UNIVERSITY OF PADOVA

DEPARTMENT OF INFORMATION ENGINEERING

PH.D. SCHOOL IN INFORMATION ENGINEERING
INFORMATION SCIENCE AND TECHNOLOGY
XXXI CLASS

High Performances Systems for Applications of Quantum Information

---

Ph.D. Candidate:
Andrea STANCO

Supervisor:
Ch.mo Prof. Paolo VILLORESI

Course coordinator:
Ch.mo Prof. Andrea NEVIANI

"It would be possible to describe everything scientifically, but it would make no sense; it would be without meaning, as if you described a Beethoven symphony as a variation of wave pressure."

Albert Einstein

Abstract

This thesis work is about the realization of hardware and software systems for Quantum Random Number Generation (QRNG) and Quantum Key Distribution (QKD). Such systems were developed to guarantee a full functionality for a broader investigation of these two cutting edge applications of Quantum Information field. The thesis describes in details both the hardware and the software that were developed for FPGA-CPU board, Time-to-Digital converter (TDC) devices and computers, along with QRNG and QKD specific applications and their results.

Randy was the first FPGA-based QRNG device to be developed; it uses a light source attenuated to single-photon level and one single-photon avalanche diode (SPAD). From the sampling of the SPAD electrical signal, the device produces random numbers through dedicated generation protocols and through the Peres unbiasing algorithm in order to maximize the output generation bit rate. Furthermore, the device allows to generate real time random numbers. This feature is used for the time setting of electro-optical components for extending Wheeler's delayed-choice experiment to space.

The same techniques were applied to a second device, *LinoSPAD*; it combines an FPGA-chip and a CMOS-SPADs array. Moreover, in this device, a TDC improves the photon detection time accuracy. Along with a dedicated post-processing based on Zhou-Bruk algorithm, the TDC allowed to reach a final bit rate equivalent to 300 Mbit/s.

As far as QKD systems are concerned, within the collaboration among the University of Padova, the Italian Space Agency (ASI) with the Matera Laser Ranging Observatory (MLRO) and the Chinese Academy of Sciences (CAS) a TDC device management software was developed. The project aim is to realize a quantum cryptographic key exchange between the Chinese satellite Micius and MLRO. The software was designed to manage the entire data acquisition synchronized with UTC time. Furthermore, another software was designed to deal with electro-optomechanical and electro-optical components. The software is aim at the time-variant compensation of the beam angular changes through the optical path.

Once again, within a collaboration between ASI and University of Padova, a full free space QKD system over tenths of kilometers was developed. It required the design of various components. This work describes the QKD source along with the dedicated FPGA board design. Such board generates the electrical impulses to control the qubit laser along with the electro-optic phase and intensity modulators.

Sommario

Il presente lavoro di tesi tratta la realizzazione di sistemi hardware e software per Quantum Random Number Generation (QRNG) e Quantum Key Distribution (QKD). Tali sistemi sono stati sviluppati al fine di garantire una completa funzionalità per l'investigazione a tutto campo di queste due applicazioni che ad oggi risultano essere le più promettenti nell'ambito della Quantum Information. Vengono presentati in dettaglio sia l'hardware sia i software utilizzati che sono stati sviluppati per schede FPGA-CPU, dispositivi di Time-to-Digital converter (TDC) e computer. Vengono inoltre descritte le applicazioni specifiche di QRNG e QKD assieme ai risultati ottenuti.

Randy è stato il primo dispositivo QRNG sviluppato su scheda FPGA e utilizza una sorgente luminosa attenuata a singolo fotone e un single-photon avalanche diode (SPAD). A partire dal campionamento del segnale elettrico dello SPAD, il dispositivo produce numeri randomici tramite protocolli di generazione appositi e tramite l'applicazione dell'algoritmo di unbiasing di Peres per massimizzare il bit rate. Il dispositivo permette inoltre di generare numeri randomici in tempo reale. Questa caratteristica viene utilizzata per la gestione temporizzata di componenti elettro-ottici per l'estensione allo spazio dell'esperimento a scelta ritardata di Wheeler's.

Le stesse tecniche sono state in seguito applicate ad un secondo dispositivo, *LinoSPAD*, che integra un chip FPGA e un array di CMOS-SPAD. Tale dispositivo prevede inoltre un TDC per aumentare la precisione temporale di dectection dei fotoni. Questa caratteristica, unita all'uso di una procedure di post-processing appositamente sviluppata e basata sull'algoritmo di Zhou-Bruk, ha permesso di raggiungere un bit rate finale pari a 300 Mbit/s.

Per quanto riguarda i sistemi QKD, all'interno di un progetto di collaborazione tra l'Università di Padova, l'Agenzia Spaziale Italiana (ASI) insieme al Matera Laser Ranging Observatory (MLRO) e la Chinese Academy of Sciences (CAS) è stato sviluppato un software di gestione di un dispositivo TDC. Il progetto prevede la realizzazione di uno scambio di chiave crittografica quantistica tra il satellite cinese Micius e l'osservatorio di Matera. Il software è stato progettato per la gestione dell'intera acquisizione dati sincronizzata al tempo UTC. Inoltre è stato sviluppato anche un software per la gestione di componenti elettro-optomeccanici e elettro-ottici atti alla compensazione tempo variante delle variazioni angolari del fascio nel percorso ottico.

Sempre all'interno di una collaborazione tra ASI e Università di Padova, è stato

sviluppato un sistema completo di QKD free space per distanze nell'ordine di decine di chilometri. Lo sviluppo del sistema ha richiesto la progettazione di molteplici componenti. In questo lavoro viene descritta la parte della sorgente QKD e quindi della progettazione della scheda FPGA dedicata. Tale scheda ha il compito di generare gli impulsi elettrici per il controllo del laser per la produzione dei qubit e per il controllo dei modulatori di fase e di intensità elettro-ottici.

Ringraziamenti

Prima dell'inizio della descrizione del lavoro di tesi desidero fare alcuni dovuti ringraziamenti.

Il primo ringraziamento va al mio supervisore, il professor Paolo Villoresi. Anni fa, quando decisi di svolgere la mia tesi di laurea magistrale presso il suo gruppo di ricerca, non immaginavo cosa tale scelta avrebbe comportato. Gli anni passati al Luxor prima come tesista, poi come assegnista e infine come dottorando sono stati degli anni di incredibile scoperta e crescita sotto ogni aspetto.

Il secondo ringraziamento va al dottor Giuseppe Vallone, in arte Pino, che con la sua incredibile intuitività e immediatezza ha sempre rappresentato un saldo punto di riferimento a cui affidarsi.

In terzo luogo vorrei ringraziare i due post-doc che mi hanno fatto da mentori durante il percorso. Davide Giacomo Marangon è stato un collega prezioso, oltre che un amico, con cui ho condiviso momenti di lavoro e di discussione che mi hanno letteralmente aperto gli occhi e sono stati uno stimolo importante per la realizzazione della mia ricerca di dottorato. Daniele Dequal mi ha invece insegnato che le cose si possono realizzare senza doversi perdere in inutili paranoie.

Mi sembra doveroso ringraziare anche tutto il resto della ciurma del Luxor che è stata la mia casa per tutti questi anni. Ringraziarvi uno per uno mi è veramente impossibile ma vi porto tutti nel cuore, chi più chi meno (Costa tu sei tra i meno, ovviamente).

Infine desidero ringraziare tutta la mia famiglia per il sostegno che mi ha sempre dimostrato e per l'immancabile affetto. Ringrazio mio padre e mia madre per avermi fatto conoscere il mondo della ricerca (senza la loro storia non so se avrei mai fatto il dottorato) e mio fratello e mia sorella perchè sì.

In ultimo il ringraziamento più importante va a mia moglie Elena che ogni giorno mi ricorda che la risata è il motore che fa muovere l'essere umano e perché senza di lei le mie giornate sarebbero grigie.

Contents

List of Figures

List of Tables

List of Abbreviations

AP	**A**fterpulse
AXI	**A**genzia **S**paziale **I**taliana
AXI	**A**dvanced e**X**tensible **I**nterface
BEL	**B**asic **EL**ement
CLB	**C**onfigurable **L**ogic **B**lock
DT	**D**ead **T**ime
FM	**F**araday **M**irror
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
FSM	**F**inite **S**tate **M**achine
GPS	**G**lobal **P**ositioning **S**ystem
GUI	**G**raphical **U**ser **I**nterface
HDL	**H**ardware **D**escription **L**anguage
IDE	**I**ntegrated **D**evelopment **E**nvironment
LUT	**L**ook**U**p **T**able
MLRO	**M**atera **L**aser **R**anging **O**bservatory
MZI	**M**ach-**Z**ehnder **I**nterferometer
NTP	**N**etwork **T**ime **P**rotocol
PL	**P**rogrammable **L**ogic
PMF	**P**olarization **M**aintaining **F**iber
PPS	**P**ulse **P**er **S**econd
PS	**P**rocessing **S**ystem
QBER	**Q**uantum **B**it **E**rror **R**ate
QI	**Q**uantum **I**nformation
QKD	**Q**uantum **K**ey **D**istribution
QRNG	**Q**uantum **R**andom **N**umber **G**eneration
RTT	**R**ound **T**rip **T**ime
SNTP	**S**imple **N**etwork **T**ime **P**rotocol
SoC	**S**ystem **o**n a **C**hip
SPAD	**S**ingle **P**hoton **A**valanche **D**iode
SPD	**S**ingle **P**hoton **D**etector
TCP	**T**ransmission **C**ontrol **P**rotocol
TRNG	**T**rue **R**andom **N**umber **G**eneration
UTC	**C**oordinated **U**niversal **T**ime
VHDL	**V**HSIC **H**ardware **D**escription **L**anguage

Introduction

Quantum Information (QI) finds its roots in the previous century but only in the last decade most of its applications have reached the notoriety. *Quantum Random Number Generation* (QRNG) and *Quantum Key Distribution* (QKD) are two of the most promising applications of QI. QRNG allows to produce true random numbers that can be used in many different fields like lotteries, Montecarlo analysis, cryptography and quantum mechanics experiments. As a matter of fact, the validity of such application relies on true randomness. Nowadays, the realization of a QRNG device does have reasonable costs and can be implemented through different technological solutions. This is the reason why QRNG, at this moment, is almost ready for a massive commercial spreading. On the other hand, QKD promises to solve every cryptographic issues in telecommunication. In principle, it can guarantee unconditional security over every transmission, i.e. an external attacker indeed has no chances to decrypt the secret message. Nevertheless, the practical realization of a QKD system requires expensive parts and fine calibrations which anyway allows limited key bit rate. Moreover, bringing the QKD in every day communications implies the building of dedicated infrastructures, which requires the involvement of different institutions and organizations outside the academic world. Therefore, QKD is yet to be consider a usable technology. The next 10-year challenge is to break through these hindrances and claim the QKD as the worldwide adopted cryptographic paradigm. In this framework, satellite QKD truly plays a major role since it can give a turn in the arrangement of the worldwide QKD net.

Such scientific investigations, which aim at the achievement of new accomplishments, often go hand in hand with the creation of new technological solutions which meet the needs of the investigation itself. Thus, the realization of QRNG and QKD systems relies on the design of dedicated technological supports. *Field Programmable Gate Array* (FPGA) gives the possibility to deal with the hardware at low level and to have full control over any apparatus. In fact, it becomes more and more essential to most of the high precision scientific experiments. Moreover, its integration with a software counterpart, the so-called *System on a Chip* (SoC), is indeed a matter of fact and gives almost no limits to what can be done. The integration between hardware and software is also exploited by many different commercial devices which are designed for advanced and cutting-edge applications. The programming and calibration of such devices can also be essential to scientific investigations.

This thesis work represents a linking point between these two aforementioned topics: hardware and software systems at the service of Quantum Information.

The work is divided in three parts: Part I describes the basics of the hardware and software designs that were used in the QI applications; Part II focuses on two different QRNG applications while Part III on two different QKD applications.

In Part I, chapter 1 gives a brief description of the FPGA and SoC technologies along with a description of the ZedBoard development board and the development enviroments (Vivado and SDK by Xilinx). In chapter 2 the main structure design and the development solutions are described. The system will be adapted to a QRNG application, described in chapter 5, and to the Wheeler's delayed-choice experiment for the investigation of the quantum nature of light. In chapter 3 an introduction on the quTAU TDC device is given along with the description of a performing management software, Quntroller, developed on Qt IDE. This software will be used for the satellite QKD experiment described in chapter 9.

Part II talks about Quantum Random Number Generation applications. Chapter 4 introduces the topic and explains some of the generation protocols, based on the photon time of arrival, and algorithms that can be used to produce random numbers. In chapter 5 the QRNG device called *Randy* is described. The device is FPGA-based and was realized from the ZedBoard system described in chapter 2. Its optical setup comprehends a light source attenuated to single-photon level and a single-photon avalanche diode (SPAD) which output is sampled by the board. The device allows to choose among different protocols and to have a full control over the generation process. A new generation approach is investigated and applies the Peres unbiasing algorithm directly to the sampled string after the treatment of SPAD non idealities known as deat time and afterpulse. Under the condition of a photon rate equivalent to 200 kcounts/s and a sampling rate of 100 MHz, this approach led to a final bit rate of 1.8 Mbit/s. Through a dedicated modification, the real time feature of Randy was used to generate timed random numbers used to set specific optical components in the Wheeler's delayed-choice experiment extended to space. The experiment, described in chapter 6, was part of a collaboration with the Agenzia Spaziale Italiana (ASI) and was realized at the Matera Laser Ranging Observatory (MLRO). The experiment investigated the dual nature of light through the realization of a Mach-Zehnder interferometer (MZI) extended to space. Chapter 7 describes the analysis conducted on a second device, *LinoSPAD*, produced by the AQUA lab at Delft University & EPFL. The device features 256 CMOS-SPAD pixels arranged in four linear array of 64 pixels each as well as an FPGA TDC which gives a time precision of $\sim$17.9 ps. A dedicated post-processing procedure was developed to exploit LinoSPAD as a QRNG. Such procedure, based on the Peres and Zhou-Bruk algorithms, takes into account the presence of known SPAD non idealities and also deals with the FPGA TDC non linear behavior. The final generation bit rate is equivalent to 300 Mbit/s.

Part III is dedicated to the description of two QKD applications. The first one is part of a collaboration with ASI and the Chinese Academy of Sciences (CAS) and is described in chapter 9. The objective is the realization of a QKD link between the Chinese satellite Micius and MLRO. A preliminary test was conducted in July 2018

to investigate the feasibility of such link and the capability to distinguish between a light polarization and another. The test required a proper setting of the Quntroller software for acquiring the significant data from the setup. The software was also updated to synchronized the acquisition to the UTC time, which required a dedicated GPS receiver. Moreover, a brand new software was developed in order to keep the optical axis alignment between Micius satellite and the MLRO telescope. The software controlled electro-optomechanical and electro-optical components which implemented a time-variant compensation of the beam angular changes through the optical path. The second application is described in chapter 10. The main objective is to realize a full free-space QKD system over a distance of tenths of kilometers through the synthesization of qubits as polarized single photons. The project was developed by the Quantum Future group as a whole and required the design of various parts in order to implement the QKD system from A to Z. The chapter only describes a dedicated design developed for the FPGA board, once again the ZedBoard, at the transmitter. In the QKD source, the qubit is produced by a DFB laser attenuated to single-photon level which is driven by an electrical signal from the FPGA. Since the optical pulse width is directly proportional to the electrical pulse width, a smart FPGA design is used to shorten this width and thus reduce the time slot which the photon is emitted in. From two 100 MHz signals is produced a 350 ps pulse which is at the limit of the physical analog bandwidth of the board. As a directly consequence, a shorter time slot implies a better capability to discriminate the qubit over the background noise leading to a lower QBER and hence to a higher secret key bit rate. Similar techniques are applied to drive the electro-optical phase and intensity modulator to set the photon polarization and implement the *decoy*.

Part I

Hardware and software designs

Chapter 1

SoC system and related tools

This chapter gives an introduction to FPGA and SoC technology and the ZedBoard development board. It also gives an overview of the software tools and technical knowledge used to program and design the board.

1.1 Introduction to FPGA and SoC

Field Programmable Gate Array (FPGA) technology was born in 1980s and allows the configuration of an electronic circuit through an Hardware Description Language (HDL). It comprehends a fabric of Configurable Logical Blocks (CLBs) embedded in a huge array of interconnecting wiring. Usually, CLBs contain logic elements like Lookup Tables (LUTs), multiplexers, flip-flops and memory elements. The configuration program is stored in a dedicated memory which can be SRAM, Antifuse or EPROM [1]. The invention of FPGA is generally attributed to Ross Freeman [2] who was co-founder of Xilinx company in 1984. FPGA is considered to be an evolution of prior Programmable Logic Device (PLD) technologies developed during the 1970s [3, Chapter 11]. During the 1990s, FPGA was mainly used for telecommunication and networking applications and then it rapidly spread into industrial and consumer ones. At the end of 2000s the CPU evolution trend found one of its limits. Increasing the clock frequency in order to achieve higher performances was not possible anymore: power consumption started to become way too high and other solutions needed to be investigated. While GPU technology could indeed represent a solution offering powerful advantages, FPGA technology had a major spread thanks to its high computational power and task-parallelization keeping its power consumption within reasonable limits [4, 5]. Furthermore, boards with FPGA chip began to integrate additional technologies, which enlarged its range of possible applications and brought it into contact with new users communities. A huge achievement was the integration of FPGA and CPU which brought together the best of both worlds: high performances along with the easiness to integrate huge softwares. This solution is part of the *System on a Chip* (SoC) category and today is universally adopted [6–9]. A valid alternative is represented by the so-called *Soft-Core processors* which are hardware-described-processors implemented directly on an FPGA chip [10]. Microblaze and Picoblaze are two well-known examples of this

technology [11–13]. Although they can be customized for specific applications and thus shortner task-to-complete time, they also require additional design efforts and do not have the high level flexibility of SoC. Furthermore, the power consumption can also be an issue. For a complete comparison between these two technologies please refer to [14]. Since our applications do not require particular or critical processor functionalities, SoC was preferred over Soft-Core.

The two main producers of FPGA chip are *Xilinx* company and *Intel-Altera* company; both offer a wide range of solutions fitting the applications required by the market. Nowadays, applications can vary from *cloud management* [15] to *high performances application for space environment* [16, 17], from *neural networks* [18, 19] to *machine learning* [20, 21] and from *cryptography* [22–26] to *random number generator* [27–31]. One of the FPGA main advantages that can be exploited in the scientific investigation is its deterministic behavior. Every sequential operation defined in the design can be temporally described as a multiple of the time period of a *clock* signal. Within this framework, the term *latency* assumes a different meaning with respect to the software world: a general operation has a known latency perfectly defined and cannot change. Moreover, the magnitude order of such latency is well below the performances of any software-based technology. Therefore, this feature can have fundamental role in every application that requires very precise timing and synchronization. Moreover, the FPGA integration in a SoC gives the possibility to easily expand the number of tests that can be conducted on a setup since system parameters can be changed through the software counterpart.

1.2 ZedBoard development board

At this moment, the FPGA boards market can offer a great variety of solutions. The cutting-edge technology is represented by the Ultrascale chip families which are realized in 16 and 20 nm process and offer the possibility to realize architectures with incredible computational power. Clearly, such solutions have a certain cost since they are usually mounted on dedicated board with high connectivity pin ports, large memories, transceiver modules and other high performance components. Nevertheless, these products can be an overkill for many applications. Hence, a low cost board could fit the specifications required for the realization of the first prototypes of the QRNG and QKD applications described in this work. Furthermore, the developed designs can be easily adapted to the Ultrascale architectures for future improvements of the applications. For this reason, the chosen board was the *ZedBoard* development board.

Released in 2013, the ZedBoard (figure 1.1) is a development kit produced by Avnet in collaboration with Digilent and Xilinx. Thanks to presence of the Zynq-7000 SoC which allows high flexibility and low costs, it has remarkably spread over the past few years and had a major role in different applications [32–34]. ZedBoard

elements are listed below and illustrated in 1.2. For a full and detailed description see [35].

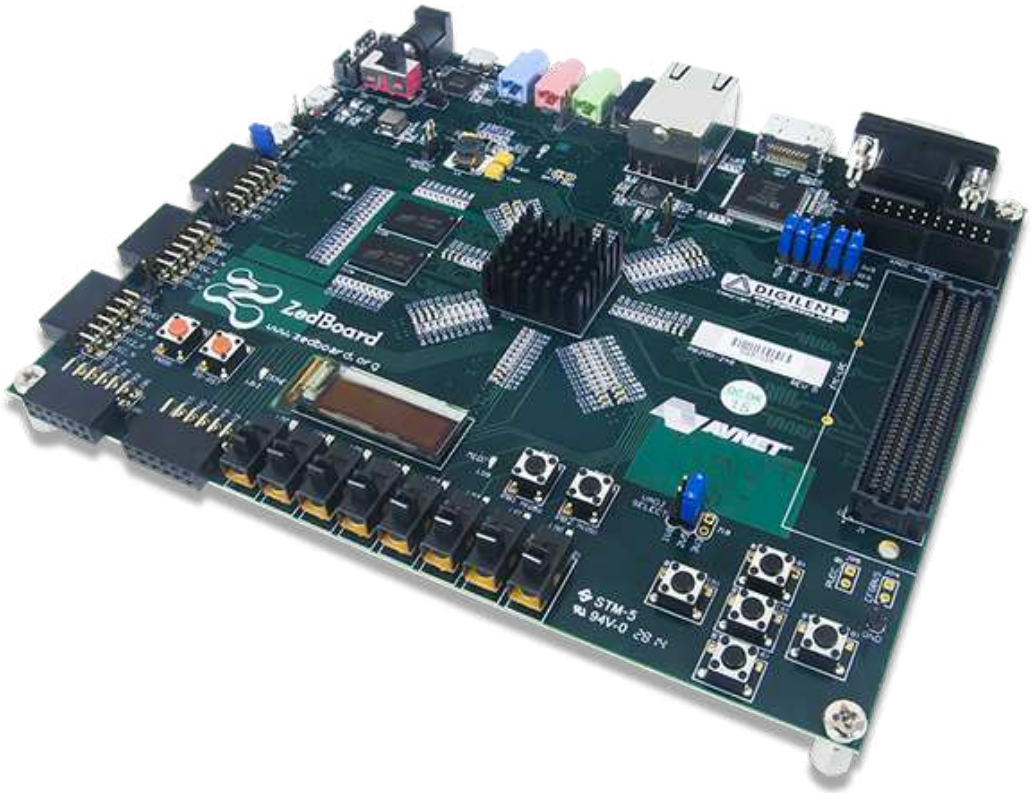

FIGURE 1.1: ZedBoard development kit. Picture by Digilent.

- Xilinx® XC7Z020-1CLG484C Zynq-7000 AP SoC
  - Primary configuration = QSPI Flash
  - Auxiliary configuration options
    - Cascaded JTAG
    - SD Card
- Memory
  - 512 MB DDR3 (128M x 32)
  - 256 Mb QSPI Flash
- Interfaces
  - USB-JTAG Programming using Digilent SMT1-equivalent circuit
    - Accesses PL JTAG
    - PS JTAG pins connected through PS Pmod
  - 10/100/1G Ethernet

  - USB OTG 2.0
  - SD Card
  - USB 2.0 FS USB-UART bridge
  - Five Digilent Pmod™ compatible headers (2x6) (1 PS, 4 PL)
  - One LPC FMC
  - One AMS Header
  - Two Reset Buttons (1 PS, 1 PL)
  - Seven Push Buttons (2 PS, 5 PL)
  - Eight dip/slide switches (PL)
  - Nine User LEDs (1 PS, 8 PL)
  - DONE LED (PL)
- On-board Oscillators
  - 33.333 MHz (PS)
  - 100 MHz (PL)
- Display/Audio
  - HDMI Output
  - VGA (12-bit Color)
  - 128x32 OLED Display
  - Audio Line-in, Line-out, headphone, microphone
- Power
  - On/Off Switch
  - 12V @ 5A AC/DC regulator

1.2.1 Zynq-7000 chip

The Zynq-7000 is a SoC family produced by Xilinx company. It integrates ARM-based processor with an FPGA. The XC7Z020-1CLG484C mounted on the ZedBoard has a two-core ARM Cortex A9 and a 28 nm Artix-7 based programmable logic and comprehends 85 thousands logic cells, 200 DSP Slices and 200 I/O pins. These features are suitable for a wide range of applications. The FPGA part is also indicated as *Programmable Logic* (PL) while the CPU is indicated as *Processing System* (PS). This is the preferred nomenclature used throughout this work. Full description and specifications can be found on the official documentation [36, 37].

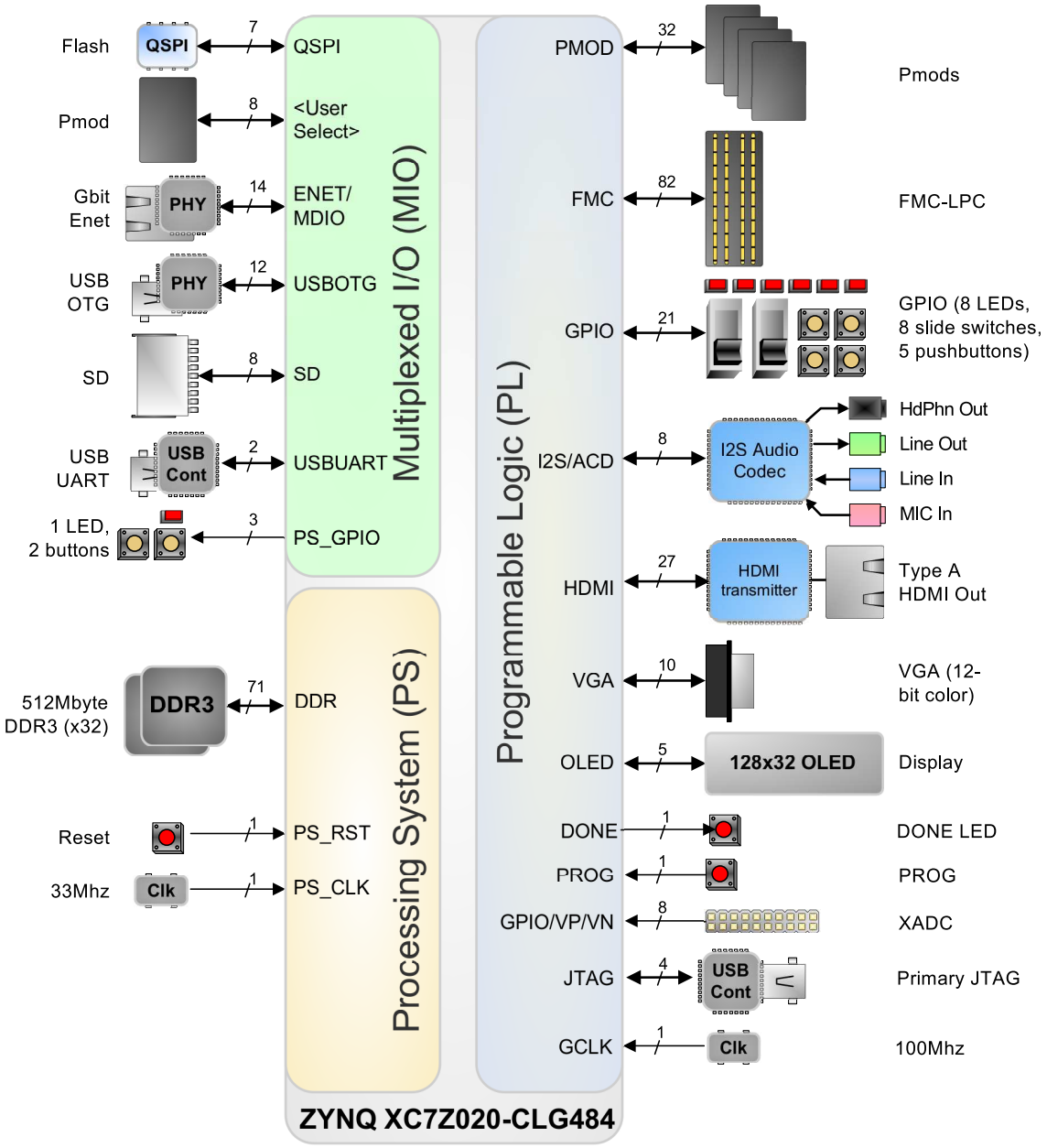

FIGURE 1.2: ZedBoard Block Diagram. Picture by Avnet.

1.3 Vivado Design Suite

Vivado Design Suite is a software suite developed by Xilinx [38] which allows the design of Xilinx FPGA-chip architecture. A chip project can be organized by using the useful *Block Design* which gives a schematic overview of the blocks interconnection and automatizes many procedures. The software offers full control over compilations passages, i.e. *Synthesis*, *Implementation* and *Bitstream Generation*. Furthermore, it includes several IP cores produced by Xilinx, which broadens the design possibilities.

1.4 Xilinx SDK

The Xilinx Software Development Kit (XSDK) [39], also indicated as SDK, is a design environment for creating embedded applications. It is based on the Eclipse framework and allows to create standalone and OS-based applications to be run on a CPU counterpart of a Xilinx SoC. Standalone applications can be programmed in `C` or `C++` language while the OS ones can be Linux-based or FreeRTOS-based [40].

1.5 AXI Protocol

The Advanced eXtensible Interface protocol [41] is a part of ARM AMBA, which is a standard for the blocks connection, management and communication in SoC designs. The main aim is to make easier the handling and the development of multi-processor systems, its controllers and peripherals. This protocol is very robust and guarantees a safe and fast communication between different modules which are linked together with a master-slave interface system. AXI protocol is used by most of the Xilinx IP Cores represents a link between the SDK software with the Vivado hardware design: each AXI-based module has its own driver which can be initialized in the software code calling specific functions. Therefore, this protocol is fundamental for communication with hardware blocks.

1.6 Finite State Machine

A finite state machine, also indicated as FSM, is an important element in the design of sequential logical circuits. Basically, it shapes the behavior of the circuit abstracting from its *state* of operation. Therefore, the circuit moves from a state to another and performs a specific task depending on the state. Furthermore, it organizes the VHDL code in a more intuitive way making the code more readable and easier to understand. For this reason, most of the custom VHDL modules described throughout this work were designed with a dedicated finite state machine.

1.6.1 Mealy and Moore state machine

There are two different types of state machine: Mealy state machine [42] and Moore state machine [43]. In the Mealy machine the output depends both on the state and the input, whereas the Moore one depends only on the state and not on the input. In general, the Mealy state machine requires less states (then less hardware) and can work faster than the Moore one. Nevertheless, its outputs can change asynchronously, which undermines the predictability of working, and its design might be complicated. Since the developed systems required a great accuracy in the execution of their functions and since there were no restrictions on the hardware usage and on the reactivity, only Moore state machines were used.

1.6.2 Coding styles

An FSM can be implemented in VHDL code using different styles [44]. The basic two-processes-single-decoder style creates two state-type signals (the state type defines the list of all the possible states): one for the present state and one for the next state. Hence, the first synchronous process simply assigns the next state to the present state, while the second process describes the behavior of each state (that is the decoder) and assigns the future state to the next state signal. A variant of this style is the two-processes-two-decoders style, which uses two processes: the first one decodes the state transitions and the second one decodes the state to generate the output. In this variant there is only one state-type signal, since the next state signal is not required. The third style is the one-process-one-decoder characterized by one process only in which both state transitions and outputs assignment are decoded. This style is maybe the most difficult to design because requires thinking one clock cycle ahead (the outputs are registered) but deletes any glitch-error possibilities. This last style was chosen for the hardware design.

1.6.3 State encoding

Another important feature of an FSM is how the states are encoded. As a matter of fact, the names assigned to states are fictional and the synthesizer does not waste hardware using lots of bits for each name. Therefore, an encoding operation is required in order to assign a more affordable bits sequence to each name. There are different encoding styles like *gray*, *johnson*, *one-hot* and others. Usually this operation is completely automatic and the synthesizer chooses the best one depending on the state machine. However, Vivado synthesizer allows the user to choose between the different encoding styles to Vivado official documentation [45]. According to [46, 47], the one-hot encoding style is the most recommended for FPGA design and was used all along the described FPGA design.

Chapter 2

FPGA-based system design

In this chapter an overview description of the system is given without focusing on any possible applications. Actually, the illustrated system structure can be adapted to many different tasks. A version of this system, called *Randy*, developed for QRNG application is described in chapter 5.

The main goal was to develop a robust FPGA-based system that could interface with a generic optical system. The system was designed for the ZedBoard development board. The presence of a 512 MB RAM memory as well as the possibility to use the AXI protocol for parameters setting and then a reliable TCP connection for a computer-software connection, makes this device an appropriate choice for QRNG and QKD applications. Moreover, the 100 MHz clock and the high number of I/O ports with different *electrical standards* (for instance LVCMOS33 and LVDS18) along with selectable *drive strengths* (the value of the driving electric current) complete the application scenario.

2.1 Basic system schematic

The whole system was designed on three main layers each one responsible for a dedicated role.

- Hardware layer: time critical operations, sequential and combinatorial logic, I/O management
- Board software layer: low memory and hardware registers management
- PC software layer: user interface, offline analysis

The three layers belonged to three different technologies and development frameworks. The three layers were linked together through AXI protocol and TCP connection. A summary schematic diagram is visible in figure 2.1.

2.1.1 Hardware layer

The hardware layer was developed on the Vivado Design Suite and included the whole FPGA design. The design was structured using both custom IP cores and

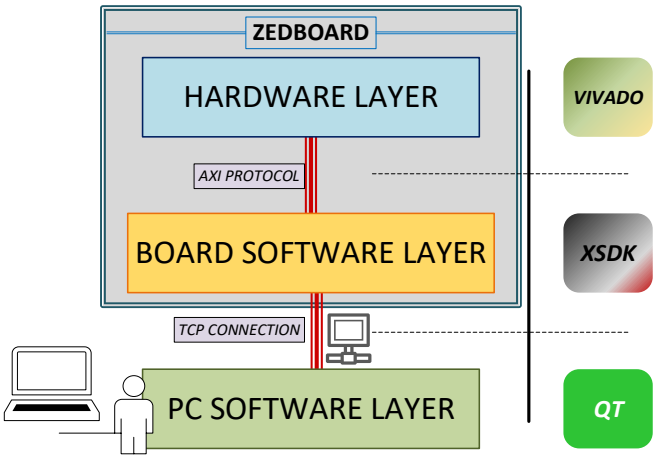

FIGURE 2.1: System layers scheme.

Xilinx IP cores. The custom ones were developed in VHDL language [48] and their function depended on the specific tasks required by an experiment. The used Xilinx IP cores mostly concerned parameters and data managing as well as the interconnections between PL and PS through the AXI protocol. According to the schematic shown in figure 2.2, the following IP core were used:

- AXI GPIO [49]: this core was aimed at receiving/sending parameter values from/to the Board Software as well as implementing interrupt procedures (see 2.2 for details). The core can have up to 2 input and 2 output ports with settable length (from 1 to 32 bits).
- AXI Central Direct Memory Access (CDMA) [50]: this core can access to the PL memory block like the BRAM as well as the DDR RAM. Therefore, it worked as a fundamental link between the two memories. For more details see section 2.2.
- AXI Interconnect [51]: this core works a funnel to allow the connection of multiple AXI core slave ports to just one master port. As a matter of fact, the Zynq PS core only has two master ports.
- AXI BRAM controller [52]: this core can be controlled by the AXI CDMA and must be used in order to access to a BRAM memory via AXI protocol.
- Zynq Processing System IP core [53]: this core represents the CPU part of the chip. Even though it cannot be programmed from Vivado, it must be inserted in any case in order to create the required links with AXI cores.
- Block Memory Generator [54]: also indicated as Block RAM or BRAM, this block can be controlled either via a custom VHDL block or an AXI BRAM Controller. Furthermore, it allows a simultaneous access to the memory thanks to

the *Dual-Port* functionality. This option is fundamental in order to set the data communication between PL and PS since one memory port can be accessed by a custom VHDL block while the other one can be accessed by the AXI BRAM Controller.

- Clocking Wizard [55]: this core initializes the Phase Locked Loop (PLL) and/or the Digital Clock Manager (DCM) of the chip. Basically, it allows to generate and distribute several clock signals through the chip. The output clock signals can be set to have specific frequency and phase. Its input is connected to the primary 100 MHz oscillator. The output signal is used to clock every blocks of the custom logic. For this design, the output frequency was set equal to the input one. Hence, the purpose of this core was to correctly distribute the clock signal avoiding timing violations.

Furthermore, the design communicated with the outside world through I/O ports. Most of these ports were connected to the custom logic. The used I/O ports were the ones on the PMOD modules which allowed an electrical link with the experiment apparatus like the electrical output signal of a single-photon detector. Moreover, user switches, pushbuttons and LEDs worked as I/O ports. For example a pushbutton could be used to give a start signal while an LED could warn the user about the state of the system. I/O ports connection can be defined by a dedicated *Constraints* file which describes to which physical pin a VHDL must be connected. It also allows to specify the port termination, IO standard, slew and drive options [56] which influences the electrical shape and coupling of the signals involved. For this system there was no need for any articular constraints settings. An example of specific constraints setting required by an application can be found in chapter 10. A schematic view of this design is visible in figure 2.2.

Compiling strategies and results

Vivado software offers the opportunity to choose among many different *Synthesis* and *Implementation* strategies [45, 57] which offer advantages and disadvantages depending on the required performances. Usually, FPGA compiling strategies concern area, power and timing performances optimization. Nevertheless, this design required less than 50% of the chip area and negligible power consumption. Hence, there was no need to prefer a compiling strategy over others and default strategy was used. Furthermore, the clock frequency was set to be equal to 100 MHz which did not required a dedicated timing approach since no timing-violations arose.

2.1.2 Board software layer

An application software was developed on XSDK. The software function was to allow a full control of the system and was a necessary link between the Hardware software layer and the PC one. It was implemented as a standalone application (no OS)

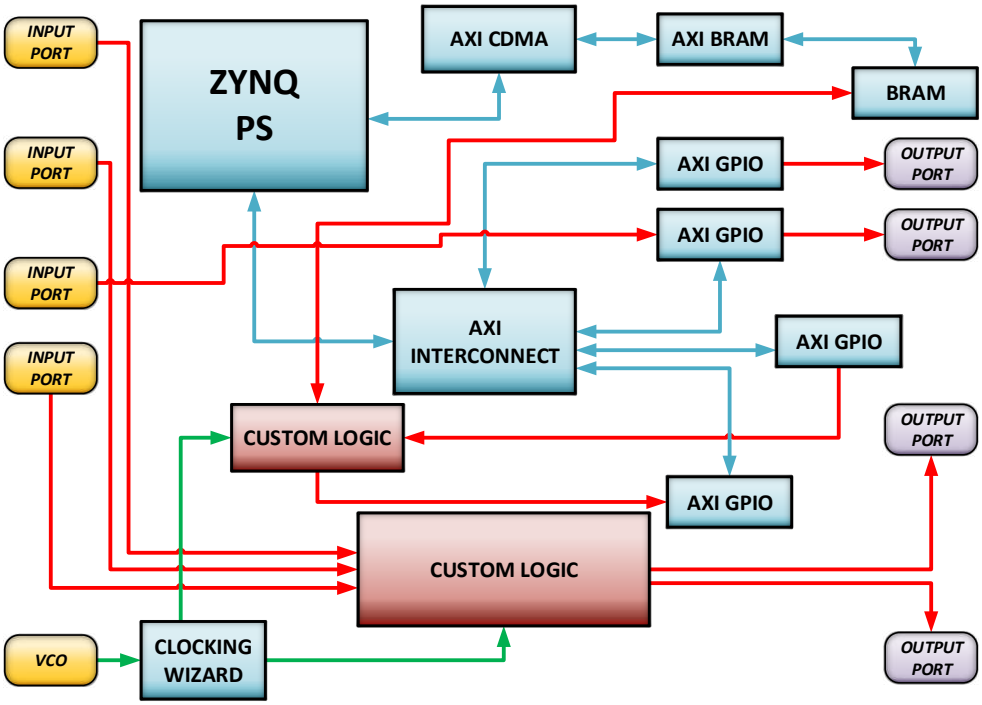

FIGURE 2.2: ZedBoard hardware scheme.

and was responsible for parameters and memory management, interrupts initialization, AXI drivers initialization and TCP connections setup with lwIP. The required libraries were given in a proprietary package called *Board Support Package* (BSP). Details on memory management, interrupts and TCP connection are given in section 2.2 and section 2.3. Parameters setting were handled using several AXI GPIOs which can set the value of their output ports by calling `XGpio_DiscreteWrite(instance name, channel number, value)`.

