Downloaded from orbit.dtu.dk on: Apr 19, 2024

DTU Library

Secure Real-Time Transmitter for Continuous Variable Quantum Key Distribution

Solar Nikolic, Dino

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Solar Nikolic, D. (2019). *Secure Real-Time Transmitter for Continuous Variable Quantum Key Distribution*. Department of Physics, Technical University of Denmark.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

PhD Thesis

Secure Real-Time Transmitter for Continuous Variable Quantum Key Distribution

Department of Physics

Technical University of Denmark

Dino Solar Nikolic

Supervisor: Professor Ulrik Lund Andersen
Co-supervisor: Assistant Professor Tobias Gehring

December 2019

Acknowledgments

This work would not be possible without many people who supported me in various ways for the past three years. It is owed to Ulrik L. Andersen that I had a chance to work in his group in the exciting field of quantum cryptography.

My second supervisor Tobias Gehring is a person with whom I interacted the most. His invaluable knowledge on both quantum optics and engineering was critical for the success of this work.

Experimental quantum cryptography is a field that requires dedicated and coordinated work of a team of people who are focused on different parts of the system. Nitin Jain and Hou-Man Chin did an impressive job on the quantum key distribution optical setup and the receiver. Quantum random number generators were the focus of research of my fellow PhD student Arne Kordts, with whom I had fruitful discussions. Hossein Mani tackled the challenge of efficient error correction codes in continuous variable quantum key distribution. Ruben Grigoryan helped me in the beginning of my studies with his knowledge in electronics. I am greatful that the work of these people is implanted in this thesis.

Special credits go to our collaborator Cosmo Lupo from the University of Sheffield, who significantly contributed to my work with his knowledge of quantum information theory. I am especially thankful for his input for the topics of security of the randomness extraction and Gaussian sampling.

Besides Nitin, Hou-Man, Hossein and Tobias, I thank Haitham El-Ella for reviewing parts of my thesis.

I had a pleasure of spending a few weeks in Cambridge at Toshiba Research Europe thanks to Andrew Shields, Davide Marangon and Zhiliang Yuan.

I thank Alberto Nannarelli from DTU Compute for discussions in the field of FPGA technology.

Tine Hougaard Kitmøller is a person who made sure everything ran smoothly in our group.

Besides all the people who directly contributed to my work, I was lucky to have many friends, both in and out of DTU, who brought happiness to my life and gave me energy to pursue my dreams.

I dedicate this thesis to my parents who supported me unconditionally through my whole life.

Abstract

Quantum key distribution (QKD) is a cryptographic technique which utilizes quantum phenomena, such as superposition and entanglement, to enable the sharing of a secret random key for encrypted communication. Continuous variable QKD (CV-QKD) offers a practical way for performing a secure key exchange by means of broadband modulation of laser light, where the information is encoded in the light field quadratures.

We focus on building a real-time high-rate CV-QKD transmitter using flexible field programmable gate array (FPGA) chip technology. Composable security for CV-QKD has been proven only for Gaussian modulated quantum states. We implement a high-rate Gaussian sampler using cumulative distribution table (CDT) method. Important security parameters are calculated and the algorithms are realized with respect to maintaining the overall security of the CV-QKD system. Furthermore, digital signal processing (DSP) blocks are implemented in order to support fast (50 MBd/s) exchange of quantum states.

A necessary resource in any QKD protocol is a supply of uniformly distributed random numbers. Such a resource is particularly facilitated by quantum processes, which can be used to generate provably secure random numbers. We employ a high-rate vacuum fluctuation-based quantum random number generator (QRNG) with 8 Gb/s random number output. Our standalone QRNG system includes a real-time entropy testing based on monitoring the power spectral density according to a rigorous security model, instead of using common statistical tests that do not prove security. We address the issue of traditionally slow post-processing with a fast randomness extraction method based on Toeplitz hashing. To our knowledge, this is currently the fastest real-time QRNG implementation. A consequence of this is the fastest implementation of the Gaussian sampler used in CV-QKD. The thesis concludes with the evaluation of the transmitter as a part of our CV-QKD setup. It demonstrates for the first time an all-in-one implementation of transmitter functions for high-rate CV-QKD links. We recognize the potential for wide adoption of CV-QKD in near-future secure communication networks.

Keywords Quantum cryptography, CV-QKD, QRNG, FPGA, Gaussian modulation, Toeplitz hashing, Entropy testing

Resumé

Kvantenøglefordeling (QKD) er en teknik hvor en delt, hemmelig tilfældighed bliver skabt ved brug af kvantefænomener som superpositioner eller sammenfiltring. I særdeleshed tilbyder kvantenøglefordeling med kontinuerte variable (CV-QKD) en praktisk metode til at udveksle en hemmelig nøgle ved at gøre brug af bredbåndet modulation af laser lys, hvor informationen bliver indkodet i lysfeltets kvadraturer.

Vi fokuserer på at bygge en realtid, højrate CV-QKD udsender ved at bruge fleksible felt-programmerbare-port-tabel-chip (FPGA) teknologi. Kombinerbar sikkerhed for CV-QKD er blevet bevist for kun Gaussisk modulerede kvantetilstande. Vi implementerer en højrate Gaussisk sampler ved at gøre brug af den kumulative-fordelings-tabel-metode (CDT). Vigtige sikkerhedsparametre bliver udregnet, og algoritmerne bliver realiseret i forhold til at bevare den overordnede sikkerhed CV-QKD-systemet. Ydermere, bliver digital-signal-behandlings-blokke (DSP) implementeret for at undersøtte hurtig (50 MBd/s) fordeling af kvantetilstande.

En nødvendig ressource i enhver QKD protokol er jævnt fordelte tilfældige tal i særdeleshed beviseligt sikre tilfældige tal genereret af kvanteprocessor. Vi anvender en højrate vakuum fluktuations-baseret QRNG der kan generere tilfældige tal med 8 Gb/s. Vores alene-stående QRNG system inkluderer realtids entropi testning baseret på effekt-spektraltæthedsovervågning ifølge en stringent sikkerhedsmodel i stedet for at benytte almindelige (men ikke informations-teoretisk-sikre) statistiske test. Indtil videre er en flaskehals i tilfældighedsgenereringen efterbehandlingen. Vi adresserer dette problem med en hurtig tilfældigheds-udtræknings-metode baseret på Toeplitz hashing. Efter vores bedste overbevisning, er dette på nuværende tidspunkt den hurtigste realtids QRNG implementering. En konsekvens af dette er den hurtigste implementering af den Gaussiske sampler brugt i QKD.

Denne afhandling afsluttes ved at evaluere udsenderen som en del af vores CV-QKD setup. Den demonstrerer brugbarheden af FPGA-baseret teknologi inden for kvantekryptografi. Vi anerkender potentialet for bred adoption af CV-QKD in nærfremtids sikker-kommunikationsnetværk.

Contents

1 Introduction **1**

2 Gaussian quantum optics and information theory **5**
2.1 Introduction to Gaussian quantum optics 5
2.1.1 Quantization of the electromagnetic field and Fock space 5
2.1.2 Phase space representation 8
2.1.3 Gaussian states 9
2.2 Classical information theory 11
2.3 Quantum information theory 13

3 Introduction to continuous variable quantum key distribution **15**
3.1 Security of CV-QKD 15
3.2 Building blocks in CV-QKD experiments 18
3.2.1 Lasers 19
3.2.2 Beamsplitters and couplers 19
3.2.3 Modulators 20
3.2.4 Automatic bias controller 24
3.2.5 Homodyne detection 24
3.2.6 FPGA chips 29
3.3 CV-QKD protocols 36
3.4 CV-QKD setup 38
3.5 Noise sources in CV-QKD 39

4 Quantum random number generation **43**
4.1 The science of randomness generation 43
4.2 Vacuum fluctuation-based QRNG 48
4.3 Theory of randomness extraction 53
4.3.1 Universal hashing 53
4.3.2 Almost universal hashing 56
4.3.3 Leftover hash lemma and strong randomness extractors 56
4.4 FPGA implementation of the randomness extractor 60
4.5 FPGA implementation of the online entropy test 66

4.6 Aurora protocol with SATA interface 69
4.7 Standalone QRNG: architecture and performance 70

5 Real-time transmitter for continuous variable quantum key distribution 77
5.1 QKD transmitter architecture 77
5.2 Security of quantum key distribution with realistic Gaussian source 78
5.3 Generating discrete Gaussian random numbers 83
5.4 CDT algorithm for Gaussian sampling 86
5.5 FPGA implementation of the Gaussian sampler 87
5.6 FPGA implementation of the upsampling and upconversion modules 92
5.7 Performance of the QKD transmitter 94

6 Conclusion and outlook 101

A FPGA and software framework 103
A.1 AXI protocol preliminaries 103
A.2 Intellectual property (IP) cores 104
A.3 FPGA and ADC/DAC boards 108
A.4 Abaco (4DSP) framework 111

Chapter 1

Introduction

The quantum revolution started more than 100 years ago at the beginning of the 20th century with Planck's explanation of black body radiation. Planck's idea involved fundamental discrete units of electromagnetic radiation, energy quanta – *photons*. The discovery helped Einstein to develop a theory of the photoelectric effect not long after. It was soon realized that the new discoveries open the door to a new physics. In the following decades a completely new mathematical formalism was developed to describe the strange phenomena observed mostly in the tiny world of particles at small scales and short time frames. The famous double slit experiment revealed that photons[1] can still behave as a wave. This wave-particle duality is the consequence of the *superposition*. Superposition is one of the basic principles of quantum mechanics. It is without a classical analog and it leads to many other non-intuitive phenomena. The superposition principle was exemplified by Erwin Schrodinger with his famous thought experiment with a cat being dead and alive at the same time in a sealed box.

Another quantum mechanical effect first discovered by Einstein, Podolsky and Rosen [1], known as EPR paradox, deals with quantum systems of (at least) two particles. In their gedanken experiment two interacting particles with indeterminate momentum were to be sent far away from each other, for instance to separate ends of the galaxy. Then a measurement of one of the particles would instantly give information about the momentum of the other, because of the conservation of momentum. This would not be strange if in quantum mechanics the initial momentum of each of the particles is fundamentally without an exact value. Einstein referred to this phenomenon as a spooky action at a distance. It illustrates the non-local nature of quantum mechanics and the effect is called *quantum entanglement*.

In 1927 Werner Heisenberg derived his famous uncertainty principle [2] that

[1] This effect is not exclusive to photons however.

asserts a fundamental limit to the precision with which certain pairs of physical properties of a particle, known as canonically conjugate variables, such as position and momentum, can be known. The principle stems from the same mathematical formalism as superposition and entanglement and is closely related to *no-cloning theorem* [3]. The theorem states that it is impossible to create an identical copy of an arbitrary unknown quantum state. This has big implications to quantum information theory.

Classical information theory deals with efficient coding of information for storage or for transmitting through a noisy channel. It started the information technology revolution that made huge impact in the world especially in the last several decades. Information is quantified through entropy and the basic unit is *bit* that takes values 0 or 1. First ideas of connecting the information theory and quantum mechanics arose in the 70s. A quantum generalization of the basic unit of information is called quantum bit or *qubit*. In 1984 Bennett and Brassard invented the first *quantum key distribution* (QKD) protocol [4] using qubits, now called BB84, after which the field saw explosive growth.

Fast forward to today, we are in the midst of the second quantum revolution [5]. The first quantum revolution gave us new rules that govern physical reality and helped shape 20th century, while the second one will take the rules and develop new technologies that should shape 21st century. The world has seen a huge increase of investments in quantum technologies in the last several years.

This work focuses on quantum communication and the most mature technologies in the field – quantum key distribution and quantum random number generation. QKD is a subfield of quantum cryptography. In a typical QKD setup, there is a transmitter (Alice) who prepares a quantum state and sends it to a receiver (Bob). The goal of such communication is generation of a classical bit-string that is a shared secret between Alice and Bob, such that there is almost no leaked information about the string values to anybody else. In a typical model, an adversary (Eve) is in control of the channel and can perform physical operations allowed by quantum mechanics. No-cloning theorem is a core principle in QKD. It assures that Eve cannot perfectly copy a quantum state generated by Alice. This means that Alice and Bob, after Bob's measurement of the state, can detect the noise introduced by Eve's snooping as she inevitably introduces the noise due to the no-cloning theorem. QKD is seen to be a (partial) replacement to the current public key distribution protocols. Today's public key infrastructure is not only completely broken by the future quantum computers [6], but the non-existence of an efficient classical algorithm that breaks the schemes has not been proved. Considering ever rising cyber security threats

from hackers and governments, the increasing computational power and the fact that the world economy depends on secure cryptography, it is not hard to find motivation for developing information theoretic secure QKD systems.

Along with quantum key distribution there is another subfield of quantum cryptography that has significant impact to both technology and fundamental science and philosophy. *Quantum random number generation* is a process of obtaining true random numbers. Classical physics is seen as a deterministic theory. Only quantum mechanics allows true stochastic behaviour, assuming the validity of known laws of physics. Measurements of quantum states in principle give random results that cannot be predicted by anybody in the universe. This is therefore the only way of generating information theoretic secure random numbers. Random numbers are necessary in most cryptographic protocols, including QKD. Since quantum-generated randomness is the only information-theoretic true randomness, it is a mandatory part of every QKD setup. Devices that generate quantum randomness are quantum random number generators (QRNGs).

Field programmable gate array (FPGA) is a programmable chip technology that has seen an incredible growth for the past two decades. The applications range from telecommunications to medicine. The affordability paired with incredible processing power for some purposes makes this technology very attractive for QKD and QRNG. We use a high-end FPGA to perform research in one flavour of quantum cryptography that uses continuous variables (CV). It was shown twenty years ago that quantum continues variables can be used for quantum cryptography [7]. Continuous variable quantum key distribution (CV-QKD) and continuous variable quantum random generators (CV-QRNG) are still young technologies where proof-of-principle experiments were demonstrated only recently. CV technology promises lower cost and high data rates, therefore it is a perfect place where FPGAs can show their potential.

This thesis aims to be a tiny step forward towards building a widely used technology. Its nature is inherently interdisciplinary. Building a quantum cryptography system requires knowledge in quantum optics, telecommunications, electronics and security proofs. We hope we captured the important concepts from the relevant fields, so that the thesis could be read by researchers in both fundamental science and engineering.

Thesis structure

This thesis consists of three main parts. The introductory chapters (1, 2 and 3) include condensed information about topics that are prerequisite for

systematic approach to QKD and QRNG. In a sense they represent the author's journey of acquiring knowledge necessary for research and technology development in the field of CV quantum cryptography. The other two parts are dedicated to the QRNG and QKD implementations in more detail.

There are six chapters in total:

- **Chapter 1** is a general introduction with a brief historical overview of quantum technologies.
- **Chapter 2** introduces general theoretical concepts. It consists of the basic theory of Gaussian quantum optics, following by a brief introduction to classical and quantum information theory.
- **Chapter 3** is more specific and is introducing topics in CV-QKD, from the basic security model, through a description of important experimental elements, to the CV-QKD setup used as a testbed in this thesis.
- **Chapter 4** is dedicated to high-rate randomness extraction and real-time entropy tests developed for our implementation of quantum random number generator. The chapter also describes a standalone implementation of the QRNG.
- **Chapter 5** focuses on the FPGA implementation of the QKD transmitter. QRNG described in Chapter 4 is used as a randomness source here. Test results of the transmitter performance are presented at the end of this chapter.
- **Chapter 6** gives a conclusion of the work and the outlook for the future research.

Chapter 2

Gaussian quantum optics and information theory

The first section of this chapter introduces basic knowledge of quantum theory of light that lays foundation for understanding continuous variable quantum cryptography. It begins with the quantization the electromagnetic field. Illustrative notions such as Wigner representation and phase space are presented. Furthermore there is a brief overview of the Gaussian states and transformations. Finally, the last section gives a short introduction to entropy and Holevo information.

2.1 Introduction to Gaussian quantum optics

2.1.1 Quantization of the electromagnetic field and Fock space

We start by establishing a connection between a light wave and a (quantum) harmonic oscillator [8]. Let us consider a light field confined within a cavity of length L along the z-axis and polarized along x-axis:

$$E_x(z,t) = \sqrt{\frac{2\omega^2}{V\varepsilon_0}} q(t) \sin kz, \tag{2.1.1}$$

where ω is the angular frequency of the light, $k = \frac{\omega}{c}$ is the wave number, V is the cavity volume, ε_0 is vacuum permittivity and c is the speed of light in vacuum. $q(t)$ is a time dependent amplitude which is recognized as canonical position. Using Maxwell equations in vacuum the corresponding magnetic field is found:

$$B_y(z,t) = \frac{\mu_0\varepsilon_0}{k} \sqrt{\frac{2\omega^2}{V\varepsilon_0}} \dot{q}(t) \cos kz, \tag{2.1.2}$$

where $p(t) = \dot{q}(t)$ is canonical momentum and μ_0 is vacuum permeability. The energy of this field is given by its Hamiltonian (in units where the speed of light is $c = 1$)

$$H = \frac{1}{2}(p^2 + \omega^2 q^2). \tag{2.1.3}$$

Hamiltonian of the same form is found in a harmonic oscillator with the unit mass. Quantization of the harmonic oscillator is known procedure, so generalized position and momentum operators, and the ladder operators, are defined [9]. Quantum Hamiltonian of the field is

$$\hat{H}_k = \hbar\omega_k(\hat{a}_k^\dagger \hat{a}_k + \frac{1}{2}) = \frac{1}{2}(\hat{p}_k^2 + \omega_k^2 \hat{q}_k^2), \tag{2.1.4}$$

where $\hbar$ is the reduced Planck constant and $\hat{a}_k^\dagger$ and $\hat{a}_k$ are creation and annihilation operators respectively

$$\hat{a}_k^\dagger = \frac{1}{\sqrt{2\hbar\omega}}(\omega\hat{q}_k - i\hat{p}_k), \tag{2.1.5}$$

$$\hat{a}_k = \frac{1}{\sqrt{2\hbar\omega}}(\omega\hat{q}_k + i\hat{p}_k). \tag{2.1.6}$$

Here an index k is introduced which denotes a mode of the electromagnetic field. The notion of mode is found in different contexts and it is good to point out the right meaning for a particular case. Modes are orthogonal solutions for the propagation equations of light [10]. Two types of modes can be distinguished: *spatial modes* define a pattern of the field in the plane orthogonal to the polarization, and *temporal modes* which relate to time and frequency. When light is interfering, polarization is important, so sometimes polarization modes are also defined. In the case of this analysis, the index k denotes a frequency of the harmonic oscillator, or in other words, frequency mode of the light wave in the cavity described by equations 2.1.1 and 2.1.2.

The *number operator* of a mode is defined as $\hat{n}_k = \hat{a}_k^\dagger \hat{a}_k$. Eigenstates of the number operator are states with exact number of photons. These states span a *Fock space* $|n_k\rangle$. The ladder operators could also be defined by their action on a Fock state [12, 25]

$$\hat{a}_k^\dagger |n_k\rangle = \sqrt{n_k + 1}\,|n_k + 1\rangle\,, \tag{2.1.7}$$

$$\hat{a}_k |n_k\rangle = \sqrt{n_k}\,|n_k - 1\rangle\,, \quad \hat{a}_k |0\rangle = 0, \tag{2.1.8}$$

$$\hat{a}_k^\dagger \hat{a}_k |n_k\rangle = n_k |n_k\rangle\,. \tag{2.1.9}$$

Canonical position and momentum can be conversely defined using the ladder operators:

$$\hat{q}_k = \sqrt{\frac{\hbar}{2\omega_k}}(\hat{a}_k + \hat{a}_k^\dagger), \tag{2.1.10}$$

$$\hat{p}_k = -i\sqrt{\frac{\hbar\omega_k}{2}}(\hat{a}_k - \hat{a}_k^\dagger). \tag{2.1.11}$$

For these operators, the following commutations relations hold:

$$[\hat{q}_k, \hat{p}_{k'}] = i\hbar\delta_{kk'}, \quad [\hat{a}_k, \hat{a}_{k'}^\dagger] = \delta_{kk'}, \quad [\hat{a}_k, \hat{a}_{k'}] = 0, \tag{2.1.12}$$

where $\delta_{kk'}$ is Kronecker delta. One can see that the position and momentum are real and imaginary part of the annihilation operator respectively. Dimensionless counterparts of the position and momentum are defined

$$\hat{Q}_k \equiv \sqrt{\frac{\omega_k}{2\hbar}}\hat{q}_k, \quad \hat{P}_k \equiv \frac{1}{\sqrt{2\omega_k\hbar}}\hat{p}_k. \tag{2.1.13}$$

Measuring these variables one gets classical quadratures of the mode k, i.e. they represent real and imaginary parts of the complex amplitude [9]. The commutation relation for these operators is:

$$[\hat{Q}_k, \hat{P}_{k'}] = \frac{i}{2}\delta_{kk'} \tag{2.1.14}$$

This is the same as if condition $\hbar = 1/2$ is put to the commutation relation in equation 2.1.12. Since the two variables are not commuting, the uncertainties of measurements of both quadratures at the same time obey Heisenberg's relation:

$$\langle(\Delta\hat{Q}_k)^2\rangle\langle(\Delta\hat{P}_k)^2\rangle \geq \frac{1}{4}|\langle[\hat{Q}_k, \hat{P}_k]\rangle|^2 = \frac{1}{16}, \tag{2.1.15}$$

where $\langle(\Delta\hat{Q}_k)^2\rangle$ is the variance of operator $\hat{Q}_k$. This means that measuring both quadratures at the same time with perfect precision is not possible and this fact has a profound impact to continuous variable quantum protocols.

For the purpose of connecting the mean number of photons in a mode with the measured quadratures, it is useful to define the number operator using quadrature operators:

$$\hat{n}_k = \hat{a}_k^\dagger\hat{a}_k = \hat{Q}_k^2 + \hat{P}_k^2 - \frac{1}{2}. \tag{2.1.16}$$

Quadrature operators are observables with continuous eigenspectra. Their eigenvalues are continuous and real, $Q, P \in \mathbb{R}$ and their eigenstates $|Q\rangle$ and $|P\rangle$ identify two bases which are connected with Fourier transform (for mode k):

$$\hat{Q}_k \left|Q_k\right\rangle = Q_k \left|Q_k\right\rangle, \quad \hat{P}_k \left|P_k\right\rangle = P_k \left|P_k\right\rangle, \tag{2.1.17}$$

$$\left|Q_k\right\rangle = \frac{1}{2\sqrt{\pi}} \int dP_k e^{-iQ_k P_k/2} \left|P_k\right\rangle, \quad \left|P_k\right\rangle = \frac{1}{2\sqrt{\pi}} \int dQ_k e^{iQ_k P_k/2} \left|Q_k\right\rangle. \tag{2.1.18}$$

The vectors in the bases are orthogonal and complete:

$$\langle Q_k | Q_k' \rangle = \delta(Q_k - Q_k'), \quad \langle P_k | P_k' \rangle = \delta(P_k - P_k'), \tag{2.1.19}$$

$$\int_{-\infty}^{\infty} \left|Q_k\right\rangle \left\langle Q_k\right| dQ_k = 1, \quad \int_{-\infty}^{\infty} \left|P_k\right\rangle \left\langle P_k\right| dP_k = 1. \tag{2.1.20}$$

It is worth noting that quadrature eigenstates are not square-integrable and are unphysical. [1]

2.1.2 Phase space representation

Information about a physical system is contained in its quantum state defined by its density operator $\hat{\rho}$. The density operator is a positive operator with the unit trace. The operator can be defined for N-mode system and it represents the mapping $\hat{\rho} : \mathcal{H}^{\otimes N} \to \mathcal{H}^{\otimes N}$. An equivalent representation of quantum states that lies in *phase space* (also called quadrature or symplectic space) is *Wigner function*. It is a quasi-probability distribution defined over phase space and it mostly behaves as a normal probability distribution, though it can have negative values. It was first proposed in a paper from 1932 by Wigner [11].

Definition 2.1.1. (Wigner function)

$$W(\boldsymbol{x}) = \frac{1}{(2\sqrt{\pi})^{2N}} \int\limits_{\mathbb{R}^{2N}} d^{2N}\boldsymbol{\xi} e^{-i\boldsymbol{x}\Omega\boldsymbol{\xi}/2} \chi(\boldsymbol{\xi}), \tag{2.1.21}$$

where $\chi(\boldsymbol{\xi})$ is Wigner characteristic function that is defined as $\chi(\boldsymbol{\xi}) = \mathrm{Tr}[\hat{\rho} D(\boldsymbol{\xi})]$ where $D(\boldsymbol{\xi}) = \exp(i\boldsymbol{x}\Omega\boldsymbol{\xi})$ is Weyl operator[12]. The vector $\boldsymbol{x}$ is the vector of quadrature operators of N modes, and Ω is the matrix that defines commutation relations $[\boldsymbol{x}_i, \boldsymbol{x}_j] = i2\Omega_{ij}$.

Let $W(Q, P)$ be a Wigner function of a one-mode state (here we drop index k for simplicity). Even though the function is not a real probability

[1] It is impossible to produce a light field with state $|Q_k\rangle$ or $|P_k\rangle$ for some $Q_k, P_k \in \mathbb{R}$. This would mean infinite *squeezing* (see the following section).

distribution, it resembles it in some properties and is very convenient for depicting different bosonic states. The function is normalized:

$$\int W(Q,P)dQdP = 1. \tag{2.1.22}$$

By integrating over a quadrature, one gets the probability distribution for the other one

$$\int W(Q,P)dQ = \langle P|\,\hat{\rho}\,|P\rangle\,, \quad \int W(Q,P)dP = \langle Q|\,\hat{\rho}\,|Q\rangle\,. \tag{2.1.23}$$

Wigner function $W(Q,P)$ can also be defined as a function of a complex variable $W(\alpha)$, where $\alpha = Q + iP$. This variable is called the complex amplitude. It can be seen from 2.1.16 that $\hat{n} = \alpha^2$ (again we dropped the index k denoting a mode). Given the fact Wigner function behaves like a probability distribution, important parameters of a quantum state are the statistical moments. The first moment, also called the *displacement vector* is the mean value of the vector ($\boldsymbol{x}$):

$$\bar{\boldsymbol{x}} = \langle \boldsymbol{x} \rangle = \mathrm{Tr}(\boldsymbol{x}\hat{\rho}) \tag{2.1.24}$$

The second moment is in matrix form and is called the *covariance matrix* $\boldsymbol{V}$ whose elements are defined as:

$$V_{ij}^{\kappa} = \frac{1}{2}\langle\{\Delta\hat{x}_i^{\kappa}, \Delta\hat{x}_j^{\kappa}\}\rangle, \tag{2.1.25}$$

where $\Delta\hat{x}_i^{\kappa} \equiv \hat{x}_i^{\kappa} - \langle\hat{x}_i^{\kappa}\rangle$. Here ${\hat{x}_i}^{\kappa}$ is an element of $\boldsymbol{x}$ where κ counts the two quadratures. $\{,\}$ is the anti-commutator. Clearly, the diagonal elements of the covariance matrix are the variances of the quadratures:

$$V_{ii}^{\kappa} = V(\hat{x}_i^{\kappa}) = \langle\hat{x}_i^{\kappa 2}\rangle - \langle\hat{x}_i^{\kappa}\rangle^2, \tag{2.1.26}$$

The most important class of states for our work are the states that are fully described by the first two moments. They are called ***Gaussian states*** since the Wigner function for these states has a Gaussian shape:

$$W(\boldsymbol{x}) = \frac{1}{(2\pi)^N\sqrt{\det\boldsymbol{V}}}e^{-\frac{1}{2}(\boldsymbol{x}-\bar{\boldsymbol{x}})^T\boldsymbol{V}^{-1}(\boldsymbol{x}-\bar{\boldsymbol{x}})}. \tag{2.1.27}$$

2.1.3 Gaussian states

A state with zero photons $|0\rangle$ or the *vacuum state* is a Gaussian state. Annihilating a photon from the vacuum state gives the same state, hence the vacuum state is an eigenstate of the annihilation operator with zero eigenvalue $\hat{a}\,|0\rangle = 0$. The covariance matrix of such state is identity $\boldsymbol{V} = \mathbb{1}$,

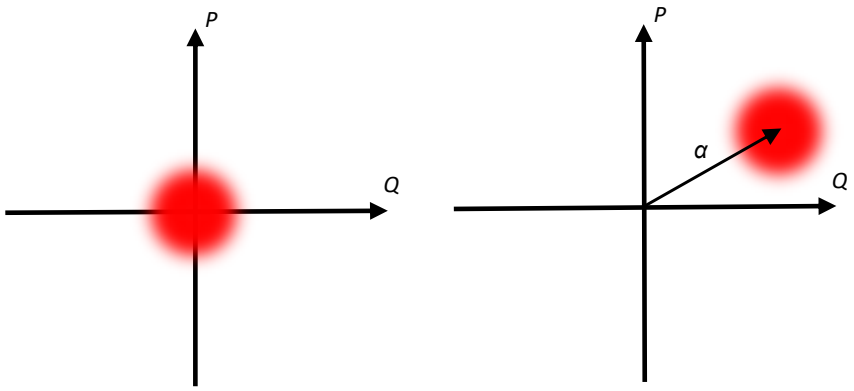


Figure 2.1: Vacuum state and displaced state (coherent state) in phase space

therefore the variances of the quadratures are 1. This state is symmetrical and the variances are a consequence of the uncertainty principle. This uncertainty causes a noise when measuring the quadratures. This noise is called *shot noise* or *vacuum noise*. Shot noise is one of the central concepts in continuous variable quantum key distribution and quantum random number generation.

The *displacement operator* is a Weyl operator for the complex variable α. This operator 'displaces' the vacuum state in the phase space. Fig. 2.1 shows the Wigner function of the vacuum state and a displaced state in the phase space. Displaced vacuum state is called the *coherent state* and is characterized by its complex amplitude $|\alpha\rangle = D(\alpha)\,|0\rangle$. The covariance matrix of the coherent state stays identity, but the mean values of the quadratures are now $\bar{\boldsymbol{x}} = (Q, P)^T$.

Coherent states [13] were discovered by Scroedinger in 1926 when he was solving his equation with the aim of finding a solution for quantum harmonic oscillator that is asymptotically close to classical states. They are the states with the smallest possible uncertainty. Intuitively, they are the closest to perfectly noiseless classical states.

Coherent states are the eigenstates of the annihilation operator:

$$\hat{a}\,|\alpha\rangle = \alpha\,|\alpha\rangle\,. \tag{2.1.28}$$

In the number basis, coherent states are expanded as:

$$|\alpha\rangle = e^{-\frac{1}{2}|\alpha|^2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}}\,|n\rangle\,. \tag{2.1.29}$$

The mean energy of a coherent state is

$$\langle H\rangle = \langle\alpha| H |\alpha\rangle = \hbar\omega \langle\alpha| \hat{a}^\dagger a + \tfrac{1}{2} |\alpha\rangle = \hbar\omega(|\alpha|^2 + \tfrac{1}{2}). \tag{2.1.30}$$

As we have seen, the number states $|n\rangle$ form an orthogonal basis. This is not the case with coherent states as they are not mutually orthogonal and form an overcomplete basis. The overlap between two coherent states is

$$|\langle\beta|\alpha\rangle|^2 = e^{-|\beta-\alpha|^2}. \tag{2.1.31}$$

It was shown that coherent states are symmetrical in the phase space as the variances of the both quadratures are the same. The uncertainty principle does not forbid a quadrature variance to be smaller than 1 (in natural units) as long as the other quadrature variance increases. Such states exist and they are called *squeezed states*. They can be produced by pumping a nonlinear crystals with a bright laser. Even though there are quantum cryptography protocols using the squeezed states [14], they are not the focus of this thesis, and due to experimental complexities, this field is yet to be explored in more depth [15, 16].

Thermal state is a mixture of eigenstates of a Fock space. This state is Gaussian and in Fock space it is represented with the density operator

$$\hat{\rho}(\bar{n}) = \sum_{n=0}^{+\infty} \frac{\bar{n}^n}{(\bar{n}+1)^{n+1}} |n\rangle\langle n|, \tag{2.1.32}$$

where $\bar{n} = \mathrm{Tr}(\hat{\rho}\hat{n})$is the mean number of photons in the bosonic mode. It can be shown that thermal states maximize the von Neumann entropy $S = -\mathrm{Tr}(\hat{\rho}\log\hat{\rho})$.[2] Their Wigner function is Gaussian with zero mean and covariance matrix $V = (2\bar{n}+1)\mathbb{1}$.

2.2 Classical information theory

Classical information theory was born in 1948 with the work of Claude Shannon [17]. There is a vast number of resources on the topic. Definitions in the following paragraphs can be found in [18].

The starting point is Shannon entropy:

Definition 2.2.1. (Shannon entropy). Let there be a random variable A, with alphabet $\mathcal{A}$ that is distributed according to the probability distribution

[2] Von Neumann entropy is defined in Section 2.3.

$P_A(a), a \in \mathcal{A}$. Shannon entropy of the random variable is defined as

$$H(A) = -\sum_{a\in\mathcal{A}} P_A(a)\log P_A(a). \tag{2.2.1}$$

Entropy quantifies the average uncertainty about a stochastic source. If the variable is continuous, the sum becomes an integral

$$H(A) = -\int_{\mathcal{S}} P_A(a)\log P_A(a)da, \tag{2.2.2}$$

where $\mathcal{S}$ is the support of the random variable A, i.e. it is the subset of $\mathcal{A}$ for which $P(a) > 0$.

The joint entropy of two discrete random variables A and B with alphabets $\mathcal{A}$ and $\mathcal{B}$ respectively is

$$H(AB) = -\sum_{a\in\mathcal{A}}\sum_{b\in\mathcal{B}} P_{AB}(a,b)\log P_{AB}(a,b). \tag{2.2.3}$$

Conditioning distribution A on the outcomes of distribution B, one gets the conditional entropy

$$H(A|B) = H(AB) - H(B) = -\sum_{a\in\mathcal{A}}\sum_{b\in\mathcal{B}} P_{AB}(a|b)\log P_{AB}(a|b). \tag{2.2.4}$$

Now the mutual information between the two variables can be defined. Mutual information quantifies the amount of information one obtains about a random variable when observing another.

$$I(A:B) = H(A) - H(A|B). \tag{2.2.5}$$

Shannon entropy quantifies the amount of randomness of a randomness source *on average*, which means it is asymptotic in nature. A generalization of Shannon entropy is *Rényi entropy*:

Definition 2.2.2. Rényi entropy of order α, where $\alpha \geq 0$ and $\alpha \neq 1$ is defined as

$$H_\alpha(A) = \frac{1}{1-\alpha}\log\left(\sum_{a\in\mathcal{A}} P_A(a)^\alpha\right). \tag{2.2.6}$$

Shannon entropy is obtained as a special case of Rényi entropy when $\alpha \to 1$. Another special case of Rényi entropy is *min-entropy* when the parameter converges to infinity $\alpha \to \infty$:

Definition 2.2.3. (Min-entropy) The min-entropy $H_{\min}(A)$ (also often denoted H_∞ in literature) of A is

$$H_{\min}(A) = \min_{a\in\mathcal{A}} \log\frac{1}{P_A(a)}. \tag{2.2.7}$$

Min-entropy is important quantity in quantum cryptography as it is the main measure of randomness of a finite bit array. It is used in both quantum key distribution to quantify the amount of secret bits shared by Alice and Bob [19], and quantum random number generation and randomness extraction [20, 21] to quantify the amount of randomness in a finite QRNG output. While Shannon entropy tells how many random bits are in a sample *on average*, the min-entropy gives the *lower bound* on the number of random bits *in every sample*.

2.3 Quantum information theory

Quantum information theory is an extension of Shannon theory by including the quantum formalism. A good resource in this field is [22].

The quantum analogy of Shannon entropy is *von Neumann entropy*, defined for a quantum state $\hat{\rho}$.

Definition 2.3.1. (Von Neumann entropy)

$$S(\hat{\rho}) = -\operatorname{Tr}(\hat{\rho}\log\hat{\rho}), \tag{2.3.1}$$

where Tr is the trace of an operator.

Von Neumann entropy can be seen as a measure of uncertainty after measuring the quantum state $\hat{\rho}$. For example, if a pure state (where $\operatorname{Tr}(\hat{\rho}^2) = 1$) is measured in appropriate basis, no uncertainty remains, therefore $S(\hat{\rho}) = 0$. The entropy is invariant under unitary operations

$$S(\hat{\rho}) = S(\hat{U}\hat{\rho}\hat{U}^\dagger). \tag{2.3.2}$$

This means von Neumann entropy remains unchanged under Gaussian unitaries. For separable states (of modes A and B) the entropy is additive

$$S(\hat{\rho}_A \otimes \hat{\rho}_B) = S(\hat{\rho}_A) + S(\hat{\rho}_B). \tag{2.3.3}$$

For a two-mode state $\hat{\rho}_{AB}$ one can define the joint entropy

$$S(\hat{\rho}_{AB}) = -\operatorname{Tr}(\hat{\rho}_{AB}\log\hat{\rho}_{AB}). \tag{2.3.4}$$

If the partial trace is defined (over mode B and similarly for A) of the two-mode state $\hat{\rho}_A = \operatorname{Tr}_B \hat{\rho}_{AB}$, then the marginal entropies are:

$$S(\hat{\rho}_A) = -\operatorname{Tr}(\hat{\rho}_A\log\hat{\rho}_A), \tag{2.3.5}$$
$$S(\hat{\rho}_B) = -\operatorname{Tr}(\hat{\rho}_B\log\hat{\rho}_B). \tag{2.3.6}$$

The conditional von Neumann entropy is defined analogously to its classical counterpart:

$$S(A|B) = S(AB) - S(B). \tag{2.3.7}$$

Similarly the quantum mutual informationcan be defined. A quantity that is important in QKD and QRNG security formalism, and does not have a classical analog, is *Holevo information*. It is also called Holevo bound and it establishes the upper bound to the amount of information that can be known about a quantum state.

