POLITECNICO DI TORINO

Master's Degree in Quantum Engineering

Master's Degree Thesis

FPGA Implementation of High-Rate Randomness Extraction for Quantum Random Number Generators

Supervisors

Prof. Roberto PROIETTI

Dott. Ing. Giuseppe DE FALCO

Dott. Ric. Carlo LIORNI

Candidate

Emanuele ZITOLI

Academic Year 2024/2025

Abstract

Random number generators are fundamental in several real-world applications, like cryptography and simulations techniques, such as Monte Carlo methods. This need for randomness is commonly bridged using pseudo-random number generators, which rely on deterministic mathematical algorithms to produce uniform random sequences. It is not rare to find situations in which the deterministic aspect of these tools is not acceptable. To make up for this need, true random number generators can be used, in particular Quantum Random Number Generators (QRNGs), that exploit the inherently random nature of quantum mechanical phenomena to generate numbers.

Several paradigms of QRNGs are possible, regarding the elements of the system that can be considered as trusted. Semi-device-independent schemes offer a trade-off between the speed of trusted-devices and the security assurances of device-independent generators.

Multiple possibilities are available also concerning the physical phenomenon exploited to produce the numbers. Optical QRNGs, that use continuous variables of the electromagnetic field, have proven to be cost-efficient solutions that support very high generation rates.

Real implementations of QRNGs are always affected by noise, because of the imperfections that characterize the experimental apparata. This aspect reduces the quality of the randomness in the numbers produced, therefore post-processing and in particular randomness extraction, is needed to eliminate the unwanted correlations. Multiple strategies are possible for this kind of post-processing in real time, including Central Processing Unit (CPU) and Graphics Processing Unit (GPU) implementations. With these, the generation rate does not meet the demand set by real high-speed systems, such as those for encryption of communication links. A possibility that has proven to be fast enough is a hardware-based approach with a Field Programmable Gate Array (FPGA). This device combines the flexibility of a re-programmable component with the efficiency of a hardware implementation and its concurrent computation capabilities.

This work reports on the randomness extraction from two QRNG implementations: a trusted QRNG, based on the fluctuations of the phase of a laser diode and a source device-independent QRNG, based on the fluctuations of the vacuum state.

The post-processing is realized on a System on Chip (SoC), including a CPU and a FPGA. The circuit on the programmable logic has been designed to efficiently execute randomness extraction, based on Toeplitz hashing. It consists in a three-stage pipeline, instantiated multiple times, to allow for parallel computation of multiple data blocks. Both an online and an offline design have been implemented.

The former demonstrates the possibility of high-rate randomness extraction, by processing the data coming from the QRNG in real time. The latter has been implemented to conveniently read raw data batches from an SD card and save the processed numbers on this same storage device.

The bit strings obtained from randomness extraction have been tested using the autocorrelation function and the NIST test suite, that certify the quality of the randomness of the produced numbers. The analysis has been carried for both the QRNGs and the results satisfy the NIST requirements, showing uniformly distributed $P - values$ and an acceptable proportion of successful sequences.

These results reinforce the applicability of programmable logic to quantum communication applications, representing a powerful and convenient solution to real-time data processing.

Table of Contents

List of Figures

Chapter 1

Introduction

The generation of random numbers is fundamental in numerous domains, from simulations to secure communications. Nowadays classical random number generation methods based on Pseudo-Random Number Generators (PRNGs) provide a cheap yet still effective solution. However for advanced applications, requiring true randomness, the deterministic behavior of such systems does not represent a valid solution. In particular, due to the continuously growing interest in Quantum Key Distribution (QKD) protocols, like BB84 [1], the generation of cryptographic keys requires trusted randomness to avoid potential safety issues. At the forefront of meeting these demands are optical Quantum Random Number Generators (QRNGs), in which true randomness is guaranteed by fundamental quantum mechanical processes [2], [3].

These kinds of apparata can be realized in many ways, with some of the most relevant being vacuum fluctuation-based [4] and phase fluctuation-based [5]. The obtained bits, being extrapolated from actual imperfect devices, will inevitably be correlated with classical noise, reducing the security of the system [6].

To prevent an opponent from launching an attack using this classical and quantum side-information, universal hashing functions are used, with Trevisan extractor [7] and Toeplitz-hash extractor [8] being the most relevant options, as they are often used in privacy amplification schemes [9].

Efficient software-based algorithms are available for this kind of post-processing, like those involving Fast Fourier Transform (FFT) [10], but the bit-rate requirements for actual applications are hardly met. A solution could lie in the parallel processing capabilities of the Field Programmable Gate Array (FPGA), providing a hardware-based alternative that supports real-time post-processing [11].

This chapter provides an overview on Random Number Generators (RNGs) in Sec. 1.1, with a focus on QRNGs and their possible applications. Additionally an explanation on the importance of FPGAs in this context is given in Sec. 1.2.

1.1 Random Number Generators

The need for random numbers is a problem that spans a wide range of fields, from aspects that are closer to every-day life, like weather forecast and lottery, to more technical ones, like simulations and cryptography related tasks. The latter field has the primary objective of securely exchanging information between parties. This requires the use of cryptographic keys to hide the content of communications or to authenticate a user trying to access, for example, a banking account. The possibility that a malicious entity could be able to predict the generation outcome of such cryptographic keys, has brought a lot of attention on the mechanisms used to produce random numbers [2].

1.1.1 Pseudo-Random Number Generators

Most of existing random number generators are based on mathematical functions that, given an initial seed, deterministically produce an output string. An example is the Linear-Feedback Shift Register (LFSR), which is defined by a primitive polynomial. Based on its degree and structure, a shift register and a proper feedback network are constructed to produce an output string. However, the sequence of random numbers produced has a periodicity that depends on the degree of the primitive polynomial. Periodicity is a characteristic that implies a recurring pattern inside the generated string. Moreover, it is evident that knowing the initial seed and the primitive polynomial, one is able to reconstruct the exact sequence. This is the critical aspect present in every deterministic source of randomness [12].

Some variations on this simple structure are present to increase the unpredictability of the output sequence [13].

Another popular PRNG is based on linear recursion in modular arithmetic and it is referred to as Linear Congruential Generator (LCG). It is able to generate a random sequence by exploiting a simple equation, that includes several parameters and a starting seed. Again it is evident that knowing this first element of the sequence and the parameters involved, it is possible to reproduce the entire sequence. One more possibility for a PRNG is based on Cellular Automata (CA). This model of computation uses a grid of cells, each characterized by a particular state. This state depends on the one of the neighboring cells and on a particular rule. The state evolution of the system may be used to generate a sequence of random numbers with excellent characteristics [14].

It is evident how these generators can be beneficial in many use cases, providing random sequences with a uniform distribution by starting from a certain set of determined parameters. Notice that by knowing these, it is possible to reproduce a result, which may prove useful in some simulation scenario. Moreover they are the simplest and fastest RNGs with respect to the alternatives that are described in

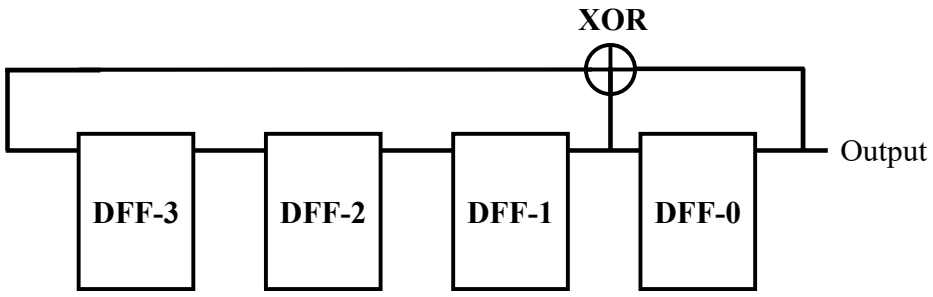


Figure 1.1: 4-bit LFSR with primitive polynomial $x^4 + x + 1$. Data Flip-Flops (DFFs) implement a shift register and a feedback network is composed of an XOR operation between the outputs of DFF-0 and DFF-1, which will feed the input of DFF-3.

the following sections. However the deterministic behavior that characterizes them is something that cannot be always overlooked. Therefore other technologies may be necessary [15].

1.1.2 Classical True Random Number Generators

As said, for many scenarios the deterministic behavior of PRNGs is not acceptable. Therefore it is necessary to obtain a random number generator that does not rely on mathematical functions but rather on the unpredictability of some physical process. This is what is done with Classical True Random Number Generators (CTRNG), which use phenomena like atmospheric noise, cosmic background radiation, thermal noise, noises in electronic circuits and chaotic systems. The numbers obtained from these systems, however, appear random because the model that describes them is too complex for a machine to make any predictions on the state of the system at a given time. Therefore, the randomness is not certified by the nature of the phenomenon itself, but rather by the complexity of the problem. In the future, thanks to the technological improvements or the development of a more mature theory, such complexity could be overcome. Moreover there are limitations regarding the generation rate, which is generally slower compared to that of PRNGs [2].

1.1.3 Quantum Random Number Generators

The search for random numbers inevitably leads to quantum mechanics, which is an inherently non-deterministic theory, that describes peculiar phenomena like superposition states and non-local correlations between particles. These phenomena can be used to provide novel sources of true randomness [16].

Some examples reported in literature include non-optical randomness generation processes like radioactive decays, which are only explainable using the uncertainty principle of quantum mechanics. It is possible to use the arrival times of the pulses produced by a Geiger counter, to generate random numbers. Such systems are obviously to be used with particular care, due to the dangerous nature of the randomness source, and do not offer a high generation rate [17].

Other non-optical sources are atomic systems, like trapped ions, which require very complex setups and do not give particularly high generation rates [18].

Systems that appear to be more suited for randomness generation are the ones based on optical phenomena, thanks to their simpler setups and faster generation rates [2].

Single Photon Detector-Based Optical QRNGs

One of the possible approaches to optical QRNGs relies on single-photon detection and generation, which are realized using expensive instrumentation and inherently restrict the generation rate [19]. Specifically, the response time of single photon detectors severely limits the achievable throughput. However, this type of QRNG gives some insights on the possibility of using quantum mechanics to generate randomness. An interesting strategy relies on the superposition principle and the collapse of a qubit state upon measurement. For example take a setup like in Fig. 1.2: it is possible to encode the quantum state of a single photon in its polarization. Such state can be represented using the basis $|H\rangle$ and $|V\rangle$, which indicate respectively horizontal and vertical polarization. Let's suppose that the single photon source is able to produce exactly a single photon in the diagonal polarization state using a polarizer at 45 degrees. In the Dirac notation, this photon can be represented as $(|H\rangle + |V\rangle)/\sqrt{2}$. When the photon reaches the Polarization Beam Splitter (PBS) it will exit through one of its output ports, based on the photon collapsing to the $|H\rangle$ or $|V\rangle$ quantum state. This happens with probability 1/2 for each of the two states. To which state the system collapses, can be understood by determining which of the two single photon detectors has clicked [20].