2.1.3 PC software layer

The PC software layer is an application developed on Qt creator framework [58] and was meant to be run on a Windows computer in order to manage the whole system. Qt creator is a robust multi-platform IDE used for many different applications [59–61] thanks to its reliability and flexibility along with the possibility to create nice and efficient GUIs by using QWidget elements [62]. The software was developed for Windows OS and was designed to communicate with the Board software through a TCP connection (details in section 2.3). Moreover, a GUI was designed to set parameters, retrieve data from the board and save them to file. This software did not require any real time performances or particular structures. A more detailed application example is described in subsection 5.5.2. An example of high performance Qt software with more details about its working principle will be given in chapter 3 in which the Quntroller software is described as well.

2.2 Memory management on board

A fundamental prerequisite of the system is the chance to store large volume of data on a non-volatile memory. Hence, it was essential to design a data transfer from the board to the computer and to its system hard disk. The first step was to configure a data transfer from the PL side to DDR RAM memory. The data transfer was design as follow. A dual-port BRAM was instantiated and was connected to an AXI BRAM Controller (port A) and to the Data Manager custom logic block (port B). The AXI BRAM Controller was in turn controlled by the AXI CDMA which was driven by proper `C` code. Furthermore, the AXI CDMA had access to the DDR-RAM. Therefore, the AXI CDMA allowed a data transfer from BRAM to RAM. A schematic view of the memory management setup is visible in figure 2.3.

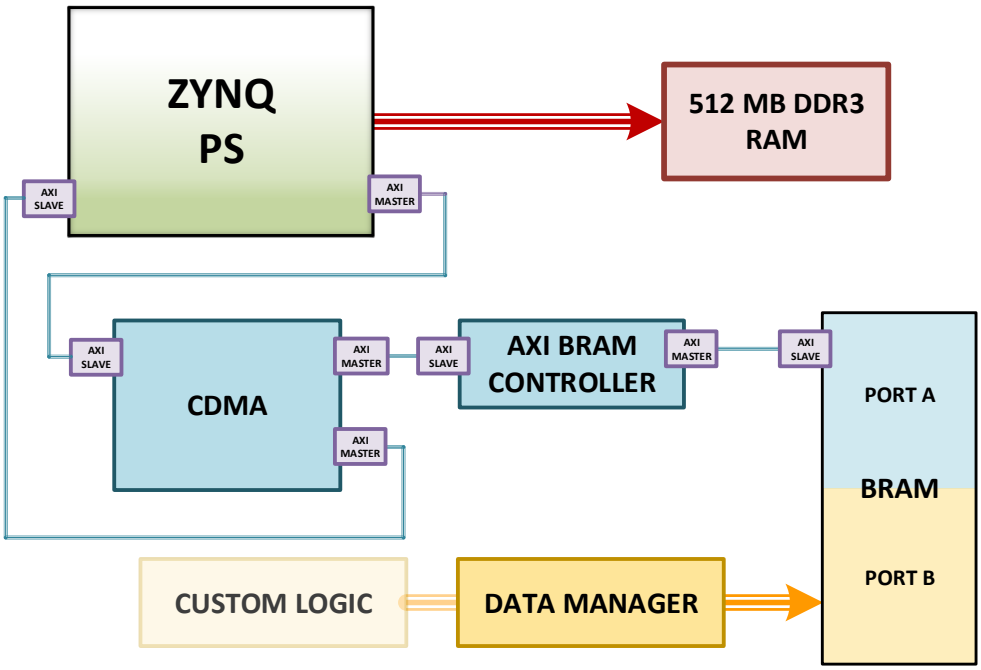

FIGURE 2.3: ZedBoard memory scheme. The BRAM IP core can be accessed both by the AXI BRAM Controller and by the custom VHDL block Data Manager. The whole data path is characterized by an AXI *master-slave* linking.

The transferring procedure is the following: the Data Manager custom block, linked to port B of the BRAM, was design to write a bit string data in a specific address memory. Thanks to an hardware counter, it can compare whether the counter reaches the half or the end of the memory. In both cases, an output port of Data Manager is asserted and its value is read by an AXI GPIO which, due to the value change, emits an *interrupt* signal. When the end of memory is reached, the block returns to the first address overwriting any existing data. The interrupt is read by the PS and handles in Board software layer. The software checks the value on the input port of the AXI GPIO and, if equal to 1 (corresponding to a 0-1 transition), reads

the first half of the BRAM and transfers data to a specific address in RAM memory. When another interrupt occurs, the second half of the BRAM is read and transfered to RAM memory. Data to be transfered are stored as an array of *u32*, an unsigned 32-bit integer type definition by Xilinx. A numerical variable is used and properly updated to trace the RAM address while a boolean variable is used to check which half of the BRAM memory has to be read. Finally, it is possible to transfer a desired portion of the data stored in the RAM to the computer through a TCP connection. For more details see section 2.3 This design can also be easily reversed and used to transfer data from the computer to the FPGA-level.

2.3 TCP connection

A TCP connection [63, 64] was set for the communication between the PS and the computer. The connection was developed from the *lightweight IP* (lwIP), an open source TCP/IP stack designed for embedded systems. It was firstly developed by Adam Dunkels at the Swedish Institute of Computer Science [65]. lwIP is widely used by embedded systems manufacturers like Analog Devices [66] and Xilinx [67]. The TCP connection was chosen over an UDP one [68] due to the need to guarantee the correctness of data over the lower latency. Two different TCP connections are established, one for data transfer and another one for instructions communication. This separation guarantees a higher clearness in the code structure and reduces the risk of slowing down data transfer.

2.3.1 Instructions communication

Parameters communication was defined by a simple *identifier* + *instruction* + *data* structure:

$$\underbrace{\texttt{YYYY}}_{Identifier} + \underbrace{\texttt{ZZZZ}}_{Instruction} + \underbrace{\texttt{DATA}}_{Instruction\,data}$$

The software is able either to read instructions or to send them. As an example, the software starts or stops to operate by receiving the following commands `YYYY + START` and `YYYY + START` (no instruction data required in this case). Otherwise, it can receive a full list of parameters to communicate to the FPGA with the command `YYYY + PARAM + PARAM1, PARAM2, PARAM3,...`.

2.3.2 Data transfer

The data transfer was set by the following procedure:

1. `sendData()` function:

- it initializes two pointers to the data array. The first pointer, *readDataPtr*, addresses the first element of the array while the second one, *lastDataPtr*, addresses the last one.
- it enables L1 first level cache memory [69] by calling `Xil_L1DCacheEnable()`. This cache was previously disabled to allow the memory transfer described in 2.2. But at this point it needs to be enabled to use full speed capability of the TCP connection. At the end of the TCP data transmission the cache is disabled again to allow new memory transfers.
- it calls `sendDataChunck()` function

2. `DataSentCallback(....)` is a callback function called after a successful transmission. The connection between the success signal and the function calling is set using the lwIP function `tcp_sent`. The `DataSentCallback(....)` function checks if there are data to transfer by simply comparing the distance between the two pointers *readDataPtr* and *lastDataPtr*. If so, it calls `sendDataChunck()`. If not, it stops the transferring and disables L1 first level cache memory by calling `Xil_L1DCacheEnable()`.

3. *sendDataChunck()* function determines the length of data to be sent by checking whether the TCP buffer size is smaller or larger than the *lastDataPtr-readDataPtr* size. Then, it calls the function *tcp_write* to send the data portion starting from the address *readDataPrt* and updates the pointer adding the data length value. The function *tcp_write* is part of lwIP.

This procedure allows to reach a speed transfer of around 700-800 Mbit/s which practically is quite close to maximum allowed for a Gigabit port. The gap is due to the latency introduced by the functions introduced. The operations within these functions were defined in the most basic way and it could be difficult to reduce their timing-impact even more. Anyway, this speed rate does not represent a system-bottleneck since it totally fits the performances requires for the QRNG and QKD applications.

Chapter 3

TDC system design: Quntroller software for quTools devices

3.1 quTAU device by quTools

The quTAU (figure 3.1) is a time-to-digital converter device developed by quTools company. It can discern events with a time resolution of *sim*81 ps over 8 channels. Every event is tagged with a 64-bit-resolution number. The tag describes the time at which the event was detected with respect to the time zero reference which corresponds to the device turn-on instant. It also provides an event counter which can count how many events were detected in a specific amount of time called *Exposure Time*. These two features belong to separate and dedicated hardwares and support different rate performances. The maximum event rates for tagging is 3 Mevents/s over 8 channels while the maximum rate for counting is 25 Mevents/s over 8 channels or 10 Mevents/s on a single channel (more details about the performances limit of the device is given in Appendix A). Through a dedicated USB connection the quTAU can share the tags list as well as the counts. Dedicated `C` libraries are provided by quTools company in order to develop custom softwares.

FIGURE 3.1: quTAU device by quTools. Picture by quTools.

3.2 Introduction to Quntroller software

A dedicated acquisition software, named *Quntroller*, was developed in order to manage quTAU device. The design was realized in `C++` language in Qt [58] environment using quTools `C` libraries [70]. The required specifications were the following:

- Basic device managing like *Channel selection, Termination type, Buffer size, Buffer clearing after data retrieving*.
- Suitable GUI
- Tags acquisition with a configurable rate
- Counts acquisition with a configurable rate
- Saving to file with configurable name, path and rate
- Stability and reliability during acquisition
- Possibility to support multiple quTools devices
- Possibility to lock the acquisition to a PPS signal through the use of a GPS receiver

3.3 Software structure

Quntroller was designed to guarantee reliability over data acquisition with quTAU device. An organized functions structure was defined in order to minimize latency and avoid errors. Furthermore, a *multi-thread* functionality was developed to manage the required multiple callings to save to file the acquired data. The used quTools functions were defined in the two header files `tdcbase.h` and `tdcmultidev.h`.

The software details are given with qualitative description of the functions following the schematic view of the structure shown in figure 3.2. Functions written in *italic* style represent pseudo names while functions written in `teletypefont` style represent actual names. The structure was divided in three parts: *Device Manager*, *QuTools Device* and *GUI*.

3.3.1 Device Manager

Device Manager includes the following functions:

- *Basic initializations*: this function is responsible for all the preliminary initialization required by the software, i.e. main variables, arrays, pointers, graphic elements. Plus, it does a preliminary automatic search for any connected quTools devices.
- *Get error messages*: this function simply translates in a printed message a code error returned by the call of a quTools function.

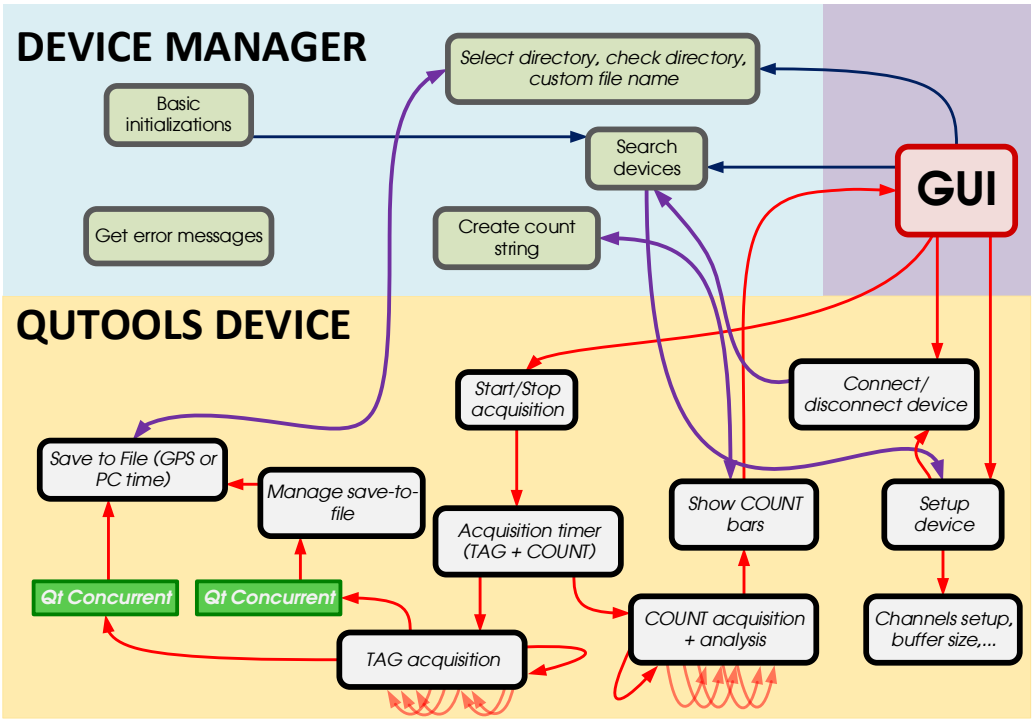

FIGURE 3.2: Quntroller software schematic.

- *Create count string*: this function converts the COUNT value in a readable format suitable for the GUI. For instance, a COUNT value of 987654 is difficult to read on the monitor; this function converts the number into the *987 k* format truncating the hundreds since they are negligible in this case.
- *Select directory, check directory, custom file name*: these functions set up the file name and the directory which the file must be saved in.
- *Search devices*: this function searches for any connected quTools devices by calling the `TDC_discover` function. It is called at the software startup but it can be also called by pushing the dedicated QPushButton [71] in the *Main* tab of the GUI. If a device is found, it calls the *Setup device* function.

3.3.2 QuTools Device

QuTools Device includes the following functions:

- *Start/Stop acquisition*: this function is called every time the *Start/Stop Acquisition* QPushButton is pressed. The function calls the Acquisition timer function to start/stop the timers-loop. Thanks to a simple boolean variable, the QPushButton text changes according to what the software is doing: *Stop Acquisition* is shown during acquisition while *Start Acquisition* is shown when no acquisitions are on.

- *Connect/Disconnect device*: this function is called by pressing the QPushButton "*Connect/Disconnect to name-Of-Device*". In case of request for connection it calls in turn the *Search devices* function to set up a new connection otherwise it calls the quTools `TDC_disconnect()` function.

- *Setup device*: this function is responsible for the setup of a quTools device. Firstly, it connects to the device by calling the `TDC_connect()` function. Then, it sets the device *time stamps buffer size* and *exposure time* using the two quTools functions `TDC_setTimestampBufferSize()` and `TDC_setExposureTime()` according to the values set in GUI through dedicated QSpinBox elements [72]. It also sets up the graphic frame with the channels QRadioButtons [73].

- *Channel setup, buffer size,...*: these functions select and set up the device channels along with their electrical termination by calling the quTools functions `TDC_enableChannels()` and `TDC_switchTermination()`. They can also directly set the *time stamps buffer size* and *exposure time* without calling *Setup device*.

- *Acquisition timer (TAG + COUNT)*: this function is one of the most important. It is responsible for the acquisition timing by starting or stopping two QTimer variables [74]. Whenever a QTimer element is started, it emits a *timeout* signal after an amount of time, expressed in milliseconds, set as a parameter at the function call. Through the *Signal-Slot* connection [75], the *timeout* signal can be connected to a specific function. The COUNT QTimer *timeout* signal is connected to the *COUNT acquisition* function while the TAG QTimer *timeout* signal is connected to the *TAG acquisition* function. COUNT QTimer timeout value corresponds to the *Exposure Time* while TAG QTimer time out value corresponds to the *Acquisition Step*. When an acquisition must be stopped, the QTimers are stopped avoiding extra calls to the *COUNT acquisition* and *TAG acquisition* functions.

- *COUNT acquisition + analysis*: this function is called in infinite loop that can be started and stopped by the *Acquisition timer* function. Whenever it is called, it asks the quTools device for the latest count values evaluated during an *Exposure Time* period. The used quTools function to retrieve the counting values is `TDC_getCoincCounters()`. Then, it makes a brief analysis on the retrieved values to determine the maximum value along with the total count and also set the scale of the COUNT bars.

- *Show COUNT bars*: the function is responsible for the management of different QProgressBar [76] elements which allow the user to quickly evaluate the amount of events on every channels. They are clearly visible in figure 3.3.

- *TAG acquisition*: this function is called in an infinite loop that can be started and stopped by the *Acquisition timer* function. At every call, it asks the quTools device for the latest tags list, i.e. the contents of the buffer. The tags retrieving

is possible by calling the quTools function `TDC_getLastTimestamps`. Then, it calls the *Save to File* function or the *Manage save-to-file* one, depending on the user options. In any case, this call is made on a separate thread by using QtConcurrent [77] and the `QtConcurrent::run` function. This choice was made to create a multi-thread architecture which offered huge benefits [78–80] and avoided entanglements between functions: *TAG acquisition* is totally independent from the *Save to File* and the *Manage save-to-file* functions. As a matter of fact, *TAG acquisition* must be called every *Acquisition Step* period with no room for any delays.

- *Save to File (GPS or PC time)*: this function is called by *TAG acquisition* and is responsible for writing to file the tags data. The function has two parameters: a QDateTime [81] and a QTime [82] which stand for the date and time and label the file. The two values are determined during the function calling by *TAG acquisition*. In case of a normal acquisition, the two values correspond to the system date and time and are determined by calling `QDateTime::currentDateTime` and `QTime::currentTime`. In case of a GPS synchronized acquisition, the two values are the GPS date and time variable managed by the GPS part (this feature is described in section 9.5). The writing to file process occurs according to the following steps. First of all, the existence of the assigned directory is checked. If it does not exist, a QDir variable [83] is initialized to allow the creation of a new directory. A QFile variable [84] is initialized as well. Then, the process starts building the file by putting tag and channel values into a QByteArray variable [85] using proper type conversion. Furthermore, the QByteArray is prepared with an header that recaps all qualitative information about that acquisition. Once the QByteArray is ready it is written on file through QDataStream [86] in a *.txt* format. An example file is given in 3.3.4.

- *Manage save-to-file*: this function accumulates multiple tag acquisitions and writes them in a single file. This option is necessary in order to avoid the creation of too many files during an acquisition. At each call the function copies tag values in a specific location of an array. When a *Different-Save-to-File Rate* tag acquisition occurs, the *Save to File* function is called in order to write the array that comprehends multiple acquisitions.

3.3.3 GUI

The *GUI*, visible in figure 3.3, was designed to allow a full access to the software. It was structured in different tabs: one for each quTools device and a main tab for an overall view. In a quTools device tab it is possible to:

- connect and disconnect the device
- select channels and their electrical termination

- start and stop the acquisition
- set the Acquisition Step
- set the Buffer Size
- clear data after retrieving. This option allows to clear or not the buffer after tags retrieving
- set *Different Save-to-File Rate*
- lock acquisition to PPS signal
- set *Exposure Time*
- set the scale of QProgressBar

Furthermore, a dedicated QProgressBar stands nearby the *Start/Stop Acquisition* push button. During the acquisition, the value of this bar is continuously updated with a specific set of values that create a motion illusion. When events are detected, the bar color is set to green while it is set to red in case of no events. This graphic effect allows the user to immediately determine whether there are detected events or not. Moreover, this bar is duplicated on the top of the window in order to constantly monitor the acquisition of all the connected devices.

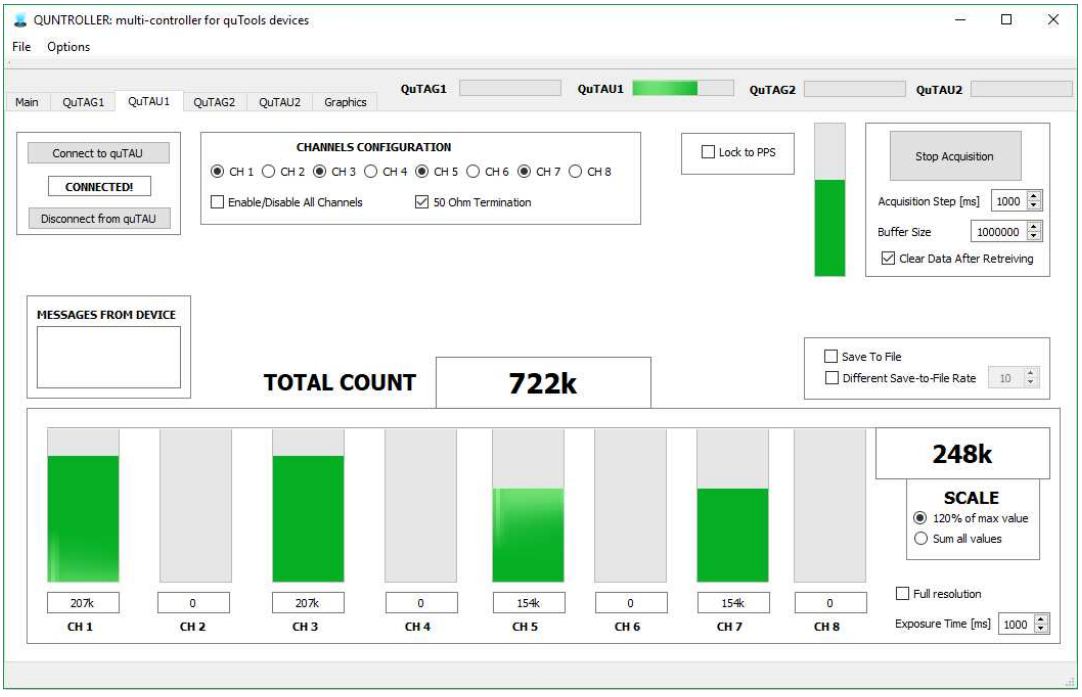

FIGURE 3.3: Quntroller main window.

3.3.4 File example

A prerequisite of the tag file was to be compatible with the existing format used by the research group in the post-processing analysis framework. The file has a description header which indicates the date and time. It also indicates the time resolution of the quTools device. The lists of the tag and channel values separated by a semicolon are written below the header. An example file is visible in figure 3.3.4.

```
# Time Tags
# 2018-07-18 12:38:22 UTC
# Unit Time Tag: 81.000 ps
# Unit Channel: 1
# Time Tag ; Channel
    1278624999709 ;                    1
    1278625000752 ;                    2
    1278625003826 ;                    1
    1278625004870 ;                    2
    1278625007944 ;                    1
    1278625008987 ;                    2
    1278625012061 ;                    1
    1278625013102 ;                    2
    1278625016177 ;                    1
    1278625017219 ;                    2
    1278625020291 ;                    1
    1278625021337 ;                    2
    1278625024412 ;                    1
```

FIGURE 3.4: File example of a Quntroller acquisition

3.3.5 Overall performances

A thorny issue to deal with was the high amount of data that needed to be written to file in very short time. The performances limit had to be bound only by the transfer capability of the storage device (hard disk). Two stress tests were conducted in order to evaluate performances.

The first test was run by feeding 2 Mevents/s into the quTAU device. The *Acquisition Step* was set to 250 ms. The *Buffer Size* was set to its maximum, i.e. 1 million events. *Different Save-to-file Rate* equivalent to 4. Hence, the software saved one file every second, which represented four different acquisitions. The file size was equivalent to 80 MB. The test lasted 15 minutes and all files were written. Therefore, the actual saving rate was 80 MB/s.

The second test was run with 3 Mevents/s, *Acquisition Step* equivalent to 200 ms, *Buffer Size* to its maximum and *Different Save-to-file Rate* equivalent to 5. Hence, the software saved one file every second, which represented five different acquisitions. The file size was equivalent to 120 MB. In a 15-minute acquisition, the software lost one file. The cause of this missing file was the too high rate which produced the skip of one of the call to the *Save to File* function. In fact, the function was run on a new thread only if a free thread was available. If not, the function could not be not called.

These tests showed that the Quntroller is totally reliable in a 2 Mevents/s scenario while it can lead to some data loss in a 3 Mevents/s scenario. This result is

acceptable for the purposes that the software needs to satisfies within quantum experiments environment. Nevertheless, possible future developments could consider a further improvement of these performances.

Part II

Quantum Random Number Generation systems

Chapter 4

Quantum Random Number Generation

This chapter briefly introduces Quantum Random Number Generation (QRNG) and gives an overview of some available generation protocols and unbiasing algorithms suitable for the extraction of randomness by processing the photon time of arrival.

4.1 Introduction

Over the past few years, there has been a widespread interest in random number generators based on physical processes of quantum nature. In fact these devices, the so-called quantum random number generators (QRNGs), represent the ultimate way to obtain reliable randomness, free from the typical non-random issue that affects the Pseudo Random Number Generators (PRNGs) and non-quantum True Random Number Generators (TRNGs). While the first category comprehends algorithm-based generators and hence totally deterministic, the second one comprehends generators that produce random numbers by sampling a natural random process. Such processes could be for example free-running oscillators, semiconductor thermal noise or Zener diode shot noise. However, these generators are quite sensitive to environment conditions and require a robust statistical model of the physical system in order to consider any possible source of non-randomness [87–89]. On the other hand, also quantum-based random number generator systems can suffer of non-idealities but their perturbations on the system are independent from the bit generation and can be measured with precision [87]. Therefore, QRNG can play a major role in different fields like lotteries, Montecarlo analysis [90], cryptography [91–93] and quantum mechanics experiments [94–98]. Certainly, in the case of cryptographic applications, a faulty random number generator, used for the secret key generation, could leak relevant information to an eavesdropper and undermine the effectiveness of a cryptographic protocol, no matter its security.

Typically, QRNGs exploit the quantum properties of the optical radiation field in many "recipes". The different architectures exploit different properties; setups using entangled systems and violating Bell inequalities feature the highest unpredictability in the so-called device-independent (DI) framework [99–103]. Systems that trust

the measurement apparatus but not the states of the quantum system or vice versa, the so-called semi-device-independent generators, work under less strict assumptions [104]: for this reason, they feature, in principle, less unpredictability than DI-QRNG but are more feasible to implement and reach even larger generation rates. The last category includes those generators that work in a framework of complete trust both of the quantum system and of measurement apparatus. This means that the absence of side information, exploitable by an adversary, is assumed. The most famous example of such generator is the *welcher weg* QRNG that produces randomness according to which path a photon takes after interacting with a beam-splitter [105, 106]. This work focuses on the sub-category of trusted QRNG that were developed in order to limit the number of single-photon detectors. Random number generation with just one detector is indeed possible by exploiting time as an additional degree of freedom and by leveraging on the statistical features of the photon detection distribution. This could be done with the following procedure: time sampling of the photon detector with a sampling rate almost equal to the photon rate; application of dedicated generation protocols; application of dedicated unbiasing algorithms. Further in this chapter some of these protocols and algorithms are described. Then, chapters 5 and 7 show that it is possible to generate true random numbers without the application of dedicated generation protocols. Using a higher sampling rate, unbiasing algorithms are directly applied to the samples stream achieving higher generation rate of true random numbers. This technique was used on two different systems: the first one, Randy, discussed in chapter 5, uses a standard clock to sample the signal of one single-photon detector; the second one, LinoSPAD, discussed in chapter 7, uses both a standard clock and a time-to-digital converter (TDC) to sample the signals from a matrix of 256 single-photon detectors.

4.2 Generation protocols and unbiasing algorithms

This section introduces and presents two generation protocols that can extract randomness from a photons distribution. These protocols work on the symmetry of photons time distribution.

4.2.1 Stipčević generation protocol

A first example is the generation protocol introduced by Stipčević *et al.* in [107], denominated here "Diff-QRNG". The Diff-QRNG comprises: a light source attenuated to single-photon level illuminating a single-photon detector; a clock that counts the time between the detection events. A binary random variable $X_j = \{0, 1\}$ is obtained by comparing the length of time intervals between three consecutive detections. It is therefore convenient to define the discrete random variable $\mathcal{T}^j$, associated to the detection instants, such that $\mathcal{T}^j = \left\{t_0^j, t_1^j, t_2^j\right\}$. Hence, X_j takes its value x_j according to the following "rule": given $\Delta T_1 = t_1 - t_0$ and $\Delta T_2 = t_2 - t_1$,

- if $\Delta T_1 > \Delta T_2$ then $x_j = 0$
- if $\Delta T_1 < \Delta T_2$ then $x_j = 1$
- if $\Delta T_1 = \Delta T_2$ then $x_j = \emptyset$, i.e. no bit is generated

The rule is iterated so that for the next bit b_{j+1} the new time interval starts with the end of the previous one, i.e., $t_2^j = t_0^{j+1}$. The physical principle that guarantees the identical and independent distribution of the bits, described by the equations

$$\Pr\left[x_j = 0\right] = \Pr\left[x_j = 1\right] \tag{4.1}$$

$$\Pr\left[x_j | x_{j-1}\right] = \Pr\left[x_j | x_{j-1}, x_{j-2}, \ldots, x_1\right] = \frac{1}{2} \tag{4.2}$$

follows from the memoryless property that characterizes the exponential distribution of the interarrival times ΔT, namely

$$\Pr\left[\Delta T_1 > \Delta T_2\right] = \Pr\left[\Delta T_2 > \Delta T_1\right] = \frac{1}{2} \tag{4.3}$$

As a consequence, a random string $X = \{X_1, X_2, \ldots, X_n\}$ with $n \to \infty$, is characterized by full Shannon binary entropy [108], i.e. $\mathcal{H}(X) = 1$ bit. This implies that the average number of i.i.d. bits that are generated per unit time, is equal to $R_{\text{i.i.d.}} = R_{\text{phot}}/2$, where R_{phot} is the number of photo-detections per unit time [1].

4.2.2 Fürst generation protocol

A second example is the generation protocol introduced by Fürst *et al.* in [109], denominated here "OddEven-QRNG". As with the previous example, in the OddEven-QRNG a light source attenuated to single-photon level illuminates a photomultiplier but in this case a counter enumerates the number of photons detected within a fixed time interval, τ, corresponding to the period of a sampling signal. Defining n_τ^j as the number of detections within the interval τ_j, a random binary variable X_j assumes its value x_j, with the following rule:

- if $n_\tau^j \mod 2 = 0$ then $x_j = 0$
- if $n_\tau^j \mod 2 = 1$ then $x_j = 1$

In other words, the bit value is determined according whether an even or odd number of detections is registered in the time interval τ. For a Poisson distribution [110] the probability of having an even or odd number of detections can be written as:

$$\Pr[2n] = \sum_{n=0}^{\infty} \frac{(\lambda\tau)^{2n}}{(2n)!} e^{-\lambda\tau} = \frac{1 + e^{-2\lambda\tau}}{2} = \Pr[x_j = 0] \tag{4.4}$$

[1] With the exception of bit x_1 and x_n two photodetection are necessary to generate an i.i.d. bit.

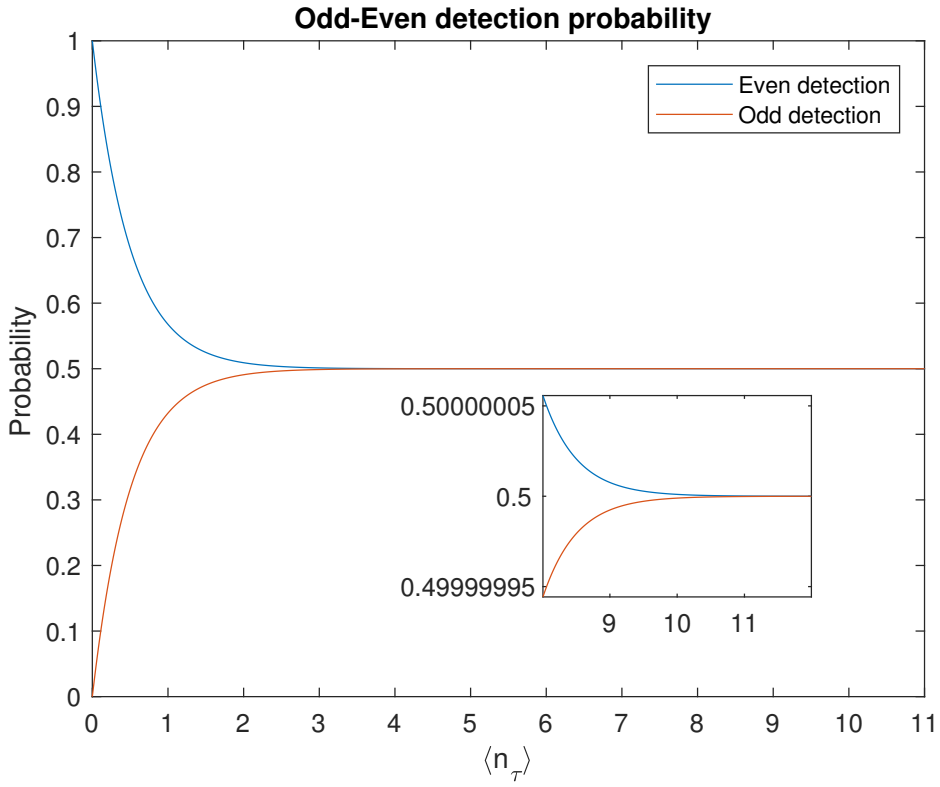

FIGURE 4.1: OddEven detection probability.

$$\Pr[2n+1] = \sum_{n=0}^{\infty} \frac{(\lambda\tau)^{2n+1}}{(2n+1)!} e^{-\lambda\tau} = \frac{1-e^{-2\lambda\tau}}{2} = \Pr[x_j = 1] \tag{4.5}$$

where λ is the mean number of photons per second and $\lambda\tau = \langle n_\tau \rangle$ is the mean number of photons per time interval τ. The two probabilities are plotted as function of $\langle n_\tau \rangle$ in figure 4.1. To avoid a bias in the output, $\langle n_\tau \rangle$ has to be sufficiently large so that $\Pr[x_j = 0] = \Pr[x_j = 1]$. Therefore, since $\mathcal{H}(X)$ is a function of $\langle n_\tau \rangle$, the generation rate is given by $R = \mathcal{H}(X)/\tau$.

4.2.3 John von Neumann algorithm

In 1951, John von Neumann presented an unbiasing algorithm which allows to transform a biased bit string s_w into an unbiased one [111]. Given a biased but not correlated binary string

$$s_w = w_0||w_1||w_2||...||w_i||w_{i+1}||...||w_n \tag{4.6}$$

where w_i represents a bit value and the symbol $||$ indicates the concatenation of two bits, it is possible to remove any bias by manipulating pair of bits $w_i||w_{i+1}$. The output binary random variable X_j assumes its value x_j with the following rule:

- if $w_i||w_{i+1} = 01$ then $x_j = 1$
- if $w_i||w_{i+1} = 10$ then $x_j = 0$
- otherwise, $x_j = NULL$

Given that $\Pr[w_i = 1] = p_1$ and $\Pr[w_i = 0] = p_0$ with $p_1 \neq p_0$, it is clear that $\Pr[x_j = 1] = \Pr[x_j = 0] = p_0 p_1$. Therefore, despite the initial bias the output string has the same probability to have a 1 or a 0. On the other hand, the disadvantage of this procedure is that 00 and 11 bit pairs are drop off. Hence, the extraction efficiency is equal to $\eta = 2p_0p_1/2$ bringing to a best case efficiency of 25% when $p_0 = p_1 = 1/2$.

4.2.4 Peres algorithm

Peres elaborated a reiterative version of the von Neumann algorithm [112] in order to increase its efficiency, asymptotically achieving optimal information efficiency $\eta \to 1$. Furthermore, it is computational reasonable. Other unbiasing algorithms, like the Elias one [113], has optimal information efficiency but very high computational complexity and memory requirements [114].

Taking a sequence of bit s_w, the Peres procedure is defined as

$$\Psi(s_w) = \Psi_N(s_w)||\Psi(\Psi_U(s_w))||\Psi(\Psi_V(s_w)) \tag{4.7}$$

where $\Psi_N(s)$ is the von Neumann algorithm applied to the string, $\Psi(\Psi_U(s))$ is the Peres algorithm applied to a new sub-string $\Psi_U(s)$ and $\Psi(\Psi_V(s))$ is the Peres algorithm applied to another new sub-string $\Psi_V(s)$. $\Psi_U(s)$ is created by an XOR operation on samples pair while $\Psi_V(s)$ is created by taking the latest samples of every 00 and 11 pairs. The un-biasing demonstration is straight forward. Given that $\Pr[w_i = 1] = p_1$ and $\Pr[w_i = 0] = p_0$ with $p_1 \neq p_0$, the probability that a generic bit pair, u_i, in a $\Psi_U(s)$ sequence is equal to 01 is the same as 10, i.e.

$$\Pr[u_i = 01] = \Pr[u_i = 10] = ({p_1}^2 + {p_0}^2)2p_1p_0 \tag{4.8}$$

On the other hand, the probability that a generic bit pair, v_i in a $\Psi_V(s)$ sequence is equal to 01 is the same as 10, i.e.

$$\Pr[v_i = 01] = \Pr[v_i = 10] = {p_1}^2{p_0}^2 \tag{4.9}$$

With a bit string of enough length, the output binary entropy $\mathcal{H}_o$ is ~ 1. Therefore the extracted length $\mathcal{L}_\mathcal{O}$ can be evaluated by the length of the input string $\mathcal{L}_\mathcal{I}$ and the initial binary entropy $\mathcal{H}_i$ with the following:

$$\mathcal{L}_\mathcal{O} = \mathcal{L}_\mathcal{I}\frac{\mathcal{H}_i}{\mathcal{H}_o} \simeq \mathcal{L}_\mathcal{I}\mathcal{H}_i \tag{4.10}$$

For an accurate description of the procedure refer to [112].

4.2.5 Zhou-Bruk algorithm

Zhou-Bruk algorithm described in [115] allows to apply Peres over a *Loaded Dice* string. With the term *Loaded Dice* it is indicated a biased stream of numbers defined in an arbitrary basis different from the binary one. A bias in one basis implies a correlation in a lower level description basis. For instance, given a set of numbers in the decimal basis, a bias in this set produces a correlation on the equivalent binary description of that set. Therefore, it is not possible to apply Peres algorithm to such a bit string. Through the Zhou-Bruk algorithm it is possible to remove the bias. A brief description of the procedure follows. Given a biased set of numbers which can be described in binary basis using b bits, said b=8, the set can be written in column order as follows:

N_1	0	1	0	1	0	0	1	0
N_2	1	0	0	0	0	1	0	0
N_3	0	0	0	0	1	0	1	0
N_4	0	0	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...
N_i	1	0	1	0	1	0	0	1

TABLE 4.1: Zhou-Bruk table

It is possible to create up to 2^b subsets according to the following procedure.

- The first subset X_1 corresponds to the first column of bits.
- The second subset X_2 corresponds to all the bits of the second column that have a bit 0 on their left.
- The third subset X_3 comprehends all the bits of the second column that have a bit 1 on their left.
- The fourth subset X_4 comprehends all the bits of the third column that have a 00 sequence on their left.
- The fifth subset X_5 comprehends all the bits of the third column that have a 01 sequence on their left.
- The sixth subset X_6 comprehends all the bits of the third column that have a 10 sequence on their left.
- The seventh subset X_7 comprehends all the bits of the third column that have a 11 sequence on their left.
- So on till the X_{2^b} is created.

Then, discarding the empty sets, it is possible to apply Peres algorithm independently to every subsets and concatenate the resulting substrings to obtain the final

random string S_R:

$$S_R = \Psi(X_1)||\Psi(X_2)||\Psi(X_3)||....||\Psi(X_i) \tag{4.11}$$

which will be without any correlation and bias. For a complete explanation about Zhou-Bruk procedure, refer to [115].