Definition 2.3.2. (Holevo information) Let a transmitting party prepare a set of quantum states $\{\hat{\rho}_B^j\}$ with probability $P_A(j)$, so the receiving party receives the state $\hat{\rho}_B = \sum_j P_A(j)\hat{\rho}_B^j$. Then the amount of information extractable by an optimal measurement of the state $\hat{\rho}_B$ is bounded by Holevo information

$$\chi(A:B) = S(\hat{\rho}_B) - \sum_j P_A(j) S(\hat{\rho}_B^j). \tag{2.3.8}$$

For example, this quantity is calculated in QKD protocols with respect to eavesdropper's quantum system in order to upper bound the accessible information of the secret key to her.

Chapter 3

Introduction to continuous variable quantum key distribution

3.1 Security of CV-QKD

Even though theoretical security formalism is not the focus of this thesis, it is important to be aware of security notions as, in the end, the goal of our work is implementation of secure quntum cryptographic systems. This section gives an overview of security concepts in CV-QKD systems with coherent states. The core principle of QKD security is the quantum no-cloning theorem. The theorem forbids perfect copying of an arbitrary quantum state with the unit probability. This means that quantum states sent by Alice (transmitter) will be disturbed (on average) if measured by Eve (eavesdropper) on their way to Bob (receiver)(Fig. 3.1[1]). As the no-cloning theorem assumes arbitrary states, Alice needs a source of true randomness at her disposal. Using CV-QKD protocols based on coherent states, the information is encoded into the quadrature of the transmitted light. When Eve tries to eavesdrop by intercepting a portion of the transmitted light, she inherently introduces noise which is subsequently detected by Bob as he receives the signal. Eve's optimal strategies for measuring quantum states, in order to gain as much information as possible, are known and are subject of research in quantum cloning [23]. We can put together a standard list of attacks often found in the literature, that Eve can perform on the quantum state that Alice is sending. The attacks are listed from the weakest to the most powerful [19, 24, 25]:

[1] In this thesis we are working only with prepare-and-measure protocols where Alice prepares a quantum state and sends it to Bob who measures it. On the other hand there are *entanglement-based* protocols where Alice prepares a two-mode state and keeps and measures one half of the state, while the other is sent to Bob.

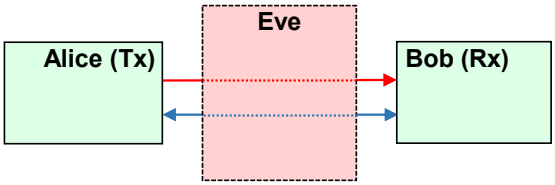


Figure 3.1: Three parties in a *prepare-and-measure* QKD protocol: Alice (transmitter) and Bob (receiver) are trusted, while Eve (eavesdropper) is the untrusted party who controls the channel. Unidirectional quantum channel (from Alice to Bob) is designated by the red line and bidirectional authenticated classical channel is designated by the blue line.

- **Individual attack** Eve performs independent and identically distributed (i.i.d.) attack on each pulse separately. She measures her quantum states before Bob performs post-processing steps.
- **Collective attack** Eve performs an i.i.d. attack and measures her state collectively after Bob performs post-processing steps.
- **Coherent attack** Eve couples a pre-entangled multimode ancilla with all the pulses from Alice. She optimally measures her system after Bob performs post-processing. This should be the most powerful attack and notoriously hard to implement in real world.

Fig. 3.2 shows a model of collective attack to a CV-QKD system. This model is used for proving security to the CV-QKD system used as an experimental testbed for the QKD transmitter developed for the thesis. The channel is fully controlled by Eve, so the losses and the noise of the channel are untrusted. Trusted loss and noise originating from inside Bob's box are also defined and are quantified during receiver calibration.

First continuous variable QKD protocols proven to be secure were protocols with squeezed states [26, 27], shortly followed by the first protocol with coherent states [28]. For all the earlier protocols, security was proven in the *asymptotic* regime where Alice sends an infinite number of pulses to Bob, so the quantum state is exactly defined by obtained average parameters. The Devetak-Winter formula gives the key rate for asymptotic regime and collective attacks [29]:

$$R_{\text{coll}}^{\text{asymptotic}} = \beta I(A:B) - \chi(B:E). \tag{3.1.1}$$

The key rate is the classical mutual information between Alice's and Bob's symbols reduced by Holevo information between Bob's and Eve's quantum system. β is *reconciliation efficiency*. These quantities are calculated within a security model and are a function of trusted and untrusted noise and

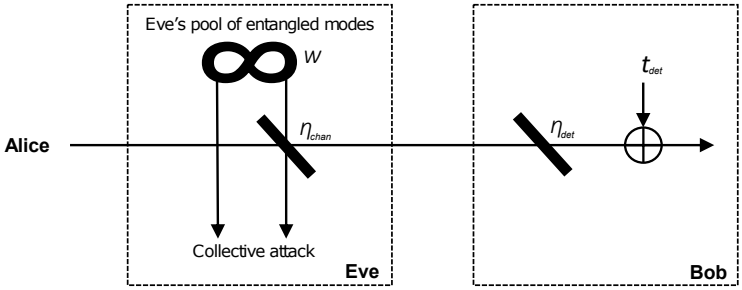


Figure 3.2: QKD model under optimal entangling cloner collective attack. Eve prepares a pool of entangled Gaussian modes and injects one part of the entangled state to the channel. η_{chan} – channel transmittance modelled with a beamsplitter (untrusted loss). w – the variance of each of the Eve's modes. η_{det} – detector transmittance (trusted loss). t_{det} – variance of trusted noise. The variance of the untrusted noise at the detector is $\eta_{\text{det}}(1-\eta_{\text{chan}})w$.

loss. Information reconciliation is a post-processing step where Alice and Bob use an error correcting code (such as a LDPC code) to match the data strings obtained after Bob's measurement. In the equation above the *reverse reconciliation* is used, where Alice is changing her data to match Bob's, after Bob provides the error correction syndrome through authenticated classical channel. It has been proven that the reverse reconciliation is better than the *direct reconciliation* [30], in which case Holevo information would be calculated between Alice's and Eve's state: $\chi(A:E)$. In CV-QKD protocols with coherent states and Gaussian modulation, Alice and Bob can calculate the covariance matrix V_{AB} of the bipartite state [2] ρ_{AB}. The upper bound on the Holevo information is a function of the covariance matrix. The matrix values depend on transmissivity and noise in the system.

There are three main steps in post-processing:

- **Parameter estimation** After Bob performs the measurement of the quantum state sent by Alice, the two choose a random set of symbols for which they calculate the covariance matrix V_{AB} in order to get the upper bound for $\chi(B:E)$.

- **Information reconciliation** Bob sends the error correction syndrome to Alice. Some information is leaked to Eve in this step.

[2] A are the symbols Alice sends, and B are the symbols Bob receives. ρ_{AB} is generally calculated in the entanglement-based scheme where Alice obtains her symbols by measuring one part of two-mode squeezed vacuum state while the other part is sent to Bob. Such a scheme is proven to be equivalent to prepare-and-measure and it is used due to the fact it is easier to prove security this way.

- **Privacy amplification** After reconciling, Alice and Bob share identical strings. Eve has some information about it, therefore a privacy amplification protocol is performed to reduce Eve's information about the secret key to a negligible value. Most often, universal hashing is used for this step, same as with randomness extraction for QRNGs, however with much larger block lengths.

In the last years there has been a great progress in proving security of Gaussian modulated CV-QKD for the finite regime (finite key instead of infinite one) and proving the optimality of Gaussian collective attacks [31, 32].

3.2 Building blocks in CV-QKD experiments

Here we describe telecommunication components used for CV-QKD. Alice's system consists of:

- **Signal laser** – produces coherent light beam which is the carrier of information.
- **Optical modulator** – modulates the laser light and therefore encode information in the carrier.
- **Bias controller** – an electrical device that applies slow changing bias volages to the modulator, thus making sure the modulator works at a desired set point so the signal inputs to the modulator are translated to proper optical modulation.
- **Symbol generator** – generates data symbols in the baseband and is connected to a DSP block that is in turn connected to the input of the modulator. In the case of our CV-QKD system, the symbols are generated in field-programmable gate array (FPGA) chips using random numbers generated by the QRNG.

Bob's receiver consists of devices that are used to extract transmitted symbols from the optical channel. Bob's system consists of:

- **Local oscillator laser** – used as a phase reference to the input optical signal.
- **Homodyne detector** – used to optically mix the local oscillator and the input signal in order to 'filter out' the optical bandwidth of interest. It uses photodiodes and transimpedance amplifiers to convert optical power to voltage.
- **Signal processing block** – an electrical device that does the electrical and digital post-processing (phase recovery, demodulation etc.) in

order to recover transmitted symbols. CPU, GPU or FPGA-based processing systems are used for this purpose.

3.2.1 Lasers

Laser is a source of light and therefore the key component in optical experiments. For the purpose of our experiments, we consider laser to be the source of coherent states of light. It may also be regarded as a source of a single mode of radiation.

Laser consists of a narrow band optical amplifier which generates a well defined optical beam in both frequency and spatial domains. We use continuous wave (CW) (in contrast to pulse lasers) that produce a steady output beam. Geometrical properties of the beam are matched to spatial modes of the laser cavity.

There are several parameters that are important for characterizing a laser: optical power, spatial mode structure, output wavelength, frequency spectrum and noise spectrum [33]. Spatial properties of the beam are determined by the optical resonator. Generally, a single spatial mode that also the simplest one $\mathrm{TEM}_{0,0}$ is of interest. One of the goals of CV-QKD is to use the off-the-shelf telecommunication equipment [24]. Therefore, the laser wavelength would generally be in one of the standard telecom windows such as 1550nm or 1310nm [34] .

3.2.2 Beamsplitters and couplers

One of the most common elements in free-space optical experiments is the *beamsplitter*. *Fiber couplers* are equivalent devices in fiber optics. A beamsplitter[3] can be modelled as semi transparent mirror with reflectivity and transmissivity coefficients r and t. For an input beam with classical complex amplitude E_1 there are two output beams, one reflected E_3, the other transmitted E_4 as it is shown in Fig. 3.3. Generally, beamsplitters have two inputs, as another input with amplitude E_2, propagating from direction shown in the Fig. 3.3, will produce at the output a wave in the same place and propagating in the same direction as E_3 and E_4 [33]. The reflectivity and transmissivity coefficients give the ratio of the amplitudes of the input and output fields (in convenient matrix form):

$$\begin{pmatrix} E_3 \\ E_4 \end{pmatrix} = \begin{pmatrix} r_{31} & t_{32} \\ t_{41} & r_{42} \end{pmatrix} \begin{pmatrix} E_1 \\ E_2 \end{pmatrix}, \tag{3.2.1}$$

[3]Even though we do not perform free-space optical experiments, in the context of mathematical analysis, it is common to talk about beamsplitters and not fibre couplers.

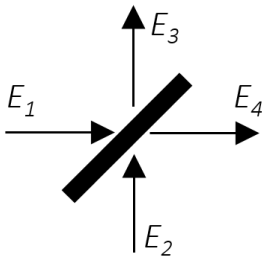


Figure 3.3: Representation of a lossless beamsplitter showing the notation for electric field amplitudes of the input and output beams.

where $r_{31,42}$ and $t_{32,41}$ are complex numbers whose values are determined by using energy conservation law and boundary conditions in the dielectric mirror materials [35]. In our experiments we use the balanced beamsplitter, also called 50/50 beamsplitter, where $|r_{31,32}|^2 = |t_{41,42}|^2 = 1/2$.

In the expression above, we consider the beamsplitters to be ambiguous to input beam polarizations. There are however polarizing beamsplitters (PBSs) that reflect only one polarization and transmit the orthogonal polarization [33]. In general, the coefficients for PBS are not trivial, they depend on the polarization. They are used in *polarization diverse receivers* [45]. Such receivers are expected to be used in future CV-QKD designs as the technology matures.

3.2.3 Modulators

In telecommunications and electronics, modulation is a process of changing one or more properties of a periodic signal (generally a sine wave) called *the carrier signal*. In optical telecommunications the carrier is typically a strong laser light. *Modulator* is a device that performs modulation. As an input, it takes the carrier and *the modulating* signal, which carries information to be transmitted. Most often the modulator changes one of the three parameters of the carrier – amplitude (amplitude modulation -AM), frequency (FM) or phase (PM).

Historically, in classsical optical telecommunications, there was a period when both researchers and industry focused mostly on relatively simple amplitude modulated, or in this case more often called *intensity modulated* (IM) optical systems, with *direct detection* (DD) [36]. Owing to simplicity these systems were cheap and also provided high rates. Direct detection means that the information bit is inferred by directly detecting signal power during the symbol interval. However, by introducing coherent detection schemes, one is able to use more complicated and spectral efficient modulation tech-

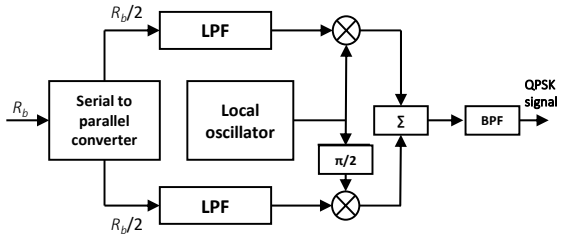


Figure 3.4: Electrical QPSK modulator block diagram. LPF: low-pass filter, BPF: band-pass filter

niques [37].

One such modulation scheme is *quadrature phase shift keying* (QPSK). In this scheme two phase shifted carriers are independently modulated by two independent modulating signals and added at the output. The block diagram is shown in Fig. 3.4. The two carriers are shifted by $\pi/2$, therefore the two modulating signals are the quadratures of the output signal. The output signal can be described with the following expression:

$$\alpha_{\text{out}}(t) = \alpha_0 q(t)\cos(\omega t) + \alpha_0 p(t)\sin(\omega t), \tag{3.2.2}$$

where α_0 is the fixed amplitude and $q(t)$ and $p(t)$ are quadrature digital signals, which for QPSK take values $q(t), p(t) \in \{-1, 1\}$. A figure of merit that is often used to characterize the performance of a system with QPSK modulation is the *error vector magnitude* (EVM). The error vector is defined as a vector in phase space (quadrature IQ plane) between the ideal constellation point and the point received by receiver. EVM is defined as a percentage

$$\text{EVM}(\%) = \sqrt{\frac{P_{\text{error}}}{P_{\text{reference}}}} \times 100\%, \tag{3.2.3}$$

where P_{error} is the power of the error vector and $P_{\text{reference}}$ is the power of the reference signal. Fig. 3.5 shows QPSK constellation in the phase space. It is important to note that QPSK modulator is the general quadrature modulator, since by changing the modulating signals q and p one is able to achieve different quadrature modulation schemes including the Gaussian one used in QKD setup in this thesis. It is common that the modulators are called *the IQ modulators* since the two quadratures are denoted as I – ' the in-phase component', and Q – 'the quadrature component'. We use this notation in the following chapters.

Optical IQ modulator

Modulators in optics are realized using crystals with electro-optical characteristics where the refractive index can be manipulated by applying external electric field [38]. This property leads directly to optical phase modulation.

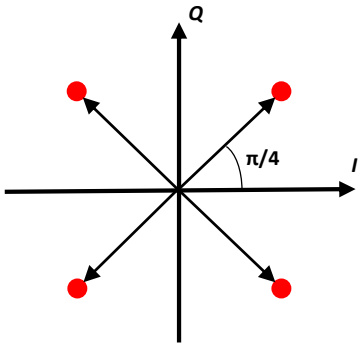


Figure 3.5: QPSK constellation in phase space. Here we introduce a notation that is common in telecommunication where the quadratures are denoted I and Q instead of Q and P respectively.

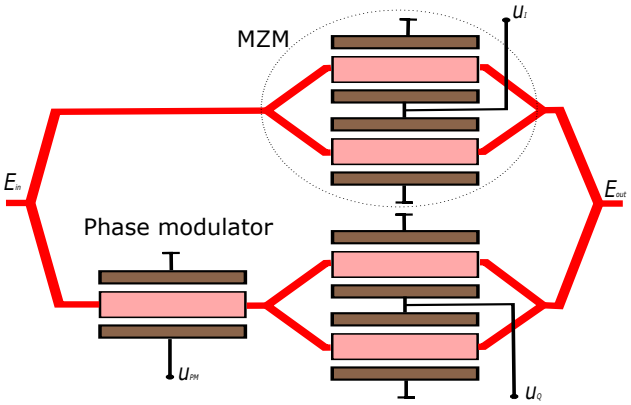


Figure 3.6: IQ modulator

By putting two phase modulators in parallel while using two optical couplers for splitting the input beam and combining the two beams at the output, one gets the Mach-Zehnder modulator (MZM) (encircled in Fig. 3.6). MZM can be operated in two modes: push-push where MZM behaves as a pure phase modulator, and push-pull where MZM behaves as a pure amplitude modulator. The quadrature point is a voltage configuration of MZM where the transfer characteristics of the modulator can be considered linear with respect to the input bias voltages. By putting together two MZMs and a phase modulator one gets the optical IQ modulator shown in Fig. 3.6. The phase modulator shifts the phase by $\pi/2$, therefore creating two orthogonal carriers in the two arms of the interferometer. Two MZMs act as amplitude modulators. $u_I(t)$ and $u_Q(t)$ are bias voltages which are also input modulating signals for the two quadratures. The transfer function

of the modulator with linear approximation around the quadrature point is:

$$\frac{E_{\text{out}}(t)}{E_{\text{in}}(t)} = \frac{\pi}{4V_\pi}u_I(t) + i\frac{\pi}{4V_\pi}u_Q(t) \tag{3.2.4}$$

A feature of IQ modulators is their ability to create *single-sideband modulation* (SSB). SSB modulation was used for a long time in radio frequency, microwave and optical telecommunication. Its primary purpose is to reduce the bandwidth of transmitted signal by a factor of two, therefore reducing noise and increasing signal-to-noise (SNR) ratio and bandwidth efficiency. Furthermore, SSB modulation uses less power to transmit the same amount of information compared to a two-sided modulation. The reduction of power can be achieved also by manipulating the carrier, so there is a difference between SSB with full carrier and SSB with *suppressed carrier* (SSB-SC). Sometimes it is favorable that the carrier is present for easier carrier recovery and synchronization at the receiver. Single sideband modulation is graphically depicted in Fig. 3.8.

Given a real signal $E(t)$ with double-sideband (DSB) spectrum, the SSB version of the signal is obtained by the following operation:

$$E_{\text{SSB}}(t) = E(t) \pm i\hat{E}(t) \tag{3.2.5}$$

where plus sign gives the *lower sideband* (LSB), while minus sign gives the *upper sideband* (USB). By $\hat{E}(t)$ is denoted the Hilbert transform of the signal. A Hilbert transformation is defined in frequency domain as a fixed phase shift depending on the frequency sign. The modulo of the amplitude characteristics of the Hilbert transform is unity for all frequencies $|H(\omega)| = 1$, and the phase characteristics is $\theta_H(\omega) = \text{sgn}(\omega)\frac{\pi}{2}$ as shown in Fig. 3.7[39]. Using the previous analysis and the properties of Fourier transform of relevant signals, one can obtain the formula for SSB signal modulated onto a carrier of frequency ω_c:

$$E_{\text{SSB}-\text{mod}}(t) = E(t)\cos(\omega_c t) \pm \hat{E}(t)\sin(\omega_c t) \tag{3.2.6}$$

It is clear from Eq. s 3.2.4 and 3.2.6 that by applying a signal to one of the inputs of the optical IQ modulator, and its Hilbert transform to another input, it is possible to achieve single-sideband modulation in optical domain.

Optical IQ modulation is a mature technology [40] and is especially convenient when implemented in integrated optics [41–43], therefore enabling miniaturization and reduction in component cost.

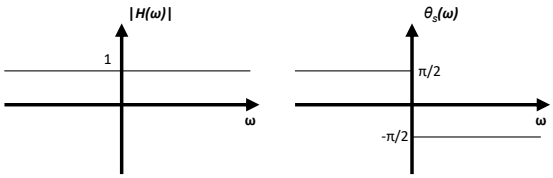


Figure 3.7: Hilbert transform amplitude and phase characteristics

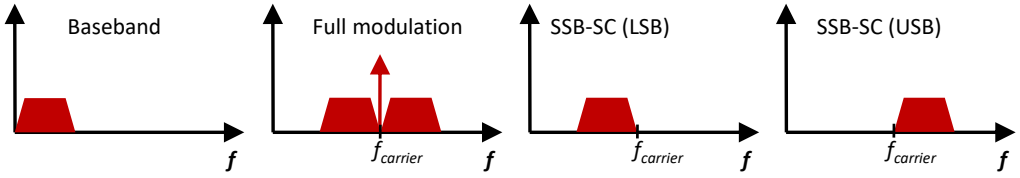


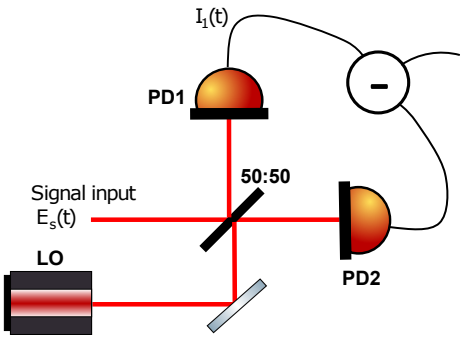
Figure 3.8: Comparison between spectra of a baseband signal and their modulated variants

3.2.4 Automatic bias controller

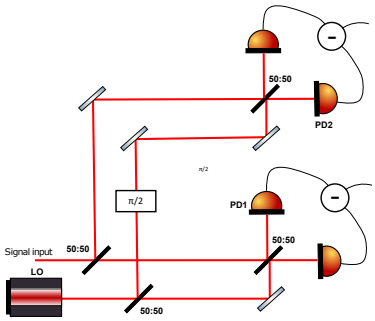
Controlling bias voltages in an IQ modulator is not a trivial task. In a regular IQ modulator there are three voltages to be kept under constant control in order to achieve optimal performance. The situation is complicated even more if *dual polarization* (DP-MZM) modulator is employed, where one needs to control six bias voltages. DP-MZM are two IQ modulators put in parallel using a polarizing beamsplitter in order to work with both polarization degrees of freedom. *Automatic bias controllers* (ABC) are used for fine tuning the bias voltages. The optical input provides the necessary feedback in order to maintain the DC bias voltage at the desired spot. This kind of ABC generally works well with any litium niobate (LiNbO3) optical modulator and at any data rates.

3.2.5 Homodyne detection

It was mentioned in Section 3.2.3, there was a period when both research and industry (in classical telecommunications) were focused on relatively simple and cheap direct detection schemes for optical communications. Physicists on the other hand, were interested in phase-sensitive measurements [46–49], as a complete description of a quantum state of light requires its associated phase information. For the purpose of having a phase-sensitive measurement, a *coherent receiver* configuration is employed. Coherent receivers use a reference coherent beam called the *local oscillator* (LO) in order to measure the complex amplitude of the receiving signal, as shown in Fig. 3.9. If the coupler used in the receiver is 50/50, the receiver is called the *balanced*



(a) Homodyne receiver



(b) Phase-diverse homodyne receiver

Figure 3.9: Optical coherent receivers

coherent receiver. Two output beams of the coupler are detected using a pair of photodiodes. In our work, this kind of device is called the *balanced homodyne receiver*, or just the *homodyne reciver*. Note that with the homodyne receiver, it is possible to perform a *homodyne, heterodyne* or a *phase-diverse homodyne* measurement. The difference between these terms is explained in the following paragraphs.

The derivation presented below can be found in [36]. Let $E_s(t)$ be the input signal electric field (Fig. 3.9):

$$E_s(t) = \alpha_s(t)e^{i\omega_s t}, \tag{3.2.7}$$

where $\alpha_s(t)$ is the complex amplitude and ω_s is the frequency of the signal laser. Similarly, $E_{\mathrm{LO}}(t)$ is the local oscillator field with α_{LO} being the constant complex amplitude

$$E_{\mathrm{LO}}(t) = \alpha_{\mathrm{LO}}e^{i\omega_{\mathrm{LO}}t}. \tag{3.2.8}$$

Using the complex amplitudes, signal power of the both fields is calculated as

$$W_s = k|\alpha_s|^2/2, \tag{3.2.9}$$

$$W_{\mathrm{LO}} = k|\alpha_{\mathrm{LO}}|^2/2, \tag{3.2.10}$$

where $k = S_{\mathrm{eff}}/Z_0$, and S_{eff} is the effective beam area and Z_0 is the impedance of the free space.

The signal field and the LO filed are coupled at the balanced coupler (beamsplitter). Balanced detection is used in coherent receivers as a tool to suppress the DC component and maximize the beat between the signal and the

LO [36]. Using the beamsplitter transformation covered in Section 3.2.2, the output fields are calculated as:

$$E_1 = \frac{1}{\sqrt{2}}(E_s + E_{\mathrm{LO}}), \tag{3.2.11}$$

$$E_2 = \frac{1}{\sqrt{2}}(E_s - E_{\mathrm{LO}}). \tag{3.2.12}$$

Now the photocurrents on the two photodiodes are calculated as

$$\begin{aligned} I_1(t) &= k\frac{q_e\eta}{\hbar\omega_s}\left\langle \mathrm{Re}\left(\frac{\alpha_s(t)e^{i\omega_s t} + \alpha_{\mathrm{LO}}e^{i\omega_{\mathrm{LO}}t}}{\sqrt{2}}\right)^2\right\rangle \\ &= \frac{q_e\eta}{2\hbar\omega_s}[W_s(t) + W_{\mathrm{LO}} + 2\sqrt{W_s(t)W_{\mathrm{LO}}}\cos(\omega_{\mathrm{IF}}t + \theta_s(t) - \theta_{\mathrm{LO}}(t))], \end{aligned} \tag{3.2.13}$$

$$\begin{aligned} I_2(t) &= k\frac{q_e\eta}{\hbar\omega_s}\left\langle \mathrm{Re}\left(\frac{\alpha_s(t)e^{i\omega_s t} - \alpha_{\mathrm{LO}}e^{i\omega_{\mathrm{LO}}t}}{\sqrt{2}}\right)^2\right\rangle \\ &= \frac{q_e\eta}{2\hbar\omega_s}[W_s(t) + W_{\mathrm{LO}} - 2\sqrt{W_s(t)W_{\mathrm{LO}}}\cos(\omega_{IF}t + \theta_s(t) - \theta_{\mathrm{LO}}(t))], \end{aligned} \tag{3.2.14}$$

where ω_{IF} is the difference between the signal and the LO frequencies $\omega_{IF} = |\omega_s - \omega_{\mathrm{LO}}|$, and $\theta_s(t)$ and $\theta_{\mathrm{LO}}(t)$ are the phases of the signal and the LO respectively. η is the quantum efficiency of the photodiode and q_e is the electron charge.

The output of the homodyne detector is the difference between the two currents:

$$I(t) = I_1(t) - I_2(t) = 2\frac{q_e\eta}{\hbar\omega_s}\sqrt{W_s(t)W_{\mathrm{LO}}}\cos(\omega_{IF}t + \theta_s(t) - \theta_{\mathrm{LO}}(t)). \tag{3.2.15}$$

Eq. 3.2.15 shows that the output signal is proportional to the square root of the LO power. This means that by increasing the LO power, the amplitude of the received signal increases, and even for low input signals P_s there is a possibility of bringing the output signal level above the electronic noise of the detector. This is very important for measuring low level signals and intrinsic quantum noise, which is one of the central concepts in quantum optics. The detector which can bring the intrinsic quantum noise signal above the electronic noise is called the *shot noise limited* detector.

After getting the general expression for output current of the homodyne receiver, one can define different types of measurements. If the frequency of the LO is not the same as the frequency of the signal $|\omega_{IF}| > 0$, and

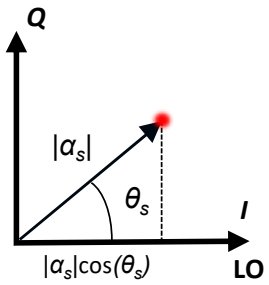


Figure 3.10: The in-phase component measured with the homodyne measurement

specifically if $|\omega_{IF}| \gg B_s$, where B_s is the measured signal bandwidth, this is a *heterodyne* measurement. In the literature, the case where $|\omega_{IF}| \approx B_s$ is called *intradyne* scheme [50]. On the other hand, *homodyne* measurement is the one where there is the exact match between the frequencies $|\omega_{IF}| = 0$. In homodyne measurement the output current becomes

$$I(t) = 2\frac{q_e\eta}{\hbar\omega_s}\sqrt{W_s(t)W_{\mathrm{LO}}}\cos(\theta_s(t) - \theta_{\mathrm{LO}}(t)). \tag{3.2.16}$$

If for now $\theta_{\mathrm{LO}}(t)$ is fixed to be constant and zero, since the time variation just represents the phase noise of the LO (LO phase is precisely controlled as it represents the reference), a simple expression is obtained that tells that the result of the homodyne measurement is nothing else but the in-phase component of the optical signal as shown in Fig. 3.10.

Homodyne measurement is not able to retrieve the full information about the complex amplitude. It is however possible to choose whether, during the symbol period, one wants to measure the in-phase or the quadrature component. This is done by fixing the LO phase for the in-phase measurement, and introducing LO phase shift by $\pi/2$ when measuring the quadrature component. On the other hand, the heterodyne measurement is able to retrieve information about both quadratures:

$$\begin{aligned} I(t) &= 2\frac{q_e\eta}{\hbar\omega_s}\sqrt{W_s(t)W_{\mathrm{LO}}}\cos(\omega_{IF}t + \theta_s(t)) \\ &= 2\frac{q_e\eta}{\hbar\omega_s}\sqrt{W_s(t)W_{\mathrm{LO}}}(\cos(\omega_{IF}t)\cos(\theta_s(t)) - \sin(\omega_{IF}t)\sin(\theta_s(t))) \\ &\overset{\text{down+LPF}}{\longrightarrow} \frac{q_e\eta}{\hbar\omega_s}\sqrt{W_s(t)W_{\mathrm{LO}}}\cos(\theta_s(t)) \quad \text{or} \quad \frac{q_e\eta}{\hbar\omega_s}\sqrt{W_s(t)W_{\mathrm{LO}}}\sin(\theta_s(t)). \end{aligned} \tag{3.2.17}$$

There is a third possibility when it is convenient to immediately get the baseband signal using a homodyne measurement, but also retreive the

information about both quadratures. This is called the *phase-diversity measurement* and it employs two homodyne detectors in parallel, couplers for the signal and the LO and a $\pi/2$ phase shifter for the LO, as shown in Fig. 3.9b. The expression for the output currents is identical to the output of the heterodyne measurement, with a factor of 1/2, due to the additional 3 dB splitting.

In classical telecommunications, historically there were different reasons for using homodyning or heterodyning, such as the difficulty of having high intermediate frequency (IF) or optical phase locked loop (OPLL) [37]. In quantum communications there is another parameter of interest when deciding about the measurement scheme – the shot noise.

Measurement of quantum states and shot noise

The intrinsic quantum uncertainty between the two quadratures of a light field is manifested in the inherent noise that is present in the absence of all other extrinsic noise. Homodyne measurement in a quantum setting is described in [51]. The setup is the same one as before shown in Fig. 3.9.

Fields are described now using field operators, $\hat{E}_s(t)$ and $\hat{E}_{\mathrm{LO}}(t)$ for the signal and the local oscillator respectively. The local oscillator is in a coherent state with a large power. It can be denoted by a ket vector $|\sqrt{W_{\mathrm{LO}}/\hbar\omega_{\mathrm{LO}}}e^{-i\omega_{\mathrm{LO}}t}\rangle$. LO power is much larger than the signal power $W_{\mathrm{LO}} \gg W_s$. Considering the homodyne measurement, where the signal carrier frequency and the LO frequency are equal, the detector output current operator is calculated as:

$$\hat{I}_{\mathrm{hom}}(t) = 2q_e g\sqrt{\frac{\eta W_{\mathrm{LO}}}{\hbar\omega_{\mathrm{LO}}}}\int d\tau\,\mathrm{Re}\left(\hat{E}_s(\tau)\phi_{\mathrm{LO}}e^{i\omega_{\mathrm{LO}}\tau}\right)h_{\mathrm{LP}}(t-\tau), \qquad (3.2.18)$$

where q_e is the electron charge, g is preamplifier gain, $\phi_{\mathrm{LO}} = e^{i\theta_{\mathrm{LO}}}$ is the phase of the local oscillator and h_{LP} is the impulse response of a low-pass filter. Previously shown, the homodyne detection measures the signal component that is in-phase with the LO phase θ_{LO}.

For the heterodyne measurement, the same setup is used, however now the LO frequency differs from the signal carrier by ω_{IF}. The state of the LO is $|\sqrt{W_{\mathrm{LO}}/\hbar\omega_{\mathrm{LO}}}e^{-i(\omega_s-\omega_{IF})t}\rangle$. Here, one needs to take care of the image band of the signal mirrored by the LO. The image band lies around $\omega_s - 2\omega_{IF}$ and the modes of this band are not active – they are in the vacuum state. Field operators associated with the signal and image band are $\hat{E}_s(t)$ and $\hat{E}_{\mathrm{image}}(t)$ respectfully. In semiclassical theory, the image band does not produce any current. This is not true in quantum theory for which the output current

operator can now be written as

$$\hat{I}_{\text{het}}(t) = 2q_e g\sqrt{\frac{\eta W_{\text{LO}}}{\hbar\omega_{\text{LO}}}}\,\text{Re}\left((\hat{E}_s(t) + \hat{E}_{\text{image}}(t))\phi_{\text{LO}}e^{i(\omega_s-\omega_{IF})t}\right). \quad (3.2.19)$$

Here the band-pass filter impulse response is omitted for simplicity.

Eq. 3.2.18 shows that the noise obtained by measuring the output current does not come from the local oscillator, but from the measurement of the field operator $\hat{E}_s$. The signal can be in the vacuum state. Since this is also a coherent state, the homodyne measurement output will be the shot noise attributed to quantum uncertainty. The power spectral density of the shot noise is

$$\mathcal{S}_{\text{shot}}(f) = \frac{g^2\eta q_e^2 W_{\text{LO}}}{\hbar\omega_{\text{LO}}}|H_{LP}(f)|^2, \quad (3.2.20)$$

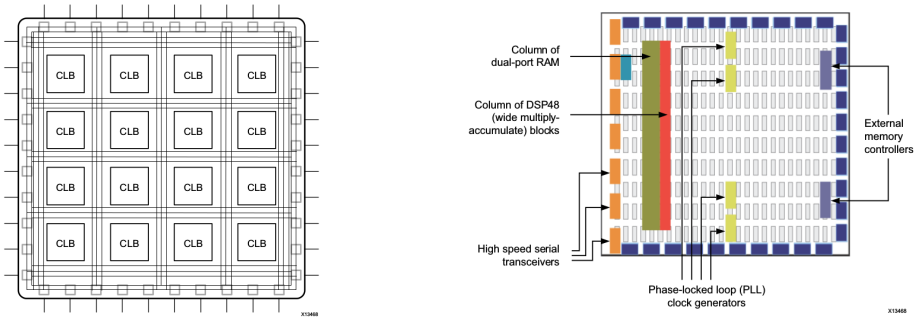
where $H_{LP}(f)$ is the frequency response of a low-pass filter that models the response of imperfect system. In CV-QKD and QRNG it is quite convenient take the shot noise level as a reference, as it is intrinsic and stays constant for constant LO power, quantum efficiency and detector gain. When this is the case, the power of the shot noise is called the *shot noise unit* (SNU).

3.2.6 FPGA chips

This section provides an introduction to *field programmable gate array* (FPGA) chips and the motivations for their usage in quantum communications. It gives an overview of FPGA architecture, basic hardware primitives, and some designing techniques with FPGAs. There is a large set of textbooks and other literature about FPGAs and hardware design [52–55]. Note it is implicitly assumed Xilinx-made FPGAs are used, in particular Kintex UltraScale, that is used in this thesis. Appendix A provides detailed description of intellectual property (IP) cores used for our designs and other FPGA board related topics.

General FPGA architecture

There has been an ever increasing need in technological applications for more processing power and parallelization. Engineers can chose one of the several options when implementing a functionality or an algorithm. One option are single core microprocessors that can be very flexible and cheap, but may be too slow for some purposes. Another option are multi-core systems or graphical processing units (GPU) with thousands of cores, still very flexible, but more expensive and harder to program. On the other end of this spectrum are application-specific integrated circuits (ASIC), specialized pieces of hardware built only for one purpose. They are very fast, but expensive and not flexible. Somewhere in the middle of the processing device



(a) Basic FPGA architecture (b) Modern FPGA architecture

Figure 3.11: Taken from [54](Xilinx)

spectrum lie FPGAs. It is a chip technology that addresses the cost and flexibility problem by implementing a large set of multipurpose and programmable primitives, but also keeping the high performance regarding the speed and latency compared to CPUs and GPUs. These facts makes FPGAs an optimal solution for a plethora of applications, especially in research and development.

