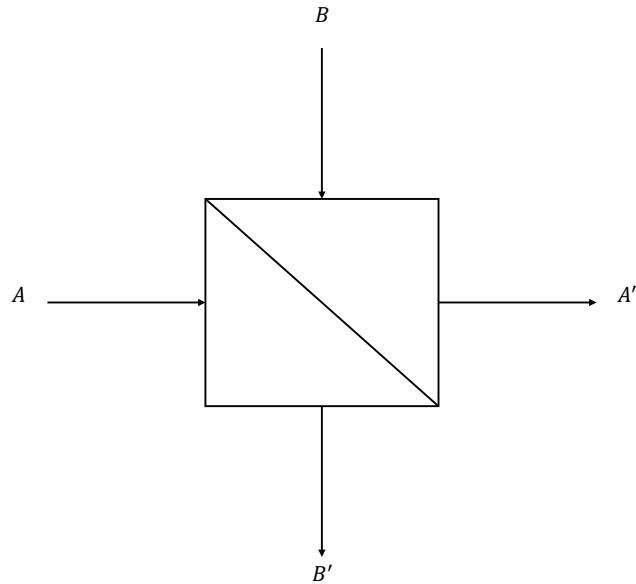


Figura 6.5: 2x2 coupler

6.7 Coupler 2 by 2

In general, the matrix representing 2x2 coupler can be summarized in the following way,

$$\begin{bmatrix} A' \\ B' \end{bmatrix} = \begin{bmatrix} T & iR \\ iR & T \end{bmatrix} \cdot \begin{bmatrix} A \\ B \end{bmatrix} \quad (6.4)$$

Where, A and B represent inputs to the 2x2 coupler and A' and B' represent output of the 2x2 coupler. Parameters T and R represent transmitted and reflected part respectively which can be quantified in the following form,

$$T = \sqrt{1 - \eta_R} \quad (6.5)$$

$$R = \sqrt{\eta_R} \quad (6.6)$$

Where, value of the $\sqrt{\eta_R}$ lies in the range of $0 \leq \sqrt{\eta_R} \leq 1$.

It is worth to mention that if we put $\eta_R = 1/2$ then it leads to a special case of "Balanced Beam splitter" which equally distribute the input power into both output ports.

6.8 Decoder

Header File	:	decoder.h
Source File	:	decoder.cpp

This block accepts a complex electrical signal and outputs a sequence of binary values (0's and 1's). Each point of the input signal corresponds to a pair of bits.

Input Parameters

Parameter	Type	Values	Default
m	int	≥ 4	4
iqAmplitudes	vector<t_complex>	—	{ { 1.0, 1.0 }, { -1.0, 1.0 }, { -1.0, -1.0 }, { 1.0, -1.0 } }

Tabela 6.3: Binary source input parameters

Methods

Decoder()

```
Decoder(vector<Signal *> &InputSig, vector<Signal *> &OutputSig) :Block(InputSig,
OutputSig)
```

```
void initialize(void)
```

```
bool runBlock(void)
```

```
void setM(int mValue)
```

```
void getM()
```

```
void setIqAmplitudes(vector<t_iqValues> iqAmplitudesValues)
```

```
vector<t_iqValues>getIqAmplitudes()
```

Functional description

This block makes the correspondence between a complex electrical signal and pair of binary values using a predetermined constellation.

To do so it computes the distance in the complex plane between each value of the input signal and each value of the *iqAmplitudes* vector selecting only the shortest one. It then converts the point in the IQ plane to a pair of bits making the correspondence between the input signal and a pair of bits.

Input Signals

Number: 1

Type: Electrical complex (TimeContinuousAmplitudeContinuousReal)

Output Signals

Number: 1

Type: Binary

Examples

As an example take an input signal with positive real and imaginary parts. It would correspond to the first point of the *iqAmplitudes* vector and therefore it would be associated to the pair of bits 00.

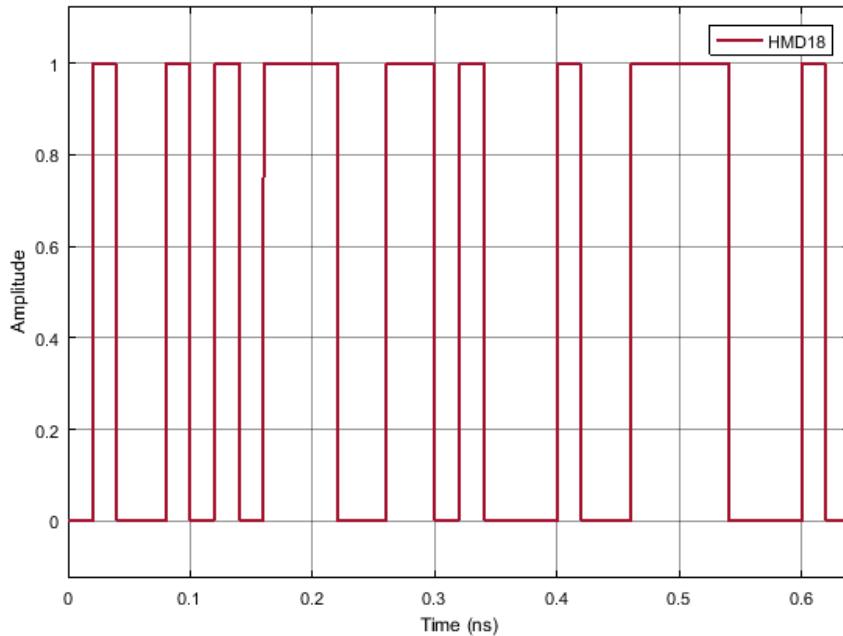


Figura 6.6: Example of the output signal of the decoder for a binary sequence 01. As expected it reproduces the initial bit stream

Sugestions for future improvement

6.9 Discrete To Continuous Time

Header File	:	discrete_to_continuous_time.h
Source File	:	discrete_to_continuous_time.cpp

This block converts a signal discrete in time to a signal continuous in time. It accepts one input signal that is a sequence of 1's and -1's and it produces one output signal that is a sequence of Dirac delta functions.

Input Parameters

Parameter	Type	Values	Default
numberOfSamplesPerSymbol	int	any	8

Tabela 6.4: Binary source input parameters

Methods

```
DiscreteToContinuousTime(vector<Signal * > &inputSignals, vector<Signal * > &outputSignals) :Block(inputSignals, outputSignals){};

void initialize(void);

bool runBlock(void);

void setNumberOfSamplesPerSymbol(int nSamplesPerSymbol)

int const getNumberOfSamplesPerSymbol(void)
```

Functional Description

This block reads the input signal buffer value, puts it in the output signal buffer and it fills the rest of the space available for that symbol with zeros. The space available in the buffer for each symbol is given by the parameter *numberOfSamplesPerSymbol*.

Input Signals

Number : 1

Type : Sequence of 1's and -1's. (DiscreteTimeDiscreteAmplitude)

Output Signals

Number : 1

Type : Sequence of Dirac delta functions (ContinuousTimeDiscreteAmplitude)

Example

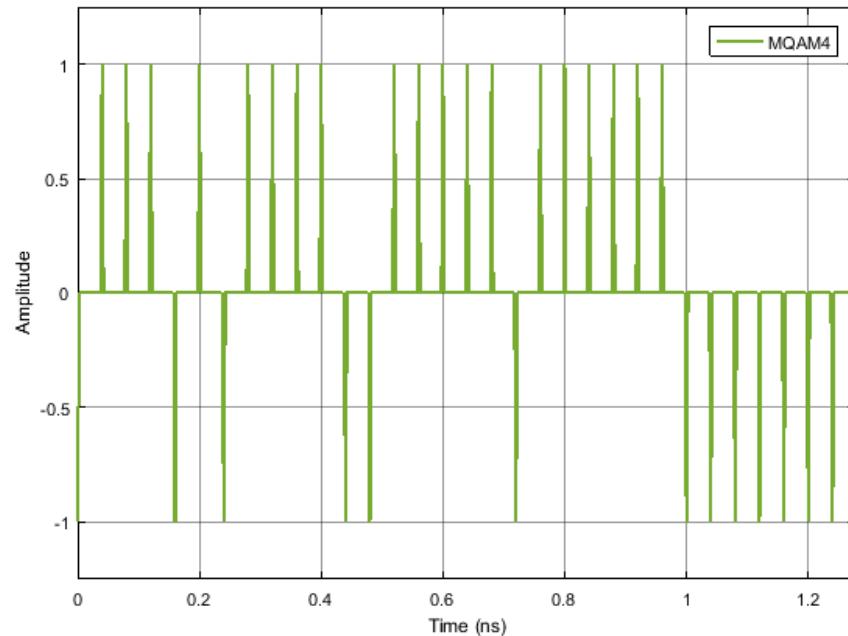


Figura 6.7: Example of the type of signal generated by this block for a binary sequence 0100...

6.10 Fork

Header File	:	fork_20171119.h
Source File	:	fork_20171119.cpp
Version	:	20171119 (Student Name: Romil Patel)

Input Parameters

— NA —

Input Signals

Number: 1

Type: Any type (BinaryValue, IntegerValue, RealValue, ComplexValue, ComplexValueXY, PhotonValue, PhotonValueMP, Message)

Output Signals

Number: 2

Type: Same as applied to the input.

Number: 3

Type: Same as applied to the input.

Functional Description

This block accepts any type signal and outputs two replicas of the input signal.

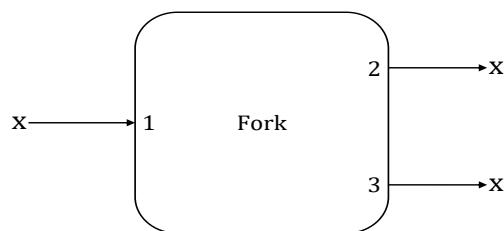


Figura 6.8: Fork

6.11 MQAM Receiver

Header File	:	m_qam_receiver.h
Source File	:	m_qam_receiver.cpp

Warning: *homodyne_receiver* is not recommended. Use *m_qam_homodyne_receiver* instead.