Chapter 5

Randy Quantum Random Number Generator

This chapter describes the study of a novel approach for QRNG based on the sampling of a single-photon detector and the application of unbiasing algorithms. It also gives a full description of the FPGA-based system on which such QRNG was implemented.

5.1 Novel approach

The aim of this research is to take into consideration the simplest way to generate random numbers by using the temporal degree of freedom without the need to devise complex "rules". In its essence, the process of random number generation with a single-photon detector, for instance a single-photon avalanche diode (SPAD), can be considered as a process with two signals: a squared-wave signal $\mathcal{S}_{\text{det}}(t)$ generated by the single-photon detector that is sampled by a periodic signal $\mathcal{S}_{\text{clk}}(t)$, which is generated by a clock with a period τ_{clk}. Without impinging photons, $\mathcal{S}_{\text{det}}(t)$ has a typical value L. Given a photodetection at the time instant t_i, the $\mathcal{S}_{\text{det}}(t)$ toggles its state from $\mathcal{S}_{\text{det}}(t)(t < t_i) = L$ to $\mathcal{S}_{\text{det}}(t)(t_i \leq t < \tau_U) = U$. τ_U is the specific fixed time interval a SPAD keeps the state U before returning to L and typically $\tau_U < \tau_{\text{dead}}$, being τ_{dead} the SPAD dead-time. This means that the random signal toggles back to the L state before being able to detect another photon.

The binary random variable X_j takes its value x_j at the instant $j\tau_{\text{clk}}$ according to the following rule:

$$x_j = \begin{cases} 1 & \text{if } \mathcal{S}_{\text{det}}(j\,\tau_{\text{clk}}) = U \neq \mathcal{S}_{\text{det}}((j-1)\,\tau_{\text{clk}}) \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

with the index $j \in \{0, 1, 2, \dots\}$. The above rule is equivalent to have $x_j = 1$ whenever the clock detects a rising edge of $\mathcal{S}_{\text{det}}(t)$ and $x_j = 0$ otherwise. It is worth to be noticed that all existing paradigms of QRNG based on photon time of arrival can be seen as a (non-optimal) post-processing algorithm of the sequence $X = (x_1, x_2, \cdots, x_n, \cdots)$. The maximum content of randomness that can be extracted by

the above physical process is fully included in the X sequence, and in particular it is given by the Shannon entropy (in the large n limits):

$$\mathcal{H}(X) = -p_0 \log_2(p_0) - p_1 \log_2(p_1) \tag{5.2}$$

where p_0 and p_1 are the probability of obtaining $X = 0$ and $X = 1$ respectively.

It is clear that the maximum generation rate is given by $r = \tau_{\text{sampl}}^{-1}$. Given a mean photon number per second $\lambda = R_{\text{phot}}$, for a Poisson distribution the probability of no detections within the τ_{sampl} interval is equal to $\Pr[\emptyset] = e^{-R_{\text{phot}}\tau_{\text{sampl}}}$. To achieve the rate $r_{\max}$ it is necessary to avoid a bias in the output string X, by tuning the photon detection rate in order to obtain at least one detection within a sampling period, namely $1 - \Pr[\emptyset] = 1/2$ from which we obtain $R_{\text{phot}} = \ln 2/\tau_{\text{sampl}}$.

It is therefore clear that the faster the clock rate is, the larger the detection rate should be in order to keep the bias in 0 low. However, R_{phot} cannot be set arbitrarily large but it is strongly limited by the physical limit of the single-photon detector, such as dead-time, after-pulse, dark counts. By keeping R_{phot} fixed, the idea was to increase the generation rate r by deliberately increasing the sampling rate and to produce a highly biased string with plenty of 0s and very few 1s and then re-balance the bias through the Peres algorithm.

The extraction rate after Peres algorithm, under asymptotic assumptions, is given by

$$r_{\text{i.i.d.}} = \tau_{\text{clk}}^{-1} \times \mathcal{H}(X)\,. \tag{5.3}$$

As shown in figure 5.1, by fixing R_{phot}=200 kcounts/s, the extraction rate increases with the sampling rate. Therefore, the optimal choice was to have $\tau_{\text{clk}}^{-1} >> R_{\text{phot}}$ since the increase in $\mathcal{H}(X)$ did not change the extraction rate significantly. The maximum entropy is reached with a sampling rate equal to $R_{\text{phot}}/\ln 2$ = 288.539 kHz. These plots confirm the following: with a fixed R_{phot} the generation rate is more influenced by the actual input length than by the bias value. Furthermore, choosing another value of R_{phot} does not change these considerations but only the evaluated values. The actual value of R_{phot} can only be limited by the physical performances of the detector. In this case, the detector begins to saturate around 800 kcounts/s undermining the validity of the model and the photon rate was chosen to stay well below this value.

To summarize, any QRNG based on the photon time of arrival can be modeled by a binary signal $\mathcal{S}_{\text{det}}(t)$ sampled by a clock $\mathcal{S}_{\text{clk}}(t)$ that gives an output sequence X. Different post-processing on X give rise to different protocols, such as the Diff-QRNG or the OddEven-QRNG. Further, this chapter proposes an optimal post-processing able to extract the maximum available entropy from the string X based on Peres algorithm. The next section explains how to apply the aforementioned methods to physical generators, taking care of the non-idealities of the detectors.

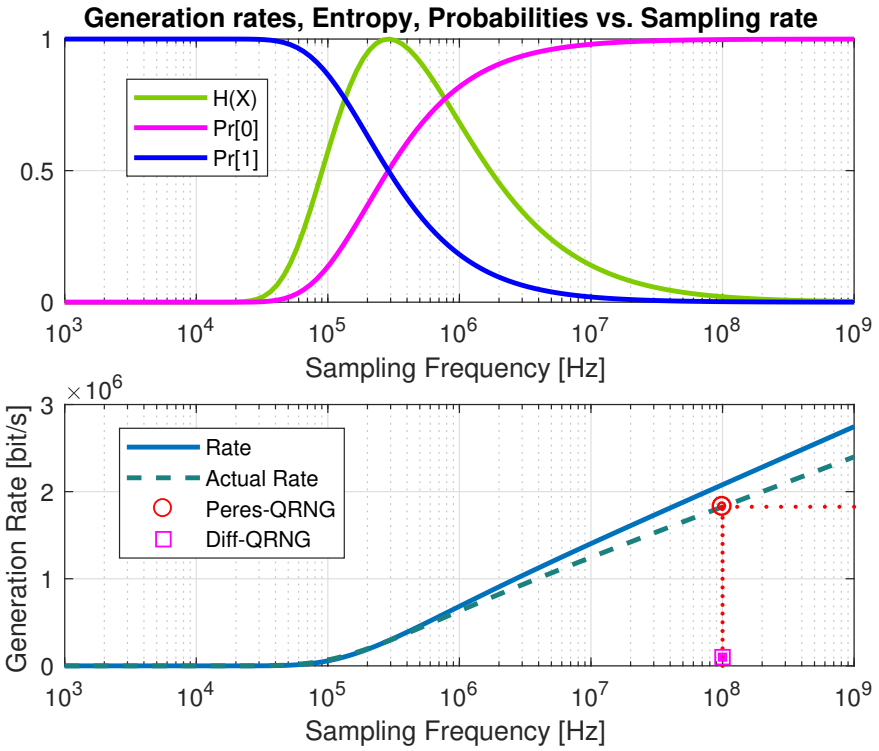

FIGURE 5.1: *Top*: the probabilities to sample 1 or 0 as function of the sampling rate. Probabilities are determined from the Poisson distribution with R_{phot}=200 kcounts/s. Shannon binary entropy is plotted as well. The cross-point between the two probabilities, which produces the maximum entropy value, is at 288.539 kHz. *Bottom*: the solid line represents the rate evaluated from the nominal values of single-photon events. The dashed line represents the rate evaluated from the actual values of single-photon events, i.e. with the afterpulses removing. The evaluation of the final rate, circle spot, takes into account the actual sampling frequency which is equal to 97 MHz instead of 100 MHz due to dead time removal. To make a comparison, the Diff-QRNG rate is also indicated (square spot).

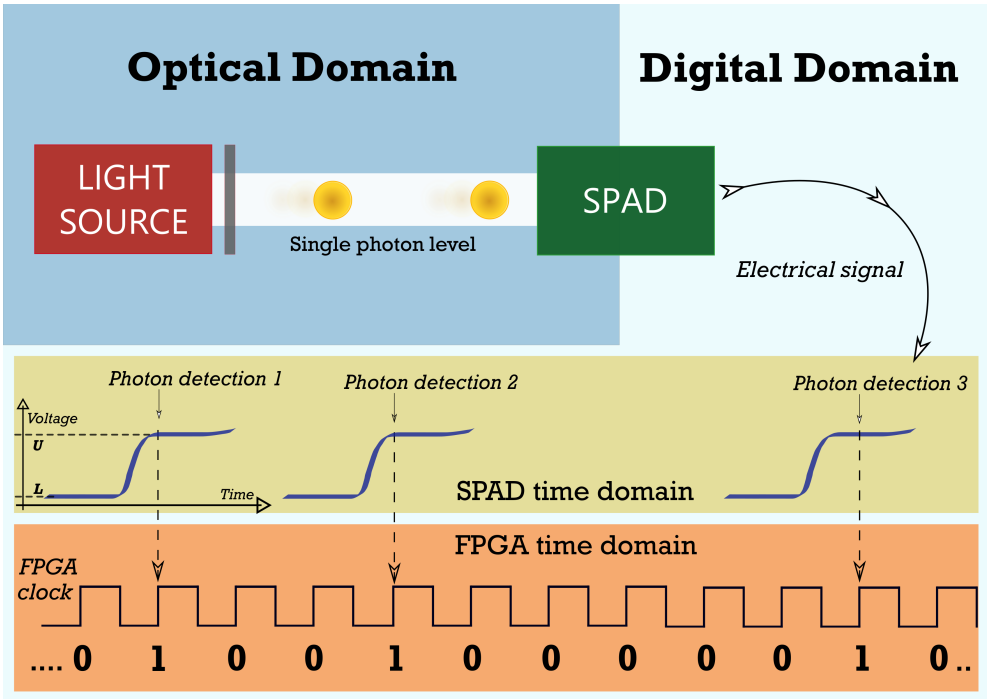

FIGURE 5.2: A schematic view of Randy system. The SPAD links the optical domain to the digital domain detecting photons from a light source -attenuated to single-photon level- and sending electrical signals to the FPGA. Through its internal clock, the FPGA samples the SPAD events shifting them from the asynchronous SPAD time domain to its own time domain. The detected time difference $\Delta 1$ and $\Delta 2$ will be compared in order to produce a random bit. Clock and SPAD rates are not on scale.

5.2 System overview

A system was developed in order to produce true random numbers using different generation protocols as well as Peres algorithm. The use of an FPGA allows a full control over the generation process and great flexibility to easily switch from one protocol to another. This design was called *Randy*. The default 100 MHz system clock was used to sample the SPAD signals and the SPAD photon count rate was set to 200 kcounts/s due to its non-linear behavior on higher rates. A schematic view of the setup is shown in figure 5.2. The advantages are quite clear since an FPGA allows a full description of the time evolution of the system: every operation can be described as multiple of the fundamental time unit τ_{sampl} defined by the system clock. Therefore, the behavior of the system is fully deterministic apart from the non-deterministic side due to the true randomness of the photon time of arrival. The generation protocols [107, 109] are implemented directly on the FPGA in order to make it a real-time device. This feature was successfully used to produce timed random numbers in the work of Vedovato *et al.* [98] as explained in chapter 6. On the other hand, these protocols have low rate performances: a photon count rate of 200 kcount/s produces a rate of true random bits of 100 kbit/s for the Diff-QRNG protocol and of 20 kbit/s for the OddEven-QRNG[1].

[1]The time interval was set in order to have $\langle n_\tau \rangle \simeq 10$ according to what stated in subsection 4.2.2.

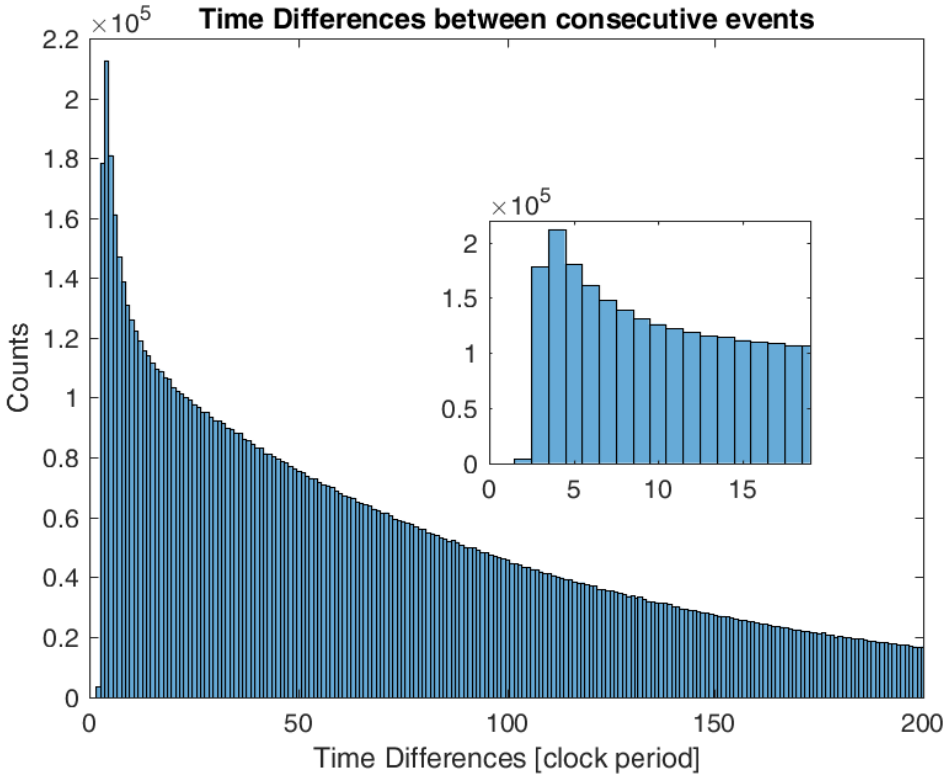

FIGURE 5.3: Time differences histogram without any post-processing. The inset highlights the presence of dead time.

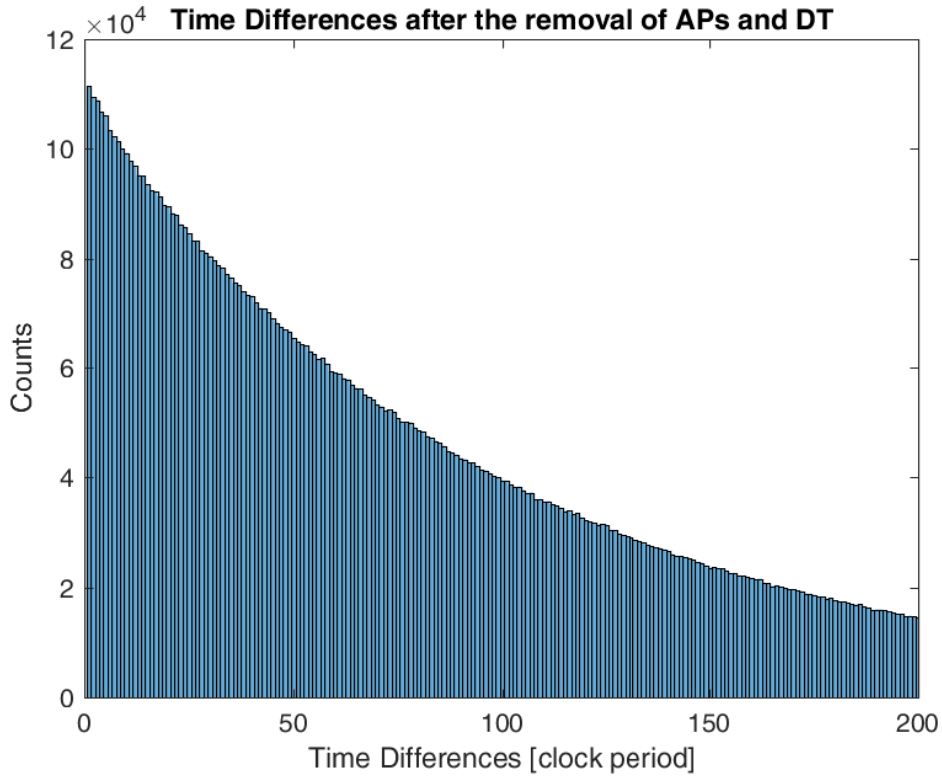

FIGURE 5.4: Time differences histogram after afterpulses and dead time removal.

To increase the total generation rate, the system also applies Peres algorithm directly to the samples string. As stated before, the FPGA saves a logical zero for every clock cycle whenever the SPAD signal is low, a logical 1 otherwise. The raw string is then post-processed on a dedicated CPU[2]. Due to the huge difference between the clock rate and the photon rate (100 MHz over 200 kcount/s), the raw strings are heavily biased in zero with a percentage of 99.8%. The role of Peres algorithm is to reduce the bias to, ideally, 50%. Nevertheless, Peres cannot work around any correlation. On the contrary, it could emphasize it. Therefore, any correlation must be removed before the application of Peres. In this case, correlation comes from afterpulses and dead time of the detectors. Further, a description of how to remove them is given.

The distribution of the time interval between consecutive events is expected, for ideal Poissonian events, to be a decreasing exponential. As shown in figure 5.3, the experimental distribution differs from the ideal distribution by two effects whose values depend on the specific SPAD model used[3]. The first effect is the *dead time* and represents the minimum time distance between two rising edges of $\mathcal{S}_{\text{det}}(t)$. In this case $\tau_{\text{dead}} \simeq 5$ clock cycles (50 ns). The second one is *afterpulse*. Afterpulses are spurious events that could occur within a fixed time interval from a real detection. As a result, they produce a peak which undermines the exponential trend of the events difference distribution. The cross-point between the only-true-events region and the afterpulses-region was evaluated to be equivalent to 18 clock cycles (180 ns).

Dead time and afterpulses introduce correlations in the output string as shown in figure 5.5 (top and bottom). Clearly, after a value $x_j = 1$, some of the few following bits in X are not independent. From the evaluated afterpulse threshold, it is possible to remove the correlation by simply remove 18 values of X following any value $x_j = 1$.

As a result, this procedure eliminates a portion of valid random events which is around 14%. However, the resulting distribution of the difference between consecutive 1's follows the expected decreasing exponential (see figure 5.4 bottom). This is unavoidable in order to minimize the probability of keeping multiple afterpulse events. Once dead time and afterpulse are compensated[4], Peres algorithm is applied to the bit string. With an actual photon count rate of 172 kcounts/s (due to afterpulses removal) and an actual sampling frequency of 97 MHz (due to dead time removal), Peres algorithm has a final true random bit rate of 1.8 Mbit/s according

[2]The realization of a dedicated FPGA design for the Peres algorithm was taken into account. However, due to the recursive nature of the algorithm and its non-deterministic convergence, the required hardware resources could be way too much and the design process quite demanding. The implementation of the algorithm on the Board software or on the PC software was taken into account as well but would not bring significant results besides the ease of use of the system and minor improvements in the computational time. Anyhow, future improvements will consider these options.

[3]The SPCM-ARQH by Excelitas [116].

[4]For the sake of completeness, also dark counts should be taken into account. However, the dark count rate was equal to 100 counts/s and thus totally negligible.

to the rates shown in figure 5.1. As shown in figure 5.6, the autocorrelation over the output string is within the limits of statistical acceptance.

5.2.1 Two SPADs for improved entropy

To increase the overall entropy, the setup was doubled. Another light source and another SPAD were added to the setup. The doubled system allows either two different random number streams or a merging of the two. The merging was done through an XOR port which allows to combine the entropies of the two streams. This function is not available with Peres procedure but only allows to merge the random numbers produced directly on the FPGA with the generation protocols. A brief test on two 30 Mbit random strings was done to evaluate the entropy improvement. The strings were produced with the Diff-QRNG protocol. For the first string only one SPAD was used, while for the second one the XOR option was used. The analysis showed that the entropy was equal to 1 with an approximation of 10^{-9} in the first case and of 10^{-12} in the second one.

5.3 FPGA design

The FPGA design concerned the development of custom VHDL blocks, the setup of IP cores and the required connections. A schematic diagram is visible in figure 5.7. The diagram shows the main blocks of the design recalling the Vivado block design. It also highlights the different clock domains with different background colors.

5.3.1 Time domains

In the project different time domains needed to be taken into account. The first one was the one coming from the clocking wizard module which produced a 100 MHz clock signal distributed to each VHDL custom block. Throughout this work, this domain is indicated as the main one. As shown in figure 5.7, this domain is identified by the green background color, i.e. all the blocks in the green area belong to the main clock. On the contrary, all the AXI modules use a different clock signal which comes from a different oscillator and can be set by the Zynq PS module which outputs this clock on one of its ports, namely FCLK_CLK0. This time domain is identified by the azure color in the schematic diagram.

Actually, the two BRAM memories have two different clocks, one for each port. The port controlled by the Tags Manager/Bit Ben belongs to the main clock while the other port, controlled by the AXI BRAM module, is clocked by FCLK_CLK0. In fact, a BRAM memory can be used to transfer data from one time domain to another since its ports are completely independent.

These two time domains are combined also by the Reset Handler module which controls the reset of the hardware (see below for more details).

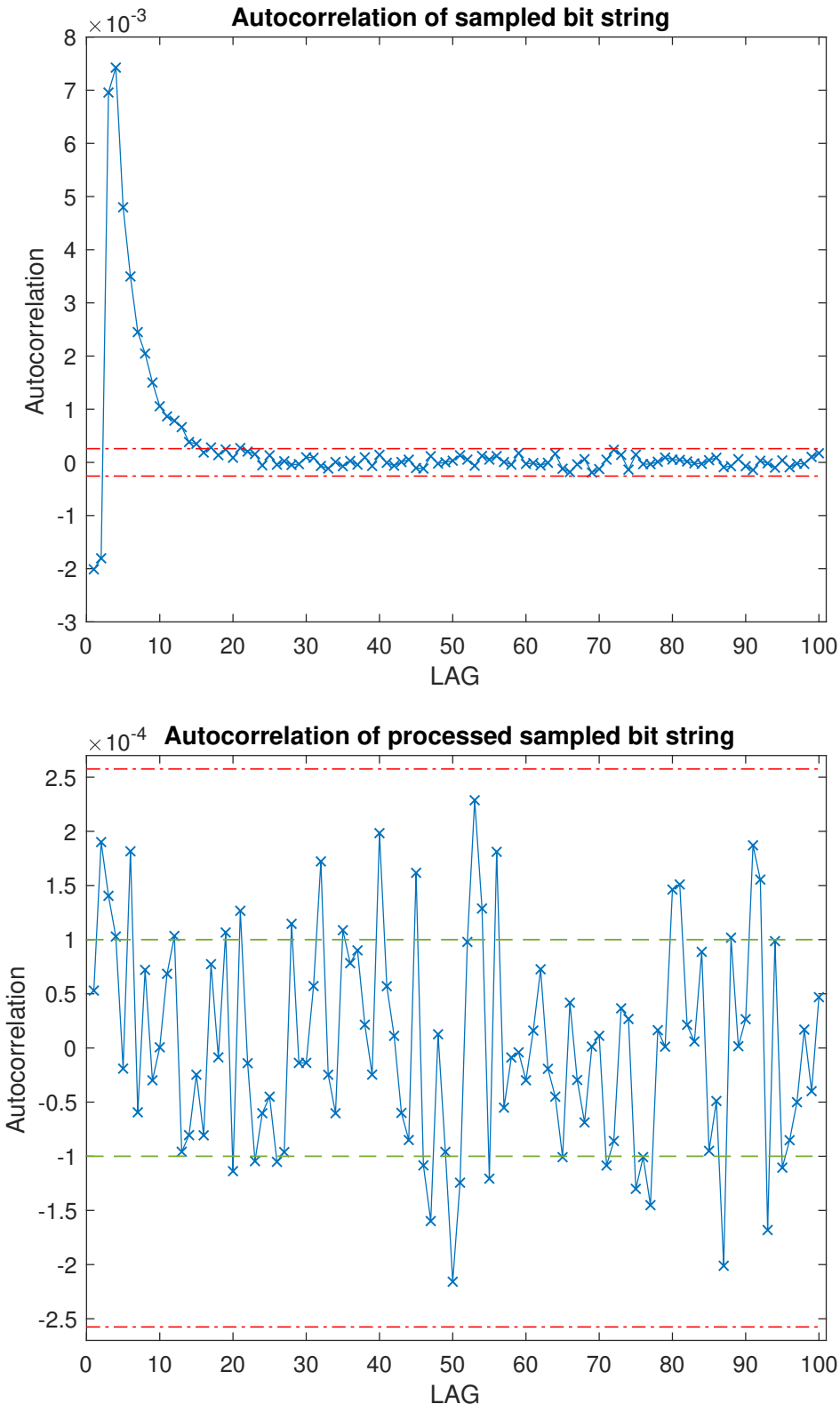

FIGURE 5.5: *Top*: the serial correlation coefficients evaluated on a sampled bit string without any post-processing. Clearly, the visible correlation on lower lags is due to dead time and afterpulse. *Bottom*: the serial correlation coefficients evaluated on a post processed sampled bit string. The correlation introduced by dead time and afterpulse is removed. Green dashed lines represent standard deviation while red dot-dash lines are 99% confidential limits.

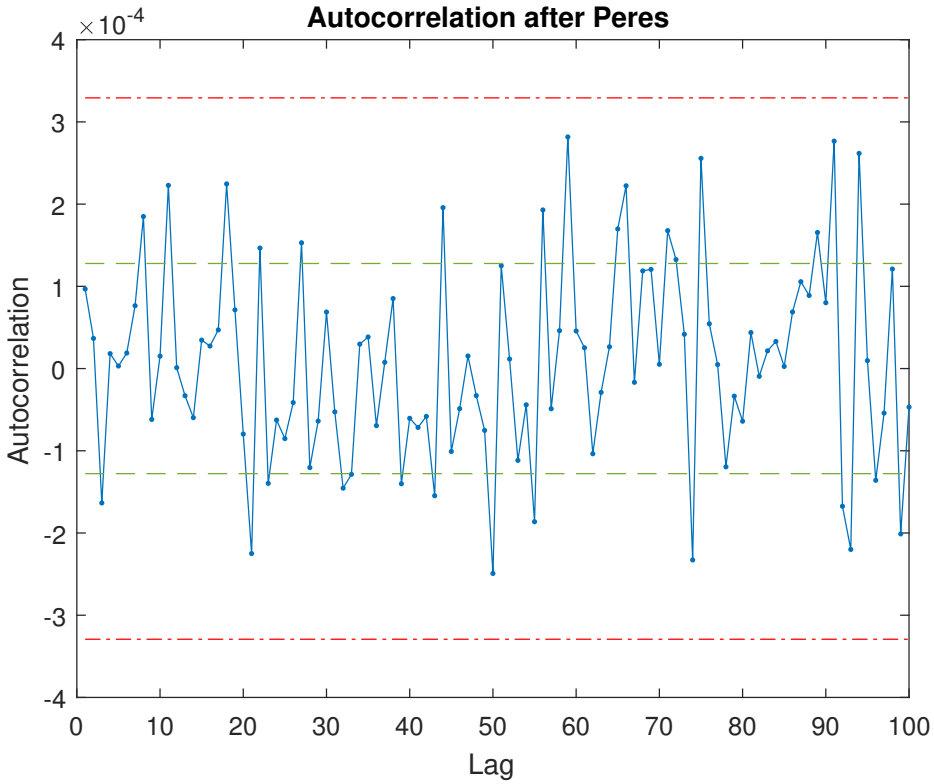

FIGURE 5.6: The serial correlation coefficients evaluated on a bit string debiased with Peres. The serial correlation re-enters within the limit of acceptance and shows no behavior change in respect to the autocorrelation plot of figure 5.5 (bottom): Peres algorithm works just on the bias and not on the autocorrelation. Green dashed lines represent standard deviation while red dot-dash lines are 99% confidential limits.

Other time domains to deal with were the ones which belonged to the external world, i.e. the SPAD signal and the pushbutton. These different time domains are highlighted by orange shades. These two signals are read by VHDL modules in the main clock domain. Therefore, it was necessary to make a time domain translation to deal with these signals. The two blocks responsible for the translation are the Reset Handler and the Async-to-Sync modules which are analyzed further below.

5.3.2 Async-to-Sync

This module is responsible for the temporal translation of external signal into the time domain of the design. Specifically, this module samples the input port according to the main clock and outputs a synchronized version of the input signal with a bit mapping $0 \rightarrow 0$ and $1 \rightarrow 1$. Therefore, it produces a delayed version of the input signal. The delay depends on the phase shift between the main clock and the sampled signal. Metastability may arise if the SPAD signal asserts too close to the main clock rising edge. The problem was handled with a standard solution, i.e. a series of three flip-flops which adequately reduced the probability of a metastable signal.

5.3.3 Events Counter

The *Events Counter* module is used to evaluate the number of events within a desired time interval. The used modules are two: one for the OddEven-QRNG implementation, where the time window is set according to the photon rate, and one for the calibration of the setup where the time window is set according to the user choice (usually 1 second) in order to estimate the overall photon rate and calibrate adequately the light source. The counter module used for QRNG outputs its value to the Random Number Generator module which then evaluates the even or odd parity of the count number while the other counter outputs its value to a dedicated AXI GPIO.

5.3.4 Tag Creator

The *Tag Creator* block is the heart of the sampling process. By using a proper counter it evaluates the time distance between two events and produces a tag which represents that number of clock cycles. The tag is then output to the Random Number Generator to implement the Diff-QRNG protocol and is also output to the Tag Manager module to store it and save it on the computer. For encoding a tag value a 32 bit unsigned number is used. This bit length allows an easier managing over the BRAM-DDR transfer and it can guarantee not to overflow even with very low photon rate.

5.3.5 Tags Manager and Bit Ben

These two blocks are responsible for the data storing. They accumulate data and output them to a BRAM port according to a specific memory address. Furthermore, they keep notice of the position within the BRAM and send an interrupt signal when the data address corresponds to the half or the end of the memory, as explained in section 2.2. The BRAMs are set to store 32-bit element. The only difference between the two blocks is the data length. In the *Tags Manager* a single datum has a length of 32 bits (the tag) while in the *Bit Ben* a single datum corresponds to just one bit (the random bit). Therefore, while the *Tags Manager* immediately outputs the received tag to its BRAM, the *Bit Ben* accumulates multiple bits to form a word of 32 bits and then it outputs to the BRAM.

5.3.6 Random Number Generator

The *Random Number Generator* is the block where the magic happens; it implements the generation protocols described in section 4.2. When the Diff-QRNG is selected, it waits for tag values from the Tag Creator module and compares them to produce a random bit. When the OddEven-QRNG is selected, it evaluates the value received from the Events Counter module and produces a random bit. In both cases, the produced bit is output to the Bit Ben which is notified of the presence of a new random bit by the assertion of the NEW BIT signal.

5.3.7 Purificaxor

The *Purificaxor* module is used to implement the XOR operation for the merging of two different random numbers streams, as explained above in subsection 5.2.1. For the sake of simplicity this block is not shown in figure 5.7. Besides, figure 5.7 is referred to a single SPAD setup. To use both SPADs this architecture was doubled and the Purificaxor block acts as a link between the two architectures. This module also manages which random bits from the two streams must be XORed. The bits from both streams are stored on two different FIFO-like buffers. The XOR operation is done taking the first bit of the first buffer and first bit of the second buffer. Then, it outputs the new random bit and shifts the remained bits by one position in the buffer. It may happen that one of the two buffers accumulates a huge amount of bits without XORing any of them while the other buffer is empty. In this case the non-empty buffer is reset to avoid to make a XOR operation between two bits generated in very different time moments. The number of bits by which the buffer must be reset can be set by the user on the GUI. A standard choice is to set 10 as a limit value. In other words, if one SPAD "generates" 10 bits while the other one "generates" no bits at all, the 10 bits are thrown away and the generation starts all over.

5.3.8 Reset Handler

This module is designed to deal with the reset of the system. It allows to reset all the VHDL blocks (not the AXI ones) through software (via a dedicated AXI GPIO) or manually with a push button. A flip-flop reset signal can be asynchronous or synchronous with respect to the flip-flop clock. These two design philosophies offer different advantages and disadvantages. The chosen reset strategy is an hybrid solution which guarantees to get the best from both the asynchronous and synchronous reset styles. According to [117], the definitive solution is an asynchronous reset and a synchronous reset removal, i.e. the reset signal deassertion must be synchronous to the clock in order to avoid any metastability. Therefore, the module allows to produce a reset signal which assertion is synchronous with the input (pushbutton or AXI GPIO) and deassertion is synchronous with the main clock. It also implements a debouncing procedure to avoid debounce during the pushbutton pressure.

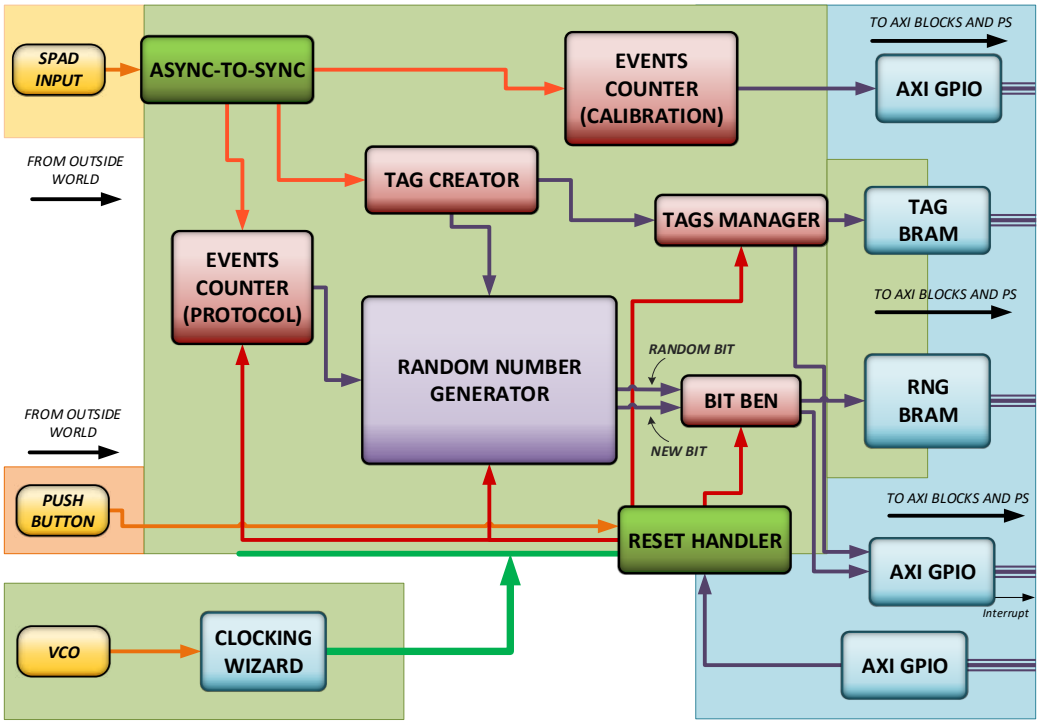

FIGURE 5.7: Randy block diagram

5.3.9 BRAM memories

In the block design, three different BRAM memories are instantiated. A first BRAM memory, called RNG-BRAM, is used to store random numbers from generation protocol blocks output. A second BRAM memory, TAG1-BRAM, stores tag values of SPAD 1 while the last BRAM memory, TAG2-BRAM, stores tag values of SPAD 2.

5.3.10 Interrupts

The system uses different interrupt signals. The first one is the one responsible for data moving from the RNG-BRAM. The second one and the third one are responsible for data moving from the TAG1-BRAM and TAG2-BRAM. Another two interrupts are responsible for the updating of the SPADs count through two dedicated AXI GPIOs[5]. The interrupt signals are linked to the IRQ_F2P input port of the ZYNQ Processing System IP Core. The port has a 5-bit depth and a Concat IP Core [118] is used to concatenate the five interrupt signals. Interrupts priority[6] can be chosen following the bits order of the bus: LSB has highest priority while MSB has the lowest one. Priority chart was set according to the importance of the respective data. The order is the following: RNG-BRAM interrupt, TAG1-BRAM, TAG2-BRAM, SPAD1 count, SPAD2 count. RNG-BRAM, TAG1-BRAM and TAG2-BRAM interrupts follow the memory management procedure described in section 2.2. For SPAD1 count and SPAD2 count interrupts a dedicated routine was developed: it simply reads the corresponding AXI GPIO input ports, updates a software variable and communicates that value to the Qt software in order to update the *Events Counter* on the GUI (figure 5.9).

5.4 Area, power and timing

There were no requirements on area and power consumption and no particular design or compiling strategy was chosen. As a matter of fact, the only area limit was the size of the chip itself while there was no power limitation. For example, in the application described in chapter 6, the device was put in a huge equipment setup which required conspicuous power consumption many orders of magnitude higher than the device one.

For the sake of completeness, the area and power statistics are shown in figure 5.8. The first graph shows that the hardware usage nearly reaches the 25% and that the most prominent element is the BRAM. Hence, despite the many functions implemented on the chip, there was still room for other hardware. The second one shows the overall power consumption and its partition between the chip elements. It is notable that most of the power (82 %) is required by the PS7 (the Zynq PS module). By the way, the overall power consumption is totally negligible for a standard environment application. Possible future developments for specific applications, e.g. space environment application, will investigate how to reduce power consumption.

[5] According to [49], an AXI GPIO can send an interrupt signal every time there is a change on its input ports.

[6] When two different interrupts occurr at the same time, the system handles the high-priority one for first.

Furthermore, the timing performances were evaluated. In a 100 MHz based design, which is a standard option for Zynq-7000, it is unlikely to have timing violations and in fact the design presented no violations at all. This means that there is no possibility of unpredictable behavior.

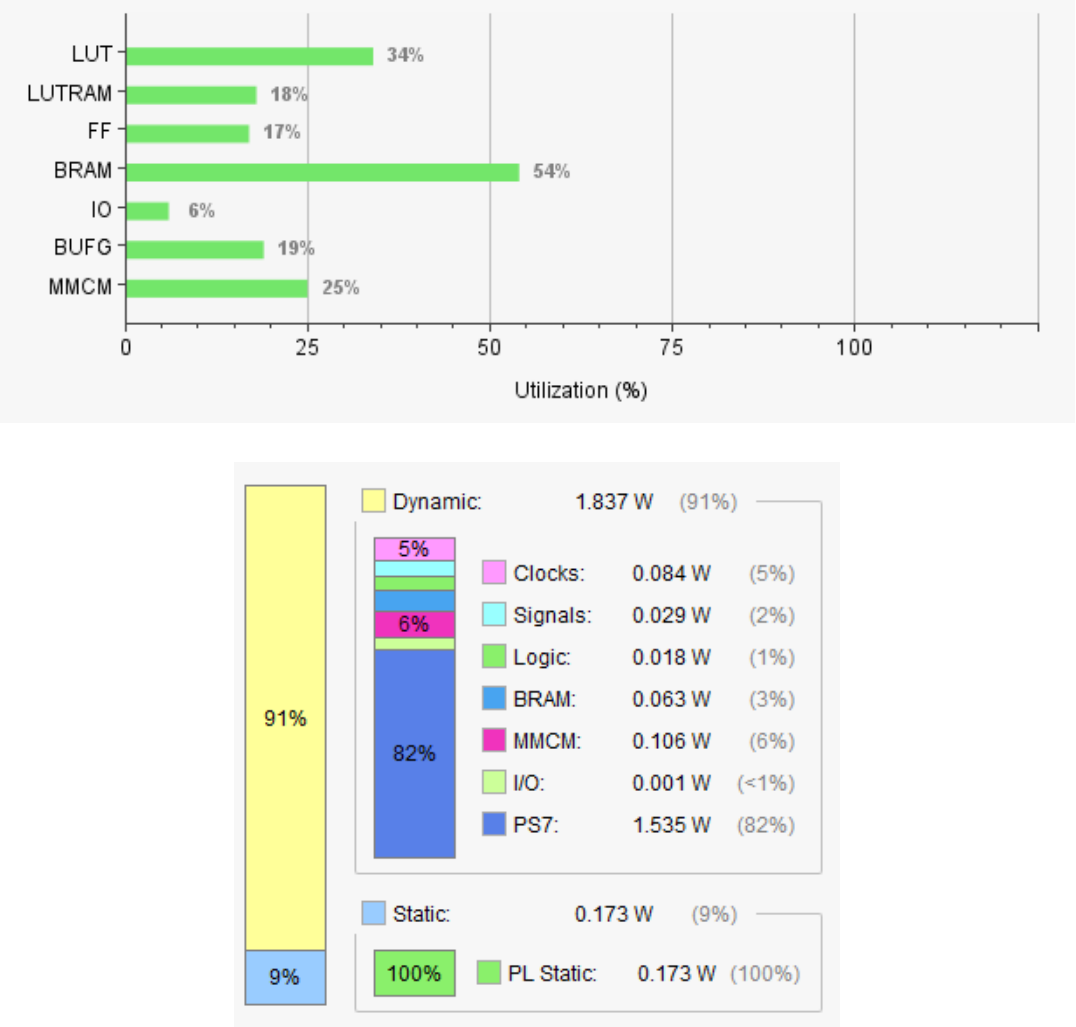

FIGURE 5.8: Randy Zynq-7000 utilization graphs. *Top*: area utilization. *Bottom*: power consumption.