FPGAs consist of hardware primitives. Bellow are listed hardware primitives of modern Xilinx chips. Not every design utilizes all of them, however they all have been an integral part of the firmware designs in this thesis.

- Every FPGA (even the simplest and cheapest ones) is composed of ***configurable logic blocks*** **(CLB)** and interconnections between them. There are also external inputs and outputs which are represented as wires on the edge of the chip in Fig. 3.11a [54]. CLBs contain the programmable logic for the FPGA and this basic logic was present in FPGAs from the birth of the technology. One of the primitives found in CLBs is four-input *lookup table* (LUT). LUT can implement any four-input Boolean function and is equivalent to a system of logic gates. It is basically a truth table where different combinations of the inputs yield different output values for different functions. The number of memory locations for N-input LUT is 2^N and the total number of functions which can be implemented is 2^{N^N}. CLBs also contain *Multiplexers* (MUXs) that are statically programmed. That means the data paths are defined upon FPGA programming. Furthermore, flip-flops are the basic storage units within the FPGA fabric. Flip-flops are always paired with LUTs so they assist in pipelining and data storage. Flip-flops incorporate data and clock inputs, clock enable input, reset and data output. The value from the data input is passed to the data output on every clock pulse. Clock enable pin is used for storing a value for more than one clock cycle.

- Modern FPGAs contain, besides typical CLBs, several other specialized primitives that can sometimes drastically increase performance and platform flexibility. Fig. 3.11b shows the typical architecture of modern FPGAs. Standard CLBs are shown as grey rectangular boxes. For enabling high-rate multi-gigabit per second serial communications, modern FPGAs employ high speed **serializer/deserializer (serdes)** circuitries in high speed transceivers. The examples are GTX and GTH transceivers on Xilinx FPGAs.

- **Block RAM** (BRAM) is a dual-port RAM module instantiated into the FPGA fabric to provide on-chip storage for a relatively large set of data. Xilinx devices use two types of BRAM with different capacities, 18kb and 36kb. These elements can be grouped to virtually appear as a *read-only memory* (ROM) or a shift registers. They are also extensively used for implementing *first-in-first-out* (FIFO) buffers.

- **DSP algorithms** are vital to modern telecommunications. Implementing computationally intense algorithms, such as *fast Fourier transform* (FFT), require a large number of arithmetic operations. Doing that using only LUTs and flip-flops from basic CLBs might be very complicated and potentially resource-consuming. For the purpose of tackling this problem, mid-to-high-end FPGAs implement a huge number of specialized DSP primitives. Structure of a DSP block is shown

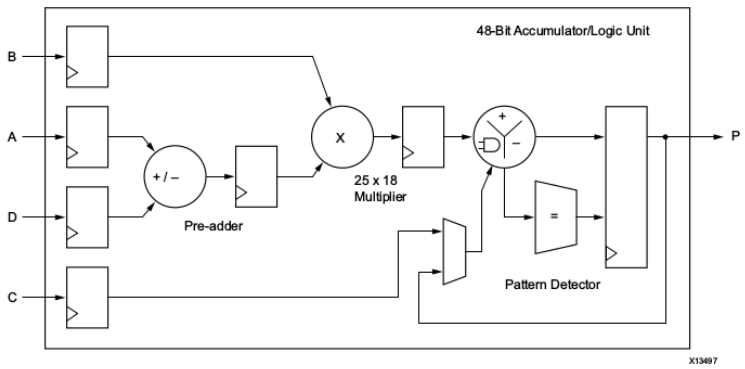


Figure 3.12: Architecture of a DSP block. Taken from [54](Xilinx).

in Fig. 3.12. A single block implements add and subtract units, followed by a multiplier, followed by another set of add/subtract units with accumulative capability. Multiplication is an example of an operation where the basic FPGA architecture cannot deliver optimal performance, so dedicated DSP blocks are essential. Our Kintex UltraScale chip is specifically well-suited for DSP operations with the large number of these primitives compared to other models.

- The essential part of any processing unit is their **clocking infrastructure**. FPGAs saw drastic increase in size, complexity and speed during the last decade. Therefore they require high quality clocking resources. Even in small FPGA designs, clock signals might be used by hundreds or even thousands of elements. The number of elements that use a clock signal is called the *fanout* of that particular clock. In

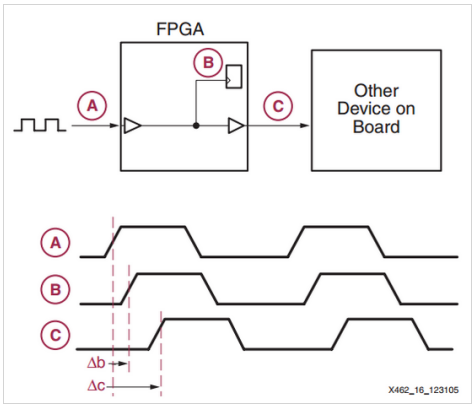


Figure 3.13: An example of clock skew. Taken from [56](Xilinx).

Fig. 3.13, a clock is distributed from point A to points B and C. The clock enters the chip at point A and is passed through a clock buffer (depicted as a triangle). Output of the buffer is connected to high performance specialized clock lines. This clock is then distributed to all regions (if it is a global buffer) in the chip, in this example to points B and C. In general, there will be some phase delay between the original clock and the one that arrives at B and C. This delay is called the *clock skew*. It is always desirable to have as low skew as possible, however not every skew will cause a timing violation. There are two types of timing violations: *hold violation* and *setup violation*. If the clock travels slower than the data from one register to another, there is a danger that the data is not held long enough at the destination flip-flop to be properly clocked through. This is a hold violation. Setup violation occurs when the destination flip-flop receives the clock earlier than the source flip-flop because the new data was not set up and stable before the next clock tick arrived. Hold violations are more severe than setup violations as they cannot be fixed by increasing the clock period. FPGAs employ circuits to fight the excess skew and they are called de-skew circuits. There are two de-skew mechanism, delay line-based clock de-skew (DLL-based) and phase locked loop-based (PLL-based) [58].

Another important parameter is the *clock jitter*. Jitter is the deviation of the clock frequency. It can be expressed as a phase noise, or in

time domain in ps (since the frequencies of modern FPGAs are in the range of hundreds of MHz). It can occur both on the leading edge and the trailing edge of the signal. Jitter can also be frequency dependant. Excessive jitter can cause violation of timing margins. Furthermore, high-rate communication systems and analog-to-digital (ADC) converters are sensitive to jitter where it can increase the bit error rate (BER). For this reason FPGAs require a good clock source. In many applications the source clock is supplied outside FPGA by a high-quality clock generator (such as Silicon Labs Si5338 [161] that we are using in our hardware), so the clock can be used even for sensitive serial transceivers.

In Xilinx FPGAs, de-skew and jitter cleaning functions are implemented in cores such as **Mixed-Mode Clock Manager (MMCM)**. MMCM is a multifunctional core that can also generate clocks of different frequencies than the frequency of the input clock. It can be also used as a PLL. Often times the specialized cores are not necessary since the clock signals are distributed throughout the chip using low skew clock networks. First of all, there are global clock networks, accessed by the *global buffers* (BUFG). Clocks generated like this are used in most cases and they are available in the whole chip. In addition there are regional clock networks, accessed by the *regional buffers* (BUFR). The clocks generated like this are available only in designated clock regions and are sometimes used for some smaller time critical designs.

Designing with FPGA

A *hardware description language* (HDL) is a necessary ingredient for describing complex digital logic circuits. It enables engineers to describe the intended design in a precise and formal way, which further allows the circuits to be simulated. A (high-level) HDL is *synthesized* to produce a *netlist*, which is a specification of electric components and a description of how they are connected together. The netlist is passed to the next stage of HDL compiling process called *place and route* where the exact hardware primitives are assigned, according to the netlist, for a specific hardware device. HDLs generally look like software programming languages, however they require a certain paradigm shift in the way of how designer should look at the problem. This mostly comes from the fact that operations are performed in parallel and that the events occur at precicely defined discrete clock cycles. The most well known hardware description languages are *Verilog* and *VHDL*. VHDL is used for designs in this thesis.

Two very common and important programming techniques that are widely

used when designing a firmware[4] are *parallelizing* and *pipelining*. They are used in different scenarios, depending on what the particular goal of a design is. Sometimes the design is optimized for speed. In other words the hardware is placed such that it enables the highest clock frequencies without timing fails. In other cases, the resources on the chip are scarce, so the design is optimized to use the fewest primitives possible. There are also other design strategies like minimizing power usage etc.

- **Parallelizing** – If FPGA has enough resources and the algorithm in question allows it, it is generally a good strategy to parallelize functions as much as possible. Engineer's task is to recognize tasks that can run in parallel. The notion of discrete time is paramount. Compared to software programming, it is much easier to estimate, or know exactly, what is the time (or number of clock cycle) some operation requires when implementing in hardware. A simple example would be a calculation of a scalar product of two vectors. This operation requires a multiplication of the vector elements with addition in the end. In general-purpose processors each multiplication is done one at a time. In FPGA there could be many multiplication modules working in parallel. Furthermore, each module can be designed to have a fixed latency, so the results are easily forwarded to the adder at the same time.

- **Pipelining** – If an algorithm contains operations where there are no feedback loops, and the operations are done one after another, it is possible to use pipelining. Pipelining is a digital design technique where the data processing elements are connected in series using registers and the output of one of the element is the input to the next one. The top part of Fig. 3.14 shows an implementation of a function, in this case for inputs (a, x, b, c) it computes $y = a \times x + b + c$. This design is a *combinatorial* network, where the elements are simple Boolean functions and are not clocked. The wires and the arithmetic functions each have some non zero delay. This means that the voltage at the output might have unstable value for a certain period of time. The solution for the proper readout of y would be to introduce a register at the output. However, if the combinatorial logic introduces high enough latency, there is a danger of setup timing violation described in the previous subsection. The solution is to introduce fully pipelined design shown in the bottom part of Fig. 3.14. Each step now has a fixed latency of one clock cycle. Also the inputs to each of the arithmetic modules do not depend on the outputs in the 'downstream'. This way the module

[4]When talking about FPGAs we use the words hardware and firmware, and quite often they are synonymous. We mostly use the word firmware when talking about a particular HDL written design that is downloaded to an FPGA.

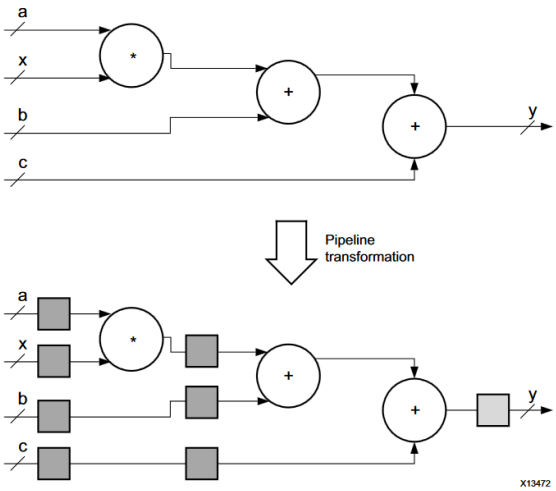


Figure 3.14: Pipelining in a digital desing. Taken from [54](Xilinx).

produces a result every clock cycle, however with the initial latency of (in this case) three clock cycles.

Another important concept in both hardware and software design is the concept of *the finite state machine* (FSM) [59, 60]. In automata theory, FSM lies above combinatorial logic and is more powerful computational tool. FSM is an abstract machine that can be in exactly one of the finite number of states at any given time. The state transition is governed by external input signals. Every transition happens on a logical condition which needs to be fulfilled. Fig. 3.15 shows a simple example of a two-state state machine of a

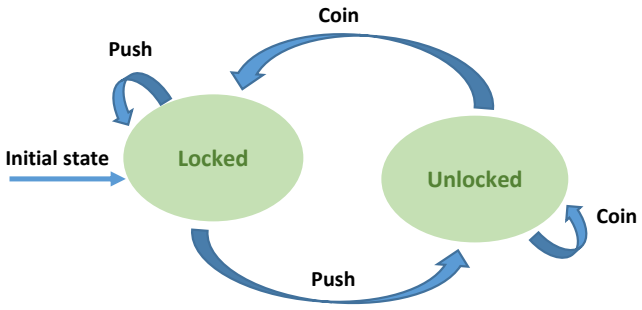


Figure 3.15: An example of state machine of a turnstile

turnstile. The external signals that influence the state transitions are coin-inserted and the turnstile-pushed. Therefore, in combinatorial circuits, the output depends only on the current values of inputs, and in state machines the output depends on the inputs and the current stored information (state).

If the output of a state machine depends on the states only, this is *Moore state machine*. If the output depends on the states and some external input signals, this is called *Mealy state machine*. Both FSM types are used in FPGA development [61]. The state is saved in a register. This registers consists of certain number of bits, depending on the number of states and preferred coding method. There are several coding types, most common being *one-hot, binary* and *gray*. One-hot encoding uses as many bits as there are states, and is used most often by the synthesis tool. It uses maximum number of bits, but is also the easiest to decode, as opposed to binary coding that takes longer to decode.

3.3 CV-QKD protocols

History of continuous variable QKD begins in the early years of 21st century with squeezed state protocols [26, 27], immediately followed by the first coherent state protocol [28]. For all of them, the basic idea is the same, Alice modulates only one or both quadratures of the signal laser and sends to Bob through a fiber-optic link[5]. Bob measures the received signals using homodyne or heterodyne detection[6]. The original protocol with coherent states [28] used Gaussian modulation and homodyne detection which meant the detection of only one quadrature, so Bob needed to perform fast switching between the quadratures and key sifting post-processing procedure like in some discrete variable QKD protocols, most notably the original BB84 [4]. It turned out the switching limits the rate of the protocol, so a no-switching protocol was proposed by Lance et al. [62].

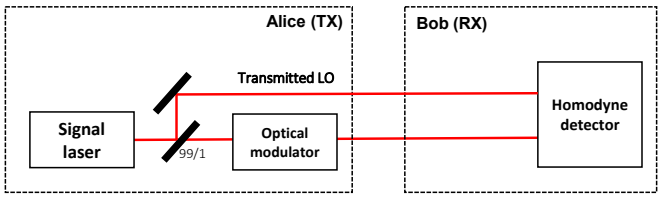
In order to perform a coherent detection of the signal, Bob requires a strong local oscillator. Earlier CV-QKD protocols required Alice to prepare both the quantum signal and the LO [63–66]. This way the LO experiences the same phase drifts in the channel as the quantum signal, leading to lower noise of the coherent detection. Unfortunately, the transmitted LO has several critical drawbacks. Due to losses in the channel, the effective distance of the protocol is reduced, as the LO needs to have sufficient power for the shot-noise limited detection. Insufficient power introduces additional noise. Furthermore, multiplexing the LO can introduce noise in the quantum sig-

[5]There are experimental implementations of free-space CV-QKD protocols, however they are not the topic of this thesis.

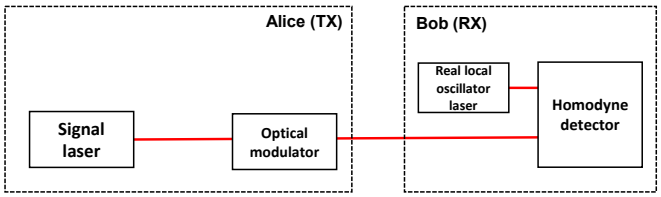
[6]There has been a misunderstanding in the QKD community regarding the definitions of homodyne and heterodyne detection. In classical telecommunications, the heterodyne detection uses the same hardware as the homodyne, however the LO frequency is detuned so the signal of interest is observed at some intermediate frequency. Some groups in QKD community call phase-diverse receiver (simultaneous measurement of both quadratures) as heterodyne receiver. We stick to the traditional definitions as in classical telecommunications.

nal bandwidth. The last, and maybe the most important drawback, is that transmitted LO opens a loophole for attacking the QKD system. These vulnerabilities were shown for example in [67].

A remedy for the LO loophole is the *real local oscillator* (RLO). RLO is located in Bob's box and is fully controlled by him, i.e. it is trusted. Fig. 3.16 shows the difference between the two setups. Experiments using RLO appeared relatively recently [68–73]. For establishing a reliable phase reference in such a configuration, Alice multiplexes pilot tones/pulses in frequency/time that can be detected by Bob with a signal-to-noise ratio significantly higher than that for the quantum signal. Bob typically employs



(a)



(b)

Figure 3.16: (a) CV-QKD with transmitted LO and (b) with real LO

digital signal processing algorithms operating on the pilot signals to compensate for the phase drift between the lasers. This technique has been well known in classical optical telecommunications.

We note that very recently there has been reported a possible attack on CV-QKD systems with the pilot tone [74]. The attack only applies for schemes with trusted phase noise. Obviously one solution could be to treat the phase noise as it originates from Eve (untrusted). The other approach to completely mitigate pilot tone attacks is not to use the pilot, but try to infer the phase at Bob's side using novel techniques for phase recovery by employing low-SNR quantum signal only [75]. However, the proposed technique is in its infancy and only tested with a QPSK modulated signal.

The system implemented in our lab is a pilot-assisted Gaussian modulated

CV-QKD system with the real local oscillator and untrusted phase noise. As such it represents a state-of-the-art research setup with a big potential for practical applications in industry in the following years. To evaluate our system we also use QPSK modulation. Security of QPSK-modulated CV-QKD was proven in asymptotyc regime under collective attacks [76]. The QKD transmitter developed for this thesis evolved along the optical setup and was tested as a part of this system.

3.4 CV-QKD setup

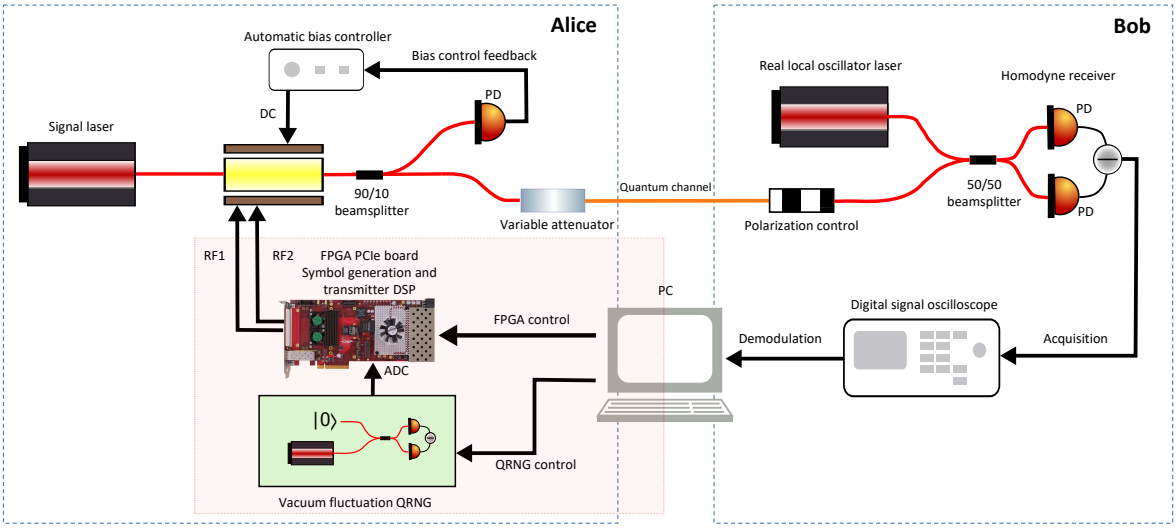


Figure 3.17: Block diagram of our CV-QKD setup. Red lines – polarization maintaining optical fiber. Orange line – single mode optical fiber. Black lines – electrical links. Light pink rectangle – part of the setup this thesis focuses on (QKD transmitter). RF – radio frequency signal. PD – photodiode.

Fig. 3.17 shows a block diagram of the QKD setup implemented in our lab. Standard optical equipment is employed, such as off-the-shelf DFB lasers and IQ modulator. This way the design aims to meet one of the initial requirements for CV-QKD – lower cost compared to single photon-based DV-QKD systems.

The thesis focuses on part of the setup designated with the light pink rectangle. In this context we call this part the *QKD transmitter*[7]. Vacuum

[7]Even though the whole transmitter consists of all the optical devices in Alice's box as well.

fluctuation-based QRNG is used for feeding random numbers to the system. PC821 PCIe-enabled board hosts FMC120 board and Kintex UltraScale FPGA (see Appendix A.3). Homodyne output from the QRNG is sampled with an on-board *analog-to-digital converter* (ADC). The FPGA performs DSP tasks such as randomness extraction, Gaussian sampling, upsampling and upconversion of transmitted symbols. A *digital-to-analog converter* DAC is used to generate two quadrature RF signals that are fed to the IQ modulator. When appropriate DC biases along with the RF waveforms are applied to the IQ modulator, the modulator performs optical carrier suppression and single sideband modulation (SSB-SC). While carrier suppression may not be strictly necessary, optical SSB has a lot of significance for CV-QKD as it prevents potential security vulnerabilities. The security issue can arise if the other sideband is not tracked and freely exploited by the adversary. SSB modulation also ensures better spectral efficiency. A commercial bias controller from ID Photonics is employed.

Key devices, such as the lasers, FPGA, QRNG and the oscilloscope, are computer-controlled via Ethernet or USB interfaces. In order to get the optimal mean photon number according to the security proof, Alice uses a variable attenuator (VATT) to reduce the power of the optical signal. Quantum channel consists of a single mode fiber (SMF). Bob's receiver consists of a manual polarization controller (PC) used to maximize the mixing between the received signal and the RLO. Bob performs a heterodyne detection (by mixing the received signal with his RLO) using home-built wideband balanced receiver. This way the restrictions on the location of the carrier tone are relaxed, unlike in intradyne receiver configurations [50]. The (RLO) is detuned with respect to the signal laser. Fig. 3.18 shows the relevant signals in frequency domain at different stages of the transmission.

3.5 Noise sources in CV-QKD

The eavesdropper's activity can be quantified by the noise Bob detects that is above the fundamental shot noise. This is called the excess noise. More conservative security models attribute all the excess noise to a potential eavesdropping. This however may lead to modest key rates as Eve's power is overestimated in practical sense. It is justified to relax some assumptions and attribute a part of the noise power to trusted noise sources which are not in Eve's control. A proper device calibration can improve key rates and increase usability of a QKD system.

Different noise sources are identified in order to optimize the performance of QKD systems. They are classified and quantified in [25, 77]. Large portion of the total noise comes from the detector, i.e. the electronic noise of the

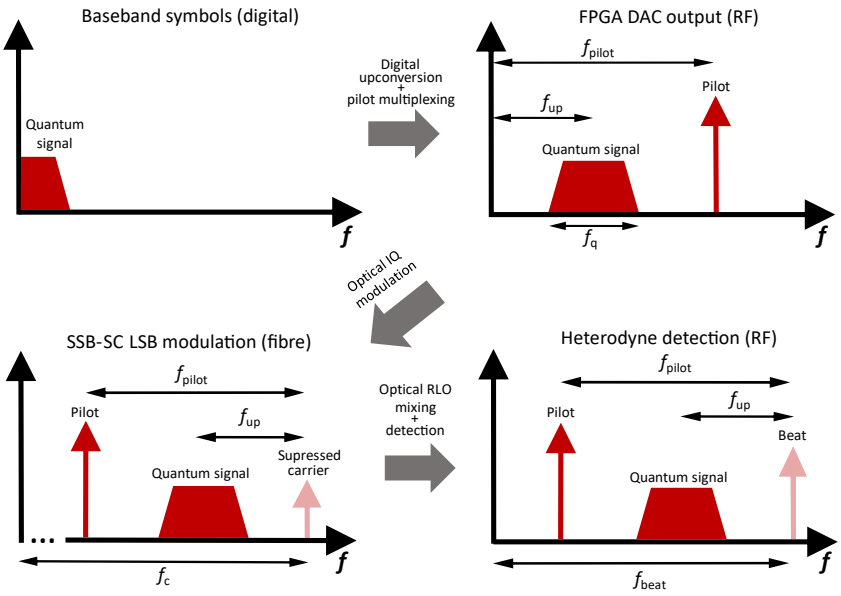


Figure 3.18: The data symbols (quantum signal) are generated at the rate of R_s. They are upsampled and pulse shaped using a RRC filter with roll-off β_{RRC}. The quantum signal is then upconverted to f_{up}. At this point the signal has a bandwidth of $f_q = (1 + \beta_{\text{RRC}})R_s$. A pilot tone is frequency multiplexed at f_{pilot}. The top right figure shows a spectrum of the RF signal at IQ modulator input. The optical carrier (Alice's signal laser) is at frequency f_c. After mixing the optical signal at Bob's side with his RLO, the heterodyne output includes the beat between the signal and RLO lasers at f_{beat}.

homodyne receiver. We mentioned earlier that in a typical device-dependent CV-QKD model, Alice's and Bob's boxes are trusted and are out of Eve's reach. This is a valid assumption as Bob should be able to characterize his device[8]. Bob characterizes the electronic noise in the calibration stage before the quantum communication starts. Some of the other noise sources include:

- **Raman scattering** – This noise source is significant in wavelength multiplexed systems. Currently our system does not have multiplexed classical channels, however this will be important in future research.
- **Phase recovery noise** – Comes from imperfect phase recovery at

[8]Device-independent QKD and QRNG protocols exist [78], where the set of assumptions on Alice's and Bob's devices are minimal or non-existent. However they all rely on large Bell inequality violations and are hard to implement with notoriously low key or randomness rates.

the receiver's side. For our system this means imperfect measurement of the pilot signal which is sent along the quantum signal as a phase reference. We consider this noise source to be untrusted.

- **ADC quantization noise** – A consequence of the finite precision of ADC and non-perfect sampling. We consider this noise source to be untrusted.

- Other noise sources that contribute less in the total excess noise such as relative intensity noise (RIN) of the signal laser and the local oscillator, finite common-mode rejection caused noise that is proportional to the RIN, transmitter induced noise from the IQ modulator and DSP imprecision etc. All the sources are considered to be untrusted.

Chapter 4

Quantum random number generation

Random numbers are ubiquitous in modern society. They have numerous applications ranging from cryptography, modelling and lottery to fundamental science. For most of these applications the quality of random numbers is of utmost importance. The seemingly simple task of producing random numbers is far from trivial. *(Quantum) random number generators* ((Q)RNGs) are devices which produce streams of (high quality) random numbers. The field that tries to answer what 'quality randomness' is, is a fruitful research field, and it even encourages philosophical thought [79]. This chapter focuses on two aspects of randomness generation – randomness extraction and online entropy tests. These functionalities were developed as a part of vacuum fluctuation-based QRNG built in our lab. Security is analyzed for all building blocks of the system. Even though the results of this chapter represent a standalone high-speed QRNG system, in the context of this thesis, the optical setup and the randomness extraction module were optimized to be part of the CV-QKD transmitter with Gaussian modulation. The chapter begins with a brief history and motivation for QRNG research. [80, 81] are good resources where QRNG technology is reviewed and where different implementations are compared.

4.1 The science of randomness generation

In the beginning it is important to quantify randomness as different applications require different set of randomness source properties. There are two important categories of random number sources – sources that produce numbers that 'look' random, or in other words mimic the statistics of a random distribution, and sources of true random numbers generated from some unpredictable physical event. We are mostly interested in the second category, however a brief overview of the first kind is welcome. This way we

have a certain insight and avoid possible dangers of ill-defined randomness (especially for cryptographic applications). A device that produces random-looking numbers from a deterministic algorithm is called *pseudo-random number generator* (PRNG). PRNGs generally use some initial randomness (*the seed*) as the input to the algorithm which further *expands* the randomness. PRNGs are often easily implemented in software and one of the advantages is that they are fast. Some of them are based on number theory using congruence formulas, others are implemented based on *linear feedback shift registers* (LFSR).[1] PRNGs have a periodic output sequence due to the deterministic algorithm. The period is one of the most important properties and the aim is to make it as large as possible. The most widely used pseudorandom number generator is MT19937 with a period of $2^{19937} - 1$ [81]. PRNGs are quite convenient for simulations where there is a need for repeatability. A methods for characterizing PRNG would be applying some of the standardized randomness tests. Randomness tests are used to analyze large chunks of random generator output in order to find possible patterns that would indicate the stream is recognizable and not (looking) random. Two most popular statistical tests are NIST suite [82] and Diehard battery of tests [83].

Though the tests are powerful tools, they are by no means the only valid metric when evaluating true random numbers, the ones that are a result of some intrinsically random physical process. A very simple but very illustrative example why statistical tests are not enough is what is called a *the memory stick attack* [84]. Let there be a RNG device provided by some untrusted third party. The customer can test the output and the data might pass all the tests. However, the manufacturer could just install a high-capacity flash drive with a random-looking sequence. This means that from the point of view of the manufacturer, the sequence is completely predictable, therefore completely broken for cryptographic purposes. This example shows that randomness should be certified with a formal physical model of a device with precisely defined *assumptions* [85].

Generators that generate random numbers from truly random physical processes are called *true random number generators* (TRNG). One can argue whether this is possible at all. In a super-deterministic model where everything, our whole Universe and its history, is predetermined and known by some hypothetical external observer, the generation of true randomness is impossible. Therefore, the first assumption is that the laws of (quantum) physics are true.[2] Generally, a model of a RNG is better if it is based on fewer assumptions. However, sometimes adding more assumptions increases

[1] LFSR-based generators are quite convenient for implementing in FPGA chips too.

[2] At least the interpretations of quantum mechanics where true randomness is possible.

the usability while not compromising the security for a particular application [84]. A formal definition of randomness that goes along the discussion above can be found in [20]. Before defining true randomness, a definition of statistical distance of two distributions is necessary.

Definition 4.1.1. (ε-distance) Let there be two random variables A and B defined over the same domain Ω. Two distributions P_A and P_B are ε-close if their statistical distance is less or equal than ε:

$$\frac{1}{2}\|P_A - P_B\|_1 = \frac{1}{2}\sum_{x\in\Omega}|P_A(x) - P_B(x)| \leq \varepsilon. \tag{4.1.1}$$

Definition 4.1.2. True randomness. Let A be a random variable defined over domain Ω. A is called ε-truly random if it is ε-close to uniform and uncorrelated to all other space time variables which are not in the future light cone of A. Denoting this set by Γ_A, this can be expressed as

$$\frac{1}{2}\|P_{A\Gamma_A} - U_\Omega \times P_{\Gamma_A}\|_1 \leq \varepsilon, \tag{4.1.2}$$

where U_Ω is the uniform distribution

$$U_\Omega = \frac{1}{|\Omega|}, \qquad \forall x \in \Omega. \tag{4.1.3}$$

It is a good question whether TRNGs can be classical at all. There are designs that exploit chaotic physical processes such as meteorological phenomena or thermal noise in electronics. These processes are, however, very hard to precisely model, and may be deterministic in nature (a hypothetical computer might be able to model such systems using laws of classical physics giving sufficient processing capabilities). Even though TRNGs on the first glance might seem more desirable than PRNGs, historically there were several difficulties for wider usage of TRNGs:

1. Limited generation rate. Depending on a physical process, the generators have a limited speed which was often smaller than deterministic PRNGs.[3]

2. Trustworthiness and failure detection. Physical system might be hard to rigorously characterize. Also the real-time detection of anomalies in the system itself is a challenge.

3. Necessity for physically adding a new device.

[3]For example Quantis QRNG by ID Quantique is relying on photon counting technology and produces 4Mb/s random output. On the other hand some software-based PRNGs can easily achieve higher rates. Lehmer's generator uses only two multiplication and one addition instructions on a x64 processor.

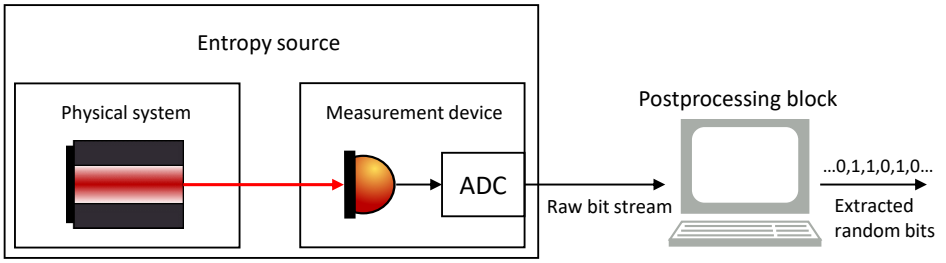


Figure 4.1: General QRNG block diagram

The work done for this thesis tackles the first two challenges.

Our view and analysis of QRNGs only considers cryptographic applications. At the end of 19th century, Kerckhoff put out his famous principle regarding cryptographic systems, still used today: a cryptosystem should remain secure even if everything about the system, except the (random secret) key, is public knowledge [86]. Security of cryptosystems rely on unpredictability of the secret key. In information theoretic framework this means the key should be generated by an ε-random source.

QRNGs can be seen as a set of blocks with well defined tasks. There are three main units, two of which need to have a quantum mechanical description (Fig. 4.1). These two blocks are collectively known as the *entropy source*. The entropy source consists of a *physical system* and the *measurement device*. Measurements are typically analog and need to be converted to digital format with analog-to-digital converters (ADCs). The output bit stream is called *raw bit stream*. These raw bits enter into the *postprocessing block* which is classical. The block removes possible correlations between the raw bit string and the outside world. Post-processing may have different phases, but the most important one is *randomness extraction*. This model is quite general as there is vast number of options to how each of the blocks may be constructed, and it can accommodate different security scenarios and assumptions.

Our first assumption is that the laws of quantum mechanics are true. The first 'place' where we look for inherent randomness is the projective measurement. *Born rule*[4] in its simplest form states that when measuring an observable with discrete eigenspectrum $\{|\lambda_i\rangle\}$, on a system in state $|\psi\rangle$, will yield a result λ_i with probability $|\langle\lambda_i|\psi\rangle|^2$. To illustrate better, we start by the simplest two dimensional quantum system – a qubit, with computational basis $Z = \{|0\rangle, |1\rangle\}$ (Fig. 4.2). If the qubit is prepared (in the 'physical sys-

[4]Formulated in 1926 by German physicist Max Born.

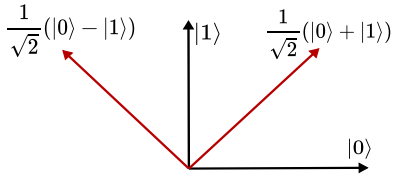


Figure 4.2: Two dimensional Hilbert space of a qubit with Z and X bases

tem' block of QRNG) in one of the eigenstates of the $X = \{|+\rangle, |-\rangle\}$ basis, where $|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$, then when measuring this state in the Z basis (using the second, mesurement block) one will get one of the two results with equal probability. Even though this is a very simple idea, there are many practical and theoretical challenges, such as low rate or non-perfect preparation and measurement. It should be noted that in order for such a scheme to be secure, the preparation and measurement devices cannot be controlled by an adversary. In other words the devices need to be trusted. Such QRNG schemes are called *device-dependent* QRNGs (DD-QRNG). On the other hand, there are QRNG designs where neither preparation nor measurement has to be trusted [84]. They are called *device-independent* QRNGs (DI-QRNGs)[87, 88]. They exploit non-locality, a purely quantum effect. For such protocols, the user needs two separate measurement devices (this is in contrast to the framework put forward above) whose task is to show the violation of Bell inequality. In practice DI-QRNG has very low output bit rate and technical difficulties regarding loophole free Bell measurements. The third kind of generators that are in between DI-QRNGs and DD-QRNGs are called *semi-device-independent generators* (SDI-QRNGs). Compared to DI implementations, they introduce more assumptions and try to increase rates without compromising too much on security. Two main categories here are source-independent [89, 90], where the source is untrusted, and measurement-device independent [91, 92], where the measurement device is untrusted.

Even though DI and SDI QRNGs promise unprecedented security, device-dependent schemes in practise might be very secure if modelled properly, promising high rates along the way. DD-QRNG research has been fruitful in the past decade [80, 81]. Bellow are listed four different technologies.

- Measuring the temporal mode using single photon detectors (SPD). Laser power can be tuned so on average one photon arrives at a detector in some predetermined time period. Then, the time of arrival is random and is consequence of a quantum processes inside the laser. The measurement frequency is limited by the detector dead time, typically 100ns. The raw bitrate for this system is in the order of magnitude of hundreds of Mb/s.

- Measuring the spatial mode of a photon using an array of SPDs. This technique offers similar rates as the temporal mode-based QRNG.
- Measuring the number of photons in a coherent state using photon number-resolving SPD. Coherent states are superposition in photon number basis as stated by Eq. 2.1.29. Thus, by measuring the photon number one can get Poissonian distribution.
- Measuring phase or intensity noise of amplified spontaneous emission (ASE) which is quantum mechanical in nature. For the phase noise QRNG, there is a coherent signal state with phase oscillating around some mean value. If this mean value is for example, 0 or $\pi/2$, the phase noise can be probed by measuring one of the two quadratures of the signal. Some of the experiments with the phase noise achieved multi-gigabit per second key rates [99].

