NetXPTO - LinkPlanner

26 de Setembro de 2017

Conteúdo

Capítulo 1

Introduction

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

You can run the System

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 is how 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.

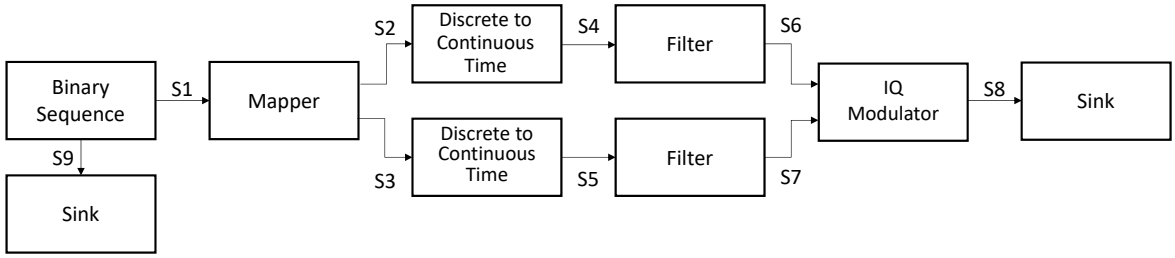


Figura 5.1: QPSK transmitter block diagram.

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 my 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 par 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

Introduction

This system simulates a BPSK transmitter in back-to-back configuration with additive white Gaussian noise at the receiver. The main objective of this system is to test the developed bit error rate block.

Functional Description

A simplified diagram of the system being simulated is presented in the Figure 5.2. A random binary string is generated and encoded in an optical signal using a BPSK modulation format. 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 BER.

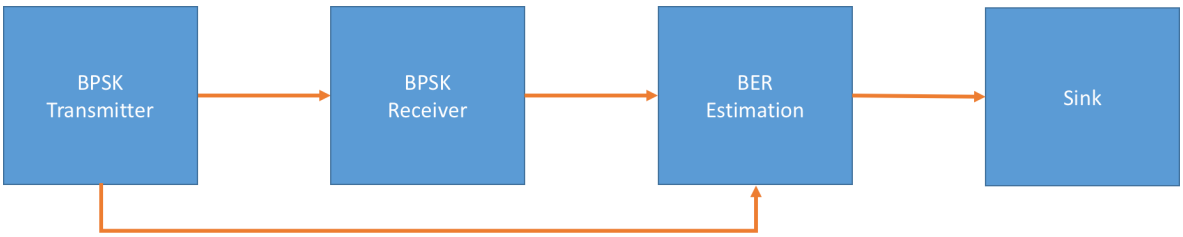


Figura 5.2: Overview of the BPSK system being simulated.

System Blocks	netxpto Blocks
BPSK Transmitter	MQamTransmitter
BPSK Receiver	HomodyneReceiver
BER Estimator	BitErrorRate

Required files

Header Files

File	**Description**
netxpto.h	Generic purpose simulator definitions.
m_qam_transmitter.h	Generates the signal with coded constellation.
homodyne_reciever.h	Performs coherent detection on the input signal.
sampler.h	Samples the input signal at a user defined frequency.
bit_decider.h	Decodes the input signal into a binary string.
bit_error_rate.h	Calculates the bit error rate of the decoded string.
sink.h	Closes any unused signals.

Source Files

File	Description
netxpto.cpp	Generic purpose simulator implementations.
m_qam_transmitter.cpp	Generates the signal with coded constellation.
homodyne_reciever.cpp	Performs coherent detection on the input signal.
sampler.cpp	Samples the input signal at a user defined frequency.
bit_decider.cpp	Decodes the input signal into a binary string.
bit_error_rate.cpp	Calculates the bit error rate of the decoded string.
sink.cpp	Closes any unused signals.

System Input Parameters

This system takes into account the following input parameters:

System Parameters	Description
numberOfBits	Gives the number of bits to be simulated
bitPeriod	Sets the time between adjacent bits
samplesPerSymbol	Establishes the number of samples each bit in the string is given
pLength	PRBS pattern length
iqAmplitudesValues	Sets the state constellation
outOpticalPower_dBm	Sets the optical power, in units of dBm, at the transmitter output
loOutOpticalPower_dBm	Sets the optical power, in units of dBm, of the local oscillator used in the homodyne detector
localOscillatorPhase	Sets the initial phase of the local oscillator used in the homodyne detector
transferMatrix	Sets the transfer matrix of the beam splitter used in the homodyne detector
responsivity	Sets the responsivity of the photodiodes used in the homodyne detector
amplification	Sets the amplification of the trans-impedance amplifier used in the homodyne detector
noiseAmplitude	Sets the amplitude of the gaussian thermal noise added in the homodyne detector
delay	Sets the delay factor of the homodyne detector
posReferenceValue	Set the positive and negative reference values for the bit decision block
negReferenceValue	
confidence	Sets the confidence interval for the calculated QBER
midReportSize	Sets the number of bits between generated QBER mid-reports

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)

Simulation Results

The following results show the dependence of the error rate with the signal power assuming a constant Local Oscillator power of $-20\ dBm$. For reference, the eye diagram at 3 different power levels are also presented. The full line represents the expected results, note that it has been computed assuming a gaussian distribution of the thermal noise, which is not exact given the effect of the matched filter applied before the decoding of the bits, this explains the deviation between the simulation results and the expected values.

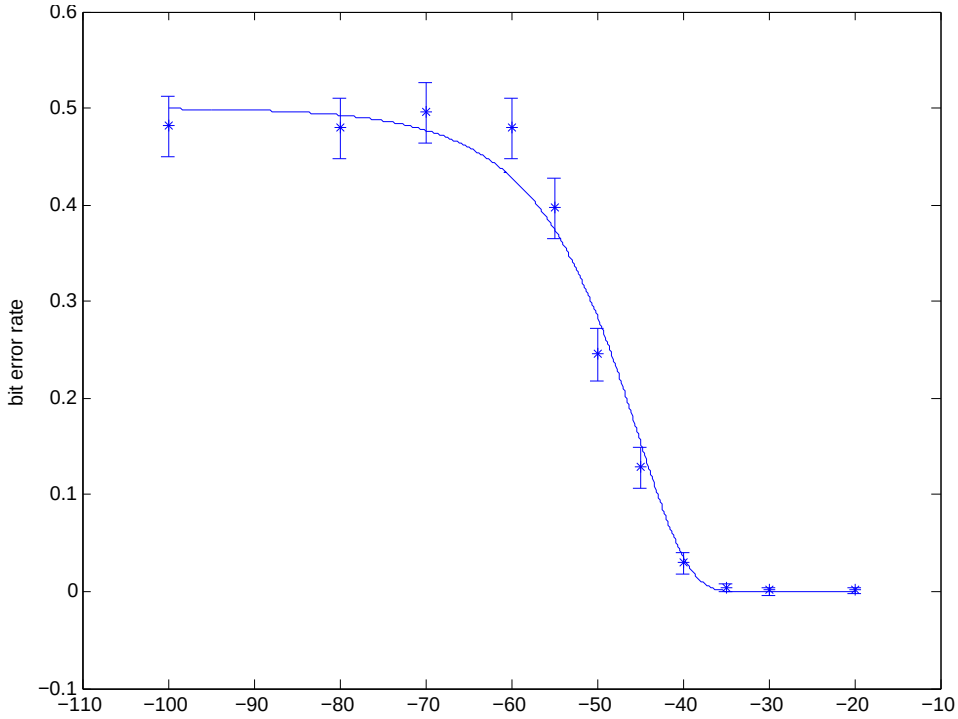
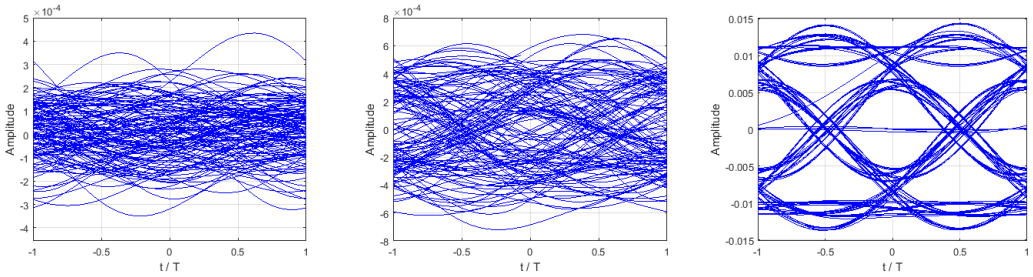


Figura 5.3: Bit Error Rate in function of the signal power in dBm.



(a) Power=-180dBm (b) Power=-130dBm (c) Power=-100dBm

Figura 5.4: Eye diagrams at different signal powers.

5.3 Quantum Noise

Introduction

This document describes a simple emission and detection system that uses coherent states as it's means?? of transmission???.
The transmitted information consists in a binary sequence which is ??translated?? in a sequence of coherent states. In this simulation, the used constellation is formed by the states $\{|\alpha\rangle, |i\alpha\rangle, |-\alpha\rangle, |-i\alpha\rangle\}$, in which α is defined as $\langle n\rangle = |\alpha|^2$ ($\langle n\rangle$ is the expected number of photons in a state). (METER MELHOR)

One of the main effects studied in this system is quantum noise, which is an intrinsic effect?? to coherent states(VER MARK FOX). In principle???? (VER REFERENCIAS) the variance of a coherent state is given by $\Delta X_1 \Delta X_2 = \frac{1}{4}$.

But, given that we combine two photocurrents to obtain an output current, then the total noise will have a combined value of SOMETHING??? Procurar referencias.

Therefore, assuming Gaussian?? (WHY GAUSSIAN?) shot noise, for each quadrature we want $\text{Var}(X_i) = \frac{1}{4}$?????
(TENHO DE PROCURAR REFERENCIAS)

In this simulation, we introduce quantum noise in the photodiodes. We know that a coherent state has an expected number of photons distributed by a Poisson distribution, which has an average number equal to it's variance. Therefore, when the photodiode detects the power of signal, which is proportional to the number of photons, then it's variance must also be proportional to the number of photons.

In fact the last step in detecting the resulting signal introduces an difference between currents, but that only will increase the variance. Assuming the independence between detections, and it's intrinsic noise (PROCURAR MELHOR PALEIO), then:

$$\text{Var}(I_{out}) = \text{Var}(I_1) + \text{Var}(I_2)$$

Therefore, the best result we can achieve will be $\text{Var}(X) = \frac{1}{4}$???? (PROCURAR PALEIO SOBRE ISTO)

Functional Description

The simulation setup is described by diagram in figure 5.5. We start by generating a state from one of the four available ones.???? Then, the signal is received in a Hybrid

Detector??? where the signal is compared with a local oscillator giving four different signals in it's output. Two of those signals are detected by a photodiode which output will be the difference of the two photocurrents. The other two signals will be also be detected by another photodiode, which will obtain the other quadrature of the signal.????? (TEM QUE FICAR MELHOR EXPLICADO).

System Blocks	netxpto Blocks
-	MQAM
-	LocalOscillator
-	Hybrid??
-	Photodiode??
-	Sampler ??

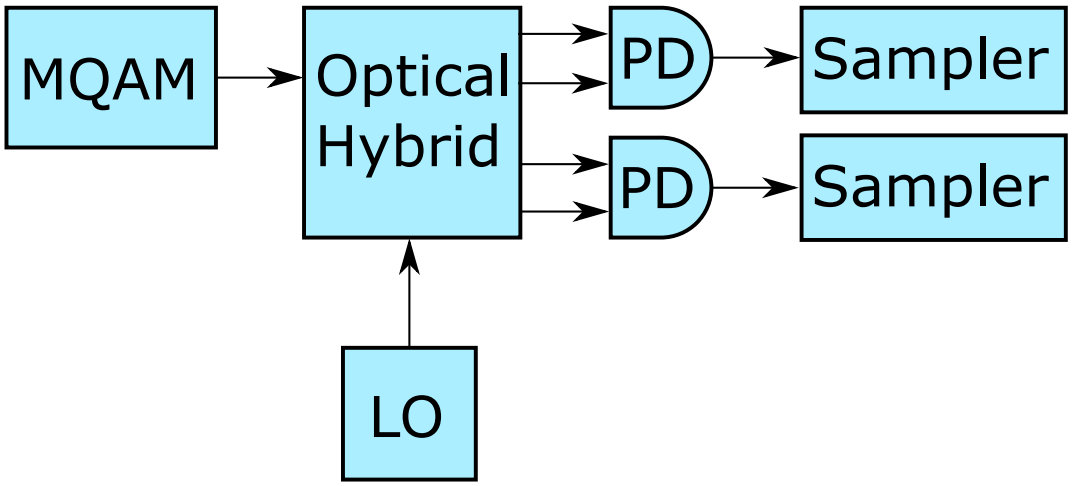


Figura 5.5: Overview of the optical system being simulated.

Required files

Header Files

File	Description
netxpto.h	Generic purpose simulator definitions.
m_qam_transmitter.h	—
local_oscillator.h	Generates continuous coherent signal.
optical_hybrid.h	—
photodiode.h	—
sampler.h	—
sink.h	Closes any unused signals.

Source Files

File	Description
netxpto.cpp	Generic purpose simulator definitions.
m_qam_transmitter.cpp	—
local_oscillator.cpp	Generates continuous coherent signal.
optical_hybrid.cpp	—
photodiode.cpp	—
sampler.cpp	—
sink.cpp	Closes any unused signals.

System Input Parameters

This system takes into account the following input parameters:

System Parameters	Description
numberOfBitsGenerated	Gives the number of bits to be simulated
bitPeriod	Sets the time between adjacent bits
wavelength	Sets the wavelength of the local oscillator in the MQAM????
samplesPerSymbol	Establishes the number of samples each bit in the string is given
localOscillatorPower1	Sets the optical power, in units of W, of the local oscillator inside the MQAM
localOscillatorPower2	Sets the optical power, in units of W, of the local oscillator used for Bob's measurements
localOscillatorPhase	Sets the initial phase of the local oscillator used in the detection
transferMatrix	Sets the transfer matrix of the beam splitter used in the homodyne detector
responsivity	Sets the responsivity of the photodiodes used in the homodyne detectors
bufferLength	Sets the length of the buffer used in the signals
iqAmplitudeValues	Sets the amplitude of the states used in the MQAM????
shotNoise	Chooses if quantum shot noise is used in the simulation
samplesToSkip	Sets the number of samples to skip when writing out some of the signal files.

Inputs

This system takes no inputs.