5.5 Softwares

In this section a brief description of the two softwares is given. The softwares working principle is described in detail in chapter 2 and no noticeable changes were introduced for Randy QRNG.

5.5.1 ZYNQ CPU software

The software is based on the one described in chapter 2 and was adapted to control the added functions in the hardware. The software implements all the functions described in chapter 2 guaranteeing the memory transfer and the parameters set. Due to the high number of parameters, there is an intensive usage of the `XGpio_DiscreteWrite(instance name, channel number, value)` function. The parameters values are received via TCP connection as described in section 2.3. No other particular procedures or design, different from the ones described in chapter 2, were required.

5.5.2 RandyApp

The Qt application, called *RandyApp*, was developed in order to be the main reference for the whole setup. Therefore, a clear and well organized GUI was designed (figure 5.9). It allows to:

- connect to the device, i.e. initialize the TCP connection (top-left corner pushbutton)
- start and stop the random number generation (top-left pushbuttons). Pushing one of these pushbuttons allows to send a parameter value to the board software which in turn sets a dedicated AXI GPIO output port which is connected to the VHDL modules (Events Counters, Random Number Generator, Tag Creator,...) as a start/stop signal. When the signal asserts, the modules FSMs move out from their IDLE state and start their function. On the contrary, when the signal deasserts, the FSMs return to IDLE state no matter what they are doing.
- set the "key" length. Before starting to generate random number it was necessary to decide the length of the random string (for example 100 Mbit)[7].
- set *Purificaxor* options (left side). This frame can set which option between SPAD1, SPAD2 or both (with XOR operation) has to be used. It also allows to set the buffer size (see subsection 5.2.1 for details).
- set which generation protocol must be used (*Generation Type* frame). *Tag Middle* is the name of the Diff-QRNG protocol while *OddEven* corresponds to the OddEven-QRNG. In the GUI two extra protocols are visible, *Tag Short* and *Tag Long*. These two protocols are a modified version of the Diff-QRNG and were prepared for a possible future investigation on random generation. *Window Size* spin box is used to set the time window in which evaluates the number of detected photons.

[7]The term "key" was chosen due to a mixture with the QKD world where a cryptographic key has to be truly random for unconditional security.

- manage the events counter (*Events Counter* frame). With a proper check box it is possible to enable the events counter to have a real time monitor of the number of photons sent to the SPAD. The detected number is shown in a QLCDNumber element [119]. *Window Size* spin box is used to set the time window in which evaluates the number of detected photons. Usually this value is set equal to 1 to have the counts/s value. This feature was necessary to know the operating point of the SPAD and also to avoid any possible damages or malfunction.

- select whether to store the raw samples without applying any protocols. This option is used to extract the unprocessed tags and then apply Peres algorithm. It also features a *Sample Mode* which extracts the sampled string, i.e. a sequence of 0s and 1s. This option is used just to do a cross-checking about the correct functioning of the hardware.

The *BIT BEN Interrupt* check box is just a service option to make some verification tests on the interrupt procedure.

The *Selfxor* frame was prepared to control an extra option called self-XOR which nevertheless was never used and therefore is not described.

On the right part of the GUI two frames are visible; they were used for the Wheeler's delayed choice experiment and were implemented specifically for that application. More details are given in section 6.3.

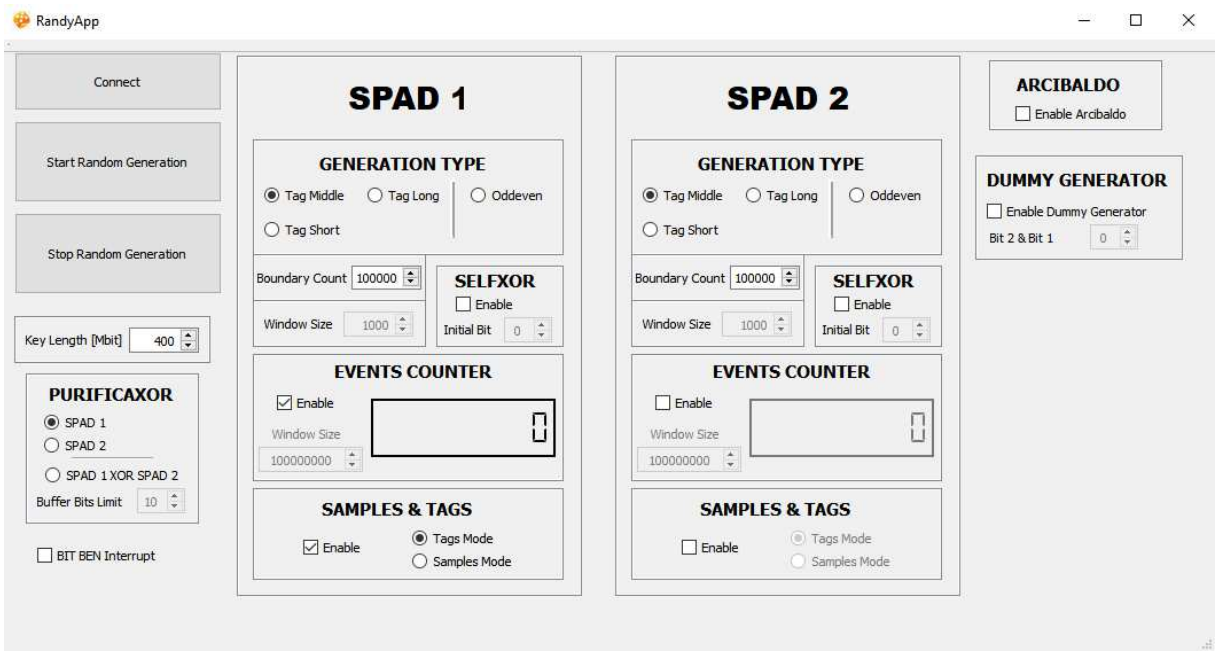

FIGURE 5.9: A view of the main window of RandyApp.

Chapter 6

Real time QRNG for quantum mechanics experiment

This chapter describes the realization of the real time QRNG device used in the Wheeler's delayed-choice experiment extended to space [98]. The QRNG was designed as a modified version of Randy with required changes that allowed to interface it with the experiment apparatus and output random number in a specific time window.

6.1 Wheeler's delayed-choice experiment

Wheeler's delayed-choice is a Gedankenexperiment proposed by J. A. Wheeler [120, 121]. The experiment investigates the dual nature of light showing its counterintuitive dichotomy between *particle* and *wave*. The experiment has been realized using different setups and in various frameworks [122, 123]. Our contribution extended the Wheeler's delayed choice experiment to space scale [98]. The experiment was realized at the *Matera Laser Ranging Observatory* (MLRO) (more details in section 9.2).

The paradoxical interpretation of the experiment is that a photon travels through the first beam splitters (BS) of a Mach-Zehnder interferometer (MZI) and "decides" its behavior according to the presence or not of a second BS at the outside of the MZI. If the BS is not in the apparatus, the photon acts like a particle choosing only one path of the MZI and randomly making fire just one of the two detectors. On the contrary, with the BS presence the photon shows its wave nature by creating interference highlighting the fact that it traveled both routes. If the measurement choice is performed after the entrance of the photon in the MZI then a classical interpretation leads to a seeming violation of causality while a quantum interpretation does not have such drawback.

6.2 Experiment setup

A faithful realization of Wheeler's experiment requires that the entrance of the photon in the interferometer is not in the future light cone of the measurement choice. Moreover, the latter must be realized in a random manner: this prevents any causal

influence of the measurement choice on the behavior of the photon. The implementation is performed over a space channel with a length of the order of thousands of kilometers, corresponding to a Round Trip Time (RTT) of the order of 10 ms. We designed the experiment to guarantee that the choice of the measurement apparatus is space-like separated from the reflection of the photon from the satellite, as shown in the Minkowski diagram in figure 6.2. This guarantees that, in a purely classical interpretation, a photon "should have decided its nature" at most at its reflection from the satellite.

The setup is shown in figure 6.1. A pulsed laser[1] diagonally polarized and paced by an atomic clock, entered into an unbalanced MZI. Due to the presence of a polarizing BS (MZI-BS) and of the imbalance of the MZI, every laser pulse was transformed into a superposition of two temporal and polarization modes. The two polarization components traveled along two different paths of the MZI: the horizontal polarization along the short arm while the vertical one along the long arm. The pulses then passed through two *Liquid Crystal Retarders* (LCRs) whose combined action was equivalent to a single *Switchable* (on/off) *Half–Wave Plate* (sHWP) inclined at 45° with respect to the fast axis. During the transmission period, the sHWP was always off, producing no perturbation on the outgoing beam. The light was then directed to a target satellite equipped with polarization-maintaining corner-cube retroreflectors via a telescope. The corner cubes of the target satellite redirected the beam back to the ground station.

The photons returning from the satellite were collected by the same telescope and injected into the optical table, where they re-encountered the same sHWP and MZI. At an exit port of the MZI-PBS, a polarization measurement was performed in the diagonal and antidiagonal basis $\{|+\rangle, |-\rangle\}$ with $|\pm\rangle = (|H\rangle \pm |V\rangle)/\sqrt{2}$, where $|H\rangle, |V\rangle$ were the horizontal and vertical polarization states, respectively.

For each cycle, we performed two independent choices that will affect the detections in the acceptable temporal window t by driving the QRNG with the same FPGA used for the shutters that separate the transmission phase from the reception phase. The sHWP behavior at the photon return was set according to the bits b_1 and b_2 extracted by the QRNG. The bit value set the voltages V_b applied to the LCRs, determining the on or off behavior of the sHWP. The latter determines whether we performed a measurement that reveals the particle-like (sHWP on) or wave-like (sHWP off) behavior of the photons returning from the satellite. The first choice is performed at t_{b1}, corresponding to the middle of the shutter transition phase. The second choice is at t_{b2}, which occurs with a delay RTT/2 with respect to the first choice. The detected photons are divided into two groups, each characterized by a value of the bit choice. In this way, all the photons of a given group were already reflected by the satellite when the corresponding bit choice was performed.

[1] Repetition rate, 100 MHz; wavelength λ = 532 nm; energy per pulse, $\sim$ 1 nJ.

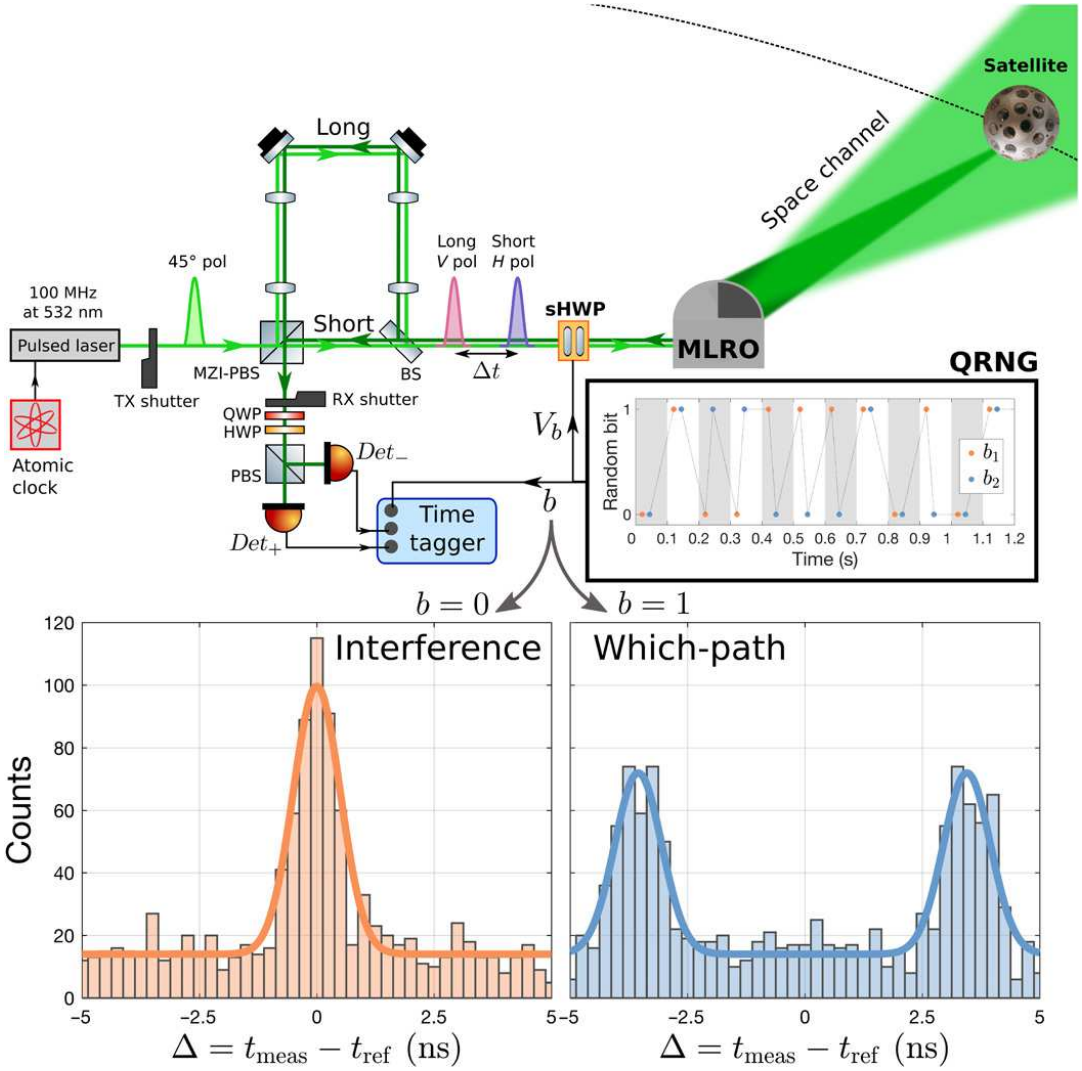

FIGURE 6.1: Scheme of the experimental setup and detection histograms. A pulsed laser synchronized with the MLRO atomic clock exits the MZI in two temporal and polarization (pol) modes. The sHWP leaves the pulses unperturbed, and the telescope directs the beam to a target satellite. After the reflection, the photons are collected on the ground by the same telescope and injected into the optical table. The photons pass through the sHWP whose behavior is set according to the bit b extracted from an on-demand QRNG. The QRNG is inquired twice in each 100-ms cycle of the experiment, as detailed in the main text. In the inset, a 1-s sample of the extracted bits is shown. At the MZI output, two wave plates, a PBS, and two single-photon detectors (SPDs) perform a polarization measurement in the $|+\rangle$, $|-\rangle$ basis. According to the value b of the random bit, interference or which-path measurement is performed, as shown by the detection histograms for a passage of the Starlette satellite. The counts in the central peak on the left histogram are comparable to the sum of the counts associated to the lateral peaks on the right one, as expected. HWP, half–wave plate; QWP, quarter–wave plate. Image and caption are taken from [98].

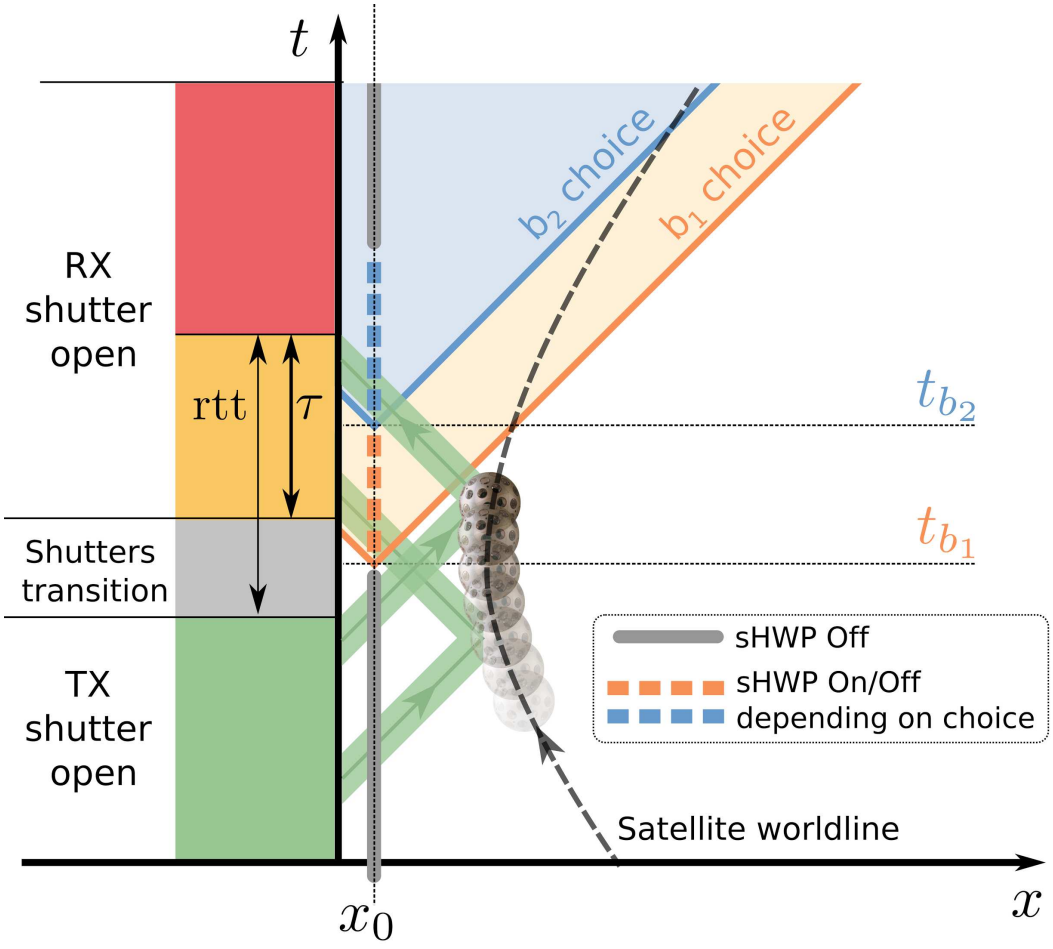

FIGURE 6.2: Along the temporal axis (not to scale) a 100-ms cycle between two SLR pulses is represented. The x axis represents the radial coordinate (not to scale) from the detectors, where x_0 is the position of both the sHWP and the QRNG. The dotted line is the satellite worldline. We only considered the detections in the temporal window τ, as detailed in the main text. A fast FPGA controller synchronized in real time with the MLRO tracking system drives the two shutters and the QRNG. For each cycle, we performed two independent measurements via the random bit extracted by the QRNG at times t_{b1} and t_{b2}, causally disconnected from the photon reflection at the satellite. The cycle is repeated for each 100-MHz train between two SLR pulses. Image and caption are taken from [98].

6.3 Adapting Randy to the experiment

In order to use *Randy* for the realization of the experiment, some changes were required. Main changes regarded the synchronization of the QRNG with an external trigger coming from the experiment and the possibility of setting the sHWP according to the QRNG output. The design mostly concerned the FPGA part and is described in the following subsections. Some minor changes were also made on the software parts to control the system but their description is omitted since they did not required the development of particular solutions.

6.3.1 Synchronizing the QRNG

In figure 6.3 a block schematic of the synchronization process is visible. Randy received a trigger signal from the experiment setup, it made a comparison with its local time reference, i.e. the clock, and then reset the QRNG and produced fresh random number which could be retrieved from the experiment. For this application, the usable generation protocols were the OddEven-QRNG and the Diff-QRNG. As explained in chapter 5, these protocols allow a real-time generation while Peres choice was not an option. In the final experiment, we chose to use Diff-QRNG since it requires a shorter time to produce random number. We also decided to use the double mode for higher entropy (see subsection 5.2.1 for details). Fixing the photon count rate to 200 kcounts/s, the Diff-QRNG has a bit rate equal to 100 kbit/s which leads to an average of one random bit every 10 μs. This interval was quite negligible than the magnitude order of the round trip time. Nevertheless, on the off chance that a bit was not produced in time[2], no QUTAU AUX signal (see next subsection) was produced, which made clear that nothing happened on that round of the experiment.

6.3.2 FPGA design

The FPGA design of Randy was modified as shown in figure 6.4. Major changes regarded the introduction of new input and output ports as well as custom VHDL blocks and some AXI GPIO modules for parameters setting. Description of new AXI GPIOs linking and usage is omitted since they were quite similar to the ones described in chapter 5. Unchanged blocks and links are mostly omitted from the schematic for the sake of clearness.

Two inputs ports were added, MLRO START and MLRO STOP. These two signals indicated when a laser ranging pulse was directed to satellite and received back. Two extra Async-to-sync module were added to move these two signals from their time domain to the FPGA one. The synced signals were used to determine the round trip time which the whole synchronization relied on. Furthermore, three output

[2]According to the Poisson distribution formula, the probability to detect just four photons within RTT/2 is in the order of magnitude of e^{-1000}.

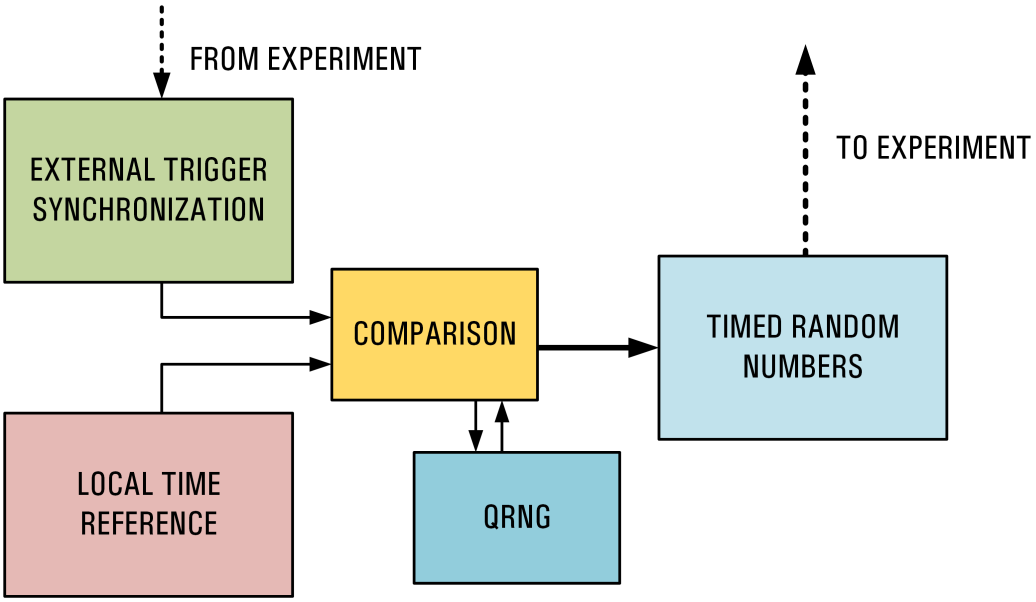

FIGURE 6.3: A concept schematic of the synchronization structure used for synchronizing random numbers in the Wheeler's delayed-choice experiment.

ports were added: QUTAU AUX, LC1 and LC2. All these three ports were controlled by *Arcibaldo* block.

Arcibaldo block was the main block responsible for synchronization. Three different FSMs were designed. The first one, the *Main FSM*, was responsible for clearing the QRNG and for setting the two Liquid Crystals. The second one, *quTAU FSM*, output the PWM code to the quTAU while the third one, *Delta FSM*, evaluate the round trip time.

In the Main FSM, a *CLEAR QRNG* signal was sent at proper time to reset the QRNG blocks. This signal fed an OR port along with the reset signal coming from the *Reset Handler* module. Therefore, the QRNG modules could be reset by a standard reset signal (via pushbutton or via AXI GPIO) or by a CLEAR QRNG. The non-QRNG blocks like the Bit Ben, the Events Counter and Arcibaldo itself, could not be reset by the CLEAR QRNG signal but only by the standard reset. After a RTT/2 time, computed by the Delta FSM, the QRNG was reset by the CLEAR QRNG signal and then the FSM waited for the first random bit produced. Once the QRNG output the first bit, the LC1 was set and a new CLEAR-WAIT-SET loop was launched after a selected delay and a second random bit was produce to set LC2.

In the quTAU FSM, a coded signal was built and sent to the QUTAU AUX port which was connected to one channel of a quTAU device by a BNC cable. The quTAU device was used to collect all the significant signals during the experiment and it was necessary to know which was the bit choice. Since it would have been quite complicated to save and extract the generated random bits from the board, we decided to use a PWM-coded signal to communicate whether it was a 1 or a 0. With dedicated analysis on the tag data it was possible to reconstruct the signals evolution.

In the Delta FSM, the rising edge instant of MLRO START and MLRO STOP

(synced) were analyzed to determine the round trip time. The computed values were stored in dedicated variables which were then used in the main FSM. Therefore, the Delta FSM updated these values at every round. The Main FSM set the QRNG along with LC1 and LC2 according to the previous computed value and not the actual one. This was not a problem since the round trip time value changed very slowly. Furthermore it would be quite impossible to determine the actual value beforehand.

Arcibaldo block could be enabled by checking the relative checking box in Randy GUI (figure 5.9).

Dummy Generator block was a mock QRNG module that was used during the experiment calibration to simulate the generation of random bits and to set LC1 and LC2 deterministically according to the user's choice. On the right part of Randy GUI, shown in figure 5.9, a dedicated frame to enable and set the Dummy Generator is visible.

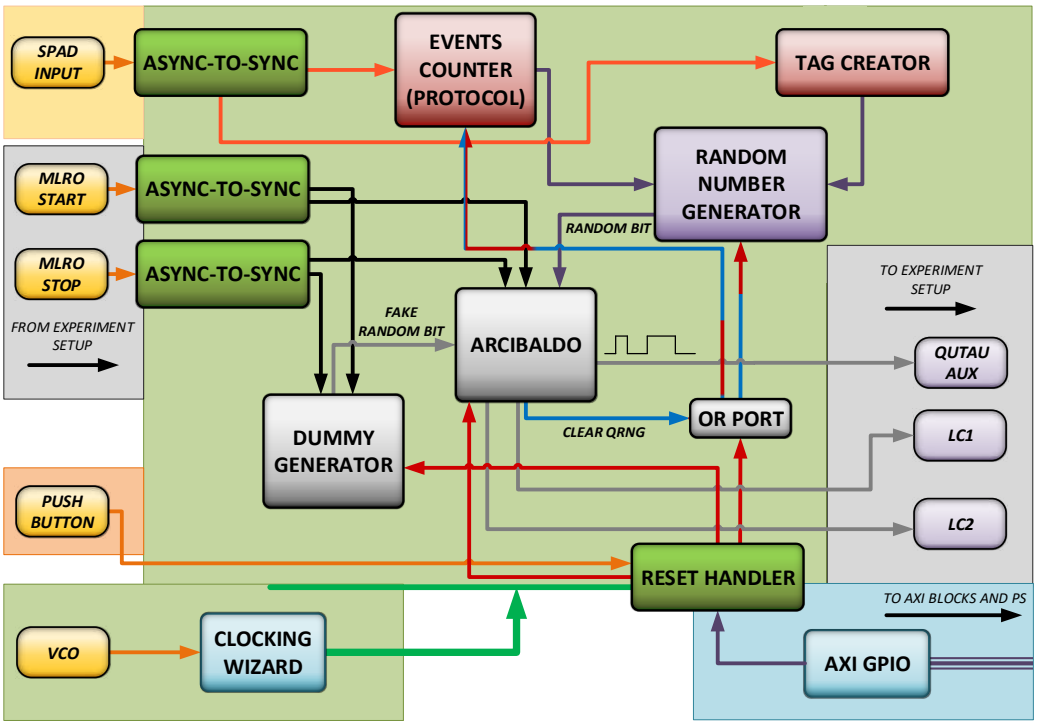

FIGURE 6.4: Randy block design for Wheeler's delayed-choice experiment. The schematic is similar to the one in figure 5.7 but introduces new VHDL blocks and new I/O ports.

Chapter 7

LinoSPAD Quantum Random Number Generator

This chapter gives a description of LinoSPAD device and the developed techniques for the random numbers extraction. These techniques are based on the ones already described in chapter 5 and are adapted to LinoSPAD.

7.1 Introduction to LinoSPAD: CMOS-SPADs array

The main limitation concerning the usage of QRNG based on SPADs, i.e. discrete variable QRNG, is the limited generation rate achievable. As a matter of fact, SPADs typically feature maximum count rates of a few Mcps (counts per second). QRNGs based on continuous variable protocols are therefore preferred when it comes to obtain rates in the order of Gbps [124, 125]. However, the recent advancement of miniaturization techniques, especially the creation of deep-submicron CMOS SPADs, has led to arrays and matrices with hundreds or even thousands of SPADs [126, 127]. Hence, each SPAD can be considered as a pixel of an extremely sensitive light sensor. Its application to random number generation is then straightforward: given that every pixel works independently, it is possible to multiplex the random signals and then fill the rate gap with CV-QRNGs [27, 128]. The typical approach is to generalize the paradigm of the welcher weg QRNG - a beam splitter and two photon paths - to a generator where photons can take N possible paths, with N being the total pixels number of the sensor. Nevertheless, like in the Randy case, the temporal degree of freedom can be exploited as well allowing higher bit rate and easier calibration. Therefore, the sensor is illuminated with light of uniform intensity so that each SPAD has the same probability to click within a given time interval, which corresponds to the exposure time for a frame. Random numbers are indeed produced by periodically sampling each pixel: every frame yields a bit string of N length, with bit 1 corresponding to a photon detection and 0 otherwise.

It is now considered a sensor of recent introduction, *LinoSPAD* [129], from the AQUA lab at Delft University & EPFL, which features 256 pixels arranged in four linear array of 64 pixels each. The peculiarity of LinoSPAD is a time tagging functionality, in addition to the spatial pixel coordinate, in order to associate a temporal

coordinate to every detection, thus potentially enabling a further increase in the generation rate. In the following, the application to LinoSPAD of some of the techniques described in chapter 5 is described. LinoSPAD features 64 FPGA-based Time-to-digital converters (TDCs) [130, 131] that tag the detections of each SPAD in a given bank. Each TDC is implemented by a delay line carrying 35 carry elements of 4 bits and it is sampled with a frequency $f_{\text{clock}} = 400$ MHz. At every time interval $\tau_{\text{clock}} = 2.5$ ns the TDC emits an output code $b \in \{0, 139\}$. The TDC has therefore a sub-resolution of $\tau_{\text{sub}} = \tau_{\text{clock}}/140 \simeq 17.86$ ps and this represents the fundamental time resolution of the system. The following considerations take into account that the actual number of valid pixel is 64 and not 256 due to the limitations to 64 TDC. The measurements of the photon time of arrival are taken with respect to a "reference" time signal, whose period determines the integration time of a frame. The buffer of the system can add output codes to a maximum of 2^{28} bins. Hence, the longest measurable time interval between the reference clock signal and a photon detection cannot be larger than $\tau_{\text{sub}} \cdot 2^{28} \simeq 4.8$ ms. Since the device registers a maximum of 512 tags per pixel during the integration time, every frame is composed at most by 64 x 512 tags. Like the paradigm adopted in the previous chapters, random bits can be directly extracted from the bare physics of the process: a string $\tau_{\text{frame}}/\tau_{\text{sub}}$ bit long is associated to each frame and bit 1 is collocated according to the tag values. A limit of this approach is that the strings are consistently biased towards zero. This bias is the result of two concurring causes: the first one is the dead time of the SPADs, $\tau_{\text{dead}} \simeq 40$ ns, which implies that every bit 1 is necessarily followed by $\tau_{\text{dead}}/\tau_{\text{sub}} \simeq 2240$ bit 0. However, the zeros due to the dead time can be removed, as previously done. The second cause lies in the limited buffer size: each string produced in a frame will always feature at most 512 bit ones. It was then evident that, given the constrained number of bit 1, the extraction approach, by means of Peres unbiasing procedure, can be indicated even for this framework. Like in Randy, in order to detect the artifacts induced by the physical limits of the device, the interarrival detections time was analyzed, the distribution of which is shown in figure 7.1.

The histogram starts approximately at 40 ns due to the dead time where a peak, that starts at 40 ns and extends up to 200 ns, indicates the presence of afterpulses. A noticeable feature is an unusual peaks pattern. This pattern was due to a non-linear behavior of the TDC (further details in section 7.3). Hence, to manage these non-idealities the tags resolution was split in *COARSE* resolution (Clock) and *FINE* resolution (TDC) implementing two different post-processing procedures. This separation allowed a more efficient and easy way to produce random numbers out of this device. The analysis was run on a collected data corresponding to the acquisition of $8 \cdot 10^3$ frames where each frame has an integration time $\tau_{\text{frame}} = 320 \cdot 10^{-6}$ s. The photon rate was tuned to obtain approximately 400 counts per frame. This value was chosen in order to keep low the probability to saturate the buffer of 512 detections and then loose the frame. Results show that the final achieved generation bit rate was equal to 300 Mbit/s.

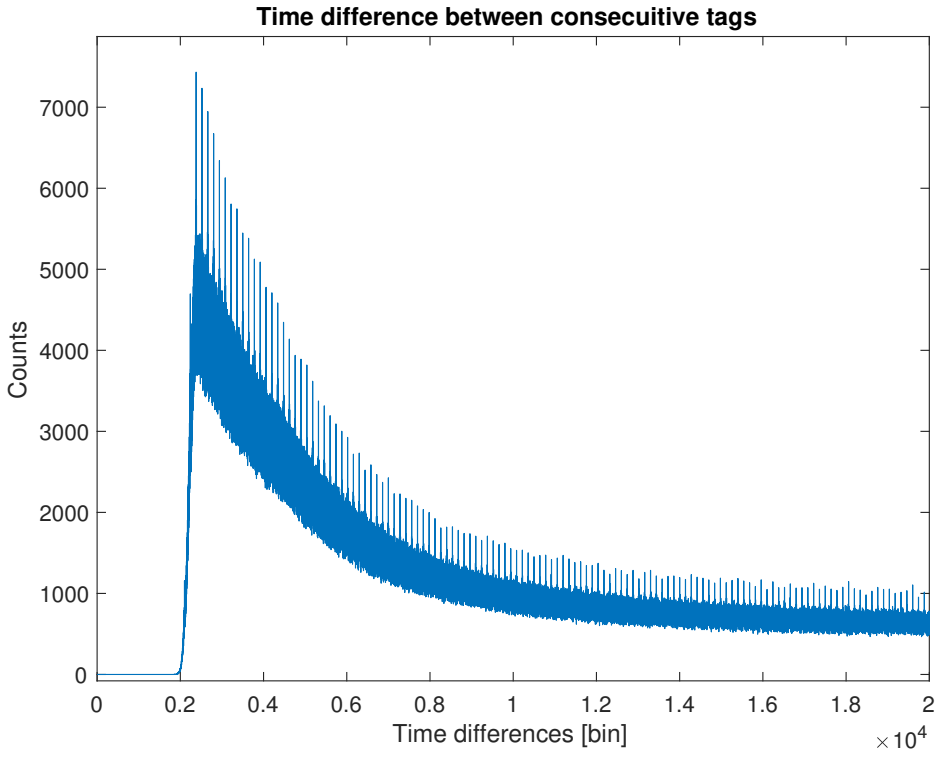

FIGURE 7.1: Distribution of the time differences between successive detections.

7.2 COARSE resolution

The *COARSE* resolution is defined by the 400 MHz system clock. The tag information is related to the number of clock cycle in which an event is detected. Therefore, it describes the temporal distance between an event and the zero reference with a resolution of 2.5 ns. The list of tags is ordered with respect to the zero reference giving an actual time progression of the detected events. SPADs dead time and afterpulses were removed with the same technique described in section 5.1. However, the SPADs array presents an issue defined as *pixel cross-correlation* which causes a pixel to output an event when its neighbor receives a photon, i.e. it produces a fake event like in the afterpulse. In order to remove it and to be sure that no cross correlation exists, approximately one third of the pixels of a bank is discarded losing all the events from those pixels. This procedure reduces the actual number of pixel from 64 to 22. Once the cross correlation is removed as well as the dead time and the afterpulse, Peres algorithm is applied to each selected pixel independently. As discussed in section 5.1 and according to figure 5.1, higher rates can be achieved by preferring longer heavily biased bit strings over shorter slightly biased bit strings. Therefore, each selected pixel is treated as an autonomous QRNG. The plot of figure 7.2 shows that autocorrelation stays within the limit of acceptance. The final bit rate summing the bit rate of the selected pixels is $\simeq$ 87 Mbit/s. The value is approximated since the events rate varies from pixel to pixel as shown in figure 7.3. Considering the average number of events per pixel and the actual number of pixels, the mean extraction rate

per pixel is equal to $\simeq 4$ Mbit/s.

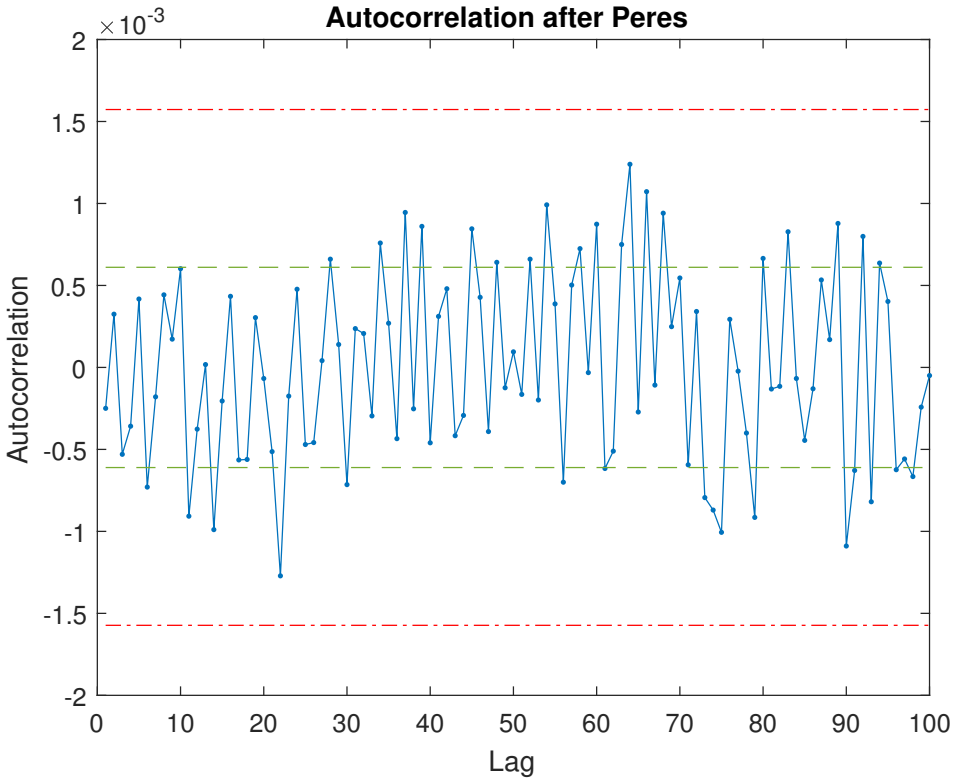

FIGURE 7.2: The serial correlation coefficients evaluated on a *COARSE* bit string of a single pixel after applying Peres algorithm. Like in Randy, the serial correlation is within the limit of acceptance. Green dashed lines represent standard deviation while red dot-dash lines are 99% confidential limits.