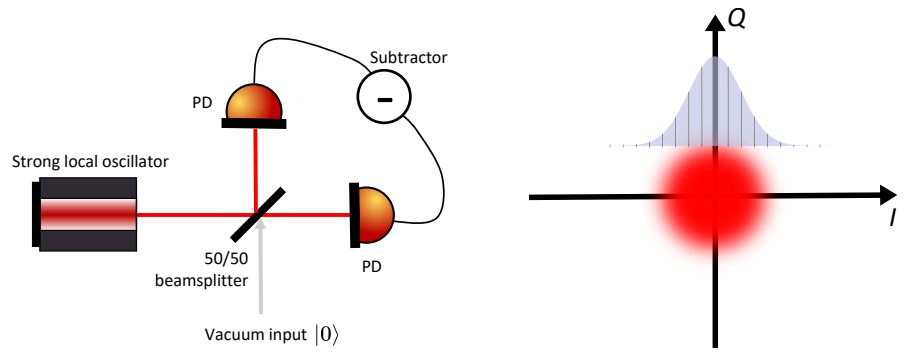
Photon counting-based QRNGs (first three in the list) inherently have possess limitations on speed. Phase noise based QRNGs on the other hand have no such limitations, however it is challenging to build a rigorous model in terms of describing the preparation and measurement of the quantum state. Vacuum fluctuation-based QRNGs aim to tackle all the forementioned challenges.

4.2 Vacuum fluctuation-based QRNG

Probing the vacuum state (shot noise) of the electromagnetic field (see Chapter 2) can be done using a balanced homodyne receiver (Fig. 4.3a). Strong local oscillator is split in a 50/50 beamsplitter. Vacuum mode 'enters' through another input port of the beamsplitter. The two outputs are measured using two photodiodes.

There are several advantages of vacuum noise-based QRNG:

- The resource of quantum randomness – vacuum state, is easily prepared
- Compared to some SPD-based QRNG designs, this one is insensitive to detector loss. The loss can be compensated by increasing the local oscillator power
- The field quadratures are continuous variables, therefore one can obtain more than one bit per measurement, which increases speed
- The speed can be increased by increasing the detector bandwidth. Building a shot noise-limited high bandwidth detector is a challenge for itself [93, 94], but the rates are still much higher than with single photon-based QRNGs.



(a) (b)

Figure 4.3: (a) A simple homodyne measurement of vacuum fluctuations. BS: beamsplitter, PD: photodiode. (b) Red circle is the Wigner function of the vacuum state in phase space. Homodyne output is a Gaussian distributed discrete signal.

The idea of vacuum noise-based QRNG has been previously investigated [95–98]. In the experiment implemented in our lab, however, this idea was extended by taking 'paranoid' approach to min-entropy calculation and developing the technology to support such theoretical framework.

The goal of this analysis can be summed up to a single task – how to calculate the min-entropy of a randomness source. The min-entropy is conditional, which means that we are trying to find out how random is our output bit sequence conditioned on any information that outside world has about the system. Traditionally, the outside world is the adversary Eve. An assumption here is that there is some boundary where Eve might not have full access to. The model is constrained within device-dependent framework. Compared to some previous QRNG implementations [95–97, 99–104], where all the side-information was classical, Eve's information here is treated as quantum. Fig. 4.4 shows two trusted devices, the signal source (producing

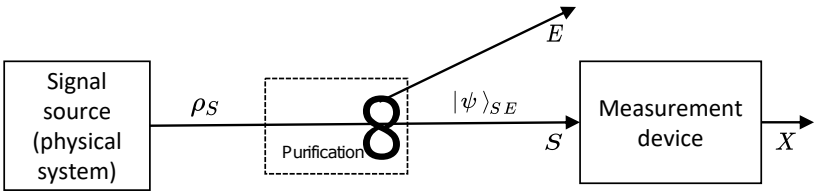


Figure 4.4: Graphical depiction of quantum side information

the physical system) and the measurement device. The state of the physical system could be entangled to an adversary's state E (a purification). It is

interesting to make parallels with QKD, as the general setup is very similar. QKD systems could be seen as quantum random generators that produce shared randomness. Along with Definition 4.1.2, in [20] there is a formal definition of QRNG:

Definition 4.2.1. A QRNG is defined by a density operator ρ_S on a system S together with a projective measurement $\{\Pi_S^x\}_{x\in X}$ on S. The raw randomness is the random variable X obtained by applying this measurement to state ρ_S.

The probability distribution of the variable X is then

$$P_X(x) = \mathrm{tr}(\Pi_S^x \rho_S). \tag{4.2.1}$$

This definition translated to a real system implicitly considers two assumptions:

- The state of the system is pure
- The measurement of the system is projective

In reality, these assumptions are very strong and neither of them is true, therefore weaker assumptions should be made. The prepared state is generally in a mixed state, and the measurements act like a general *positive-operator valued measurement* (POVM).This all means that there might be side information correlated to the measurement results. The task is to properly model and quantify this side information in order to lower bound the min-entropy, so the correlations are removed in the act of randomness extraction. Even though the side information is not explicitly mentioned in the definition, a purification of the input signal state is considered, denoted by E which is held by Eve as shown in Fig. 4.4.

There is focus on five critical issues in our QRNG. The issues have not yet been addressed before in a single implementation.

1. Treating the adversary to be quantum-capable as described above.
2. Limited bandwidth and a realistic transfer function inevitably introduce correlations between samples, so the i.i.d. assumption is not justified.
3. Using metrology-grade characterization of important parameters of the devices used. Confidence intervals for the parameters are calculated, so the value for the (lower bounded) min-entropy is trustworthy with the same level of confidence. This way a higher level of security is possible within DI-QRNG framework.
4. Information-theoretic secure high-rate randomness extraction.

5. Real-time monitoring of the min-entropy where the randomness offload is stopped if the value of the min-entropy goes below a certain limit, thus retaining security of QRNG users.

Calculating the min-entropy

Detailed security analysis can be found in the supplemental material of the preprint of our paper [105]. It goes beyond the scope of the thesis so here we present only final expressions.

First, an i.i.d. scenario is assumed where signals at different times are independent and identically distributed. Quantum side-information is accounted by assuming the source is emitting thermal light with mean photon number n_{th}, and the adversary is holding a purification of that thermal state. The purified state can be assumed to be a two-mode squeezed vacuum state $\hat{\rho}_{XE}$, without loss of generality. For such a state, the output of the homodyne measurement has the probability density distribution $P_X(x) = G(0, g^2(1 + 2n_{\text{th}}))$, where G is Gaussian distribution (with zero mean and variance $g^2(1 + 2n_{\text{th}})$ and g is a gain factor. A lower bound on min-entropy is obtained and it is a function of the mean photon number, the gain factor and the ADC parameters $H_{\min} = H_{\min}(n_{\text{th}}, g, ADC)$.[5]

After this, a non-i.i.d. scenario is analyzed, where the measured signal has a finite bandwidth. Similar to QKD, there is a distinction between the signal and the excess noise that is assigned to Eve. The signal is defined as a result of the homodyne measurement including all the noise sources, while the excess noise is obtained by subtracting the vacuum fluctuations from the signal. Let us assume K samples of the QRNG output X are captured during a measurement. For estimating the power spectral density of the captured signal we use Bartlett's method [106].[6]The samples are divided into K/M non-overlapping blocks of the same length M. To obtain the PSD estimate, a periodogram using FFT transformation of length M is calculated for each block. Let us denote the variance of the measured signal as σ^2. The variance, including the confidence interval for the PSD estimate, is calculated as

$$\sigma^{2\pm} = \left(1 \pm 4\sqrt{\frac{M}{K}\ln\frac{2M}{\epsilon}}\right)\sum_{j=0}^{M}\frac{1}{M}\mathcal{S}_X(f_j), \tag{4.2.2}$$

[5]We avoid writing the whole formula as it is lengthy and not very insightful. This is expression (27) in [105].

[6]Bartlett's method is a special case of widely used Welch's method for estimating PSDs where there is no overlap between periodograms. More on the FPGA implementation of Bartlett's method is in Section 4.5.

where $\mathcal{S}_X(f_j)$ is the PSD estimate and ϵ is the probability that the value lies outside the interval. The non-i.i.d. influence is included by modelling a noise source that is i.i.d. and has the same Shannon entropy as the measured signal. The variance of this signal is called the conditional variance (conditioned on i.i.d.) and it is calculated as

$$\sigma_X^2 = \frac{1}{2\pi e} 2^{\int_0^{2\pi} \frac{df}{2\pi} \log(2\pi e \mathcal{S}_X(f))}, \tag{4.2.3}$$

or using PSD samples and including the confidence interval

$$\sigma_X^{2\,\pm} = \frac{1}{2\pi e} 2^{\sum_{j=1}^{M} \frac{1}{M} \log(2\pi e \mathcal{S}_X(f_j))} 2^{\pm 4 \log e \sqrt{\frac{M}{K} \ln \frac{2M}{\epsilon}}}. \tag{4.2.4}$$

A random variable N is assigned to the excess noise, so corresponding power spectral density is $\mathcal{S}_N(f)$, with the conditional noise variance being σ_N^2:

$$\sigma_N^{2\,\pm} = \frac{1}{2\pi e} 2^{\int_0^{2\pi} \frac{df}{2\pi} \log(2\pi e \mathcal{S}_N(f))} 2^{\pm 4 \log e \sqrt{\frac{M}{K} \ln \frac{2M}{\epsilon}}}. \tag{4.2.5}$$

By definition of the excess noise, we identify $\sigma_N^2 \equiv 2g^2 n_{\text{th}}$. By exploiting the conditional distributions the non-i.i.d. setting is mapped to the i.i.d. model with

$$g^2 = \sigma_X^2 - \sigma_N^2, \tag{4.2.6}$$

$$n_{\text{th}} = \frac{1}{2} \frac{\sigma^2}{\sigma_X^2 + \sigma_N^2} - \frac{1}{2}. \tag{4.2.7}$$

To obtain the worst-case estimate of the min-entropy we consider the smaller value for the conditional variance ${\sigma_X^2}^-$, and the larger one for the noise ${\sigma_N^2}^+$.

In order to get the conditional variance of the excess noise, we make a conservative estimate of the vacuum noise in the calibration step before the QRNG is started. Our strategy is to measure the transfer function of the homodyne detector and calibrate the vacuum fluctuations PSD (Eq. 3.2.20). The transfer function is measured by injecting a coherent state in form of a second laser beam and producing a beat signal. The transfer function is calculated by detuning the two lasers and swiping the frequency range. For each frequency the beat signal power is recorded. The transfer function includes imperfections such as optical loss, quantum efficiency of the photodiodes, the analog bandwidth of the ADC etc. For the implementation of the system, including the calibration, see Section 4.7.

The signal variance and the conditional variance can be measured in real time as they only depend on the measured samples. With the vacuum noise PSD calculated in the calibration step we are able to monitor the min-entropy in real time. Fig. 4.5 shows how min-entropy depends on the correlations and excess noise power.

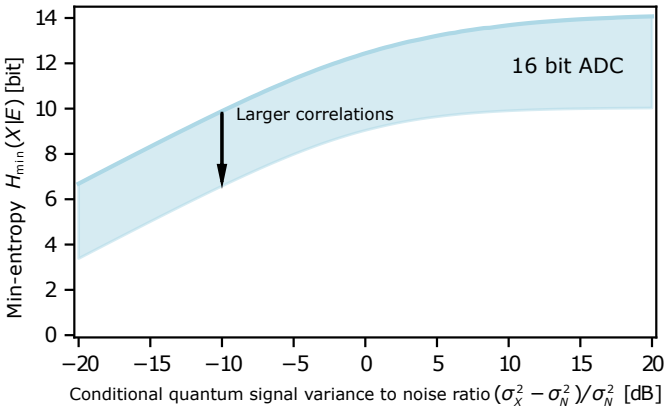


Figure 4.5: Simulated values of the min-entropy for different values of the ratio of conditional variance of the vacuum fluctuations and the conditional variance of the excess noise for a 16-bit ADC like the one we use. The upper and lower boundaries indicate low correlations and high correlations respectfully.

4.3 Theory of randomness extraction

This section focuses on information-theoretic secure randomness extraction. It is an overview of important concepts starting with the universal hashing discovered forty years ago, until the last decade when the classical information concepts were applied to quantum cryptography scenarios and proven to be secure even against quantum adversaries. If we look at parts of QRNG (QKD) protocol in a sequential manner, the randomness extraction (the privacy amplification) comes after the measurement of quantum states (and after the parameter estimation step for the case of QKD). That means the min-entropy is already estimated. Their task is now to extract as much information and as efficiently possible from a string of raw bits, where min-entropy represents an upper bound of extractable secret bits. Note that the thesis focuses on randomness extraction for QRNG, and even though privacy amplification is the same concept, it operates with different block lengths and brings other implementational challenges, so it is one of the topics for future work.

4.3.1 Universal hashing

Following this philosophy of information-theoretic security, we focus our interest to classes of hash functions called two-universal (or universal2) that have desired properties. Universal classes of hash functions were introduced in 1979 in a seminal paper by Carter and Wegman [107]. In this paper the authors use universal classes for a storage and retrieval algorithm on keys.

They showed some of the properties of the classes.

Definition 4.3.1. (Two-universal hash functions) Let $\mathcal{H} = (h_k)_k$ be a family of functions, the set of functions from $A = \{0,1\}^n$ into $B = \{0,1\}^m$ indexed by a key $k \in \{0,1\}^d$. It is said that $\mathcal{H}$ is a two-universal hash function family if

$$\text{for any } x, y \in A,\ x \neq y,\ \Pr_k[h_k(x) = h_k(y)] \leq \frac{1}{|B|}, \quad |B| = 2^m, \quad (4.3.1)$$

where $\Pr(\cdot)$ is the probability of the collision occurring.

In other words, no pair of distinctive messages x and y *collide* under more than $1/|B|$ of the functions. The expression two-universal is intended to emphasize that the definition constraints the behaviour of H only on pairs of elements of A. There are other definitions, such as strong universality, however the authors argue that the defined property is powerful enough for many purposes. It was shown that this was true, as it is possible to build a strong randomness extractor using universal hashing.

The existence of universal hashing does not mean much for real life applications if the calculation algorithms are inefficient. The problem of computational complexity was tackled in [108, 109]. In [109], a proof was given for the universality of a family of hash functions defined by convolution:

Theorem 4.3.1. ***(Universality of convolution)*** *Let $A = \{0,1\}^n$ and $B = \{0,1\}^m$. For $x \in \{0,1\}^n$ and $y \in \{0,1\}^{n+m-1}$ we define the convolution of y and x, $y \circ x$, to be the m-bit string z whose i-th bit is given by $z_i = \sum_{j=1}^{n} x_j y_{i+j-1} (\mathrm{mod}\, 2)$. For two m-bit strings x and y let $x \oplus y$ denote the bitwise exclusive-or of the two strings. Then the following family is a universal family of hash functions:*

$$H = \{(a \circ x) \oplus b | a \in \{0,1\}^{n+m-1}, b \in \{0,1\}^m\}.$$

Due to importance of this theorem for this work, a sketch of proof from the same paper is presented.

Sketch of the proof. One fixes $x_1 \neq x_2$ and y_1, y_2 and counts the number of vectors a and b in order to calculate the probability from the universal hashing definition. For a single a, there is single b which satisfies the expression, therefore counting the number of a is sufficient. Let $x = x_1 \oplus x_2$ and $y = y_1 \oplus y_2$, we have that $x \neq 0$. Consider the set $a | a \circ x = y$. It is the solution space of n equations. Since $x \neq 0$, the equations are independent, thus it defines a hyperspace of dimension exactly $m - 1$ in Z_2^{n+m-1}. □

This theorem implicitly introduces *Toeplitz hashing*, since the multiplication of a vector with a Toeplitz matrix can be represented as (part of)

convolution of the same vector with Toeplitz matrix *seed*. It should be noted that the paper [109] deals with algorithm complexity for software implementations. However, Toeplitz constructions are even more convenient for FPGA implementations which is shown in the following sections of this chapter.

Definition 4.3.2. (Toeplitz matrix) The original definition of Toeplitz matrix [110] states that it is a $n \times n$ matrix $T_n = [t_{i,j}|j = 0, , \ldots, n-1]$ where $t_{i,j} = t_{i-j}$, i.e., a matrix of the form

$$T_n = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \ldots & t_{-(n-1)} \\ t_1 & t_0 & t_{-1} & & \\ t_2 & t_1 & t_0 & & \vdots \\ \vdots & & & \ddots & \\ t_{n-1} & & & \ldots & t_0 \end{bmatrix} \tag{4.3.2}$$

For our applications it is convenient to define a generalization where Toeplitz matrix is of size $n \times m$ with the form:

$$T_{m\times n} = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \ldots & t_{-(n-1)} \\ t_1 & t_0 & t_{-1} & & \\ t_2 & t_1 & t_0 & & \vdots \\ \vdots & & & \ddots & \\ t_{m-1} & & & \ldots & t_0 \end{bmatrix} \tag{4.3.3}$$

Definition shows that Toeplitz matrix is defined by its first column and first row. This means it is defined by $n + m - 1$ bits in total. We call this string Toeplitz matrix seed. Multiplication of an array of length n, with a Toeplitz matrix of size $m \times n$ yields an array of size m. One can easily check that this array is just a part of a longer array which is the result of the discrete convolution between the matrix seed array and the original array of length n: let the seed array be defined as $S = [t_{-(n-1)}, \ldots, t_0, \ldots, t_{m-1}]$, therefore the corresponding Toeplitz matrix is defined with $T_{m\times n} = [t_{i,j}|i = 0, \ldots, m-1; j = 0, \ldots, n-1; t_{i,j} = t_{i-j}]$ An array $A_n = a_0, \ldots, a_{n-1}$ is multiplied with the Toeplitz matrix. On the other hand, let the convolution of A and S be defined as

$$(A \circ S)(k) = \sum_{i=0}^{n-1} a(i)t(k-i). \tag{4.3.4}$$

It is obvious that the convolution is the same as the result of matrix multiplication $T_{m\times n}A_n$. This proves that Toeplitz matrices represent a universal family of hash functions. Each function is defined with a seed of length $n + m - 1$, hence the total number of functions in this family is 2^{n+m-1}.

4.3.2 Almost universal hashing

Sometimes it is very useful for practical purposes to relax the limit for the collision probability of a hash function family. This leads to a definition of *almost universal hash functions*. Here the maximum collision probability is higher than $1/|B|$ [111].

Definition 4.3.3. (ε-almost universality) Let $\mathcal{H} = (h_k)_k$ be a family of functions in $\mathcal{F}_{n,m}$, the set of functions from $A = \{0,1\}^n$ into $B = \{0,1\}^m$ indexed by a key $k \in \{0,1\}^d$. We say that $\mathcal{H}$ is an ε-almost universal hash (ε-AUH) function family if

$$\text{for any } x, y \in A,\ x \neq y,\ \Pr_k[h_k(x) = h_k(y)] \leq \frac{1}{|B|} + \varepsilon. \tag{4.3.5}$$

It is clear that for $\varepsilon = 0$ this becomes universal hash function family. To illustrate how the almost universal definition helps in practice there is an example of efficient Toeplitz hashing proposed in 1994 by Krawczyk [108]. It was shown that Toeplitz matrices defined by $n+m-1$ bits are a universal family of hash functions. For some applications the key (seed) of this length is too large, as the key itself is longer than the message length n. By relaxing the universal hashing requirement one is able to produce a hash function that is *almost* as secure as the universal one. The author proceeds with a proof of *LFSR-based Toeplitz hashing*. This matrix is defined by its first column. Each column is built as a state of linear feedback shift register (LFSR); consecutive columns are consecutive states. The first column is the initial state of the register. This scheme requires $2m$ bits – m bits for initial state and m bits for defining the polynomial (LFSR pads). Matrices built this way are a subset of $m \times n$ Toeplitz matrices. The paper proofs that if the LFSR represents an irreducible polynomial, the generated matrix is a member of an almost universal hash function family. The theorem states[7]:

Theorem 4.3.2. *The LFSR-based Toeplitz construction defined above is ε-almost universal for $\varepsilon = \frac{2n-1}{2^m}$.*

Obvious advantage of this scheme is much smaller seed, therefore very convenient for cryptographic algorithms, such as authentication where the seed (the secret authentication key) is valuable resource. By manipulating the message length n, one is able to achieve the right trade-off between implementation efficiency and ε.

4.3.3 Leftover hash lemma and strong randomness extractors

As Definition 4.1.2 implies, a random number generator output not only needs to be uniform, but also independent of other variables. Again, malev-

[7] The original theorem is formulated a bit differently. Krawczyk uses the term ε-balanced hashing. This formulation is equivalent to the one from the paper.

olent adversary Eve is introduced. Eve could be introducing classical noise to a classical RNG system that could in turn enable her to gain some partial knowledge of the future RNG outputs. In QRNG systems, the adversary could gain quantum side information about the physical system. This quantum side information needs to be treated differently than the classical one. It might also give more power to the adversary. Regardless of the attack model, randomness extraction is the final step required in order to get uniform random numbers. After the raw numbers are obtained from a physical system, and a security model was made, one needs to process the data so the (quantum enabled) adversary is left with almost zero information about the output string. The final result of parameter estimation is the min-entropy of the measured and discretized entropy source raw output.

Definition 4.3.4. (Extractor) A $(h, \varepsilon, n, d, m)$-extractor is a function

$$\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m, \tag{4.3.6}$$

such that for every probability distribution P_X on $\{0,1\}^n$, with $H_{\min}(P_X) \geq h$, the probability distribution $\mathrm{Ext}(P_X, U_d)$ is ε-close to the uniform distribution on $\{0,1\}^m$. U_d denotes uniform distribution on alphabet $\{0,1\}^d$.

Definition 4.3.5. (Strong extractor) A $(h, \varepsilon, n, d, m)$-strong extractor $\mathrm{Ext}(P_X, U_d)$ is an extractor such that the probability distribution $\mathrm{Ext}(P_X, U_d) \circ U_d$ is ε-close to the uniform distribution on $U_{m+d} = \{0,1\}^{m+d}$.

The definition of strong extractors points out that the random seed is not correlated with the extractor output. This means the seed can be reused for multiple blocks of data and can be chosen only once in the beginning of the extraction operation [20]. This feature is very convenient for practical purposes.

Leftover hash lemma (LHL) has a central role in applications of universal hash functions to randomness extraction:

Theorem 4.3.3. ***(Leftover Hash Lemma)*** *Let $\mathcal{H} = \{h_1, h_2, ..., h_{2^d}\}$ be a two-universal hashing family, mapping from $\{0,1\}^n$ to $\{0,1\}^m$, and P_X be a probability distribution on $\{0,1\}^n$ with $H_{\min}(P_X) \geq h$. Then for $x \in X$ and $h_k \in \mathcal{H}$ where $k \in U_d$, the probability distribution formed by $\mathrm{Ext}(P_X, U_d)$ is $\varepsilon = 2^{(m-h)/2}$-close to U_{m+d}. This means it forms a $(h, 2^{(m-h)/2}, n, d, m)$-strong extractor.*

It was proven earlier that the set of Toeplitz matrices is a family of universal hash functions. According to LHL this means these matrices could be used as strong extractors. Toeplitz matrices are a well known choice for implementing randomness extraction and privacy amplification used in QKD

[112, 113]. Here, the seed of a Toeplitz matrix and the one of an extractor are the same. Note that if the matrices were not universal hash family, the randomness extraction would not be possible due to the fact that the seed is longer than the output vector. In other words one would need to spend more randomness for *reseeding* than they gain.

The follow-up question is: can almost universal hash functions be used as strong randomness extractors? Luckily, the answer is yes [111, 120, 121]. The leftover hash lemma with almost universal hash functions (AUH) is defined:

Theorem 4.3.4. ***(Leftover Hash Lemma with ε-AUH).*** *Let $\mathcal{H} = \{h_1, h_2, ..., h_{2^d}\}$ be an ε-almost two-universal hashing family, mapping from $\{0,1\}^n$ to $\{0,1\}^m$, and P_X be a probability distribution on $\{0,1\}^n$ with $H_{\min}(P_X) \geq h$. Let e be an integer and $m \leq \alpha - 2e$ where $\alpha = \min(h, \log(1/\varepsilon))$. Then for $x \in X$ and $h_k \in \mathcal{H}$ where $k \in U_d$, the probability distribution formed by $\mathrm{Ext}(P_X, U_d)$ is $\varepsilon = 2^{-e}$-close to U_{m+d}. This means it forms a $(h, 2^e, n, d, m)$-strong extractor.*

The proof in [111] is the more general than some previous proofs and is valid for any ε. The usage of almost universal hashing enables the usage of very practical schemes such as Krawczyk's LFSR hashing.

So far we considered the choosing of a hash function to be perfect, i.e. we assumed the seed was chosen from a perfect uniform distribution. Obviously, this is never the case in real world, and further analysis is required. The problem of imperfect seed was tackled in [116]. The main theorem in the paper gives the security parameter for ε-biased sequences. Before stating the theorem of the paper, there is a definition of ε-biased distributions:

Definition 4.3.6. (ε-biased distributions) Let P_X be a distribution on $X = \{0,1\}^n$. Let (χ, x) denote the scalar product modulo 2 of $\chi \in \{0,1\}^n$ and $x \in \{0,1\}^n$. Then, P_X is said to pass the linear test χ with bias ε if $|Pr((\chi, x) = 1) - \frac{1}{2}| \leq \varepsilon$. P_X is said to be an ε-biased distribution if it passes all linear tests $\chi \neq 0$ with bias ε.

Intuitively, the notion of ε-bias distributions is very similar to ε-closeness. Let us prove the following theorem:

Theorem 4.3.5. *If distribution P_X, with elements $x \in \{0,1\}^n$ is ε-close to the uniform one, then it is ε-biased.*

Proof. It is easy to see that for any $\chi \neq 0$, the condition $Pr((\chi, X) = 1$ is satisfied for exactly half of the elements in the set $\{0,1\}^n$. Let us denote with X' a subset of $\{0,1\}^n$, with exactly half of total elements, and with $\hat{X}'$ its complement. Now we can write a sequence of inequalities starting with the definition for ε-close distributions to the uniform one:

$$\varepsilon \geq \sum_i |P_X(x_i) - \frac{1}{2^n}| = \sum_{i|x_i \in X'} |P_X(x_i) - \frac{1}{2^n}| + \sum_{i|x_i \in \hat{X}'} |P_X(x_i) - \frac{1}{2^n}| \geq$$
$$\sum_{i|x_i \in X'} |P_X(x_i) - \frac{1}{2^n}| \geq |\sum_{i|x_i \in X'} P_X(x_i) - \sum_{i|x_i \in X'} \frac{1}{2^n}| = |\sum_{i|x_i \in X'} P_X(x_i) - \frac{1}{2}|. \tag{4.3.7}$$

In the first line we split the sum into two over subset X' and its complement $\hat{X}'$. X' is arbitrary with a constraint of having exactly half of the elements from $\{0,1\}^n$. It is obvious that $\sum_{i|x_i \in X'_\chi} P_X(x_i) = Pr((\chi, X) = 1)$ where X'_χ is the subset of elements for which the corresponding scalar product is 1. □

The main theorem from [116] states:

Theorem 4.3.6. *Let P_Y be a distribution on $\{0,1\}^{n+m-1}$. Let $\{T_Y\}$ be the set of $m \times n$ Toeplitz matrices generated by the elements from Y. If the distribution P_Y is ε-biased then the family $\{T_Y\}$ is a $2^{-m}+\varepsilon$-balanced family of hash functions.*

In other words such a family is ε-almost universal.

Apart from the extraction application, hash functions are used extensively for authentication schemes. Universal hash functions are used in constructing information-theoretic secure authentication [115] where the hash output is XOR-ed with one time pad. In our case ε-otp-security is equivalent to ε-balanced since Toeplitz hashing is XOR-linear [108].

As the previous analysis showed, engineers need to be aware of the seed creation problem, and the security parameter related to it. There are two main requirements: the seed should be independent of raw bits produced in a particular QRNG and it should be as close as possible to the uniform distribution. An interesting analysis could be found in [117]. One of the possibilities is to use several other weak random sources, XOR the outputs and generate high entropy bit. Before the XOR-ing, on each output one can use some techniques to minimize the distance to uniform distribution. The examples are von Neumann de-biasing algorithm and taking non-consecutive samples in order to minimize correlations.

When the requirements for both universality of the hashing and uniformity of the seed is relaxed a natural next step is combining everything into one expression that gives a single security parameter for the randomness extraction. Intuitively, all error parameters should add up. A proof in [20] shows

this intuition is justified. It is assumed the hashing is done in blocks of data using (almost)-universal hash function h_k which is chosen according to the seed. Let the hash function be applied to N blocks of data $x_1, ..., x_N$ where each X_i is a random variable with alphabet $\{0,1\}^n$. The final distribution $h_k(x_1), ..., h_k(x_N)$ is the concatenation of the N post-processed blocks. The distribution corresponding to imperfect seed is denoted with D. The argument is also described for the case of classical side information C. The authors note that by replacing all probability distributions by density operators, the argument is generalised to the case of quantum side information. Let the statistical distance of the random seed to the uniform distribution be $\varepsilon_{\text{seed}}$:

$$\varepsilon_{\text{seed}} = \|P_{x_1...x_N CD} - P_{x_1...x_N C} \times U_D\|_1. \tag{4.3.8}$$

Let $\varepsilon_{\text{hash}}$ be the security parameter calculated according to the leftover hash lemma for (almost) universal hash families. Randomness extractor is strong hence the seed is fixed. The statistical distance of the extractor output is then:

$$\|P_{h_k(X_1)...h_k(X_N)CD} - U_n^N \times P_C \times U_D\|_1 \leq N\varepsilon_{\text{hash}} + \varepsilon_{\text{seed}}. \tag{4.3.9}$$

Finally we stress that the formulas for randomness extraction are valid even with quantum side information E held by Eve and correlated to the measurement outcome X (raw bits obtained from the measurement device), so the min-entropy $H_{\min}(X|E)$ is calculated with the regard to the quantum side information [20, 119, 122].

4.4 FPGA implementation of the randomness extractor

The two main motives for implementing Toeplitz matrix multiplication are the universality of such operation and the opportunity for significant parallelization in FPGA. Even though Kintex UltraScale is a powerful chip, Toeplitz module still uses relatively large amount of resources, which leads to slower compiling and debugging. The final result that is presented is a product of tuning the design parameters for applications in both the standalone QRNG setup, and in the CV-QKD transmitter.

For the purpose of implementing Toeplitz extractor we cloned a Capture star (see Appendix A.4). This was the most convenient solution as the star already had two AXI-Stream interfaces for data streaming, and AXI-Lite for read/write operations on memory mapped registers. This was critical for enabling the software control of the module. The Toeplitz module takes 64-bit input directly from one of the ADCs through the FMC120 star. The input is streaming four 16-bit samples constantly at the system clock frequency of 250 MHz, which totals to 1 GS/s (16 Gb/s) of constant data

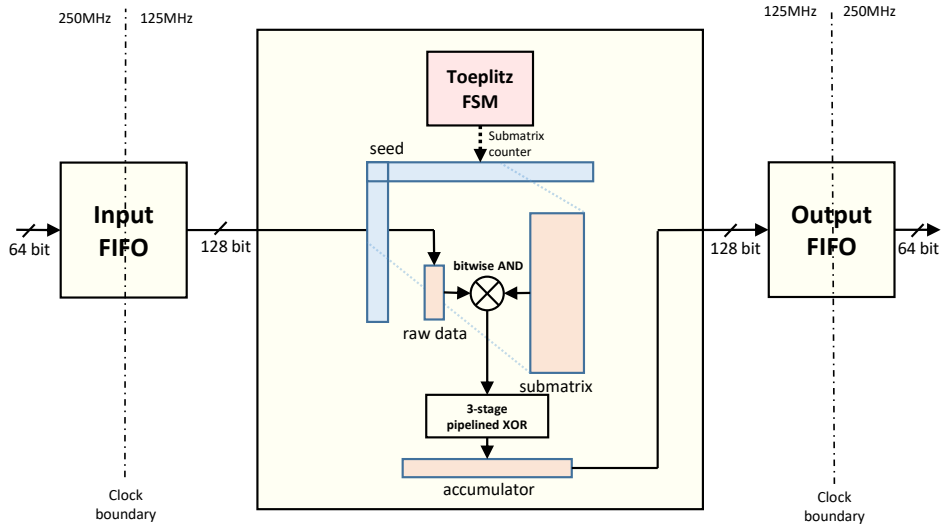


Figure 4.6: Block diagram of Toeplitz hashing module.Blue – Toeplitz matrix seed is saved in FPGA distributed memory elements and remains constant throughout the hashing procedure. Light pink – finite state machine for controlling the module. Orange – data vectors and the submatrix.

rate. Our goal was to design a module that can process all the data in real time. The output of the module is also 64-bit wide. Hashing reduces data rate, so the output data comes in bursts (since the clock rate stays the same).

Toeplitz matrix hashing algorithm was implemented in a similar fashion as in [125, 126]. On each clock cycle the module gets 64 bits of raw data for processing. Since we are aiming for the maximum possible throughput, the processing of all 64 bits must be done in one clock cycle. For this reason we employ the submatrix method. Let the full Toeplitz matrix be of size $m \times n$. This means that one raw data block is n-bit wide. The result is a m-bit vector of hashed data. The block diagram of the module is shown in Fig. 4.6.

In the development stage of the module it was realized it would be a good solution to lower the clock rate for the module, at expense of using more resources, due to a relatively large logic network and possible timing problems. In the VHDL description of the module we extensively use generic parameters, for easier modification of module parameters such as the matrix size. The experience showed that it was much harder to achieve a timing closure[8]

[8]This refers to a set of techniques a developer can use to make a design where there are no timing violations.

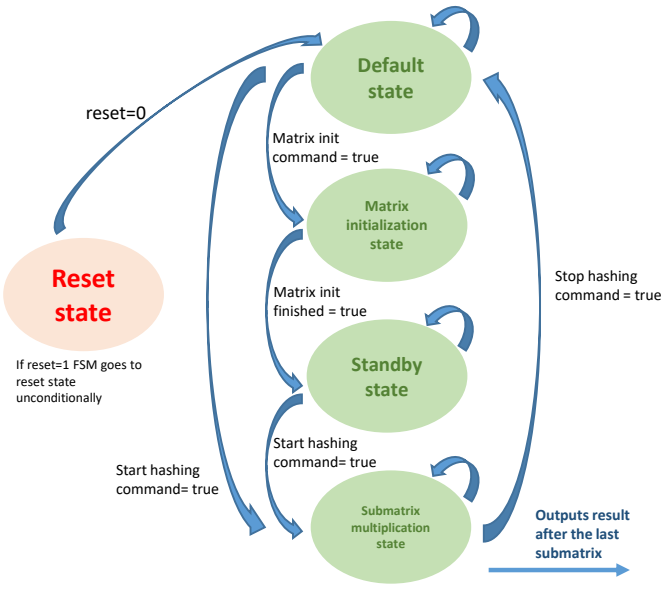


Figure 4.7: Toeplitz hashing FSM

with a smaller matrix and 250 MHz clock compared to the system with a bigger matrix and 125 MHz clock. We build the module to use 125 MHz clock by isolating it with two clock boundary FIFOs, one for the input and one for the output. Since we want to preserve the rate, the clock is divided by two, but the raw word length is multiplied by two. The algorithm takes 128-bit words at 125 MHz. With these parameters we aim for the matrix dimensions to be multiples of 128. The idea is to split the Toeplitz matrix into smaller $m \times 128$ submatrices. There are in total $n/128$ submatrices.

The algorithm works as follows[9]. The raw data vector is multiplied with each of m rows of $m \times 128$ submatrix (in one clock cycle), and the resulting m bits are stored in an accumulator register. Multiplication of the raw vector and a row is done by employing the bitwise AND and then XOR-ing all 128 bits into a single bit. The whole n-bit block is processed in $n/128$ clock cycles. All operations are done with bits and thus in the finite field $GF(2)$. Therefore, the multiplication translates to AND operation, while the addition is implemented as XOR gate. By using inherent parallel processing of FPGAs, we are able to implement the multiplication of a submatrix and a 128 bit raw data vector effectively in a single clock cycle.

For controlling the Toeplitz extractor a finite state machine (FSM) is

[9]Without worrying too much at this point of what the matrix size should be.

implemented. Some of the FSM's input signals are slow software-controlled signals. Fig. 4.7 shows the states and the transitions. The machine starts in the Reset state where the seed and all the counters are reset to zero. From there FSM can only go to the Default state from where a user can choose to proceed with the hashing (if the matrix is initialized already), or to switch to Matrix initialization state where the user initializes the Toeplitz matrix with a new seed. This is all done using reference API and software that communicates with Toeplitz module (star) through AXI-Lite interface. After the matrix initialization the FSM goes to the Standby state where it waits for a signal to start the hashing. After the hashing is started the module hashes the input data indefinitely until the user stops the operation.

With this basic idea of submatrix multiplication we proceed planning the optimal solution for the real FPGA implementation. Leftover hash lemma (LHL) shows how large the Toeplitz matrix should be for a particular security bound ε. Let us rewrite LHL using the particular Toeplitz hashing parameters:

$$m = nH_{\min} - 2\log(1/\varepsilon), \tag{4.4.1}$$

where m and n are matrix dimensions defined above and $H_{\min}$ is calculated min-entropy[10] for our QRNG. ε is the information-theoretic security bound. It is defined as a statistical distance between the output sequence and the perfect uniform sequence. The security bound can also be interpreted as an advantage[11] of the adversary – a probability higher than guessing probability with which she can differentiate the QRNG output from a perfect uniform stream. To get a feeling of what the matrix dimension should be, if we assume ε is less than 10^{-10}, we need the difference $nH_{\min} - m$ to be at least 60 bits. Now we define extraction efficiency as

$$\text{eff} = \frac{m}{n \times H_{\min}}, \tag{4.4.2}$$

which is according to LHL always less than 1. The efficiency can be made arbitrarily close to 1 with increasing the matrix size. Therefore, there is a trade-off between the matrix size (the implementation complexity) and the output rate (the efficiency) on one side, and the security bound on the other.