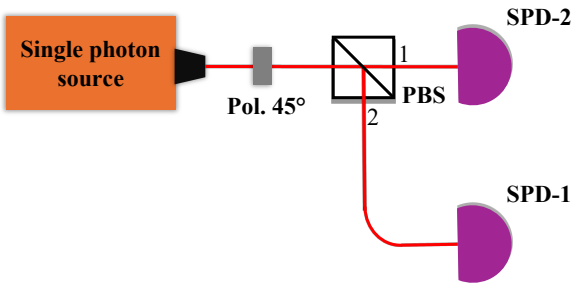


Figure 1.2: Single photon detector based QRNG. Single photons produced by the source are prepared with diagonal polarization with a polarizer and, based on the port of the Polarization Beam Splitter (PBS) they exit, it is possible to state that either vertical or horizontal polarization has been measured for that state. This measurement has to be performed with Single Photon Detectors (SPDs). Adapted from [20].

Macro-Scopic Photo Detection

In order to circumvent the major limitation of using single photon detectors, it is possible to measure quantities like amplitudes of an electromagnetic field to generate random numbers, while still exploiting quantum mechanical phenomena. Notable examples use vacuum noise, which will be explained in depth in Sec. 2.2, phase noise of spontaneous emission, in which a possible implementation will be explained in Sec. 2.1, and randomized phase and amplitude of an electromagnetic field due to Raman scattering [2].

Trusted, Self-Testing and Semi-Self-Testing QRNGs

A further distinction can be made on the possible implementations of a QRNG, based on which part of the device is considered as trusted. A trusted QRNG means that its behavior is sufficiently well characterized and verified, that its outputs can be assumed to follow the intended theoretical model and not depend on uncontrolled classical variables or imperfections. However it is not possible to determine if the generated numbers are truly random or if they have been manipulated to be highly predictable for an adversary, since the QRNG behavior depends on the entire device implementation and an external user may take control of it. Generally trusted

implementations have the simplest setups and can achieve high generation rates at the cost of having no security assurances [21].

In contexts like cryptography, this is hardly acceptable. For this reason self-testing QRNGs can be used, in which the randomness of the output is certified to be independent of the device implementation. In fact they are also known as Device-Independent (DI). For the no-signalling condition in the Bell tests, it said that an output cannot be random or it may be predetermined by local hidden variables, if it does not violate the Bell inequalities. Under this assumption several randomness expansion schemes have been proposed against both classical and quantum adversaries. In the first case the malicious entity has information about the classical noise affecting the system, while in the second they might get information by entangling with the quantum memories of the device. These protocols require perfect random input seeds in order for the Bell inequality violation to hold. To obtain these, randomness amplification can be used, starting from a state of partial initial randomness. QRNGs based on these principles are still very demanding in terms of implementation complexity and are poor in performance [21].

A tradeoff between the high generation rate of trusted solutions and the security assurances of self-testing implementations is possible and it is referred to as semi-self-testing. QRNGs in general are composed of a source, that emits quantum states, and a measurement device, that detects such states. Instead of having both these modules perfectly characterized, like in the trusted scenario, there are some realizations in which only one of the two respects this requirement. Instead the other can be considered untrusted [2]. The possibilities are:

- Source-device-independent QRNG (SDI-QRNG), in which the source is considered untrusted and the measurement device is well characterized. They are usually based on the switching between different measurement settings, so that the adversary is not able to make any predictions about the output random sequence [22]. While this is a possibility, some other configurations exist, in which, by characterizing the measurement device, it is possible to lower bound the conditional min-entropy of the produced numbers. This is done without an active base measurement switch and still considering the source as untrusted. This case will be thoroughly analyzed in Sec. 2.2.

- Measurement-device-independent QRNG (MDI-QRNG), in which the measurement device is considered untrusted while the source is well characterized. The idea here is to check the measurement device by using randomly chosen states [23].

- Other configurations are possible for semi-self-testing. Notably there is one in which the source and the device are considered to occupy independent

two-dimensional quantum subspaces and the randomness of the process is certified using a dimension witness. Still the performance of these alternatives does not match the one of the two configurations mentioned above [21].

In conclusion, experiments show that semi-device-independent systems are a valid solution, allowing for generation rates close to those in the fully trusted case, while guaranteeing security properties that may be compliant with the requirements of critical applications, like cryptography [21].

1.1.4 QRNG Applications

There are several applications in which true randomness is needed. In this section some of the most relevant are described.

- Quantum key distribution protocols, are theoretically secure cryptographic mechanisms to exchange keys between parties. It is evident that the protocols become useless if the securely exchanged information can be predicted during the generation phase. It is also important to underline that these protocols are composed by more steps than the actual key exchange itself. These include phases like error correction and privacy amplification, which are fundamental to have an identical key between the parties, about which a possible adversary has no information. These additional steps also require randomness to be realized [24].

- Deterministic cryptographic protocols are nowadays at the core of cryptographic systems. They are based on the complexity of resolution of a discrete logarithm problem in Galois or elliptic curve groups. Some notable examples are Diffie-Hellman key establishment protocol and Rivest–Shamir–Adleman (RSA). Since the nature of the problem is deterministic, a great part of the security comes from the randomness of the numbers involved in such protocols. Moreover, an attack on a partial-PRNG of an Advanced Encryption Standard (AES) based commercial cryptographic system has been proven. Therefore, even though the cryptographic system is not based on any quantum mechanical principle, it may still be useful that the source of randomness does [24].

- Simulations are a fundamental aspect in many fields of science. To study a phenomenon it may be important to analyze many cases chosen uniformly at random. The most prominent example are Monte Carlo simulations, in which the solution of a complex problem is obtained by averaging many random instances. It is evident that in this field the focus is on obtaining a uniformly distributed sequence. PRNGs seem the best solution for the high generation rate and output string randomness, however there are several instances in

which long distance correlations are present inside their produced sequences. Therefore QRNGs should be considered also for simulations [15].

1.2 FPGA and QRNGs

As already mentioned, due to the imperfections in the realization of any of the possible QRNGs, it is mandatory to apply some kind of post-processing on the sequences produced. This is done to eliminate the classical correlations present, which pose an issue both on the uniformity and the security of the numbers obtained. This post-processing consists in a series of computationally heavy mathematical operations [24].

Several solutions have been proposed to tackle the complexity of these algorithms, starting from software based ones. A simple strategy would be to store the raw data inside a hard disk and implement the post-processing algorithm with a script written with some high-level programming language. In this case the speed at which random numbers are generated, refers to the speed that the software is able to generate random bits by processing the pre-loaded data batch. It is reported that this approach can reach rates of 100 Gbps. This method is referred to as offline and it is hardly practical in a system that continuously needs to be supplied with numbers. Moreover this definition of generation rate has to be taken with care. Even though the numbers seem high, it is not the rate at which it is actually possible to feed data in a free-running system that requires random number generation. In fact it is referred to as offline generation rate [9].

Online software implementations exist and notable examples are present in which the Toeplitz hashing (the post-processing technique that is used also in this work and that is explained in Sec. 3.2) is transformed into a FFT algorithm. This is done in order to increase the computational efficiency. This strategy has been applied with a Central Processing Unit (CPU), reaching generation rates of some Mbps, which is evidently not enough for most applications. Another possibility includes the execution of this type of algorithm with a Graphics Processing Unit (GPU), that exploits many parallel FFT threads to increase the generation rate. Even though the performance improvement is remarkable, reaching some Gbps, this may still not be enough [25].

Building on the idea of exploiting parallel computation, it is possible to adopt a hardware-based approach rather than a software-based one. In this field, FPGAs stand as the most flexible solution, allowing for easy prototyping while still maintaining hardware level performances. Therefore these systems are at the forefront of real-time randomness extraction for QRNGs, reaching generation rates compatible with the needs of applications like quantum cryptography, which are now in the order of some Mbps but are expected to increase considerably in the future [25].

Chapter 2

Quantum Random Number Generator Models and Experimental Setups

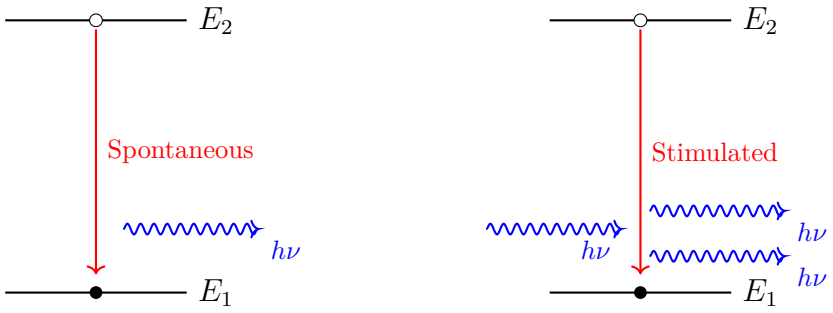
This chapter presents the mathematical model and the implementation of the QRNGs realized in this work, including a phase fluctuation-based approach in Sec. 2.1 and a source-device-independent-vacuum state-based solution in Sec 2.2. For the two technologies, both an online and an offline configuration are described. This has been done because the ADC available, compatible for direct connection to the evaluation board containing the FPGA, has a very low sampling rate. This leads to a drastic reduction of the useful bits of the converted voltage values, due to the inherent averaging performed on the signal during conversion. For this reason the online configuration has been implemented only to demonstrate a possible implementation. However the result discussion in Ch. 4 is based on data acquired with the offline configuration. This one is based on saving the data acquired from a high sampling rate ADC on a SD card. The storage device is then inserted in the evaluation board to perform the post processing.

2.1 Phase Fluctuations-Based QRNG

One of the paradigms of QRNG that has received more attention is the one based on the phase fluctuations of a laser emission. It can be demonstrated that the output field of a laser is influenced by the randomness of spontaneous emission photons and conforms a Gaussian distribution [26].

2.1.1 Spontaneous Emission and Stimulated Emission in a Laser Diode

Spontaneous and stimulated emission are processes in which a quantum mechanical system, composed of a ground and an excited state, performs an energetic transition that involves the emission of a photon. The former phenomenon implies the presence of an atom in the excited state that spontaneously decays to the ground state, due to the instability of the higher energy level. As a result a photon is emitted with the following characteristics: random polarization, random direction and random phase. Instead, stimulated emission involves the presence of an atom in the excited state, that decays to the ground state due to the interaction with an incoming photon. This one has the same energy as the separation between the ground and the excited states. The resulting emitted photon will have the same polarization, direction and phase as the one that has interacted with the system [19].



(a) Spontaneous Emission (b) Stimulated Emission

Figure 2.1: Schematics of spontaneous emission (a) and stimulated emission (b). Consider an atom with excited state of energy E_2 and ground state of energy E_1. In (a) the atom in the excited state decays spontaneously, emitting a photon with energy $h\nu = E_2 - E_1$. In (b) an incoming photon with energy $h\nu = E_2 - E_1$, interacts with the system, composed of an atom in the excited state, inducing a relaxation with consequent emission of a photon with the characteristics explained in the main text.

Stimulated emission is the fundamental process underlying the operation of a laser diode. In fact, by considering N_1 the number of atoms in the ground state and N_2 the number of atoms in the excited state, it is possible to achieve, with this process, the emission of a coherent and collimated beam of light by electrically pumping the diode, which means biasing it with a certain amount of electrical current, in order to achieve the so called population inversion ($N_2 > N_1$) [19].

However, as cited above, the focus of the QRNGs discussed in this section, is on the detection of phase fluctuations due to spontaneous emission. This is an hint on the fact that the laser diode used, must be operated near the threshold current in

order to produce a detectable optical signal but still obtaining mostly incoherent light, since the main mechanism involved in the photon production is spontaneous emission [27].