Outputs

The system outputs the following objects:

Parameter: Signals:

Description: Binary Sequence used in the MQAM; (S_0)

Description: Local Oscillator used in the MQAM; (S_1)

Description: Local Oscillator used in the detection; (S_2)

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

Description: In phase Photodiode output; (S_7)

Description: Quadrature Photodiode output; (S_8)

Description: In phase Sampler output; (S_9)

Description: Quadrature Sampler output; (S_{10})

Simulation Results

The objective of this simulation was to get the (quantum noise???) associated to the detection of coherent states.

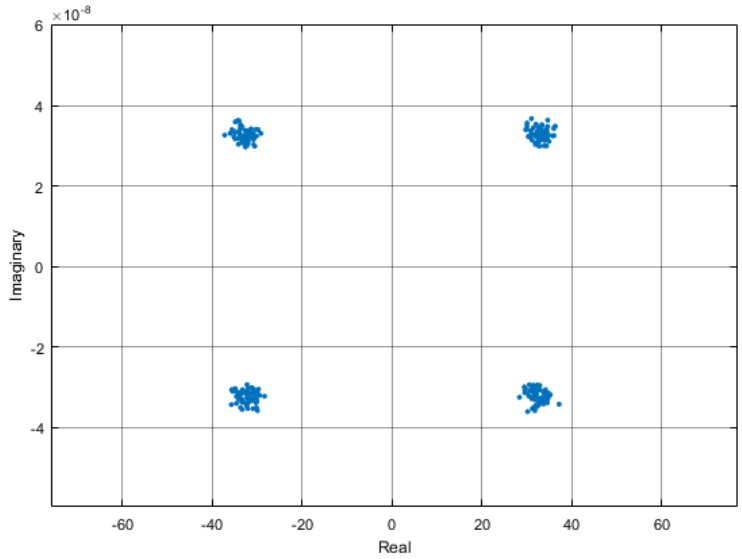


Figura 5.6: Simulation of a constelation of 4 states (n = 100)

We expect that the variance is invariant with the number of photons sent from Alice. The plot in 5.7 show that the simulation also shows this invariance with the number of photons.

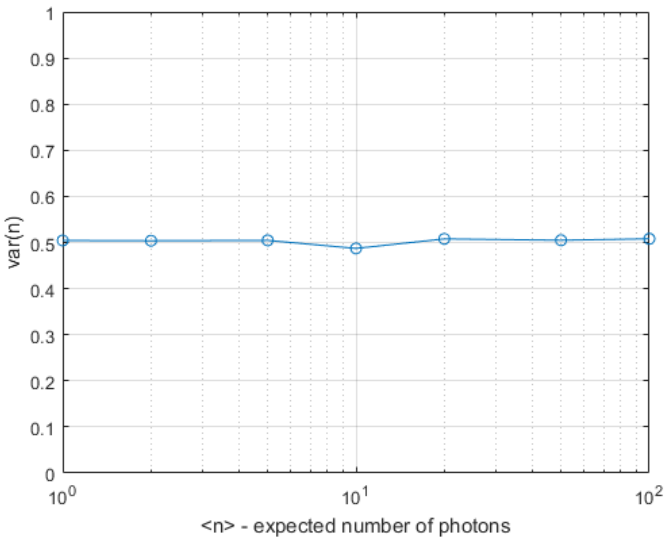


Figura 5.7: Simulation of the variance of n.

We can conclude that the expected variance will give us Var(X) $= \frac{1}{2}$.
The results obtained in our simulations are in accordance with the theoretical prevision???

Known Problems

1. —-

5.4 Continuos Variable Quantum Transmission System

Student Name: Daniel Pereira

Starting Date: May 1, 2017

Goal: Simulation and experimental validation of a continuous variable transmission system.

Description of the Laboratorial and Simulation Setup:
Theoretical Description of the System:
Laboratorial and Simulation Results:
Discussion:
In this section a continuous varible quantum transmission system is analyzed. The results here presented follow closely the [1]. In [1], the security of a continuous variable quantum key distribution (CV-QKD) system is studied theoretically, here we complete that theoretical study with simulations results.

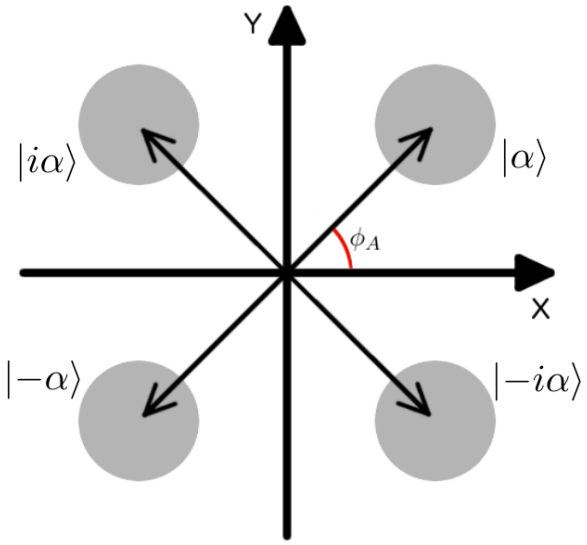


Figura 5.8: State constellation

The state constellation used in the system is presented in Figure 5.8. The emitter (usually named Alice) is going to use two basis, the 45^o base and the -45^o base. In the 45^o base, Alice sends one of two values, 1 and -1, which correspond to the states $|\alpha\rangle$ and $|-\alpha\rangle$. In the -45^o base, Alice can also send one of two values, 1 and -1, which correspond to the states $|-i\alpha\rangle$ and $|i\alpha\rangle$. At the end Alice is going to send one of the four states $|\alpha\rangle$, $|-\alpha\rangle$, $|-i\alpha\rangle$, and $|i\alpha\rangle$, with equal probability.

Because we don't know $\grave{a}$ prior which state is going be transmitted, neither which basis is going to be used, and to incorporate our "ignorance" in the system description, we can

work with the density operator. The density operator is a proper tool to describe “statistical mixtures". A “statistical mixtures"is one state, from a possible set, but we don’t know which state it is. There is no state superposition.

Since all states have the same probability of occurring, the state density operator is given by:

$$\hat{\rho} = \frac{1}{4}\left(|\alpha\rangle\langle\alpha| + |-\alpha\rangle\langle-\alpha| + |i\alpha\rangle\langle i\alpha| + |-i\alpha\rangle\langle-i\alpha|\right). \tag{5.1}$$

The probability to detect at the receiver the state $|\alpha\rangle$ is given by

$$P(\alpha) = \langle\alpha|\,\hat{\rho}\,|\alpha\rangle = \frac{1}{4}. \tag{5.2}$$

Note that the density operator is equivalent to the wave function in terms of the system description.

From the receiver perspective, i.e. from the Bob perspective, and after knowing the base used by Alice. The density operator can be reduce to,

$$\hat{\rho}_1 = \frac{1}{2}\left(|\alpha\rangle\langle\alpha| + |-\alpha\rangle\langle-\alpha|\right), \tag{5.3}$$

$$\hat{\rho}_2 = \frac{1}{2}\left(|i\alpha\rangle\langle i\alpha| + |-i\alpha\rangle\langle-i\alpha|\right). \tag{5.4}$$

where 1 corresponds to the 45^{o} base and -1 corresponds to -45^{o}.

Single Base Homodyne Detection

The probability of obtaining a quadrature $\hat{X}_\phi = \hat{X}_1\cos\phi + \hat{X}_2\sin\phi$ when measuring the coherent state $|\alpha\rangle$ is given by the following gaussian distribution:

$$|\langle X_\phi|\alpha\rangle|^2 = \sqrt{\frac{2}{\pi}}e^{-2(X_\phi - \alpha\cos\phi)^2}, \tag{5.5}$$

We can define the "correct"and "wrong"basis measurement probability density, respectively, as:

$$\langle X_i|\,\hat{\rho}_j\,|X_i\rangle = \begin{cases} \frac{1}{\sqrt{2\pi}}\left(e^{-2(X_i-\alpha)^2} + e^{-2(X_i+\alpha)}\right), & i = j \\ \sqrt{\frac{2}{\pi}}e^{-2X_i^2}, & i \neq j \end{cases}. \tag{5.6}$$

The post selection efficiency (PSE) can be defined as the probability of a measurement in the correct basis yields a result that satisfies the limit value X_0:

$$\begin{aligned} P(X_0,\alpha) &= \int_{-\infty}^{-X_0}\langle X_1|\,\hat{\rho}_1\,|X_1\rangle\,dX_1 + \int_{X_0}^{\infty}\langle X_1|\,\hat{\rho}_1\,|X_1\rangle\,dX_1 \\ &= \frac{1}{2}\left[\text{erfc}(\sqrt{2}(X_0+\alpha)) + \text{erfc}(\sqrt{2}(X_0-\alpha))\right]. \end{aligned} \tag{5.7}$$

The bit error rate (BER) is the normalized probability of, after choosing the correct basis, obtaining the wrong bit value:

$$Q(X_0,\alpha) = \frac{1}{P(X_0,\alpha)} \int_{-\infty}^{-X_0} |\langle X_i|\alpha\rangle|\, dX_i = \frac{\text{erfc}\left(\sqrt{2}(X_0+\alpha)\right)}{2P(X_0,n)} \tag{5.8}$$

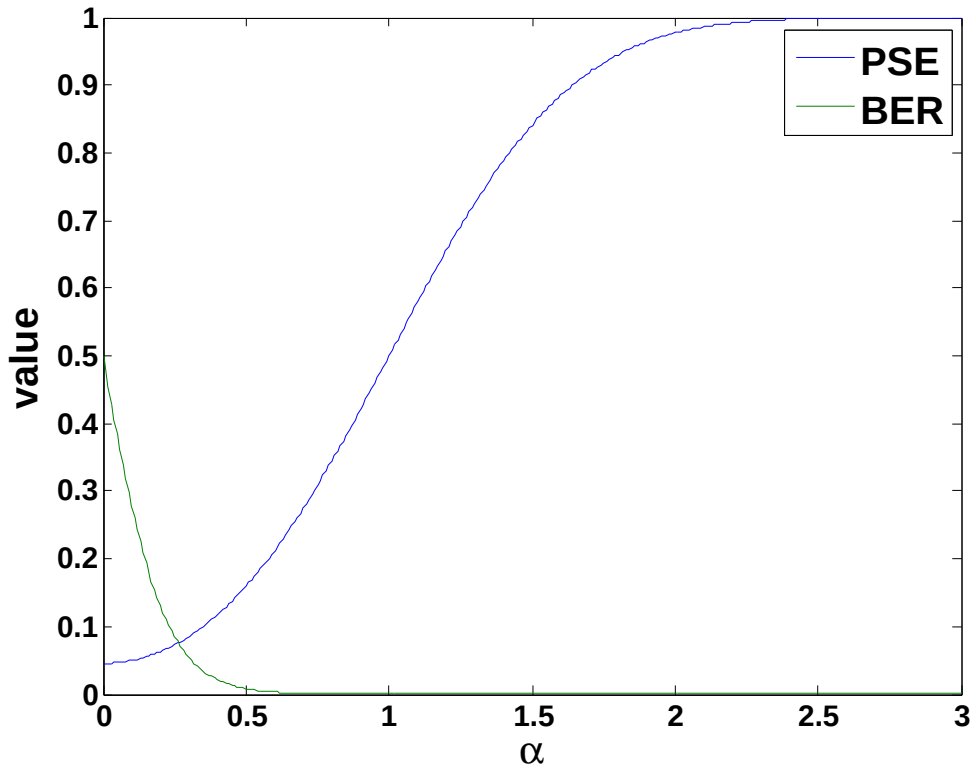


Figura 5.9: BER and PSE in function of α for the single homodyne setup. $X_0 = 1$ was used

Double Homodyne setup

In our proposed double homodyne protocol both quadratures are measured simultaneously, as such the concept of correct and wrong basis measurements has no value. Our protocol also makes use of a locally generated Local Oscillator (LO), obtained from a different laser than the one used to generate the signal, thus we have to take into account the phase drift between both lasers. High intensity reference pulses are sent periodically to allow for an estimation of the phase drift. The double homodyne setup requires the signal to be divided into the two utilized detectors, so each measurement is made on a coherent state with half the amplitude of the incoming signal $\alpha \to \frac{\alpha}{\sqrt{2}}$

For each incoming pulse we measure quadratures X_ϕ and Y_ϕ. ϕ has contributions from both the encoded angle, θ, and the phase difference between lasers, ϵ, we assume $\phi = \theta + \epsilon$.