This block of code simulates the reception and demodulation of an optical signal (which is the input signal of the system) outputing a binary signal. A simplified schematic representation of this block is shown in figure 6.9.

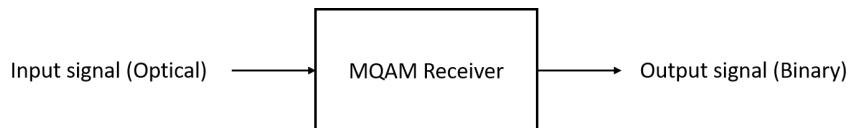


Figura 6.9: Basic configuration of the MQAM receiver

Functional description

This block accepts one optical input signal and outputs one binary signal that corresponds to the M-QAM demodulation of the input signal. It is a complex block (as it can be seen from figure 6.10) of code made up of several simpler blocks whose description can be found in the *lib* repository.

In can also be seen from figure 6.10 that there's an extra internal (generated inside the homodyne receiver block) input signal generated by the *Clock*. This block is used to provide the sampling frequency to the *Sampler*.

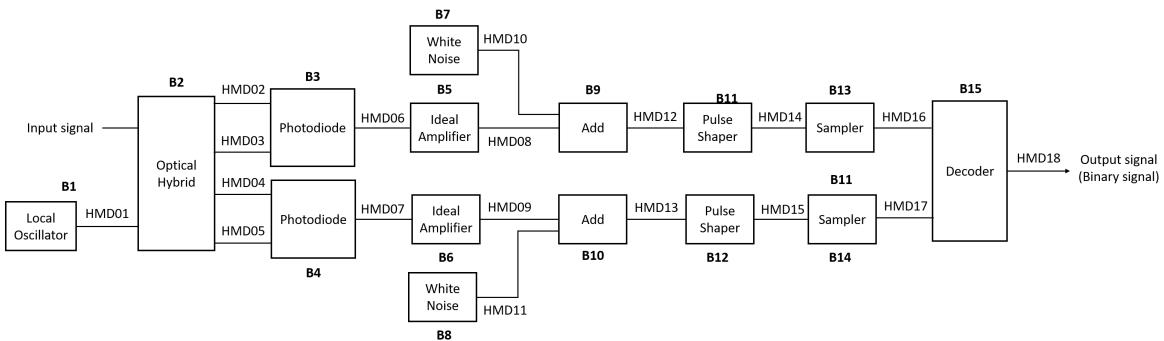


Figura 6.10: Schematic representation of the block homodyne receiver.

Input parameters

This block has some input parameters that can be manipulated by the user in order to change the basic configuration of the receiver. Each parameter has associated a function that allows for its change. In the following table (table 6.9) the input parameters and corresponding functions are summarized.

Input parameters	Function	Type	Accepted values
IQ amplitudes	setIqAmplitudes	Vector of coordinate points in the I-Q plane	Example for a 4-QAM mapping: { { 1.0, 1.0 }, { -1.0, 1.0 }, { -1.0, -1.0 }, { 1.0, -1.0 } }
Local oscillator power (in dBm)	setLocalOscillatorOpticalPower_dBm	double(t_real)	Any double greater than zero
Local oscillator phase	setLocalOscillatorPhase	double(t_real)	Any double greater than zero
Responsivity of the photodiodes	setResponsivity	double(t_real)	$\in [0,1]$
Amplification (of the TI amplifier)	setAmplification	double(t_real)	Positive real number
Noise amplitude (introduced by the TI amplifier)	setNoiseAmplitude	double(t_real)	Real number greater than zero
Samples to skip	setSamplesToSkip	int(t_integer)	
Save internal signals	setSaveInternalSignals	bool	True or False
Sampling period	setSamplingPeriod	double	Given by $symbolPeriod / samplesPerSymbol$

Tabela 6.5: List of input parameters of the block MQAM receiver

Methods

HomodyneReceiver(vector<Signal *> &inputSignal, vector<Signal *> &outputSignal)
(constructor)

```
void setIqAmplitudes(vector<t_iqValues> iqAmplitudesValues)
vector<t_iqValues> const getIqAmplitudes(void)
void setLocalOscillatorSamplingPeriod(double sPeriod)
void setLocalOscillatorOpticalPower(double opticalPower)
void setLocalOscillatorOpticalPower_dBm(double opticalPower_dBm)
void setLocalOscillatorPhase(double lOscillatorPhase)
void setLocalOscillatorOpticalWavelength(double lOscillatorWavelength)
void setSamplingPeriod(double sPeriod)
void setResponsivity(t_real Responsivity)
void setAmplification(t_real Amplification)
void setNoiseAmplitude(t_real NoiseAmplitude)
void setImpulseResponseTimeLength(int impResponseTimeLength)
void setFilterType(PulseShaperFilter fType)
void setRollOffFactor(double rOffFactor)
void setClockPeriod(double per)
void setSamplesToSkip(int sToSkip)
```

Input Signals

Number: 1

Type: Optical signal

Output Signals

Number: 1

Type: Binary signal

Example

Sugestions for future improvement

6.12 IQ Modulator

Header File	:	iq_modulator.h
Source File	:	iq_modulator.cpp

This blocks accepts one input signal continuous in both time and amplitude and it can produce either one or two output signals. It generates an optical signal and it can also generate a binary signal.

Input Parameters

Parameter	Type	Values	Default
outputOpticalPower	double	any	$1e - 3$
outputOpticalWavelength	double	any	$1550e - 9$
outputOpticalFrequency	double	any	speed_of_light/outputOpticalWavelength

Tabela 6.6: Binary source input parameters

Methods

```
IqModulator(vector<Signal *> &InputSig, vector<Signal *> &OutputSig) :Block(InputSig, OutputSig){};
```

```
void initialize(void);
bool runBlock(void);
void setOutputOpticalPower(double outOpticalPower)
void setOutputOpticalPower_dBm(double outOpticalPower_dBm)
void setOutputOpticalWavelength(double outOpticalWavelength)
void setOutputOpticalFrequency(double outOpticalFrequency)
```

Functional Description

This block takes the two parts of the signal: in phase and in amplitude and it combines them to produce a complex signal that contains information about the amplitude and the phase.

This complex signal is multiplied by $\frac{1}{2}\sqrt{outputOpticalPower}$ in order to reintroduce the information about the energy (or power) of the signal. This signal corresponds to an optical signal and it can be a scalar or have two polarizations along perpendicular axis. It is the signal that is transmitted to the receptor.

The binary signal is sent to the Bit Error Rate (BER) measurement block.

Input Signals

Number : 2

Type : Sequence of impulses modulated by the filter
(ContinuousTimeContiousAmplitude)

Output Signals

Number : 1 or 2

Type : Complex signal (optical) (ContinuousTimeContinuousAmplitude) and binary signal (DiscreteTimeDiscreteAmplitude)

Example

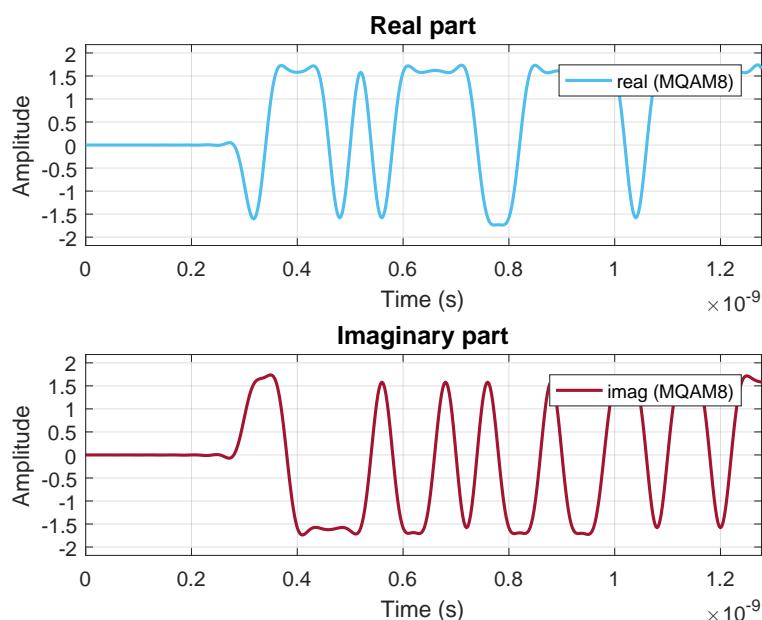


Figura 6.11: Example of a signal generated by this block for the initial binary signal 0100...

6.13 Local Oscillator

Header File	:	local_oscillator.h
Source File	:	local_oscillator.cpp

This block simulates a local oscillator with constant power and initial phase. It produces one output complex signal and it doesn't accept input signals.

Input Parameters

Parameter	Type	Values	Default
opticalPower	double	any	1e - 3
outputOpticalWavelength	double	any	1550e - 9
outputOpticalFrequency	double	any	SPEED_OF_LIGHT / outputOpticalWavelength
phase	double	$\in [0, \frac{\pi}{2}]$	0
samplingPeriod	double	any	0.0

Tabela 6.7: Binary source input parameters

Methods

LocalOscillator()

```
LocalOscillator(vector<Signal * > &InputSig, vector<Signal * > &OutputSig)
:Block(InputSig, OutputSig){};
```

```
void initialize(void);
bool runBlock(void);
void setSamplingPeriod(double sPeriod);
void setOpticalPower(double oPower);
void setOpticalPower_dBm(double oPower_dBm);
void setWavelength(double wlenght);
void setPhase(double lOscillatorPhase);
```

Functional description

This block generates a complex signal with a specified phase given by the input parameter *phase*.