7.3 FINE resolution

The *FINE* resolution is defined by the TDC and has a time resolution of $\simeq 17.86$ ps. The TDC outputs a number between 0 and 140 which describes a precise moment within a clock cycle in which an event occurs. Hence, the access to the TDC value was done every clock cycle. Like *COARSE* resolution, every single pixel is treated independently. The peaks in figure 7.1 are exactly 2.5 ns far, which implies a dead time in accessing the TDC. This behavior brings to the tag distribution shown in figure 7.4 that represents an entire 2.5 ns clock cycle.

The figure shows that lower bins have no events while other bins have a significant bias. This behavior is due to the implementation of a TDC on an FPGA technology which introduces non-linearity caused by different propagation delays over hardware blocks. This distribution is far to be acceptable for randomness and the initial information entropy is equal to $\mathcal{H}_i = 6.8$ bits instead of $\mathcal{H}_i = 7.12$ bits, which is the maximum value achievable with 140 decimal values. Furthermore, this distribution introduces a correlation on the binary description of these events. Numbers

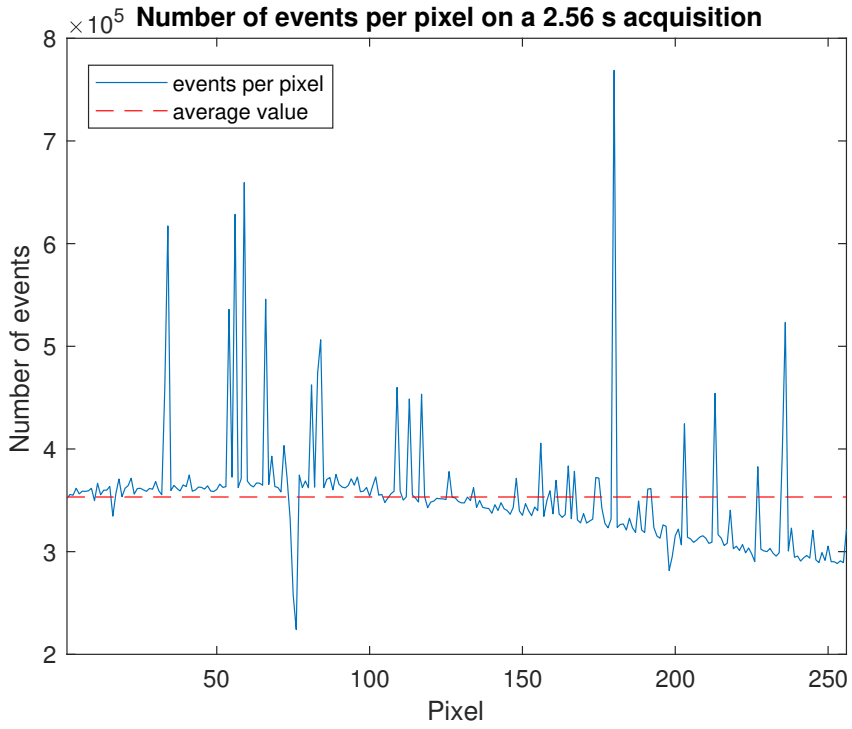

FIGURE 7.3: Number of events for each pixel. The graph shows that the actual number of events varies from pixel to pixel and some of them are quite far from the average value. By a dedicated measurement, the dark counts rate for every pixel was evaluated to be less than 10% of the events and thus it does not give a significant contribution to the plot. Hence, this behavior represents the pixels efficiency and is due to the physical realization of the chip.

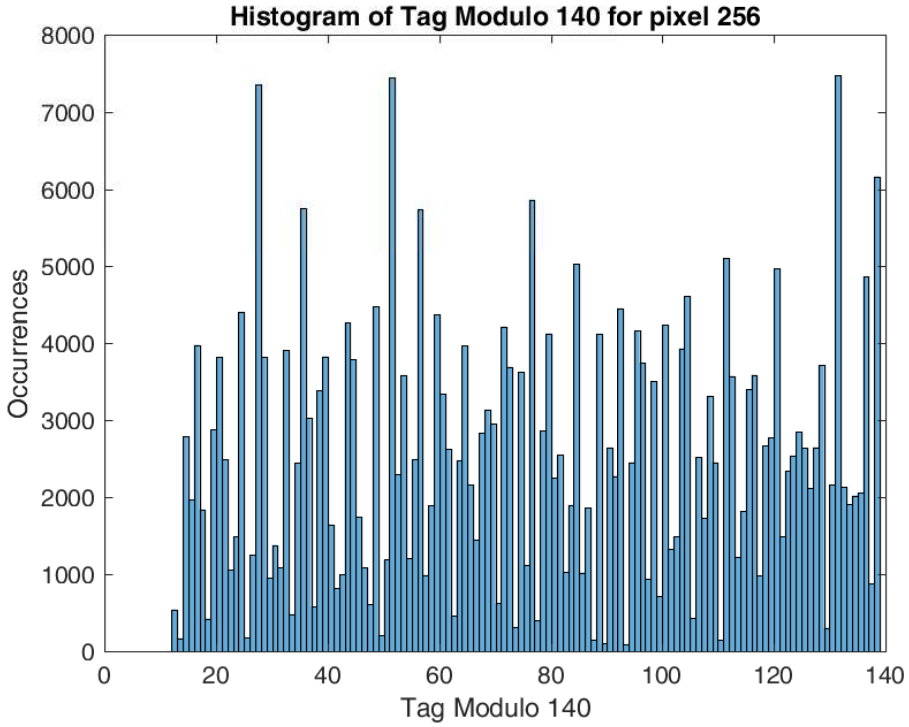

FIGURE 7.4: Distribution of the tags modulo 140 for a single pixel. The distribution shows the non linear response of the TDC disclosing more probable values (all the peak values) and zero-probability values (approximately between 0 and 20).

in decimal basis are just biased and not correlated; but, taken the binary description, the single bits are correlated due to the non uniform distribution. Moreover, in order to describe 140 different values, 8 bits are required implying that no events occur from 140 to 255, which worsens the situation even more. Clearly, Peres algorithm cannot be applied to such a string. In order to remove the bias and correlation on the modulo 140 distribution, the usage of a specific post-processing method was established: Zhou-Bruck algorithm which can generate random bits from loaded dice [115]. Such method removes the bits correlation and allows the application of Peres algorithm. The basic idea of this procedure is to group together the i^{th} bits of the binary description of a sequence of number. The way to group such bits depends on the the previous bits combination of the considered number. After this manipulation, one gets N different groups where $N = 2^b$ and b is the number of bits required for a binary description of the numbers sequence (in this case 8). Clearly, a group which corresponds to a missing value (0-18, 140-255) does not contain any bits. Then, it is possible to apply Peres algorithm to every non-empty group reaping an unbiased and uncorrelated bit string. For a full and precise description of the method refer to [115]. As a matter of fact, Zhou-Bruck method is quite efficient getting a final bit entropy of $\mathcal{H}_i = 6.3$ bits losing only 0.4 bit due to the algorithm. The obtained bits are evaluated in terms of bias (figure 7.5) and binary entropy extraction (figure 7.6) as well as usual correlation and p-value tests. The final rate of the *FINE* resolution is then evaluated to be equal to $R_{Fine} = 223$ Mbit/s for the whole pixel array. In order to evaluate the rate for a single pixel, it must be taken into account that there are only 64 TDC which switch between the four pixel banks. Therefore, the actual number of pixels (in terms of rate) truly is 64 and the final rate per pixel is then $R_{Fine}/64 = 3.8$ Mbit/s per pixel.

7.4 About FPGA-based TDC

A Time-to-digital converter can be implemented on an FPGA. The design and the performances are different from an ASIC-TDC (like the one on quTAU) [132–134] and various solutions have been investigated recently [130, 131, 135, 136]. For the sake of completeness, a brief description of a possible way to implement a TDC on an FPGA is given.

A TDC can be implemented on an FPGA with one of its basic built-in block: the Full-Adder (FA). The FA computes the sum of two input bits, A and B, with regards to a value C, the carry, which represents the result of a previous sum by another FA block. Hence, the FA outputs the result of the sum and a new carry signal. For a TDC configuration many FAs are used in block-chain and their inputs are specifically set in order to have a sum equal to one and a carry equal to zero. The carry signal is the event signal which asserts from 0 to 1 making every output sums of the chain switch: the carry is propagated through the FAs. Scanning the values of the sum bits, it is possible to find out how long the carry/event propagates through the chain. Since

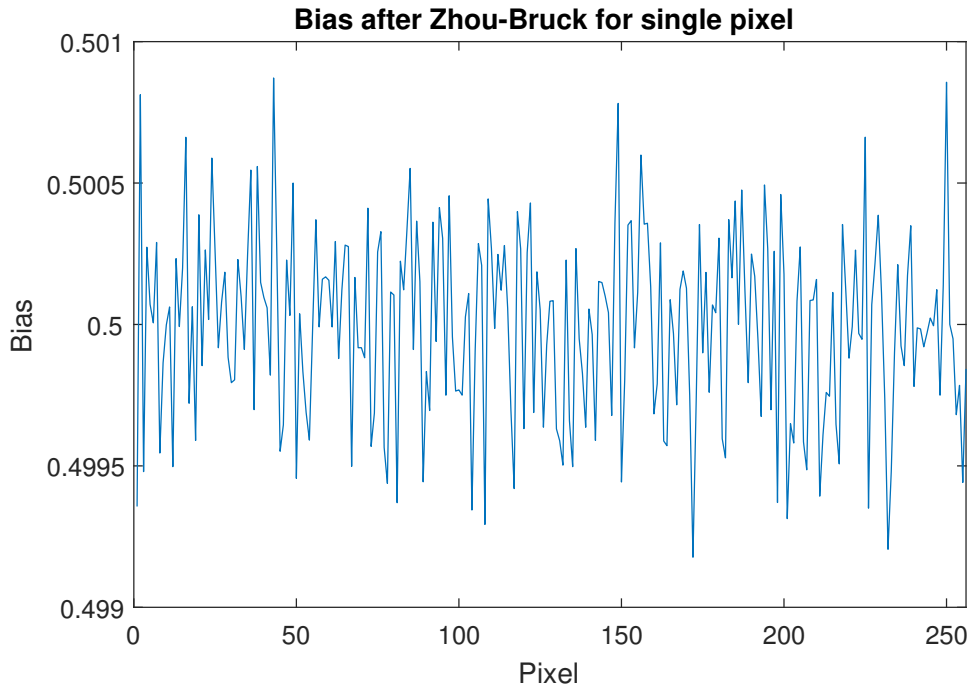

FIGURE 7.5: Bias after the application of Zhou-Bruck method. The graph shows the bias values for every single pixel which is in the order of 10^{-4}. This result clearly shows the validity of Zhou-Bruck method in terms of bias removing.

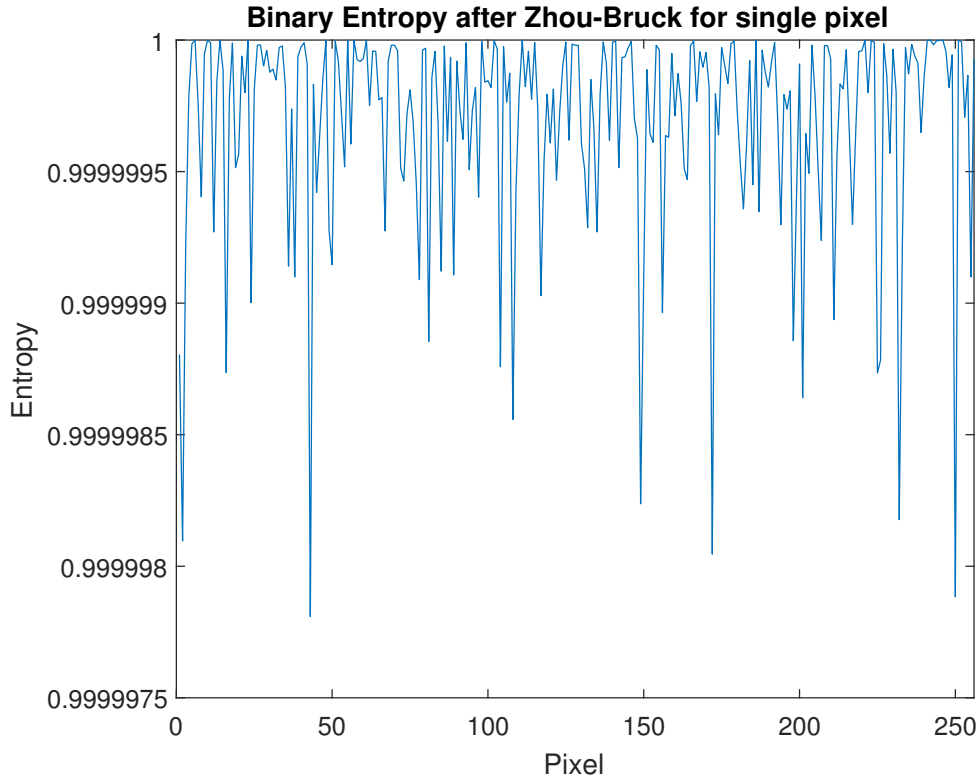

FIGURE 7.6: Binary entropy after the application of Zhou-Bruck method. The graph shows the binary entropy values for every pixel which is quite near to the maximum value.

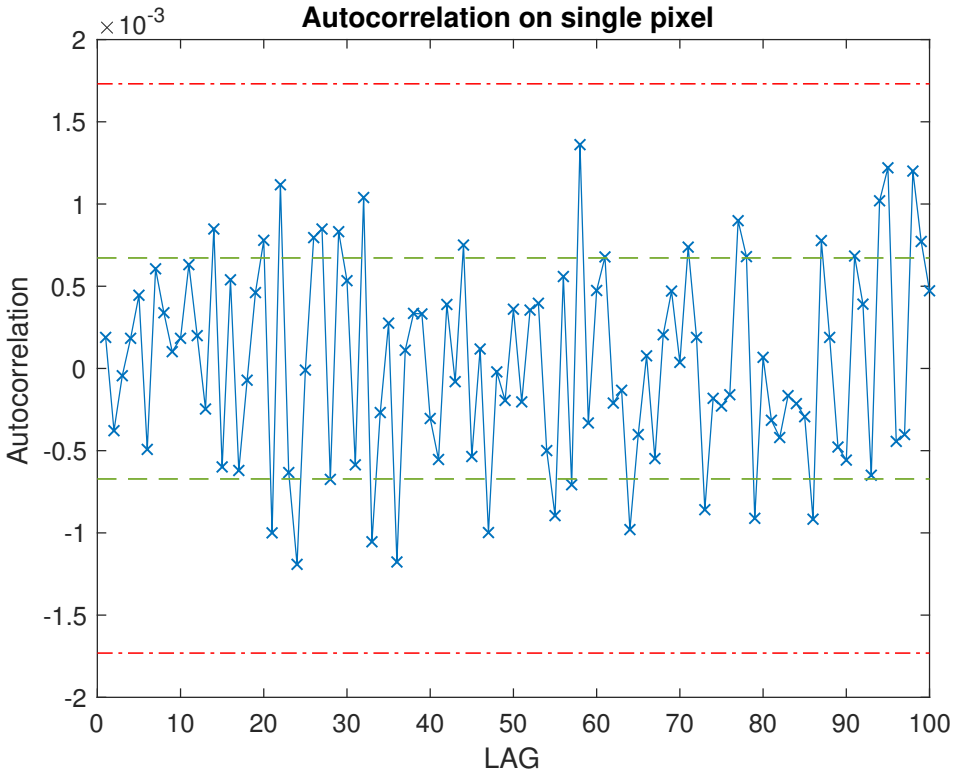

FIGURE 7.7: Autocorrelation on a string after the application of Zhou-Bruk algorithm. The serial correlation re-enters within the limit of acceptance. Green dashed lines represent standard deviation while red dot-dash lines are 99% confidential limits.

every FA has a known propagation delay (in the order of ps), it is possible to estimate the time of the event with very high resolution. Unfortunately, FAs may not have the same propagation delay and they also integrate a fast carry propagation option in order to speed up the propagation of the carry. These features, which instead are well required for standard computation applications, make the TDC not perfectly linear, introducing artifacts as shown in figure 7.4.

7.5 Rates summary

Figure 7.8 shows a comparative bar diagram which recaps the different rates of the analyzed QRNGs. The diagram highlights the relation between generation protocols, number of SPADs and sampling frequency. Clearly, a huge contribution for the total rate comes from the introduction of a TDC and Zhou-Bruck method.

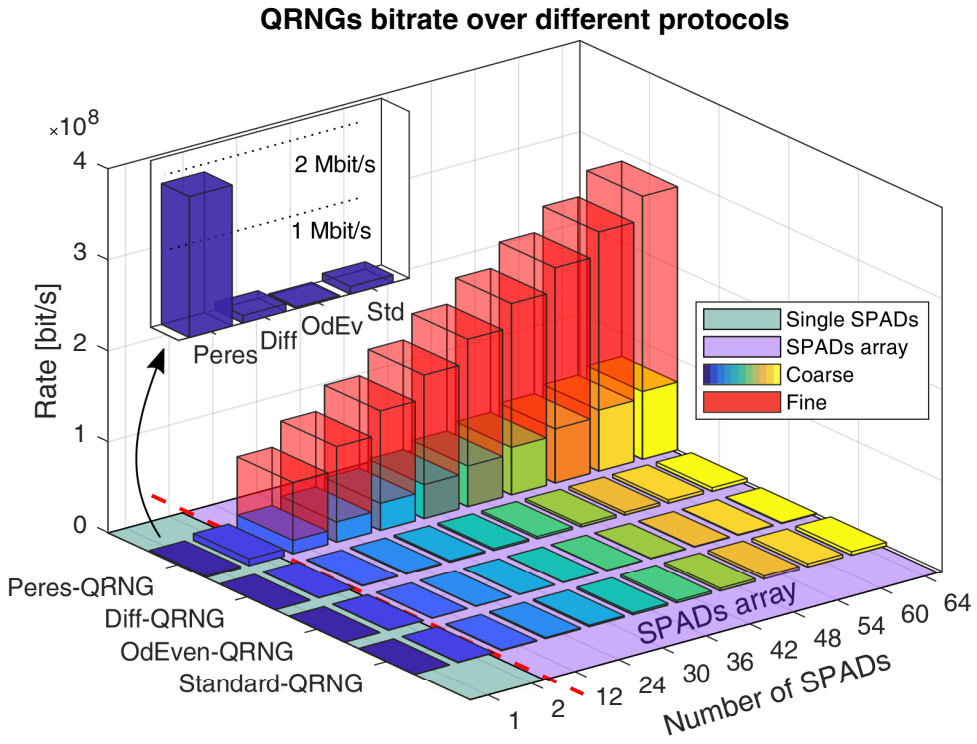

FIGURE 7.8: A comparative bar diagram of the different protocols rates. The diagram shows the bit rates differences among different protocols. It is divided in two areas which represent the use of "discrete" SPADs (like the ones used for Randy) and SPADs array (like LinoSPAD).

Due to pixels cross-correlation, the rate does not increase linearly adding detectors.

Part III

Quantum Key Distribution systems

Chapter 8

Cryptography and Quantum Key Distribution

Nowadays the need for secret data transmissions has significantly increased. Without a doubt, the modern world communication infrastructure completely relies on the concept of *secrecy* and *security*. The disappearance of these two fundamental features could bring to a catastrophic scenario where the telecommunication world stops working.

The terms "secrecy" and "security" convey two different meanings. The first one is the *authentication* which concerns the certainty that the party who wants to transmit a message, also indicated as *plaintext*, is truly talking with the party who has to receive it and not with someone else. The second one is the *encryption* which guarantees that a sent message has an information value exclusively for the designated parties owning a decryption key. A system that can guarantee a perfect authentication and encryption is said to be *Unconditionally Secure*: a potential eavesdropper will get no hint about the message from the system itself but could only randomly guess what the content of the message is. Nevertheless, the realization of such a system is quite a challenge. During its history, mankind has always tried to improve the existing cryptographic paradigms in order to reach the unconditionally secure system. The final step of this journey is the Quantum Key Distribution which protocols can guarantee unconditionally security.

8.1 Introduction and brief history of cryptography

The term cryptography comes from the ancient Greek words $\kappa\rho\upsilon\pi\tau\acute{o}\varsigma$, "hidden" and $\gamma\rho\acute{\alpha}\varphi\epsilon\iota\nu$, "writing" and concerns the manipulation of a message in order to make it unreadable and information-less except for the two parties.

Cryptography is rooted in ancient ages. It was closely tied to military needs, which boosted the invention of different encryption methods during centuries. Two of the most famous examples are *Caesar cipher* and the *Enigma machine*. The first one was named after Julius Caesar who used it for its private correspondence. It was based on a very simple rule: every letter of the plaintext was shifted by n^{th}-positions

in the alphabet. Oddly enough, this cipher was never solved at that time as enemies who stole a plaintext thought it was written in some foreign language.

The second example, the Enigma machine, is well known for its usage by Germans during World War II. Its working principle was quite a cutting-edge technology. Thanks to a electro-mechanical system, the machine could change a letter of the plaintext with another one located n^{th}-position in the alphabet. Unlike Caesar cipher, the shifting value was not the same for every letter of the plaintext but changed letter by letter. The shifting values were determined by specific settings of the machine. These settings were changed every morning and were written on a secret notepad that only German army had. Therefore, each Enigma machine was equipped with a file including the setting for every day of the year. Despite of this well-structured system, an English team led by Alan Turing found the way to determine the every-day setting without having the German notepad. The team worked at the British codebreakers headquarters sited in Bletchley Park where it built the first crypto-analysis computer, known as *The Bomb*, which could decrypt Enigma messages implementing specific frequency analysis algorithm.

During the second half of the 20^{th} century, the need for the exchange of secure-encrypted messages grown up quite fast and it was strictly linked to the evolution of information technology. Therefore, more sophisticated and robust protocols were required in order to guarantee a safety way to transmit information. Nowadays cryptographic systems are based on mathematical procedures and are reasonably appropriate for the existing information infrastructure. They are essential for the survival of the information economy and most of the information companies rely on the concept of encrypted data.

As an example, a brief description of the most famous one is given. The RSA protocol [137], created by Ronald Rivest, Adi Shamir and Leonard Adleman, is based on the public/private key dichotomy. The main idea is to encrypt a message with a public key which belongs to the receiver party, Bob. The public key is known to everybody. When a transmitter party, Alice, wants to send a message to Bob, she can use the public key to encrypt the message through a specific function. Then, she can send the encrypted message to Bob who will use the private key to decrypt the message. Thanks to mathematical models and the usage of prime number factorization, decrypting the message without the private key is quite difficult and requires a huge amount of computational power. As a matter of fact, the security of this protocol, like other modern protocols, relies on computational complexity. However, it is far from the unconditional security concept. Furthermore, the increase of computational power of modern computers and the forthcoming consolidation of the Quantum Computer technology will dreadfully undermine the possibility to use such protocols.

Unconditional security can be guaranteed only by protocols like *Vernam cipher* or *one-time-pad* invented by Gilber Vernam in 1920. Given a plaintext and a key of the same length, it is possible to perform a sum modulo two between every bit in order

to obtain a ciphertext. If the key is used only once and comes from a TRNG, there is no way to decrypt the ciphertext without the key, as demonstrated by Shannon in [138]. This protocol seems to be the holy grail of cryptography but has a practical obstacle that jeopardize its spreading: the two parties must share the key. Needless to say, they cannot share it over the communication channel but only *by hand* in order to be sure that no one else has any kind of information about it. A hand-sharing is not impossible but really inconvenient for the amount of encryption required in moder age. Plus, transmitting a key encrypted with another one not only does not solve the problem but also reduces the overall security of the system.

This is where Quantum Key Distribution finds its place. It is a method to substitute a hand-sharing of a key with the assurance that no one else has any kind of information on it. The key can then be used to encrypt messages with a one-time-pad protocol.

8.2 Quantum key distribution

Quantum Key Distribution, abbreviated as QKD, is a protocol that uses the transmission of qubits to create a secret encryption key which will be shared only by the two parties. The secrecy of the key comes from the qubit itself and from quantum mechanics laws. A formal and basic overview on quantum information and quantum mechanics is given in appendix B. QKD was invented in 1984 by Charles H. Bennett and Gilles Brassard [139] who defined the first quantum cryptographic protocol named after them, the BB84 and later developed most of the quantum cryptography foundations [140–142]. The main idea is that Alice sends a sequence of qubits and Bob performs a measurement on the sequence. The outcome of Bob's measurement is unpredictable to anyone before the measurement itself. If an intruder, known as eavesdropper or Eve, tries to extract some information during the transmission, she will irreparably change the outcome of Bob's measurement. As a matter of fact, thanks to the no-cloning theorem [143, 144] there is no way to exactly clone a qubit. Once Bob has performed the measurement, Alice and Bob evaluate the Quantum Bit Error Rate (QBER). If the QBER is below a specific threshold they can process their data in order to create a key which is completely uncorrelated with any potential information eavesdropped by Eve. This key can then be used to encrypt a plaintext of the same length. QKD requires two different communication channels: the *quantum channel*[1] which is used for qubits transmission and a *classical channel* which has to be an authenticated channel and is used for the exchange of supporting and not sensitive data.

[1]The term "quantum" does not refer to any quantum characteristic but only identifies the channel where the quantum states are transmitted.

Bit/Basis	+	×
0	$\|H\rangle$	$\|D\rangle$
1	$\|V\rangle$	$\|A\rangle$

TABLE 8.1: BB84 bit encoding lookup table

8.2.1 BB84 and other QKD protocols

The BB84 protocol encodes information using four different quantum states from two mutually unbiased bases. The protocol is described with the formalism of polarization encoded qubits which can be easily created with single photons. The two bases can be chosen as the H/V basis (+) and the 45° diagonal D/A polarization basis (×). Therefore, the four states are $|H\rangle$, $|V\rangle$ and $|D\rangle$, $|A\rangle$ and they encode classical information bit according to table 8.1.

The protocol steps are the following:

1. Starting from a fully random sequence of classical bits, Alice prepares a sequence of qubits.

2. Alice sends the sequence encoded as polarized single photons over the quantum channel.

3. Bob performs a measurement on the incoming photons. The measurement cannot discriminate the basis of the incoming qubits and Bob will randomly choose the basis. Then, Bob can determined the orthogonality of the states.

4. Bob communicates to Alice the basis sequence he used to measure the states. Then, they discard the qubits sent in one basis and measured in the other one. As a matter of fact, measuring an H/V basis qubit on the D/A basis will destroy any information about its original state. The resulting qubits sequence is called *sifted key*. The efficiency of this step is equivalent to 50% since the number of quantum states in the H/V basis is the same as the D/A one.

5. Alice and Bob evaluate the QBER of the sifted key by publicly sharing a portion of it.

6. If the QBER is above a certain threshold they abort the protocol; if it is below they will make a procedure of error correction in order to remove every error in their bit strings. This step is known as *information reconciliation* [145].

7. The last step is the *privacy amplification* [146] where Alice and Bob distill a new key completely uncorrelated with the previous one.

If, between step 2 and 3, Eve intercepts the qubit, performs a measurement and sends a new qubit according to the measurement, then she can introduce QBER into the system. For example, Alice sends an $|H\rangle$ state, Eve measures on the D/A basis and sends to Bob a $|D\rangle$ (or $|A\rangle$) state. Then, Bob receives the states and makes a

measurement on the H/V basis (if not, the qubit is discarded during sifting). Clearly, with a 50% probability, Bob can measure the $|D\rangle$ (or $|A\rangle$) state as a $|V\rangle$ introducing de facto QBER[2].

This protocol procedure is different from any classical information one. Actually, the two parties do not transmit an a priori information since the original quantum states sequence is uncorrelated with the final key. They just share a sequence of quantum states in order to get correlated data which can be used to have a shared and secret information. Furthermore, the key length is also unpredictable since it relies on the QBER value. According to [149] where $\mathcal{H}$ is the Shannon binary entropy, Q is the QBER and $I_E(Q)$ is Eve's information (which is equal to $\mathcal{H}(Q)$), the final key rate can be written as

$$r = 1 - \mathcal{H}(Q) - I_E(Q) = 1 - 2\mathcal{H}(Q) \tag{8.1}$$

It is clear that the higher the QBER the higher Eve's information is. As a matter of fact, the QBER represents how much the system was perturbed by Eve and therefore how much information she gained. In order to have a key completely uncorrelated to Eve's information, Alice and Bob need to throw away a portion of information. By simple math, it is possible to find that the rate goes to zero when $Q \simeq 11\%$. In other words, a 11% QBER will not allow the generation of a key fully uncorrelated with Eve's information. The final rate is also influenced by the sifting step in which half of the information is thrown away since Bob will choose the wrong basis half of the times. Furthermore, from a practical point of view, losses and actual key bit rate must be taken into account. In case of polarized single photon, losses can make the key bit rate very marginal and QKD quite impracticable. For these reasons, other QKD protocols and other ways to synthesize the qubit were investigated during the last decades [150].

BB84 variations

Improved QKD protocols can work around the allowed QBER threshold as well as the sifting efficiency. A brief list of some of these protocols is given.

- *Six-states-protocol*. Introduced by Pasquinucci and Nicolas Gisin in 1999 [151], this protocol adds an extra mutually unbiased basis. In case of polarized single photon, the extra basis can be chosen as the L/R which corresponds to the D/A basis rotated by 45° on the equator of the Poincarè sphere. This extra basis makes more difficult for Eve to extract useful information from quantum states. According to [149] the QBER threshold is $\simeq 12.61\%$. Since a single polarized photon is defined over a two-dimensional Hilbert space, the maximum number of mutually unbiased bases is three.

[2]For the sake of clearness, the described Eve's attack is just an example and does not represent the optimal one. Other eavesdropping techniques exist with much higher effectiveness [147, 148].

- *More mutually unbiased bases*. As explained in [152], by increasing the number of mutually unbiased bases of the protocol it is possible to have higher QBER threshold. Nevertheless, the maximum number of mutually unbiased bases allowed belongs to the dimensionality d of the Hilbert space in which the quantum state is defined. A d-dimensional Hilbert space allows a maximum of $d+1$ mutually unbiased bases. This is the reason why in case of a qubit the maximum number of mutually unbiased bases is three. If the dimensionality is higher than two, the quantum state is no longer defined as *qubit* but is called *qudit*. As shown in [152], it is possible to have a 33.36% QBER threshold in case of d=11 and twelve mutually unbiased bases. Practically, even in the "simple" case of d=3, it is quite a challenge to deal with a system capable of synthesizing qudit.

- *Depolarizing*. Another protocol with no official name was proposed by Renner, Gisin and Kraus in 2005 [153]. The protocol introduces a simple change in BB84 or Six-states-protocol. With a probability q Alice introduces some noise in the quantum states sequence. This noise can be described either as a bit-flip or as a depolarization of a quantum states. With an appropriate choice of q, it is shown that the correlation between Alice and Bob is reduced but the correlation between Alice and Eve is reduced even more, which allows a higher QBER lower bound. For BB84 the threshold changes from 11% to 12.6% and for Six-states-protocol from 12.61% to 14.1%.

- *Efficiency-BB84*. An interesting solution was proposed in [154]. In BB84 the probability of choosing the H/V basis is $p_{H/V} = 1/2$ and is equal to the probability of choosing the D/A basis $p_{D/A} = 1/2$, while in the proposed protocol the two probabilities tend to the following: $p_{H/V} \to 1$ and $p_{D/A} \to 0$. The unconditionally security is proven to be guaranteed even with this unbalanced proportion and with the great advantage of a higher sifting efficiency which tends towards 100%.

More on QKD

The QKD secret key rate can be improved by changing the way in which the qubit is synthesized. The standard example of polarized single photon is part of the so called Discrete Variable (DV) QKD. Nevertheless, it is also possible to use Continuous Variable (CV) QKD firstly introduced by T. C. Ralph in [155]. A CV-QKD system offers a higher bit rate in respect to a DV-QKD one since it does not deal with weak coherent pulses but with coherent pulses: the available technology allows to send such-synthesized qubit at a higher rate. Therefore, it does not require single photon detectors but homodyne receivers which, by now, are more cost-effective. Nevertheless, DV-QKD system requires a simpler setting up and less post-processing. As

a matter of fact, within the framework of modern QKD, these two paradigms coexist together since one can be preferred over the other depending on the specific application.

Another example is the Time-Bin encoding proposed in [156]. The qubit is defined in the temporal degree of freedom. Unlike DV-QKD, this synthesization offers a more robust encoding since it is more resistant to depolarization. As a direct consequence, less depolarization brings to a lower QBER and then to a higher key bit rate. However, the setting up of the system is quite demanding since it requires a fine calibration of a Mach–Zehnder interferometer.

Therefore, the realization of a QKD system can be accomplished in many different ways some of which require more efforts than the others. Choosing one realization over the others also depends from the environment where the QKD has to be realized. For instance, on a free-space optical link, the polarization of a light wave is one of the easiest way to synthesize a qubit. Practically, the implementation of a proper optical apparatus can be realized with standard components and gives acceptable performances [157]. At the moment, the polarized-single-photon solution is one of the most preferred within free-space quantum communication. In chapter 10, design techniques for a free-space QKD system are described.

8.2.2 Satellite quantum communication

Exploiting satellite communication environment to spread secret information through quantum communication is more than just a thought. The no-cloning theorem is a fundamental cornerstone of quantum cryptography but necessarily leads to the impossibility of regenerate a qubit during its traveling in order to face channel losses. *Quantum Repeater* [158] offers a solution but is still a delicate and young technology. Hence, a practical answer could come from the so called *Trusted Repeater* [159] and satellites network perfectly fit this idea. This is the reason why, in recent years, *Satellite quantum communication* was part of theoretical works [160–165], mission proposal [166–168] and experimental demonstration [169–177]. In the next chapter a *satellite-to-ground* link test is described.

Chapter 9

Satellite-to-ground QKD

Micius experiment is part of a collaboration between *Quantum Future Group*, *Matera Laser Ranging Observatory* and the *Chinese Academy of Sciences*. The aim of the collaboration is to investigate *Quantum Communication* in a satellite-to-ground link. An important preliminary test was conducted in July 2018.

FIGURE 9.1: An evocative long-exposure picture of the passage of Micius satellite in July 2018.

9.1 MICIUS Satellite

Micius low earth orbit (LEO) satellite, launched in 2016, is part of the *Quantum Experiments at Space Scale* (QUESS) international research project in quantum physics field and is operated by the CAS. The name Micius comes from the name of a fifth century B.C. Chinese scientist who discovered that light travels in straight lines. Micius satellite has already been part of the successful experiment described in [178]. The optical setup of the satellite comprehends 8 different 850 nm for the implementation of BB84 protocol with decoy [179], a CW 532 nm laser used for system tracking and synchronization. The aim of the preliminary test was to verify the tracking and

synchronization as well as the capability to discriminate between different polarizations. Therefore, the 850 nm beams was not used at single-photon level.

9.2 Matera Laser Ranging Observatory

Matera Laser Ranging Observatory (MLRO) is an observatory of the *Agenzia Spaziale Italiana* (ASI) and is part of the *Centro di Geodesia Spaziale* (CGS) Giuseppe Colombo founded in Matera, Italy, in 1983. Its main objective is conducting fine space geodetic measurements [180]. After attesting the feasibility of free space Quantum Communication over long distance [181–183], the Quantum Future group collaborated with MLRO in order to experimentally study Quantum Communication in space [98, 169, 171, 173, 174, 177, 184].

MLRO telescope is a 1,5 m telescope and is part of a seven mirrors path, the *Coudé*. Every mirror is named according to its path position starting from the primary mirror M_1 to the last one M_7. A schematic of the Coudé is given in figure 9.2. The Coudé system applies a unitary transformation on the polarization state of light which depends on the telescope position. Due to a mechanical uncoupling between mirrors M_3 and M_4 and mirrors M_6 and M_7, the unitary transformation is telescope-position-dependent. Hence, if a specific telescope moving is scheduled in a certain time window, the unitary transformation is time-variant. In order to couple the Coudé output light with the reference optical axis used in the optical table, a dedicated compensation is required. Plus, if required during a telescope scheduled moving, the compensation has to be time-variant too. Moreover, in Micius experiment there was an extra compensation to take into account: optical axis difference between Micius satellite and MLRO. Compensation method will be explained in detail in next section 9.3.

9.3 Angle compensation and Thorlabs device controller software

The compensation was made using two half-wave waveplates (HWPs), one quarter-wave waveplate (QWP) and one liquid crystal retarder (LCR). The optical elements were set in the following order:

$$\text{QWP} - \text{HWP}_1 - \text{LCR} - \text{HWP}_2$$

We defined

- $\mathcal{U}_{QH}$ as the unitary transformation applied by QWP and HWP_1
- $\mathcal{U}_{LCR}$ as the unitary transformation applied by LCR
- $\mathcal{U}_{Micius}$ as the unitary transformation applied by HWP_2

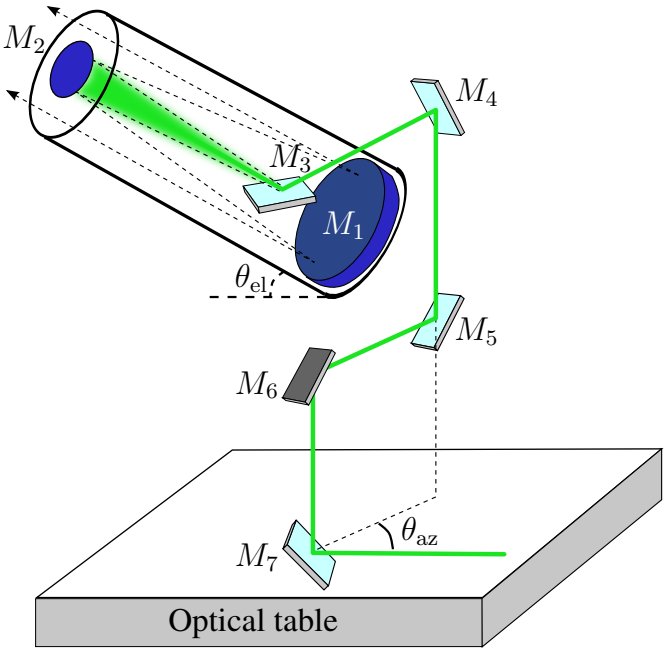

FIGURE 9.2: A schematic view of the Coudé path. Picture from [171].