On the reference pulses no phase is encoded, that is $\theta = 0$, thus ϵ can be estimated. Assuming ϵ doesn't change between a reference pulse and the following signal pulse, the measured quadratures can be cast into the originally sent quadratures X_θ and Y_θ via:

$$\begin{aligned} X_\theta &= X_\phi \cos\epsilon - Y_\phi \sin\epsilon \\ Y_\theta &= X_\phi \sin\epsilon + Y_\phi \sin\epsilon \end{aligned} \tag{5.9}$$

Assuming an announcement of the coding basis, the density operators (5.3) and (5.4) still apply. We can now define the probability density of obtaining results X_θ and Y_θ, assuming a state in the X_1 base was sent, as:

$$\langle X_\theta|\hat{\rho}_1|X_\theta\rangle = \frac{\sqrt{\frac{2}{\pi}}}{4}\left(e^{-2\left(x_\theta - \frac{\alpha}{\sqrt{2}}\cos\theta\right)^2} + e^{-2\left(x_\theta + \frac{\alpha}{\sqrt{2}}\cos\theta\right)^2}\right), \tag{5.10}$$

$$\langle Y_\theta|\hat{\rho}_1|Y_\theta\rangle = \frac{\sqrt{\frac{2}{\pi}}}{4}\left(e^{-2\left(y_\theta - \frac{\alpha}{\sqrt{2}}\sin\theta\right)^2} + e^{-2\left(y_\theta + \frac{\alpha}{\sqrt{2}}\sin\theta\right)^2}\right). \tag{5.11}$$

Now each state needs to satisfy two limit values, X_0 and Y_0, to be accepted. Thus, the PSE is now defined as:

$$\begin{aligned} P_{DH}(X_0, Y_0, \alpha) = &\int_{-\infty}^{-X_0} \langle X_\theta|\hat{\rho}_1|X_\theta\rangle\, dx_\theta \int_{-\infty}^{-Y_0} \langle Y_\theta|\hat{\rho}_1|Y_\theta\rangle\, dy_\theta + \\ &\int_{X_0}^{\infty} \langle X_\theta|\hat{\rho}_1|X_\theta\rangle\, dx_\theta \int_{Y_0}^{\infty} \langle Y_\theta|\hat{\rho}_1|Y_\theta\rangle\, dy_\theta \\ &= \frac{1}{4}\left\{\text{erfc}\left[\sqrt{2}\left(X_0 - \frac{\alpha}{\sqrt{2}}\cos\theta\right)\right] + \text{erfc}\left[\sqrt{2}\left(X_0 + \frac{\alpha}{\sqrt{2}}\cos\theta\right)\right]\right\} \\ &\left\{\text{erfc}\left[\sqrt{2}\left(Y_0 - \frac{\alpha}{\sqrt{2}}\sin\theta\right)\right] + \text{erfc}\left[\sqrt{2}\left(Y_0 + \frac{\alpha}{\sqrt{2}}\sin\theta\right)\right]\right\}, \end{aligned} \tag{5.12}$$

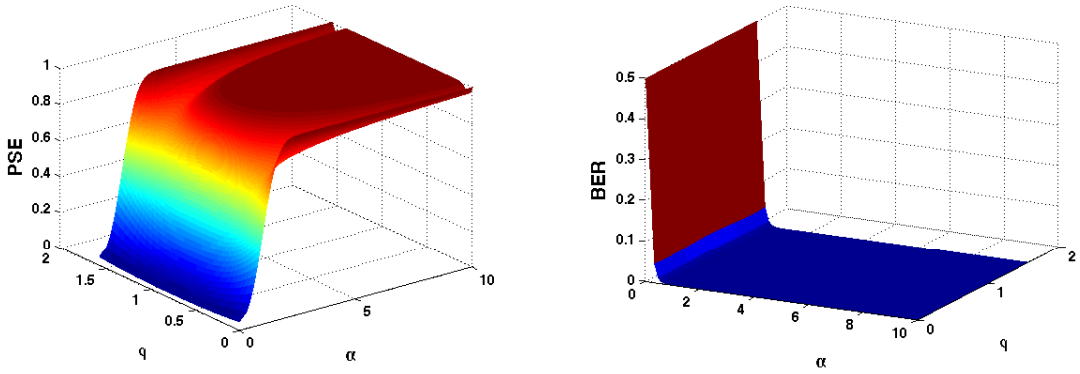
The DH subscript denotes Double Homodyne. In a somewhat similar manner, the BER is now defined as:

$$\begin{aligned} Q_{DH}(X_0, Y_0, \alpha) = &\frac{1}{P_{DH}}\left(\int_{-\infty}^{-X_0} \left|\langle X_\theta|\frac{\alpha}{\sqrt{2}}\rangle\right|^2 dx_\theta \int_{-\infty}^{-Y_0} \left|\langle Y_\theta|\frac{\alpha}{\sqrt{2}}\rangle\right|^2 dy_\theta + \right. \\ &\left.\int_{X_0}^{\infty} \left|\langle X_\theta| - \frac{\alpha}{\sqrt{2}}\rangle\right|^2 dx_\theta \int_{Y_0}^{\infty} \left|\langle Y_\theta| - \frac{\alpha}{\sqrt{2}}\rangle\right|^2 dy_\theta\right) \\ &= \frac{1}{2P_{DH}}\text{erfc}\left[\sqrt{2}\left(X_0 + \frac{\alpha}{\sqrt{2}}\cos\theta\right)\right]\text{erfc}\left[\sqrt{2}\left(Y_0 + \frac{\alpha}{\sqrt{2}}sin\theta\right)\right], \end{aligned} \tag{5.13}$$

note that, in this definition for BER, only values $\theta \in \left[0, \frac{\pi}{2}\right]$ make sense (the sent state was α).

Functional Description

Simplified diagrams of the systems being simulated are presented in Figures 5.11a. and 5.11b. Two optical signals are generated, one with a constant power level of 10 dBm and the other with power in multiples of the power corresponding to a single photon per

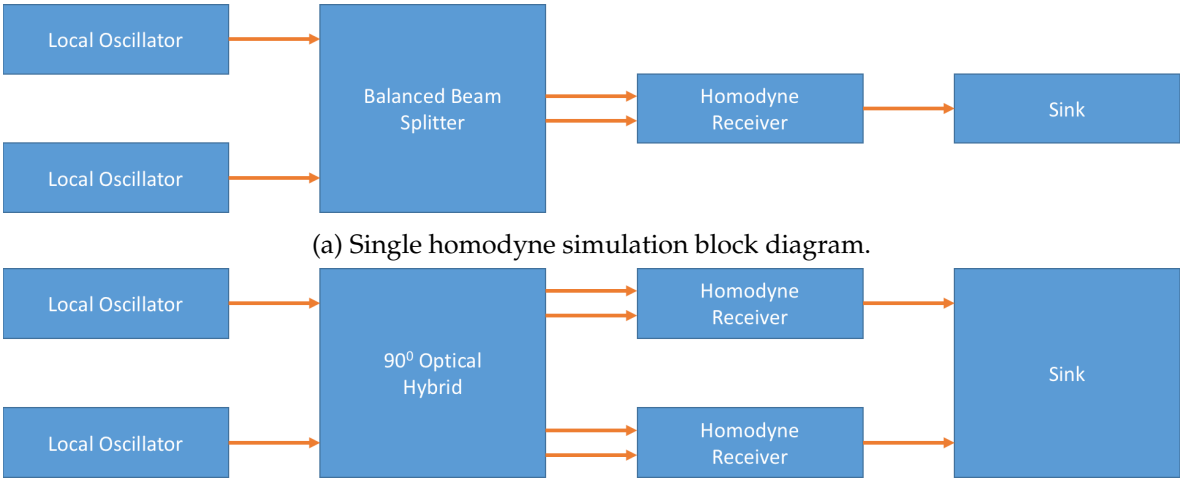


(a) PSE in function of α and θ for the double homodyne setup. $X_0 = 1$ was used

(b) BER in function of α and θ for the double homodyne setup. $X_0 = 1$ was used

Figura 5.10: Theoretical results for double homodyne setup.

sampling time (6.4078e$\times 10^{-13}$ W for a sampling time of 200 ns). The two signals are mixed, with a Balanced Beam Splitter in the single homodyne case and with a 90º Optical Hybrid in the double homodyne one, and are subsequently evaluated with recourse to Homodyne Receivers.



(a) Single homodyne simulation block diagram.

(b) Double homodyne simulation block diagram.

Figura 5.11: Block diagrams of both simulation results presented in this report.

System Blocks	netxpto Blocks
Local Oscillator	LocalOscillator
Homodyne Receiver	I_HomodyneReceiver
Balanced Beam Splitter	BalancedBeamSplitter
90º Optical Hybrid	OpticalHybrid

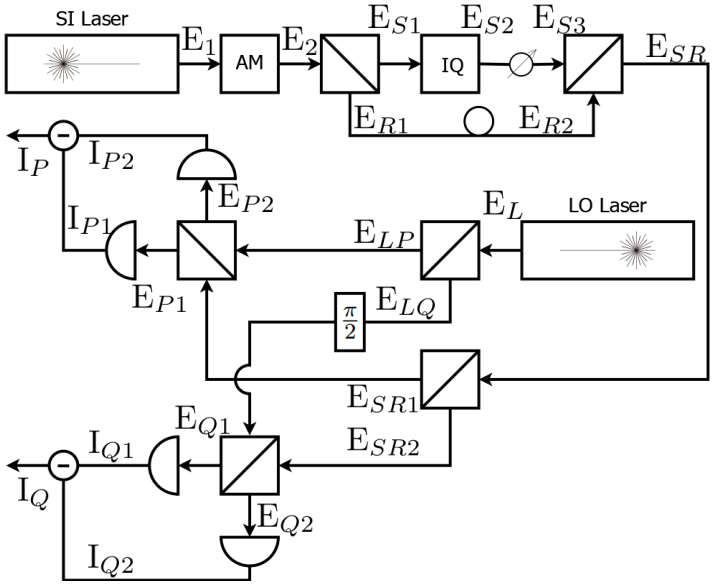


Figura 5.12: Simplified diagram of experimental setup.

Theoretical study of the setup

A detailed diagram of the system studied is presented in Figure 5.12, with the respective mathematical treatment presented in equations (5.14) through (5.33).

$$E_1 = |E_S| e^{i(\omega_S t + \xi_S(t))} \tag{5.14}$$

$$p(t) = \begin{cases} 1, \ \frac{T}{8} \leq t < \frac{T}{8} \\ 0, \ \text{other} \end{cases} \tag{5.15}$$

$$E_2 = p(t)|E_S| e^{i(\omega_S t + \xi_S(t))} \tag{5.16}$$

$$E_{S1} = \frac{p(t)}{\sqrt{2}}|E_S| e^{i(\omega_S t + \xi_S(t))} \tag{5.17}$$

$$E_{S2} = \frac{p(t)}{\sqrt{2}}|E_S| e^{i\left(\omega_S t + \phi_{IQ}(t) + \xi_S(t) + \xi_{IQ}(t)\right)} \tag{5.18}$$

$$E_{S3} = \frac{p(t)}{\sqrt{2}\text{Att}}|E_S| e^{i\left(\omega_S t + \phi_{IQ}(t) + \xi_S(t) + \xi_{IQ}(t)\right)} \tag{5.19}$$

$$E_{R1} = \frac{p(t)}{\sqrt{2}}|E_S| e^{i(\omega_S t + \xi_S(t))} \tag{5.20}$$

$$E_{R2} = \frac{p(t - \frac{T}{2})}{\sqrt{2}}|E_S| e^{i\left(\omega_S (t - \frac{T}{2}) + \xi_S (t - \frac{T}{2})\right)} \tag{5.21}$$

$$E_{SR} = \frac{|E_S|}{\sqrt{2}} \left[\frac{p(t)}{\text{Att}} e^{i\left(\omega_S t + \phi_{IQ}(t) + \xi_S(t) + \xi_{IQ}(t)\right)} + p\left(t - \frac{T}{2}\right) e^{i\left(\omega_S \left(t - \frac{T}{2}\right) + \xi_S \left(t - \frac{T}{2}\right)\right)} \right] \tag{5.22}$$

$$E_{SR1} = \frac{|E_S|}{2} \left[\frac{p(t)}{\text{Att}} e^{i\left(\omega_S t + \phi_{IQ}(t) + \xi_S(t) + \xi_{IQ}(t)\right)} + p\left(t - \frac{T}{2}\right) e^{i\left(\omega_S \left(t - \frac{T}{2}\right) + \xi_S \left(t - \frac{T}{2}\right)\right)} \right] \tag{5.23}$$

$$E_{SR2} = \frac{|E_S|}{2}\left[\frac{p(t)}{\text{Att}}e^{i\left(\omega_S t+\phi_{IQ}(t)+\xi_S(t)+\xi_{IQ}(t)\right)} + p\left(t-\frac{T}{2}\right)e^{i\left(\omega_S\left(t-\frac{T}{2}\right)+\xi_S\left(t-\frac{T}{2}\right)\right)}\right] \tag{5.24}$$

$$E_L(t) = |E_L|e^{i(\omega_L t+\xi_L(t))} \tag{5.25}$$

$$E_{LP}(t) = \frac{1}{\sqrt{2}}|E_L|e^{i(\omega_L t+\xi_L(t))} \tag{5.26}$$

$$E_{LQ}(t) = \frac{1}{\sqrt{2}}|E_L|e^{i\left(\omega_L t+\xi_L(t)+\frac{\pi}{2}\right)} \tag{5.27}$$

$$E_{P1}(t) = \frac{1}{\sqrt{2}}\left\{\frac{|E_L|}{\sqrt{2}}e^{i(\omega_L t+\xi_L(t))} + \frac{|E_S|}{2}\left[\frac{p(t)}{\text{Att}}e^{i(\omega_S t+\phi_{IQ}t+\xi_S(t)+\xi_{IQ}(t))} + p\left(t-\frac{T}{2}\right)e^{i\left(\omega_S\left(t-\frac{T}{2}\right)+\xi_S\left(t-\frac{T}{2}\right)\right)}\right]\right\} \tag{5.28}$$

$$E_{P2}(t) = \frac{1}{\sqrt{2}}\left\{\frac{|E_L|}{\sqrt{2}}e^{i(\omega_L t+\xi_L(t))} - \frac{|E_S|}{2}\left[\frac{p(t)}{\text{Att}}e^{i(\omega_S t+\phi_{IQ}t+\xi_S(t)+\xi_{IQ}(t))} + p\left(t-\frac{T}{2}\right)e^{i\left(\omega_S\left(t-\frac{T}{2}\right)+\xi_S\left(t-\frac{T}{2}\right)\right)}\right]\right\} \tag{5.29}$$

$$E_{Q1}(t) = \frac{1}{\sqrt{2}}\left\{\frac{|E_L|}{\sqrt{2}}e^{i\left(\omega_L t+\xi_L(t)+\frac{\pi}{2}\right)} + \frac{|E_S|}{2}\left[\frac{p(t)}{\text{Att}}e^{i(\omega_S t+\phi_{IQ}t+\xi_S(t)+\xi_{IQ}(t))} + p\left(t-\frac{T}{2}\right)e^{i\left(\omega_S\left(t-\frac{T}{2}\right)+\xi_S\left(t-\frac{T}{2}\right)\right)}\right]\right\} \tag{5.30}$$

$$E_{Q2}(t) = \frac{1}{\sqrt{2}}\left\{\frac{|E_L|}{\sqrt{2}}e^{i\left(\omega_L t+\xi_L(t)+\frac{\pi}{2}\right)} - \frac{|E_S|}{2}\left[\frac{p(t)}{\text{Att}}e^{i(\omega_S t+\phi_{IQ}t+\xi_S(t)+\xi_{IQ}(t))} + p\left(t-\frac{T}{2}\right)e^{i\left(\omega_S\left(t-\frac{T}{2}\right)+\xi_S\left(t-\frac{T}{2}\right)\right)}\right]\right\} \tag{5.31}$$

$$I_P(t) = \frac{|E_S||E_L|}{\sqrt{2}}\left\{\frac{p(t)}{\text{Att}}\cos((\omega_S-\omega_L)t+\phi IQ(t)+\xi_S(t)+\xi_{IQ}(t)-\xi_L(t)) + p\left(t-\frac{T}{2}\right)\cos\left((\omega_S-\omega_L)t+\omega_S\frac{T}{2}+\xi_S\left(t-\frac{T}{2}\right)-\xi_L(t)\right)\right\} \tag{5.32}$$

$$I_Q(t) = \frac{|E_S||E_L|}{\sqrt{2}}\left\{\frac{p(t)}{\text{Att}}\sin((\omega_S-\omega_L)t+\phi IQ(t)+\xi_S(t)+\xi_{IQ}(t)-\xi_L(t)) + p\left(t-\frac{T}{2}\right)\sin\left((\omega_S-\omega_L)t+\omega_S\frac{T}{2}+\xi_S\left(t-\frac{T}{2}\right)-\xi_L(t)\right)\right\} \tag{5.33}$$

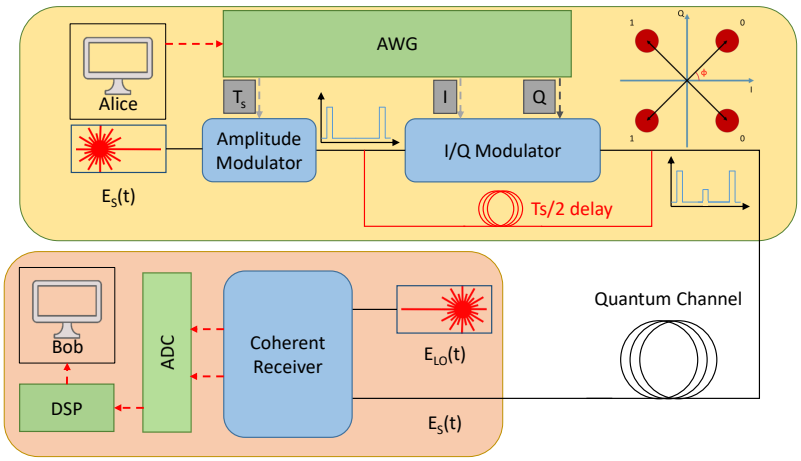


Figura 5.13: Diagram of full experimental setup.