Input Signals

Number: 0

Output Signals

Number: 1

Type: Optical signal

Examples

Sugestions for future improvement

6.14 MQAM Mapper

Header File	:	m_qam_mapper.h
Source File	:	m_qam_mapper.cpp

This block does the mapping of the binary signal using a m -QAM modulation. It accepts one input signal of the binary type and it produces two output signals which are a sequence of 1's and -1's.

Input Parameters

Parameter	Type	Values	Default
m	int	2^n with n integer	4
iqAmplitudes	vector<t_complex>	—	{ { 1.0, 1.0 }, { -1.0, 1.0 }, { -1.0, -1.0 }, { 1.0, -1.0 } }

Tabela 6.8: Binary source input parameters

Methods

```
MQamMapper(vector<Signal * > &InputSig, vector<Signal * > &OutputSig)
:Block(InputSig, OutputSig) {}

void initialize(void);

bool runBlock(void);

void setM(int mValue);

void setIqAmplitudes(vector<t_iqValues> iqAmplitudesValues);
```

Functional Description

In the case of $m=4$ this block attributes to each pair of bits a point in the I-Q space. The constellation used is defined by the *iqAmplitudes* vector. The constellation used in this case is illustrated in figure 6.12.

Input Signals

Number : 1

Type : Binary (DiscreteTimeDiscreteAmplitude)

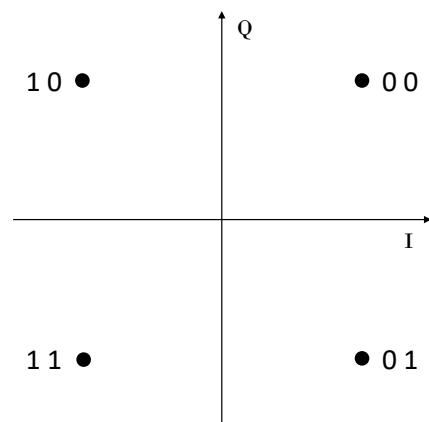


Figura 6.12: Constellation used to map the signal for $m=4$

Output Signals

Number : 2

Type : Sequence of 1's and -1's (DiscreteTimeDiscreteAmplitude)

Example

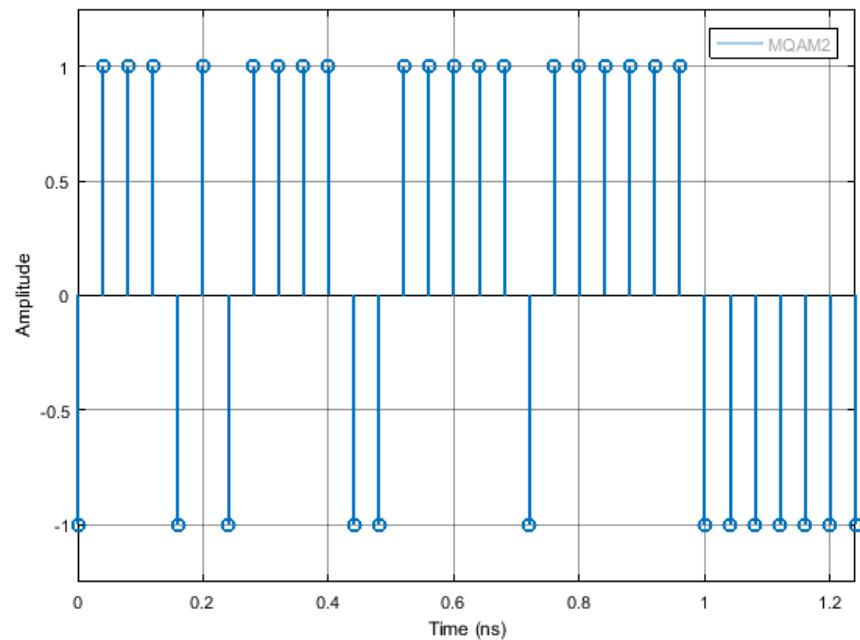


Figura 6.13: Example of the type of signal generated by this block for the initial binary signal 0100...

6.15 MQAM Transmitter

Header File	:	m_qam_transmitter.h
Source File	:	m_qam_transmitter.cpp

This block generates a MQAM optical signal. It can also output the binary sequence. A schematic representation of this block is shown in figure 6.14.

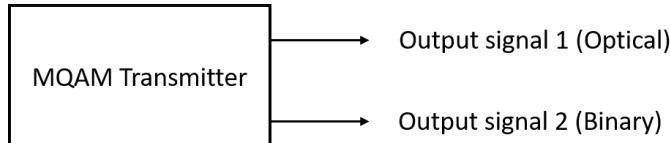


Figura 6.14: Basic configuration of the MQAM transmitter

Functional description

This block generates an optical signal (output signal 1 in figure 6.15). The binary signal generated in the internal block Binary Source (block B1 in figure 6.15) can be used to perform a Bit Error Rate (BER) measurement and in that sense it works as an extra output signal (output signal 2 in figure 6.15).

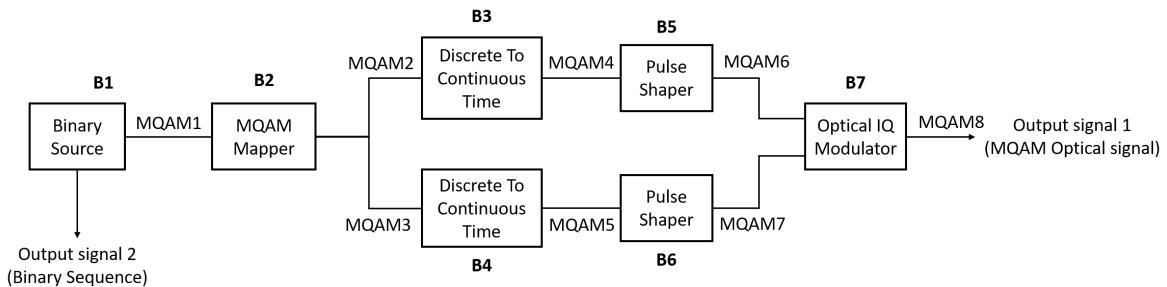


Figura 6.15: Schematic representation of the block MQAM transmitter.

Input parameters

This block has a special set of functions that allow the user to change the basic configuration of the transmitter. The list of input parameters, functions used to change them and the values that each one can take are summarized in table 6.9.

Input parameters	Function	Type	Accepted values
Mode	setMode()	string	PseudoRandom Random DeterministicAppendZeros DeterministicCyclic
Number of bits generated	setNumberOfBits()	int	Any integer
Pattern length	setPatternLength()	int	Real number greater than zero
Number of bits	setNumberOfBits()	long	Integer number greater than zero
Number of samples per symbol	setNumberOfSamplesPerSymbol()	int	Integer number of the type 2^n with n also integer
Roll off factor	setRollOffFactor()	double	$\in [0,1]$
IQ amplitudes	setIqAmplitudes()	Vector of coordinate points in the I-Q plane	Example for a 4-qam mapping: { { 1.0, 1.0 }, { -1.0, 1.0 }, { -1.0, -1.0 }, { 1.0, -1.0 } }
Output optical power	setOutputOpticalPower()	int	Real number greater than zero
Save internal signals	setSaveInternalSignals()	bool	True or False

Tabela 6.9: List of input parameters of the block MQAM transmitter

Methods

MQamTransmitter(vector<Signal *> &inputSignal, vector<Signal *> &outputSignal);
(constructor)

void set(int opt);

void setMode(BinarySourceMode m)

BinarySourceMode const getMode(void)

void setProbabilityOfZero(double pZero)

double const getProbabilityOfZero(void)

void setBitStream(string bStream)

```
string const getBitStream(void)

void setNumberOfBits(long int nOfBits)

long int const getNumberOfBits(void)

void setPatternLength(int pLength)

int const getPatternLength(void)

void setBitPeriod(double bPeriod)

double const getBitPeriod(void)

void setM(int mValue) int const getM(void)

void setIqAmplitudes(vector<t_iqValues> iqAmplitudesValues)

vector<t_iqValues> const getIqAmplitudes(void)

void setNumberOfSamplesPerSymbol(int n)

int const getNumberOfSamplesPerSymbol(void)

void setRollOffFactor(double rOffFactor)

double const getRollOffFactor(void)

void setSeeBeginningOfImpulseResponse(bool sBeginningOfImpulseResponse)

double const getSeeBeginningOfImpulseResponse(void)

void setOutputOpticalPower(t_real outOpticalPower)

t_real const getOutputOpticalPower(void)

void setOutputOpticalPower_dBm(t_real outOpticalPower_dBm)

t_real const getOutputOpticalPower_dBm(void)
```

Output Signals

Number: 1 optical and 1 binary (optional)

Type: Optical signal

Example

Sugestions for future improvement

Add to the system another block similar to this one in order to generate two optical signals with perpendicular polarizations. This would allow to combine the two optical signals and generate an optical signal with any type of polarization.

6.16 Alice QKD

This block is the processor for Alice does all tasks that she needs. This block accepts binary, messages, and real continuous time signals. It produces messages, binary and real discrete time signals.

Input Parameters

Parameter: double RateOfPhotons{1e3}

Parameter: int StringPhotonsLength{ 12 }

Methods

```
AliceQKD (vector <Signal*> &inputSignals, vector <Signal*> &outputSignals) :  
Block(inputSignals, outputSignals) {};  
void initialize(void);  
bool runBlock(void);  
void setRateOfPhotons(double RPhotons) { RateOfPhotons = RPhotons; }; double const  
getRateOfPhotons(void) { return RateOfPhotons; };  
void setStringPhotonsLength(int pLength) { StringPhotonsLength = pLength; }; int const  
getStringPhotonsLength(void) { return StringPhotonsLength; };
```

Functional description

This block receives a sequence of binary numbers (1's or 0's) and a clock signal which will set the rate of the signals produced to generate single polarized photons. The real discrete time signal **SA_1** is generated based on the clock signal and the real discrete time signal **SA_2** is generated based on the random sequence of bits received through the signal **NUM_A**. This last sequence is analysed by the polarizer in pairs of bits in which each pair has a bit for basis choice and other for direction choice.