2.1.2 Phase Fluctuations-Based QRNG Model

In order to convert phase fluctuations into intensity fluctuations, which is the quantity that is actually measured, an interferometer is usually built. Notable examples include the use of a Mach-Zender Interferometer (MZI) [3] as well as more sophisticated strategies, such as phase-reconstruction [28]. The focus of this section will be centered around the mathematical model behind the architecture shown in Fig. 2.2 that allows for a simple and compact implementation of a QRNG [27].

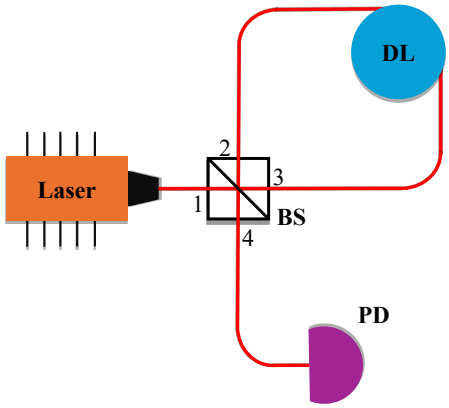


Figure 2.2: Optical setup for phase fluctuation-based QRNG. It is composed of a laser diode, biased at a current around its threshold value. The emitted light enters port 1 of a Beam Splitter (BS). The beams exit through port 3 and 4. Photons coming out from port 4 are directly collected by a photodetector (PD), while the ones coming out from port 3 go through the Delay Line (DL) and will be fed back to the BS through port 2. Adapted from [27].

The optical field that enters port 1 of the beam splitter can be written as:

$$E_{port1}(t) = Ae^{i[\omega t+\varphi(t)]} \tag{2.1}$$

with A the amplitude of the optical field, ω the optical center angular frequency and $\varphi(t)$ the phase fluctuation of the laser. The field entering port 2 of the beam splitter, is instead the sum of all the contributions coming from the circulations inside the delay line. In fact, light beams that exit from port 3 and enter the delay line, when they reach port 2, may get transmitted to the photodetector at port 4,

or may be injected again in the delay line by reflection to port 3. This behavior can be modeled by considering N circulations in the delay line and $N \to \infty$:

$$E_{port2} = A \sum_{k=1}^{N} \left(\frac{1}{\sqrt{2}} \right)^{k} \beta^{k/2} e^{i[\omega(t-k\Delta t)+\phi(t-k\Delta t)]} \tag{2.2}$$

where $k \in [1, N]$ is an integer, β is the overall effective gain induced by the components in the fiber loop and Δt is the time delay induced by the delay line. The intensity of the field resulting from the interference of E_{port1} and E_{port2}, that is detected by the photodetector, is given by the following expression:

$$\begin{aligned} I = \frac{1}{2}A^2 \sum_{k=0}^{N} \left(\frac{\beta}{2} \right)^{2} + A^2 \sum_{k=1}^{N} \left(\frac{\beta}{2} \right)^{k/2} [-\cos(k\omega\Delta t + \Delta\varphi_0^k)] + \\ + A^2 \sum_{k=1}^{N} \left(\frac{\beta}{2} \right)^{k/2} \left(\sum_{j=1}^{k-1} \left(\frac{\beta}{2} \right)^{j/2} \cos[(k-j)\omega\Delta t + \Delta\varphi_j^k] \right) \end{aligned} \tag{2.3}$$

with $k \in [0, N]$ and $j \in [1, k)$ integers, $\Delta\varphi_j^k = \varphi[(N-j)\Delta t] - \varphi[(N-k)\Delta t]$ the phase fluctuation between the circulation number $(N-j)$ and the circulation number $(N-k)$. This $\Delta\varphi_j^k$ is the Gaussian quantity due to spontaneous emission and by Eq. 2.3 the optical intensity is a superposition of this quantity; so, neglecting the DC component and measuring the fluctuations in the intensity, it is possible to conclude that a QRNG is being realized [27].

2.1.3 Implementation

The optical setup used for the realization of the phase fluctuation-based QRNG obviously replicates the structure around which the mathematical model explained above is built.

The schematics of the actual implementation are reported in Fig. 2.3 and are explained in depth below.

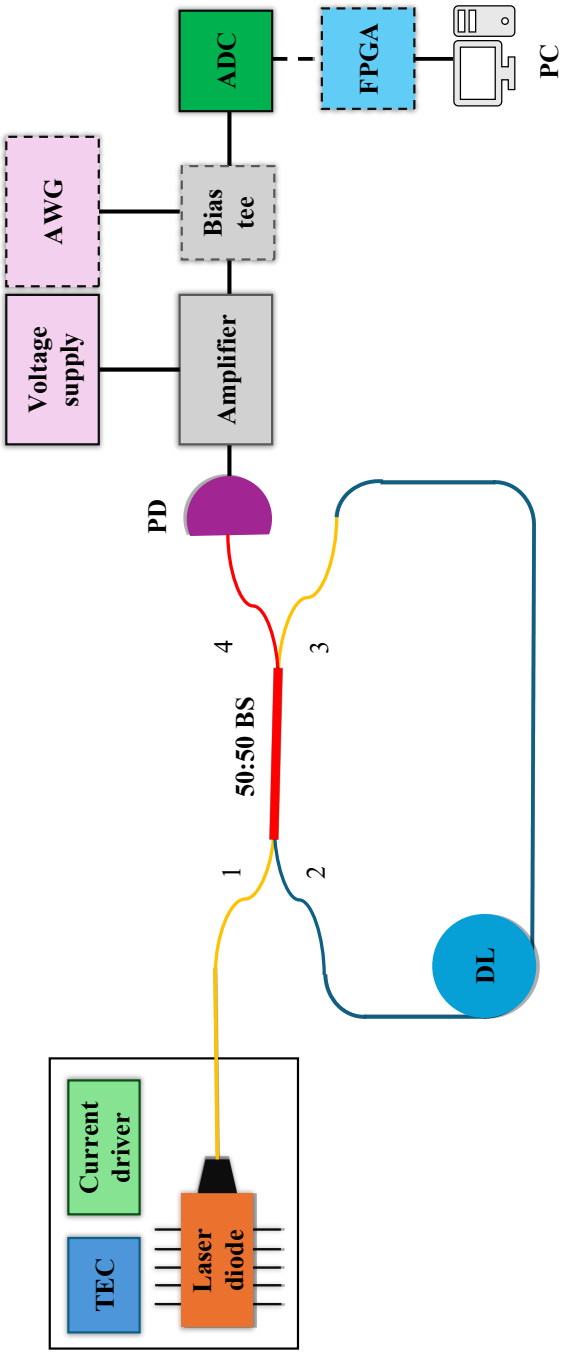


Figure 2.3: Phase fluctuation-based QRNG implementation. The components with a dashed outline are only present in the online implementation. TEC: Thermo-Electric Cooler; DL: Delay Line; BS: Beam Splitter; PD: Photodetector; AWG: Arbitrary Waveform Generator; ADC: Analog to Digital Converter.

The laser source is realized using a pigtailed butterfly packaged Distributed Feedback (DFB) laser diode, emitting at around 1539.7 nm, while operated around its threshold current of about 13.5 mA at 24°C, resulting in a linewidth of about 0.0653 nm. The device is specifically chosen to have the largest linewidth possible between the available laser diodes, in order for the photons to lose coherence in the least amount of time and space. For this reason an External Cavity Laser (ECL) has been initially considered but later discarded after characterization, since this type of lasers generally have a narrower emission spectrum. The diode is mounted on an ad-hoc component with an embedded Thermo-Electric Cooler (TEC), to regulate the temperature and the possibility to set the bias current.

The laser is coupled with an optical fiber and is connected to, referencing Fig. 2.3, port 1 of a 50:50 fiber coupler, that works as the beam splitter. The other input port (port 2), is connected to the delay line, implemented with a simple optical fiber wounded on a reel. It has been estimated that 5 m of delay line are enough for the experiment. The other side of this optical fiber is connected to port 3 of the coupler. The last output port is instead connected to an AC coupled photodetector, providing a RF output. It is useful for the evaluation of the fluctuations of the obtained electromagnetic field, without measuring the DC component, that provides no information to the purpose of the experiment. The signal is sent to an Analog to Digital Converter (ADC), to obtain the results in digital form. Based on the necessities set by the particular ADC used, the signal goes before through an amplifier and then either goes through a biasing stage or it is directly sent to the digital conversion component.

As previously mentioned, a free running configuration (online) has been implemented, in which the ADC is directly connected to the evaluation board containing the FPGA. This conversion stage is characterized by a low sampling rate of 1 MHz and unipolar input range. This means that the voltages obtained need to be within a positive range of values between 0 V and 3.3 V, which is the maximum possible convertible voltage. For this purpose a bias tee has been employed to set the average value of the signal at the midpoint of the detectable range. The amount of bias is set with an Arbitrary Waveform Generator (AWG) outputting a DC signal at the desired voltage. The AWG connects to the bias tee via an RF coaxial cable.

The offline configuration is based instead on the acquisition of a batch of data, that is later sent to the FPGA via a SD card to be processed. In this case the ADC is much more performant, reaching sampling rates of 100 MHz and supporting bipolar inputs. For this reason the bias tee is not necessary in this configuration.

In any case, with the signal either biased or directly coming from the photodetector, an amplification stage is needed to have the signal occupying most of the resolution range of the ADC used. To obtain this, a coaxial amplifier is employed, connected to a voltage supply of 15 V and providing about 10 dB of amplification.

Finally, the signal reaches the conversion stage, which uses either a low or high

sampling rate ADC, depending on the configuration.

The last stage is composed of a PC: in the online configuration the data is processed in real-time by the FPGA and the generated numbers are displayed on the PC using a serial communication. This is established through a simple USB connection between the evaluation board and the computer. In the other case the data converted by the ADC are directly stored inside the computer hard disk using a LabView script to set the data acquisition configurations.

2.2 Vacuum State Heterodyne Measurement-Based QRNG

Another way to implement a QRNG is based on the uncertainty of continuous quantum observables, that are the amplitudes of the quadratures of the vacuum state. Many implementations of QRNGs based on this phenomenon exploit homodyne measurements [4], [22].

2.2.1 Vacuum State

The vacuum state is the quantum state associated with the lowest possible energy and generally contains no particles. This particular system has interesting characteristics that are at the core of the working principles of the QRNG. In this section they will be briefly outlined.

It can be demonstrated that there is an equivalence between an harmonic oscillator and a light wave, so it is possible to associate some known properties to a quantized electromagnetic field, including:

1. The energy of the electromagnetic wave is quantized, according to the relation:

$$E_n = (n + \frac{1}{2})\hbar\omega \tag{2.4}$$

   With n the index of the energy level and $\hbar\omega$ the energy quantum.

2. The position and momentum variables satisfy the Heisenberg uncertainty principle:

$$\Delta x \Delta p_x \geq \frac{\hbar}{2} \tag{2.5}$$

By defining the dimensionless quadratures:

$$X_1(t) = \left(\frac{\omega}{2\hbar}\right)^{1/2} x(t) \tag{2.6}$$

$$X_2(t) = \left(\frac{1}{2\hbar\omega}\right)^{1/2} p_x(t) \tag{2.7}$$

It can be seen how these depend to the generalized position and momentum coordinates. Therefore, it is possible to relate the Heisenberg uncertainty principle to the quadrature uncertainties, as follows:

$$\Delta X_1 \Delta X_2 = \frac{1}{2\hbar}\Delta x \Delta p_x \geq \frac{1}{4} \tag{2.8}$$

it can be concluded that the energy associated with the vacuum state is the zero-point energy $\hbar\omega/2$ and that the vacuum field is characterized by an amplitude equal to zero but non-zero field quadratures uncertainty, which are the fluctuations measured in the QRNG implementation [19].