Experimental description

The main experimental setup utilized for the study presented in this document is presented in Figure 5.13. The setup contained two Yenista OSICS Band C/AG TLS lasers, tuned to a 1550 nm wavelength. A JDSU dual drive Mach-Zehnder Modulator with a Picosecond 5865 RF driver, employed at the output of one of the lasers, set the repetition rate at 1 GHz with a pulse time of 125 ps. 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 ∼500 ps. The employed IQ Modulator was a u2t photonics 32 GHz IQ Modulator with a SHF 807 RF driver, the driving signal being generated by a Tektronix AWG70002A Arbitrary Waveform Generator (AWG). A Variable Optical Attenuator (VOA) was set at the output of the IQ modulator to allow a fine tuning of the optical power to the desired level. The two arms output created by the first beam-splitter are combined by a 50/50 beam combiner. A fibre channel of $\sim$ 10 km was set between Alice's and Bob's setup. A Picometrix CR-100D 100G Integrated Balanced Receiver for Coherent Applications was employed to perform double homodyne measurements, recovering 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 μs.

Experimental results

Required files

Header Files

File	Description
netxpto.h	Generic purpose simulator definitions.
local_oscillator.h	Generates continuous coherent signal.
balanced_beam_splitter.h	Mixes the two input signals into two outputs.
optical_hybrid.h	Mixes the two input signals into four outputs.
homodyne_reciever.h	Performs coherent detection on the input signal.
sink.h	Closes any unused signals.

Source Files

File	Description
netxpto.cpp	Generic purpose simulator definitions.
local_oscillator.cpp	Generates continuous coherent signal.
balanced_beam_splitter.cpp	Mixes the two input signals into two outputs.
optical_hybrid.cpp	Mixes the two input signals into four outputs.
homodyne_reciever.cpp	Performs coherent detection on the input signal.
sink.cpp	Closes any unused signals.

System Input Parameters

This system takes into account the following input parameters:

System Parameters	Description
numberOfBitsGenerated	Gives the number of bits to be simulated
bitPeriod	Sets the time between adjacent bits
samplesPerSymbol	Establishes the number of samples each bit in the string is given
localOscillatorPower_dBm1	Sets the optical power, in units of dBm, at the reference output
localOscillatorPower2	Sets the optical power, in units of W, of the signal
localOscillatorPhase1	Sets the initial phase of the local oscillator used for reference
localOscillatorPhase2	Sets the initial phase of the local oscillator used for signal
transferMatrix	Sets the transfer matrix of the beam splitter used in the homodyne detector
responsivity	Sets the responsivity of the photodiodes used in the homodyne detector
amplification	Sets the amplification of the trans-impedance amplifier used in the homodyne detector
electricalNoiseAmplitude	Sets the amplitude of the gaussian thermal noise added in the homodyne detector
shotNoise	Chooses if quantum shot noise is used in the simulation

Inputs

This system takes no inputs.

Outputs

The single homodyne system outputs the following objects:

Parameter: Signals:

Description: Local Oscillator Optical Reference; (S_1)

Description: Local Oscillator Optical Signal; (S_2)

Description: Beam Splitter Outputs; (S_3, S_4)

Description: Homodyne Detector Electrical Output; (S_5)

The double homodyne system outputs the following objects:

Parameter: Signals:

Description: Local Oscillator Optical Reference; (S_1)

Description: Local Oscillator Optical Signal; (S_2)

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

Description: Homodyne Detector Electrical Output; (S_7)

Simulation Results

Single homodyne results

The numerical results presented in Figure 5.14 were obtained with the simulation described by the block diagram in Figure 5.11a. Theoretical results are a direct trace of (5.8). One can see that the numerical results adhere quite well to the expected curve.

Double homodyne results

The numerical results presented in Figure 5.15 were obtained with the simulation described by the block diagram in Figure 5.11b. Theoretical results are a direct trace of (5.13) with $\theta = \frac{\pi}{4}$. One can see that the numerical results adhere quite well to the expected curve.

Known Problems

1. Homodyne Super-Block not functioning

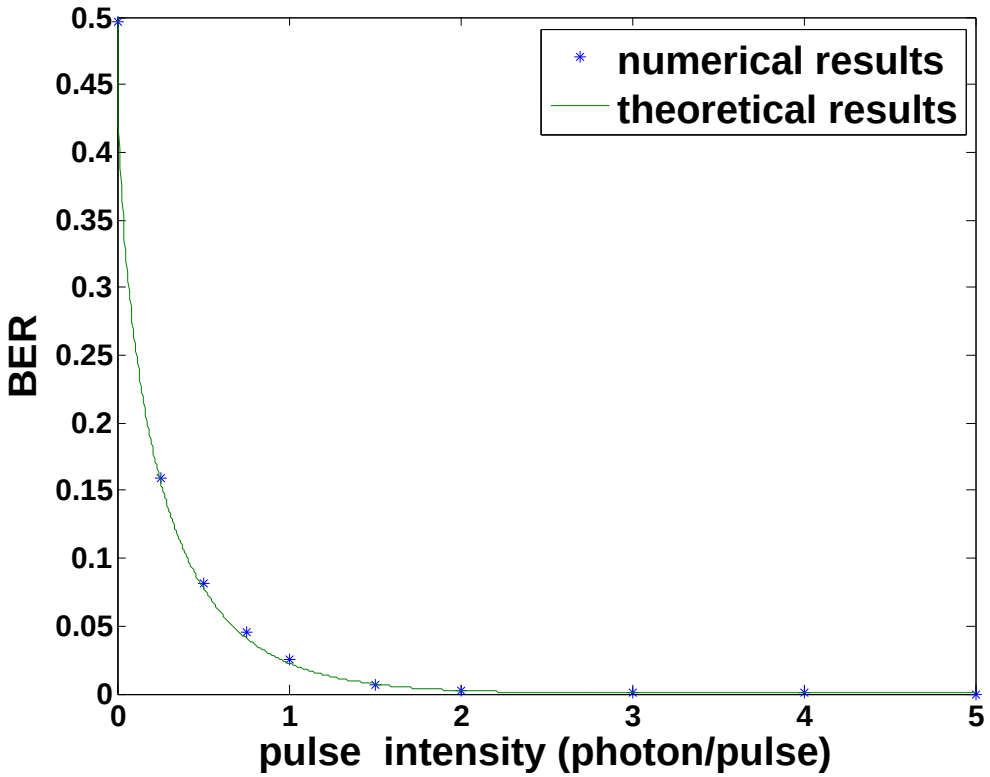


Figura 5.14: BER in function of α for the single homodyne setup. $X_0 = 0$ was used

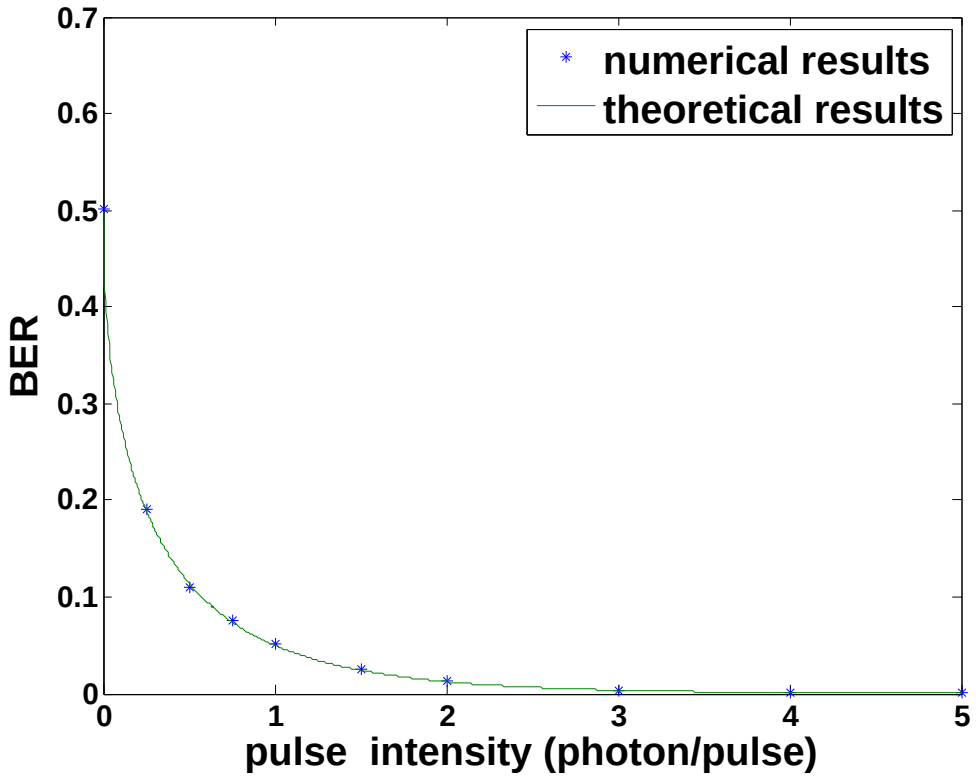


Figura 5.15: BER in function of α for the double homodyne setup. $X_0 = 0$ was used

Bibliografia

[1] Ryo Namiki and Takuya Hirano. Security of quantum cryptography using balanced homodyne detection. *Physical Review A*, 67(2):022308, 2003.

5.5 Simplified Coherent Receiver

Student Name	:	Romil Patel
Starting Date	:	August 16, 2017
Goal	:	Develop a simplified structure (low cost) for a coherent receiver, that can be used in coherent PON, inter-data center connections, or metropolitan networks (optical path lengths should be < 100 km).

In recent days, homodyne detection has been discussed and investigated a lot due to the advancement in the DSP in the electrical domain. However, a major drawback of homodyne detection is the incoming signal should be separated into inphase and quadrature (I/Q) signals in the optical domain. Therefore, it demands more hardware to accommodate the requirement of the signal separation in the optical domain. For instance, 4 balanced photodetectors with double hybrid structures and 4-channel ADCs are required. On the other hand, heterodyne receiver simplifies the detection scheme to some extent with the requirement of having only half of photodetectors and ADC.
Such coherent detection scheme constitute the solution for the medium-to-long-reach application; however, the cost of coherent receiver becomes a major obstacle in the case of short-reach links applications like PON, inter-data-center communications, metropolitan network etc. In order to get rid of higher cost and to make the transceiver more efficiently applicable in short-reach links, a new architecture of optical receiver has been proposed which combines the advantages of coherent transmission and cost effectiveness of direct detection. The working principle of the receiver is based on the famous Kramers-Kronig(KK) relationship which facilitates digital post-compensation of linear propagation impairments. The proposed Kramers-Kronig(KK) receiver structure is highly efficient in terms of spectral occupancy and energy consumptions.

5.5.1 Minimum Phase Signal

The communication scheme discussed here relies on the identifying a specific condition that ensures the received signal is minimum phase. This condition facilitates the unique way to extract the phase of the received signal from its intensity. If we denote s(t) as a complex data-carrying signals whose spectrum is contained between -B/2 to B/2, and consider a single sideband signal of the form,

$$h(t) = A + s(t)exp(i\pi Bt) \tag{5.34}$$

Where A is a constant. Here, Nyquist stability criterion can be used to ensure that $s(t)$ is a minimum phase signal. The condition of minimum phase signal is satisfied when $|A| > |s(t)|$ by guaranteeing Nyquist stability of the received signal. When $h(t)$ is a minimum-phase

signal, its phase $\phi(t)$ and absolute value $|h(t)|$ are uniquely related by Hilbert transform:

$$\phi(t) = \frac{1}{\pi} p.v. \int_{-\infty}^{\infty} dt' \frac{log[|h(t')|]}{t - t'} \tag{5.35}$$

where *p.v.* stands for *principal value*. The relationship depicted in Equation 5.35 can also be conveniently implemented in frequency domain as,

$$\tilde{\phi}(\omega) = isign(\omega)\mathcal{F}\{log[|h(t)|]\} \tag{5.36}$$

where $sign(\omega)$ is the sign function which is equal to 1 when $\omega > 0$, to 0 when $\omega = 0$ and to -1 when $\omega < 0$. Symbol $\mathcal{F}$ denotes the Fourier transform.