This block also produces classical messages signals to send to Bob as well as binary messages to the mutual information block with information about the photons it sent.

Input Signals

Number : 3

Type : Binary, Real Continuous Time and Messages signals.

Output Signals

Number : 3

Type : Binary, Real Discrete Time and Messages signals.

Examples

Sugestions for future improvement

6.17 Polarizer

This block is responsible of changing the polarization of the input photon stream signal by using the information from the other real time discrete input signal. This way, this block accepts two input signals: one photon stream and other real discrete time signal. The real discrete time input signal must be a signal discrete in time in which the amplitude can be 0 or 1. The block will analyse the pairs of values by interpreting them as basis and polarization direction.

Input Parameters

Parameter: m{4}

Parameter: Amplitudes { {1,1}, {-1,1}, {-1,-1}, { 1,-1} }

Methods

```
Polarizer (vector <Signal*> &inputSignals, vector <Signal*>&outputSignals) :  
Block(inputSignals, outputSignals) {};  
void initialize(void);  
bool runBlock(void);  
void setM(int mValue);  
void setAmplitudes(vector <t_iqValues> AmplitudeValues);
```

Functional description

Considering m=4, this block attributes for each pair of bits a point in space. In this case, it is be considered four possible polarization states: 0° , 45° , 90° and 135° .

Input Signals

Number : 2

Type : Photon Stream and a Sequence of 0's and '1s (DiscreteTimeDiscreteAmplitude).

Output Signals

Number : 1

Type : Photon Stream

Examples

Sugestions for future improvement

6.18 Bob QKD

This block is the processor for Bob does all tasks that she needs. This block accepts and produces:

1.

2.

Input Parameters

Parameter:

Parameter:

Methods

Functional description

Input Signals

Examples

Sugestions for future improvement

6.19 Eve QKD

This block is the processor for Eve does all tasks that she needs. This block accepts and produces:

1.

2.

Input Parameters

Parameter:

Parameter:

Methods

Functional description

Input Signals

Examples

Sugestions for future improvement

6.20 Rotator Linear Polarizer

This block accepts a Photon Stream signal and a Real discrete time signal. It produces a photon stream by rotating the polarization axis of the linearly polarized input photon stream by an angle of choice.

Input Parameters

Parameter: m[2]

Parameter: axis { {1,0}, { $\frac{\sqrt{2}}{2}$, $\frac{\sqrt{2}}{2}$ } }

Methods

```
RotatorLinearPolarizer(vector <Signal*> &inputSignals, vector <Signal*> &outputSignals) :
    Block(inputSignals, outputSignals) {};
    void initialize(void);
    bool runBlock(void);
    void setM(int mValue);
    void setAxis(vector <t_iqValues> AxisValues);
```

Functional description

This block accepts the input parameter m, which defines the number of possible rotations. In this case m=2, the block accepts the rectilinear basis, defined by the first position of the second input parameter axis, and the diagonal basis, defined by the second position of the second input parameter axis. This block rotates the polarization axis of the linearly polarized input photon stream to the basis defined by the other input signal. If the discrete value of this signal is 0, the rotator is set to rotate the input photon stream by 0°, otherwise, if the value is 1, the rotator is set to rotate the input photon stream by an angle of 45°.

Input Signals

Number : 2

Type : Photon Stream and a Sequence of 0's and '1s (DiscreteTimeDiscreteAmplitude)

Output Signals

Number : 1

Type : Photon Stream

Examples

Sugestions for future improvement

6.21 Optical Switch

This block has one input signal and two input signals. Furthermore, it accepts an additional input binary input signal which is used to decide which of the two outputs is activated.

Input Parameters

No input parameters.

Methods

```
OpticalSwitch(vector <Signal*> &inputSignals, vector <Signal*> &outputSignals) :  
Block(inputSignals, outputSignals) {};  
void initialize(void);  
bool runBlock(void);
```

Functional description

This block receives an input photon stream signal and it decides which path the signal must follow. In order to make this decision it receives a binary signal (0's and 1's) and it switch the output path according with this signal.

Input Signals

Number : 1

Type : Photon Stream

Output Signals

Number : 2

Type : Photon Stream

Examples

Sugestions for future improvement

6.22 Optical Hybrid

Header File	:	optical_hybrid.h
Source File	:	optical_hybrid.cpp

This block simulates an optical hybrid. It accepts two input signals corresponding to the signal and to the local oscillator. It generates four output complex signals separated by 90° in the complex plane. Figure 6.16 shows a schematic representation of this block.

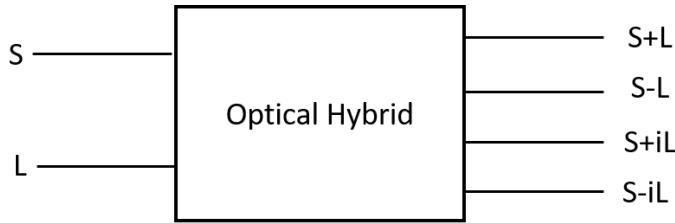


Figura 6.16: Schematic representation of an optical hybrid.

Input Parameters

Parameter	Type	Values	Default
outputOpticalPower	double	any	$1e - 3$
outputOpticalWavelength	double	any	$1550e - 9$
outputOpticalFrequency	double	any	SPEED_OF_LIGHT / outputOpticalWavelength
powerFactor	double	≤ 1	0.5

Tabela 6.10: Optical hybrid input parameters

Methods

OpticalHybrid()

```
OpticalHybrid(vector<Signal * > &InputSig, vector<Signal * > &OutputSig)
:Block(InputSig, OutputSig)
```

```
void initialize(void)
```

```
bool runBlock(void)
```

```
void setOutputOpticalPower(double outOpticalPower)
```

```
void setOutputOpticalPower_dBm(double outOpticalPower_dBm)
```

```
void setOutputOpticalWavelength(double outOpticalWavelength)  
void setOutputOpticalFrequency(double outOpticalFrequency)  
void setPowerFactor(double pFactor)
```

Functional description

This block accepts two input signals corresponding to the signal to be demodulated (S) and to the local oscillator (L). It generates four output optical signals given by $powerFactor \times (S + L)$, $powerFactor \times (S - L)$, $powerFactor \times (S + iL)$, $powerFactor \times (S - iL)$. The input parameter $powerFactor$ assures the conservation of optical power.

Input Signals

Number: 2

Type: Optical (OpticalSignal)

Output Signals

Number: 4

Type: Optical (OpticalSignal)

Examples

Sugestions for future improvement

6.23 Photodiode pair

Header File	:	photodiode_old.h
Source File	:	photodiode_old.cpp

This block simulates a block of two photodiodes assembled like in figure 6.17. It accepts two optical input signals and outputs one electrical signal. Each photodiode converts an optical signal to an electrical signal. The two electrical signals are then subtracted and the resulting signals corresponds to the output signal of the block.

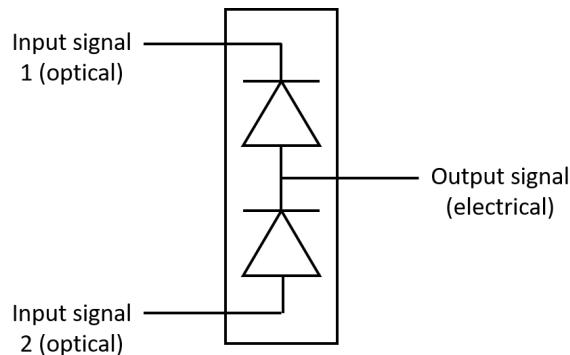


Figura 6.17: Schematic representation of the physical equivalent of the photodiode code block.

Input Parameters

Parameter: responsivity{1}

Parameter: outputOpticalWavelength{ 1550e-9 }

Parameter: outputOpticalFrequency{ SPEED_OF_LIGHT / wavelength }

Methods

Photodiode()

```
Photodiode(vector<Signal * > &InputSig, vector<Signal * > &OutputSig)
:Block(InputSig, OutputSig)
```

```
void initialize(void)
```

```
bool runBlock(void)
```

```
void setResponsivity(t_real Responsivity)
```

Functional description

This block accepts two input optical signals. It computes the optical power of the signal (given by the absolute value squared of the input signal) and multiplies it by the *responsivity* of the photodiode. This product corresponds to the current generated by the photodiode. This is done for each of the input signals. The two currents are then subtracted producing a single output current, that corresponds to the output electrical signal of the block.

Input Signals

Number: 2

Type: Optical (OpticalSignal)

Output Signals

Number: 1

Type: Electrical (TimeContinuousAmplitudeContinuousReal)

Examples

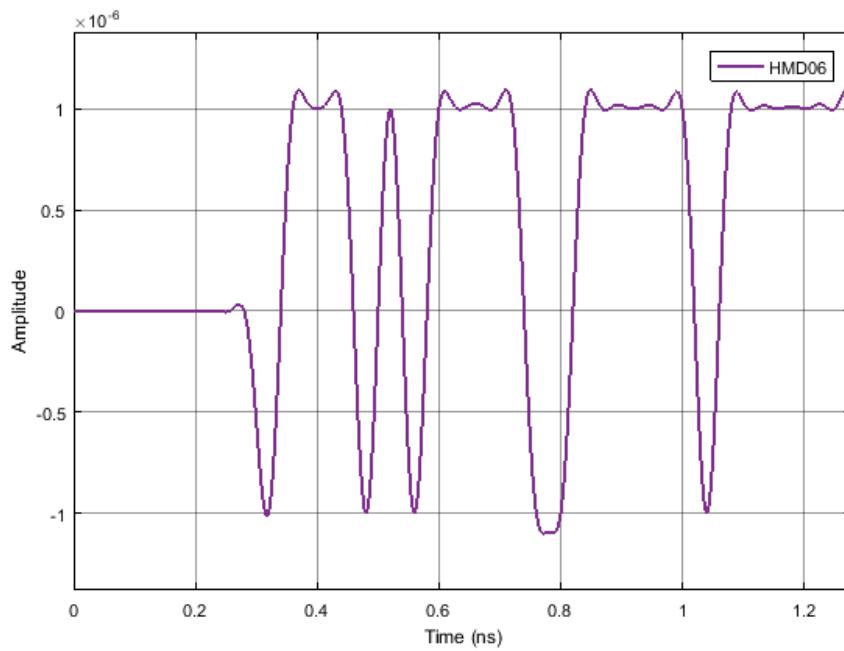


Figura 6.18: Example of the output singal of the photodiode block for a bunary sequence 01

Sugestions for future improvement

6.24 Pulse Shaper

Header File	:	pulse_shaper.h
Source File	:	pulse_shaper.cpp

This block applies an electrical filter to the signal. It accepts one input signal that is a sequence of Dirac delta functions and it produces one output signal continuous in time and in amplitude.