Due to the submatrices being created on each clock cycle, the seed is required to be readily available to the rest of the logic. That is why the seed is not stored in Block RAM (BRAM) cores, as they require sequential reading. It might have been stored in the external DDR4 RAM, however the RAM is used already for a different purpose in the firmware.[12] The seed is

[10] Here min-entropy is given per bit and not per sample.

[11] The notion of advantage is often used in classical cryptography.

[12] In standalone QRNG the RAM is used for buffering the raw sample.

initialized as a vector and kept in FPGA logic flip-flops. Even though the FPGA is large enough, it would be incredibly hard, but also unnecessary, to build a whole Toeplitz matrix at once. Instead, on each clock cycle the logic takes part of the seed, builds a submatrix and performs the multiplication.

Compiling time for different designs			
	Parameter set 1	Parameter set 2	Parameter set 3
Rate[Gb/s]	5.3	8	8.8
Matrix dimensions (m, n)	(256,768)	(384,768)	(640,1152)
Extraction efficiency [%]	56	83	93
Security parameter ε_{hash}	2^{-102}	2^{-38}	2^{-25}
Compiling time [h]	7	10	32

Table 4.1: Comparison of three sets of test parameters. Assumed $H_{min} = 0.6$.

Submatrix multiplication with the raw data word implies several hundreds of multiplications of 128 bit vectors, which produce huge combinatorial network that might lead to setup timing violations. The bitwise AND is implemented easily, however XOR-ing of 128 bits is done in three stages. 128 bits are first XOR-ed in groups of 8 and then two times in groups of 4 ($8 \times 4 \times 4 = 128$). This introduces latency of three clock cycles, however it does not influence the rate since the operation is pipelined.

During the development stage we tested several sets of parameters for the Toeplitz module. Table 4.1 illustrates the trade-off between the output rate and the complexity by comparing three different parameter sets with different output rates. It is interesting to present the compiling time as a figure of merit to show the practical differences. Note that the reference min-entropy for all three sets is chosen to be $H_{\min} = 0.6$ bits per one bit of the raw input, and this is an arbitrary chosen value at this point for comparison purposes. The theoretical maximum for the output rate in this case would be $H_{\min} \times 16, Gb/s = 9.6, Gb/s$. Since the submatrix width is kept fixed at 128 bits, the only parameter that is relevant for the size of the design is the number of rows m. It is obvious how the complexity and the security parameter grow as the output rate and the extraction efficiency asymptotically rise to their respective maximum values.

Final parameter set	
Rate[Gb/s]	8
Matrix dimensions (m, n)	(640,1280)
Extraction efficiency [%]	75
Reference min-entropy [per bit]	0.669
Security parameter ε_{hash}	$\approx 2^{-108} \approx 10^{-30}$
Compiling time [h]	34

Table 4.2: Parameters for the final design used in QRNG and QKD transmitter setups

Toeplitz FPGA implementation resource utilization			
FF	**LUT**	**BRAM**	**FIFO**
26943	91540	0	6

Table 4.3

Parameters and performance of the final Toeplitz module design

The most important requirement for the final working design is to achieve a proper level of security. According to Eq. 4.3.9, the total security parameter $\varepsilon_{\text{QRNG}}$ grows linearly with the number of hashed blocks. We make a requirement that our QRNG should run for 10 years with the security parameter $\varepsilon_{\text{QRNG}} < 10^{-10}$. If the number of blocks is N, this condition can be written as

$$\varepsilon_{\text{QRNG}} = N\varepsilon_{\text{hash}} + \varepsilon_{\text{seed}} < 10^{-10}. \tag{4.4.3}$$

Table 4.2 shows the final parameter set. We aimed for the maximum number of rows of the Toeplitz matrix $m = 640$ that we tested before, in order to achieve a good output rate and decent extraction efficiency for a very small ε_{hash}. The number of hashed blocks in ten years for the output rate of 8 Gb/s is $N \approx 4 \times 10^{15}$. With $\varepsilon_{hash} \approx 10^{-30}$, the total security parameters is $\varepsilon_{QRNG} \approx 10^{-15} + \varepsilon_{seed}$. Since the seed provided from a completely independent system we are free to chose any source that gives $n + m - 1 = 1919$ bits with $\varepsilon_{seed} < 10^{-10}$.

Table 4.3 shows the number of FPGA resource blocks used by the Toeplitz module. Even though the module is the largest one in the whole FPGA firmware, it utilizes only 17% of all the LUTs and all the other resources are used in even smaller percentage.

4.5 FPGA implementation of the online entropy test

From the point of view of the firmware there are two modes of QRNG operation: calibration mode and standard mode. As it was described in Section 4.2, we treat the homodyne detector as a box that is characterized by its transfer function. During the calibration mode, the firmware supports the transfer function measurement. In the standard mode of operation, Toeplitz hashing is activated and random numbers are offloaded.

During the standard mode of operation, QRNG constantly performs the randomness extraction and is sending random numbers to a user. From the security point of view it is essential for the user to be sure that the produced numbers are indeed random for the whole time of operation, and with the claimed security parameter. For this purpose randomness generators implement *online entropy tests* to confirm in real time that the RNG output is indeed random. Pseudo-random number generators generally implement statistical tests on the output to verify that the numbers look random enough. In order to be secure, our QRNG must include an online test according to the security proof. Since the PSD of the vacuum noise is calculated in the begining as a result of the detector characterization, the real-time monitoring of the signal PSD is enough to calculate the min-entropy. The QRNG produces random numbers until the real-time min-entropy value goes beyond a threshold value. At that point offloading is stopped and an alarm is asserted.

Fig. 4.8 shows the most important modules (or stars, see Appendix A) of the QRNG firmware.[13] The homodyne detector output is discretize with a 16-bit ADC. In the calibration stage, only the data path above is active and the online test module is not used. Star '64 to 256' (and the others similarly named) are basically FIFOs for converting bus width so stars with different bus width are connected. During the calibration stage DEMUX is set to route raw sampled data to the PCIe interface (to the host machine). The DEMUX is controlled by the software. A clock boundary FIFO is introduced to synchronize the data to the host interface clock. Router star is a standard part of the reference firmware and can be controlled by software. In this case it just routes the data to the PCIe host interface. The captured data is later analyzed using software on the host machine for the transfer function calculation. This operation is not time-critical as it is done only once before the standard mode of operation where the random numbers are produced. When the calibration is finished, the host machine passes the reference values of the noise variances that will be used by the online test module for comparison.

[13] Note this is the standalone QRNG firmware not used as a part of the QKD transmitter.

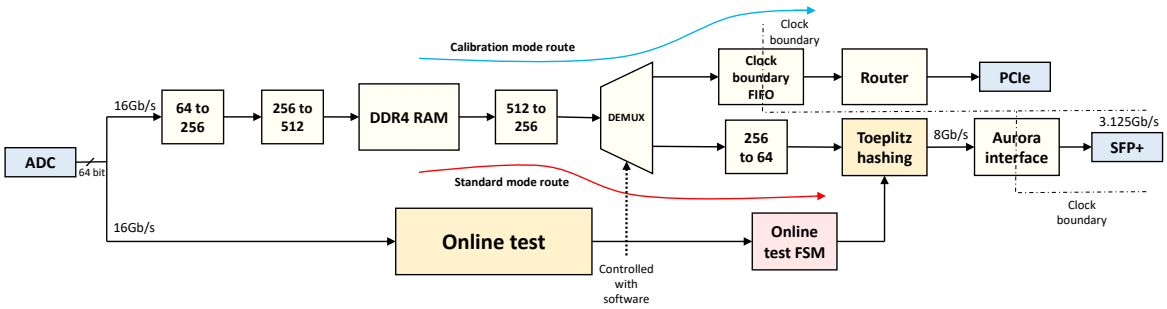


Figure 4.8: Block diagram of QRNG firmware with the online test logic implemented in FPGA. Light blue – interfaces. Darker yellow – more complex modules described in more detail in this chapter. Light pink – finite state machine for firmware control. Light yellow – other modules. The blue (red) arrow shows the data flow in calibration (standard) mode of operation.

In the standard mode of operation the online test module is activated and accepts the raw sampled data. The module is pipelined so the real-time performance is assured, however it has a significant latency. This is why at this point DDR4 RAM and the FIFOs on the standard mode route (red arrow in Fig. 4.8) are critical. They need to buffer exactly the amount of data corresponding to the online test latency, so no data is offloaded before a sufficient min-entropy is confirmed. The online test provides a test valid/fail flag for the online test finite state machine that controls the Toeplitz hashing module. If the min-entropy is above the threshold, the Toeplitz extractor performs the hashing and offloads the random numbers to SFP+ interface used by Aurora serial communication protocol. This interface is explained in more detail in the next section.

Fig. 4.9 shows the detailed block diagram of the online test module. PSD of the signal is necessary to calculate the signal variance and the conditional signal variance. The module calculates the PSD, sums up the PSD samples, and compares the values with the reference threshold values provided by the software. The input raw data is routed with a DEMUX controlled by the input FSM. Four parallel channels are necessary as the data comes in four samples on each clock cycle, and the PSD is estimated with consecutive samples in real time. For calculating the PSD we are using Bartlett's method of averaged periodograms. The PSD of a continuous function $x(t)$ is the Fourier transform of its autocorrelation function:

$$\mathcal{S}_x(f) = \mathcal{F}(x(t) * x(-t)) = X(f)X^*(f) = |X(f)|^2, \tag{4.5.1}$$

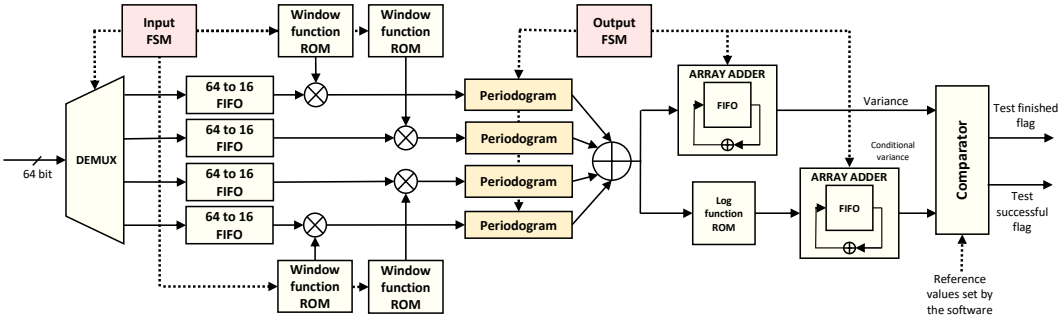


Figure 4.9: Block diagram of the online test module implemented in FPGA. Darker yellow – periodogram module described in more detail in the next figure. Light pink – finite state machines for firmware control. Light yellow – other modules. Note the window function read-only memory (ROM). User can choose a window function that is used in the periodogram averaging before programming the FPGA.

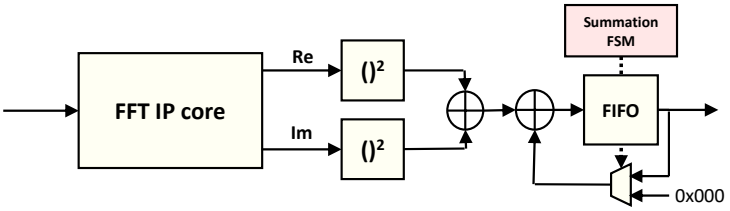


Figure 4.10: Block diagram of the periodogram module. Light pink - finite state machines for the module control, light yellow - other modules.

where $X(f) = \mathcal{F}(x(t))$. In real systems we can only estimate the PSD of a sampled signal of finite length by employing the fast Fourier transform (FFT) algorithm. Let the length of the sample array be K and T is the runtime of the experiment. The basic periodogram method would be to calculate FFT of the whole sample array and take the modulo squared to obtain the PSD. However, one way to improve the estimation of the PSD is to divide the sample array into smaller segments of length M, calculate periodograms for each and then average out. This is Bartlett's method. Welch's method on the other hand uses overlapping segments. We decided to implement Bartlett's method. Overlapping would be very challenging to implement at this data rate in FPGA. This is mostly due to the fact that there are four samples on the input at each clock cycle instead of one.

Fig. 4.10 shows the periodogram module in more detail. It consists of an FFT core (see Appendix A) with the transform length M (the segment length) that is created in the pipelined (streaming) mode in order to support real-time calculations. The core offloads real and imaginary part of a discrete Fourier transform, that are further squared and summed to get the value of modulo squared. After that a special circuit is implemented for summing up K/M segments. The length of the FIFO used for this purpose is M so it can store all the samples of the segment. We chose M to be a power of two since the possible FFT transform lengths are also a power of two. Summation FSM is responsible for counting samples and segments. When K/M segments are processed, the FSM will assert a control signal so the FIFO offloads the summed up segments. Since the number of segments K/M is also chosen to be a power of two, the averaging is done simply by truncating the least significant bits.

Focusing back to Fig. 4.9, the averaged periodograms are once more averaged between the four parallel data paths. After the final averaging, the PSD array of length M is provided to two datapaths, the first one immediately calculates the integral sum, while the second one performs a logarithm operation. This operation is realized with a simple lookup table (log function ROM) for 16-bit samples, so the capacity is $2 \times 2^{16} = 128$kB. After the log operation there is another module for integral sum calculation. The two results are proportional to the variance and conditional variance respectively and are compared with reference values provided by the software. After the comparison, the 'test finished' flag is asserted and the 'test successful' flag is asserted if the variance values are within their limits.

4.6 Aurora protocol with SATA interface

It was shown in Fig. 4.8 that the output of the Toeplitz extractor is connected to Aurora interface module. A test of our QRNG is planned in collaboration with another group. One of the requirements is the usage of SATA interface for the QRNG output. Note that only physical SATA layer is implemented, as the link-layer protocol is Aurora. Aurora is a simple point-to-point serial protocol developed by Xilinx (Appendix A.2). Aurora IP core instantiates a GTH multi-gigabit serial transceiver (Section 3.2.6). SATA interface does not exist on FMC120 board, therefore we decided to use a commercially available SFP-to-SATA (SFP2SATA) adapter. The adapter is a simple passive circuit, however with a limited bandwidth. Fig. 4.11 shows the test environment for the SATA interface. A 13 GHz oscilloscope was used for acquisition and 8b/10b decoding. The maximum data rate the SFP2SATA adapter can support is 3.125 Gb/s. This translates to 2.5 Gb/s effective useful data rate on the user side (due to 8b10b coding). The

oscilloscope was able to decode the stream without errors.

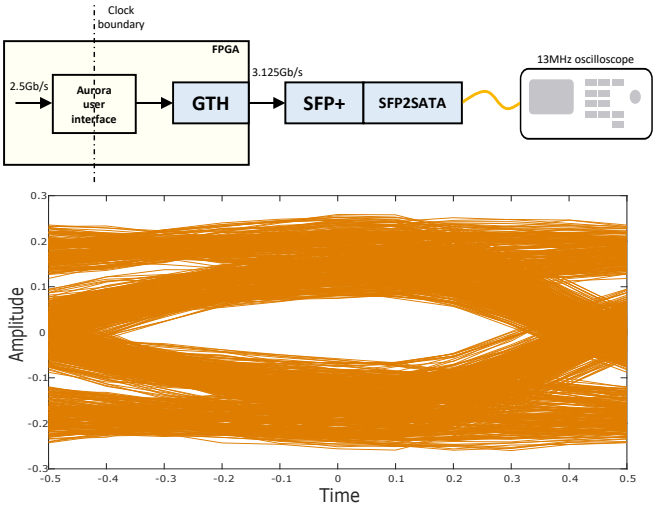


Figure 4.11: SATA output test environment and the eye diagram obtained from the measurement of the serial data.

4.7 Standalone QRNG: architecture and performance

This section provides description of the standalone QNRG setup. Standalone means the FPGA board is programmed with a full QRNG firmware with the online entropy tests and Aurora/SATA output interface. The QKD transmitter firmware contains the Toeplitz extraction module, however due to lack of additional RAM memory resources we did not implement the online test module there. Even though the focus of this thesis is the QKD transmitter, the standalone QRNG design is an important result as well, and we believe it represents a step toward more practical and secure high-speed QRNG.

Fig. 4.12 shows the experimental setup of the QRNG in the standard mode of operation. For the local oscillator a 1550 nm laser is used. The beam is split into two beams by a 3 dB (50/50) fiber coupler. The two beams are detected by two 120 μm InGaAs photo diodes. The homodyne detector output is amplified using a microwave MAR-6 amplifier, then filtered with a 400 MHz low-pass filter. The signal is further digitized with Texas instruments ADS54J60 ADC at 1 GS/s and 16-bit resolution. The ADC is a part of FMC120 (ADC/DAC) board that is connected to PC821 FPGA board (see Appendix A.3). The 16 Gbit/s stream is fed to the FPGA where the

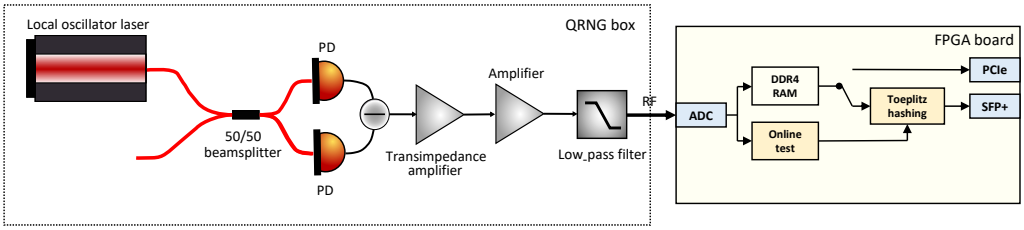


Figure 4.12: Experimental setup of the QRNG in standard mode of operation. Note in FPGA the RAM memory output is switched to the Toeplitz extractor for the standard mode of operation where the random numbers are produced and offloaded. More detailed diagram of this FPGA design is shown in Fig. 4.8.

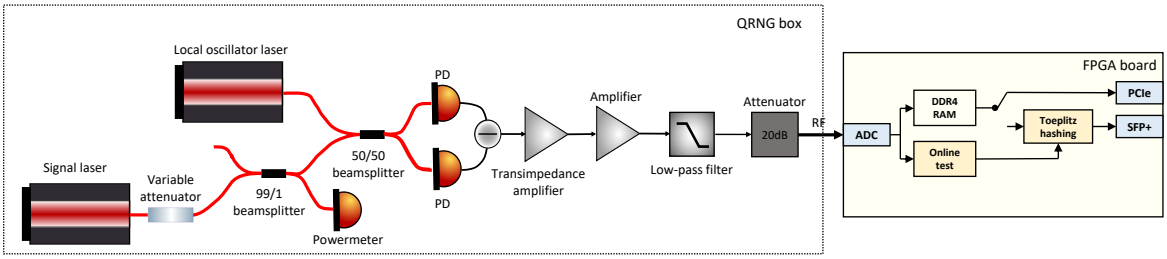


Figure 4.13: Experimental setup for the QRNG calibration mode where the transfer function of the homodyne detector is measured. Additional laser and an attenuator are introduced. The data path in the FPGA is switched to the PCIe interface.

randomness extraction and the online testing is performed.

Fig. 4.13 shows the experimental setup for the calibration step when the transfer function measurement of the homodyne detector is performed. The transfer function is measured by injecting a coherent state from the signal laser to the second port of the 3 dB fiber coupler. Note that the whole setup is in a box, so the signal laser is always present, however used only during the calibration. Behind the low-pass filter we introduced a 20 dB electrical attenuator in order to avoid the ADC saturation. This was needed since the noise amplitude is optimized for the whole ADC range, and the beat signal has a much higher amplitude. The attenuator has a flat frequency response over the bandwidth of interest so the frequency response of the system is not distorted upon removing the attenuator. The FPGA modules are programmed to offload raw ADC samples to the PCIe interface so the vacuum noise PSD estimation could be calculated in software. During the calibration stage, the signal laser is detuned from the LO so a beat signal is

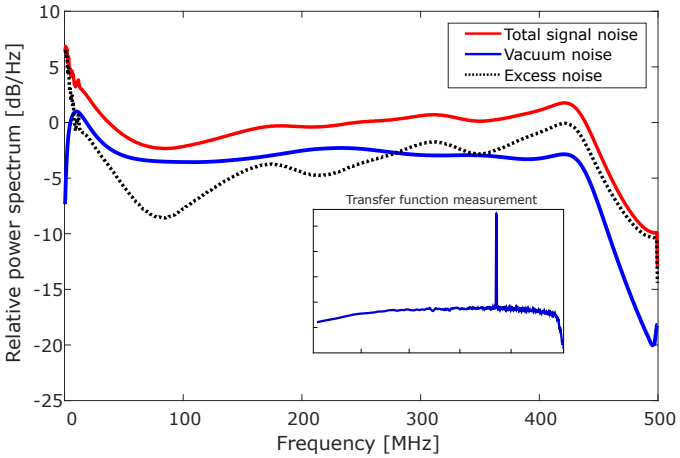


Figure 4.14: Power spectral densities obtained during the calibration stage. Blue trace shows the result of vacuum noise estimation using the measured transfer function. Red trace is the PSD of the measured signal. Dashed black trace is the PSD of the excess noise. Inset: Typical power spectrum during the transfer function measurement.

produced.

Fig. 4.14 shows the obtained vacuum noise PSD (blue trace), which is proportional to the transfer function of the detector. Using the vacuum noise PSD we are able to calculate the excess noise for the measured signal. After the calibration we performed 24-hour-long stability measurement where we monitored the variances, and hence the min-entropy. The results are shown in Table 4.4. Using the theoretical analysis from Section 4.2 we are able to estimate the best parameters for the online entropy test, the FFT length M and the number of averaged periodograms K/M. We immediately fix the FFT length to $M = 2048$. This value is not a result of a rigorous analysis, however it seems to represent a good trade-off between the complexity of the FPGA implementation and the confidence interval size (Eq. s 4.2.2 and 4.2.3). The parameter M fixes the complexity of the online entropy

Mean paramaters from 24h measurement	
Signal variance σ^2	5.20×10^6
Conditional signal variance σ_X^2	4.43×10^6
Conditional excess noise variance σ_N^2	2.04×10^6
Min-entropy [bits/symbol]	11.01

Table 4.4: Results for the variances are presented in ADC bin units, where ADC range is $[-32769, 32767]$.

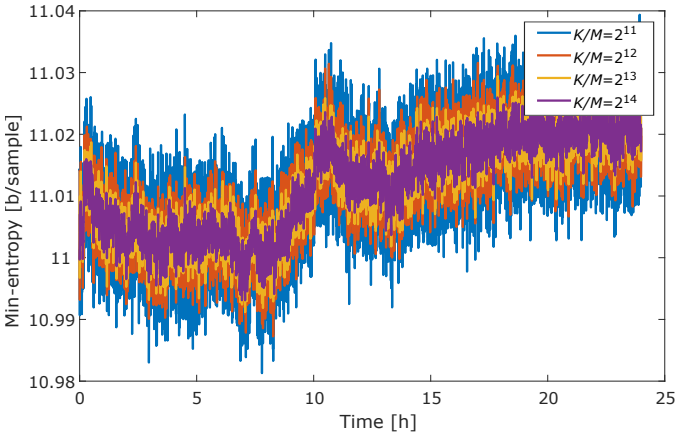


Figure 4.15: Measured min-entropy in 24h-long measurement for different values of K/M.

test module, as it depends only on M and not on K/M. This is because the summation of an array is performed in a loop using a fixed FIFO size that depends only on M. However the increase of the averaging number increases the memory requirement for the buffering of the raw samples. We set the number of averaged periodograms to $K/M = 64$. This is a small value, however it gives us the opportunity to capture the online test results often and perform the analysis for larger K/M by averaging in the software. This number should not be set too low in order to avoid a wide confidence interval that leads to a far too small trusted value of the min-entropy. Fig. 4.15 shows how averaging influences the measured min-entropy. Note the confidence intervals are not shown here, just the min-entropy values in time. Fig. 4.16 shows the worse estimate of the min-entropy for different values of the number of averaged periodograms. Finally, we define the threshold values for the minimum tolerable min-entropy. We use the same condition (Eq. 4.4.3) as before, the total security parameter in 10 years of constant QRNG operation should satisfy $\varepsilon_{\text{QRNG}} < 10^{-10}$. We set the probability of the min-entropy being outside the confidence interval $\epsilon = 10^{-10}$. The min-entropy thresholds are calculated for two different QRNG output data rates that were used at some point $R_1 = 8$ Gb/s and $R_2 = 6.6$ Gb/s.[14] Using leftover hash lemma and the 10-year condition, the calculated thresholds are $H^-_{\text{min}}[8Gb/s] = 10.08$ and $H^-_{\text{min}}[6Gb/s] = 8.50$. Fig. 4.17 shows the lower bounds of the min-entropy. For the QRNG output rate of 8 Gb/s, the security condition requires $K/M = 2^{14}$ periodograms for averaging. The lower rate has a higher margin, therefore $K/M = 2^{11}$ averaged periodograms are enough.

[14]The rate is changed by changing the parameters of the Toeplitz matrix. Lower rate gives smaller $\varepsilon_{\text{hash}}$, thus having broader security margin.

The results show the stability of the optical setup. The online entropy test is implemented successfully and it is able to support the maximum projected rate of 8 Gb/s. It is worth noting the random numbers produced in this QRNG passed the standard NIST and Diehard statistical tests.

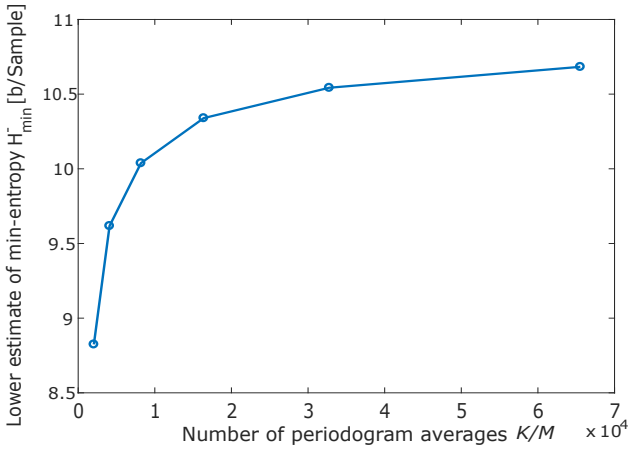


Figure 4.16: Lower bound of the min-entropy confidence interval as a function of K/M.

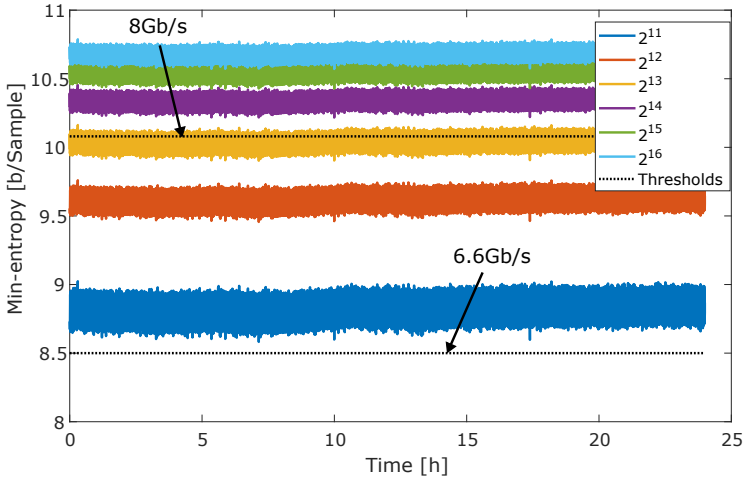


Figure 4.17: Coloured traces show lower bounds of the min-entropy confidence interval from the measured data for different values of K/M. Dotted traces are the min-entropy thresholds for two different output rates.

Chapter 5

Real-time transmitter for continuous variable quantum key distribution

In Section 3.4 it is defined what QKD transmitter means in the context of the thesis. The required signal processing functionalities of the transmitter are also presented there. This chapter describes the sampler and signal processing modules in detail.

5.1 QKD transmitter architecture

We defined a few requirements for the QKD transmitter:

- **High throughput** – the transmitter must be able to process large number of samples in real time and produce high bandwidth quantum signal.
- **Upsampling and upconversion** – flexible pulse shaping and placement of the quantum signal and the pilot in frequency domain.
- **Precision** – performing operations with minimized numerical errors.
- **Gaussian/QPSK** – easy switching between the two modulation schemes.

Fig. 5.1 shows a block diagram of the transmitter modules implemented in FPGA (see also Fig. 3.18 for better understanding of signal processing requirements). QRNG box is not shown as the same configuration is used that was already described in the previous chapter. Unprocessed homodyne RF output from QRNG is sampled with 16-bit ADC. The raw random numbers are processed with Toeplitz hashing module so the output is uniform with 8 Gb/s rate. The random numbers are then used to generate Gaussian or QPSK symbols. The user chooses which of the two is passed to

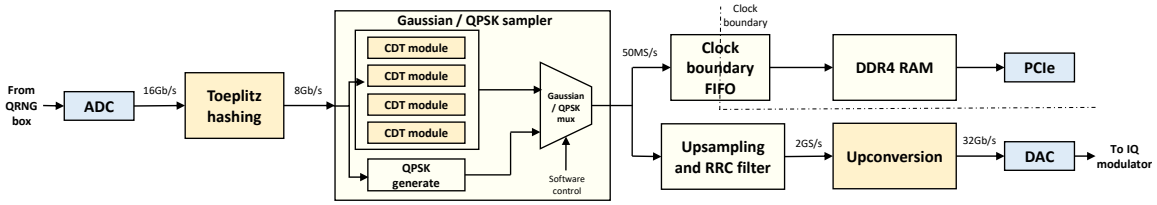


Figure 5.1: Block diagram of the QKD transmitter modules implemented in FPGA. Blue boxes – input and output interfaces. Yellow boxes – firmware modules described in more detail in the thesis. Light yellow boxes – other firmware modules. Note the data rate after the upsampling is 2 GS/s since there are two streams for each quadrature. Corresponding total DAC output rate is 32 Gb/s. Physically two DACs are used, here represented with a single box.

the output of the module with a simple software control. Output of the Gaussian/QPSK sampler is routed to two parallel datapaths. The first datapath runs through a clock boundary FIFO buffer. This buffer is used to transfer data to the host interface clock domain. Alice's symbols are this way saved on the host machine for the post-processing in later stages of the protocol.[1] DDR4 RAM is used here as a high-capacity buffer to mitigate possible problems with saving (up to 50 MS/s) data in software. The second datapath routes data to two DACs for each quadrature. Necessary DSP operations are performed along the way in the upsampling, pulse shaping and upconversion modules.

5.2 Security of quantum key distribution with realistic Gaussian source

Current state-of-the-art security proofs for continuous-variable quantum key distribution assume perfect Gaussian modulation. This means the possible values for x and y when producing a coherent state $|\alpha\rangle = |x + iy\rangle$ are continuous and unbounded. In practice there might be a coherent state centered at $|x' + iy'\rangle$, where x' and y' are close to x and y but still different. One needs to quantify this difference and analyse the impact to the final key rate according to a security proof. The real values for the quadratures are different to theoretical ones due to two reasons: a finite number of bits used in transmitter's DAC and a need to limit the input voltage levels to the IQ modulator. These problems were identified in a paper by Jouguet et

[1] At this stage Alice and Bob use authenticated classical channel to perform parameter estimation, error reconciliation and privacy amplification.

al. [127]. In this paper the low-rate supply of random numbers was also recognized as a hardware constraint. We address the hardware issue with our high-rate QRNG solution. The challenges due to imperfect Guassian modulation are described here and a theoretical analysis is given to quantify the effects on the final key rate.

Generating samples from an arbitrary probability distribution from a uniform source can be a challenging task from both technical and security point of view. Both challenges apply to QKD. A QKD transmitter requires very high rate of Gaussian sample production in FPGA, which is suitable for fixed-point arithmetics but not so much for floating-point calculations [129]. According to the state-of-the-art security proofs, Alice needs to produce coherent states where the complex amplitude α is sampled from two dimensional continuous Gaussian distribution with mean $\mu = 0$. In other words, she is generating a state $|x + iy\rangle$ where x and y are real numbers and sampled from the same distribution:

$$P_{\text{cont}}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}, \tag{5.2.1}$$

where σ is the standard deviation. The challenge here is to find a way to produce Gaussian samples so it is efficiently implementable in FPGA and satisfy the security requirements for QKD. We have been exploring a set of possible solutions to find one that gives the required security, after which it is implemented as a high-speed solution in FPGA.

There are two approaches to generating Gaussian random numbers. The first is *continuous* [133–140], where one is aiming to produce real numbers according to the standard real Gaussian distribution. The second is *discrete* [141–144], where the numbers are produced according to a discrete Gaussian distribution

$$P_{\text{dis}}(x) = \frac{1}{K} e^{-\frac{x^2}{2\sigma^2}}, \tag{5.2.2}$$

where K is normalization parameter and $x \in \mathbb{Z}$. The domain is finite and the maximum (or minimum) x is called the *tail-cut* and is commonly expressed in multiples of the standard deviation. Continuous Gaussian algorithms are used more often as they are mostly implemented in software. In our FPGA-based system, whatever the algorithm used, we need to somehow map the values of Gaussian sampler to a fixed number of bits defined by the ADC and other signal processing blocks. For example, when converting a real number to a fixed-point number with smaller number of bits, one uses a method like truncation or rounding, which effectively bins all real numbers from an interval to a single fixed-point value. Let there be an interval $[x_i, x_i+\Delta x]$ and let x_i^{fix} be a fixed-point value to which all the other values are approximated.

It follows that the probability of occurrence of this fixed-point value is:

$$P_{\mathrm{c}\to\mathrm{d}}(x_i^{\mathrm{fix}}) = \frac{1}{\sigma\sqrt{2\pi}} \int_{x_i}^{x_i+\Delta x} e^{-\frac{x^2}{2\sigma^2}} dx = \frac{1}{2}\left[\mathrm{erf}\left(\frac{x_i + \Delta x_i}{\sigma\sqrt{2}}\right) - \mathrm{erf}\left(\frac{x_i}{\sigma\sqrt{2}}\right)\right], \tag{5.2.3}$$

where erf is the error function. This expression and Eq. 5.2.2 define the candidate distributions for the hardware sampler.

Until new security proofs are potentially developed which will prove security of QKD system with discrete modulations, we need to operate with modulations 'close' to the perfect Gaussian. As usual, the relevant quantity is the trace distance between the quantum state produced with the perfect continuous Gaussian and the one produced using the approximated Gaussian. If Eve cannot distinguish the perfect thermal state and the approximated one sent by Alice, the security is not be compromised. Using the standard reasoning in composable framework [128], if the trace distance of these two states is less than $\varepsilon_{\mathrm{prep}}$, and with the standard protocol being ε-secure[2], than the protocol with the imperfect source is $(\varepsilon + \varepsilon_{\mathrm{prep}})$-secure. This approach was taken in [127]. Here the Gaussian sampler and the rest of the protocol are seen as two black boxes, each with their corresponding security parameter, so the sum of the two presents the upper bound to Eve's advantage.