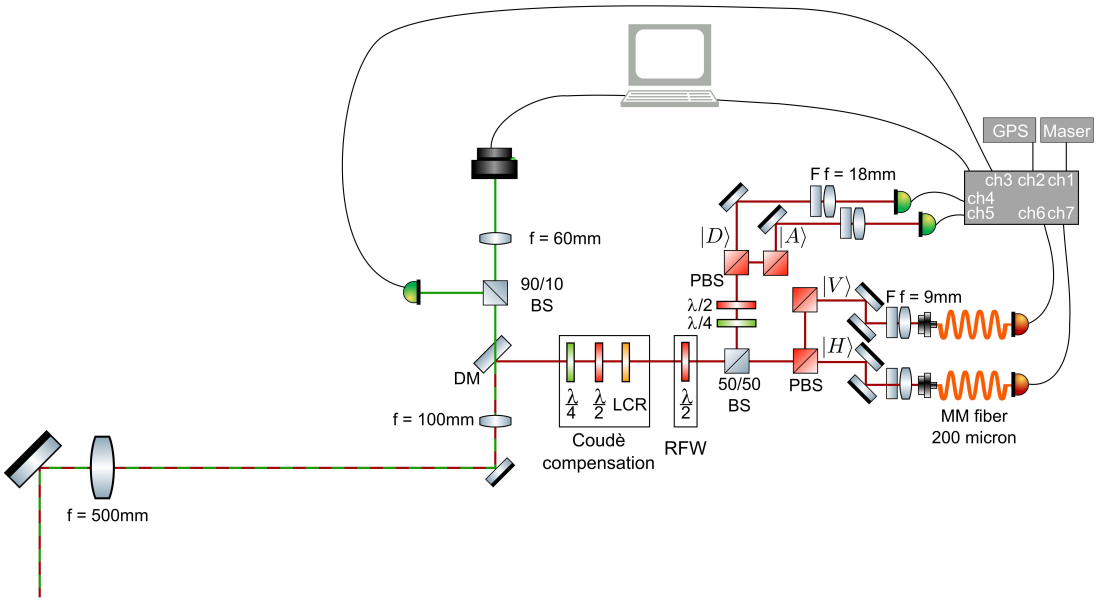

FIGURE 9.3: A schematic view of the optical setup at MLRO.

- $\mathcal{U}_{Coudé}$ as the unitary transformation applied by the *Coudé* path

Given that $|H'\rangle$ and $|V'\rangle$ are the horizontal and vertical polarization angles of the telescope optical axis, the HWP and QWP allowed to rotate any $|H'\rangle$ or $|V'\rangle$ polarized light at the telescope input to $|H\rangle$ and $|V\rangle$, the horizontal and vertical polarization angles of the optical axis used in the optical table. Therefore, given that

$$\mathcal{U}_{Coudé}\,|H'\rangle = |\psi\rangle \tag{9.1}$$
$$\mathcal{U}_{Coudé}\,|V'\rangle = \left|\psi^{\perp}\right\rangle \tag{9.2}$$

the followings were valid

$$\mathcal{U}_{QH}\,|\psi\rangle = |H\rangle \tag{9.3}$$
$$\mathcal{U}_{QH}\left|\psi^{\perp}\right\rangle = |V\rangle \tag{9.4}$$

Anyhow, in case a superposition of $|H'\rangle$ and $|V'\rangle$, said $|D'\rangle = |H'\rangle + e^{i\gamma'}\,|V'\rangle$, the rotation applied by the HWP and the QWP could not remove the phase value on vertical component outputting $|H\rangle + e^{i\gamma}\,|V\rangle$. The LCR served the purpose to remove the $e^{i\gamma}$ term. Hence, given that

$$\mathcal{U}_{Coudé}\,|D'\rangle = |\psi\rangle + e^{i\gamma'}\left|\psi^{\perp}\right\rangle \tag{9.5}$$

the followings were valid

$$\mathcal{U}_{QH}(|\psi\rangle + e^{i\gamma'}\left|\psi^{\perp}\right\rangle) = |H\rangle + e^{i\gamma}\,|V\rangle \tag{9.6}$$
$$\mathcal{U}_{LCR}(|H\rangle + e^{i\gamma}\,|V\rangle) = |H\rangle + |V\rangle \tag{9.7}$$

As a matter of fact, these three elements allowed the rotation from a $|H'\rangle$, $|V'\rangle$, $|D'\rangle$, $|A'\rangle$ polarization to $|H\rangle$, $|V\rangle$, $|D\rangle$ and $|A\rangle$.

The last element of the compensation chain is another HWP. The goal was to rotate the optical reference of Micius satellite, said $|H^*\rangle$ and $|V^*\rangle$, to the MLRO telescope one. As shown in figure 9.4, the angle difference ϕ could be compensate by a HWP set at proper angle $\alpha = \phi/2$. Since this HWP had nothing to do with the *Coudé* path compensation, it was possible to consider the HWP to be at the input of the telescope in order to simplify mathematical expressions. To summarize, the HWP allowed the followings

$$\mathcal{U}_{Micius}\,|H^*\rangle = |H'\rangle \tag{9.8}$$
$$\mathcal{U}_{Micius}\,|V^*\rangle = |V'\rangle \tag{9.9}$$
$$\mathcal{U}_{Micius}\,|D^*\rangle = |D'\rangle \tag{9.10}$$
$$\mathcal{U}_{Micius}\,|A^*\rangle = |A'\rangle \tag{9.11}$$

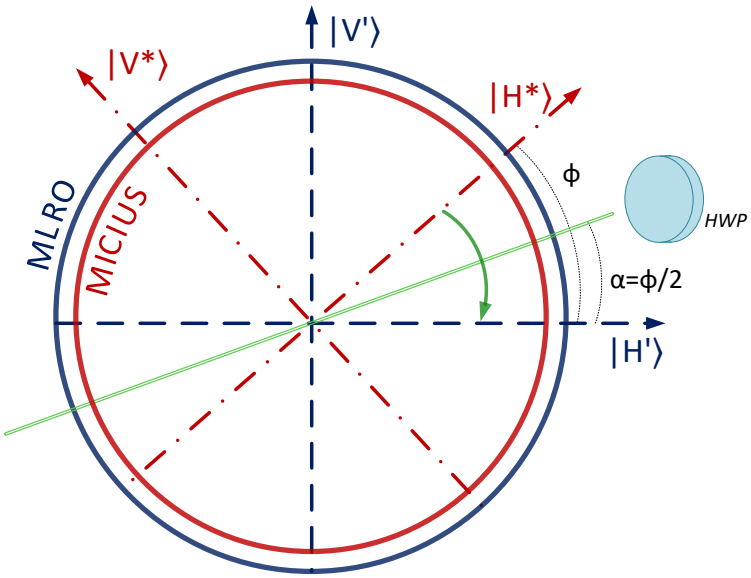

FIGURE 9.4: Reference frame angle rotation between Micius satellite and MLRO telescope.

Due to the time-variant characteristic of the whole apparatus, all the four elements had to be synchronized and continually updated. Therefore, the unitary transformations had to be written as $\mathcal{U}_{Coudé}(t_u)$, $\mathcal{U}_{QH}(t_u)$, $\mathcal{U}_{LCR}(t_u)$ and $\mathcal{U}_{Micius}(t_u)$ where t_u is a discrete time value.

Like the setup of [178], the system was synchronized to the UTC time and updated just once per second. Anyhow, the performances of these devices did not allow a considerably higher moving speed. The synchronization process required to define every value of every unitary transformation between the first transmission instant t_0 and the last one t_{end}.0

9.3.1 Thorlabs motorized rotation stage and controller

Three different *K-Cube DC Servo Motor Controller KDC101* (figure 9.5) were set up. Each one controlled one KPRM1E motorized rotation stage (figure 9.6) where the waveplate was mounted. According to the selected angle, the K-Cube could mechanically rotate the KPRM1E in the proper position. The K-Cube were controlled by a specific USB driver which libraries could be used within a programmable environment according to the reference documentation of the Thorlabs APT Motion Control Software [185].

9.3.2 Thorlabs liquid crystal retarder

The LCR is a Thorlabs *Compensated Full-Wave Liquid Crystal Retarder LCC1413B* which can be controlled by a voltage regulator: the Thorlabs *Liquid Crystal Controller LCC25* (figure 9.7). This controller communicated with the computer through a standard

FIGURE 9.5: Thorlabs KDC101 K-Cube Servo DC Motor COntroller. Picture by Thorlabs.

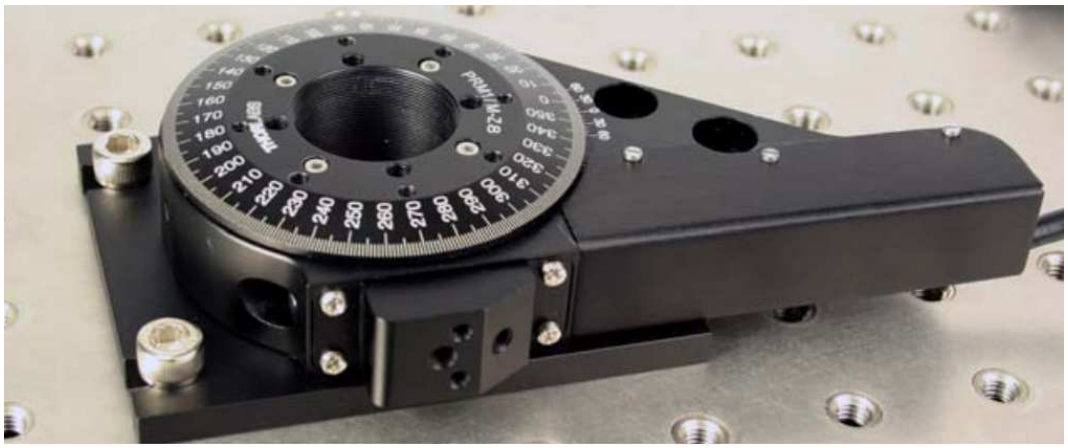

FIGURE 9.6: Thorlabs PRM1Z8 motorized stage. Picture by Thorlabs.

USB-serial port and could receive specific instructions according to the product official documentation [186].

FIGURE 9.7: Thorlabs Liquid Crystal Controller LCC25. Picture by Thorlabs.

The *Thorlabs Device Controller* software (figure 9.8) allows the control of all the four Thorlabs devices. Like Quntroller, the software was developed in `QT` environment in `C++` language. The structure is divided in three parts: *main*, *servo motor controller*, *liquid crystal controller*. A schematic view of the software structure is visible in figure 9.9. The description of the software follows:

- The main part is responsible for the initialization of software variables and graphic elements as well as for the UTC time reading and updating. Due to the slow compensation required, a precision within one second was quite sufficient. Therefore, the usage of Windows UTC time was chosen over the UTC time of the Trimble Thunderbolt receiver used on Quntroller software for UTC locking (see subsections 9.4 and 9.5 for more details). Quntroller software was running on the same machine and the possibility to linked the two softwares was taken into account: the UTC time could be sent by Quntroller to the Thorlabs Device Controller through a TCP-socket. Nevertheless, this procedure was not cost-effective and could undermine the stability of Quntroller. Furthermore, it would have made the Thorlabs Device Controller dependent on Quntroller. Another solution was to read the UTC data from the Trimble serial port. However, a serial port allows only one connection per time and a workaround to setup a multiple connection to one serial port was not cost-effective either.

  Windows UTC time was kept up-to-date by an automatic synchronization with a time server through the *Simple Network Time Protocol* (SNTP), a modified version of the *Network Time Protocol* (NTP) [187]. The two available servers were `'time-windows.com'` and `'time.nist.gov'`. By a swift comparison with the Trimble UTC time, we chose to use the first one since it offered a little bit more

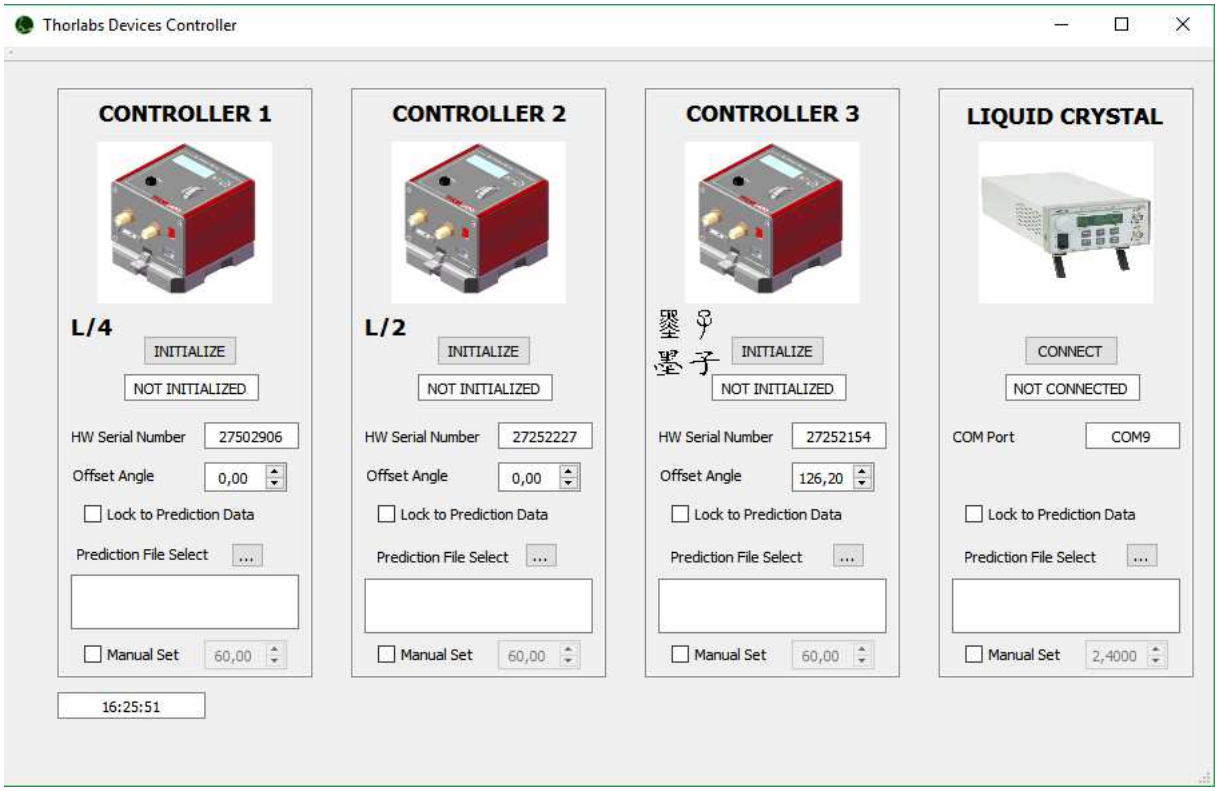

FIGURE 9.8: A view of the main window of the Thorlabs device controller software.

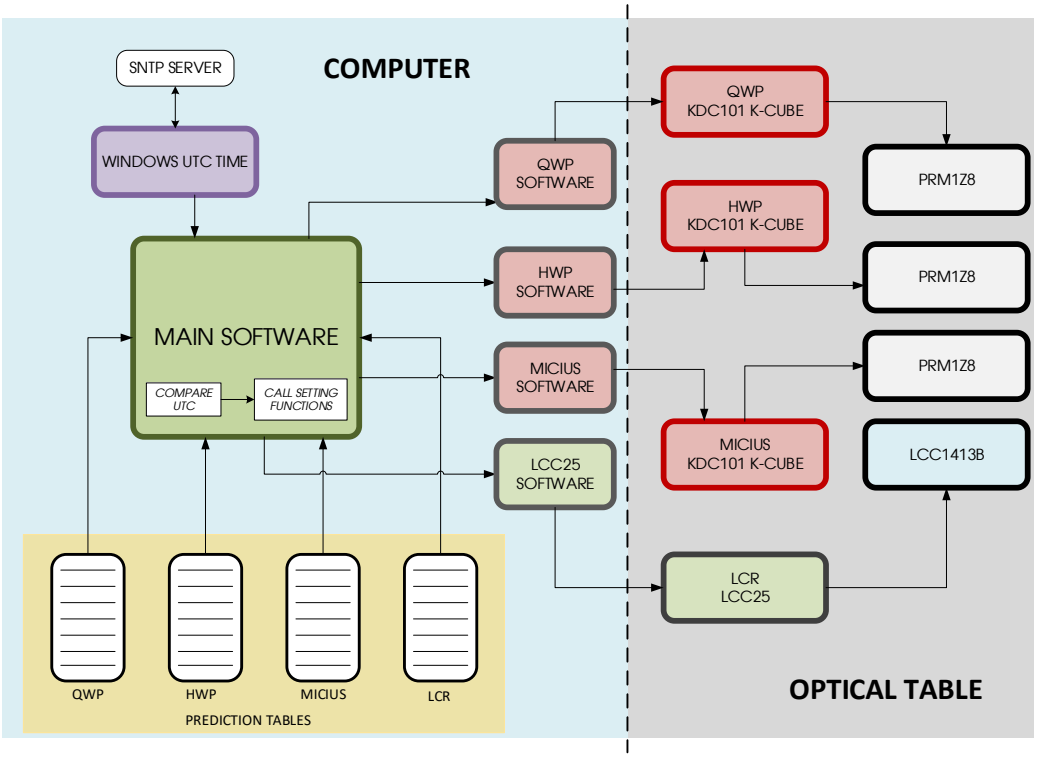

FIGURE 9.9: Structure of the Thorlabs device controller software.

precision. Using a stand QTimer variable, the software checked any changes in the UTC time every 10 ms in order to make the check-time negligible with respect to the updating time scale. When the actual UTC time changed, the software checked whether the new time corresponded to one of the UTC time of the *Prediction Tables* (see next point). If so, it called functions in order to set the devices to proper value.

- The second one describes the functions required to manage the Servo Motor Controller. As visible in figure 9.8, it was possible to initialize the the Servo Motor. A different serial number could be inserted depending on the specific Servo Motor used. By checking the *Manual Set* check box it was possible to select a custom angle. This functionality was added to allow a preliminary check on the working of the device. Moreover, an offset angle could be added to every set angle in order to compensate the offset position of the waveplate. A synchronized rotation was possible by the *Lock to Prediction Data* check box and *Prediction File Select* tool button. By uploading a .csv format with a table of specific angles associated with a specific UTC time, the software could check whether the actual UTC time corresponded to one of the UTC time in the table and, if matched, set the Servo Motor to the corresponding angle value. In order to avoid any possible mistake during the table-file selection and upload (manually done by the user), the path of the uploaded table-file was written in a text-box right below the *Prediction File Select* tool button.

- The third part is responsible for the LCR control. Through a dedicated push-button it was possible to initialize the serial connection with the LCC25 according to the selected COM port. Like the Servo Motor Controller, a *Lock to Prediction Data* check box and a *Prediction File Select* tool button allowed the synchronization with the UTC time according to the uploaded table. The only difference was that the LCR worked with voltage values instead of angle ones.

9.4 GPS receiver

For the UTC time synchronization the *Thunderbolt E-GPS Disciplined Clock* by *Trimble* company was used (figure 9.10). The device worked with a dedicated GPS antenna and with different time and position performances. The receiver offers a Oven-Controlled Crystal Oscillator (OCXO). Unlike normal Voltage Controlled Oscillator (VCO) and Temperature-Compensated Crystal Oscillator (TCXO), an OCXO offers better performances since it can keep the crystal oscillator to a fixed temperature allowing high time-precision thanks the reduction in possible time drifts.

Standardly, the device offered a Pulse Per Second (PPS) output and a 10 MHz output, both on BNC connectors. These signals are time-locked to the OCXO.

According to the documentation (available at [188]), the device could be controlled through a serial connection and the Trimble dedicated software. The software

FIGURE 9.10: Thunderbolt GPS receiver. Picture by Trimble.

allowed to set the parameters and monitor the device performances. Several tests revealed optimal parameters choice in order to maximize temporal performances over position ones. Sure enough, the PPS timing precision was in the order of tenths of nanoseconds which absolutely fit the required precision[1].

9.5 PPS and quTAU synchronization

Synchronizing the data acquisition of the quTAU was one of the most important part of the entire project. The transmission from Micius satellite was lock to the UTC time. Hence, it was necessary to correctly save the detected data on the quTAU. In order to do so, the PPS signal was connected to one of the inputs of the quTAU allowing to have a time correlation over the data. Nevertheless, the signal is analog and did not bring any information on the tag of the PPS itself. That information was coded as a message and sent through the serial port of the receiver. In order to distinguish which PPS is which, a serial connection between Quntroller and the Thunderbolt receiver had to be established as well. The connection was set according to Thunderbolt documentation [188] and required a dedicated programming since the Thunderbolt receiver sent data and messages using a custom protocol. The goal was to synchronize the read UTC time from the serial port to an acquired data from

[1] Actually, such device with such overkilling precision was bought to be used also for the experiment described in chapter 10. The experiment synchronization required a quite higher timing precision and is currently under investigation.

the quTAU which possibly had the relative PPS event within it. The lock-to-PPS procedure is outlined in figure 9.11 and was designed as follow:

- The quTAU acquisition step was set to 1 second. As described in section 3.2, Quntroller used a QTimer class to count to 1 second. The QTimer was set to be a precise timer by calling `setTimerType(Qt::PreciseTimer)` which, according to [74], it could guarantee an average precision of +1 ms. The QTimer could not last less than 1 second and could not have a negative precision error.

- A serial connection was established with the Thunderbolt GPS receiver. The signal *ReadyRead*, emitted every time there were new data to read, triggered a function that read the data and checked whether there was a new UTC time. If so, it updated a `QDateTime` variable according to the new UTC information.

- A pressing event on the *Start Acquisition* push-button happened between two different UTC times updates, e.g. *UTC 0* and *UTC 1* as in figure 9.11. When the push-button was pressed, the software waited until the next UTC updating (*UTC 1*).

- After *UTC 1*, the software waited for 100 ms. Like the Acquisition Step timer, the waiting process was handled with a QTimer set to be a precise one. Then, the software cleared all the previous data collected by the quTAU[2] and began a standard acquisition routine by starting a 1 second QTimer. The 100 ms waiting was introduced in order to prevent the PPS to be too near to an acquisition call and reduce any misalignment possibility even more. This procedure could be set by checking the *Lock to PPS* check-box. Without checking it, the software started a standard acquisition.

- As clearly shown in schematic 9.11, the first acquisition ended after *UTC 2* updating and the data were saved in a file identified with the *UTC 2* time. With respect to every possible system delays which could include cable lengths and software idles or queuing and with respect to *Acquisition Timer* possible time drift, the PPS of UTC 2 could be detected by the quTAU right after the UTC 2 instant. Therefore, the PPS tag was not likely to happen at the beginning or at the end of 1-second acquisition, which could cause a misalignment as the PPS fell in the wrong acquisition.

- Micius passage lasted 10 minutes. Considering an overestimated 50 ms delay for the updating of the UTC time (new UTC time, send through serial connection, software data reading and variable updating), the extra 100 ms timer, the propagation delay of the PPS BNC cable (approx. 20 meters long with a delay of hundreds of nanoseconds) and a clock drift of 1 ms for every second, the acquisition would not foul up before 14 minutes. Even in the unlikely situation

[2]This was possible by simply calling the function to acquire the data from the quTAU buffer but without saving them in any variable.

with a missing PPS from an acquisition file, it could be possible to reconstruct the right alignment with a simple post processing workaround. In fact, there was no way the first PPS could not be in the first acquisition and actually, that was the only necessary information in order to align transmitted data.

9.6 Results

The polarization of the CW laser at 850 nm was evaluated from the measurement on the four channels. In figure 9.12 the QBER values for the H/V and D/A bases as function of UTC time are shown. The figure also shows the degree of polarization assessed by using the linear and diagonal components of the polarization vector on the Bloch sphere. The degree of polarization is larger than 0.9 (1 s average), that is compatible with an observation of linearly polarized light, throughout the whole passage, correctly compensating the 7 mirrors of the Coudé system and the mirror for beam redirection on the optical table. The overall value of degree of polarization, without the correction for non-physical values, i.e. larger than one[3], is equal to 0.9630 +/- 0.0397.

The plots show promising values of distinguishability and contrast of the four different polarizations. The three gray areas identify the instants when the offset angle of Micius HWP was manually changed. The size of the area is due to the time needed to change the angle value, i.e. the mechanical delay of the Thorlabs PRM1Z8 device. After dedicated verifications it was established that the systems (acquisition, synchronization and rotation) worked properly and that the substandard QBER values are due to a suboptimal evaluation of the angle between Micius and MLRO. Surely, future rounds of the experiment will consider a better calibration of the optical system in order to achieve a small percentage value QBER as in [178].

To summarize, these results show the feasibility of QKD between Micius satellite and MLRO. Thanks to the optical characteristics of MLRO telescope, such QKD promises to reach a significant secret key rate in the order of Mbit/s. Hence, future developments will aim to obtain such achievement in the framework of a multi-part and international academic collaboration.

[3]Values higher than one are not possible and are due to multiple detections on different polarizations.

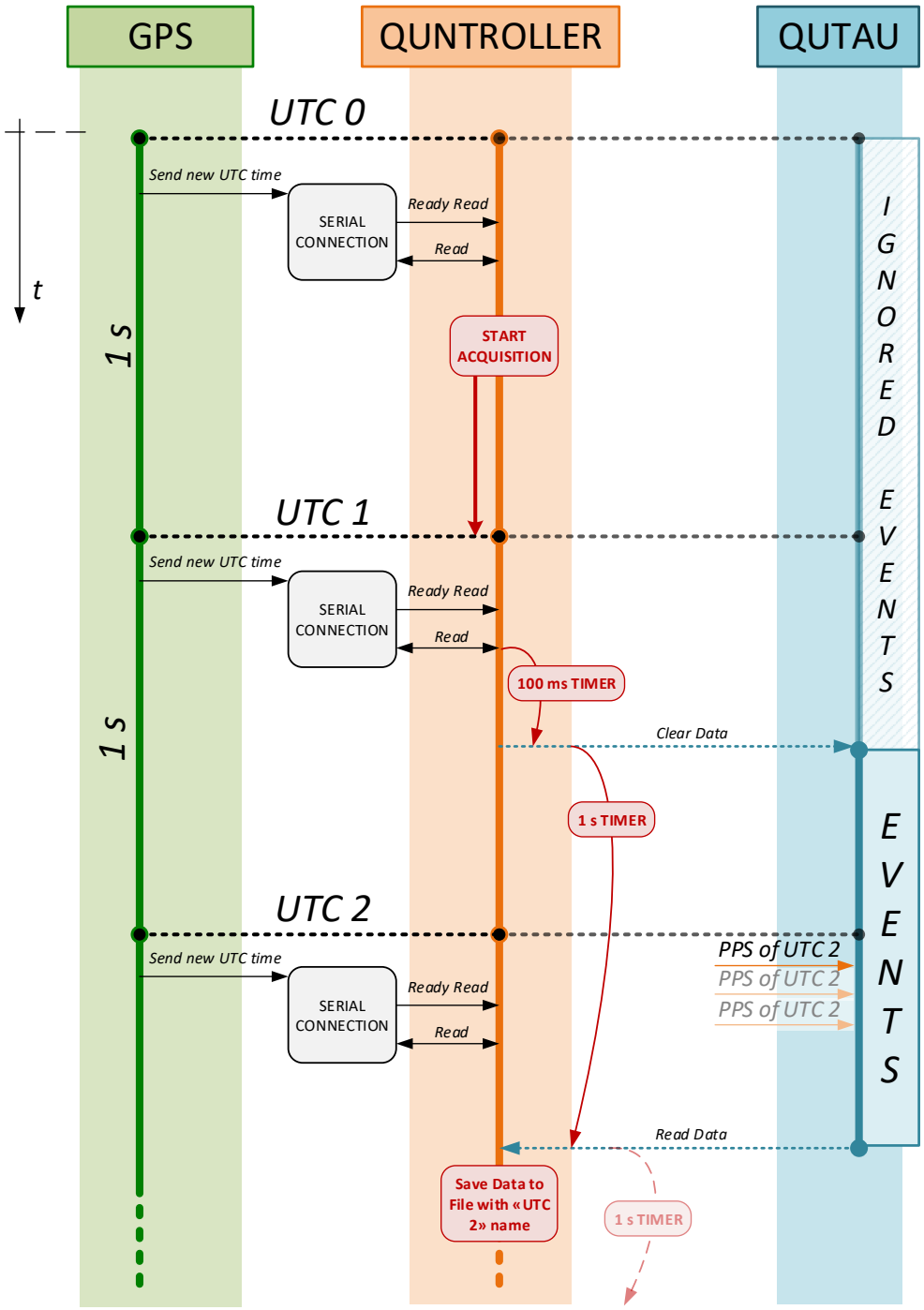

FIGURE 9.11: A schematic view of the procedure used to lock the quTAU acquired data to the UTC time. Time is flowing from top to bottom. Plot is not on scale.

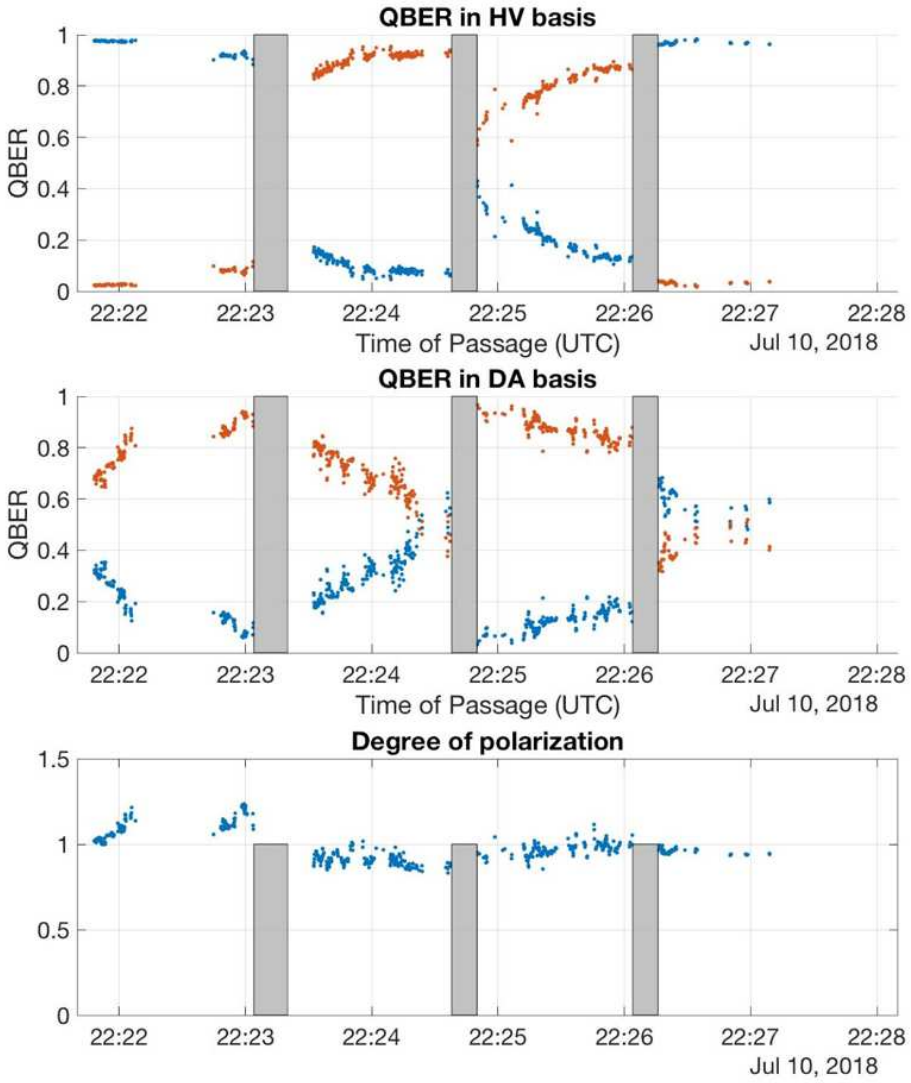

FIGURE 9.12: QBER evaluation and degree of polarization during Micius passage as function of the UTC time. Each point is an average of 100 ms. The two different colors identify the two orthogonal states of a basis. The Micius HWP was rotated by an angle θ(t)+θ_0, where θ(t) changed each second according to the satellite position, and θ_0 is an additional offset changed in order to select the detection output. The gray areas correspond to the time needed to change the offset θ_0 and observe the different bases. The first (UTC<22:23) and third zones (22:25<UTC<22:26) correspond to $\theta_0 = 0°$, the second zone (22:23<UTC<22:25) corresponds to $\theta_0 = 22.5°$, while the fourth zone (UTC>22:26) corresponds to $\theta_0 = 45°$.

Chapter 10

Ground-to-ground free-space QKD

An experimental realization of a free-space QKD link was set up as a collaboration between *ASI* and *QFuture* group. The main objective was to build a system capable of supporting a full QKD procedure. The experiment was structured in many layers and a full description of it goes beyond the intention of this work. Furthermore, at the moment the experiment is still ongoing.

In the following sections a brief overview of the setup is given along with a focus on the electrical design of the transmitter.

10.1 Overview of the setup

The system was designed to handle all the practical aspects required for a QKD. Therefore, the setup included:

- GPS receivers for the synchronization between the two parts
- two beacon lasers for tracking
- Fast steering mirrors (FSMs) and Position sensitive devices (PSDs)
- dedicated FPGAs, TDC and computer for the generation, detection and control of the whole system
- a QRNG to produce the string which determined the sequence of states to send
- distributed feedback (DFB) laser for the generation of qubit
- electro-optical intensity and phase modulator for decoy implementation and discrimination of the three states
- one transmitter and one receiver telescope
- four Superconducting nanowire single-photon detector (SNSPD)
- other optical elements

The whole schematic of the experiment is shown in figure 10.1. The setup implemented the qubit synthesization as polarized single photon as well as its transmission and reception. The chosen QKD protocol was a recent modification of the BB84

which only requires three states instead of four [189]. This protocol was proven to be secure even in the finite key regime [190]. Moreover, decoy states [179, 191, 192] protocol was also implemented.

10.1.1 Transmitter setup

In the red square of figure 10.1 the QKD source is highlighted and the following components are visible:

- the DFB laser, attenuated to single-photon level, which was used to generate the qubit. The used model was the EM655 produced by Gooch & Housego [193].
- the Amplitude modulator (azure rectangle) which implemented the decoy states through the use of a intensity electro-optical modulator. The modulator was used to vary the mean number of photons per pulse μ between $\mu = 0.1$ and $\mu = 0.5$. The modulator was controlled according to the decoy statistic by an electrical impulse. In a 10 ns slot there always was one modulation impulse. Even when no modulation was required on the incoming qubit, the modulation impulse was performed in a qubit-free area of the time slot. This allowed to keep the working point of the modulator stable and prevent possible deviations. The used model was the LN81S-FC-Zero-Chirp, 10 GHz by Thorlabs [194].
- the Polarization modulator (violet rectangle) which allowed to change the polarization of the incoming qubit outputting the three required states $|H\rangle$, $|V\rangle$ and $|D\rangle$. Faraday Mirror (FM) allowed the use of just two voltage levels to set the phase electro-optical modulator. The phase modulator was custom made with a PM fiber angled to 45°. Therefore, the input modulator polarization was $|D\rangle$. Hence, the three polarizations could be created with the following rule: when voltage equal to zero, the qubit polarization did not change and remained $|D\rangle$; when voltage equal to $V_\pi/2$ only during the *there* path, the qubit polarization changed to $|H\rangle$; when voltage equal to V_π only during the *back* path, the qubit polarization changed to $|V\rangle$. The used model was the MPZ-LN-20 by iXblue [195].

In figure 10.7 a photograph of the implemented QKD source is visible.

10.2 Shortening the electrical pulse

The system was designed for a Zynq-7000 chip on a ZedBoard Evaluation Board. The single-photon repetition rate was set to 100 MHz with a pulse duration of 360 ps. The advantage in producing short pulse (electrical and then optical) is that it allows to discriminate the photon from the background noise with more precision,

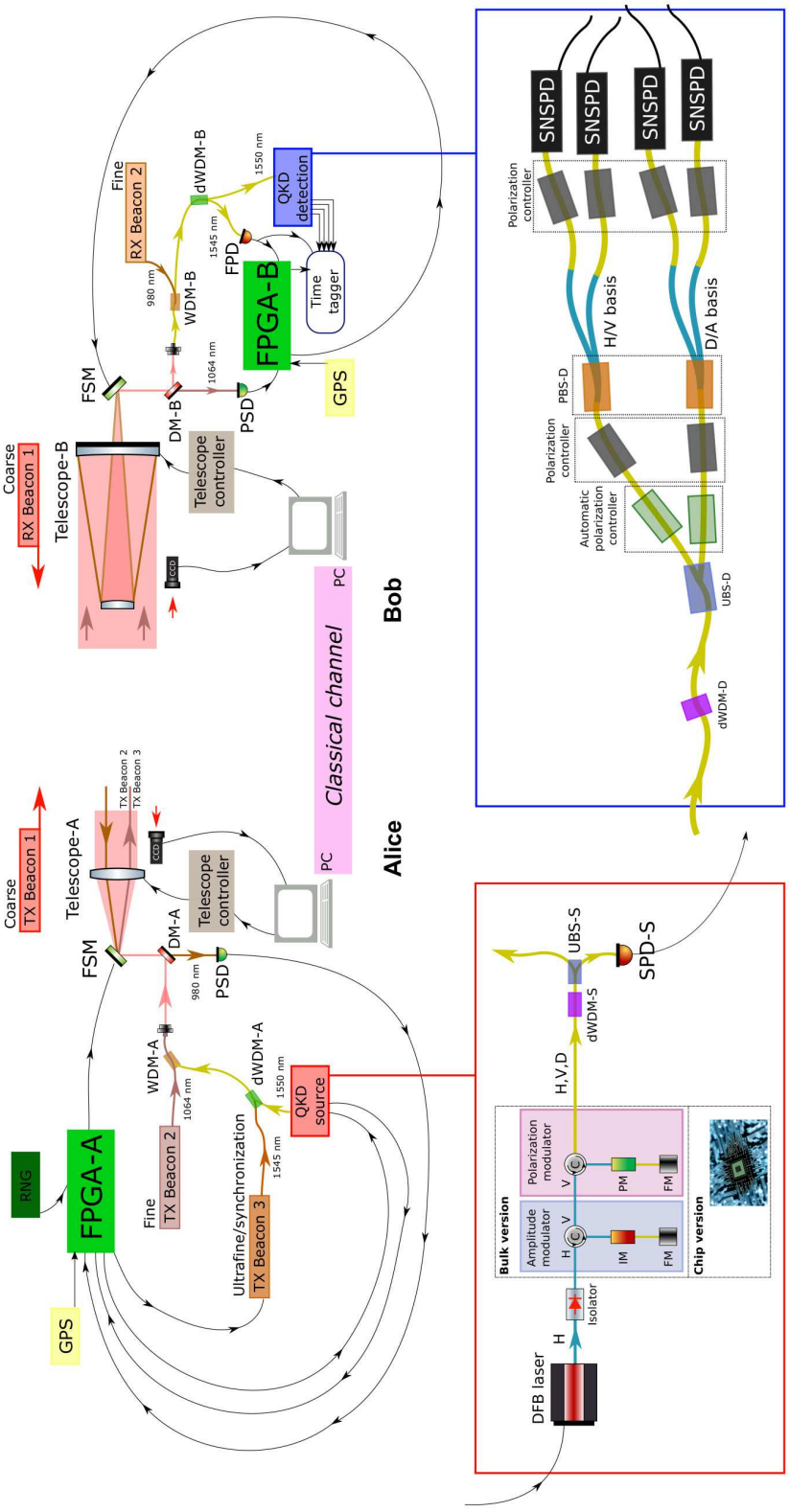

FIGURE 10.1: A schematic view of the experiment setup.