Input Parameters

Parameter	Type	Values	Default
filterType	string	RaisedCosine, Gaussian	RaisedCosine
impulseResponseTimeLength	int	any	16
rollOffFactor	real	$\in [0, 1]$	0.9

Tabela 6.11: Pulse shaper input parameters

Methods

```
PulseShaper(vector<Signal * > &InputSig, vector<Signal * > OutputSig)
:FIR_Filter(InputSig, OutputSig){};

void initialize(void);

void setImpulseResponseTimeLength(int impResponseTimeLength)

int const getImpulseResponseTimeLength(void)

void setFilterType(PulseShaperFilter fType)

PulseShaperFilter const getFilterType(void)

void setRollOffFactor(double rOffFactor)

double const getRollOffFactor()
```

Functional Description

The type of filter applied to the signal can be selected through the input parameter *filterType*. Currently the only available filter is a raised cosine.

The filter's transfer function is defined by the vector *impulseResponse*. The parameter *rollOffFactor* is a characteristic of the filter and is used to define its transfer function.

Input Signals

Number : 1

Type : Sequence of Dirac Delta functions (ContinuousTimeDiscreteAmplitude)

Output Signals

Number : 1

Type : Sequence of impulses modulated by the filter
(ContinuousTimeContinuousAmplitude)

Example

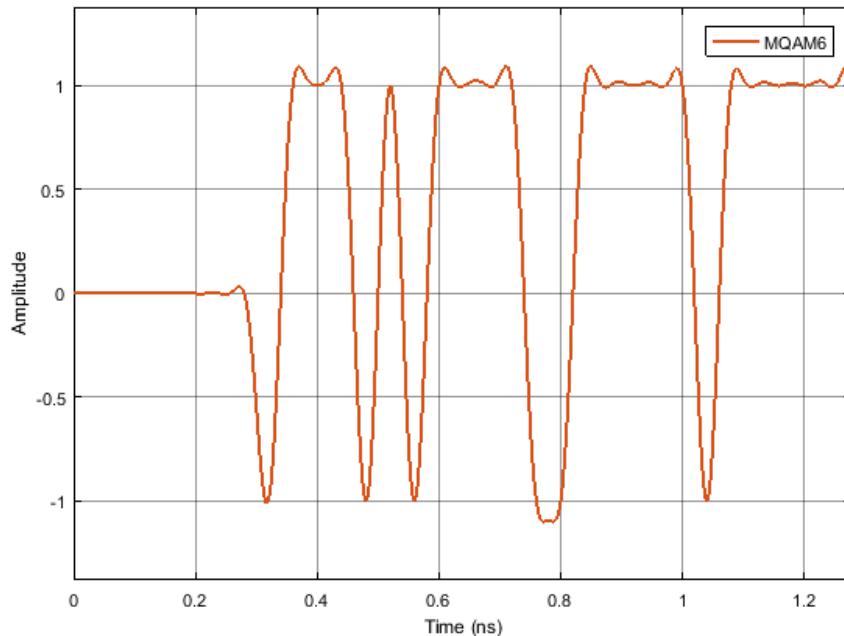


Figura 6.19: Example of a signal generated by this block for the initial binary signal 0100...

Sugestions for future improvement

Include other types of filters.

6.25 Sampler

Header File	:	sampler.h
Source File	:	sampler_20171119.cpp

This block can work in two configurations: with an external clock or without it. In the latter it accepts two input signals one being the clock and the other the signal to be demodulated. In the other configuration there's only one input signal which is the signal.

The output signal is obtained by sampling the input signal with a predetermined sampling rate provided either internally or by the clock.

Input Parameters

Parameter	Type	Values	Default
samplesToSkip	int	any (smaller than the number of samples generated)	0

Tabela 6.12: Sampler input parameters

Methods

Sampler()

```
Sampler(vector<Signal *> &InputSig, vector<Signal *> &OutputSig) :Block(InputSig,
OutputSig)
```

```
void initialize(void)
```

```
bool runBlock(void)
```

```
void setSamplesToSkip(t_integer sToSkip)
```

Functional description

This block can work with an external clock or without it.

In the case of having an external clock it accepts two input signals. The signal to be demodulate which is complex and a clock signal that is a sequence of Dirac delta functions with a predetermined period that corresponds to the sampling period. The signal and the clock signal are scanned and when the clock has the value of 1.0 the correspondent complex value of the signal is placed in the buffer corresponding to the output signal.

There's a detail worth noting. The electrical filter has an impulse response time length of 16 (in units of symbol period). This means that when modulating a bit the spike in the signal corresponding to that bit will appear 8 units of symbol period later. For this reason there's

the need to skip the earlier samples of the signal when demodulating it. That's the purpose of the *samplesToSkip* parameter.

Between the binary source and the current block the signal is filtered twice which means that this effect has to be taken into account twice. Therefore the parameter *samplesToSkip* is given by $2 * 8 * \text{samplesPerSymbol}$.

Input Signals

Number: 1

Type: Electrical real (TimeContinuousAmplitudeContinuousReal)

Output Signals

Number: 1

Type: Electrical real (TimeDiscreteAmplitudeContinuousReal)

Examples

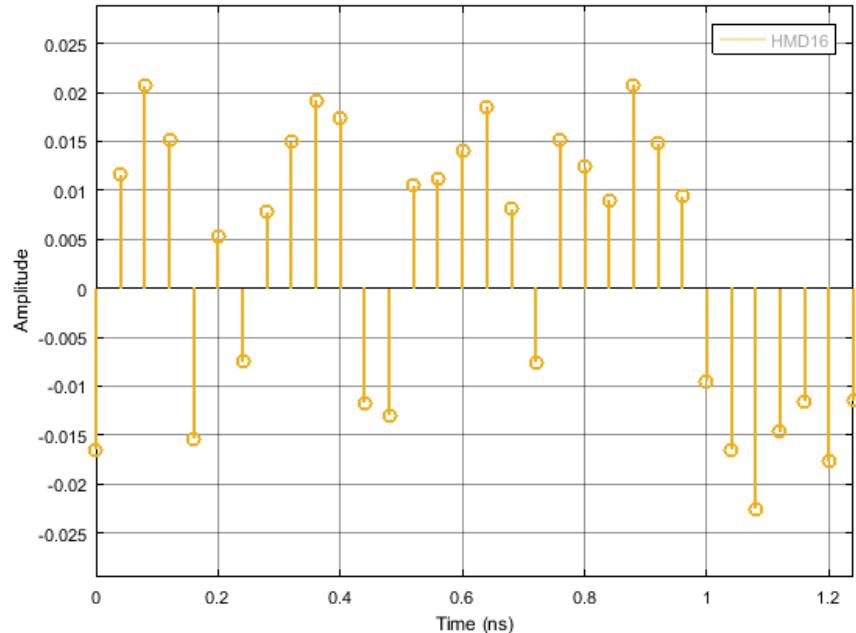


Figura 6.20: Example of the output signal of the sampler

Sugestions for future improvement

6.26 Sink

Header File	:	sink.h
Source File	:	sink.cpp

This block accepts one input signal and it does not produce output signals. It takes samples out of the buffer until the buffer is empty. It has the option of displaying the number of samples still available.

Input Parameters

Parameter	Type	Values	Default
numberOfSamples	long int	any	-1

Tabela 6.13: Sampler input parameters

Methods

Sink(vector<Signal *> &InputSig, vector<Signal *> &OutputSig)

bool runBlock(void)

void setNumberOfSamples(long int nOfSamples)

void setDisplayNumberOfSamples(bool opt)

Functional Description

Capítulo 7

Mathlab Tools

7.1 Generation of AWG Compatible Signals

Student Name	:	Francisco Marques dos Santos
Starting Date	:	September 1, 2017
Goal	:	Convert simulation signals into waveform files compatible with the laboratory's Arbitrary Waveform Generator
Directory	:	mtools

This section shows how to convert a simulation signal into an AWG compatible waveform file through the use of a matlab function called sgnToWfm. This allows the application of simulated signals into real world systems.

7.1.1 sgnToWfm_20171121

Structure of a function

```
[dataDecimate, data, symbolPeriod, samplingPeriod,
type, numberOfRowsInSection, samplingRate, samplingRateDecimate] = sgnToWfm_20171121
(fname_sgn, nReadr, fname_wfm)
```

Inputs

fname_sgn: Input filename of the signal (*.sgn) you want to convert. It must be a real signal (Type: TimeContinuousAmplitudeContinuousReal).

nReadr: Number of symbols you want to extract from the signal.

fname_wfm: Name that will be given to the waveform file.