The state produced according to the perfect Gaussian is thermal state:

$$\hat{\rho_G} = \frac{1}{\pi\bar{n}} \int d^2\alpha\, e^{-\frac{|\alpha|^2}{\bar{n}}} |\alpha\rangle \langle\alpha| , \tag{5.2.4}$$

where $\bar{n} = \langle n\rangle$ is the mean photon number. Let the (one-dimensional) distribution that is implemented in practice in the transmitter be P. The exact domain of the distribution is finite (therefore countable). The size of domain depends on the output number of bits of the sampler. For a complex number α, we define $P(\alpha) = P(\mathrm{Re}(\alpha))P(\mathrm{Im}(\alpha))$ where the real and imaginary part of the complex number are from the distribution domain. Let the quantum state produced by this distribution be:

$$\hat{\rho} = \sum_{\alpha} P(\alpha) |\alpha\rangle \langle\alpha| . \tag{5.2.5}$$

The trace distance between the two quantum states should satisfy

$$\|\hat{\rho} - \hat{\rho}_G\|_1 \leq 2\epsilon_{\mathrm{prep}} \tag{5.2.6}$$

so the two average quantum states are indistinguishable up to a probability smaller than $\varepsilon_{\mathrm{prep}}$. We need to be able to estimate $\varepsilon_{\mathrm{prep}}$ for the implemented

[2]Meaning the probability of the adversary discriminating between the protocol and a perfect one is ε.

distribution P. In order to do that we consider a computable upper bound on the trace distance [130, 131]:

$$\|\hat{\rho} - \hat{\rho}_G\|_1 \leq \sqrt{\mathrm{Tr}[\hat{\rho}\hat{\rho}_G^{-1/2}\hat{\rho}\hat{\rho}_G^{-1/2}] - 1} \tag{5.2.7}$$

$\hat{\rho}_G$ is a thermal state, defined above as a mixture of coherent states. In Fock space the thermal state $\hat{\rho}_G$ is:

$$\hat{\rho}_G = \frac{1}{\bar{n}+1}\sum_{n=0}^{\infty}\left(\frac{\bar{n}}{\bar{n}+1}\right)^n |n\rangle\langle n| \tag{5.2.8}$$

It follows that

$$\hat{\rho}_G^{-1/2} = \sqrt{\bar{n}+1}\sum_{n=0}^{\infty}\left(\frac{\bar{n}+1}{\bar{n}}\right)^{n/2} |n\rangle\langle n| \tag{5.2.9}$$

Using

$$\langle\alpha|n\rangle = e^{-|\alpha|^2/2}\frac{\bar{\alpha}^n}{\sqrt{n!}} \tag{5.2.10}$$

we obtain

$$\langle\alpha|n\rangle\langle n|\beta\rangle = e^{-\frac{|\alpha|^2+|\beta|^2}{2}}\frac{\bar{\alpha}^n\beta^n}{n!} \tag{5.2.11}$$

where $\bar{\alpha}$ and $\bar{\beta}$ are the mean photon numbers for some states $|\alpha\rangle$ and $|\beta\rangle$ respectively. Let P_α and P_β be short notation for $P(\alpha)$ and $P(\beta)$ in the following derivation. The expressions are combined so we get

$$\begin{aligned}
\mathrm{Tr}\left[\hat{\rho}\hat{\rho}_G^{-1/2}\hat{\rho}\hat{\rho}_G^{-1/2}\right] &= \sum_{\alpha,\beta} P_\alpha P_\beta \mathrm{Tr}\left[|\alpha\rangle\langle\alpha|\hat{\rho}_G^{-1/2}|\beta\rangle\langle\beta|\hat{\rho}_G^{-1/2}\right] \\
&= \sum_{\alpha,\beta} P_\alpha P_\beta \langle\alpha|\hat{\rho}_G^{-1/2}|\beta\rangle\langle\beta|\hat{\rho}_G^{-1/2}|\alpha\rangle \\
&= \sum_{\alpha,\beta} P_\alpha P_\beta |\langle\alpha|\hat{\rho}_G^{-1/2}|\beta\rangle|^2 \\
&= (\bar{n}+1)\sum_{\alpha,\beta} P_\alpha P_\beta \left|\sum_{n=0}^{\infty}\left(\frac{\bar{n}+1}{\bar{n}}\right)^{n/2}\langle\alpha|n\rangle\langle n|\beta\rangle\right|^2 \\
&= (\bar{n}+1)\sum_{\alpha,\beta} P_\alpha P_\beta e^{-|\alpha|^2-|\beta|^2}\left|\sum_{n=0}^{\infty}\left(\frac{\bar{n}+1}{\bar{n}}\right)^{n/2}\frac{\bar{\alpha}^n\beta^n}{n!}\right|^2 \\
&= (\bar{n}+1)\sum_{\alpha,\beta} P_\alpha P_\beta e^{-|\alpha|^2-|\beta|^2}\left|e^{\bar{\alpha}\beta\sqrt{\frac{\bar{n}+1}{\bar{n}}}}\right|^2 \\
&= (\bar{n}+1)\sum_{\alpha,\beta} P_\alpha P_\beta e^{-|\alpha|^2-|\beta|^2}e^{(\bar{\alpha}\beta+\alpha\bar{\beta})\sqrt{\frac{\bar{n}+1}{\bar{n}}}}.
\end{aligned} \tag{5.2.12}$$

Representing coherent state eigenvalues as a complex sum of quadrature values

$$\alpha = q + ip, \tag{5.2.13}$$

$$\beta = u + iv, \tag{5.2.14}$$

we obtain

$$|\alpha|^2 + |\beta|^2 = q^2 + p^2 + u^2 + v^2, \tag{5.2.15}$$

$$\bar{\alpha}\beta + \alpha\bar{\beta} = 2(qu + pv). \tag{5.2.16}$$

Using these expressions we are able to calculate the upper bound for the trace distance for a descrete distribution. In case of perfect discrete Gaussian we have

$$P_\alpha = K^{-2} e^{-\frac{q^2}{\bar{n}}} e^{-\frac{p^2}{\bar{n}}}, \tag{5.2.17}$$

$$P_\beta = K^{-2} e^{-\frac{u^2}{\bar{n}}} e^{-\frac{v^2}{\bar{n}}}, \tag{5.2.18}$$

where K is the normalization factor and q, p, u and v are unitless numbers that take value on the same set of points. They are a consequence of producing integer numbers in FPGA which are then mapped to voltage levels in the ADC and in turn mapped to discrete values of quadratures of the electromagnetic field.

After getting the expression for calculating $\varepsilon_{\text{prep}}$ we are able to quantify the influence of imperfect Gaussian distribution to the key rate. This is done by following the approach from [130]. The new method presents an improvement to the argument from [127] as the imperfection of the Gaussian sampler is directly translated to the key rate formula. We bound the Holevo information (Eve's information about Bob's state[3]) for the protocol with imperfect state creation

$$\chi(B:E)_{\hat{\rho}} \leq \chi(B:E)_{\hat{\rho}_G} + f(\varepsilon_{\text{prep}}, W), \tag{5.2.19}$$

where W is an upper bound on the energy of the adversary's share of the quantum state, and the function f is defined as [132]

$$f(\varepsilon, W) \equiv 2\varepsilon\left(2t + r_\varepsilon(t)\right) F\left(\frac{W}{\varepsilon t}\right) + 2g\left(\varepsilon r_\varepsilon(t)\right) + 4h_2\left(\varepsilon t\right), \tag{5.2.20}$$

for any $t \in \left(0, \frac{1}{2\epsilon}\right]$ and

$$r_\varepsilon(t) \equiv \frac{1 + t/2}{1 - \varepsilon t}, \tag{5.2.21}$$

$$g(x) \equiv (x+1)\log(x+1) - x\log x, \tag{5.2.22}$$

$$h_2(x) \equiv -x\log x - (1-x)\log(1-x). \tag{5.2.23}$$

[3]Reverse reconciliation is assumed.

The function $F\left(\frac{W}{\varepsilon t}\right)$ is the entropy of a Gaussian state with mean energy $\frac{W}{\varepsilon t}$. Note that this is the entropy of the state held by the eavesdropper, therefore to estimate it we need to model the Hamiltonian associated to the eavesdropper's share of the quantum system. Here we do that by assuming a Gaussian attack. This is the attack model presented in Section 3.1 (Fig. 3.2). Within this model the eavesdropper holds two bosonic modes with mean photon number w and $(1-\eta_{\text{chan}})\bar{n}+\eta_{\text{chan}}w$. Within the model of a Gaussian attack we identify

$$F\left(\frac{W}{\varepsilon t}\right)=g\left(c\,\frac{w}{\varepsilon t}\right)+g\left(c\,\frac{(1-\eta_{\text{chan}})\bar{n}+\eta_{\text{chan}}w}{\varepsilon t}\right), \tag{5.2.24}$$

where the factor c has been introduced to account for the fact that we have estimated the channel parameters η_{chan} and w using the protocol with a discrete and bounded coherent state distribution, but the energy bound needs to apply also to the ideal (but unobserved) protocol with the Gaussian input distribution. This caveat is discussed in more detail in Ref. [130], where it is shown that the bound is only mildly dependent of c. Here we work with $c=10$. Finally we note that, given the parameters c, $\bar{n}$, w, η_{chan} and $\varepsilon_{\text{prep}}$, the value of $t\in\left(0,\frac{1}{2\varepsilon}\right]$ can be optimized to obtain the tightest bound.

5.3 Generating discrete Gaussian random numbers

In order to compare the two distributions defined by equations 5.2.2 and 5.2.3, that are the outputs of two different classes of algorithms, we wrote a program that calculates $\varepsilon_{\text{prep}}$. The simulation calculates the trace distance for the set of four parameters – the number of output bits n_x (therefore the number of elements in the domain is $2^{n_x}-1$), the standard deviation of the distribution σ, the tail-cut Σ, and the mean photon number $\bar{n}$ of the quantum state. First three parameters define the distribution, while the last one can be set optically using a variable attenuator in Alice's box. Fig. 5.2 shows an example of assigning values to the elements of a distribution. Fig. 5.3 shows the calculated trace distance for the discrete Gaussian distribution defined by Eq. 5.2.2. It turns out $\varepsilon_{\text{prep}}$ converges much faster to zero with this distribution than with the one defined in Eq. 5.2.3. Fig. 5.4 shows higher values of the trace distance. We just show the result for 8 bits as the other produce even higher $\varepsilon_{\text{prep}}$.

Particular application and required security level determine the parameters of a distribution (σ, n_b, Σ) and hence the appropriate algorithm. High-rate and efficient algorithms for generating high-quality Gaussian random numbers have been developed mostly for lattice-based cryptography which is a subtype of post-quantum cryptography [145, 146, 149]. To our knowledge,

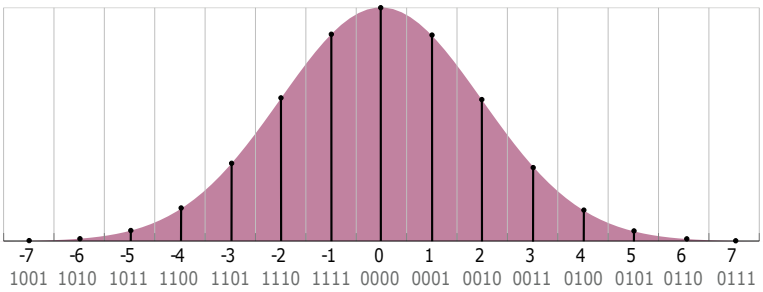


Figure 5.2: Black dots represent the values of a discrete Gaussian. Below are numbers assigned to each probability. In this case the distribution is symmetric with a tail-cut value of 7 (which is approximately $\Sigma = 3\sigma$ in this example). Numbers in the bottom are two's complement representation of the elements.

the work in this thesis is the first implementation of a high-quality hardware Gaussian sampler for QKD applications. A few notable algorithms used for discrete Gaussian sampling, some of which were implemented in hardware are:

- **Knuth-Yao sampling.** Tree-based algorithm that uses a small number of bits close to the entropy of a distribution in question, therefore very useful when the randomness is scarce. It was implemented in FPGA before [141, 147], however only for small values of σ.
- **Discrete ziggurat sampling.** This is a discrete version [144] of the original rejection sampling algorithm proposed by Marsaglia and Tsang [148]. This algorithm in general could be used for distributions with larger σ [149], however it is primarily targeted to be efficient for processor-based systems. This is due to the fact that computationally intensive calculations are done relatively rarely (a few times per hundred samples). In contrast to CPU-based systems, all hardware functions must be implemented in FPGA beforehand so it takes additional resources.
- **Cumulative distribution table (CDT) sampling**. This is a straightforward approach to sampling any probability distribution and it was implemented in software [150] and hardware [142] before. Since this algorithm is able to produce Gaussian samples with relatively large standard deviation (for example $\sigma > 16$, compared to $\sigma = 3.33$ implemented with Knuth-Yao algorithm) and the required precomputed tables are not too large for FPGA, it was the obvious choice for our system.

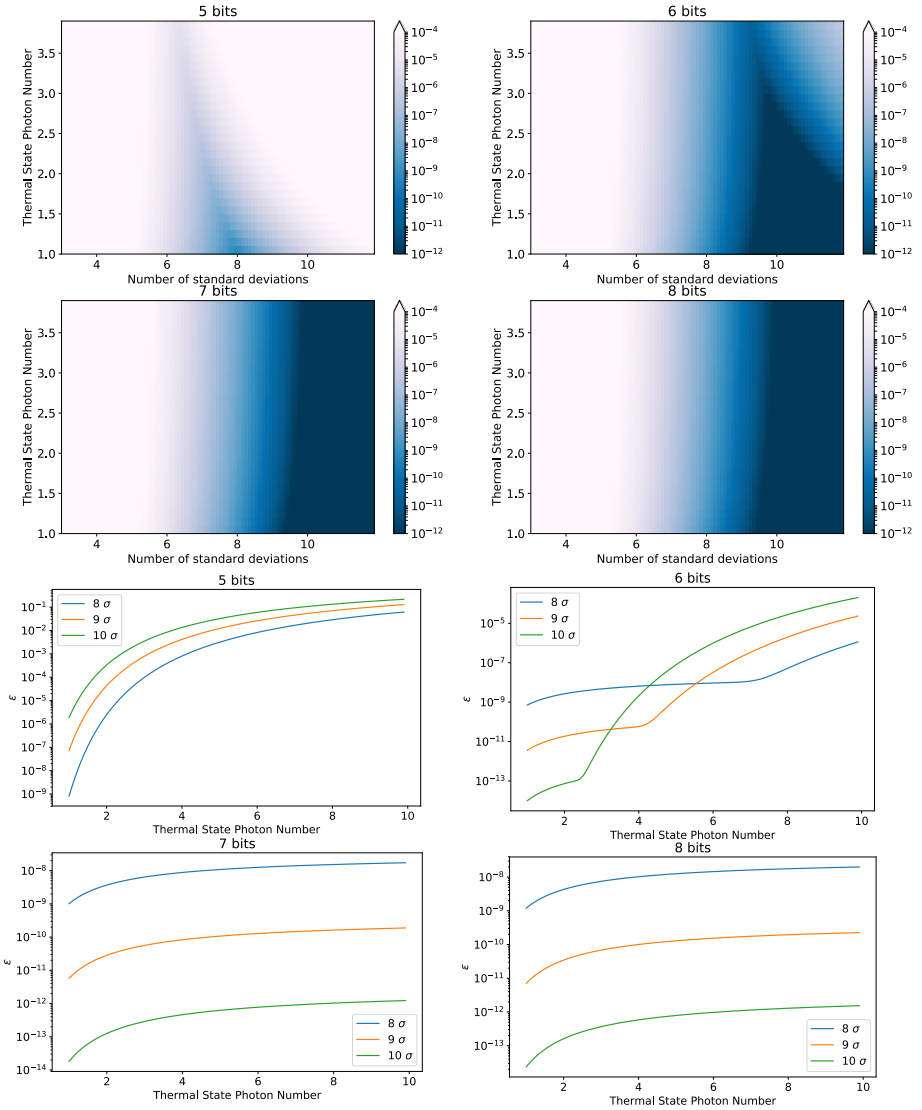


Figure 5.3: Top graphs depict the values for $\varepsilon_{\text{prep}}$ as a function of the tail-cut (x-axis) and the mean photon number (y-axis). For larger number of output bits n_x there is a weak dependancy on the mean photon number. Bottom graphs show the trace distance for three particular tail-cut values. There is an interesting effect where the larger tail-cut tends to become better as the signal power goes up.

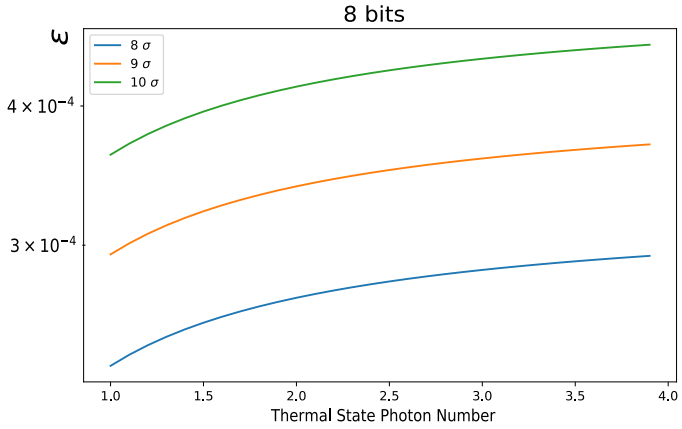


Figure 5.4: Calculated $\varepsilon_{\text{prep}}$ for the binned continuous Gaussian distribution.

5.4 CDT algorithm for Gaussian sampling

CDT sampling algorithm stores a precomputed table of calculated values of the cumulative distribution function (CDF) of a probability distribution. CDF is a monotone function, therefore the table values are sorted $0 = T[0] \leq T[1] \leq ... \leq T[X+1] = 1$, where X is the value of the largest sample and $P(x) = T[x+1] - T[x]$. Table entries $T[i]$ are real numbers represented by a certain number of bits n_b. A real number u is drawn randomly from $[0, 1)$. The algorithm performs a table search in order to find an integer x for which it holds $T[x] \leq u < T[x+1]$, and outputs x.

In general there are a few of challenges that must be addressed for an efficient implementation – table size and the average time of a single table search. Table size is simply $n_b \log_2 X$ bits where both n_b and X are lower bounded by required level of security. Note that we will be using one-sided Gaussian distribution in order to cut the table size in half for the same number of output bits. One-sided distribution can be easily modified to the standard Gaussian by using an additional random bit to determine the sign, or in the case of $x = 0$ to determine whether the sample should be discarded. This way the zero sample is not taken twice into account. The average time for a binary table search is $\sim \log_2 X$ comparisons. However the actual number of comparisons could take more than $\log_2 X$ clock cycles due to additional operations in FPGA.

Our CDT sampling solution for FPGA follows the idea from Poppelmann et. al. [142]. Let's assume the input to the algorithm is a random number v represented by n_v bits. These bits can be interpreted as the most significant bits (MSB) of number u, taken from a uniform distribution, that is defined

by the same n_b number of bits as the probabilities in the CDT. Now one can create a guide table, i.e. a table of search intervals. For each v this table stores the interval $[i_{\max}(v), i_{\min}(v)]$ where the number with most significant bits equal to v is to be found. With the reduction of the search interval in the first step, the total search time is reduced. By increasing the number of bits n_v, we reduce the interval lengths in the guide table on average, but on the other hand this increases the complexity of the comparator's combinatorial logic in FPGA and increases the size of the guide table. Note that in a majority of cases, even if n_v is relatively low, it will be enough to finish the search, so we do not need all n_b bits of u. The average randomness consumption depends on several factors and it should be calculated after all the system parameters are defined.

5.5 FPGA implementation of the Gaussian sampler

Here we state the exact values of all important parameters. Most of them can be slightly tweaked without the need for a major algorithm implementation change. The number of output bits is $n_x = \log_2 X + 1 = 8$. The algorithm itself outputs integers in $[0, 127]$, which are then mapped to $[-127, 127]$ as a two's complement signed integer with an additional bit. It was shown in Section 5.3 that 8 bits provide a high level of security when paired with sufficiently large standard deviation of the implemented Gaussian. In general, X does not need to be a power of two, however it is good that we use the whole dynamic range when upconverted samples are later on passed to the DAC. The number of bits used to represent the probabilities is $n_b = 96$. Note that we represent real numbers, using fixed point representation. In the case of the table entries, those 96 bits are all behind the implicit decimal point (probabilities are smaller than 1). Number of bits used to access the guide table is $n_v = 8$. This is at the same time the width of the uniform input to the algorithm. The standard deviation of the distribution is $\sigma = 32$ and hence the tail-cut value is $\Sigma = 8\sigma$. In Section 5.3 we showed that the security parameter attributed to non-perfect Gaussian modulation, with the parameters defined here, lies in the interval $\varepsilon_{\text{prep}} = [10^{-9}, 10^{-10}]$. This might not be low enough if we follow composability argument. However, the correction to Eve's Holevo information is still low enough for any measurable impact to the key rate. We decided to use the 8σ value since increasing the tail-cut value, assuming constant number of bits, decreases the overall power of the quantum signal compared to the pilot.[4]

[4]Power of the optical signal can be controlled with the variable attenuator, however it attenuates both the quantum signal and the pilot at the same time. During the experiments we decided to keep a certain level of the quantum signal due to the fact that the Bob's receiver (and the whole system in general), was not fully developed.

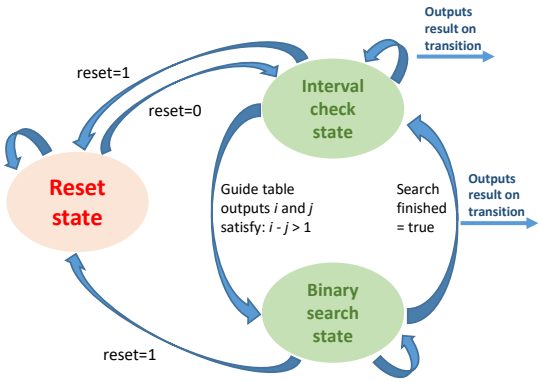


Figure 5.5: Simplified state machine of CDT algorithm in FPGA

Due to abundance of memory resources in our Kintex UlstraScale FPGA, we are storing full CDT with 96 bits for each of the samples, in total 96×128 bits = 1.5kB. The table is defined as follows: $1 = T[0] \leq T[1] \leq ... \leq T[X+1] = 0$, where $P(i) = T[i] - T[i+1]$. Guide table stores two 7-bit integers per input word v, in total $2 \times 7 \times 256$ bits = 448B. The algorithm takes single 8-bit word v_k (where k is an index to count the input sequence) fed from QRNG randomness extractor module. This 8-bit word is used for the guide table address entry. Now the guide table output interval is checked – if $i_{\max}(v_k) - i_{\min}(v_k) > 1$ then the binary search state machine is started, otherwise the search is over and $i_{\min}$ in passed to the output. The binary search state machine works as follows. The middle index is calculated as $i = [(i_{\max}(v_k) - i_{\min}(v_k))/2]$. Let j be an index which maps 8-bit words in a 96-bit number, where $j = 0$ designates MSBs and $j = 11$ corresponds to LSBs. Another random 8-bit word v_{k+1} is drawn from the input and compared to the second byte of CDT entry $T[i, j]$. Depending on the result of comparison, $i_{\min}$ and $i_{\max}$ are updated and j is reset to 1, or in the case where $T[i, j] = v_{k+1}$, another random word v_{k+2} is fetched and j is increased by 1 for further comparison. The search is performed until $i_{\max} - i_{\min} < 2$. Implementation experiments in FPGA showed that the comparison between $T[i, j]$ and v_k and subsequent $i_{\min}$ and $i_{\max}$ updates cannot be done in a single clock cycle due to built up logic delay caused by combinatorial logic. Therefore a single step of the binary search state machine was done in three clock cycles.

Algorithm 1 CDF table search with the guide table

initialization: $i_{\max}, i_{\min}, i, j \leftarrow 0$
$v_0 \leftarrow$ uniform QRNG
$i_{\max}, i_{\min}, i \leftarrow$ guide table $[v_0]$
if $i_{\max} - i_{\min} = 1$ **then**
  return $i_{\min}$
else
  $v_1 \leftarrow$ uniform QRNG
  while $i_{\max} - i_{\min} > 1$ **do**
    if $v_{j+1} > T[i, j]$ **then**
      $i_{\max} \leftarrow i, \quad i \leftarrow (i_{\min} + i)/2$
    else if $v_{j+1} < T[i, j]$ **then**
      $i_{\min} \leftarrow i, \quad i \leftarrow (i_{\max} + i)/2$
    else
      $j \leftarrow j + 1$
      $v_{j+1} \leftarrow$ uniform QRNG
    end if
  end while
  return $i_{\min}$
end if

Performance of the Gaussian sampler

In this subsection we provide performance results of implemented CDT algorithm in Kintex UltraScale FPGA. Due to CDT algorithm having conditional commands that depend on the uniformly distributed random input, there is no deterministic value for the output data rate. In fact, if the input is not random it is possible that the algorithm is constantly in *Interval check state* of the finite state machine shown in Fig. 5.5, where it would act as a pipeline so the rate would be the same as of the input. On the other hand there could be an input stream which may cause long search times, therefore reducing the rate drastically as the algorithm would spend much time in *Binary search state*. Obviously, we are interested in an average output rate when the input are uniform random numbers from QRNG. The rate is most easily calculated by measuring the average number of FPGA clock cycles per one valid output. It turns out a valid output sample comes every 2.7 clock cycles. This has been confirmed both in a VHDL simulation using pseudo-random input from Matlab and using debugging probes to capture real signals in FPGA. The stream was observed through 500,000 clock cycles. Number of occurrences of every 8-bit integer number is shown in Fig. 5.6. In principle the x-axis should show all 8-bit signed integers from -128 to 127, however we never observed an integer larger than 85 in its absolute value due to extremely low probability for the tail region integers to occur.

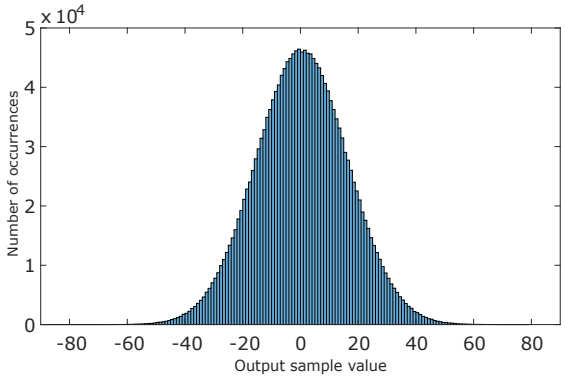


Figure 5.6: Histogram of Gaussian sampler output.

Parameters of the discrete Gaussian sampler	
Output width [bits]	8
Standard deviation σ	32
Tail-cut $\Sigma[\sigma$ units]	8
Probability precision n_b [bits]	96
Average entropy consumption per output sample H_{cons} [bits]	21
Average output rate (for 8 Gb/s input) [MS/s]	370

Table 5.1

Assuming an input rate of 8 Gb/s from the QRNG, the Gaussian sampler output rate is 370 MS/s (2.96 Gb/s) on average.

The entropy consumption H_{cons} of the algorithm is the average number of uniform bits spent for one output Gaussian sample. Since the algorithm takes an 8-bit uniform sample on each clock cycle and produces an 8-bit Gaussian sample every 2.7 clock cycles, the average entropy consumption is $8 \times 2.7 \approx 21$ bits per sample. To put this number in a context one needs to calculate the entropy of the discrete Gaussian distribution. Using Eq. 2.2.1 we obtain the entropy of our discrete Gaussian implementation is $H(P) = 7.0462$. The entropy consumption is indeed much higher then in Knuth-Yao algorithm where it is at most two bits larger than the Gaussian distribution entropy [149]. Despite this disadvantage, CDT algorithm presents the best trade-offs between the parameters for our QKD application.

After confirming the true randomness of the QRNG, we were able to do

Statistical test			
Rate [MS/s]	**χ^2 test p-value**	**Significance level**	**Test result**
367	0.5227	5%	Passed
FPGA implementation resource utilization			
FF	**LUT**	**BRAM**	**FIFO**
1745	1712	8	14

Table 5.2: Summarized properties of the Gaussian sampler used in the QKD transmitter. The implementation includes four CDT modules working in parallel. FF: flip-flop, LUT: lookup table, BRAM: block RAM, FIFO: first-in first-out buffer.

statistical tests on the CDT module Gaussian output. First, we confirmed the functional validity of the algorithm by performing cumulative distribution table search in Matlab for the same test input uniform samples. The outputs were identical as in VHDL simulation. Finally, we confirmed the CDT algorithm output is close enough to the discrete Gaussian distribution using Pearson's chi-squared test [151]. We use expression 'close enough' as chi-squared test does not tell if a population *is* sampled from a particular distribution. The statistical test instead gives a probability of finding the observed, or more extreme, results when the null hypothesis (H_0) is true. This probability is called *p-value* of the test. The test also defines the significance level α which is used to judge whether a population is significantly different ($p < \alpha$) from what we expect it to be under the null hypothesis (test failed), or the difference is not significant ($p > \alpha$) (test passed). Typical values for α are 5% and 1%. We use the standard value of $\alpha = 5\%$. These results are summarized in Table 5.2.

The CDT algorithm is convenient for FPGA chips where DSP resources are expensive. The design does not implement arithmetic operations such as multiplications, therefore no single DSP primitive is used. Instead, the module uses memory resources such as BRAMs and FIFOs. Table 5.2 summarizes the resource utilization. The results are given for the whole Gaussian sampler, i.e. for the four CDT algorithm modules working in parallel. We implemented four parallel modules since, on average, four 8-bit uniform samples are produced after the Toeplitz hashing module on each clock cycle.

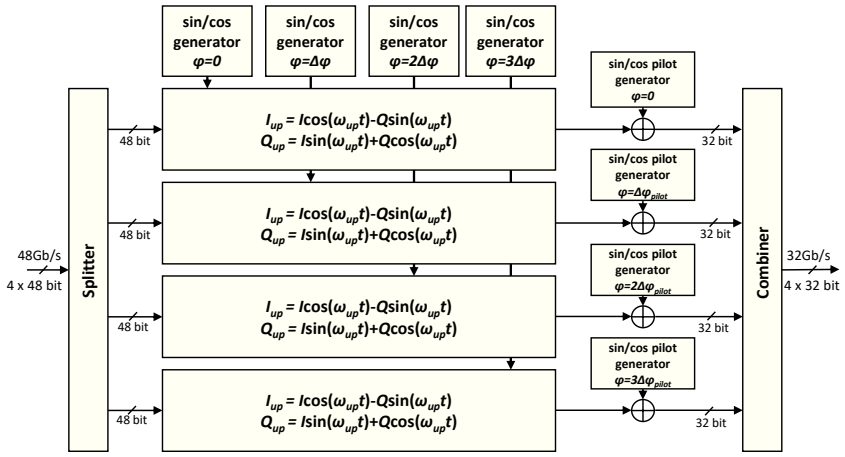


Figure 5.7: Block diagram of the upconversion module implemented in FPGA. There are four parallel submodules performing upconversion and pilot multiplexing. Note that 48-bit words are consisted of both I and Q quadraure samples. Word width is a consequence of bit growth in the upsampling module. The output is scaled to 16-bit words per quadrature, hence 32-bit words at the output.

5.6 FPGA implementation of the upsampling and upconversion modules

Looking again at Fig. 5.1, one can see the Gaussian distributed samples are fed to the upsampling and pulse-shaping module. These functions are performed by the FIR filter IP core (Appendix A.2). The module implements a root-raised cosine (RRC) filter. At this point the symbol rate R_s and the bandwidth of the quantum signal f_q are determined (see Fig. 3.18). The RRC filer performs the pulse-shaping so the total bandwidth of the quantum signal is $f_q = (1 + \beta_{\mathrm{RRC}})R_s$ where β_{RRC} is the roll-off factor.[5] Regardless of the symbol rate, the signal is upsampled to the DAC sample rate of $f_{\mathrm{DAC}} = 1$ GS/s so the upsampling factor is f_{DAC}/R_s. The module is capable of pipelined processing of multiple samples per clock cycle, therefore enabling output rates higher than 250 MS/s, since the system frequency is $f_{\mathrm{system}} = 250$ MHz. On top of this, the module supports multiple data paths. We use two paths for the two quadratures. This way the FPGA resources are saved as the upsampling and filter parameters are the same for both data paths and are written only once to the FPGA distributed memory. The parameters of the RRC filter are chosen as a result of a trade-off between the precision and complexity. Higher precision requires more resources. There

[5] Baseband bandwidth is $f_q/2$.

are three parameters that influence precision and complexity: filter span, coefficient width and output rounding mode. Filter span is the number of FIR delay registers divided by the upsampling factor. The filter coefficient width is chosen to be 16 bits. Since the filter operation consists of additions and multiplications, there is a bit growth. For our implementation, the output is 24 bits wide. Even though the IP core can implement rounding modes, we decided to let the full precision output so we can manipulate the word width and tune the precision. The module controls the rate of its input at the constant rate with AXI stream protocol (Appendix A.1).

Fig. 5.7 shows a block diagram of the upconversion/pilot multiplexing module. The data is passed to the upconversion module input from the upsampling module at 4×12 Gb/s (as a consequence of the upsampling and the bit growth). For the purpose of upconverting the signal to f_{up}, four parallel oscillators are implemented using DDS compiler IP core (Appendix A.2). DDS compiler provides sine and cosine (sin/cos) samples at the FPGA system frequency. The frequency and the phase offset of the sin/cos waveforms can be tuned. The upconversion has to be performed in complex domain for the two quadratures. Let the in-phase quadrature samples be designated with $I(t)$ and the quadrature samples with $Q(t)$. The upconversion is calculated as

$$I_{up}(t) = I(t)\cos(\omega_{up}t) - Q(t)\sin(\omega_{up}t), \tag{5.6.1}$$

$$Q_{up}(t) = I(t)\sin(\omega_{up}t) + Q(t)\cos(\omega_{up}t), \tag{5.6.2}$$

where $\omega_{up} = 2\pi f_{up}$. This way the signal is prepared for single sideband modulation as one quadrature is Hilbert transform of the other (Section 3.2.3). In order to perform the upconversion operation of four samples in one clock cycle, the sin/cos generators are phase shifted by $\varphi = n\Delta\varphi = 2\pi f_{\text{pilot}}/f_{\text{DAC}}$, where n goes from 0 to 3.

After the upconversion, a pilot tone is frequency multiplexed to the signal. Another set of four parallel sin/cos generators are employed. The pilot samples are multiplxed by a simple addition operation. At the output, the samples are rescaled to 16 bit per quadrature. Quadrature data streams are passed to two separate 16-bit DACs.

Table 5.3 shows the FPGA resource utilization by bot upsampling and upconversion module. Note the significant number of DSP cores used for multiplication and addition operations. This is still a very small number as only 2.4% of the total number of DSPs is utilized.

Upsampling and upconversion FPGA implementation resource utilization				
FF	**LUT**	**BRAM**	**FIFO**	**DSP**
2660	244	60	0	136

Table 5.3

5.7 Performance of the QKD transmitter

This section presents performance results of the QKD transmitter. As the transmitter, the CV-QKD optical setup (described in Section 3.4) and the receiver have been developed in parallel, a commercial arbitrary waveform generator (AWG) has been used initially for the purpose of generating symbols. It is shown that our FPGA-based transmitter and the AWG have comparable performances in terms of the parameters that quantify the quality of the output RF signal. We argue the results obtained during long measurements (large number of symbols) with the AWG justify the chosen transmitter parameters and open the door for longer measurements using the FPGA-based transmitter in the future. Note that the FPGA-based transmitter is capable of producing symbols and transmitting indefinitely, where the limiting factor are the host computer memory and the receiver acquisition memory.

Here we evaluate the use of both Guassian and QPSK modulation schemes. QPSK is convenient for testing purposes as the symbol synchronization at the receiver is easier due to the four distinct points of the QPSK constellation, which also facilitates easier identification of noise effects when recovering the constellation map. We perform three different measurements, each based on a different transmitter-receiver connection. The first one is *electrical back-to-back* (B2B) where RF outputs of the transmitter are directly connected to the digital signal oscilloscope (DSO) for acquisition. The second type is *optical* B2B where the RF outputs are connected to the IQ modulator, however the quantum channel between Alice and Bob is a short fibre. Finally, we also perform an optical measurement with a 20 kilometer-long fibre spool between Alice and Bob.

In Section 3.2.3 we introduced EVM as a figure of merit for quantifying the quality of received QPSK symbols. We use it in electrical B2B configuration. Similarly, for a Gaussian-modulated electrical B2B setup the relevant quantity is the *minimum variance*. We define the minimum variance as

$$V_{\min} = \min_g \langle (A - gB)^2 \rangle = \langle A^2 \rangle - \frac{\langle AB \rangle}{\langle B^2 \rangle}, \tag{5.7.1}$$

where A and B are Alice's symbols and Bob's received symbols respectively, g is a gain parameter over which we minimize the expression in order to get

QKD transmitter parameters	
Symbol rate R_s [MBd]	50
RRC filter roll-off β_{RRC}	0.4
Bandwidth of the upconverted quantum signal f_q [MHz]	70
Upconversion frequency f_{up} [MHz]	62.5
Pilot frequency f_{pilot} [MHz]	125

Table 5.4: We kept these parameters constant during the evaluation measurement of the transmitter. The main reason is that a slight change of some of the parameters did not cause noticeable changes in the excess noise (at least during the development stage where the noise is still relatively large), which is the most important figure of merit. The quantum signal and the pilot are kept within a reference bandwidth of 200 MHz. As far as the filter roll-off is concerned, EVM tends to drop as β_{RRC} increases, however the bandwidth of the quantum signal increases as well thus increasing chances for unwanted overlaps in the frequency domain. During the initial tests we fixed the value of $\beta_{\mathrm{RRC}} = 0.4$ as the optimal one.

the right side of the equation, and $\langle AB \rangle$ is the covariance between the two symbol strings. Note that EVM and minimum variance are convenient for electrical B2B since there is no shot noise involved. Contrary to that, the shot noise is inevitably present in the optical measurements, thus causing the EVM and minimum variance values to be large by default.

Table 5.4 shows the chosen parameters of the transmitter. For initial evaluation of the FPGA-based transmitter we compared the RF outputs to a 14-bit AWG. For this purpose we used both QPSK and Gaussian modulation in the electrical B2B configuration. Fig. 5.8 shows the results from electrical B2B measurement. Looking at the calculated EVM values for each of the traces, the performance of FPGA DAC is comparable or slightly better compared to the AWG. For the Gaussian modulation both devices reach the same minimum variance of $V_{\mathrm{min}} = 0.02$. We stress that the AWG just takes a user-provided floating-point values and offloads them to the output, while the QKD transmitter does fixed-point calculations, thus making AWGs inherently more precise in this regard.

We proceed with the optical measurement using a B2B configuration. The RF outputs of the transmitter are connected to the IQ modulator. The bias controller showed 30 dB of carrier suppression and 23 dB for the undesired sideband during characterization. Bob's RLO is detuned by approximately 170 MHz. In the optical measurements three distinct phases are recognized so the relevant parameters according to the security proof can be extracted.