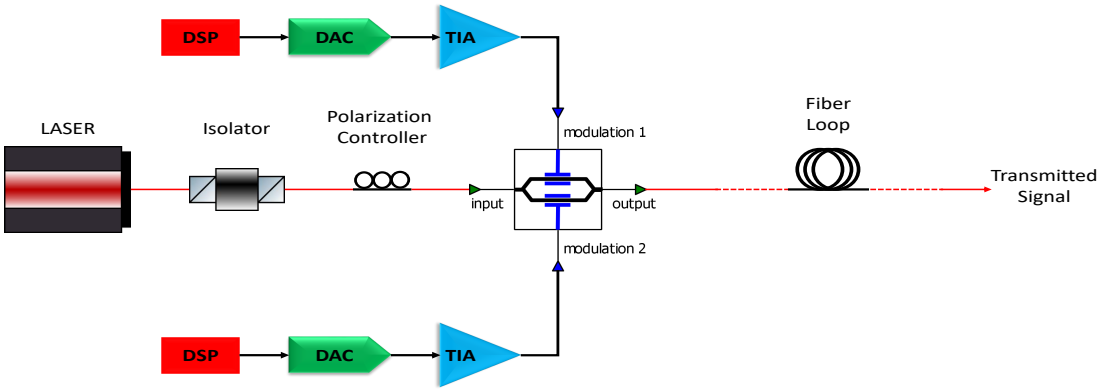


Figura 5.16: Simplified Coherent Transceiver

5.5.2 Tx side

Dual polarization(DP) DP-QPSK/DP-16QAM signals (1.25 Gbaud) to be generated at the Tx through an IQ modulator (Figure 5.17) driven by DAC mounted in a FPGA. Polarization-division multiplexing is emulated by dividing the signal in two signals using an optical splitter, where a delay of 12 symbols is applied in order to decorrelate the two polarization tributaries and then, both signals are joined orthogonally employing a polarization beam combiner (PBC).

5.5.3 Rx side

At the receiver side, signal is coherently detected using a simplified coherent receiver and a local oscillator. The optical signal is then converted into the electrical domain using two balanced photodetector (BPD), or alternatively four photodetector, and amplified by a transimpedance amplifier (TIA). Following that, the signals are sampled by two 8-bit 2.5 GSa/s ADC and the this digitized signal sent to the FPGA (Virtex-7) where all post-detection DSP implemented in real-time.

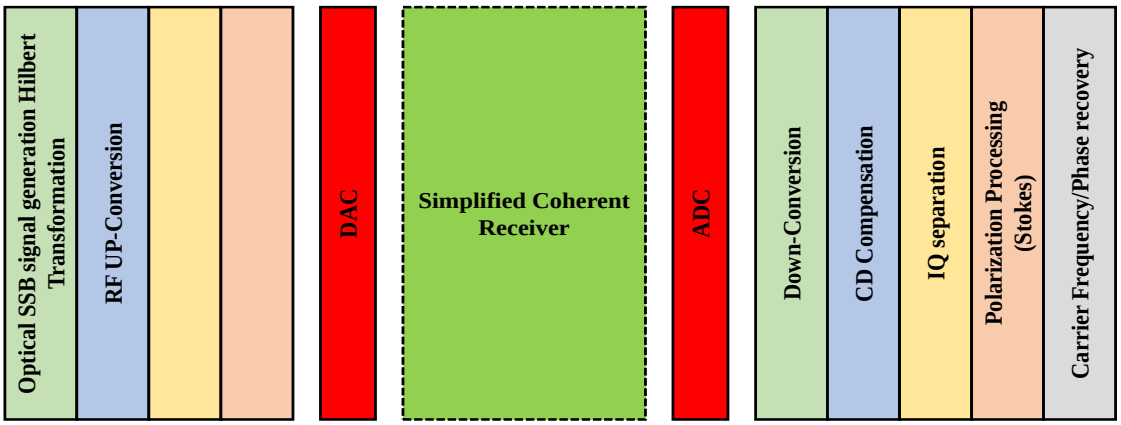


Figura 5.17: Tx/Rx DSP main subsystem

5.6 Oblivious Transfer with Discrete Variables

Student Name	:	Mariana Ramos
Starting Date	:	September 18, 2017
Goal	:	Oblivious transfer implementation 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 the messages sent by Alice, while at the end of the protocol Bob will learn the message b chose by him.
2. The protocol is oblivious, i.e Alice cannot learn anything about bit b and Bob cannot learning nothing about the other message $m_{\bar{b}}$.

5.6.1 OT Protocol Detailed

First of all, it is important to set the initial conditions of Bob and Alice's knowledge.

Lets establish for both Alice and Bob the message length ~~s~~ and the parameter ~~k~~. In addition, both know the following correspondence, where $+$ corresponds to *Rectilinear Basis* and $\times$ corresponds to *Diagonal Basis*:

~~*Bit*~~	~~*Basis*~~
0	$+$
1	$\times$

Secondly both Alice and Bob also know the bit correspondence for each direction for each basis. For *Rectilinear basis*, "$+$":

Bit	~~*Direction*~~
0	$\rightarrow$
1	$\uparrow$

and for *Diagonal Basis*, "$\times$":

Bit	~~*Direction*~~
0	$\searrow$
1	$\nearrow$

Third, only Alice knows information about messages m_1 and m_2. ~~She sets these messages using two bits for one, i.e bit 0 corresponds to "00"or "11"and bit 1 corresponds to "01"or~~

"10". This strategy will prevent eavesdropping actions, since, at the end, half of the bits are discarded. Thus, she sets the two messages $m_1' = \{0011\}$ and $m_2' = \{0001\}$ which correspond to $m_1 = \{00\}$ and $m_2 = \{01\}$, respectively.

1. Alice randomly generate a bit sequence with length ~~ks,~~ in which $k > 1$ ~~is a parameter defined at the beginning of the protocol.~~ Therefore, 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 values defined in S_{A2}.

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

3. ~~Before step 2,~~ Bob also randomly generates ks bits and fills the following array:

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

S_{B1} has the basis which Bob chooses to measure the photons sent by Alice.

4. When Bob has received photons from Alice, He measures them through the basis defined in S_{B1}:

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

and ~~H~~e will get ks results:

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

5. ~~After measurements have been taken, Bob informs Alice that he has already measured the photons and sends a *Hash Function*.~~

6. Once Alice has received ~~the *Hash Function*~~ from Bob, she sends throughout a classical channel the basis which she has used to codify the photons, $S_{A1} = \{0, 1, 1, 0, 0, 1, 0, 1\}$.

7. In order to know which photons were measured correctly, Bob does the operation $S_{B2} = S_{B1} \oplus S_{A1}$ and gets the following result:

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

. After that, Bob sends to Alice the information about the minimum number between "ones"and "zeros"

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

8. If $n < s$, being s the size of the message, Alice and Bob will repeat the steps from 1 to 7 and Bob adds more bits to sequence S_{B2}.

9. Next, lets assume that Bob is with the following sequences:

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

and

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

and Alice gets the following sequences:

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

and

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

10. Bob sends again to Alice $n = min(\#0, \#1) = 7$.

11. Alice checks if $n > s$ and reveals to Bob that she already knows that $n > s$.

12. Bob ~~sends to Alice~~ two sequences with size s:

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

and

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

where I_0 is the sequence of positions in which Bob was wrong and I_1 is the sequence of positions in which Bob was right. Thus, if Bob wants to know m_1 he sends to Alice the set $\{I_1, I_0\}$, otherwise if he wants to know m_2 he sends to Alice the set $\{I_0, I_1\}$.

13. Lets assume Bob sent $\{I_1, I_0\}$, Alice will ~~send~~ two keys

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

$$K_2 = \{0, 0, 0, 1\}$$

After that, Alice sends to Bob throughout a classical channel:

$$m = \{m_1 \oplus K_1, m_2 \oplus K_2\}$$

So, doing the above operation message m will be:

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

14. Using values which corresponds with positions given by I_1 and I_0 Bob will do the following operation:

$$m \oplus \{1, 0, 1, 0, 0, 1, 1, 0\}$$

in which Bob has guessed the last four values. He gets:

$$\{0, 0, 1, 1, 0, 1, 1, 0\}$$

The first four bits corresponds to message 1 ~~and he received~~ ~~{0, 0}~~, which is the right message m_1 and ~~{1, 1}~~ ~~which~~ is a wrong message for m_2.

5.6.2 OT Protocol - Potential Problems

There are two potential problems with the protocol described above:

1. In step 5 Bob may says to Alice that he has already measured the photon and it could be a lie.
2. In step 10 Bob may uses some values of I_1 in I_0 of positions which he knows are right in order to know correct information about message m_2.

This problems can be solved using *Bit Commitment* through *Hash Functions*.

5.6.3 Simulation

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

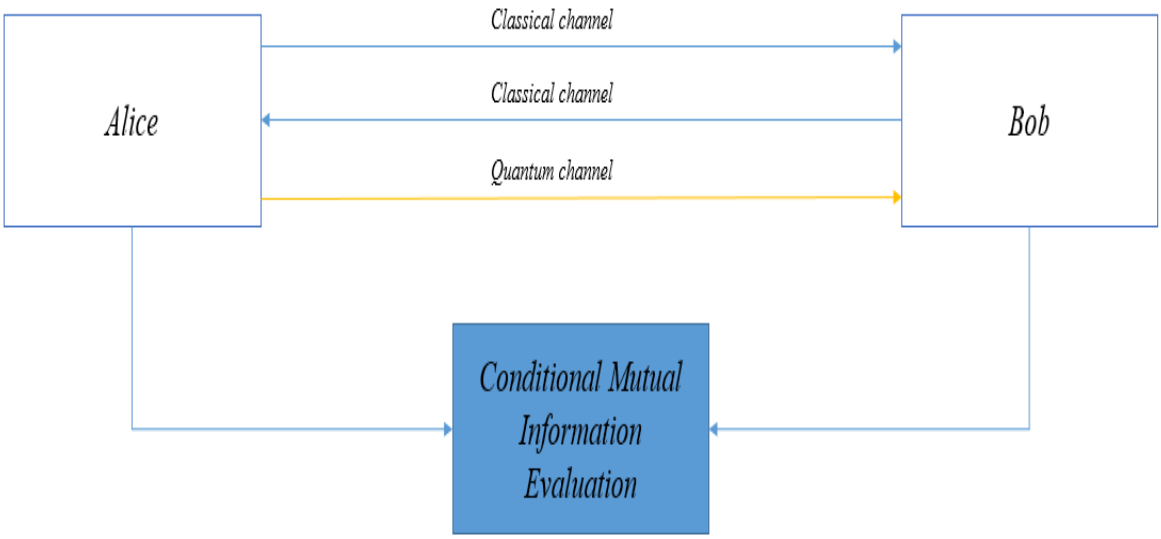


Figura 5.18: Experimental Setup

As one may see in figure 5.18 this simulation will have three 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 for the calculation of *Mutual Information*. The mutual information (MI) between Alice and Bob is defined in terms of their join distribution.

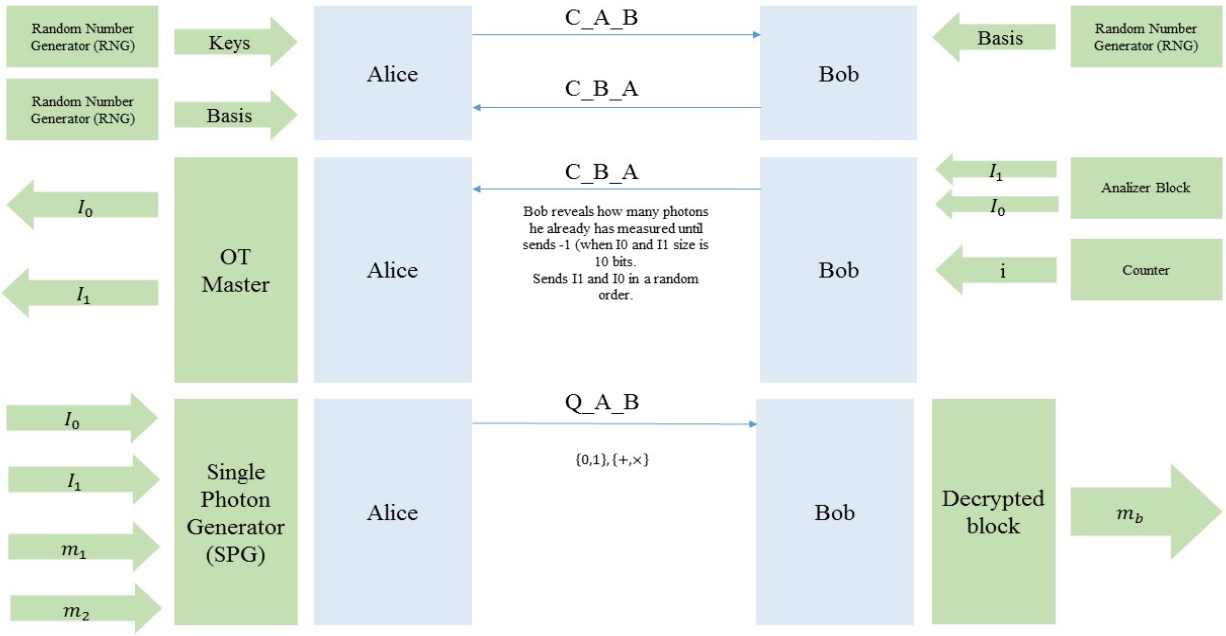


Figura 5.19: Functional Block Diagram

In figure 5.19 is presented the block diagram of the simulation that will be performed. There are two sides (Alice and Bob) and three channels throughout which they will communicate. For classical tasks Alice must have two *Random Number Generators* in order to generate random keys and basis. Likewise, Bob also need to generate the basis through which he will measure the received photons. Next, Bob must have two more blocks, one to analyze and process whether the basis information received by Alice match with his basis information, and another block for photons count. Furthermore, Alice must have a block for *Single Photon Generator* which receives four inputs to encrypt the photons to be sent. Finally, Bob has a block to decrypt messages received by Alice.

5.6.4 Experimental

In figures 5.20 and 5.21 are presented the experimental setup to be performed in the lab. Starting with Alice's side and then Bob's side.