Outputs

A waveform file will be created in the Matlab current folder. It will also return six variables in the workspace which are:

dataDecimate: A vector which contains decimated signal data by an appropriate decimation factor to make it compatible with the AWG.

data: A vector with the signal data.

symbolPeriod: Equal to the symbol period of the corresponding signal.

samplingPeriod: Sampling period of the signal.

type: A string with the name of the signal type.

numberOfSymbols: Number of symbols retrieved from the signal.

samplingRate: Sampling rate of the signal.

samplingRateDecimate: Reduced sampling rate which is compatible with AWG. (i.e. less than 16 GSa/s).

Functional Description

This matlab function generates a *.wfm file given an input signal file (*.sgn). The waveform file is compatible with the laboratory's Arbitrary Waveform Generator (Tektronix AWG70002A). In order to recreate it appropriately, the signal must be real, not exceed $8 * 10^9$ samples and have a sampling rate equal or bellow 16 GS/s.

This function can be called with one, two or three arguments:

Using one argument:

```
[dataDecimate, data, symbolPeriod, samplingPeriod, type, numberOfSymbols,
samplingRate, samplingRateDecimate] = sgnToWfm('S6.sgn');
```

This creates a waveform file with the same name as the *.sgn file and uses all of the samples it contains.

Using two arguments:

```
[dataDecimate, data, symbolPeriod, samplingPeriod, type, numberOfSymbols,
samplingRate, samplingRateDecimate] = sgnToWfm('S6.sgn',256);
```

This creates a waveform file with the same name as the signal file name and the number of samples used equals nReadr x samplesPerSymbol. The samplesPerSymbol constant is defined in the *.sgn file.

Using three arguments:

```
[dataDecimate, data, symbolPeriod, samplingPeriod, type, numberOfSymbols,
samplingRate, samplingRateDecimate] = sgnToWfm('S6.sgn',256,'myWaveform.wfm');
```

This creates a waveform file with the name "myWaveform"and the number of samples used equals nReadr x samplesPerSymbol. The samplesPerSymbol constant is defined in the *.sgn file.

7.1.2 Loading a signal to the Tektronix AWG70002A

The AWG we will be using is the Tektronix AWG70002A which has the following key specifications:

Sampling rate up to 16 GS/s: This is the most important characteristic because it determines the maximum sampling rate that your signal can have. It must not be over 16 GS/s or else the AWG will not be able to recreate it appropriately.

8 GSample waveform memory: This determines how many data points your signal can have.

After making sure this specifications are respected you can create your waveform using the function. When you load your waveform, the AWG will output it and repeat it constantly until you stop playing it.

1. Using the function sgnToWfm: Start up Matlab and change your current folder to mtools and add the signals folder that you want to convert to the Matlab search path. Use the function accordingly, putting as the input parameter the signal file name you want to convert.

2. AWG sampling rate: After calling the function there should be waveform file in the mtools folder, as well as a variable called samplingRate in the Matlab workspace. Make sure this is equal or bellow the maximum sampling frequency of the AWG (16 GS/s), or else the waveform can not be equal to the original signal. If it is higher you have to adjust the parameters in the simulation in order to decrease the sampling frequency of the signal(i.e. decreasing the bit period or reducing the samples per symbol).

3. Loading the waveform file to the AWG: Copy the waveform file to your pen drive and connect it to the AWG. With the software of the awg open, go to browse for waveform on the channel you want to use, and select the waveform file you created (Figure 7.1).

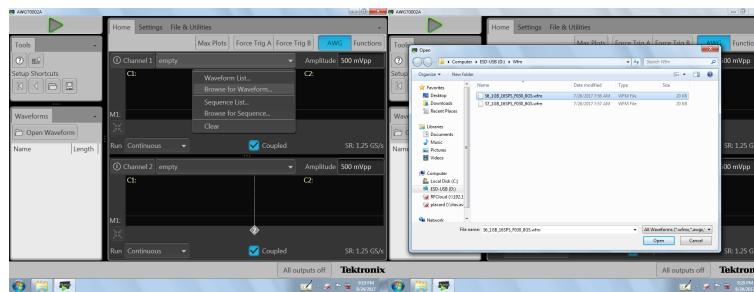


Figura 7.1: Selecting your waveform in the AWG

Now you should have the waveform displayed on the screen. Although it has the same shape, the waveform might not match the signal timing wise due to an incorrect sampling rate configured in the AWG. In this example (Figure 7.2), the original signal has a sample

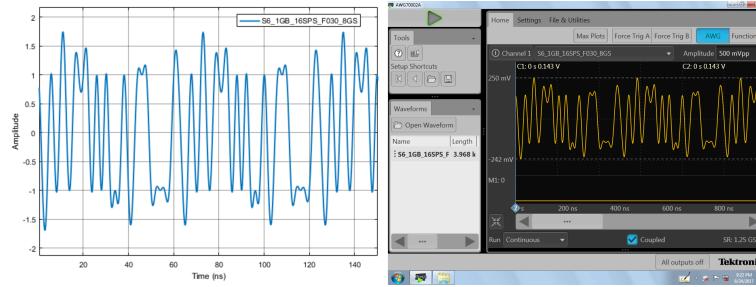


Figura 7.2: Comparison between the waveform in the AWG and the original signal before configuring the sampling rate

rate of 8 GS/s and the AWG is configured to 1.25 GS/s. Therefore it must be changed to the correct value. To do this go to the settings tab, clock settings, and change the sampling rate to be equal to the one of the original signal, 8 GS/s (Figure 7.3). Compare the waveform in

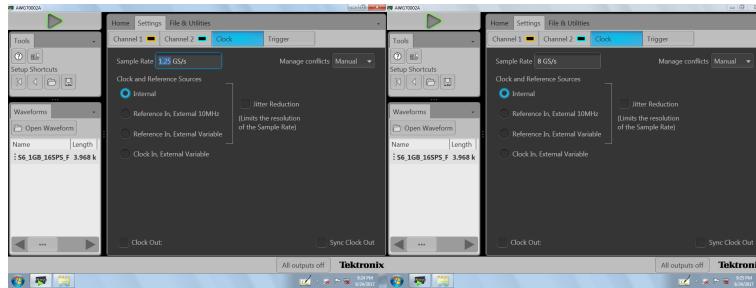


Figura 7.3: Configuring the right sampling rate

the AWG with the original signal, they should be identical (Figure 7.4).

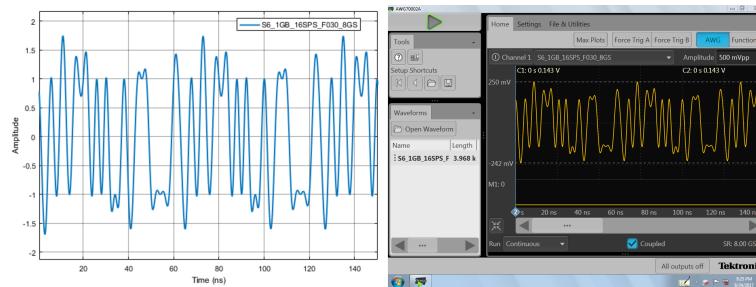


Figura 7.4: Comparison between the waveform in the AWG and the original signal after configuring the sampling rate

4. Generate the signal: Output the wave by enabling the channel you want and clicking on the play button.

Capítulo 8

Algorithms

8.1 Overlap-Save Method

Linear filtering can be easily implemented in time-domain resorting to the use of finite impulse response (FIR) digital filters and convolution property as,

$$y(n) = \sum_{k=0}^{R-1} x(n-k)h(k), \quad (8.1)$$

where $x(n)$ is the input signal, $h(k)$ is the FIR filter coefficients, R is the length of FIR filter coefficients and $y(n)$ represents the filtered output signal. Analysing this equation we can note that, for a block signal of length R , the required number of operations for the direct implementation of equation (8.2) evolves with R^2 , $\mathcal{O}(R^2)$. This limitation imposed the emergence of algorithm, where the linear convolution is calculated faster than the directly implementing of (8.2). In this sense, it is used the computation of linear filtering in frequency domain resorting to the use of fast Fourier transform (FFT) and inverse fast Fourier transform (IFFT) algorithms as However, for long input sequence, the direct implementation of frequency domain filtering in real-time is limited by the limited memory of the digital processors. Hence, the filtering in frequency-domain is implemented by sectioning or block the input signal, such that the practical implementation of FFT and IFFT is feasible. In order to implement the non-cyclic convolution with the finite-length of cyclic convolution that the FFT provides, overlap-save and overlap-add method are use, enabling that the complexity evolves in log scale $\mathcal{O}(N \log_2 N)$. The general method is to split the input signal into manageable blocks, then apply the FFT to to perform the linear convolution and at the end recombine the output blocks such that it is avoided the wrap-around errors due to the circular convolution imposed by FFT.

The overlap-save method can be computed in the following steps:

1. Step 1:

Parameter: Determine the length R of impulse response, $h(n)$;

2. Step 2:

Parameter: Define the size of FFT and IFFT operation, N ;

3. Step 3:

Parameter: Determine the length of block L to section the input sequence $x(n)$, considering that $N = L + R - 1$;

4. Step 4:

Parameter: Pad $L - 1$ zeros to the impulse response $h(n)$ to obtain the length N ;

5. Step 5:

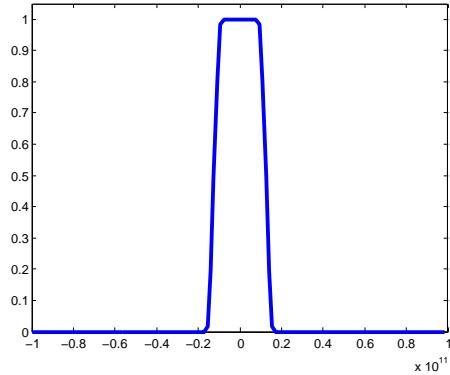


Figura 8.1: Frequency response of raised-cosine filter.