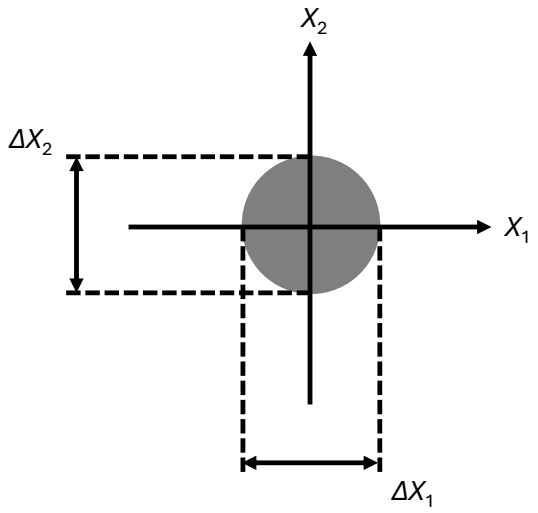


Figure 2.4: Vacuum field. Adapted from [19].

2.2.2 Homodyne and Heterodyne Detection

Homodyne detection is a common strategy for continuous variable quantum systems measurements. It is based on measurement operators, that are projectors over one of the two quadrature bases of the signal, which in Dirac notation are $|I\rangle\langle I|$ and $|Q\rangle\langle Q|$. This allows to measure one of two quadratures with complete precision [29].

Heterodyne detection, instead, consists of projection onto coherent states. This means that the measurement operator is $|\alpha\rangle\langle\alpha|$, allowing for the simultaneous

measurement of both the quadratures of the signal, with the consequent doubling of the mutual information. This comes at the cost of the introduction of noise, given by the non-commuting nature of the quadratures in question, as explained in Sec. 2.2.1 [29].

The way the two quadratures are extracted from the signal is no different to that used in any coherent detection scheme for optical communications. Consider the vacuum state signal and a local oscillator (LO) and consider the scheme in Fig. 2.5. It is possible to express the vacuum field with its two quadratures as:

$$E_{vac}(t) = E_I(t) + iE_Q(t) \tag{2.9}$$

It is fundamental to underline that the signal obtained by any photodetector is a current that gets transformed into a voltage using a transimpedance amplifier. This current is proportional to the optical power of the signal entering the detector, which can be expressed as $P(t) = E(t)^2$. It is easy to compute the optical power of each signal impinging on the photodetectors, since the fields values are known. Considering detector D1 in Fig. 2.5 and considering i_j as the current obtained from the signal on port j = 1,2:

$$\begin{aligned} i_1(t) \propto |\vec{E}_{vac} + \vec{E}_{LO}|^2 = |\vec{E}_{vac}|^2 + |\vec{E}_{LO}|^2 + 2Re\left\{\vec{E}_{vac} \cdot \vec{E}^*_{LO}\right\} = \\ = P_{vac} + P_{LO} + 2\sqrt{P_{LO}} \cdot E_I(t) \end{aligned} \tag{2.10}$$

$$\begin{aligned} i_2(t) \propto |\vec{E}_{vac} - \vec{E}_{LO}|^2 = |\vec{E}_{vac}|^2 + |\vec{E}_{LO}|^2 - 2Re\left\{\vec{E}_{vac} \cdot \vec{E}^*_{LO}\right\} = \\ = P_{vac} + P_{LO} - 2\sqrt{P_{LO}} \cdot E_I(t) \end{aligned} \tag{2.11}$$

Thanks to the balanced photodetector that computes the difference between the two currents, the output current is proportional to one of the quadratures of the vacuum [30]. The expression is:

$$i(t) = i_1(t) - i_2(t) \propto 4\sqrt{P_{LO}}E_I(t) \tag{2.12}$$

For what concerns $E_Q(t)$ the computations are analogous, so they will not be reported.

2.2.3 Vacuum State Heterodyne Measurement-Based QRNG Model

The model for the quantum random number generator based on vacuum state fluctuations is now discussed. It is important to underline that the main purpose of this implementation is to design a system, using simple optical components, that can be categorized as source-device-independent [31].

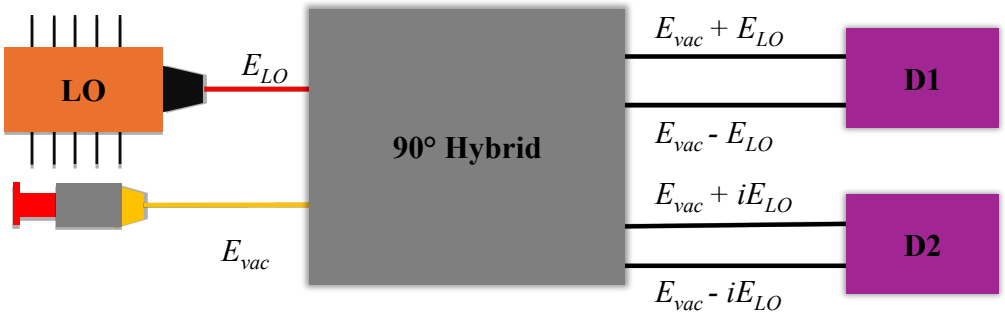


Figure 2.5: Heterodyne detection scheme: a Local Oscillator (LO) and the randomness source signal (in this case a closed optical fiber, to realize the vacuum state) are connected to a 90° optical hybrid. This component gives 4 outputs, which are the sum and the difference between the signal and the LO as well as the sum and difference between the signal and a 90° phase shifted version of the LO. The outputs are then sent to two balanced photodetectors (D1 and D2). Adapted from [30].

In a context in which the possibility of an attacker to exploit the imperfections in the physical realization of systems, that are used to produce numbers for cryptographic keys, is crucial, it is fundamental to be able to estimate the maximum amount of randomness that can be extracted, in the presence of these imperfections (generally referred to as side-information). This quantity is called conditional min-entropy and a more rigorous mathematical description of it as well as its consequences in the context of randomness extraction will be given in Sec. 3.2.

This QRNG implementation allows for the lower-bounding of the conditional-min entropy. So, there is the possibility of having an attacker controlling the device's source and manipulating it with the most favorable strategy for them to be able to predict the random numbers produced, while still being able to obtain a fast generation rate of secure numbers. As previously mentioned the model is based on a heterodyne measurement of both the quadratures of the vacuum state [31].

The heterodyne measurement corresponds to the following Positive Operator-Valued Measurement (POVM):

$$\hat{\Pi}_\alpha = \frac{1}{\pi} |\alpha\rangle \langle\alpha| \tag{2.13}$$

where $|\alpha\rangle$ is a coherent state with complex amplitude α.

Calling ρ_A the density matrix of the field, the output of the heterodyne measurement is given by the random variable X:

$$X = \{q, p\} \tag{2.14}$$

with $q = \mathrm{Re}(\alpha)$ the real part and $p = \mathrm{Im}(\alpha)$ the imaginary part of the coherent state, which are distributed according to the Husimi function:

$$Q_{\rho_A}(\alpha) = \mathrm{Tr}[\hat{\Pi}_\alpha \rho_A] = \frac{1}{\pi} \langle\alpha| \rho_A |\alpha\rangle \tag{2.15}$$

which gives a probability distribution that represents the quantum state in the space of quadratures [31].

However, in a real scenario, measurements done with the heterodyne detection are discrete and depend on the resolutions δ_q and δ_p of the measurement apparatus, that is used to acquire the quadrature values. For this reason the POVM has to be expressed in its discretized form:

$$\hat{\Pi}^\delta_{m,n} = \int_{m\delta_q}^{(m+1)\delta_q} dq \int_{n\delta_p}^{(n+1)\delta_p} dp \hat{\Pi}_{q+ip} \tag{2.16}$$

which can be substituted inside the Husimi function formula to obtain the discretized version of the probability distribution:

$$Q^\delta_{\rho_A}(m,n) = \mathrm{Tr}[\hat{\Pi}^\delta_{m,n} \rho_A] = \int_{m\delta_q}^{(m+1)\delta_q} dq \int_{n\delta_p}^{(n+1)\delta_p} dp Q_{\rho_A}(q+ip) \tag{2.17}$$

From this quantity it is possible to extract the classical min-entropy:

$$H_{min}(X_\delta) = -\log_2 \left[\max_{m,n} Q^\delta_{\rho_A}(m,n) \right] \tag{2.18}$$

However this quantity serves a purpose only in a fully-trusted scenario. In case the side-information has to be taken into account it is necessary to use the conditional min-entropy $H_{min}(X|\mathcal{E})$. It is possible to demonstrate that for any POVM $\{\hat{\Pi}_x\}_{x\in X}$ and for a quantum state τ_A in the Hilbert space $\mathscr{H}_A$, the quantum conditional min-entropy is lower bounded by the quantity:

$$H_{min} = -\max_{x\in X, \tau_A \in \mathscr{H}_A} \log_2 \left(\mathrm{Tr} \left[\hat{\Pi}_x \tau_A \right] \right) \tag{2.19}$$

In the case of a heterodyne measurement, this lower bound becomes:

$$H_{min}(X_\delta|\mathcal{E}) \geq -\max_{\{m,n,\tau_A\}} \log_2 \left(\mathrm{Tr}\left[\hat{\Pi}^{\delta}_{m,n}\tau_A\right]\right) = \log_2 \frac{\pi}{\delta_q\delta_p} \tag{2.20}$$

Therefore, it is possible to conclude that even if an adversary forges the optimal side information $\mathcal{E}$, they are not enable to guess the heterodyne outcome with a probability larger than $(\delta_q\delta_p)/\pi$. So the scheme allows for a theoretically proven secure generation of random numbers by the simple computation of a quantity, that depends only on the resolution of the measurement devices and that does not change as long as the measurement apparatus is considered trusted.

2.2.4 Implementation

The optical setup for the realization of the vacuum fluctuation-based QRNG is implemented for a typical heterodyne measurement configuration together with elements useful for the calibration part as it is shown in [31].

The schematics of the actual implementation are reported in Fig. 2.6 and explained in detail below.

The local oscillator is realized using a pigtailed ECL butterfly packaged laser diode, emitting at 1550 nm with current bias of 300 mA and 25 °C of temperature. These values are set with an external TEC and current generator. The laser is fiber coupled to an electronic Variable Optical Attenuator (VOA), which has the purpose of modifying the optical power of the local oscillator during the calibration phase. This component is piloted with an external electronic driver, using Labview scripts to control the calibration phase workflow.

The VOA is connected on the other side with the 90° hybrid, to implement the heterodyne measurement configuration. The other input port of the optical hybrid consists of a fiber connector, kept close with a dust cap. The output ports of the hybrid are connected to two different balanced AC coupled photodetectors, characterized by a 100 MHz bandwidth. In the online configuration a bias tee might be needed before the ADC, that is directly connected to the FPGA, for both the outputs of the balanced photodetectors. The processed numbers are shown on the PC thanks to the serial connection with the board.

In the offline case the RF outputs are connected each to a channel of the high-sampling-rate ADC. In order to save the files relative to the two different data batches, again a Labview script has been implemented, to select the total amount of data needed and the desired range of voltages to consider for data conversion.