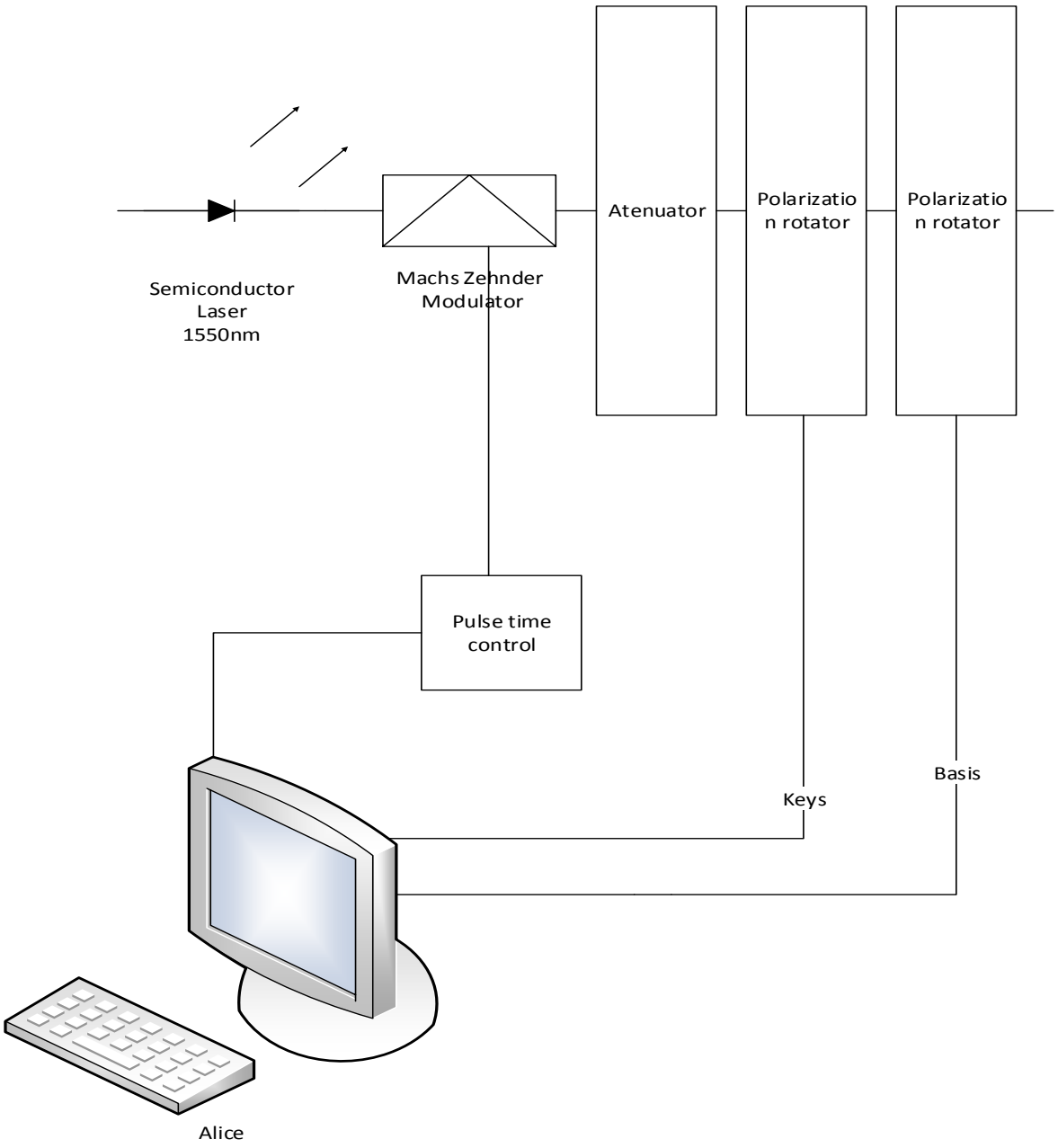


Figura 5.20: Quantum communication diagram - Alice's side

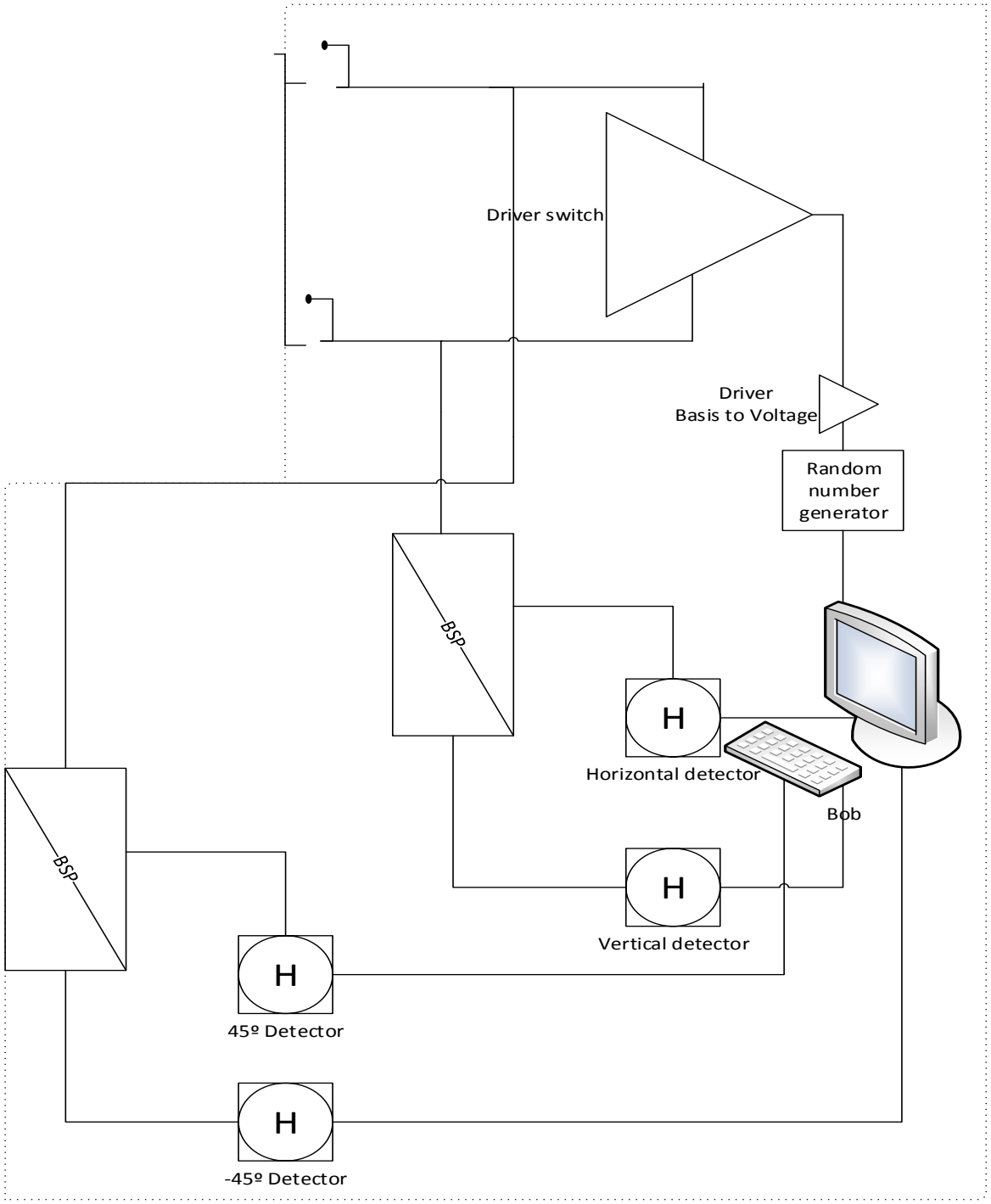


Figura 5.21: Quantum communication diagram - Bob's side

5.7 Radio Over Fiber Transmission System

Student Name	:	Celestino Martins
Starting Date	:	September 25, 2017
Goal	:	Radio Over Fiber.

Radio over fiber (RoF) technology comprises the transmission over fiber technology, where light is modulated by a radio signal and transmitted over an optical fiber link to provide broadband wireless services. The connection is established between a base station (BS) and central processing units (CPUs) as shown in the Figure 5.22.

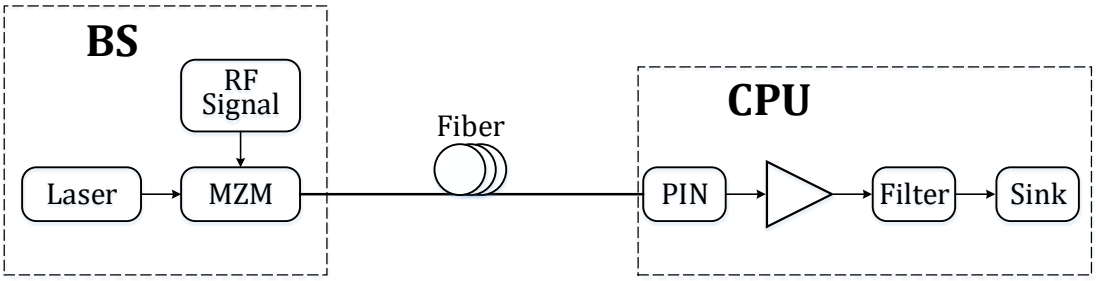


Figura 5.22: RoF transmission system block diagram.

In RoF systems, radio frequency (RF) signals are transported in optical form between a CPU and a set of BS before being radiated through the air. Each base station is adapted to communicate over a radio link with at least one user's mobile terminal located within the radio range of said BS. Considering an uplink connection in the Figure 5.22, the RF signals received from a mobile terminal are modulated using a laser and a mach-zehnder modulator (MZM) and then transmitted over optical link to CPU. In the 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.

5.7.1 Simulation

5.7.2 Experimental

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.

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

Input Parameters

Parameter: setConfidence **Parameter:** setMidReportSize

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 calculated BER as well as the estimated Confidence bounds for a given probability P. 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.

Input Signals

Number: 2
Type: Binary (DiscreteTimeDiscreteAmplitude)

Output Signals

Number: 1
Type: Binary (DiscreteTimeDiscreteAmplitude)

6.3 Binary source

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

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

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

Input Parameters

Parameter: mode{PseudoRandom}
(Random, PseudoRandom, DeterministicCyclic, DeterministicAppendZeros)

Parameter: probabilityOfZero{0.5}
(real $\in$ [0,1])

Parameter: patternLength{7}
(integer $\in$ [1,32])

Parameter: bitStream{"0100011101010101"}
(string of 0's and 1's)

Parameter: numberOfBits{-1}
(long int)

Parameter: bitPeriod{1.0/100e9}
(double)

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-*probabilityOfZero*.

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

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.

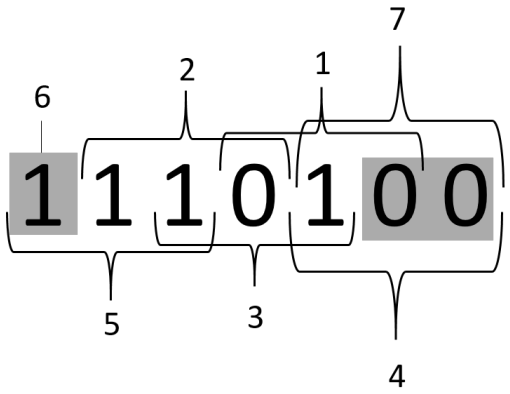


Figura 6.1: Example of a pseudorandom sequence with a pattern length equal to 3.

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.

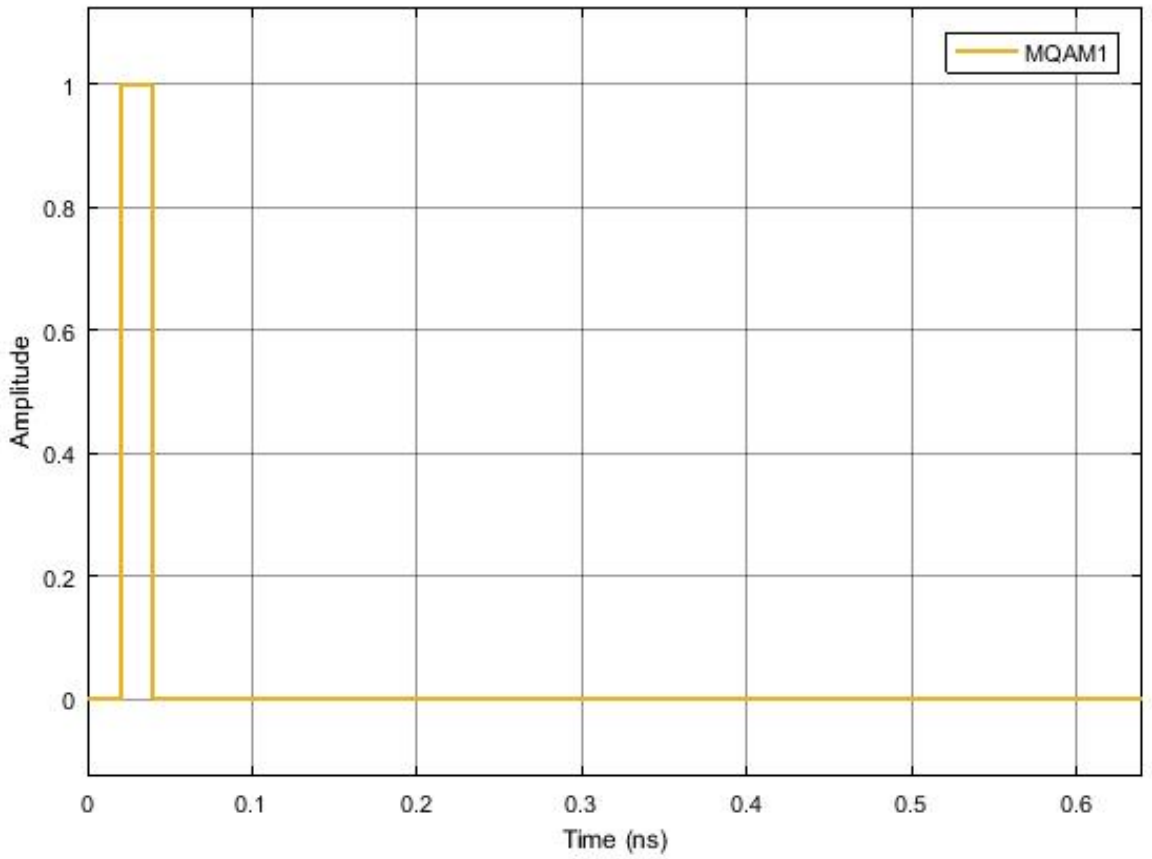


Figura 6.2: Binary signal generated by the block operating in the *Deterministic Append Zeros* mode with a binary sequence 01000...