Parameter: Make the segments of the input sequences of length L , $x_i(n)$, where index i correspond to the i^{th} block. Overlap $R - 1$ samples of the previous block at the beginning of the segmented block to obtain a block of length N . In the first block, it is added $R - 1$ null samples;

6. Step 6:

Parameter: Compute the circular convolution of segmented input sequence $x_i(n)$ and $h(n)$ described as,

$$y_i(n) = x_i(n) \circledast h(n). \quad (8.2)$$

This is obtained in the following steps:

Description: Compute the FFT of x_i and h both with length N ;

Description: Compute the multiplication of $X_i(f)$ and the transfer function $H(f)$;

Description: Compute the IFFT of the multiplication result to obtain the time-domain block signal, y_i ;

7. Step 7:

Parameter: Discarded $R - 1$ initial samples from the y_i , and save only the error-free $N - R - 1$ samples in the output record.

In the Figure 8.4 it is illustrated an example of overlap-save method.

8.1.1 Frequency Response of Filter

The frequency response of filter can be directly defined by using the frequency-domain formula, or it can be equivalently calculated from the FFT of impulse response of the filter. In this sense, we present an example of FIR filter (*raised-cosine filter*) to illustrate these two cases.

Frequency-domain Formula

The frequency-domain description of raised-cosine filter can be given as,

$$H(f) = \begin{cases} 1, & |f| \leq \frac{1-\beta}{2T} \\ \frac{1}{2} \left[1 + \cos \left(\frac{\pi T}{\beta} \left[|f| - \frac{1-\beta}{2T} \right] \right) \right], & \frac{1-\beta}{2T} < |f| \leq \frac{1+\beta}{2T} \\ 0, & \text{otherwise} \end{cases}$$

where, f is the frequency, $0 \leq \beta \leq 1$ corresponds to the roll-off factor and T is the reciprocal of the symbol-rate. A plot of direct implementation of frequency response of raised-cosine filter is presented in Figure 8.1, for a roll-off factor of 0.3. The frequency response, $H(f)$, calculated for N frequency bins, which can be defined as,

$$f = -\frac{f_{windowTF}}{2} : \frac{f_{windowTF}}{N} : \left(\frac{f_{windowTF}}{2} - \frac{f_{windowTF}}{N} \right), \quad (8.3)$$

where $f_{windowTF}$ is the sampling frequency. This, imposes that the length of $H(f)$ is N as expected for the N -point FFT and the transfer function multiplication.

FFT of Impulse Response

Alternatively to the frequency-domain formula, we can obtain the frequency response of filter by calculating the FFT of its impulse response. In the case of raised-cosine filter, the impulse response is given as,

$$h(t) = \frac{\sin(\pi t/T)}{\pi t/T} \frac{\cos(\pi t\beta/T)}{1 - (2\beta t/T)^2}, \quad (8.4)$$

where t is the time. Figure 8.2 shows a plot of impulse response of raised-cosine filter for a roll-off factor of 0.3. Before calculating the FFT of the impulse response we must zero-pad the

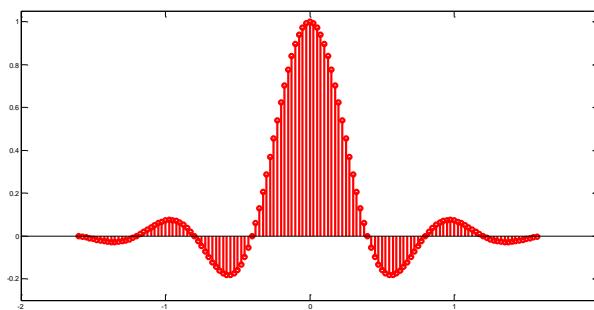


Figura 8.2: Impulse response of raised-cosine filter.

impulse response, which has the length R , to the length N . In this case N is the FFT length, which is efficiently defined as power of 2. The zero-padding process can be performed by appending $L - 1$ zeros at the end of impulse response, as shown in the Figure 8.3.

In both cases, the frequency response of the filter will be limited to the frequency interval $f_{windowTF}, [-\frac{f_{windowTF}}{2}, \frac{f_{windowTF}}{2}]$, and this range show us the N frequency components

Figura 8.3: Zero-padding of impulse response $h(n)$.

obtained from FFT. The minimum frequency bin is $-\frac{f_{windowTF}}{2}$ and the maximum bin is $\frac{f_{windowTF}}{2}$, in which $f_{windowTF}$ corresponds to the sampling frequency. We can note that the spectral width of $H(f)$ is $f_{windowTF}$, which is the inverse of sampling period, dt . It is also important to note that, for a given sampling frequency, the frequency resolution, Δf , of $H(f)$ depends on the parameter N and it increases with N , $\Delta f = \frac{f_{windowTF}}{N}$.

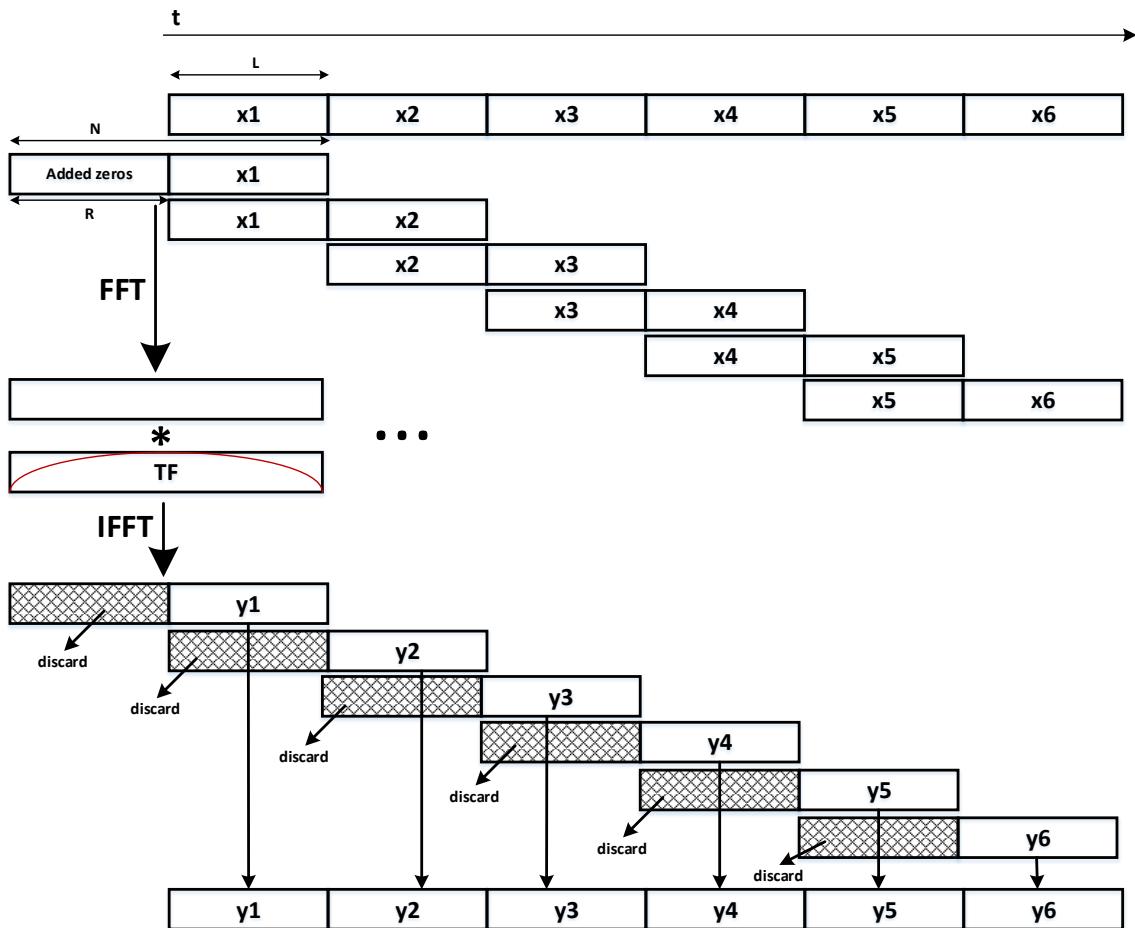


Figura 8.4: Illustration of Overlap-save method.

Bibliografia

- [1] Blahut, R.E. *Fast Algorithms for Digital Signal Processing*, Addison-Wesley, Reading, MA, 1985.
- [2] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, San Diego, CA, USA, 1997.

8.2 FFT

Algorithm

An alternate representation of the Fourier transform can give a flexibility to utilize the same code for the FFT and IFFT with appropriate input arguments. The algorithm for the FFT will follow the following formula,

$$X_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N} \quad 0 \leq k \leq N - 1 \quad (8.5)$$

Similarly, for IFFT,

$$x_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_k \cdot e^{i2\pi kn/N} \quad 0 \leq k \leq N - 1 \quad (8.6)$$

From the above manipulation discussed in equations 8.5 and 8.6, we can write only one script for the implementations Fourier algorithm and manipulate its functionality as a FFT or IFFT by applying an appropriate input arguments.

The generalized form for the algorithm can be given as,

$$OUT = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} IN \cdot e^{s \cdot i2\pi kn/N} \quad 0 \leq k \leq N - 1 \quad (8.7)$$

where,

IN → Input complex signal

OUT → Output complex signal & *s* → '-1' for FFT and '1' for IFFT

Function description

To perform FFT operation, the input argument to the function can be given as follows,

$$OUT = transform(IN, -1)$$

in a similar way, IFFT can be manipulated as,

$$OUT = transform(IN, 1)$$

Flowchart

This algorithm converts time domain signal to frequency domain for both real and complex signal type. In case of real signal, we have to manipulate imaginary part to be zero for the execution of algorithm. The figure 8.5 displays top level architecture of the FFT algorithm. If the length of the input signal is 2^N , then it'll execute Radix-2 algorithm otherwise it'll execute Bluestein algorithm.