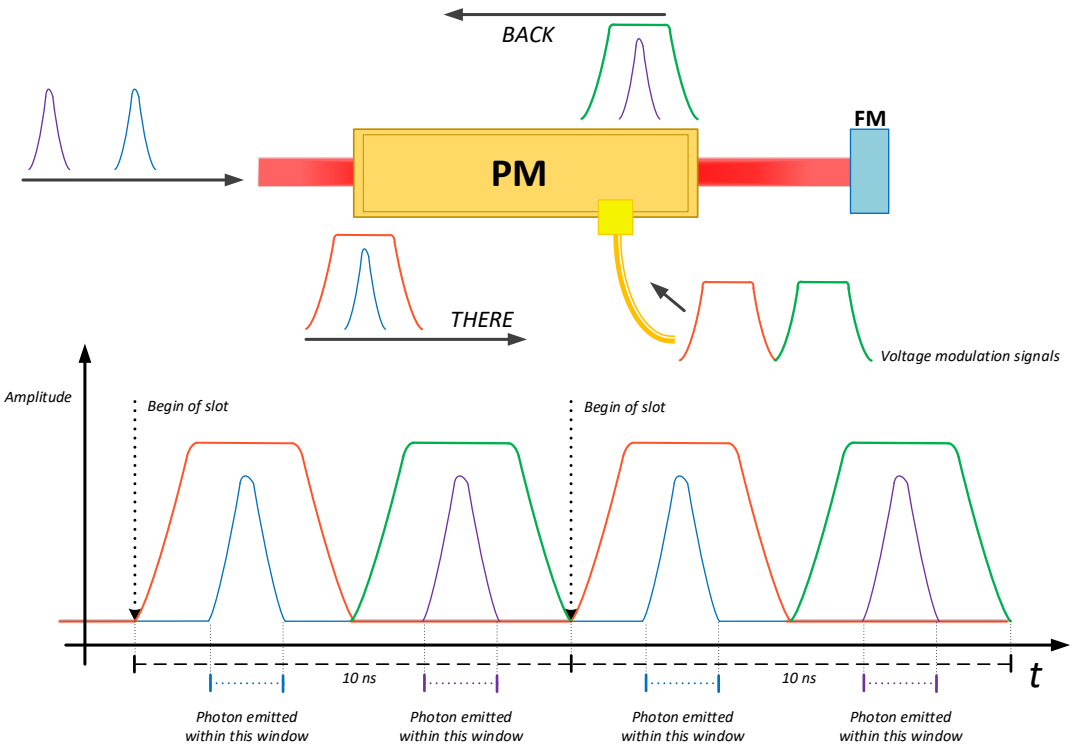

FIGURE 10.2: A schematic view of the polarization discrimination on the transmitter.

i.e. the shorter the pulse the lower the QBER the higher the bit rate. However, the ZedBoard can have a maximum clock frequency of 800 MHz (1.25 ns period). Hence, a logic hardware was designed to produce sub-ns pulses from the default 100 MHz (10 ns period) clock.

By using the clocking wizard IP core [55], two clock 100 MHz clock signals were produced. The first one was sent to a *custom logic* block which acted as a delay line, i.e. it simply output the same clock signal with no changes. The custom logic was just a custom IP core which explicitly instantiated the CARRY4 primitive [196]. This passage was the only way to deliberately introduce a delay because a VHDL code that copies the input to the output is interpreted as an unnecessary hardware and is removed during synthesis by Vivado. The inputs ports of the CARRY4 primitive were properly set in order to introduce no changes on the clock signal. Due to the path delay and to the CARRY4 intrinsic delay, the signal was output with a certain delay. The output signal was meant to be the primary 100 MHz signal and had a phase ϕ_1.

A secondary 100 MHz clock was produced with a different phase ϕ_2. The value of ϕ_2 was chosen so that the phase relation $\Delta_\phi = |\phi_1 - \phi_2|$ was equal to $180° - \delta_{pulse}$ where δ_{pulse} is the value of the desired pulse. Therefore,

$$\delta_{pulse} = 360° \times \frac{360ps}{10ns} = \frac{360°}{28.57} = 12.96° \tag{10.1}$$

This value represented the time which the two signals are both asserted in. Then,

an AND port was used to create such a pulse. Shortening the phase relation even more did not produce any pulse since the internal electrical pulse could not reach the required threshold amplitude to trigger the digital output component. A schematic view of the block design is visible in figure 10.3. Beyond the aforementioned QKD advantages, this design solution allows to reduce power consumptions since it does not require a $\frac{1}{360ps} = 2.778$ GHz with a power save of about 90%. Therefore, this technique is suitable for any application where power consumption is critical and could be essential for a satellite QKD transmitter. It can also be applied to other boards as well.

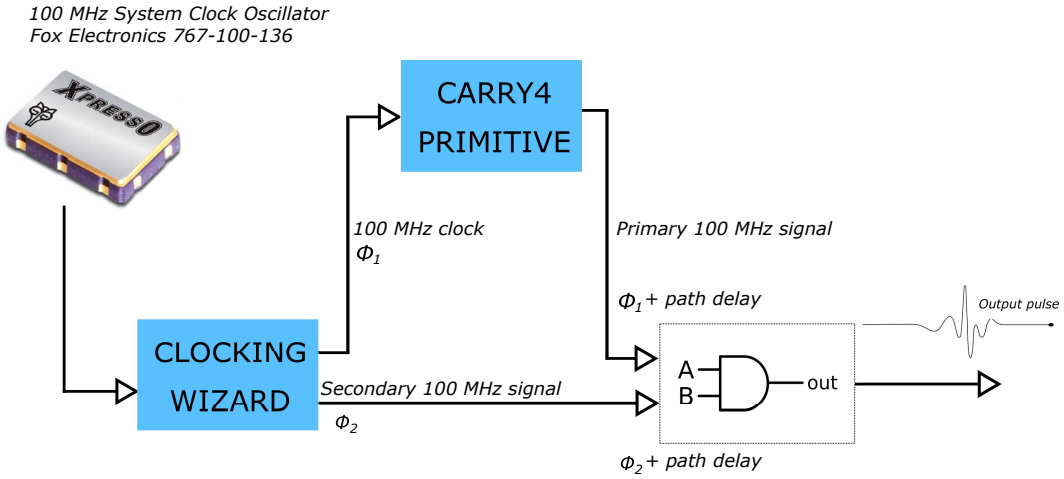

FIGURE 10.3: A schematic view of the logic blocks for the generation of the pulse.

10.2.1 Fixing the phase relation

The exact value of phase relation depends either on the phase value on the clocking wizard or on the slice placement of the implementation process. Of course, placing BELs in different slices affects the signals delay and the actual phase relation. The initial phase value was set to $\phi_2 = 15°$ on the clocking wizard. Then, the location of the primary signal register and of the LUT was set. Such BELs implemented the AND operation in order to sharpen the pulse to the desired value and brought the phase relation to 12°. The path of the secondary signal was already fixed by the location of the clocking wizard.

10.2.2 External Routing and amplification

Different solutions for the external routing were investigated. The produced pulse was routed to an FMC output pin since it can guarantee a high analog bandwidth suitable for the $\frac{0.35}{360ps}$ =0.97236 GHz pulse bandwidth. The signal was routed to the *FMC_CLK1P SMA* output[1] of the FMC XM105 Debug Card [197] (visible in figure

[1]Different routing solutions could not guarantee the required analog bandwidth. For example, routing the signal to PMOD ports (either JA-JB or JC-JD) produced a pulse duration of $\simeq$ 1.2 ns with a 2-300 mV amplitude.

10.4). The constraints parameters were set to *SLEW=FAST* and *DRIVE=24 mA*. As shown in figure 10.5 the final pulse amplitude was sufficient to drive an external RF amplifier [198]. The signal was amplified to a proper amplitude to pulse the DFB laser. Furthermore, a Bias-Tee device [199] was used to fine set the offset of the signal. After an appropriate calibration, a test was conducted[2] and the width of the optical pulse was evaluated to be $\simeq$ 150 ps as shown in figure 10.6.

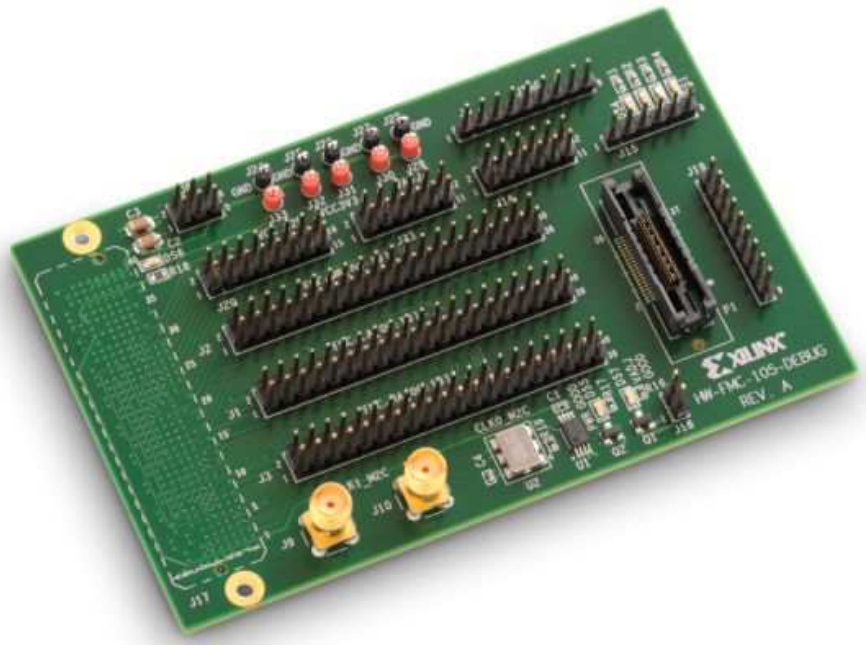

FIGURE 10.4: The Xilinx FMC mezzanine XM105 debug card. Picture by Xilinx.

10.3 Signal for electro-optical modulators

Beside the generation of the electrical pulse for the DFB laser triggering, the FPGA was also designed to generate the required electrical signals to set the two electro-optical modulators. As shown in figure 10.2 for the phase modulator, the electrical signals that control the modulator had to be of a proper shape and had to set the modulator at the right moment. The shape had to be as flat as possible during the passage of the light pulse to guarantee very low polarization fluctuations. Furthermore, the setup required two different modulations, one during the *there* travel and one during the *back* travel. Therefore, two different modulation signals with a 5 ns delay between each other were designed. These two signals were produced using the same technique described above, i.e. the shape was set thanks to CARRY4 primitives. Furthermore, extra CARRY4 primitives were added to produce the 5 ns delay and to fine center the signals with the optical pulse. So basically, these CARRY4 primitives were placed on specific slice that allowed the signal to have the desired

[2]The detection setup was composed by *ID281 Superconducting Nanowire* single-photon detectors [200] and the *ID900 Time Controller* [201] both by ID Quantique.

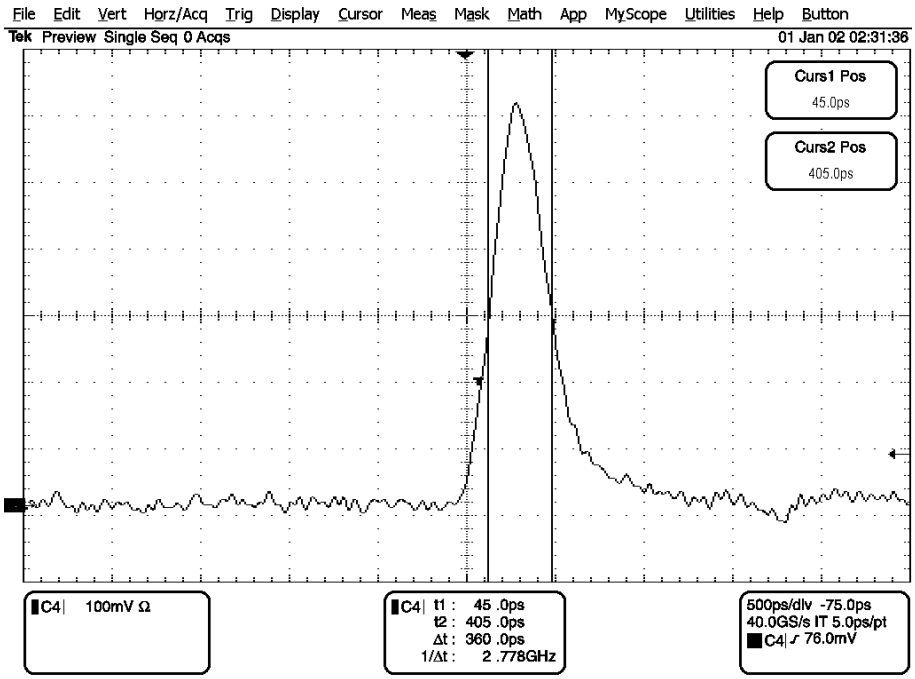

FIGURE 10.5: The produced electrical pulse. FWHM at 360 ps. Amplitude equal to 600 mV (oscilloscope input impedance set to 50 Ohm).

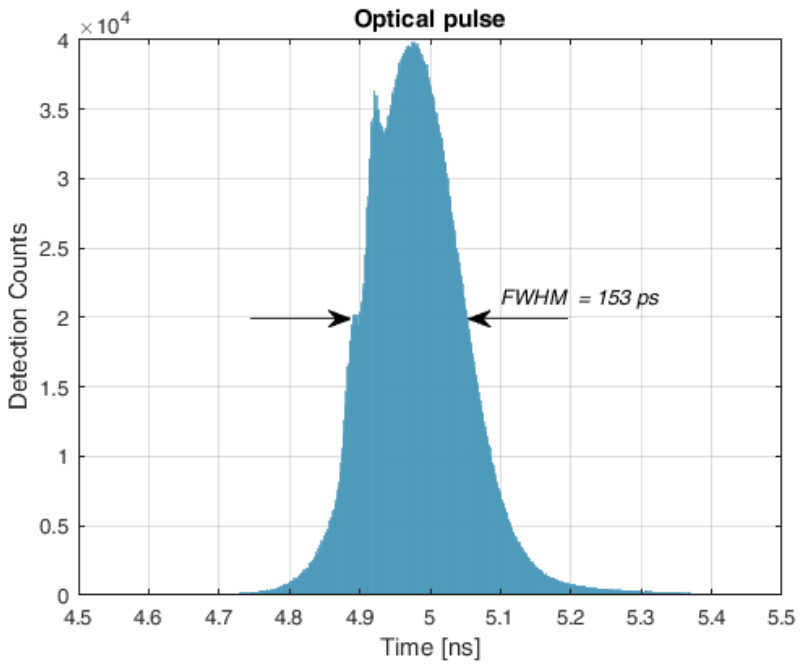

FIGURE 10.6: The produced optical pulse. FWHM at 153 ps. The plot shows that the detections follow the electrical pulse shape and most of the intensity is within an interval of 200 ps. The unexpected small peak at 4.92 ns and the irregular slope at 4.9 ns are probably due to a suboptimal calibration of one of the downstream elements of the setup (Bias-Tee, modulator, fiber, time tagger,..).

delay. This procedure was conducted manually with multiple trials and also compensated the optical delay of the setup. The average time resolution was of 100 ps that is the average delay between a slice and another. Before outputting to the board physical pin, the two signals were merged together with an AND port. The obtained polarization contrast was evaluated to be $\simeq 14$ dB. The signals for the intensity modulation were designed in a very similar way.

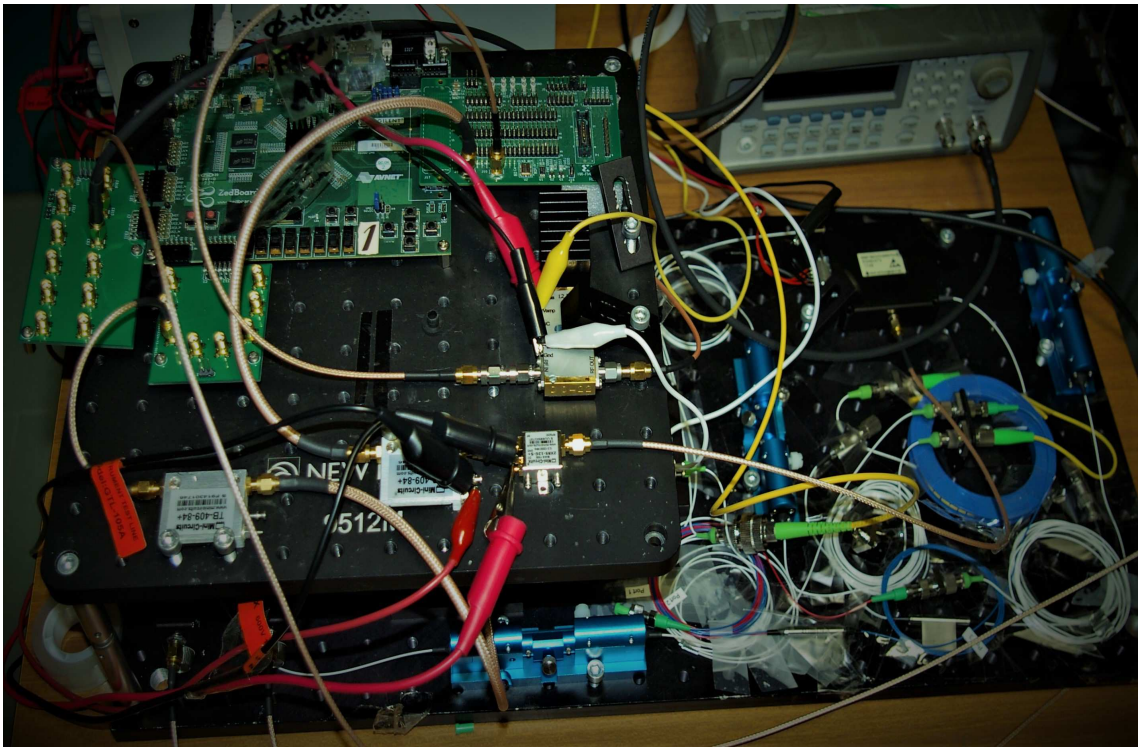

FIGURE 10.7: A photograph of the implemented QKD source setup.

Conclusion

Designing of high performances systems is undoubtedly a fundamental step in the process of investigation in modern scientific fields. It not only gives the necessary technological substructures for the realization of scientific experiments but it also motivates the research for new technological solutions that fit in with the specifications.

This thesis work described the realization of hardware and software systems for applications of Quantum Random Number Generation and Quantum Key Distribution. Along with the details of the systems designed on a dedicated FPGA-CPU board, TDC device and computer, this work described the QRNG and QKD experiments and the obtained results.

Two QRNG systems were presented: *Randy* and *LinoSPAD*. These two systems are both FPGA-based but they differ in the number of single-photon detectors and in the time resolution used for sampling the photon detection.

The first QRNG system, Randy, comprehends a light source attenuated to single-photon level, an FPGA-CPU board (ZedBoard) and a computer for system management. The design was structured as a three-layers system: the FPGA represents the lowest level which interfaces with the physical setup; the board-CPU represents the middle level and implements a first level of parameters setting and system management; the computer represents the highest level allowing a full control over the system from a dedicated GUI. Randy system allows to select over different generation protocols and also implemented a new generation procedure based on Peres algorithm. This procedure maximizes the output generation bit rate directly handling with the sampled string. With a photon count rate of 200 kcounts/s and a sampling frequency of 100 MHz, the final generation rate is equivalent to 1.8 MBit/s. Moreover, a real time generation routine were developed to output timed random numbers and to properly set electro-optical elements of an experimental setup for fundamental test of quantum mechanics. Such experiment realized the Wheeler's delayed-choice Gedankenexperiment extended to space and was conducted at the Matera Laser Ranging Observatory.

The second QRNG system, LinoSPAD, integrates an FPGA chip with an array of CMOS-SPADs. This device was produced by the AQUA lab at Delft University & EPFL and features 256 CMOS-SPAD pixels arranged in four linear array of 64 pixel each. It also features an FPGA TDC offering an overall time precision of $\sim$17.9 ps. In the wake of the developed technique used for Randy, a dedicated approach, also based on the Zhou-Bruk algorithm, was defined to remove the device non-idealities

and thus to generate random numbers. This technique allows to reach a final generation rate equivalent to 300 Mbit/s.

QKD systems were developed in collaboration with the Italian Space Agency and investigated the feasibility of satellite-to-ground and ground-to-ground free-space QKD.

In the first experiment, in which the Chinese Academy of Sciences collaborated as well, an optical link was set up between the Chinese satellite Micius and the telescope at the Matera Laser Ranging Observatory. A transmission test successfully confirmed the QKD feasibility between the two parts. A dedicated software was designed to keep the optical axis alignment between the satellite and the telescope. The software controlled electro-optomechanical and electro-optical components in order to implement a time-variant compensation of the beam angular changes through the optical path. Moreover, a software for the data management of the whole acquisition was developed. The software drove a TDC device, the quTAU, which received all the significant signals from the setup. The software, developed with multi-thread functions, stored data without any failure or losses risk. The acquisition was also synchronized to the UTC time to have the right correlation with the sent data which were locked to the UTC time too.

The second experiment realized a free-space QKD link over a distance of tenths of kilometers. The experiment required a dedicated realization of various parts. This work presented the specific techniques used on the transmitter FPGA design which controlled the qubit laser and the electro-optical phase and intensity modulators. Through the setting of the signals phase and of the electrical path delay, which required a proper logic blocks placement, the system was able to produced a 350 ps electrical pulse generated by combining two 100 MHz signals. The design also took care of the overall delays which were set to compensate the optical path delay through the optical setup.

In the framework of Quantum Information applications, these results can indeed offer a significant contribution facilitating the realization of quantum experiments and thus hastening the spreading of these new technologies in a information-dependent world.

Appendix A

Distortion and linearity on quTAU device: a behavioral analysis

As explained in chapter 3, the quTAU device has a specific bandwidth limits. However, the events coming from a setup could exceed these limits. Therefore, it was necessary to know what behavior was under stress conditions and whether or not the tagging feature spoiled the actual events order and proportion. Using four different signal generators fed into four different channels of the quTAU, seven different rate tests were made varying the channels event rates. The results are visible in tables A.1 and A.2. The "Nominal" columns show the values set on the signal generators while the "Measured" columns show the measured counts. Counts were evaluated by counting the number of tags in the acquisition and without using the counting feature. Results show that the quTAU did maintain the right proportion among the channels without preferring one channel over the others. As shown in table A.2, the percentages of detected events over the expected ones differ in a value lower than 1% over the four channels. Therefore, under stress condition the quTAU certainly loses a portion of events but stays consistent over the channels.

Nominal Rate	Meas. Rate	CH1 Nom.	CH1 Meas.	CH2 Nom.	CH2 Meas.	CH3 Nom.	CH3 Meas.	CH4 Nom.	CH4 Meas.
2.2 M	2.1988 M	100 k	99.947 k	100 k	99.947 k	1 M	999.47 k	1 M	999.47 k
3 M	2.9778 M	500 k	496.3 k	500 k	496.3 k	1 M	992.6 k	1 M	992.6 k
4 M	3.7197 M	1 M	929.92 k	1 M	929.92 k	1 M	929.92 k	1 M	929.92 k
6 M	4.4945 M	2 M	1.4981 M	2 M	1.4981 M	1 M	749.14 k	1 M	749.14 k
2.002 M	2.0003 M	1 k	999.2423	1 k	999.2423	1 M	999.16 M	1 M	999.16 M
4.002 M	3.6472 M	1 k	911.4882	1 k	911.4882	2 M	1.8227 M	2 M	1.8227 M
6.002 M	4.9337 M	1 k	818.4006	1 k	818.4006	3 M	2.466 M	3 M	2.466 M

TABLE A.1: Result of tests. The unit measure is [counts]. As standard convention, "M" symbol stands for 10^6 while "k" symbol stands for 10^3. The measured count is not the same throughout the measurement. For example, the measured count in the 6 Mcounts/s case is quite different from the count measured in the 6.002 Mcounts/s case. This means that in case of count rates that really exceed the quTAU limits, the measured count rate will not be stable over time but could vary.

Moreover, the possibility of a non-linear response was tested. It was necessary to verify whether the quTAU dropped events at the same instant for all the used

Nominal Rate	Ratio η Meas./Nom.	η CH1	η CH2	η CH3	η CH4
2.2 M	99.94 %	99.947 %	99.947 %	99.947 %	99.94 %
3 M	99.26 %	99.26 %	99.26 %	99.26 %	99.26 %
4 M	92.99 %	92.99 %	92.99 %	92.99 %	92.99 %
6 M	74.9 %	74.9 %	74.9 %	74.91 %	74.91 %
2.002 M	99.91 %	99.92 %	99.92 %	99.91 %	99.91 %
4.002 M	91.13 %	91.14 %	91.14 %	91.13 %	91.13 %
6.002 M	82.2 %	81.84 %	81.84 %	82.2 %	82.2 %

TABLE A.2: Ratio computed from the previous table.

channels. The figure A.1 shows the tag differences plot for a portion of the 6.002 Mcounts/s acquisition. The tag differences represent the numerical difference between two consecutive tags. Hence, a high count rate channel will have a small average tag difference value (the higher the frequency the smaller the time period); on the contrary, a low count rate channel will have a high average tag difference value. The figure clarifies the *saturation* behavior of the quTAU: at one instant the device almost stops detecting events on all the channels and this brings to higher difference in values. These values are fairly similar for the four channels and thus they do not depend on the channel events rate. This means that, under stress condition, the quTAU slows down the acquisition for the same amount of time on all the channels.

These results showed that the quTAU can be used without any risks of non-linear response in case of high counts rate.

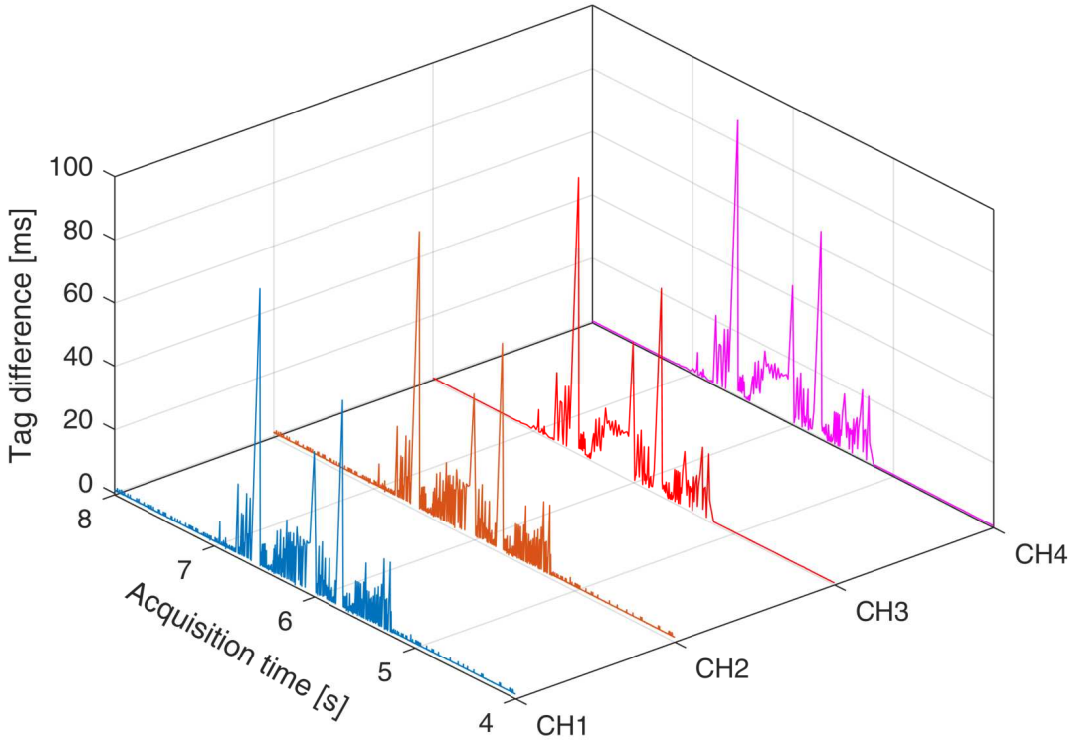

FIGURE A.1: Tag differences of the acquisition on four different channels. Events rate on CH1 and CH2 is equivalent to 1 kevents/s while on CH3 and CH4 is equivalent to 3 Mevents/s. This difference is noticeable by a different height of the base lines: a low base line implies a small time distance between consecutive events while a high base line implies a longer one.

Appendix B

Basic elements of quantum information and quantum mechanics

This appendix gives a brief description of the main concepts of quantum information and quantum mechanics [202, 203]. Quantum information is about the threating of information from a quantum point of view. It is based on quantum mechanics laws and was born as an evolution of the classical information theory. Where classical information used *bits* to describe information, quantum information uses the *qubit*, which stands for quantum bit [204].

The qubit is defined as a two-level quantum system described by a two-dimensional Hilbert space. Using two normalized and mutually orthogonal quantum states, it is possible to map the values 0 and 1 of a classical bit and write them in a vector form:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{B.1}$$

These two quantum states form a *computational basis* of this space. Thus, a qubit can be represented by the superposition of the two states $|0\rangle$ and $|1\rangle$. The general form of the states superposition is described by

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{B.2}$$

where α,$\beta \in \mathbb{C}$, with $|\alpha^2| + |\beta^2|$=1, and then it holds $\langle\psi\,|\,\psi\rangle$=1. In other words, the generic state $|\psi\rangle$ is composed by a certain part of $|0\rangle$ and a certain part of $|1\rangle$. Hence, the term $|\alpha^2|$ corresponds to the probability of measuring the state $|\psi\rangle$ as $|0\rangle$ while $|\beta^2|$ it as $|1\rangle$.

By choosing real and positive values for α and β, the generic state of a qubit can be written using spherical coordinates:

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle \tag{B.3}$$

where θ and ϕ are angles of the three-dimensional Cartesian space, known as *Bloch*

sphere (figure B.1), which allows a generic state to be represented as a unit vector lying on such sphere. The value of θ represents the angle between the state and the z axis and it holds $0 \leq \theta \leq \pi$. The value of ϕ represents the angle between the state and the x axis and it holds $0 \leq \phi \leq 2\pi$.

The representation of the qubit as a linear combination of the computational basis $\{|0\rangle, |1\rangle\}$ is not necessarily the only possible one. As a matter of facts, the state of a qubit in a two-dimensional Hilbert space can be represented as a linear combination of any two orthonormal state vectors, for instance, by the *conjugate bases*:

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{B.4}$$

$$\frac{|0\rangle + i\,|1\rangle}{\sqrt{2}}, \quad \frac{|0\rangle - i\,|1\rangle}{\sqrt{2}} \tag{B.5}$$

The pair of vectors forming each of the conjugate bases corresponds to the Pauli eigenvectors operators σ_z, σ_x and σ_y. They form a set of mutually orthogonal vectors over the Bloch sphere.

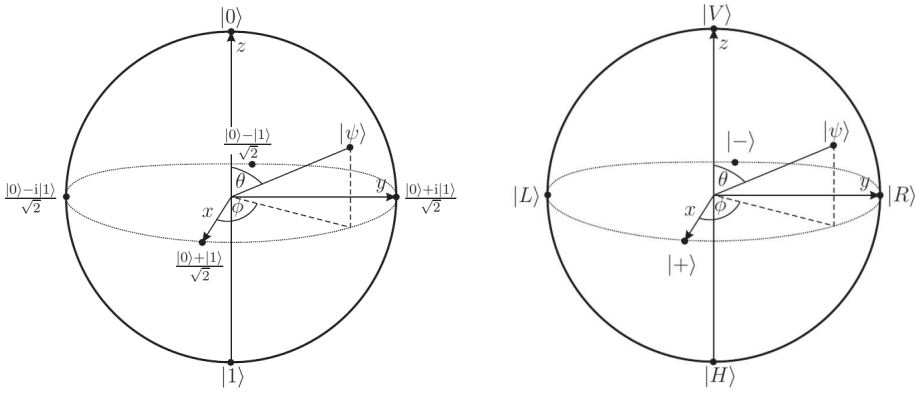

FIGURE B.1: The Bloch sphere (left) and the Poincarè sphere (right).

Now it is possible to use any two-level quantum system in order to create a physical qubit. For example, the spin of an electron, the spin of a photon or two electronic levels of an atom can be considered for qubit realization. Due to its nature, the polarization of a light wave corresponds to the easiest task [205]. Hence, many quantum communication apparatuses synthesizes the qubit as a polarized single photon associating the $|0\rangle$ and $|1\rangle$ states to the horizontal and vertical states, respectively. From a formal point of view, the new basis is still a computational one:

$$|0\rangle \rightarrow |H\rangle, \quad |1\rangle \rightarrow |V\rangle \tag{B.6}$$

The diagonal polarization basis[1] (D/A) and the left/right polarization basis (L/R) form the new conjugate bases and are related to the H/V one as follow:

[1]The symbols "D" and "A" are sometimes substituted by "+" and "-".

$$|D\rangle = \cos\left(\frac{90^\circ}{2}\right)|H\rangle + e^0 \sin\left(\frac{90^\circ}{2}\right)|V\rangle = \frac{1}{\sqrt{2}}|H\rangle + \frac{1}{\sqrt{2}}|V\rangle \tag{B.7}$$

$$|A\rangle = \cos\left(\frac{90^\circ}{2}\right)|H\rangle + e^{-i\pi} \sin\left(\frac{90^\circ}{2}\right)|V\rangle = \frac{1}{\sqrt{2}}|H\rangle - \frac{1}{\sqrt{2}}|V\rangle \tag{B.8}$$

$$|L\rangle = \cos\left(\frac{90^\circ}{2}\right)|H\rangle + e^{i\frac{\pi}{2}} \sin\left(\frac{90^\circ}{2}\right)|V\rangle = \frac{1}{\sqrt{2}}|H\rangle + \frac{i}{\sqrt{2}}|V\rangle \tag{B.9}$$

$$|R\rangle = \cos\left(\frac{90^\circ}{2}\right)|H\rangle + e^{-i\frac{\pi}{2}} \sin\left(\frac{90^\circ}{2}\right)|V\rangle = \frac{1}{\sqrt{2}}|H\rangle - \frac{i}{\sqrt{2}}|V\rangle \tag{B.10}$$

Even in this case, the three bases correspond to the Pauli eigenvectors operators σ_z, σ_x and σ_y. Through the *Poincarè sphere* (figure B.1), quite similar to the Bloch sphere, the light wave polarization states can be visually represented.

From the explanation above, it is clear that a qubit can assume an infinite number of states. Therefore, one could think that it could be possible to store an infinite quantity of information in a single qubit. Nevertheless, aside the fact that it is possible to encode a qubit as an arbitrary polarization of a photon, the subsequent measurement of the qubit state has a limited precision due to quantum mechanics laws. To be more precise, it is not possible to measure the precise state of a qubit since it would require infinite measurements. Besides, even with a large number of measurements on the same qubit, it is not even possible to estimate the state of a qubit with reasonable precision, since any measurements will inevitably modify the original qubit state. The theoretical explanation comes from the third postulate of the quantum mechanical theory. Furthermore, from the definition of *projective measurement*, derives the fact that, given the two measurement operators

$$M_+ = |H\rangle\langle H| + |V\rangle\langle V| \tag{B.11}$$

$$M_\times = |D\rangle\langle D| + |A\rangle\langle A| \tag{B.12}$$

choosing the M_+ operator to measure a state in the H/V basis returns the original state of the qubit. On the other hand, if the $M_\times$ is chosen then the state is projected on the D/A basis and collapses on one of the two states D or A with a 50% probability. For the sake of completeness, the two parts of the third postulate are recalled below as well as the definition of projective measurement.

Definition 1. *(Postulate IIIa). We associate with any observable $\mathcal{A}$ a self-adjoint operator A on the Hilbert space $\mathcal{H}_S$. The only possible outcome of a measurement of the observable $\mathcal{A}$ is one of the eigenvalues of the operator A. If the eigenvalue equation for the operator A is written as*

$$A|i\rangle = a_i|i\rangle \tag{B.13}$$

where $|i\rangle$ is an orthonormal basis of the eigenvectors of the operator A, and we expand the state vector $|\psi\rangle$ over the basis

$$|\psi\rangle = \sum_i c_i |i\rangle \tag{B.14}$$

then the probability that a measurement of the observable $\mathcal{A}$ results in an outcome a_i is given by

$$p(i) = p(a = a_i) = |\langle i | \psi\rangle|^2 = |c_i|^2 \tag{B.15}$$

Definition 2. *(Postulate IIIb). If a system is described by the state vector $|\psi\rangle$ and we measure an observable $\mathcal{A}$, obtaining the outcome a_i, then immediately after the measurement the state of the system is given by*

$$\frac{P_i |\psi\rangle}{\sqrt{\langle\psi| P_i |\psi\rangle}} \tag{B.16}$$

where P_i is the projection operator over the subspace corresponding to a_i.

Definition 3. *(Projective measurement). A projective measurement is described by a self-adjoint operator M with spectral decomposition given by*

$$M = \sum_i \lambda_i P_i = \sum_i \lambda_i |i\rangle \langle i| \tag{B.17}$$

where P_i is the projection operator over the subspace spanned by the eigenvectors corresponding to the eigenvalue λ_i. When the state $|\psi\rangle$ is measured with the operator M, the probability of getting λ_i as a result is

$$p(i) = \langle\psi| P_i |\psi\rangle \tag{B.18}$$

Bibliography

[1] J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli. "Architecture of field-programmable gate arrays". In: *Proceedings of the IEEE* 81.7 (July 1993), pp. 1013–1029. ISSN: 0018-9219. DOI: 10.1109/5.231340.

[2] R. H. Freeman. "Configurable electrical circuit having configurable logic elements and configurable interconnects". Pat. US4870302A. Sept. 1989.

[3] Paul Horowitz and Winfield Hill. *The art of electronics; 3rd ed.* Cambridge: Cambridge University Press, 2015.

[4] S. Asano, T. Maruyama, and Y. Yamaguchi. "Performance comparison of FPGA, GPU and CPU in image processing". In: *2009 International Conference on Field Programmable Logic and Applications*. Aug. 2009, pp. 126–131. DOI: 10.1109/FPL.2009.5272532.

[5] S. Kestur, J. D. Davis, and O. Williams. "BLAS Comparison on FPGA, CPU and GPU". In: *2010 IEEE Computer Society Annual Symposium on VLSI*. July 2010, pp. 288–293. DOI: 10.1109/ISVLSI.2010.84.

[6] L. Benini and G. De Micheli. "Networks on chips: a new SoC paradigm". In: *Computer* 35.1 (Jan. 2002), pp. 70–78. ISSN: 0018-9162. DOI: 10.1109/2.976921.

[7] C. He, A. Papakonstantinou, and D. Chen. "A novel SoC architecture on FPGA for ultra fast face detection". In: *2009 IEEE International Conference on Computer Design*. Oct. 2009, pp. 412–418. DOI: 10.1109/ICCD.2009.5413122.

[8] S. Velusamy et al. "Monitoring temperature in FPGA based SoCs". In: *2005 International Conference on Computer Design*. Oct. 2005, pp. 634–637. DOI: 10.1109/ICCD.2005.78.

[9] I. Bahri et al. "Hardware/Software Codesign Guidelines for System on Chip FPGA-Based Sensorless AC Drive Applications". In: *IEEE Transactions on Industrial Informatics* 9.4 (Nov. 2013), pp. 2165–2176. ISSN: 1551-3203. DOI: 10.1109/TII.2013.2245908.

[10] J. G. Tong, I. D. L. Anderson, and M. A. S. Khalid. "Soft-Core Processors for Embedded Systems". In: *2006 International Conference on Microelectronics*. Dec. 2006, pp. 170–173. DOI: 10.1109/ICM.2006.373294.

[11] R. Lysecky and F. Vahid. "A study of the speedups and competitiveness of FPGA soft processor cores using dynamic hardware/software partitioning". In: *Design, Automation and Test in Europe*. Mar. 2005, 18–23 Vol. 1. DOI: 10.1109/DATE.2005.38.

[12] H. Pham, S. Pillement, and S. J. Piestrak. "Low-overhead fault-tolerance technique for a dynamically reconfigurable softcore processor". In: *IEEE Transactions on Computers* 62.6 (June 2013), pp. 1179–1192. ISSN: 0018-9340. DOI: 10.1109/TC.2012.55.

[13] J. Kadlec, R. Bartosinski, and M. Danek. "Accelerating Microblaze Floating Point Operations". In: *2007 International Conference on Field Programmable Logic and Applications*. Aug. 2007, pp. 621–624. DOI: 10.1109/FPL.2007.4380731.