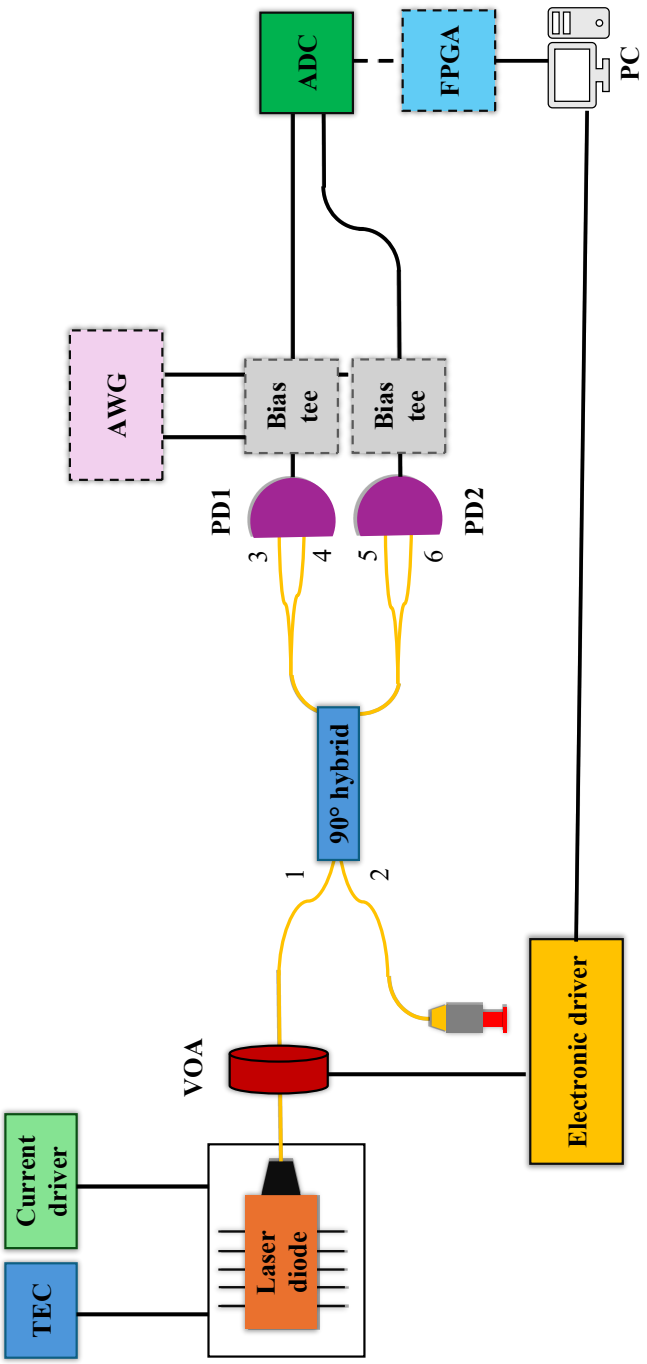


Figure 2.6: Vacuum State Heterodyne Measurement-Based QRNG implementation. The components with a dashed outline are only present in the online implementation. TEC: Thermo-Electric Cooler; VOA: Variable Optical Attenuator; PD: Photodetector; AWG: Arbitrary Waveform Generator; ADC: Analog To Digital Converter.

Chapter 3

High-Rate Randomness Extraction Using a FPGA

In this chapter an overview on FPGAs and a description of the circuit design workflow on them is presented in Sec. 3.1. Next, a mathematical description of randomness extraction is in Sec. 3.2. Finally, an analysis on the extraction algorithm on FPGA and a description of the circuit realizing it are reported in Sec. 3.3, while the complete circuit for both the offline and the online configuration is described in Sec. 3.4.

3.1 FPGA: Overview and Design

FPGAs are a type of programmable logic device whose internal logic and interconnections can be reconfigured by the user. This flexibility makes them especially well suited for prototyping and niche applications, while still delivering hardware-level performance. They are field-programmable, meaning that they can be programmed by the user, after they have been manufactured by the producer. This is the great advantage with respect to Application Specific Integrated Circuits (ASICs), that instead are produced in large volumes but cannot be modified once completed. Moreover FPGAs are reprogrammable, meaning that any error present in the design can be corrected and any update can be implemented in the circuit, without any cost, allowing for easy reuse of a single system. For these reasons FPGAs are used in various fields, like signal and image processing. In addition, the possibility to design circuits to be able to perform certain algorithms much faster and much more efficiently than any software counterpart, is present. This is possible because of the hardware-level parallelism, achievable by accurate design of the circuit for the needed application. By hardware-level parallelism it is meant that a circuit can be designed, in order for it to execute mathematical operations, by dividing

the computational workload between different parts of the circuit. Consequently a speedup of time execution with respect to a serial and software-based solution is obtained [32].

3.1.1 FPGA Structure

In principle, programmable logic is obtained by the decision to wether or not close a switch that connects some input data and a wired-OR line. In fact, by complementing the data line, it is easy to see that AND and OR logic functions can be implemented and are enough to realize any possible algorithm. This is how some programmable logic devices are implemented (like PROM, PAL and PLA) [33].

On the other hand, an FPGA is a much more sophisticated device, as it can be seen in Fig. 3.1. It leverages particular components in order to realize the characteristic functionalities. The main ones that can be found are [33]:

- Look-Up Tables (LUTs): the basic logic elements. They consist of blocks that realize the truth tables that are mapped onto them.
- Configurable Logic Blocks (CLBs): they are generally composed of LUTs, Flip Flops (FFs) and interconnection resources. This composition of elements allows for the realization of logical operations and data flow. They are configured based on the need of the implemented circuit.
- Input/Output Blocks (IOBs): this is the way the FPGA communicates with the external world. In fact they are the core components that allow for the processing of data, which might come from the environment, as well as utilization of external clock signals and acquisition, from the external, of the data processed by the FPGA.
- Routing resources: fundamental for the efficient propagation of signals between the different blocks. Routing or switch matrices are used for this purpose.
- Clock networks: Dedicated resources for the propagation of clock signals.
- Clock Management Module (CMM): handles distribution, synchronization and control of the clock signal.
- Memory elements: necessary for data storing inside the FPGA, including FFs, registers and ultimately Block RAM (BRAM) and ROMs.
- Secondary elements: like power supply circuits, I/O buffers and memory management circuits.

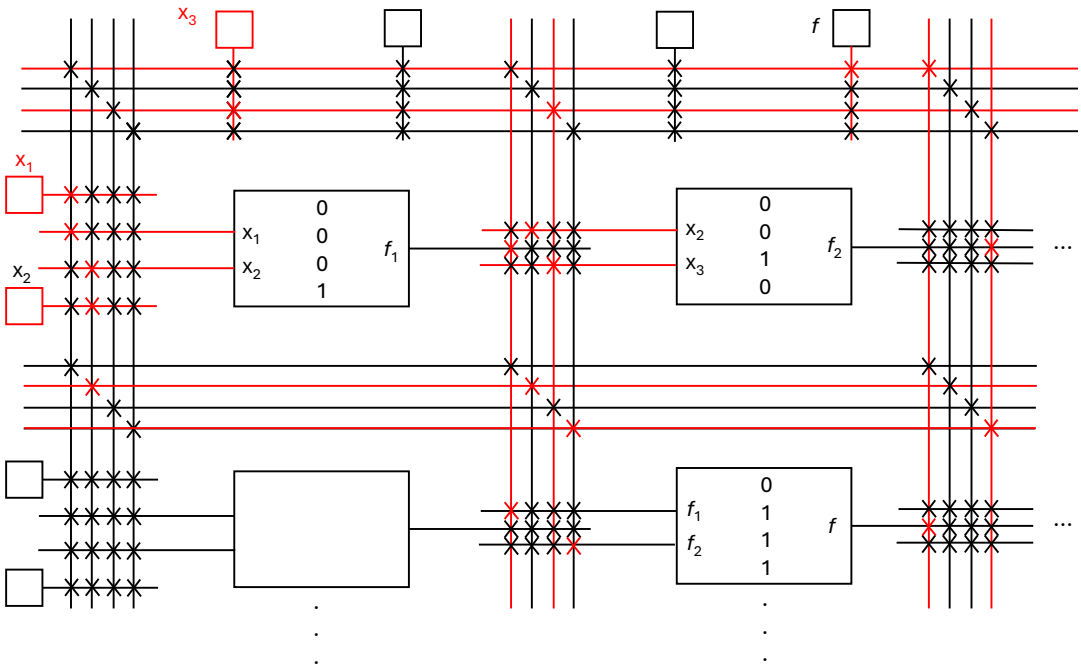


Figure 3.1: Example of the FPGA architecture schematic. Adapted from [33].

Obviously these resources are not unlimited in an actual FPGA device and how the utilization of these is usually addressed, is by means of slices. These are composed of LUTs and FFs, and are one of the resources contained in a CLB [34]. The number of the two composing elements inside a single slice is determined by the family and the producer of the FPGA device.

3.1.2 FPGA Circuit Design Workflow

Circuits that are used on FPGAs are conveniently described using Hardware Description Languages (HDLs), with the most used being VHDL and Verilog. However these languages have to be translated in the actual circuit that gets loaded on the FPGA [33]. This operation is done with software programs such as Quartus Prime (for Intel devices) and Vivado (for AMD devices).

In this work, the latter has been used. The specific workflow of Vivado for a FPGA project can be seen in Fig. 3.2, but generally the main steps are the same for any software.

Going in detail on the most relevant [35]:

1. The project starts from the design sources. In the simplest case it is VHDL code describing the circuit, which will eventually be converted to a Register

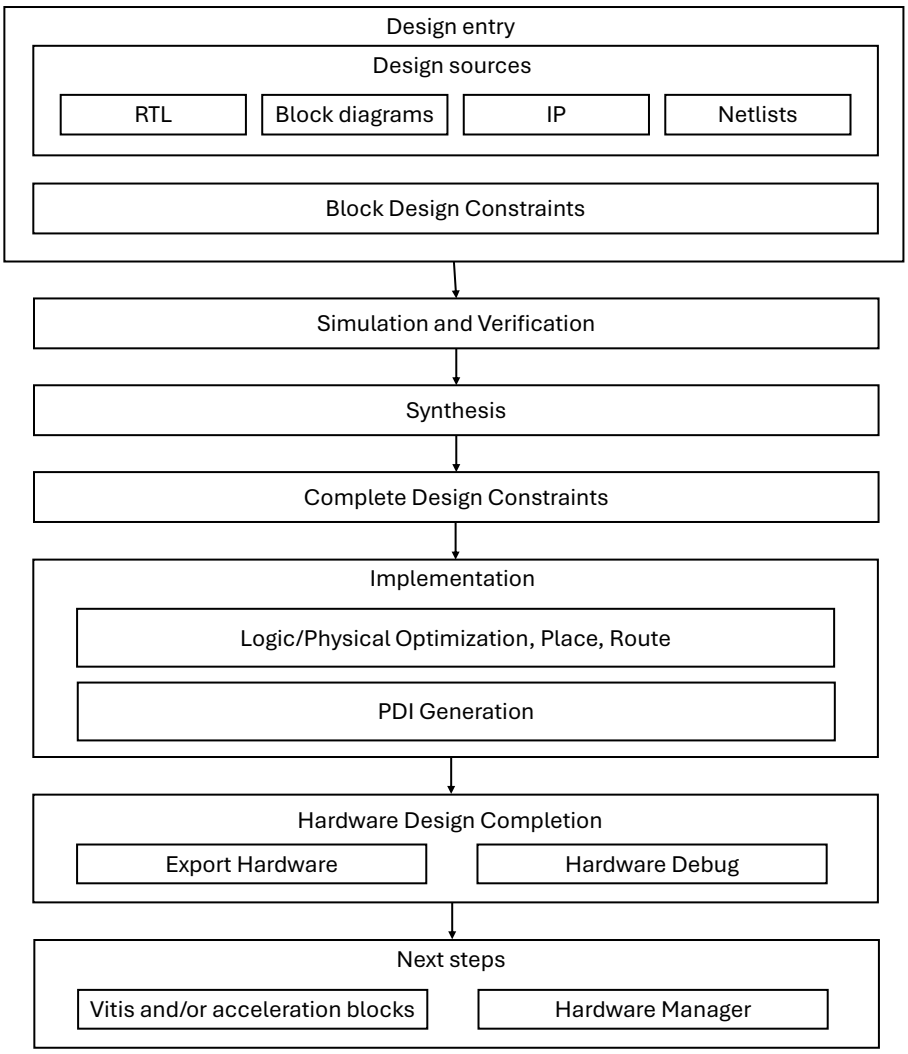


Figure 3.2: FPGA circuit design flow. Adapted from [35].