6.4 Clock

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: period{ 0.0 };

Parameter: samplingPeriod{ 0.0 };

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

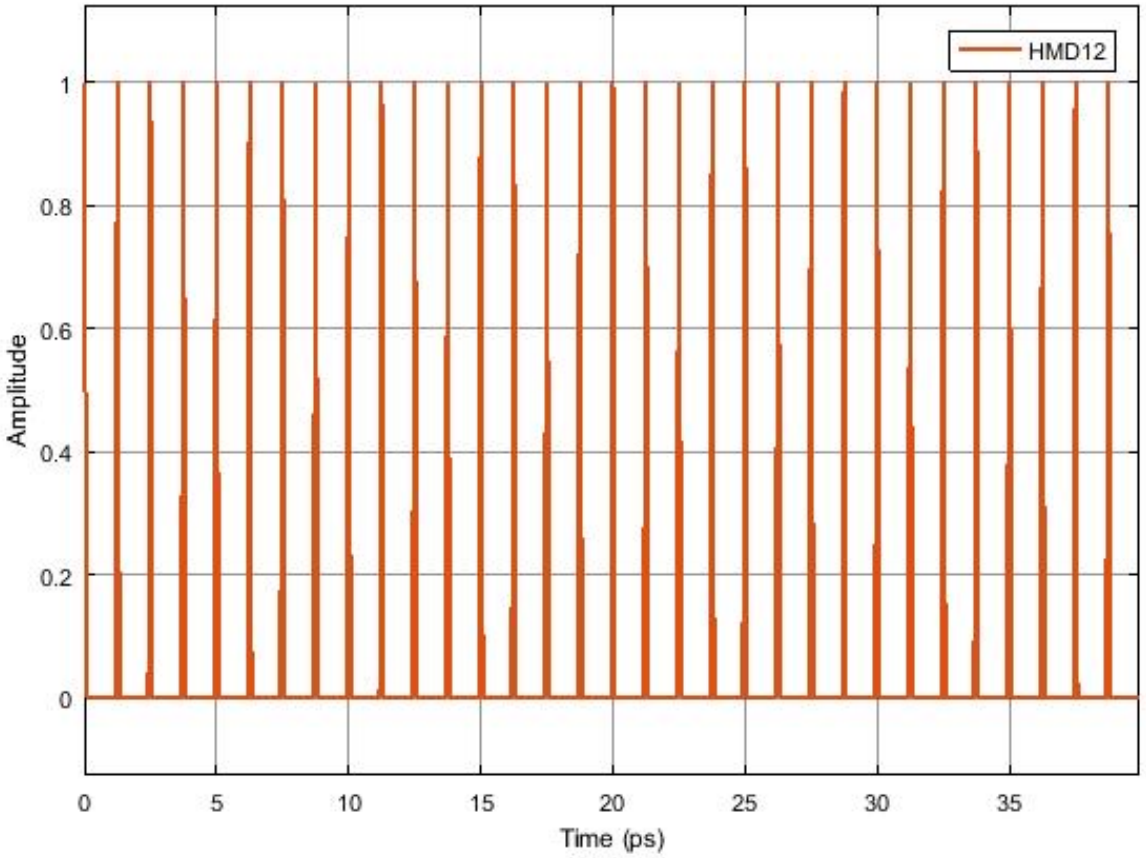


Figura 6.3: Example of the output signal of the clock

Sugestions for future improvement

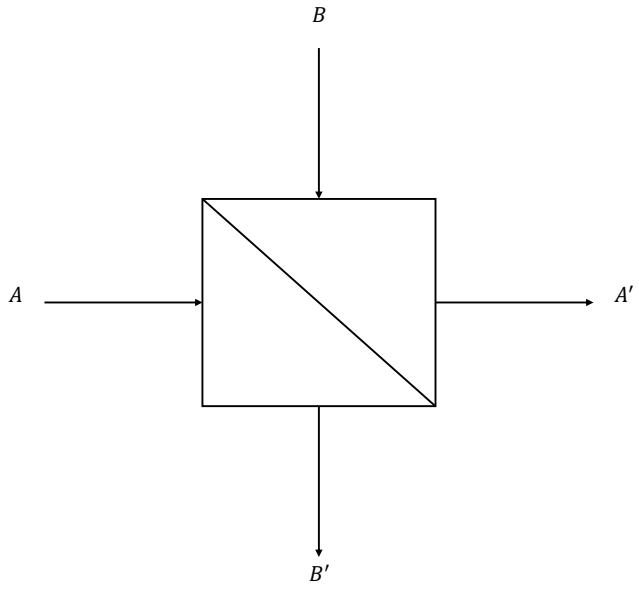


Figura 6.4: 2x2 coupler

6.5 Coupler 2 by 2

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

$$\begin{bmatrix} A' \\ B' \end{bmatrix} = \begin{bmatrix} T & iR \\ iR & T \end{bmatrix} . \begin{bmatrix} A \\ B \end{bmatrix} \tag{6.1}$$

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,

$$T = \sqrt{1 - \eta_R} \tag{6.2}$$

$$R = \sqrt{\eta_R} \tag{6.3}$$

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.

6.6 Decoder

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: `t_integer` m{ 4 }

Parameter: vector<`t_complex`> iqAmplitudes{ { 1.0, 1.0 },{ -1.0, 1.0 },{ -1.0, -1.0 },{ 1.0, -1.0 } };

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.

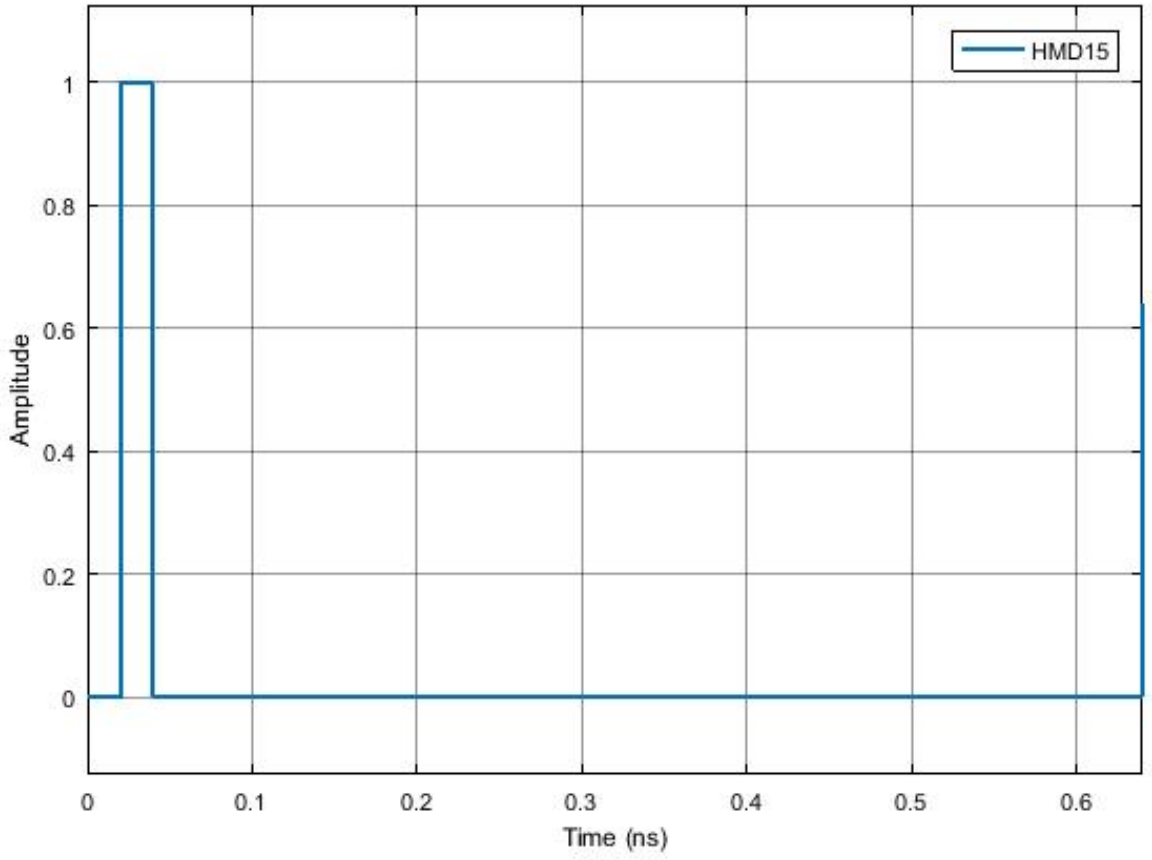


Figura 6.5: Example of the output signal of the decoder for a binary sequence 01. As expected it reproduces the initial bit stream

Sugestions for future improvement

6.7 Discrete to continuous time

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: numberOfSamplesPerSymbol{8}
(int)

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

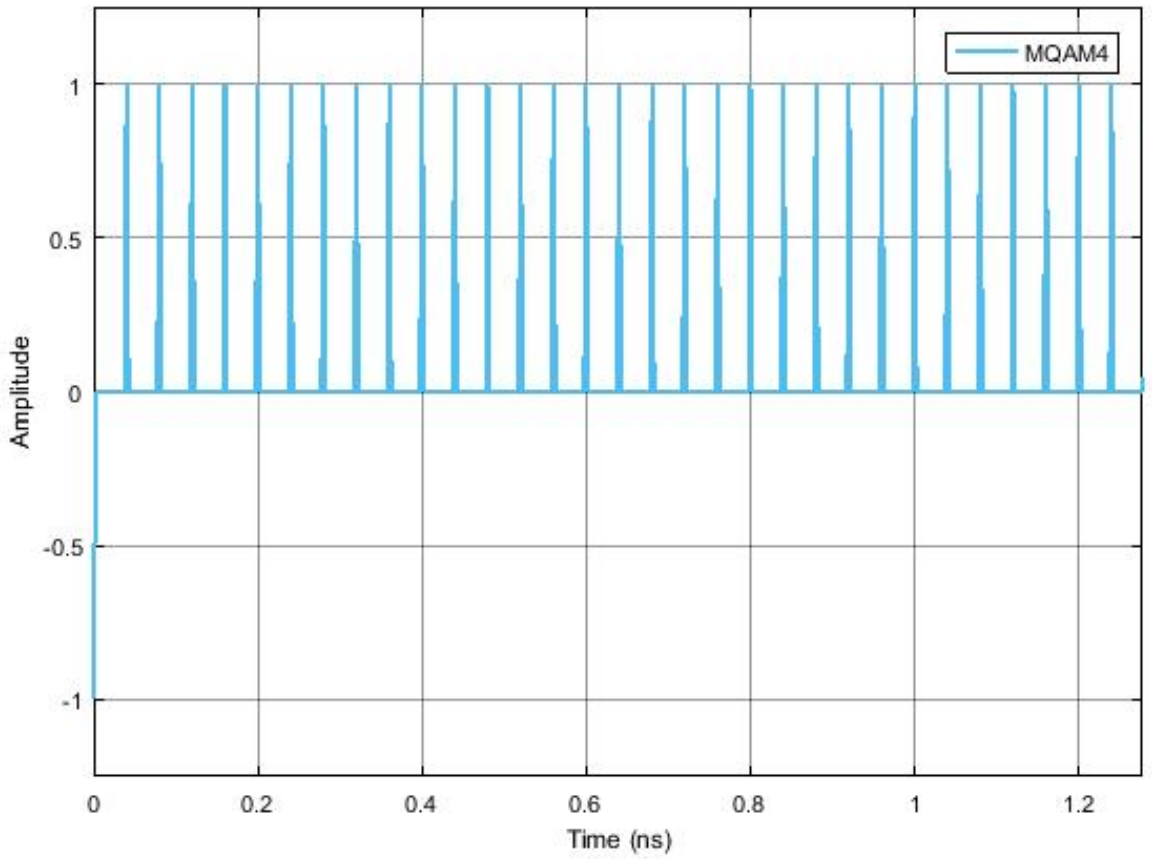


Figura 6.6: Example of the type of signal generated by this block for a binary sequence 0100...

6.8 Homodyne receiver

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 6.7.

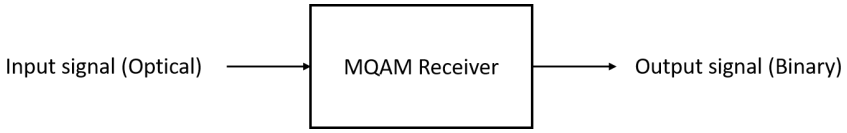


Figura 6.7: Basic configuration of the MQAM receiver

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 6.8) of code made up of several simpler blocks whose description can be found in the *lib* repository.

In can also be seen from figure 6.8 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*.

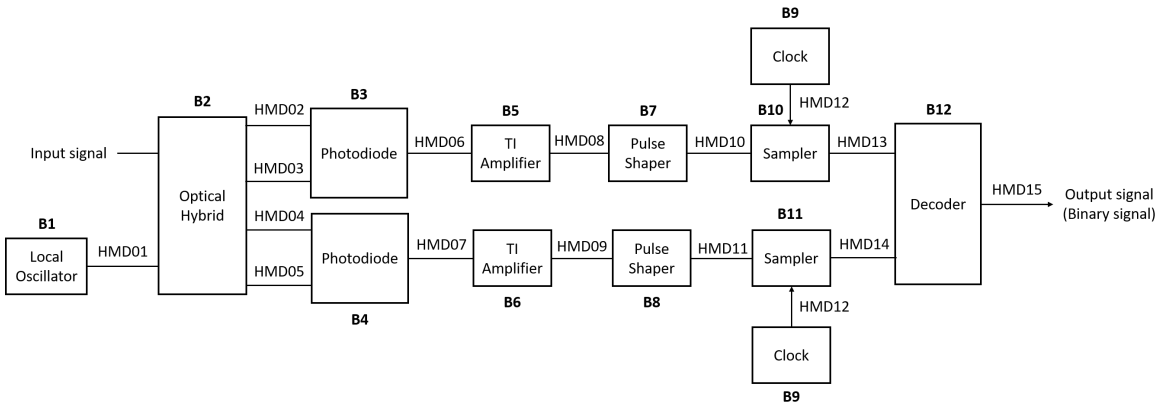


Figura 6.8: Schematic representation of the block homodyne receiver.

Input parameters

This block has some input parameters that can be manipulated by the user in order oto change the basic configuration of the receiver. Each parameter has associated a function that allows for its change. In the following table (table 6.2) 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 skipe	setSamplesToSkip	int(t_integer)	
Save internal signals	setSaveInternalSignals	bool	True or False
Sampling period	setSamplingPeriod	double	Givem by *symbolPeriod/samplesPerSy*

Tabela 6.1: 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

6.9 IQ modulator

This blocks accepts one inupt 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: outputOpticalPower{1e-3}
(double)

Parameter: outputOpticalWavelength{1550e-9}
(double)

Parameter: outputOpticalFrequency{speed_of_light/outputOpticalWavelength}
(double)

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 transmited to the receptor.