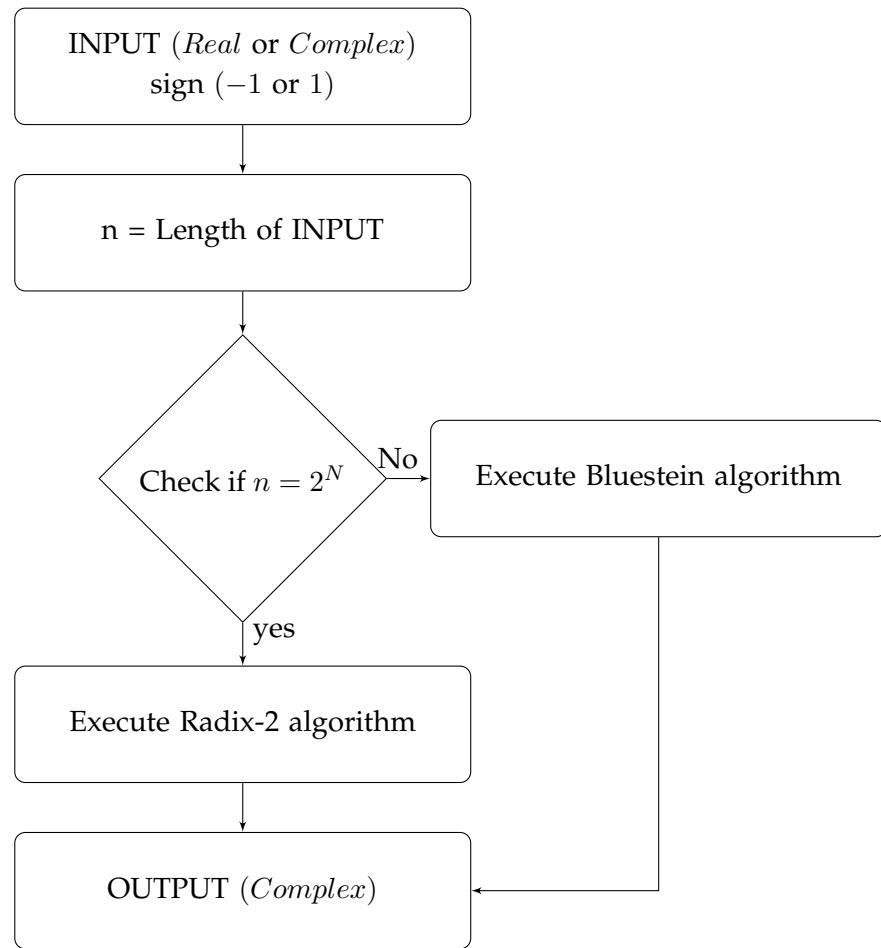


Figura 8.5: Top level architecture of the FFT algorithm

Radix-2 algorithm

The architecture of the algorithm is to accept time domain complex signal and generate frequency domain complex output signal. In the case of real input signal, first we have to convert it into the complex form by adding zero to the imaginary part.

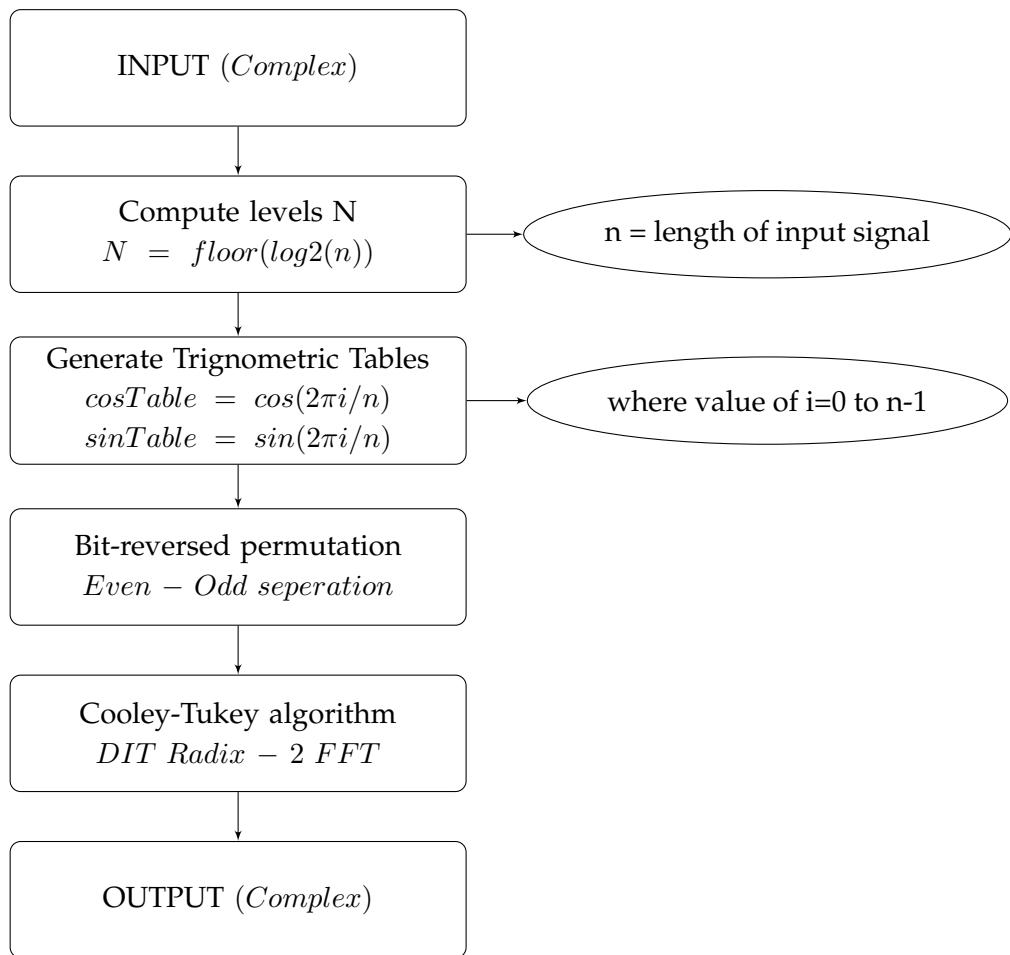


Figura 8.6: Flowchart of Cooley-Tukey DIT Radix-2 algorithm

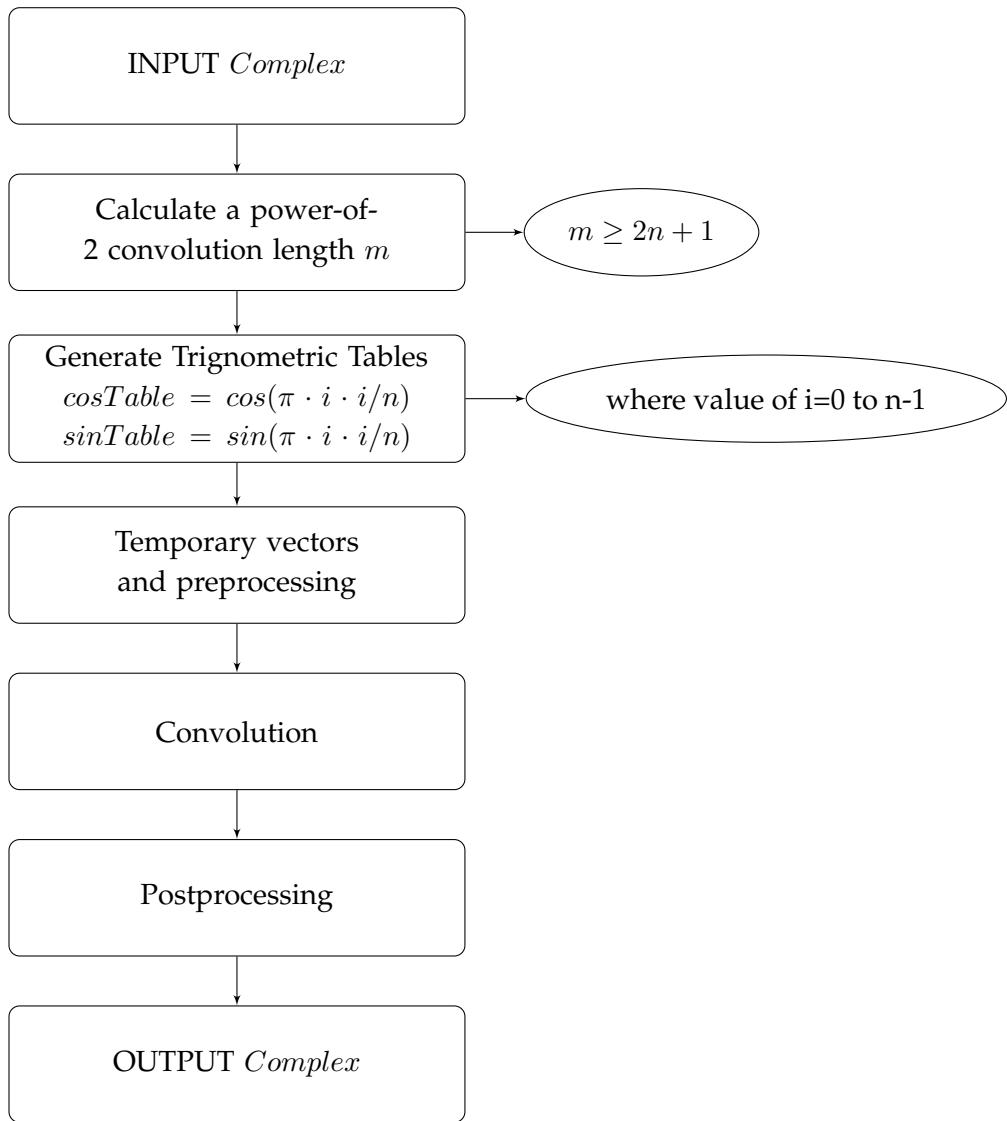
Bluestein algorithm

Figura 8.7: Flowchart of Bluestein algorithm