Transfer Level (RTL) model. This is a logical and hierarchical representation, including logical blocks interconnected and connected with the input and output interfaces. Notice that this model is still far from the actual physical representation of the circuit. As it can be seen by Fig. 3.2, this is not the only possible design source.

2. The design sources are associated with a constraint file, which determines aspects like pin assignment (decision of which of the actual pin of the FPGA, a signal in the circuit design is connected to), clock definitions, timing constraints, I/O standards (like the voltage level at which the assigned pin operates), etc. These properties are fundamental for the actual implementation of the circuit on the physical chip and its correct functioning.

3. Simulations can be run from a logical level directly on the provided VHDL code as well as after RTL elaboration, synthesis and implementation. These simulations consist in design methodology checks, logic simulation, timing and power analysis.

4. Synthesis of the RTL design, which consists in the translation from the logical abstract representation, to the gate-level representation and the consequent interconnections, under the form of a netlist. Also optimizations are performed in order to have the most efficient result.

5. Implementation (place and route) which maps the netlist onto the available resources on the specific target FPGA device and determines the routing between the placed elements, in order for the signals functionalities to be compliant with the VHDL design.

6. The bitstream is generated, which is a file containing all the information needed to configure the FPGA device and that will be loaded into it. At this point the device can be directly programmed. Another possibility is to export the hardware platform, in order for it to be used on other software programs (like Vitis, which allows for CPU programming of System on Chip devices containing a processor and an FPGA).

3.2 Randomness Extraction

The following section focuses on the mathematical aspects concerning randomness extraction, which is the procedure needed to obtain information-theoretically provable random bits [6].

3.2.1 Binary Distributions and Security Parameter

Consider a binary distribution, which is a statistical characterization of the values 0 and 1 within a binary sequence, that describes how frequently each bit value occurs. The ideal distribution to obtain in any application that requires a RNG is the discrete uniform distribution. Considering a binary string of length p, denoted by $\{0,1\}^p$, the discrete uniform distribution is referred to as U_p. It defines a sequence

$X \in \{x_1, x_2, \ldots, x_n\}$ for which the probability $P(X = x_i)$ of the sequence being equal to any possible sequence in the set is:

$$P(X = x_i) = \frac{1}{n}, \forall i = 1,2,\ldots,n \tag{3.1}$$

Therefore each sequence is equiprobable. This behavior is ideal and no RNG can achieve this distribution, however it is possible to define some metrics to compare how close a sequence is to this uniform distribution [36].

Let's consider two probability distributions X and Y, over the same domain T, it is possible to say that these are $\varepsilon - close$ if the statistical distance between them is bounded by ε:

$$\begin{aligned} ||X - Y|| &= \max_{V \subseteq T} |\sum_{v \in V} (\text{Prob}[X = v] - \text{Prob}[Y = v])| = \\ &= \frac{1}{2} \sum_{v \in T} |\text{Prob}[X = v] - \text{Prob}[Y = v]| \leq \varepsilon \end{aligned} \tag{3.2}$$

with the factor 1/2 to normalize the statistical value so that it falls into [0,1]. Notice that this quantity is composable and it is of particular interest, because it states that the probability distributions X and Y are indistinguishable except for a small probability factor ε [6].

3.2.2 Extractors

Before defining the extractors, it is critical to first define the min-entropy of a probability distribution.

Consider a probability distribution X on $\{0,1\}^n$, the min-entropy is defined as:

$$H_\infty(X) = -\log_2 \left(\max_{v \in \{0,1\}^n} \text{Prob}[X = v] \right) \tag{3.3}$$

This quantity represents the maximum possible amount of randomness that is present in a probability distribution and it is of fundamental importance in the following definition of the extractor. A $(k, \varepsilon, n, d, m)$-extractor is a function defined as:

$$\text{Ext} : \{0,1\}^n \times \{0,1\}^d \rightarrow \{0,1\}^m \tag{3.4}$$

such that for every probability distribution X on $\{0,1\}^n$, with min-entropy $H_\infty(X) \geq k$, the probability distribution of the extracted string $\text{Ext}(X, U_d)$ is ε-close to the uniform distribution on $\{0,1\}^m$. In other words a nearly uniform distribution of length m can be obtained from a certain random string of length n by employing a seed of length d. A much handier function can be used for the purposes of this work and it is called strong-extractor. A $(k, \varepsilon, n, d, m)$-strong-extractor $\text{Ext}(X, U_d)$

is an extractor such that the probability distribution $\text{Ext}(X, U_d) \circ U_d$ is ε-close to the uniform distribution on $\{0,1\}^{m+d}$. Its key advantage is that the seed can be reused for new extractions, with a penalty of ε on the security parameter [6].

3.2.3 Universal Hash Functions

Hash functions map a certain set A into a set B and it is generally assumed that the dimension of the starting set is larger than that of the final set, meaning $|A| > |B|$.

A family of hash functions $\mathcal{H}$, mapping A to B, is said to be universal$_2$ when:

$$\text{Prob}_{h\in\mathcal{H}}\left\{h(x) = h(y)\right\} \leq \frac{1}{|B|} \tag{3.5}$$

for all $x \neq y \in A$. The subscript "2" in universal$_2$ is present to underline that the behavior of $\mathcal{H}$ is constrained to only pairs of elements of A [37].

Toeplitz Hashing

Now consider a certain family of hashing methods that uses random binary matrices. Consider a $m \times n$ Booelan matrix, called T and consider a message of n bits, called M; the hashing algorithm is the Boolean multiplication between the matrix and the column vector composed of the bits of the message:

$$h : T \cdot M \tag{3.6}$$

where the result is a binary string of m bits. It can be demonstrated that this type of hashing function h, is of the universal$_2$ class [8].

Now consider that the matrix T is the Toeplitz matrix, which is characterized by having each left-to-right diagonal fixed. This means that if $k - i = l - j$ for any indices $1 \leq i, k \leq m$, $1 \leq j, l \leq n$, then $T_{i,j} = T_{k,l}$. An example of a 6×8 Toeplitz matrix can be seen in Eq.(3.7), where each entry t_i is a binary number.

$$T = \begin{bmatrix} t_0 & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \\ t_{-1} & t_0 & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \\ t_{-2} & t_{-1} & t_0 & t_1 & t_2 & t_3 & t_4 & t_5 \\ t_{-3} & t_{-2} & t_{-1} & t_0 & t_1 & t_2 & t_3 & t_4 \\ t_{-4} & t_{-3} & t_{-2} & t_{-1} & t_0 & t_1 & t_2 & t_3 \\ t_{-5} & t_{-4} & t_{-3} & t_{-2} & t_{-1} & t_0 & t_1 & t_2 \end{bmatrix} \tag{3.7}$$

This kind of matrices does not have any particular advantages for what concerns the hashing algorithms itself, but rather it is very convenient to use them because they can be constructed simply by knowing the first row and the first column of the matrix. It comes immediately that to construct a $m \times n$ matrix, it is only necessary to know $m + n - 1$ bits [8].

However one might argue that the amount of bits needed for a single hashing computation is still higher than the length of the output string. This means that no net randomness can be extracted if the universal hashing is directly used for randomness extraction. To overcome this problem it is necessary to demonstrate that the scheme constructs a strong extractor. Conveniently, it can be demonstrated that any extractor constructed from a universal hash function is a strong extractor, with the Leftover Hash Lemma [6].

It states that, given $\mathcal{H} = \{h_1, h_2, ..., h_{2^d}\}$, a universal$_2$ hash family, mapping from $\{0,1\}^n$ to $\{0,1\}^m$ and X a probability distribution on $\{0,1\}^n$ with $H_\infty(X) \geq k$, it follows that for $x \in X$ and $h_y \in \mathcal{H}$ where $y \in U_d$, the probability distribution formed by $h_y(x) \circ y$ is ε-close to U_{m+d}, with $\varepsilon = 2^{(m-k)/2}$. Therefore, it forms a $(k,2^{(m-k)/2}, n, d, m)$-strong-extractor. It can be said that, given some raw data of size n, a min-entropy of k and a security parameter of ε, the length of the extracted string will be $m = k - 2\log_2(\frac{1}{\varepsilon})$ [6].

In other words, if it is possible to estimate the min-entropy of the QRNG system, it is easy to set the proportion between the rows and the columns of the Toeplitz matrix and to perform randomness extraction to obtain a theoretically-proven secure binary string.

3.3 Toeplitz Randomness Extractor

As already mentioned, the post-processing section is based on the binary multiplication between a Toeplitz matrix and a vector of raw data. To efficiently implement this mathematical operation, a three-stage pipeline circuit has been implemented, following the scheme proposed in [11]. Before delving in the explanation of the stages of the pipeline and their composing elements, the section below describes the computational strategy used to efficiently produce a dot product of a row and a column.