The binary signal is sent to the Bit Error Rate (BER) meaurement 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

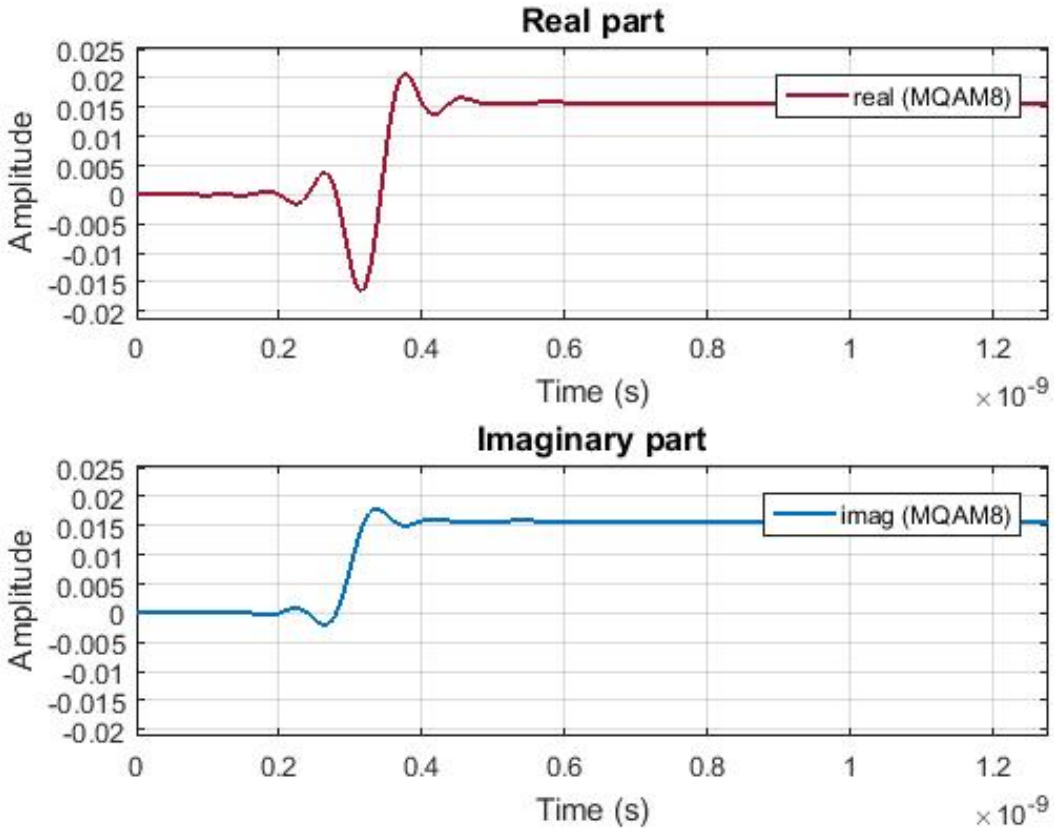


Figura 6.9: Example of a signal generated by this block for the initial binary signal 0100...

6.10 Local Oscillator

This block simulates a local oscillator which can have shot noise or not. It produces one output complex signal and it doesn't accept input signals.

Input Parameters

Parameter: opticalPower{ 1e-3 }

Parameter: wavelength{ 1550e-9 }

Parameter: frequency{ SPEED_OF_LIGHT / wavelength }

Parameter: phase{ 0 }

Parameter: samplingPeriod{ 0.0 }

Parameter: shotNoise{ false }

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 wlength);

void setPhase(double lOscillatorPhase);

void setShotNoise(bool sNoise);

Functional description

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

It can have shot noise or not which corresponds to setting the *shotNoise* parameter to True or False, respectively. If there isn't shot noise the the output of this block is given by $0.5 * \sqrt{OpticalPower} * ComplexSignal$. If there's shot noise then a random gaussian distributed noise component is added to the *OpticalPower*.

Input Signals

Number: 0

Output Signals

Number: 1

Type: Optical signal

Examples

Sugestions for future improvement

6.11 MQAM mapper

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: m{4}
(m should be of the form 2^n with n integer)

Parameter: iqAmplitudes{{ 1.0, 1.0 }, { -1.0, 1.0 }, { -1.0, -1.0 }, { 1.0, -1.0 }}

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 atributes 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 ilustrated in figure 6.10.

Input Signals

Number : 1

Type : Binary (DiscreteTimeDiscreteAmplitude)

Output Signals

Number : 2

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

Example

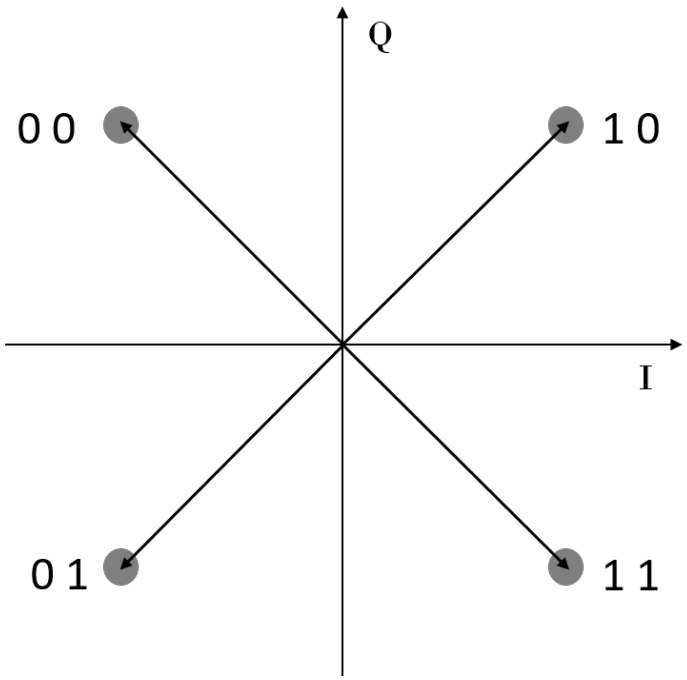


Figura 6.10: Constellation used to map the signal for m=4

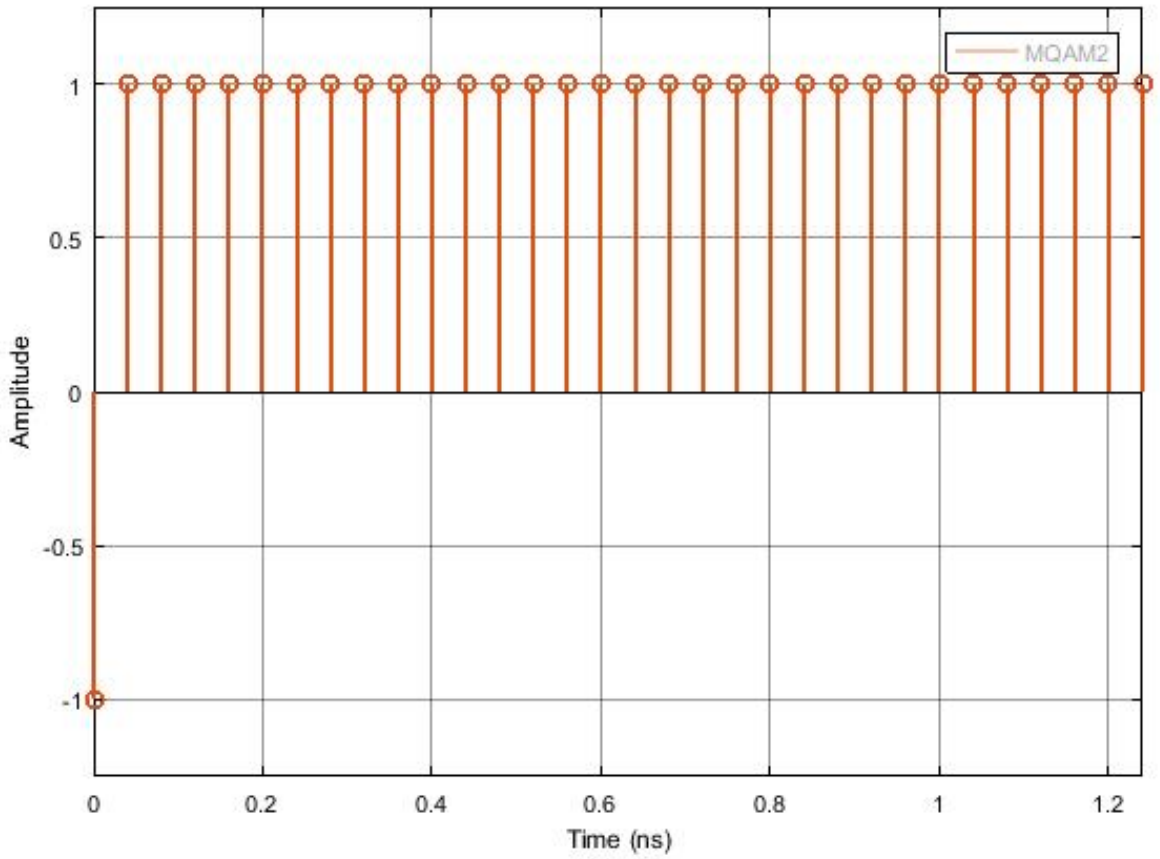


Figura 6.11: Example of the type of signal generated by this block for the initial binary signal 0100...

6.12 MQAM transmitter

This block generates a MQAM optical signal. It can also output the binary sequence. A schematic representation of this block is shown in figure 6.12.

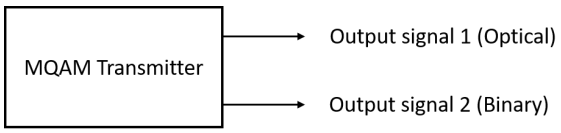


Figura 6.12: Basic configuration of the MQAM transmitter

Functional description

This block generates an optical signal (output signal 1 in figure 6.13). The binary signal generated in the internal block Binary Source (block B1 in figure 6.13) 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 6.13).

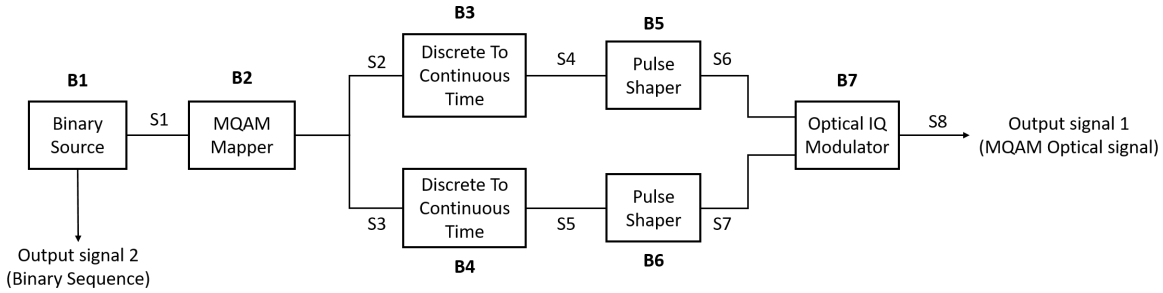


Figura 6.13: Schematic representation of the block MQAM transmitter.

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 6.2.

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 of factor	setRollOfFactor()	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 6.2: 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

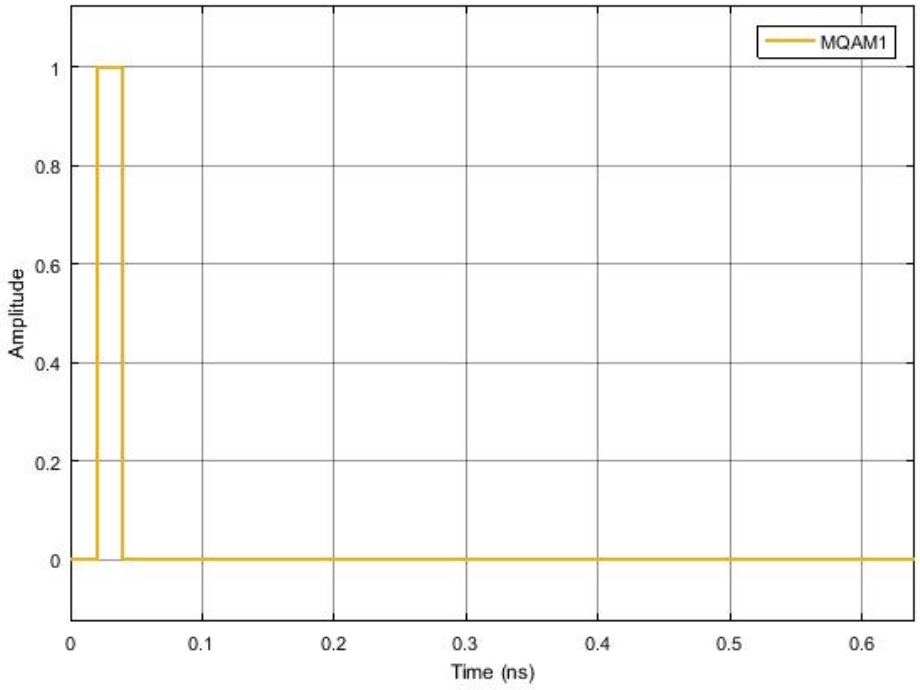


Figura 6.14: Example of the binary sequence generated by this block for a sequence 0100...

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.

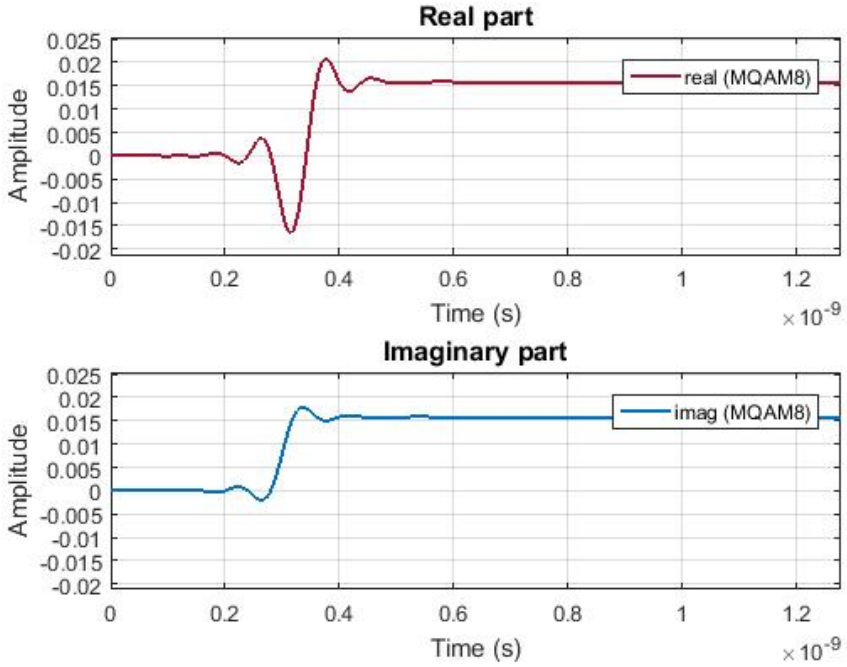


Figura 6.15: Example of the output optical signal generated by this block for a sequence 0100...