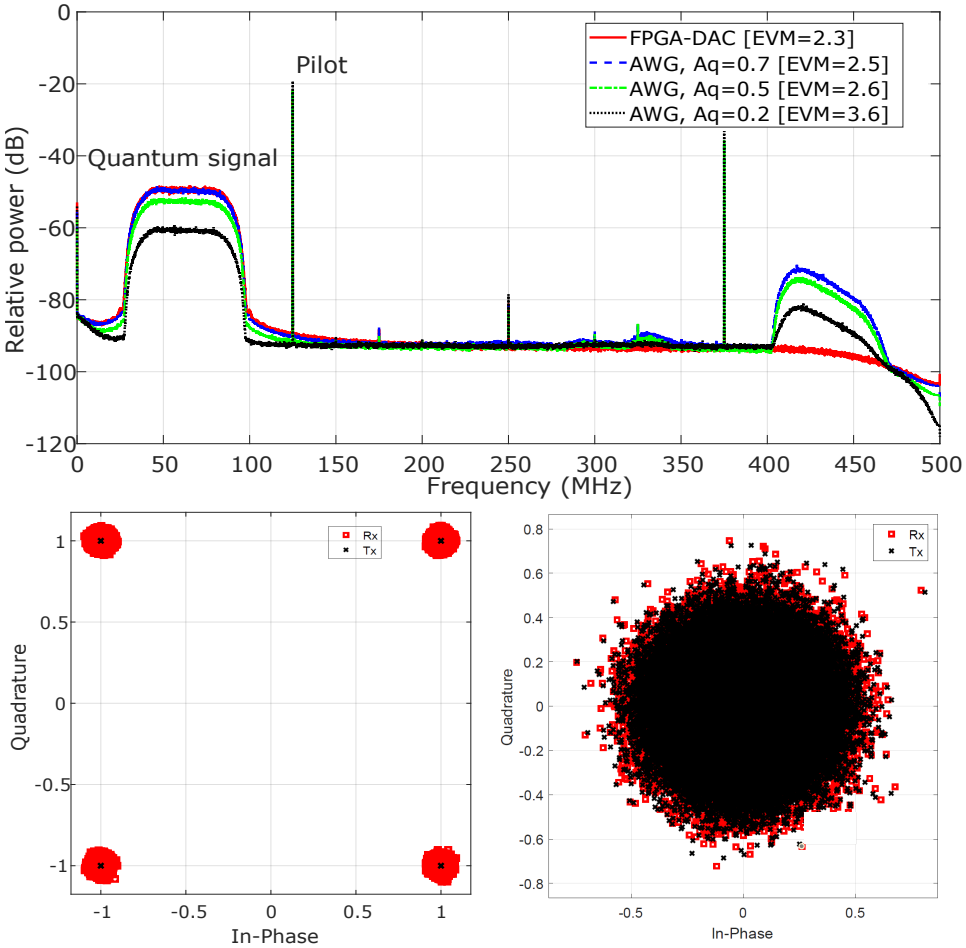


Figure 5.8: Top: comparison between the QKD transmitter and the AWG signal spectrum with QPSK modulation in electrical B2B constellation. The sample rate for the AWG is 0.5 GS/s hence the image band. Note how EVM grows with decreasing the quantum signal amplitude A_q (given here in arbitrary units). Left: recovered QPSK constellation from the QKD transmitter, with EVM=2.3. Right: Recovered Gaussian constellation from the QKD transmitter with the minimum variance $V_{\text{min}} = 0.02$. AWG signal shows similar recovered constellations for the same quantum signal amplitude, thus it is not shown.

Comparing optical B2B and 20 km setups		
	B2B	**20 km**
Modulation variance [mSNU]	1.28	0.49
Excess noise [mSNU]	39.1	35.8
Number of captured symbols	100k	100k

Table 5.5: The variances are given in mili shot noise units (mSNU).

First we measure the electronic noise, which is trusted in the security model. This is done by capturing the homodyne output with the optical input being closed and RLO disabled. After this the RLO is turned on so the inherent (and trusted) shot noise is measured.[6] Finally, the optical inputs are opened and the quantum signal is captured. The (untrusted) excess noise is now calculated as a difference between the measured signal and the shot and electronic noise. For this configuration we present only the QPSK modulation in Fig. 5.9 as it visually illustrates the effects of the shot noise, that are manifested as blurred constellation points (much more than in electrical B2B shown in Fig. 5.8).

A full optical measurement with Gaussian modulation and 20 km long fibre spool is performed. Fig. 5.10 shows the recovered spectrum and the constellation.[7] Note the highly attenuated quantum signal. This time the constellation is divided in eight regions. The synchronization is successful, however not obviously visible due to the shot noise masking the low-variance quantum signal. Table 5.5 is comparing two optical measurements with the Gaussian modulation, the first one is B2B and the other is with the 20 km spool. Note that the excess noise is lower in 20 km configuration. This can be explained with the main source of the excess noise being at the transmitter side. In this case the noise also experiences attenuation through the channel contrary to what would be expected if the source was located closer to the receiver. This does not imply that the asymptotic key rate (Eq. 3.1.1) stays unchanged with varying distance, as the varying attenuation also simultaneously varies the mutual information of Alice and Bob.

Finally we present the calculated asymptotic key rate from a measurement with the AWG and a long sequence of symbols. Table 5.6 shows the results. We use the same security model as before (Fig. 3.2). This is one of the latest measurements and it shows a good performance regarding the low excess noise. This can be attributed to the latest advancements in the

[6] This measurement also captures the electronic noise, so we must subtract it in order to get the pure vacuum noise.

[7] The cross-correlation function is omitted as there is no fundamental difference between the function in the previous measurement and this one.

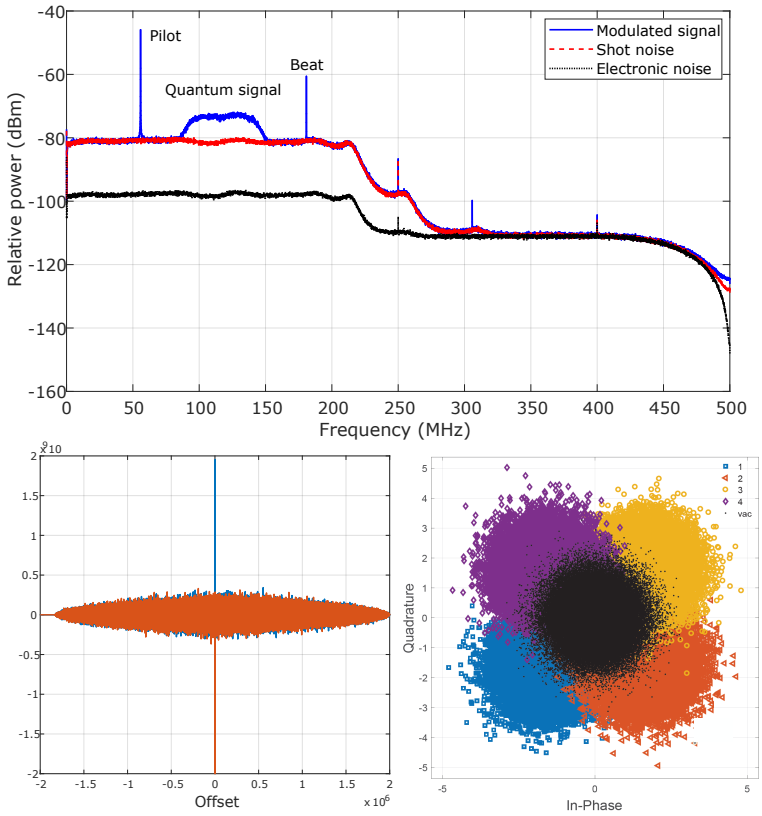


Figure 5.9: Top: Spectrum of the QPSK-modulated quantum signal with the prominent pilot and the beat. Shot noise-limited characteristics of the homodyne detector is noticeable as clearance of the shot noise (red trace) to the electronic noise (black trace). An RF filter (200MHz) is used to decrease high frequency noise. Left: Cross-correlation function between sent and received symbols. The prominent peak mean proper symbol synchronization. Right: Recovered QPSK constellation. Note how shot noise spreads the points (shot noise only is represented with black dots). The received symbols are couloured depending on which quadrant they occupied in the transmitter. The ordered colours mean good synchronization and relatively low noise.

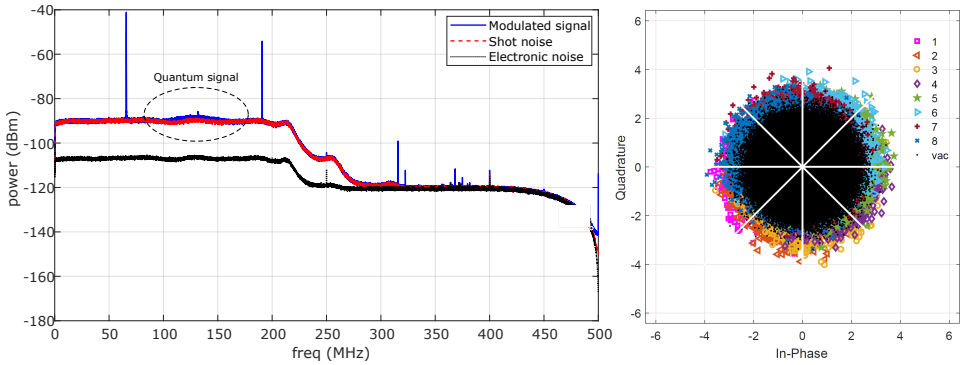


Figure 5.10: Measurement with 20 km optical link and Gaussian modulation.

Channel parameters and key rate	
Channel transmittance η_{chan}	0.2791
Detector transmittance (trusted loss) η_{det}	0.6864
Trusted noise t_{det} [mSNU]	22.3
Excess noise $w(1-\eta_{\text{chan}})\eta_{\text{det}}$ [mSNU]	3
Assumed reconciliation efficiency β	0.95
Mutual information $I(A:B)$ [bits/symbol]	0.6062
Holevo information $\chi(E:B)$ [bits/symbol]	0.4901
Key rate R [bits/symbol]	0.0858

Table 5.6: Channel parameters and calculated key rate from a long measurement using the AWG.

receiver design. To make a better connection of these results to the QKD transmitter we calculate the Holevo information correction factor caused by imperfect Gaussian sampling, described in Section 5.2. For the preparation error $\varepsilon_{\text{prep}}$ that is less than 10^{-8} for the implemented Gaussian sampler, the calculated correction is in the order of magnitude of 10^{-6} bits per symbol. This is four orders of magnitude smaller than the obtained key rate thus confirming the security of the sampler.

We recognize the lack of long measurements that would irrefutably confirm the projected performance of our QKD transmitter. A measurement with a large number of symbols is needed for a precise parameter estimation. The reason it has not been performed yet is not due to the QKD transmitter limitations, as it is capable of constant operation. Furthermore, more extensive measurements in RF domain are welcome to fine tune parameters such as RRC filter roll-off factor and signal amplitudes. Therefore, where we see a potential for improvement to the FPGA firmware is a dynamic software control over such parameters. Implementation of an equalization filter

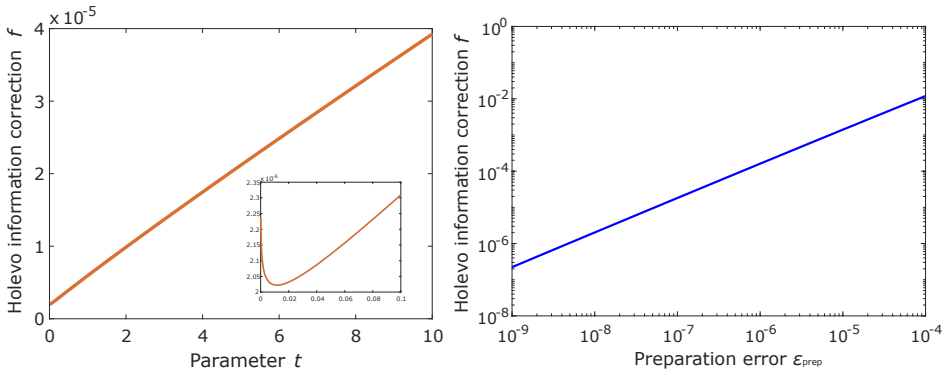


Figure 5.11: Left: For given $\varepsilon_{\text{prep}}$ we calculate the optimized Holevo information correction f with respect to the optimization parameter t (not to be confused with the trusted noise variance in the security model). For $\varepsilon_{\text{prep}} = 10^{-8}$ the minimum is achieved for $t = 0.01$. Right: Optimized f as a function of $\varepsilon_{\text{prep}}$. In the case of key rate $R = 0.08$, f is in the same order of magnitude, thus proving the necessity for a low-$\varepsilon_{\text{prep}}$ Gaussian sampler.

is another option, especially with an increase of the quantum signal bandwidth. Nevertheless, we are confident that the solution for FPGA-based QKD transmitter is a good basis for the future developments.

Chapter 6

Conclusion and outlook

The work performed for this thesis is a part of larger efforts to build secure and high speed CV-QKD and CV-QRNG. One of the motives of the CV technology is a potential to make cheaper devices. Along the same lines, we used standard off-the-shelf telecommunication equipment. FPGAs are today widely used in classical telecommunication. The work presented in this thesis proves that standard electronics and optics can be successfully used for quantum communications without reducing security levels defined in information-theoretic framework.

The effort to build a vacuum fluctuation-based QRNG has been a significant part of this work. We demonstrated a record-braking (real-time) speed of 8 Gb/s. Fast randomness extraction is one of the main reasons we achieved this goal. This extractor was used for both the standalone QRNG and the QKD transmitter. For the standalone QRNG implementation we also developed the online entropy tests. Commercial random number generators generally implement statistical tests on the output data to determine if the data looks uniformly random. This method cannot verify the true randomness, so we implement a test based on power spectral density calculation. This was done according to a rigorous security proof in non-i.i.d. setting and with quantum side information. We also demonstrated interfacing our QRNG with another device using SATA and Aurora standards.

Further development of the vacuum fluctuation-based QRNG will focus on hardware miniaturization. Our implementation is packaged in a standard 19" box, with the FPGA hosted in a separate workstation. The next step is implementing smaller components in a single board with a standard interface such as USB or PCIe. Integrated solutions are also possible, where the components such as laser , homodyne detector and post-processing module can be produced in silicon [124]. Post-processing techniques also have potential to be improved [123]. Toeplitz hashing has potential for an even

more efficient implementations that will take less resources. This is especially important for smaller and lower power devices. The security of the QRNG can also be improved by introducing the semi-device-independent framework. This direction is still more in domain of fundamental research as the security models are conservative and sometimes impractical.

High speed QRNG was used as a base for our QKD transmitter with Gaussian modulation. We built a Gaussian sampler with a peak output rate of 370 MS/s for 8-bit samples. Security parameters for this implementation were calculated. For all practical purposes it was shown that the imperfections do not decrease the key rate. To our knowledge this is the first high-rate implementation of a Gaussian sampler in QKD. The transmitter is built to support CV-QKD setups with real local oscillator, which is the current state-of-the-art. It was tested with a broadband symbol rate of 50 MBd/s.

Further work needs to be done, particularly on the receiver side, in order for our CV-QKD setup to achieve a stable positive key rate, assuming general attacks and finite regime. Novel security proofs are expected in the future that might prove security for some easier-to-implement schemes such as QPSK. Regardless of the modulation, CV-QKD systems need to prove their reliability in a real environment, especially in the existing telecommunication networks, where the usage of dark fibers and multiplexing with the classical channels may introduce too much of unwanted noise.

Appendix A

FPGA and software framework

A.1 AXI protocol preliminaries

This section of the appendix introduces AXI standard which is extensively used for communication between modules in the firmware. AXI is part of ARM AMBA, a family of micro controller buses. We are using AXI4 standard, the second major version released in 2010. The AXI specifications describe an interface between a single AXI master and AXI slave, representing IP cores that exchange information with each other. However, multiple memory-mapped AXI masters and slaves can be connected using special modules. There are four types of AXI standard: *AXI4*, *AXI4-Lite* and *AXI4-Stream*. Our firmware builds upon the last two, Lite standard used for simple low-throughput memory-mapped communication, and Stream for high-speed streaming data [153].

- **AXI4-Lite** – A lightweight, single transaction memory-mapped interface. It has a small logic footprint which is very important for building efficient FPGA designs. The interface between Lite master and Lite slave consists of five channels, two for address for read/write, two data read/write channels and write response channel. Lite standard is used in our firmware, however we didn't need to customize Lite interfaces in our modules.

- **AXI4-Stream** – The protocol does not use addressing and allows unlimited data burst size. It relies on a very simple handshaking process. The handshake is done with two 1-bit signals, *TVALID* which is the output of a master, and *TREADY* which is the output of a slave. The data is transferred only when both signals are asserted. This two-way flow control mechanism enables both master and slave to control the rate at which the data is transmitted across the interface. The master

is not permitted to wait until TREADY signal is asserted. This simple handshake is shown on Fig. A.1 as a digital timing diagram [154]. The data bus is denoted as *TDATA*. Along these three signals Stream standard defines others optional signals too, mostly for frame control, however our modules are not using them.

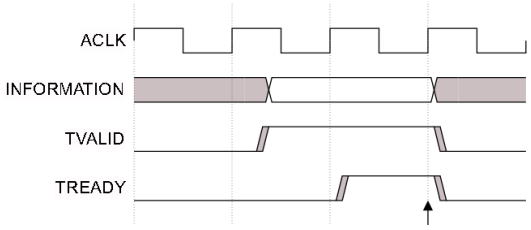


Figure A.1: AXI4-Stream handshaking timing diagram. Taken from [154].

A.2 Intellectual property (IP) cores

In this section we are providing a short description of the important IP cores for QKD and QRNG FPGA firmwares.

- **FIFO** – First-in first-out (FIFO) buffers are extensively used in FPGA designing as they are relatively simple, yet very powerful tool. As the name suggest, this kind of memory takes data and stores it in an ordered fashion, the first piece of data that was written is to be the first to be read out. In order to generate this core we are using *FIFO generator v13.1* GUI [155]. The GUI gives us different option which we use in order to customize the core for a particular application. The first option is *Interface type*. Here we used the native interface type in contrast to AXI Memory Mapped or AXI Stream, since it is simpler and FIFOs are used within AXI-enabled modules and not as a standalone module. Nevertheless native interface signals are easily used along the AXI signals of the top-level module. There are three options for which memory primitives are used to build a FIFO buffer: builtin FIFO primitives, block memory and distributed memory. Generally we use builtin FIFO primitives since they are optimized for the task. The second kind utilizes BRAM primitives, while the least desirable is the distributed memory, which means using general purpose flip-flops (this is unnecessary since our FPGA has a lot of BRAM and FIFO resources). One of the most common usages of FIFOs is when they act as clock boundary, i.e. read and write clocks are different. This is a safe way for two clock regions to interact, especially if the clocks are of similar frequency. Therefore, there is an option for common clock or

independent clocks. If common clock implementation is selected, that generally means FIFO is used purely as a memory element to buffer some data. Read mode of the FIFO can be standard, where the data is available one clock cycle after the read enable signal is asserted, or first-word-fall-through where the output data is readily available and valid even if the read enable signal is deasserted. This feature is sometimes very convenient when latency is important. Write width and read width can be different, thus enabling the user to change the data bus width which is useful in AXI protocol.

- **AURORA 8b10b** – The core implements a simple and scalable link-layer protocol for high-speed serial communication [156]. The protocol is used in chip-to-chip links, where serial communication can greatly reduce complexity of PCB (printed circuit board) or board-to-board communication, which we are using in our project.

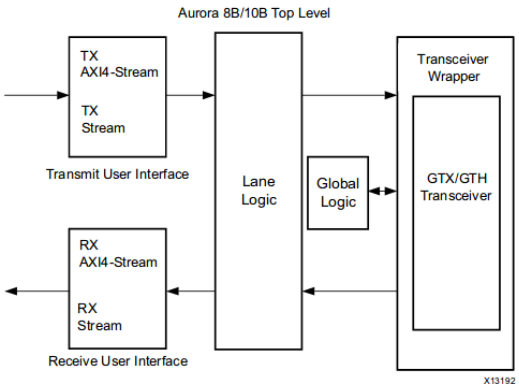


Figure A.2: Top-level architecture of Aurora core. Taken from [156](Xilinx)

In order to transmit and receive serial data, the core instantiates GTH transceivers. Fig. A.2 shows the top-level architecture of Aurora. On the user side there is Tx and Rx AXI-Stream logic for a convenient implementation with other AXI enabled modules. In the middle there is lane logic to control dataflows. Multiple lanes mean multiple GTH transceivers. We are focusing on the simplest one-lane Aurora with maximum throughput of 6.6 Gb/s. For the purpose of DC balancing, synchronization and clock recovery Aurora employs 8b/10b coding [157]. This code is widely used in telecommunications. It is a line code that maps 8-bit words to 10-bit code symbols. Symbols are made in such a way so the counts of ones and zeros are not different by more than two in a string of at least 20 bits, thus also relaxing the lower bandwidth limit requirement in a telecom system. Even if the user does not provide data to Aurora core, the transmitter will offload special 8b/10b idle 8b/10b symbols which is also convenient for de-

bugging the link. In our project we have Tx-only simplex framing link layer. We chose frames of 512 bytes, and the logic for counting the bytes is implemented as a separate module.

- **Direct digital synthesizer (DDS)** DDS core is a source of sinusoidal waveforms with many options in order to be used in wide array of applications [158]. In contrast to voltage controlled oscillators (VCOs), in the digital domain there are direct digital synthesizers (DDSs) and numerically controlled oscillators for the purpose of generating sinusoidal waveforms used in modulation/demodulation, digital up/downconversion etc. In our designs we employ DDS IP core in order to have sinusoidal samples readily available in the code where they are easily manipulated with. DDS core uses a lookup table which stores the samples of a sinusoid. A digital integrator is used to generate a suitable phase argument that is mapped by the lookup table to the desired output waveform. In the DDS IP core settings we are focusing on several options. First we set up the system clock frequency so the core knows what is the relationship between this one and the required one. The important parameter is spurious free dynamic range denoted in dB. It tells how large is the clearance between the main peak of the spectrum of the output signal, and the highest spurious peak, which is the consequence of the finite precision of the digital design. In principle one wants the highest possible spurious range, however it is limited by the output number of bits. It turns out that 16 bits output is enough for 95 dB of spurious free range.

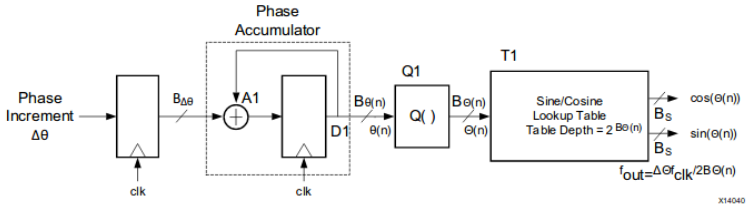


Figure A.3: Simplified schematics of the DDS core. Picture taken from [158] (Xilinx)

Fig. A.3 shows a simplified block diagram of DDS system. Note that the samples are written in, what is effectively a ROM, with the depth of $2^{B_{\Theta(n)}}$. That means that one full circle, or 2π radians are divided by an integer which is a power of two. Since the ROM is fixed during the core run time, it is clear that the generated frequency must be an integer division of the system frequency, and that integer needs to be a power of two. It is however possible to use rasterized mode of operation, where ROM length is some integer other than power of two. This can produce frequencies $f_{sys}\frac{N}{M}$, where N and M are integers.

When the requested frequency is not an integer fraction of thy system frequency, there is no way of getting the exact value. However, there are two techniques implemented in the core which can help to produce a frequency very close to the requested one – phase dithering and Taylor series corrected DDS. The detailed implementation of these techniques is beyond the scope of this appendix. What is important is the exact number of FPGA primitives such as BRAMs and DSPs which are used extra for these techniques.

- **Finite impulse response (FIR) filter** FIR core enables a user to generate a highly parameterizable, area-efficient high-performance FIR filters [159]. The conventional single-rate FIR version of the core computes the convolution sum defined with

$$y(k) = \sum_{n=0}^{N-1} a(n)x(k-n) \quad k = 0, 1, ... \tag{A.2.1}$$

This sum can be computed with the tapped delay line shown in Fig. A.4. In FPGA the filter is implemented with one or more time-shared multiple-accumulate units which are functionally equivalent to the system shown in the figure. The core uses AXI-Stream interface. Coefficients of the filter are provided by the user and are saved in the hardware using a certain number of bits. The number of bits determines the precision and the number of resources to be used (BRAMs and DSPs). There are several options for the type of the filter – single rate, interpolation, decimation, Hilbert and interpolated. In our design we focused on the interpolation type where the filter introduces zero-valued samples between consecutive input samples, and further filters the stream thus performing the pulse shaping. The exact frequency characteristics of the filter depends on the coefficients, so the filter is basically design beforehand using a tool such as Matlab. Since the samples experience many operations of multiplication and addition through the filter, there are options to manage the bit growth. Full precision leave all the bits intact while other rounding options reduce the number of bits to some fixed value, using an algorithm such as rounding up or down.

- **Fast Fourier transform (FFT)** FFT IP core implements the Cooley-Tukey FFT algorithm, a computationally efficient method for calculating the Discrete Fourier Transformation (DFT) [160]. DFT length is a power of two and it is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-ink\frac{2\pi}{N}} \quad k = 0, ..., N-1 \tag{A.2.2}$$

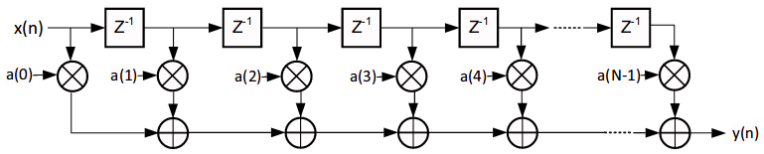


Figure A.4: Conventional tapped delay line FIR filter representation. Picture taken from [159].(Xilinx)

Here N is the transform size. There are two methods used for two different architectures, the first is decimation-in-time used for the burst architecture and the other is decimation-in-frequency used for the pipelined streaming architecture. Our design required constant pipelined designed, therefore we only used this option. As a trade-off to the real-time feature, the design consumes the highest number of resources. The user can choose a transform length between 8 and 65536. We experimented with the lengths up to 32768, since the designs with highest possible length had never ending compiling time. As with the other cores described in this appendix, the core is AXI-Stream enabled. This core enables user to operate with both floating and fixed point arithmetic. Even though floating point arithmetic is 'unnatural' for FPGAs, the core is fully capable of using it for the expense of FPGA resources. Similar to the FIR core, due to the nature of the algorithm, bit growth is allowed or managed by truncation or rounding.

A.3 FPGA and ADC/DAC boards

In the experiments for this thesis we used PC821 FPGA board and FMC120 ADC/DAC board by Abaco (former 4DSP). PC821 board is PCIe compliant card. It communicates with the host computer through PCIe interface. The board supports two FPGA Mezzanine Card (FMC) interfaces. In our system, one of the two FMCs is used for FMC120 ADC/DAC daughter card. Besides the two boards, we acquired the reference firmware and software from the vendor. The reference firmware is used to test the basic functionalities of the ADC/DAC and other features on the board, while the software is used for sending commands, fetching captured data to the host computer and similar. The reference software and firmware are described in more detail in the next section of the appendix.

PC821 block diagram is shown in Fig. A.5. The hear of the system is FPGA, in our case Xilinx Kintex UltraScale XCKU115. This is mid to high-end FPGA specially optimized for DSP tasks. XCKU115 is the most

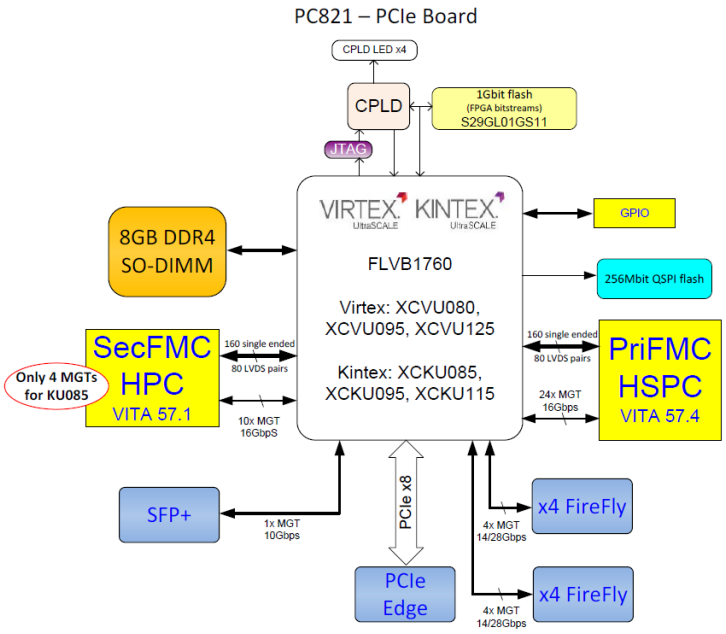


Figure A.5: PC821 block diagram (Abaco)

powerful chip in the UltraScale family. It is built of almost a million and a half flip-flops, 5500 DSP slices, 75Mb of block RAM (BRAM), in total of 64 16 Gb/s GTH multi-gigabit serial transceivers etc. For programming the FPGA, standard JTAG interface is used. For high speed serial communication there are SFP+ and FireFly interfaces. SFP+ nominal speed is 10 Gb/s and we used it to output random numbers in QRNG design. The board hosts 8 GB DDR4 RAM memory protected with a Faraday cage. We extensively used RAM memory in our designs for buffering, capturing or offloading of large chunks of data. The DDR4 interface is fast enough to capture all four ADC channels at the same time without loss, i.e. 64 Gb/s of data. The board implements also several general purpose input/output (GPIO) ports. These were used for low speed monitoring signals in QRNG project. What is not shown in Fig. A.5 is I2C controller and Si5338 clock distribution chip [161]. I2C is used for communicating with several slaves, among which is Si5338 chip. The chip takes a 100 MHz crystal output as the input and has four output channels. These outputs are used to clock SFP+ transceivers among others. We used the programmable feature to program the data rate at SFP+ interface.

FMC120 is an eight-channel ADC/DAC (four per each direction) daughter board. Both ADC and DAC channels are 16-bit at the maximum rate of 1 GS/s. The board also incorporates an external clock reference input

and one trigger/sync input. The analog signal inputs and outputs are DC coupled and are connected to SSMC coax connectors on the front of the panel. In usual operation the board uses internal 100 MHz oscillator which is used as an input to LMK04828B clock jitter cleaner which provides clock multiplication capability for ADCs and DACs.

The DAC suite implemented on the board is DAC39J84 by Texas Instruments [162]. It is convenient for customizing transmitter parameters as it implements many options which we explored. Its maximum data rate is 1.25 Gb/s. We used the default 1 Gb/s rate. Block diagram of the DAC

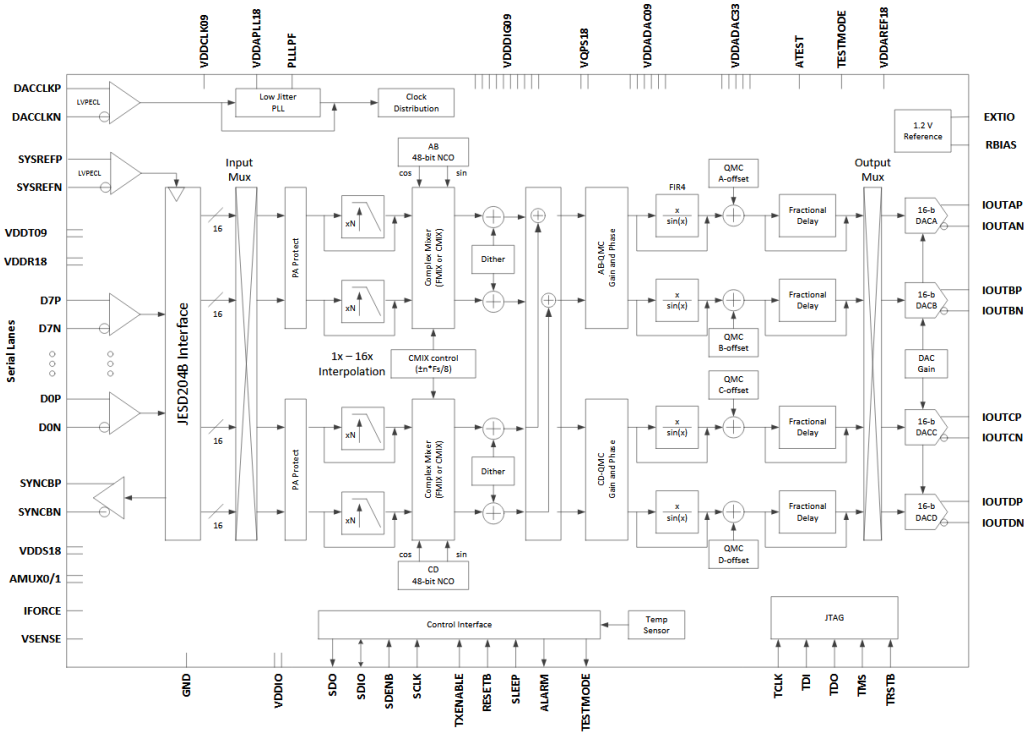


Figure A.6: DAC39J84 block diagram. Picture taken from [162]. (Texas Instruments)

is shown in Fig. A.6. Immediately after the JESD204B interface (through which the DAC is connected to the data source, in this case FPGA) there is a programmable interpolator. The maximum sample rate of the DAC is 2.8 GS/s. Since out data rate is fixed to 1 GS/s, the only possible options are $\times 1$ or $\times 2$ interpolation. The default one is $\times 2$ giving the total sample rate of 2 GS/s. In the experiments with the QKD transmitter we did not see any difference between the two interpolation options. Our frequency range was significantly below the Nyqist frequency for the $\times 1$ interpolation (500 MHz), therefore this result makes sense. After the interpolator, the DAC

implements two complex mixers for two pairs of data channels using numerically controlled oscillators (NCOs). The NCOs are controlled with 48-bit registers for precise frequency placement and 16-bit registers for the phase. The NCOs are implemented in a similar manner as DDS core in FPGA. Due to their nature of being a dedicated hardware compared to DDS soft core, the NCOs are inherently more precise and easily programmable with software. However, there is no possibility for the equalization of the signal after it was brought to the passband. Furthermore, with the upconversion performed in FPGA, we are able to implement an independent pilot signal, and use the last two DACs for trigger and sync signals that are used for synchronizing the oscilloscope. The third option we experimented with is $x/\sin(x)$ (also called sinc) built-in filter. This FIR filter is implemented with a positive gain so it flattens out the response for the first Nyqist zone. Ass with the interpolator, due to our relatively low upconversion frequency and low bandwidth, the filter does not make a difference in our measurements.

A.4 Abaco (4DSP) framework

Along with the hardware, we acquired the reference firmware and software from Abaco which we used as a base to build our own designs. The firmware is built using proprietary Stellar IP software. Stellar IP keeps libraries of the firmware modules built specifically for PC821/FMC120 system. The modules are AXI-Lite and AXI-Stream enabled so they are easily connected. In Stellar IP, the modules are called stars. For building the firmware Stellar IP shows a GUI where a box diagram can be drawn. Upon finishing the design Stellar IP generates the corresponding VHDL project in Vivado. In order to illustrate how the framework works, Fig. A.7 shows a piece of a firmware involved in transmitting the data from ADC to the host PCIe interface.

- **FMC120 star** – used for communicating with FMC120 ADC/DAC board. It utilizes GTH transceivers for getting the high speed data from the FMC connector. This high speed data is forwarded to the rest of the firmware through AXI-Stream interface. It is important that GTH transceivers of the star recover the 250 MHz clock to which ADC data is synchronous. This clock is also used in the custom Toeplitz hashing module. Here, AXI-Stream bus is 64 bit wide, and with this clock rate, it translates to 16 Gb/s full data throughput.

- **64to256 star** – this type of star is commonly used in AXI enabled firmwares. It's sole function is to convert a 64 bit AXI-Stream bus to a 256 bit one, while using the same clock. If the data is constantly present at the 64-bit bus, it goes in bursts on every fourth clock cycle on the 256-bit output bus. For this purpose, the star employs a simple

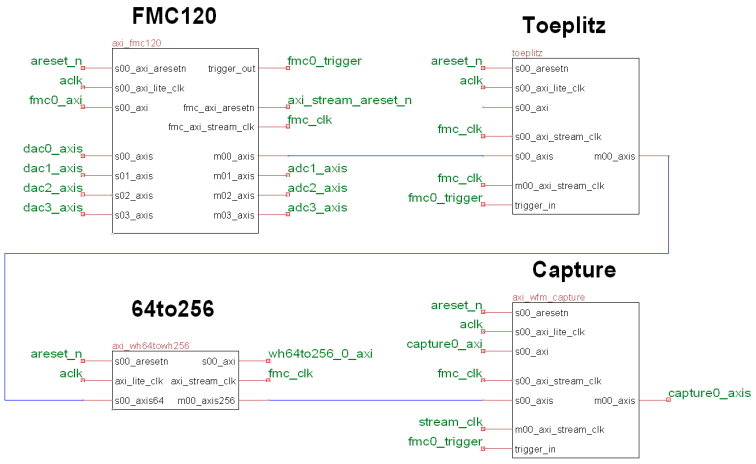


Figure A.7: Stars included in the typical ADC to memory data flow. The data is offloaded to Host star which is not shown here. Note the standard AXI infrastructure.

single-clock FIFO. Due to its simplicity, 64to256 star does not utilize AXI-Lite interface.

- **Capture star** – the star uses a FIFO buffer to capture high rate ADC data. The FIFO is able to capture around 3000 samples. Two AXI-Stream interfaces are implemented in this star: the first one used for capturing the ADC data, and the second one used for offloading the data to Host star. AXI-Lite interface is used for controlling the capturing and offloading. Note that the star can be used as a clock boundary module as it implements an independent-clock FIFO.