3.3.1 Matrices Multiplication Strategy

Let T be an $m \times n$ Toeplitz matrix. As mentioned before, such matrices have fixed left-to-right diagonals. Consequently, its structure is the following:

$$T = \begin{bmatrix} t_m & t_{m+1} & \cdots & t_{m+n-2} & t_{m+n-1} \\ t_{m-1} & t_m & \cdots & t_{m+n-3} & t_{m+n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ t_2 & \vdots & \ddots & t_n & t_{n+1} \\ t_1 & t_2 & \cdots & t_{n-1} & t_n \end{bmatrix} \tag{3.8}$$

Consider also a binary vector of n bits D:

$$D = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{3.9}$$

The extraction process consists in the dot product of each of the rows of T with the vector D. Consider the resulting vector R, of dimension m:

$$R = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix} = \begin{bmatrix} t_m & t_{m+1} & \cdots & t_{m+n-1} \\ t_{m-1} & t_m & \cdots & t_{m+n-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_1 & t_2 & \cdots & t_n \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} \tag{3.10}$$

Each of the elements r_i, with $i = 1,2,\ldots,m$ of R, has the following expression:

$$r_i = \sum_{j=1}^{n} t_{m-i+j}\, d_j, \quad i = 1,\ldots,m \tag{3.11}$$

which is the typical expression of any matrices product. By defining an integer number G and choosing it in such a way that the ratio n/G is an integer, it is possible to divide the $m \times n$ Toeplitz matrix in n/G sub-matrices of dimension $m \times G$. Each of these are multiplied by sub-vectors of D, composed of G bits. The partial results coming from these sub-operations are then accumulated to obtain the final result:

$$\begin{bmatrix} t_m & t_{m+1} & \cdots & t_{m+G-1} \\ t_{m-1} & t_m & \cdots & t_{m+G-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_2 & t_3 & \cdots & t_{G+1} \\ t_1 & t_2 & \cdots & t_G \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{G-1} \\ d_G \end{bmatrix} + \begin{bmatrix} t_{m+G} & t_{m+G+1} & \cdots & t_{m+2G-1} \\ t_{m+G-1} & t_{m+G} & \cdots & t_{m+2G-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_{G+2} & t_{G+3} & \cdots & t_{2G+1} \\ t_{G+1} & t_{G+2} & \cdots & t_{2G} \end{bmatrix} \begin{bmatrix} d_{G+1} \\ d_{G+2} \\ \vdots \\ d_{2G-1} \\ d_{2G} \end{bmatrix} +$$

$$+ \cdots + \begin{bmatrix} t_{m+n-G} & t_{m+n-G+1} & \cdots & t_{m+n-1} \\ t_{m+n-G-1} & t_{m+n-G} & \cdots & t_{m+n-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n-G+2} & t_{n-G+3} & \cdots & t_{n+1} \\ t_{n-G+1} & t_{n-G+2} & \cdots & t_n \end{bmatrix} \begin{bmatrix} d_{n-G+1} \\ d_{n-G+2} \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{m-1} \\ r_m \end{bmatrix} \tag{3.12}$$

It is easy to verify that this equality holds, since, considering the first element of the resulting vector r_1:

$$r_1 = \sum_{j=1}^{n} t_{m-1+j} d_j \tag{3.13}$$

the same expression is obtained by summing each partial result:

$$r_1 = \sum_{j=1}^{G} t_{m-1+j} d_j + \sum_{j=G+1}^{2G} t_{m-1+j} d_j + \cdots + \sum_{j=n-G+1}^{n} t_{m-1+j} d_j \tag{3.14}$$

since the overall sum spans again from 1 to n. This result holds for every element of R.

The idea is to have a circuit that is able to compute the result of each sub-matrix and sub-vector in a single clock cycle and so to compute the final result in n/G clock cycles. Moreover, notice that up to now the fact that the matrix employed is a Toeplitz one has no advantage.

The complete algorithm together with a clear explanation on the benefits obtained using a Toeplitz matrix are explained in the following section.

3.3.2 Data Extraction Algorithm

A flow chart of the data extraction algorithm implemented is shown in Fig. 3.3

It works as follows:

1. $m + n - 1$ random bits are given. They represent the constructing elements of the Toeplitz matrix and are saved in an array called t. Two numbers are set to zero, with i representing the current computational cycle and Z_0 representing the first partial result that is accumulated in the successive cycles. Obviously, it has to be set to 0 at the start of the algorithm.

2. A total of G sub-vectors of length m are obtained from t by considering $m + G - 1$ bits of the said vector. As already mentioned, this amount of numbers is enough to represent a sub-matrix of dimensions $m \times G$. By extrapolating G sub-vectors from it in a sliding window fashion, it can be easily seen that each element $T_{i,g}$ is a column of the Toeplitz sub-matrix. This happens because of the peculiar structure of the Toeplitz matrix.

   To further emphasize this concept, refer again to Eq. 3.8 and notice how successive columns change only by one element.

3. Now the g^{th} column of the sub-matrix is multiplied with the g^{th} bit of the raw data vector that is being processed and the result is saved in the element $U_{i,g}$. Again notice that each computation cycle a sub-vector of length G of raw data is being processed.

4. To effectively complete the row-by-column sub-matrix multiplication, each of the G partial results are summed with a bit-by-bit XOR operation and the result is saved as U_i.

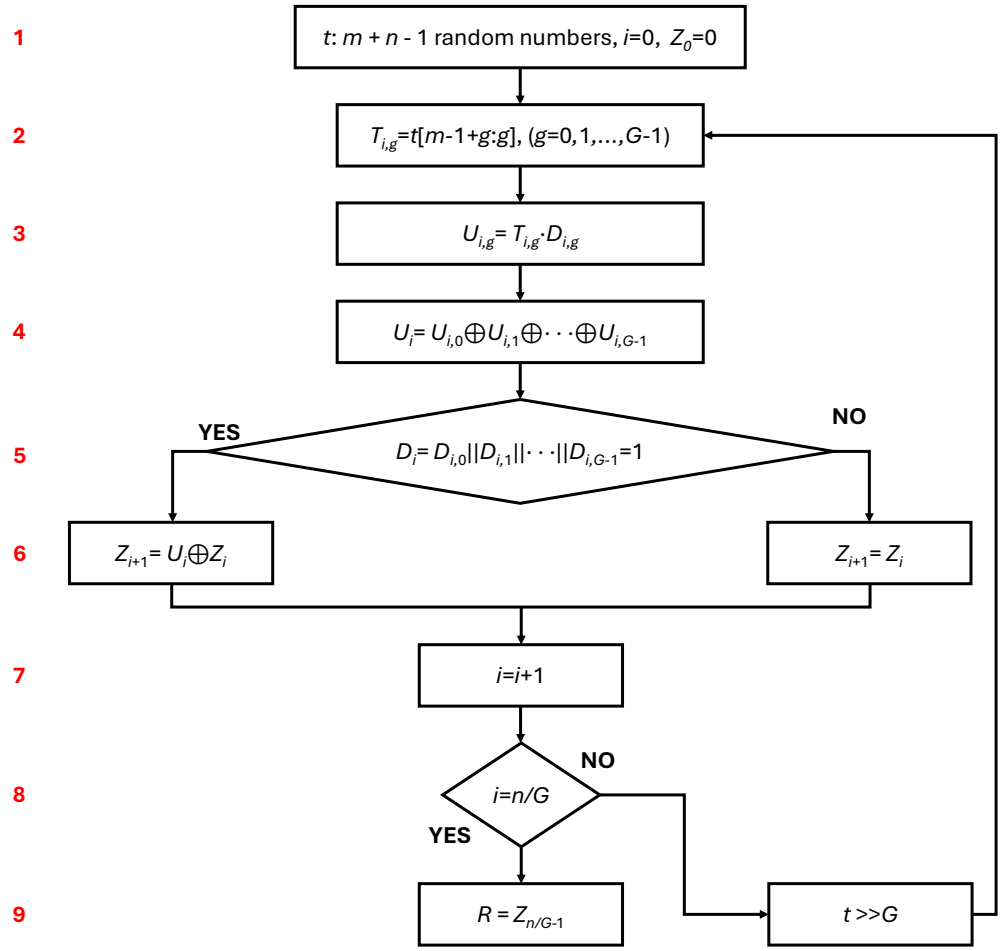


Figure 3.3: Data extraction algorithm flow chart. Adapted from [25].

5. A check on the raw data sub-vector is made to assert if at least one of the elements to be processed is equal to 1.

6. Based on the previous check the following actions are done:

   - If at least one element is 1, the partial result is updated with a bit-by-bit XOR operation between U_i and Z_i, which corresponds to the partial result of the previous cycle.
   - If U_i is composed of only 0's, the partial result remains the same as the previous cycle. This is because the accumulation would produce no effect.

7. The i index updates, stating that a computation cycle has been completed.

8. A check is made on the current computation cycle index to understand if the algorithm is completed.

9. Based on the algorithm being completed or not the following actions are done:

   - If $i = n/G$ then the last partial result corresponds to the final one.
   - If the algorithm is not complete the bit array containing the Toeplitz matrix is shifted by G bits to the right. This way in the next cycle a new sub-matrix is considered.

3.3.3 Circuit Design Strategy

Now that the algorithm behind the matrix multiplication has been explained, it is interesting to understand the circuit design strategy used to implement it. As already mentioned, the circuit is composed of a three-stage pipeline and a schematic representation can be seen in Fig. 3.4.

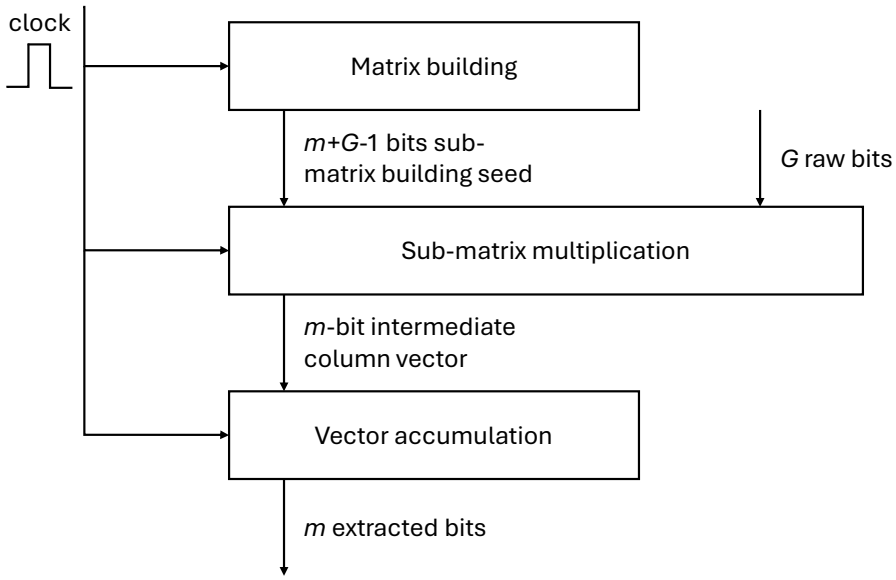


Figure 3.4: Three stage pipeline scheme. Adapted from [11].

Each stage performs a certain phase of the overall computation:

1. The first stage uses the $m + n - 1$ bits that construct the Toeplitz matrix and output the $m + G - 1$ bits necessary to construct the sub-matrix. With the same concept, G bits are taken from the n bit array of raw data.

2. The second stage computes the row by column dot product of the sub-matrix and the sub-vector obtained by the previous stage. In principle, as it will be thoroughly analyzed in the later sections, this stage can be implemented with a fully combinatorial circuit. In other words this stage can be absorbed in one of the other two. However for practically meaningful implementations of this circuit, the depth gets significant and to comply with the timing requirements, it is necessary to insert a data register also in this stage.

3. The last stage accumulates the products obtained by the sub-blocks in the previous stage, in order to obtain the final result.

This circuit alone can carry out real-time randomness extraction but given certain operational circumstances, it is possible to have a further degree of parallelization by instantiating the just described circuit multiple times [11].

In particular, this strategy is useful in case the sampling frequency of the ADC, which can be called S, is higher than the frequency of the clock available on the FPGA, called C. In order to avoid having significant dead time, because the amount of data coming from the high speed ADC is too much for the FPGA to be processed, it is possible to instantiate copies of the circuit of the randomness extractor. As soon as one of them has been replenished with all the data needed, it starts computing and in the meantime the other incoming data is redirected to the successive parallel block.