[14] A. F. Mondragon and J. Christman. "Hard Core vs. Soft Core: A Debate". In: *2012 ASEE Annual Conference & Exposition*. San Antonio, Texas: ASEE Conferences, June 2012.

[15] A. M. Caulfield et al. "Configurable Clouds". In: *IEEE Micro* 37.3 (2017), pp. 52–61. ISSN: 0272-1732. DOI: 10.1109/MM.2017.51.

[16] J. J. Wang et al. "SRAM based re-programmable FPGA for space applications". In: *IEEE Transactions on Nuclear Science* 46.6 (Dec. 1999), pp. 1728–1735. ISSN: 0018-9499. DOI: 10.1109/23.819146.

[17] L. Rockett et al. "Radiation Hardened FPGA Technology for Space Applications". In: *2007 IEEE Aerospace Conference*. Mar. 2007, pp. 1–7. DOI: 10.1109/AERO.2007.353098.

[18] S. Himavathi, D. Anitha, and A. Muthuramalingam. "Feedforward Neural Network Implementation in FPGA Using Layer Multiplexing for Effective Resource Utilization". In: *IEEE Transactions on Neural Networks* 18.3 (May 2007), pp. 880–888. ISSN: 1045-9227. DOI: 10.1109/TNN.2007.891626.

[19] Jiantao Qiu et al. "Going Deeper with Embedded FPGA Platform for Convolutional Neural Network". In: *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. FPGA '16. Monterey, California, USA: ACM, 2016, pp. 26–35. ISBN: 978-1-4503-3856-1. DOI: 10.1145/2847263.2847265.

[20] K. Irick et al. "A Hardware Efficient Support Vector Machine Architecture for FPGA". In: *2008 16th International Symposium on Field-Programmable Custom Computing Machines*. Apr. 2008, pp. 304–305. DOI: 10.1109/FCCM.2008.40.

[21] M. Papadonikolakis and C. Bouganis. "Novel Cascade FPGA Accelerator for Support Vector Machines Classification". In: *IEEE Transactions on Neural Networks and Learning Systems* 23.7 (July 2012), pp. 1040–1052. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2012.2196446.

[22] N. Walenta et al. "A fast and versatile quantum key distribution system with hardware key distillation and wavelength multiplexing". In: *New Journal of Physics* 16.1 (2014), p. 013047.

[23] Q. Shen et al. "An FPGA-Based TDC for Free Space Quantum Key Distribution". In: *IEEE Transactions on Nuclear Science* 60.5 (Oct. 2013), pp. 3570–3577. ISSN: 0018-9499. DOI: 10.1109/TNS.2013.2280169.

[24] H. Zhang et al. "A Real-Time QKD System Based on FPGA". In: *Journal of Lightwave Technology* 30.20 (Oct. 2012), pp. 3226–3234. ISSN: 0733-8724. DOI: 10.1109/JLT.2012.2217394.

[25] W. N. Chelton and M. Benaissa. "Fast Elliptic Curve Cryptography on FPGA". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 16.2 (Feb. 2008), pp. 198–205. ISSN: 1063-8210. DOI: 10.1109/TVLSI.2007.912228.

[26] A. Aysu, C. Patterson, and P. Schaumont. "Low-cost and area-efficient FPGA implementations of lattice-based cryptography". In: *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. June 2013, pp. 81–86. DOI: 10.1109/HST.2013.6581570.

[27] Damien Stucki et al. "Towards a high-speed quantum random number generator". In: *Proc.SPIE*. Vol. 8899. 2013. DOI: 10.1117/12.2029287.

[28] S. Tisa et al. "High-Speed Quantum Random Number Generation Using CMOS Photon Counting Detectors". In: *IEEE Journal of Selected Topics in Quantum Electronics* 21.3 (May 2015), pp. 23–29. ISSN: 1077-260X. DOI: 10.1109/JSTQE.2014.2375132.

[29] X. Zhang et al. "FPGA implementation of Toeplitz hashing extractor for real time post-processing of raw random numbers". In: *2016 IEEE-NPSS Real Time Conference (RT)*. June 2016, pp. 1–5. DOI: 10.1109/RTC.2016.7543094.

[30] X. Tian and K. Benkrid. "Mersenne Twister Random Number Generation on FPGA, CPU and GPU". In: *2009 NASA/ESA Conference on Adaptive Hardware and Systems*. July 2009, pp. 460–464. DOI: 10.1109/AHS.2009.11.

[31] David Barrie Thomas, Lee Howes, and Wayne Luk. "A Comparison of CPUs, GPUs, FPGAs, and Massively Parallel Processor Arrays for Random Number Generation". In: *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. FPGA '09. Monterey, California, USA: ACM, 2009, pp. 63–72. ISBN: 978-1-60558-410-2. DOI: 10.1145/1508128.1508139.

[32] F. Schwiegelshohn and M. Hübner. "Design of an attention detection system on the Zynq-7000 SoC". In: *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)*. Dec. 2014, pp. 1–6. DOI: 10.1109/ReConFig.2014.7032510.

[33] F. M. Siddiqui et al. "IPPro: FPGA based image processing processor". In: *2014 IEEE Workshop on Signal Processing Systems (SiPS)*. Oct. 2014, pp. 1–6. DOI: 10.1109/SiPS.2014.6986057.

[34] Wajdi Farhat et al. "Real-time embedded system for traffic sign recognition based on ZedBoard". In: *Journal of Real-Time Image Processing* (May 2017). ISSN: 1861-8219. DOI: 10.1007/s11554-017-0689-0.

[35] Avnet. *ZedBoard documentation*. URL: http://zedboard.org/product/zedboard.

[36] Xilinx. *Zynq-7000 SoC documentation*. URL: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf.

[37] Xilinx. *Zynq-7000 SoC Technical Reference Manual*. URL: https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.

[38] Xilinx. *Vivado Design Suite web page*. URL: https://www.xilinx.com/products/design-tools/vivado.html.

[39] Xilinx. *Xilinx Software Development Kit (XSDK) web page*. URL: https://www.xilinx.com/products/design-tools/embedded-software/sdk.html.

[40] Amazon Web Services. *The FreeRTOS™ Kernel web page*. URL: https://www.freertos.org.

[41] Xilinx. *AXI protocol reference guide*. URL: https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf.

[42] G. H. Mealy. "A method for synthesizing sequential circuits". In: *The Bell System Technical Journal* 34.5 (Sept. 1955), pp. 1045–1079. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1955.tb03788.x.

[43] Edward. F. Moore. "Gedanken-experiments on Sequential Machines". In: *Automata Studies. (AM-34) (Annals of Mathematics Studies)*. Ed. by C. E. Shannon and J. McCarthy. Vol. 34. Princeton, NJ, USA: Princeton University Press, 1956, 129—153.

[44] Andrew Rushton. *VHDL for Logic Synthesis*. 3rd. New York, NY, USA: John Wiley & Sons, Inc., 2011. ISBN: 0470688475.

[45] Xilinx. *Vivado Design Suite User Guide Synthesis UG901 (v2017.1)*. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/ug901-vivado-synthesis.pdf.

[46] K. Kuusilinna et al. "Finite state machine encoding for VHDL synthesis". In: *IEE Proceedings - Computers and Digital Techniques* 148.1 (Jan. 2001), pp. 23–30. ISSN: 1350-2387. DOI: 10.1049/ip-cdt:20010210.

[47] N. I. Rafla and B. L. Davis. "A Study of Finite State Machine Coding Styles for Implementation in FPGAs". In: *2006 49th IEEE International Midwest Symposium on Circuits and Systems*. Vol. 1. Aug. 2006, pp. 337–341. DOI: 10.1109/MWSCAS.2006.382066.

[48] "IEEE Standard VHDL Language Reference Manual". In: *IEEE Std 1076-1987* (1988). DOI: 10.1109/IEEESTD.1988.122645.

[49] Xilinx. *AXI GPIO LogiCORE IP Concat (v2.0)*. URL: https://www.xilinx.com/support/documentation/ip_documentation/axi_gpio/v2_0/pg144-axi-gpio.pdf.

[50] Xilinx. *AXI Central Direct Memory Access (CDMA) LogiCORE IP Product Guide (v4.1)*. URL: https://www.xilinx.com/support/documentation/ip_documentation/axi_cdma/v4_1/pg034-axi-cdma.pdf.

[51] Xilinx. *AXI Interconnect LogiCORE IP Product Guide (v2.1)*. URL: https://www.xilinx.com/support/documentation/ip_documentation/axi_interconnect/v2_1/pg059-axi-interconnect.pdf.

[52] Xilinx. *AXI Block RAM (BRAM) Controller LogiCORE IP Product Guide (v4.0)*. URL: https://www.xilinx.com/support/documentation/ip_documentation/axi_bram_ctrl/v4_0/pg078-axi-bram-ctrl.pdf.

[53] Xilinx. *Processing System 7 LogiCORE IP Product Guide (v5.5)*. URL: https://www.xilinx.com/support/documentation/ip_documentation/processing_system7/v5_5/pg082-processing-system7.pdf.

[54] Xilinx. *Block Memory Generator LogiCORE IP Product Guide (v8.3)*. URL: https://www.xilinx.com/support/documentation/ip_documentation/blk_mem_gen/v8_3/pg058-blk-mem-gen.pdf.

[55] Xilinx. *Clocking Wizard LogiCORE IP Product Guide (v5.3)*. URL: https://www.xilinx.com/support/documentation/ip_documentation/clk_wiz/v5_3/pg065-clk-wiz.pdf.

[56] Xilinx. *Vivado Design Suite User Guide Using Constraints UG903 (v2017.1)*. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_2/ug903-vivado-using-constraints.pdf.

[57] Xilinx. *Vivado Design Suite User Guide Implementation UG904 (v2017.3)*. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_3/ug904-vivado-implementation.pdf.

[58] The Qt Company. *Qt | Cross-platform software development for embedded & desktop*. URL: https://www.qt.io/.

[59] Zhiwei Zhou et al. "Design of a real-time fault diagnosis expert system for the EAST cryoplant". In: *Fusion Engineering and Design* 87.12 (2012). Proceedings of the 8th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research, pp. 2002 –2006. ISSN: 0920-3796. DOI: https://doi.org/10.1016/j.fusengdes.2012.04.016.

[60] A. Unluturk, O. Aydogdu, and U. Guner. "Design and PID control of two wheeled autonomous balance robot". In: *2013 International Conference on Electronics, Computer and Computation (ICECCO)*. Nov. 2013, pp. 260–264. DOI: 10.1109/ICECCO.2013.6718278.

[61] M. Alam and A. Azad. "Development of biomedical data acquisition system in Hard Real-Time Linux environment". In: *2012 International Conference on Biomedical Engineering (ICoBE)*. Feb. 2012, pp. 436–440. DOI: 10.1109/ICoBE.2012.6179053.

[62] Qt. *QWidget Class documentation*. URL: http://doc.qt.io/qt-5/qwidget.html.

[63] *Transmission Control Protocol*. RFC 793. Sept. 1981. DOI: 10.17487/RFC0793. URL: https://rfc-editor.org/rfc/rfc793.txt.

[64] W. Richard Stevens. *TCP/IP Illustrated (Vol. 1): The Protocols*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1993. ISBN: 0-201-63346-9.

[65] Adam Dunkels. "Design and Implementation of the lwIP TCP/IP Stack". In: *Swedish Institute of Computer Science* 2 (2001), p. 77.

[66] Kaushal Sanghai. *Building Complex VDK/LwIP Applications Using Blackfin Processors*. URL: http://www.analog.com/media/en/technical-documentation/application-notes/EE-312_Rev2.pdf.

[67] Stephen MacMahon Anirudha Sarangi and Upender Cherukupaly. *LightWeight IP Application Examples*. URL: https://www.xilinx.com/support/documentation/application_notes/xapp1026.pdf.

[68] *User Datagram Protocol*. RFC 768. Aug. 1980. DOI: 10.17487/RFC0768. URL: https://rfc-editor.org/rfc/rfc768.txt.

[69] Xilinx. *OS and Libraries Document Collection UG643 (2017.1)*. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/oslib_rm.pdf.

[70] quTools. *quTAU manual and documentation*. URL: https://www.qutools.com/quTAU/.

[71] Qt. *QPushButton Class documentation*. URL: http://doc.qt.io/archives/qt-4.8/qpushbutton.html.

[72] Qt. *QSpinBox Class documentation*. URL: http://doc.qt.io/qt-5/qspinbox.html.

[73] Qt. *QRadioButton Class documentation*. URL: http://doc.qt.io/archives/qt-4.8/qradiobutton.html.

[74] Qt. *QTimer Class documentation*. URL: http://doc.qt.io/qt-5/qtimer.html.

[75] Qt. *Signal & Slot documentation*. URL: http://doc.qt.io/archives/qt-4.8/signalsandslots.html.

[76] Qt. *QProgressBar Class documentation*. URL: http://doc.qt.io/qt-5/qprogressbar.html.

[77] Qt. *Qt Concurrent documentation*. URL: https://doc.qt.io/qt-5.11/qtconcurrent-index.html#.

[78] R. P. Hassett and E. H. Miller. "Multithreading Design of a Reliable Aerospace Computer". In: *IEEE Transactions on Aerospace and Electronic Systems* AES-2.6 (Nov. 1966), pp. 147–158. ISSN: 0018-9251. DOI: 10.1109/TAES.1966.4502004.

[79] D. M. Tullsen, S. J. Eggers, and H. M. Levy. "Simultaneous multithreading: Maximizing on-chip parallelism". In: *Proceedings 22nd Annual International Symposium on Computer Architecture*. June 1995, pp. 392–403.

[80] John L. Hennessy and David A. Patterson. *Computer Architecture, Fifth Edition: A Quantitative Approach*. 5th. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN: 012383872X, 9780123838728.

[81] Qt. *QDateTime Class documentation*. URL: `http://doc.qt.io/qt-5/qdatetime.html`.

[82] Qt. *QTime Class documentation*. URL: `http://doc.qt.io/qt-5/qtime.html`.

[83] Qt. *QDir Class documentation*. URL: `http://doc.qt.io/qt-5/qdir.html`.

[84] Qt. *QFile Class documentation*. URL: `http://doc.qt.io/qt-5/qfile.html`.

[85] Qt. *QByteArray Class documentation*. URL: `http://doc.qt.io/archives/qt-4.8/qbytearray.html`.

[86] Qt. *QDataStream Class documentation*. URL: `http://doc.qt.io/qt-5/qdatastream.html`.

[87] M. Stipčević. "Quantum random number generators and their use in cryptography". In: *2011 Proceedings of the 34th International Convention MIPRO*. May 2011, pp. 1474–1479.

[88] Werner Schindler and Wolfgang Killmann. "Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications". In: *Cryptographic Hardware and Embedded Systems - CHES 2002*. Ed. by Burton S. Kaliski, çetin K. Koç, and Christof Paar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 431–449. ISBN: 978-3-540-36400-9.

[89] Markus Dichtl. "How to Predict the Output of a Hardware Random Number Generator". In: *Cryptographic Hardware and Embedded Systems - CHES 2003*. Ed. by Colin D. Walter, Çetin K. Koç, and Christof Paar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 181–188. ISBN: 978-3-540-45238-6.

[90] Alan M. Ferrenberg, D. P. Landau, and Y. Joanna Wong. "Monte Carlo simulations: Hidden errors from "good" random number generators". In: *Phys. Rev. Lett.* 69 (23 Dec. 1992), pp. 3382–3384. DOI: `10.1103/PhysRevLett.69.3382`.

[91] Ian Goldberg and David Wagner. "Randomness and the Netscape browser". In: *Dr Dobb's Journal-Software Tools for the Professional Programmer* 21.1 (1996), pp. 66–71.

[92] Arjen K. Lenstra et al. *Ron was wrong, Whit is right*. Cryptology ePrint Archive, Report 2012/064. `https://eprint.iacr.org/2012/064`. 2012.

[93] Nadia Heninger et al. "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices". In: *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*. Bellevue, WA: USENIX, 2012, pp. 205–220. ISBN: 978-931971-95-9.

[94] Carlos Abellán et al. "Generation of Fresh and Pure Random Numbers for Loophole-Free Bell Tests". In: *Phys. Rev. Lett.* 115 (25 Dec. 2015), p. 250403. DOI: `10.1103/PhysRevLett.115.250403`.

[95] Lynden K. Shalm et al. "Strong Loophole-Free Test of Local Realism". In: *Phys. Rev. Lett.* 115 (25 Dec. 2015), p. 250402. DOI: 10.1103/PhysRevLett.115.250402.

[96] B. Hensen et al. "Loophole-free Bell inequality violation using electron spins separated by 1.3 kilometres". In: *Nature* 526 (Oct. 2015), 682 EP –.

[97] Marissa Giustina et al. "Significant-Loophole-Free Test of Bell's Theorem with Entangled Photons". In: *Phys. Rev. Lett.* 115 (25 Dec. 2015), p. 250401. DOI: 10.1103/PhysRevLett.115.250401.

[98] Francesco Vedovato et al. "Extending Wheeler's delayed-choice experiment to space". In: *Science Advances* 3.10 (2017). DOI: 10.1126/sciadv.1701180.

[99] S. Pironio et al. "Random numbers certified by Bell's theorem". In: *Nature* 464 (May 2010), 1021 EP –.

[100] B. G. Christensen et al. "Detection-Loophole-Free Test of Quantum Nonlocality, and Applications". In: *Phys. Rev. Lett.* 111 (13 Sept. 2013), p. 130406. DOI: 10.1103/PhysRevLett.111.130406.

[101] Peter Bierhorst et al. "Experimentally generated randomness certified by the impossibility of superluminal signals". In: *Nature* 556.7700 (2018), pp. 223–226. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0019-0.

[102] Yang Liu et al. "High-Speed Device-Independent Quantum Random Number Generation without a Detection Loophole". In: *Phys. Rev. Lett.* 120 (1 Jan. 2018), p. 010503. DOI: 10.1103/PhysRevLett.120.010503.

[103] S. Gómez et al. "Experimental nonlocality-based randomness generation with nonprojective measurements". In: *Phys. Rev. A* 97 (4 Apr. 2018), p. 040102. DOI: 10.1103/PhysRevA.97.040102.

[104] Xiongfeng Ma et al. "Quantum random number generation". In: *Npj Quantum Information* 2 (June 2016). Review Article, 16021 EP –. URL: http://dx.doi.org/10.1038/npjqi.2016.21.

[105] P. Owens J. Rarity and P. Tapster. "Quantum Random-number Generation and Key Sharing". In: *Journal of Modern Optics* 41.2435 (1994). DOI: ----.

[106] Thomas Jennewein et al. "A fast and compact quantum random number generator". In: *Review of Scientific Instruments* 71.4 (2000), pp. 1675–1680. DOI: 10.1063/1.1150518.

[107] M. Stipčević and B. Medved Rogina. "Quantum random number generator based on photonic emission in semiconductors". In: *Review of Scientific Instruments* 78.4 (2007), p. 045104. DOI: 10.1063/1.2720728.

[108] C. E. Shannon. "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27.3 (), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[109] Harald Fürst et al. "High speed optical quantum random number generation". In: *Opt. Express* 18.12 (June 2010), pp. 13029–13037. DOI: 10.1364/OE.18.013029.

[110] Rodney Loudon. *The Quantum Theory of Light*. Oxford: Clarendon Press, 1973.

[111] John von Neumann. "Various techniques used in connection with random digits". In: *Monte Carlo Method*. Ed. by A.S. Householder, G.E. Forsythe, and H.H. Germond. Washington, D.C.: U.S. Government Printing Office: National Bureau of Standards Applied Mathematics Series, 12, 1951, pp. 36–38.

[112] Yuval Peres. *Iterating von Neumann's Procedure for Extracting Random Bits*. 1992. DOI: 10.1214/aos/1176348543.

[113] Peter Elias. "The efficient construction of an unbiased random sequence". In: *Ann. Math. Statist.* 43 (1972), 865—870.

[114] B. Ryabko and E. Matchikina. "Fast and efficient construction of an unbiased random sequence". In: *Proceedings. 1998 IEEE International Symposium on Information Theory (Cat. No.98CH36252)*. Aug. 1998, pp. 472–. DOI: 10.1109/ISIT.1998.709077.

[115] H. Zhou and J. Bruck. "A Universal Scheme for Transforming Binary Algorithms to Generate Random Bits from Loaded Dice". In: *ArXiv e-prints* (Sept. 2012). arXiv: 1209.0726 [cs.IT].

[116] Excelitas. *SPCM-AQRH Single Photon Counting Module Datasheet*. URL: http://www.excelitas.com/Downloads/DTS_SPCM-AQRH.pdf.

[117] Clifford E Cummings, Don Mills, and Steve Golson. "Asynchronous & synchronous reset design techniques-part deux". In: *SNUG Boston* 9 (2003).

[118] Xilinx. *LogiCORE IP Concat (v2.1)*. URL: https://www.xilinx.com/support/documentation/ip_documentation/xilinx_com_ip_xlconcat/v2_1/pb041-xilinx-com-ip-xlconcat.pdf.

[119] Qt. *QLCDNumber Class documentation*. URL: http://doc.qt.io/qt-5/qlcdnumber.html.

[120] J. A. Wheeler. "The 'Past' and the 'Delayed-Choice' Double-Slit Experiment". In: *Mathematical Foundations of Quantum Theory*. Ed. by A. R. Marlow. Academic Press, 1978, pp. 9–48.

[121] John Archibald Wheeler. *Quantum Theory and Measurement*. Princeton University Press, 1983. Chap. Law without law, pp. 182–213.

[122] Vincent Jacques et al. "Experimental Realization of Wheeler's Delayed-Choice Gedanken Experiment". In: *Science* 315.5814 (2007), pp. 966–968. ISSN: 0036-8075. DOI: 10.1126/science.1136303.

[123] A. G. Manning et al. "Wheeler's delayed-choice gedanken experiment with a single atom". In: *Nature Physics* 11 (May 2015), 539 EP –.

[124] Davide G. Marangon, Giuseppe Vallone, and Paolo Villoresi. "Source-Device-Independent Ultrafast Quantum Random Number Generation". In: *Phys. Rev. Lett.* 118 (6 Feb. 2017), p. 060503. DOI: 10.1103/PhysRevLett.118.060503.

[125] M. Avesani et al. "Secure heterodyne-based quantum random number generator at 17 Gbps". In: *ArXiv e-prints* (Jan. 2018). arXiv: 1801.04139 [quant-ph].

[126] C. Niclass, M. Sergio, and E. Charbon. "A Single Photon Avalanche Diode Array Fabricated in Deep-Submicron CMOS Technology". In: *Proceedings of the Design Automation Test in Europe Conference*. Vol. 1. Mar. 2006, pp. 1–6. DOI: 10.1109/DATE.2006.243987.

[127] E. Charbon. "Single-photon imaging in complementary metal oxide semiconductor processes". In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 372.2012 (2014). ISSN: 1364-503X. DOI: 10.1098/rsta.2013.0100.

[128] Samuel Burri et al. "Jailbreak Imagers: Transforming a Single-Photon Image Sensor into a True Random Number Generator". In: *International Image Sensor Workshop*. EPFL-CONF-191217. 2013.

[129] Samuel Burri et al. "LinoSPAD: a time-resolved 256x1 CMOS SPAD line sensor system featuring 64 FPGA-based TDC channels running at up to 8.5 giga-events per second". In: *Optical Sensing And Detection Iv*. Proceedings of SPIE 9899 (2016), 10. UNSP 98990D.

[130] Claudio Favi and Edoardo Charbon. "A 17 ps Time-to-digital Converter Implemented in 65 nm FPGA Technology". In: *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. FPGA '09. Monterey, California, USA: ACM, 2009, pp. 113–120. ISBN: 978-1-60558-410-2. DOI: 10.1145/1508128.1508145.

[131] M. Fishburn et al. "A 19.6 ps, FPGA-Based TDC With Multiple Channels for Open Source Applications". In: *IEEE Transactions on Nuclear Science* 60.3 (June 2013), pp. 2203–2208. ISSN: 0018-9499. DOI: 10.1109/TNS.2013.2241789.

[132] M. Mota et al. "A flexible multi-channel high-resolution time-to-digital converter ASIC". In: *2000 IEEE Nuclear Science Symposium. Conference Record (Cat. No.00CH37149)*. Vol. 2. Oct. 2000, 9/155–9/159 vol.2. DOI: 10.1109/NSSMIC.2000.949889.

[133] B. K. Swann et al. "A 100-ps time-resolution CMOS time-to-digital converter for positron emission tomography imaging applications". In: *IEEE Journal of Solid-State Circuits* 39.11 (Nov. 2004), pp. 1839–1852. ISSN: 0018-9200. DOI: 10.1109/JSSC.2004.835832.

[134] H. Flemming and H. Deppe. "Development of high resolution TDC ASICs at GSI". In: *2007 IEEE Nuclear Science Symposium Conference Record*. Vol. 1. Oct. 2007, pp. 322–325. DOI: 10.1109/NSSMIC.2007.4436340.

[135] J. Zheng et al. "Low-Cost FPGA TDC With High Resolution and Density". In: *IEEE Transactions on Nuclear Science* 64.6 (June 2017), pp. 1401–1408. ISSN: 0018-9499. DOI: 10.1109/TNS.2017.2705802.

[136] Jian Song, Qi An, and Shubin Liu. "A high-resolution time-to-digital converter implemented in field-programmable-gate-arrays". In: *IEEE Transactions on Nuclear Science* 53.1 (Feb. 2006), pp. 236–241. ISSN: 0018-9499. DOI: 10.1109/TNS.2006.869820.

[137] R. L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-key Cryptosystems". In: *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342.

[138] C. E. Shannon. "Communication theory of secrecy systems". In: *The Bell System Technical Journal* 28.4 (Oct. 1949), pp. 656–715. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1949.tb00928.x.

[139] Charles H. Bennett and Gilles Brassard. "Quantum cryptography: Public key distribution and coin tossing". In: *Theoretical Computer Science* 560 (2014). Theoretical Aspects of Quantum Cryptography – celebrating 30 years of BB84, pp. 7 –11. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2014.05.025.

[140] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. "How to Reduce your Enemy's Information (extended abstract)". In: *Advances in Cryptology — CRYPTO '85 Proceedings*. Ed. by Hugh C. Williams. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 468–476. ISBN: 978-3-540-39799-1.

[141] Charles H. Bennett et al. "Experimental quantum cryptography". In: *Journal of Cryptology* 5.1 (Jan. 1992), pp. 3–28. ISSN: 1432-1378. DOI: 10.1007/BF00191318.

[142] Charles H. Bennett, Gilles Brassard, and N. David Mermin. "Quantum cryptography without Bell's theorem". In: *Phys. Rev. Lett.* 68 (5 Feb. 1992), pp. 557–559. DOI: 10.1103/PhysRevLett.68.557.

[143] W. K. Wootters and W. H. Zurek. "A single quantum cannot be cloned". In: *Nature* 299 (Oct. 1982), 802–803. DOI: 10.1038/299802a0.

[144] D. Dieks. "Communication by EPR devices". In: *Physics Letters A* 92 (6 Nov. 1982), 271–272. DOI: 10.1016/0375-9601(82)90084-6.

[145] Gilles Brassard and Louis Salvail. "Secret-Key Reconciliation by Public Discussion". In: *Advances in Cryptology — EUROCRYPT '93*. Ed. by Tor Helleseth. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 410–423. ISBN: 978-3-540-48285-7.

[146] C. H. Bennett et al. "Generalized privacy amplification". In: *IEEE Transactions on Information Theory* 41.6 (Nov. 1995), pp. 1915–1923. ISSN: 0018-9448. DOI: 10.1109/18.476316.

[147] Gilles Brassard et al. “Limitations on Practical Quantum Cryptography”. In: *Phys. Rev. Lett.* 85 (6 Aug. 2000), pp. 1330–1333. DOI: 10.1103/PhysRevLett.85.1330.

[148] Nitin Jain et al. “Attacks on practical quantum key distribution systems (and how to prevent them)”. In: *Contemporary Physics* 57.3 (2016), pp. 366–387. DOI: 10.1080/00107514.2016.1148333.

[149] Valerio Scarani et al. “The security of practical quantum key distribution”. In: *Rev. Mod. Phys.* 81 (3 Sept. 2009), pp. 1301–1350. DOI: 10.1103/RevModPhys.81.1301.

[150] Nicolas Gisin et al. “Quantum cryptography”. In: *Rev. Mod. Phys.* 74 (1 Mar. 2002), pp. 145–195. DOI: 10.1103/RevModPhys.74.145.

[151] Helle Bechmann-Pasquinucci and Nicolas Gisin. “Incoherent and coherent eavesdropping in the six-state protocol of quantum cryptography”. In: *Physical Review A - PHYS REV A* 59 (June 1999), pp. 4238–4248.

[152] Lana Sheridan and Valerio Scarani. “Security proof for quantum key distribution using qudit systems”. In: *Phys. Rev. A* 82 (3 Sept. 2010), p. 030301. DOI: 10.1103/PhysRevA.82.030301.

[153] Renato Renner, Nicolas Gisin, and Barbara Kraus. “Information-theoretic security proof for quantum-key-distribution protocols”. In: *Physical Review A - Atomic, Molecular, and Optical Physics* 72.1 (2005), pp. 1–17. DOI: 10.1103/PhysRevA.72.012332.

[154] Hoi-Kwong Lo, H.F. Chau, and M. Ardehali. “Efficient Quantum Key Distribution Scheme and a Proof of Its Unconditional Security”. In: *Journal of Cryptology* 18.2 (Apr. 2005), pp. 133–165. DOI: 10.1007/s00145-004-0142-y.

[155] T. C. Ralph. “Continuous variable quantum cryptography”. In: *Phys. Rev. A* 61 (1 Dec. 1999), p. 010303. DOI: 10.1103/PhysRevA.61.010303.

[156] I. Marcikic et al. “Time-bin entangled qubits for quantum communication created by femtosecond pulses”. In: *Phys. Rev. A* 66 (6 Dec. 2002), p. 062308. DOI: 10.1103/PhysRevA.66.062308.

[157] Davide Bacco et al. “Experimental quantum key distribution with finite-key security analysis for noisy channels”. In: *Nature Communications* 4 (Sept. 2013). Article, 2363 EP –.

[158] H.-J. Briegel et al. “Quantum Repeaters: The Role of Imperfect Local Operations in Quantum Communication”. In: *Phys. Rev. Lett.* 81 (26 Dec. 1998), pp. 5932–5935. DOI: 10.1103/PhysRevLett.81.5932.

[159] Louis Salvail et al. “Security of Trusted Repeater Quantum Key Distribution Networks”. In: *Journal of Computer Security* 18 (Jan. 2010), pp. 61–87.

[160] J G Rarity et al. "Ground to satellite secure key exchange using quantum cryptography". In: *New Journal of Physics* 4.1 (2002), p. 82. URL: http://stacks.iop.org/1367-2630/4/i=1/a=382.

[161] M. Aspelmeyer et al. "Long-distance quantum communication with entangled photons using satellites". In: *IEEE Journal of Selected Topics in Quantum Electronics* 9.6 (Nov. 2003), pp. 1541–1551. ISSN: 1077-260X. DOI: 10.1109/JSTQE.2003.820918.

[162] Andrea Tomaello et al. "Link budget and background noise for satellite quantum key distribution". In: *Advances in Space Research* 47.5 (2011). Scientific applications of Galileo and other Global Navigation Satellite Systems - II, pp. 802 –810. ISSN: 0273-1177. DOI: https://doi.org/10.1016/j.asr.2010.11.009.

[163] David Rideout et al. "Fundamental quantum optics experiments conceivable with satellites—reaching relativistic distances and velocities". In: *Classical and Quantum Gravity* 29.22 (2012), p. 224011.

[164] J-P Bourgoin et al. "A comprehensive design and performance analysis of low Earth orbit satellite quantum communication". In: *New Journal of Physics* 15.2 (2013), p. 023006.

[165] David Edward Bruschi et al. "Spacetime effects on satellite-based quantum communications". In: *Phys. Rev. D* 90 (4 Aug. 2014), p. 045041. DOI: 10.1103/PhysRevD.90.045041.

[166] Ursin, R. et al. "Space-quest, experiments with quantum entanglement in space". In: *Europhysics News* 40.3 (2009), pp. 26–29. DOI: 10.1051/epn/2009503.

[167] T. Scheidl, E. Wille, and R. Ursin. "Quantum optics experiments using the International Space Station: a proposal". In: *New Journal of Physics* 15.4 (2013), p. 043008.

[168] T. Jennewein et al. *The NanoQEY mission: ground to space quantum key and entanglement distribution using a nanosatellite*. 2014. DOI: 10.1117/12.2067548.

[169] Paolo Villoresi et al. "Experimental verification of the feasibility of a quantum channel between space and Earth". In: *New Journal of Physics* 10.3 (2008), p. 033038.

[170] Juan Yin et al. "Experimental quasi-single-photon transmission from satellite to earth". In: *Opt. Express* 21.17 (Aug. 2013), pp. 20032–20040. DOI: 10.1364/OE.21.020032.

[171] Giuseppe Vallone et al. "Experimental Satellite Quantum Communications". In: *Phys. Rev. Lett.* 115 (4 July 2015), p. 040502. DOI: 10.1103/PhysRevLett.115.040502.

[172] Zhongkan Tang et al. "Generation and Analysis of Correlated Pairs of Photons aboard a Nanosatellite". In: *Phys. Rev. Applied* 5 (5 May 2016), p. 054022. DOI: 10.1103/PhysRevApplied.5.054022. URL: https://link.aps.org/doi/10.1103/PhysRevApplied.5.054022.

[173] Giuseppe Vallone et al. "Interference at the Single Photon Level Along Satellite-Ground Channels". In: *Phys. Rev. Lett.* 116 (25 June 2016), p. 253601. DOI: 10.1103/PhysRevLett.116.253601.

[174] Daniele Dequal et al. "Experimental single-photon exchange along a space link of 7000 km". In: *Phys. Rev. A* 93 (1 Jan. 2016), p. 010301. DOI: 10.1103/PhysRevA.93.010301.

[175] Kevin Günthner et al. "Quantum-limited measurements of optical signals from a geostationary satellite". In: *Optica* 4.6 (June 2017), pp. 611–616. DOI: 10.1364/OPTICA.4.000611.

[176] Juan Yin et al. "Satellite-based entanglement distribution over 1200 kilometers". In: *Science* 356.6343 (2017), pp. 1140–1144. ISSN: 0036-8075. DOI: 10.1126/science.aan3211.

[177] L. Calderaro et al. "Towards Quantum Communication from Global Navigation Satellite System". In: *ArXiv e-prints* (Apr. 2018). arXiv: 1804.05022 [quant-ph].

[178] Sheng-kai Liao et al. "Satellite-to-ground quantum key distribution". In: *Nature* 549 (Aug. 2017), 43 EP –. DOI: 10.1038/nature23655.

[179] Hoi-Kwong Lo, Xiongfeng Ma, and Kai Chen. "Decoy State Quantum Key Distribution". In: *Phys. Rev. Lett.* 94 (23 June 2005), p. 230504. DOI: 10.1103/PhysRevLett.94.230504.

[180] Giuseppe Bianco, Vincenza Luceri, and Rosa Pacione. "The Space Geodesy Centre of the Italian Space Agency: from ITRF to EUREF". In: *Rendiconti Lincei. Scienze Fisiche e Naturali* 29.1 (June 2018), pp. 35–39. ISSN: 1720-0776. DOI: 10.1007/s12210-018-0698-3.

[181] R. Ursin et al. "Entanglement-based quantum communication over 144 km". In: *Nature Physics* 3 (June 2007). Article, 481 EP –.

[182] T. Schmitt-Manderbach et al. "Experimental Demonstration of Free-Space Decoy-State Quantum Key Distribution over 144 km". In: *2007 European Conference on Lasers and Electro-Optics and the International Quantum Electronics Conference*. June 2007, pp. 1–1. DOI: 10.1109/CLEOE-IQEC.2007.4386755.

[183] C. Bonato et al. "Feasibility of satellite quantum key distribution". In: *New Journal of Physics* 11.4 (2009), p. 045017.

[184] Costantino Agnesi et al. "Exploring the boundaries of quantum mechanics: advances in satellite quantum communications". In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 376.2123 (2018). ISSN: 1364-503X. DOI: 10.1098/rsta.2017.0461.

[185] Thorlabs. *Thorlabs APT Motion control Software*. URL: https://www.thorlabs.com/software_pages/ViewSoftwarePage.cfm?Code=Motion_Control.

[186] Thorlabs. *Thorlabs Liquid Crustal LCC25 Manual*. URL: https://www.thorlabs.com/drawings/e3dac90d26a21567-31CBEBD4-0355-ED3A-B30205D61C158FDA/LCC25-Manual.pdf.

[187] David L. Mills. *Network Time Protocol (Version 3) Specification, Implementation and Analysis*. Mar. RFC 1305, 1992. URL: https://tools.ietf.org/pdf/rfc1305.pdf.

[188] Trimble. *Thunderbolt E GPS Disciplined Clock documentation*. URL: https://www.trimble.com/timing/thunderbolt-e.aspx.

[189] Nurul T. Islam et al. "Securing quantum key distribution systems using fewer states". In: *Phys. Rev. A* 97 (4 Apr. 2018), p. 042347. DOI: 10.1103/PhysRevA.97.042347.

[190] D. Rusca et al. "Security proof for a simplified BB84-like QKD protocol". In: *ArXiv e-prints* (Aug. 2018). arXiv: 1808.08259 [quant-ph].

[191] Xiongfeng Ma et al. "Practical decoy state for quantum key distribution". In: *Phys. Rev. A* 72 (1 July 2005), p. 012326. DOI: 10.1103/PhysRevA.72.012326.

[192] Yi Zhao et al. "Experimental Quantum Key Distribution with Decoy States". In: *Phys. Rev. Lett.* 96 (7 Feb. 2006), p. 070502. DOI: 10.1103/PhysRevLett.96.070502.

[193] Gooch & Housego. *EM655 DFB laser product web page*. URL: https://goochandhousego.com/product-categories/dfb-lasers-modules/.

[194] Thorlabs. *LN81S-FC Intensity Modulator product web page*. URL: https://www.thorlabs.com/thorproduct.cfm?partnumber=LN81S-FC.

[195] iXblue. *MPZ-LN-20 Phase Modulator product web page*. URL: https://photonics.ixblue.com/product-detail/mpz-ln-20.

[196] Xilinx. *Xilinx 7 Series FPGA Libraries Guide for Schematic Designs*. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx13_2/7series_scm.pdf.

[197] Xilinx. *FMC XM105 Debug Card User Guide*. URL: https://www.xilinx.com/support/documentation/boards_and_kits/ug537.pdf.

[198] iXblue. *DR-AN-20-HO RF modulator driver documentation*. URL: https://photonics.ixblue.com/files/files/pdf/RF_Drivers_Modules/DR-AN-20-HO.pdf.

[199] Mini-Circuits. *ZX85-12G+ Bias-Tee documentation*. URL: https://ww2.minicircuits.com/pdfs/ZX85-12G+.pdf.

[200] ID Quantique. *ID281 Superconducting Nanowire web page*. URL: https://www.idquantique.com/single-photon-systems/products/id281/.

[201] ID Quantique. *ID900 Time Controller web page*. URL: https://www.idquantique.com/single-photon-systems/products/id900-time-controller/.

[202] Giuliano Benenti, Giulio Casati, and Giuliano Strini. *Principles of Quantum Computation and Information*. World Scientific, 2007. DOI: 10.1142/5838.

[203] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[204] Benjamin Schumacher. "Quantum coding". In: *Phys. Rev. A* 51 (4 Apr. 1995), pp. 2738–2747. DOI: 10.1103/PhysRevA.51.2738.

[205] B.E.A. Saleh and M.C. Teich. *Fundamentals of Photonics*. Wiley Series in Pure and Applied Optics. Wiley, 2007. ISBN: 9780471358329.