- **Host star** (or *PCIe star*) – the star takes care of important functionality of communicating with the host PC. On one side it implements PCIe protocol, while on the other it communicates with the rest of the firmware using AXI standards. The star is capable of understanding software commands encapsulated into PCIe format. It discriminates between high speed data sent from the host machine (which is for example bound to be offloaded to a DAC) and control and command data distributed by AXI-Lite. Proprietary Abaco (4DSP) API is used by user applications for this software-firmware low level communication.

- **DDR4 star** implements an interface to the external DDR4 RAM. The star is design so it effectively acts as a huge 8GB FIFO from the point of view of the other Stellar IP stars. We used the feature extensively in our designs.

- **Repeat star** acts as a waveform generator for testing the DACs. It implements a BRAM which is initialized with some custom waveform and the data is constantly offloaded in a periodic fashion (hence the usage of BRAMs and not FIFOs which get empty after one cycle).

The only way of introducing a new star into a Stellar IP design is by *cloning* an existing star. This means taking a star which is the most similar to intended design and then modifying it accordingly. Due to relative simplicity and fully working AXI-Lite interface, we were cloning Capture star for implementing Toeplitz hashing and the other modules. However, the necessary modifications were done 'at lower level', directly in VHDL code in Vivado.

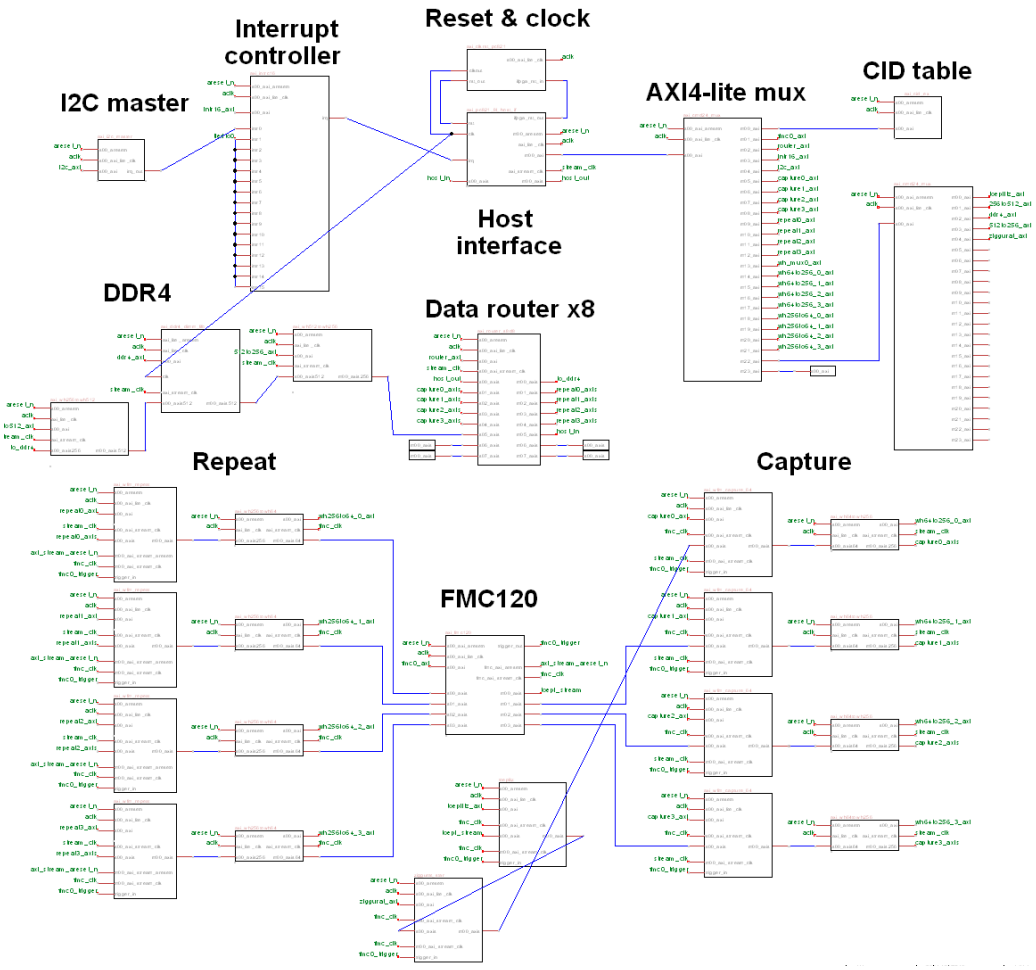


Figure A.8: A full Stellar IP design with AXI-Lite and I2C infrastructure. DDR4 interface star is also present. This particular firmware design implements Toeplitz hashing and Gaussian sampler in full streaming mode with capturing of the data to the RAM memory.

Bibliography

[1] A. Einstein, B. Podolsky, N. Rosen, "Can quantum-mechanical description of physical reality be considered complete?", Physical Review,47:777–780, (1935)

[2] W. Heisenberg, "Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik", Zeitschrift für Physik (in German), 43 (3–4): 172–198, (1927)

[3] W. Wootters, W. Zurek, "A Single Quantum Cannot be Cloned", Nature. 299 (5886): 802–803, (1982)

[4] C.H. Bennett, G. Brassard, "Quantum Cryptography: Public key distribution and coin tossing", Pro- ceedings of IEEE International Conference on Computers, Systems and Signal Processing, pages 175–179, (1984)

[5] J.P. Dowling, G.J. Milburn, "Quantum technology: the second quantum revolution", Philosophical Transactions of the Royal Society of London, 361, Series A: Mathematical, Physical and Engineering Sciences, (2003)

[6] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", SIAM J.Sci.Statist.Comput. 26, (1997)

[7] M. D. Reid, "Quantum cryptography with a predetermined key, using continuous-variable Einstein-Podolsky-Rosen correlations", Physical Review A,62(6):062308, (2000)

[8] M. Fox, "Quantum Optics: An Introduction", Physics Today - PHYS TODAY. 60. 10.1063/1.2784691, (2006)

[9] S. L. Braunstein, P. van Loock, "Quantum information with continuous variables", Rev. Mod. Phys. 77, 513, (2005)

[10] R. Daendliker, "Concept of modes in optics and photonics", Proc. SPIE 3831, Sixth International Conference on Education and Training in Optics and Photonics, 10.1117/12.388718, (2000)

[11] E. Wigner, "On the Quantum Correction For Thermodynamic Equilibrium" Phys. Rev. 40, 749, (1932)

[12] C. Weedbrook, S. Pirandola, R. García-Patrón, N. J. Cerf, T. C. Ralph, J. H. Shapiro, S. Lloyd, "Gaussian quantum information", Rev. Mod. Phys. 84, 621, (2012)

[13] M. Combescure, D. Robert , Bosonic Coherent States. In: "Coherent States and Applications in Mathematical Physics", Theoretical and Mathematical Physics. Springer, Dordrecht, (2012)

[14] N.J. Cerf, M. Lévy, G. Van Assche, "Quantum distribution of Gaussian keys using squeezed states", Phys. Rev. A. 63. 10.1103/PhysRevA.63.052311, (2001)

[15] L.S. Madsen, V. Usenko, M. Lassen, R. Filip, U.L. Andersen, "Continuous variable quantum key distribution with modulated entangled states", Nature communications. 3. 1083. 10.1038/ncomms2097, (2012)

[16] T. Gehring, V. Händchen, J. Duhme, F. Furrer, T. Franz, C. Pacher, R. Werner, R. Schnabel, "Implementation of continuous-variable quantum key distribution with composable and one-sided-device-independent security against coherent attacks", Nature Communications. 6. 8795. 10.1038/ncomms9795, (2015)

[17] C. E. Shannon, "A Mathematical Theory of Communication", Bell System Technical Journal, (1948)

[18] T. M. Cover, J. A. Thomas, "Elements of Information Theory", Wiley-Interscience, 2nd edition, (2006)

[19] R. Renner, "Security of Quantum Key Distribution", Diss. ETH No. 16242, (2006)

[20] D. Frauchiger, R. Renner, M. Troyer, "True randomness from realistic quantum devices", arXiv:1311.4547 [quant-ph], (2013)

[21] Y. Cliff, C. Boyd, and J. G. Nieto, in Proceedings of Applied Cryptography and Network Security (Springer, New York, 2009), pp. 53 – 70. How to Extract and Expand Randomness: A Summary and Explanation of Existing Results.

[22] M.M. Wilde, "Quantum Information Theory", Cambridge University Press, (2013)

[23] N. J. Cerf, P. Grangier, "From quantum cloning to quantum key distribution with continuous variables: a review (Invited)", J. Opt. Soc. Am. B 24, 324-334, (2007)

[24] E. Diamanti, A. Leverrier, "Distributing Secret Keys with Quantum Continuous Variables: Principle, Security and Implementations", Entropy, 17, 6072-6092; doi:10.3390/e17096072, (2015)

[25] F. Laudenbach et al., "Continuous-Variable Quantum Key Distribution with Gaussian Modulation – The Theory of Practical Implementations" arXiv : 1703 . 09278v1 [quant-ph] pp. 1–64, (2017)

[26] N. J. Cerf, M. Lévy, G. Van Assche, "Quantum distribution of Gaussian keys using squeezed states", Phys. Rev. A 63, 052311, (2001)

[27] D.Gottesman, J.Preskill, "Secure quantum key distribution using squeezed states", Phys. Rev. Lett. 86, 4942-4945,(2001)

[28] F. Grosshans,P. Grangier, "Continuous variable quantum cryptography using coherent states," Phys. Rev. Lett. 88, 057902, (2002).

[29] I. Devetak, A. Winter, "Distillation of secret key and entanglement from quantum state", Proc. R. Soc. Lond. A 461, (2005)

[30] F. Grosshans, P. Grangier, "Reverse reconciliation protocols for quantum cryptography with continuous variables," arXiv: quant-ph/0204127 (2002).

[31] A. Leverrier, "Composable Security Proof for Continuous-Variable Quantum Key Distribution with Coherent States", Phys. Rev. Lett. 114, 070501, (2015)

[32] A. Leverrier, "Security of Continuous-Variable Quantum Key Distribution via a Gaussian de Finetti Reduction", Phys. Rev. Lett. 118, 200501, (2017)

[33] H-A. Bachor, T. C. Ralph, "A guide to experiments in Quantum Optics", Wiley-VCH, 2nd edition, (2004)

[34] G. Agrawal, "Fiber-Optic Communication Systems: Fourth Edition", 10.1002/9780470918524, (2012)

[35] M.W. Hamilton, "Phase shifts in multilayer dielectric beam splitters" American Journal of Physics 68, 186, https://doi.org/10.1119/1.19393, (2000)

[36] K. Kikuchi, "Fundamentals of Coherent Optical Fiber Communications," J. Lightwave Technol. 34, 157-179, (2016)

[37] S. Tsukamoto, K. Katoh, K. Kikuchi, "Coherent demodulation of optical multilevel phase-shift-keying signals using homodyne detection and digital signal processing," in IEEE Photonics Technology Letters, vol. 18, no. 10, pp. 1131-1133, (2006)

[38] R. Soref, B. Bennett, "Electrooptical effects in silicon," in IEEE Journal of Quantum Electronics, vol. 23, no. 1, pp. 123-129, (1987)

[39] M. J. Sieben, "Single sideband modulation for digital fiber optic communication systems", PhD thesis, University Alberta, (1998)

[40] "Everything You Need to Know About Complex Optical Modulation", Keysight Technologies, `http://literature.cdn.keysight.com/litweb/pdf/5992-2888EN.pdf`

[41] M. Izutsu, S. Shikama, T. Sueta, "Integrated optical SSB modulator / frequency shifter," IEEE Journal of Quantum Electronics, vol. QE- 17, no. 11, pp. 2225- 2227, (1981)

[42] S. Shimotsu, S. Oikawa, T. Saitou, N. Mitsugi, K. Kubodera, T. Kawanishi, Tetsuya, M. Izutsu, "Single-sideband modulation performance of LiNbO3 integrated modulator consisting of four-phase modulator waveguides", Photonics Technology Letters, IEEE. 13. 364 - 366. 10.1109/68.917854, (2001)

[43] L. Liao, D. Samara-Rubio, M. Morse, A. Liu, D. Hodge, D. Rubin, U. D. Keil, T. Franck, "High speed silicon Mach-Zehnder modulator," Opt. Express 13, 3129-3135, (2005)

[44] D.G. Welsch, W. Vogel, T. Opatrný, "Homodyne Detection and Quantum-State Reconstruction" Prog. Opt. 39. 10.1016/S0079-6638(08)70389-5, (2009)

[45] M. Shibutani, S. Yamazaki and S. Murata, "Polarization diversity coherent optical receiver with a balanced receiver configuration," ECOC 88 (Conf. Publ. No.292), Brighton, UK, pp. 151-154 vol.1, (1988)

[46] H.P. Yuen, V. W. S. Chan, "Noise in homodyne and heterodyne detection," Opt. Lett. 8, 177-179, (1983)

[47] G. Breitenbach, S. Schiller, J. Mlynek, "Measurement of the quantum states of squeezed light", Nature 387, pages471–475, (1997)

[48] D.G. Welsch, W. Vogel, T. Opatrný, "Homodyne Detection and Quantum-State Reconstruction", Prog. Opt. 39. 10.1016/S0079-6638(08)70389-5, (1999)

[49] D. T. Smithey, M. Beck, J. Cooper, M. G. Raymer, A. Faridani, "Complete experimental characterization of the quantum state of a light mode via the Wigner function and the density matrix: application to quantum phase distributions of vacuum and squeezed-vacuum states", Physica Scripta, Volume 1993, T48, (1993)

[50] F. Laudenbach, B. Schrenk, C. Pacher, M. Hentschel, C. F. Fung, F. Karinou, A. Poppe, M. Peev, H. Hübel, "Pilot-assisted intradyne reception for high-speed continuous-variable quantum key distribution with true local oscillator", arXiv:1712.10242 [quant-ph], (2017)

[51] J.H. Shapiro, "The Quantum Theory of Optical Communications", IEEE Journal of Selected Topics in Quantum Electronics, vol. 15, issue 6, pp. 1547-1569, (2009)

[52] B. Zeidman, "Designing with FPGAs and CPLDs", CMP Books, Lawrence, KS, (2002)

[53] R. Woods, J. McAllister, Y. Yi, G. Lightbody, "FPGA-based Implementation of Signal Processing Systems", Wiley Publishing, (2017)

[54] "Introduction to FPGA Design with Vivado High-Level Synthesis", `www.xilinx.com`, (2019)

[55] "UlstraScale Architecture Clocking Resources", `www.xilinx.com`, (2018)

[56] "Using Digital Clock Managers (DCMs) in Spartan-3 FPGAs", `/www.xilinx.com`, (2006)

[57] "Si5338 - I2C-Programmable Any-Frequency, Any-Output Quad Clock Generator", Silicon Labs data sheet

[58] `https://www.allaboutcircuits.com/technical-articles/clock-management-clock-resources-of-fpgas/`

[59] T. S. Chow, "Testing Software Design Modeled by Finite-State Machines," in IEEE Transactions on Software Engineering, vol. SE-4, no. 3, pp. 178-187, (1978)

[60] D. Brand, P. Zafiropulo, "On communicating finite-state machines", Journal of the ACM (JACM), Volume 30 Issue 2, (1983)

[61] "Finite State Machines", Xilinx Lab Workbook, `https://www.xilinx.com/support/documentation/university/Vivado-Teaching/HDL-Design/2013x/Nexys4/Verilog/docs-pdf/lab10.pdf`

[62] A. M. Lance, T. Symul, V. Sharma, C. Weedbrook, T. C. Ralph, P. K. Lam, "No-Switching Quantum Key Distribution Using Broadband Modulated Coherent Light", Phys. Rev. Lett. 95, 180503, (2005)

[63] P. Jouguet, S. Kunz-Jacques, A. Leverrier, P. Grangier, E. Diamanti, "Experimental demonstration of long-distance continuous-variable quantum key distribution", Nature Photonics volume 7, pages378–381, (2013)

[64] Y. Zhang, Z. Li, Z. Chen, C. Weedbrook, Y. Zhao, X. Wang, Y. Huang, C. Xu, X. Zhang, Z. Wang, M. Li, X. Zhang, Z. Zheng, B. Chu, X. Gao, N. Meng, W. Cai, Z. Wang, G. Wang, S. Yu, H. Guo, "Continuous-variable QKD over 50 km commercial fiber", Quantum Sci. Technol. 4, 035006, (2019)

[65] J. Lodewyck, M. Bloch, R. García-Patrón, S. Fossier, E. Karpov, E. Diamanti, T. Debuisschert, N. J. Cerf, R. Tualle-Brouri, S. W. McLaughlin, P. Grangier, "Quantum key distribution over 25km with an all-fiber continuous-variable system", Phys. Rev. A 76, 042305 (2007)

[66] D. Huang, P. Huang, D. Lin, G. Zeng, Guihua, "Long-distance continuous-variable quantum key distribution by controlling excess noise", Scientific Reports. 6. 19201, (2016)

[67] P. Jouguet, S. Kunz-Jacques, E. Diamanti, "Preventing calibration attacks on the local oscillator in continuous-variable quantum key distribution," Phys Rev. A87, 062313, (2013)

[68] B. Qi, P. Lougovski, R. Pooser, W. Grice, M. Bobrek, "Generating the Local Oscillator "Locally" in Continuous-Variable Quantum Key Distribution Based on Coherent Detection", Phys. Rev. X 5, 041009, (2015)

[69] D. B.S. Soh, C. Brif, P. J. Coles, N. Lütkenhaus, R. M. Camacho, J. Urayama, M. Sarovar, "Self-Referenced Continuous-Variable Quantum Key Distribution Protocol", Phys. Rev. X 5, 041010, (2015)

[70] D. Huang, P. Huang, D. Lin, C. Wang, G. Zeng, "High-speed continuous-variable quantum key distribution without sending a local oscillator," Opt. Lett. 40, 3695-3698, (2015)

[71] S. Kleis, M. Rueckmann, C. G. Schaeffer, "Continuous variable quantum key distribution with a real local oscillator using simultaneous pilot signals," Opt. Lett. 42, 1588-1591, (2017)

[72] H. H. Brunner, L. C. Comandar, F. Karinou, S. Bettelli, D. Hillerkuss, F. Fung, D. Wang, S. Mikroulis, M. Kuschnerov, A. Poppe, C. Xie, M. Peev, "Low-noise, low- complexity CV-QKD architecture," in QCrypt 2017, Cambridge, pp. 2–4, (2017)

[73] F. Laudenbach, B. Schrenk, C. Pacher, M. Hentschel, C.F. Fung, F. Karinou, A. Poppe, M. Peev, H. Hübel, "Pilot-assisted intradyne reception for high-speed continuous-variable quantum key distribution with true local oscillator", Quantum 3, 193 (2019)

[74] S. Ren, R. Kumar, A. Wonfor, X. Tang, R. Penty, I. White, "Reference pulse attack on continuous variable quantum key distribution with local

local oscillator under trusted phase noise", Journal of the Optical Society of America B Vol. 36, Issue 3, pp. B7-B15, (2019)

[75] W. Zhao, Y. Guo, L. Zhang, D. Huang, "Phase noise estimation using Bayesian inference for continuous-variable quantum key distribution," Opt. Express 27, 1838-1853, (2019)

[76] S. Ghorai, P. Grangier, E. Diamanti, A. Leverrier, "Asymptotic Security of Continuous-Variable Quantum Key Distribution with a Discrete Modulation", Phys. Rev. X 9, 021059, (2019)

[77] F. Laudenbach, C. Pacher, C. F. Fung, M. Peev, A. Poppe, H. Hübel, "Practical noise modelsfor CV-QKD implementations", poster, Cambridge, QCrypt 2017, (2017)

[78] D. Mayers, A. Yao, "Quantum cryptography with imperfect apparatus," Proceedings 39th Annual Symposium on Foundations of Computer Science, pp. 503-509, (1998)

[79] M. N. Bera1, A. Acín, M. Kuś, M. W. Mitchell, M. Lewenstein, "Randomness in quantum mechanics: philosophy, physics and technology", Reports on Progress in Physics, Volume 80, Number 12, (2017)

[80] X. Ma, X. Yuan, Z. Cao, B. Qi, Z. Zhang, "Quantum random number generation", npj Quantum Information volume 2, Article number: 16021, (2016)

[81] M. Herrero-Collantes, J.C. Garcia-Escartin, "Quantum Random Number Generators", Rev. Mod. Phys. 89, 015004, (2017)

[82] L. E. Bassham, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, S. D. Leigh, M. Levenson, M. Vangel, N. A. Heckert, D L. Banks, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", Special Publication (NIST SP) - 800-22 Rev 1a, (2010)

[83] "DieHarder: A Random Number Test Suite", `https://webhome.phy.duke.edu/~rgb/General/rand_rate/rand_rate.abs`

[84] A. Acín, L. Masanes, "Certified randomness in quantum physics", Nature volume 540, pages 213–219, (2016)

[85] N. Beaudry, "Assumptions in Quantum Cryptography", PhD thesis, Diss. ETH No. 22269 (2015)

[86] A. Kerckhoffs, "La cryptographie militaire," Journal des sciences militaires IX, 5–83, (1883)

[87] S. Pironio, A. Acín, S. Massar, A. Boyer de la Giroday, D. N. Matsukevich, P. Maunz, S. Olmschenk, D. Hayes, L. Luo, T. A. Manning, C. Monroe , "Random numbers certified by Bell's theorem", Nature volume 464, pages1021–1024, (2010)

[88] B. G. Christensen, K. T. McCusker, J. B. Altepeter, B. Calkins, T. Gerrits, A. E. Lita, A. Miller, L. K. Shalm, Y. Zhang, S. W. Nam, N. Brunner, C. C. W. Lim, N. Gisin, P. G. Kwiat, "Detection-Loophole-Free Test of Quantum Nonlocality, and Applications", Phys. Rev. Lett. 111, 130406, (2013)

[89] D. G. Marangon, G. Vallone, P. Villoresi, "Source-Device-Independent Ultrafast Quantum Random Number Generation", Phys. Rev. Lett. 118, 060503, (2017)

[90] T. Michel, J. Y. Haw, D. G. Marangon, O. Thearle, G. Vallone, P. Villoresi, P. K. Lam, S. M. Assad, "Real-Time Source Independent Quantum Random Number Generator with Squeezed States", arXiv:1903.01071v1 [quant-ph], (2019)

[91] F. Bischof, H. Kampermann, D. Bruß, "Measurement-device-independent randomness generation with arbitrary quantum states", Phys. Rev. A 95, 062305, (2017)

[92] Z. Cao, H. Zhou, X. Ma, "Loss-tolerant measurement-device-independent quantum random number generation", New Journal of Physics, Volume 17, (2015)

[93] R. Kumar, E. Barrios, A. MacRae, E. Cairns, E.H. Huntington, A.I. Lvovsky, "Versatile wideband balanced detector for quantum optical homodyne tomography", Optics Communications, Volume 285, Issue 24, (2012)

[94] Y.M Chi, B. Qi, W. Zhu, L. Qian, H.-K. Lo, S.-H. Youn, A. I. Lvovsky, L. Tian, "A balanced homodyne detector for high-rate Gaussian-modulated coherent-state quantum key distribution", New Journal of Physics, Volume 13, (2011)

[95] C. Gabriel, C. Wittmann, D. Sych, R. Dong, W. Mauerer, U. L. Andersen, C. Marquardt, G. Leuchs, "A generator for unique quantum random numbers based on vacuum states", Nature Photonics volume 4, pages 711–715, (2010)

[96] T. Symul, S.M. Assad, P.K. Lam, "Real time demonstration of high bitrate quantum random number generation with coherent laser light", Appl. Phys. Lett. 98, 231103, (2011)

[97] Y. Shi, B. Chng, C. Kurtsiefer, "Random numbers from vacuum fluctuations", Appl. Phys. Lett. 109, 041101, (2016)

[98] Y. Shen, L. Tian, H. Zou, "Practical quantum random number generator based on measuring the shot noise of vacuum states", Phys. Rev. A 81, 063814, (2010)

[99] Y. Nie, L. Huang, Y. Liu, F. Payne, J Zhang, J. Pan, "The generation of 68 Gbps quantum random number by measuring laser phase fluctuations", Review of Scientific Instruments 86, 063105,(2015)

[100] H. Fürst, H. Weier, S. Nauerth, D. G. Marangon, C. Kurtsiefer, H. Weinfurter, "High speed optical quantum random number generation," Opt. Express 18, 13029-13037, (2010)

[101] F. Xu, B. Qi, X. Ma, H. Xu, H. Zheng, H.K. Lo,"Ultrafast quantum random number generation based on quantum phase fluctuations", Opt. Express 20, 12366-12377, (2012)

[102] J.Y. Haw, S.M. Assad, A.M. Lance, N.H.Y. Ng, V. Sharma, P.K. Lam, T. Symul, "Maximization of Extractable Randomness in a Quantum Random-Number Generator", Phys. Rev. Applied 3, 054004, (2015)

[103] C. Abellán, W. Amaya, M. Jofre, M. Curty, A. Acín, J. Capmany, V. Pruneri, M. W. Mitchell, "Ultra-fast quantum randomness generation by accelerated phase diffusion in a pulsed laser diode", Opt. Express 22, 1645-1654, (2014)

[104] M. W. Mitchell, C. Abellan, W. Amaya, "Strong experimental guarantees in ultrafast quantum random number generation", Phys. Rev. A 91, 012314 – Published, (2015)

[105] T. Gehring, C. Lupo, A. Kordts, D. Solar Nikolic, N. Jain, T. B. Pedersen, S. Pirandola, U. L. Andersen, "8 GBit/s real-time quantum random number generator with non-iid samples", arXiv:1812.05377 [quant-ph]

[106] M.S. Bartlett, "Smoothing Periodograms from Time-Series with Continuous Spectra", Nature volume 161, pages 686–687, (1948)

[107] J.L. Carter, M.N. Wegman, "Universal Classes of Hash Functions", JCSS, 18, pp. 143–154, (1979)

[108] H. Krawczyk, "LFSR-based Hashing and Authentication", Advances in Cryptology — CRYPTO '94, (1994)

[109] Y. Mansour, N. Nisan, P. Tiwari, "The Computational Complexity of Universal Hashing", STOC' 90, pp. 235–243, (1990)

[110] R. M. Gray, "Toeplitz and Circulant Matrices: A review", Foundations and Trends in Communications and Information Theory, Vol. 2: No. 3, pp 155-239, (2006)

[111] O. Chevassut, P.-A. Fouque, P. Gaudry, D. Pointcheval, "The Twist-Augmented Technique for Key Exchange", Proceedings of Public Key Cryptography-PKC 2006, pp. 410 - 426, (2006)

[112] J. Liu, J. Yang, Z. Li, Q. Su, W. Huang, B. Xu, H. Guo "117 Gbits/s Quantum Random Number Generation With Simple Structure", IEEE Photonics Technology Letters, vol. 29, no. 3, pp. 283-286, (2017)

[113] X. Ma, F. Xu, H. Xu, X. Tan, B. Qi, H.-K. Lo, "Postprocessing for quantum random-number generators: Entropy evaluation and randomness extraction", Phys. Rev. A 87, 062327, (2013)

[114] H. Krawczyk, "Cryptographic Extraction and Key Derivation: The HKDF Scheme", Proceedings of Advances in Cryptology – CRYPTO 2010, pp. 631 – 648, (2010)

[115] M. N. Wegman, J.L. Carter, "New hash functions and their use in authentication and set equality", Journal of Computer and System Sciences, Volume 22, Issue 3, 1981, Pages 265-279, (1981)

[116] H. Krawczyk, "New Hash Functions for Message Authentication", Advances in Cryptology — EUROCRYPT '95, (1995)

[117] M. Canale, "Classical processing algorithms for Quantum Information Security: PhD thesis", University of Padova, (2014)

[118] R. Renner, "Universally Composable Privacy Amplification Against Quantum Adversaries", Proc. of TCC 2005, LNCS, Springer, vol. 3378, (2005)

[119] M. Tomamichel, C. Schaffner, A. Smith and R. Renner, "Leftover Hashing Against Quantum Side Information," in IEEE Transactions on Information Theory, vol. 57, no. 8, pp. 5524-5535, (2011)

[120] D. R. Stinson, "Universal hash families and the leftover hash lemma and applications to cryptography and computing", J. Combin. Math. Combin. Comput., vol. 42, pp. 3-31, (2002)

[121] R. Impagliazzo, D. Zuckerman, "How to recycle random bits," 30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, NC, 1989, pp. 248-253, (1989)

[122] R. Konig, R. Renner, "Sampling of Min-Entropy Relative to Quantum Knowledge," in IEEE Transactions on Information Theory, vol. 57, no. 7, pp. 4760-4787, (2011)

[123] Z. Zheng, Y. Zhang, W. Huang, S. Yu, H. Guo, "6 Gbps real-time optical quantum random number generator based on vacuum fluctuation", Review of Scientific Instruments 90, 043105, (2019)

[124] L. Huang, H. Zhou, "Integrated Gbps quantum random number generator with real-time extraction based on homodyne detection", J. Opt. Soc. Am. B 36, B130-B136, (2019)

[125] X. Zhang, Y. Nie, H. Liang, J. Zhang, "FPGA implementation of Toeplitz hashing extractor for real time post-processing of raw random numbers," 2016 IEEE-NPSS Real Time Conference (RT), Padua, (2016)

[126] X.G. Zhang, Y.Q. Nie, H. Zhou, H. Liang, X. Ma, J. Zhang, J.W. Pan, "Fully integrated 3.2 Gbps quantum random number generator with real-time extraction", Rev. Sci. Instrum. 87, 076102, (2016)

[127] P. Jouguet, S. Kunz-Jacques, E. Diamanti, A. Leverrier, "Analysis of imperfections in practical continuous-variable quantum key distribution", Phys. Rev. A 86, 032309, (2012)

[128] R. Canetti, "Universally composable security: a new paradigm for cryptographic protocols," Proceedings 42nd IEEE Symposium on Foundations of Computer Science, Newport Beach, CA, USA, 2001, pp. 136-145, (2001)

[129] "White paper: Floating vs Fixed Point", Xilinx, WP491, (2017) `https://www.xilinx.com/support/documentation/white_papers/wp491-floating-to-fixed-point.pdf`

[130] E. Kaur, S. Guha, M. M. Wilde, "Asymptotic security of discrete-modulation protocols for continuous-variable quantum key distribution", arXiv:1901.10099, (2019).

[131] K. Temme, M. J. Kastoryano, M. B. Ruskai, M. M. Wolf, F. Verstraete, "The χ^2 -divergence and mixing times of quantum Markov processes" Journal of Mathematical Physics 51, 122201, (2010)

[132] M.E. Shirokov, "Tight uniform continuity bounds for the quantum conditional mutual information, for the Holevo quantity, and for capacities of quantum channels", Journal of Mathematical Physics 58, 102202, (2017)

[133] D. B. Thomas, "FPGA Gaussian Random Number Generators with Guaranteed Statistical Accuracy", IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines, pp. 149-156, (2014)

[134] D. B. Thomas, "Parallel Generation of Gaussian Random Numbers Using the Table-Hadamard Transform", IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines, pp. 161-168, (2013)

[135] R. C. C. Cheung, D. Lee, W. Luk and J. D. Villasenor, "Hardware Generation of Arbitrary Random Number Distributions From Uniform Distributions Via the Inversion Method", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 15, no. 8, pp. 952-962, (2007)

[136] Y. Li, P. Chow, J. Jiang, M. Zhang and S. Wei, "Software/hardware framework for generating parallel Gaussian random numbers based on the Monty Python method", International Conference on Field-Programmable Technology, pp. 190-197, (2012)

[137] D. -. Lee, J. D. Villasenor, W. Luk and P. H. W. Leong, "A hardware Gaussian noise generator using the Box-Muller method and its error analysis", IEEE Transactions on Computers, vol. 55, no. 6, pp. 659-671, (2006)

[138] B. G. Sileshi, C. Ferrer and J. Oliver, "Accelerating hardware Gaussian random number generation using Ziggurat and CORDIC algorithms", SENSORS, 2014 IEEE, Valencia, pp. 2122-2125, (2014)

[139] H. Edrees, B. Cheung, M. Sandora, D. Nummey, D. Stefan, "Hardware-Optimized Ziggurat Algorithm for High-Speed Gaussian Random Number Generators", Proceedings of the 2009 International Conference on Engineering of Reconfigurable Systems Algorithms, ERSA 2009, (2009)

[140] Guanglie Zhang, P. H. W. Leong, Dong-U Lee, J. D. Villasenor, R. C. C. Cheung and W. Luk, "Ziggurat-based hardware Gaussian random number generator", International Conference on Field Programmable Logic and Applications, pp. 275-280, (2005)

[141] S.S.Roy, F. Vercauteren, I. Verbauwhede, "High Precision Discrete Gaussian Sampling on FPGAs", Revised Selected Papers on Selected Areas in Cryptography, SAC 2013 - Volume 8282, (2013)

[142] T. Pöppelmann, L. Ducas, T. Güneysu, "Enhanced Lattice-Based Signatures on Reconfigurable Hardware", Cryptographic Hardware and Embedded Systems – CHES 2014, (2014)

[143] C. Du, G. Bai, "Towards efficient discrete Gaussian sampling for lattice-based cryptography", 25th International Conference on Field Programmable Logic and Applications (FPL), pp. 1-6, (2015)

[144] J. Buchmann, D. Cabarcas, F. Göpfert, A. Hülsing, P. Weiden, "Discrete Ziggurat: A Time-Memory Trade-Off for Sampling from a Gaussian Distribution over the Integers", Selected Areas in Cryptography SAC 2013, (2014)

[145] N. C. Dwarakanath, S. D. Galbraith, "Sampling from discrete Gaussians for lattice-based cryptography on a constrained device", Applicable Algebra in Engineering, Communication and Computing, Volume 25, Issue 3, pp 159–180, (2014)

[146] C. Aguilar-Melchor, M. R. Albrecht, T. Ricosset, "Sampling from Arbitrary Centered Discrete Gaussians for Lattice-Based Cryptography", International Conference on Applied Cryptography and Network Security ACNS 2017, (2017)

[147] S.S. Roy, O. Reparaz, F. Vercauteren, I. Verbauwhede, "Compact and Side Channel Resistant Discrete Gaussian Sampling", IACR Cryptology ePrint Archive, vol. 2014, p. 591, (2014)

[148] G. Marsaglia, W.W. Tsang, "The Ziggurat Method for Generating Random Variables", Journal of Statistical Software, Foundation for Open Access Statistics, vol. 5(i08), (2000)

[149] J. Howe, A. Khalid, C. Rafferty, F. Regazzoni and M. O'Neill, "On Practical Discrete Gaussian Samplers for Lattice-Based Cryptography", IEEE Transactions on Computers, vol. 67, no. 3, pp. 322-334, (2018)

[150] R. Toral, A. Chakrabarti, "Generation of Gaussian distributed random numbers by using a numerical inversion method", Computer Physics Communications, Volume 74, Issue 3, (1993)

[151] K. Pearson, "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling", Philosophical Magazine. Series 5. 50 (302): 157–175, (1900)

[152] K.Fazel, S.Keiser, "Multi carrier and spread spectrum system", John Willey and Sons, (2003)

[153] "AXI Reference Guide", Xilinx, UG1037, (2017) `https://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/latest/ug1037-vivado-axi-reference-guide.pdf`

[154] "AMBA 4 AXI4-Stream Protocol", ARM, (2010) `https://static.docs.arm.com/ihi0051/a/IHI0051A_amba4_axi4_stream_v1_0_protocol_spec.pdf`

[155] "FIFO Generator v13.1 LogiCORE IP Product Guide", Xilinx, PG057, (2017) https://www.xilinx.com/support/documentation/ip_documentation/fifo_generator/v13_1/pg057-fifo-generator.pdf

[156] "Aurora 8B/10B v11.0 LogiCORE IP Product Guide", Xilinx, PG046, (2016) https://www.xilinx.com/support/documentation/ip_documentation/aurora_8b10b/v11_0/pg046-aurora-8b10b.pdf

[157] A. X. Widmer, P. A. Franaszek, "A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code," in IBM Journal of Research and Development, vol. 27, no. 5, pp. 440-451, (1983)

[158] "DDS Compiler v6.0 LogiCORE IP Product Guide", Xilinx, PG141, (2017) https://www.xilinx.com/support/documentation/ip_documentation/dds_compiler/v6_0/pg141-dds-compiler.pdf

[159] "FIR Compiler v7.2 LogiCORE IP Product Guide", Xilinx, PG149, (2015) https://www.xilinx.com/support/documentation/ip_documentation/fir_compiler/v7_2/pg149-fir-compiler.pdf

[160] "Fast Fourier Transform v9.0 LogiCORE IP Product Guide", Xilinx, PG109, (2017) https://www.xilinx.com/support/documentation/ip_documentation/xfft/v9_0/pg109-xfft.pdf

[161] "Si5338, I2C programmable any-frequency any-output quad clock generator", SIlicon labs, (2015) https://www.silabs.com/documents/public/data-sheets/Si5338.pdf

[162] "DAC39J84 Quad-Channel,16-Bit,2.8 GSPS,Digital-to-Analog Converter with 12.5 Gbps JESD204B Interface", Texas Instruments, (2014) https://www.ti.com/lit/ds/symlink/dac39j84.pdf