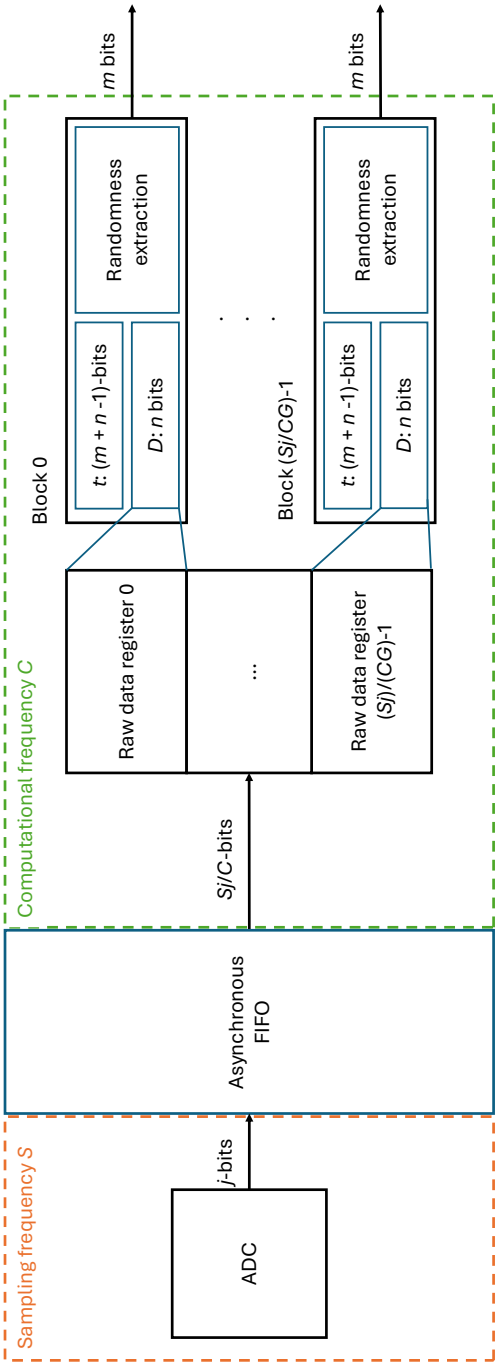


Figure 3.5: Parallel implementation of randomness extractor. Adapted from [11].

A scheme of the described design is shown in Fig. 3.5. It is possible to observe that the speed gap between the ADC and the FPGA's clock is bridged by a sophisticated memory element called asynchronous First In First Out (FIFO). Consider that the resolution of the ADC is j, this memory element is written by j-bits at the frequency S and it is read by Sj/C-bits at the frequency of C. The bit rate on the randomness extraction circuit side is Sj. This amount of data is distributed between Sj/CG parallel blocks. The raw data registers containing the bits to be extracted are filled in nC/jS clock cycles, since they have a width of n-bits. Once a register has been filled, the randomness extraction starts, and the FIFO starts sending data to the register contained in the successive block in a cyclic manner.

3.3.4 Circuit Implementation: Functional Blocks

Now that the idea behind the algorithm and the circuit have been explained, it is possible to analyze how everything has been actually implemented on the FPGA.

The randomness extractor is composed of several functional blocks that have been described using VHDL. The hierarchy of the blocks is depicted in Fig. 3.6. Notice that, as explained in the previous sections, both a free-running and an offline configuration have been implemented. Therefore, minor differences will be present but only in the top-level architecture, here called extractor Finite State Machine (FSM).

Below, the building blocks of the circuit are described following a bottom up approach with respect to the hierarchy.

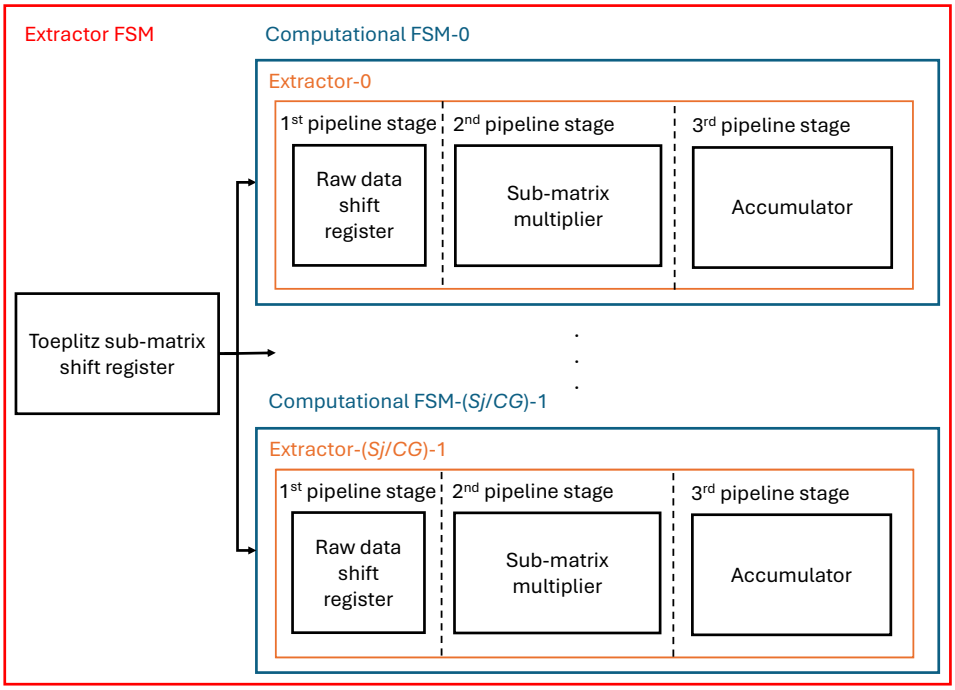


Figure 3.6: Circuit hierarchy scheme.

Toeplitz Sub-Matrix and Raw Data Shift Registers

The Toeplitz sub-matrix register and the raw data registers are core components that represent the first stage of the pipeline. They get filled with data in blocks, since it is supposed to come from the ADC, which likely has a resolution in bits, lower than the dimensions that will be set for these components (n for the raw data register and $n + m - 1$ for the Toeplitz sub-matrix one). Even in the offline configuration, it is supposed that the input data comes from a memory cell in the BRAM of the FPGA, whose width can be expected to be lower than the one of the registers.

To be compliant with the requirements, the incoming data is saved in the least relevant positions of the registers and then a shifting operation to the left is performed.

For what concerns the reading of these registers the concept is similar, with a block of data being read and a rotation operation being performed. This means that the data just read is saved back at the opposite end of the register, with respect to the shifting direction.

Notice that the Toeplitz sub-matrix register is instantiated as a component of the top-level architecture and not of the extractor component as it would be expected. This is because there is no need for the parallel computational modules to use different Toeplitz matrices. Therefore, to save some slice occupation on the FPGA, it has been instantiated a single time and it is routed to each of the parallel extractors.

Sub-Matrix Multiplier

This is the second stage of the pipeline. It is composed of a combinatorial circuit that performs the binary multiplication and a simple register. It is shown in Fig. 3.7, in a one bit example for simplicity. It is composed of G AND gates instantiated in parallel and fed to a cascade of $G - 1$ XOR gates. One input port of the AND gates is connected to one of the bits of the raw data register, while the other port is connected to a Toeplitz sub-matrix register portion representing a column of the sub-matrix. The last XOR gate, containing the final result of the multiplication is connected to the register that feeds the result to the last pipeline stage.

As mentioned, this last memory element has no function for the algorithm, but the depth of this circuit may be significant and not inserting a pipelining register may lead to failed timing constraints and consequent errors during the execution.

Accumulator

This is the last stage of the pipeline and its scheme can be seen in Fig. 3.8. It is composed of a XOR gate, that takes as input on one of the two ports, the result of

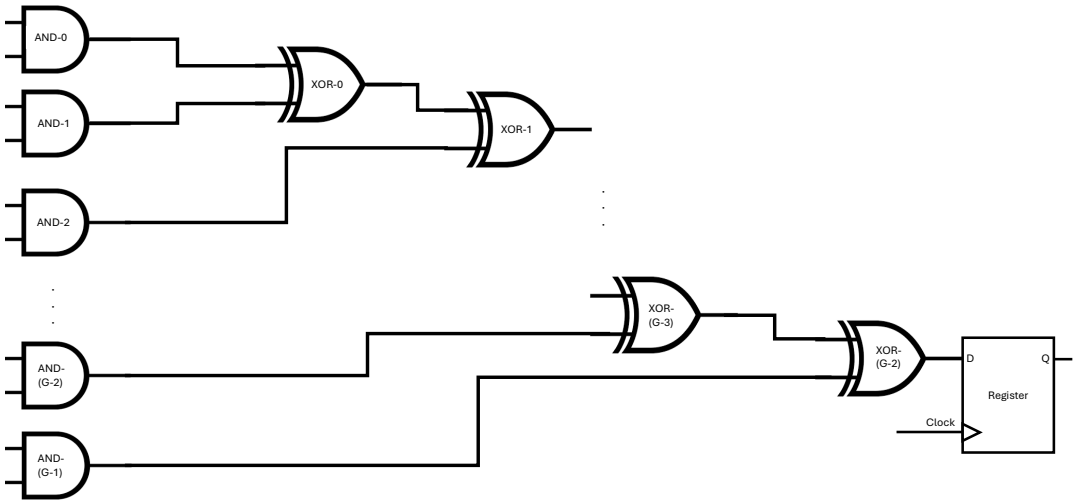


Figure 3.7: Sub-matrix multiplier circuit. Adapted from [9].

the multiplication. This product comes from the register of the previous pipeline stage. On the other input port, the output of the accumulator register is connected. This combination of connections allows to sum the partial result on the previous cycle of the algorithm with the new partial product. Referring again to the flow chart in Fig. 3.3, it is not always true that the new partial result is the result of a sum. In fact, in case the raw data vector is composed of all 0's, the new value has to be equal to the one of the previous cycle. For this reason, a two-to-one multiplexer (MUX) is present with input port 1 connected to the XOR gate output and input port 0 to the output of the accumulator register. The selection signal decides which of the two inputs is fed to the register and it simply goes high in case a bit of the raw data string processed is equal to 1.

Extractor

This is the component of higher hierarchy, containing the three pipeline stages and connecting them as described. Notice that its input ports, besides the obvious clock, synchronous reset, input data (coming from the ADC or the BRAM) and output data port, are also the three write-enable and the three read-enable signals. These are needed for the management of the pipeline. Obviously for the first stage, only the raw data register is piloted, since the Toeplitz sub-matrix register is external to this component. This last aspect is taken into account with the higher-hierarchy architectures that are promptly described below.

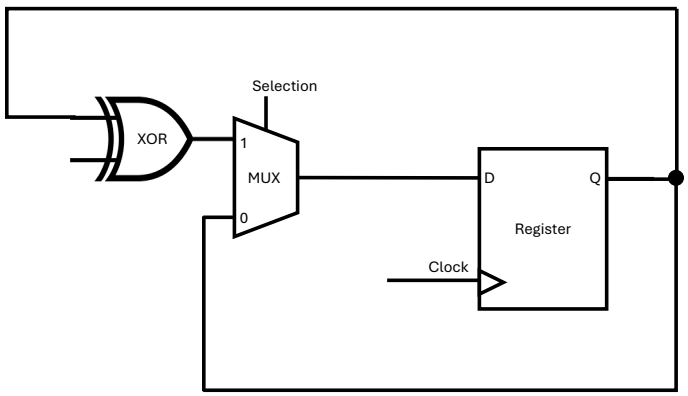


Figure 3.8: Accumulator circuit. Adapted from [9].

Computational Module FSM

This module manages the pipeline, using a Finite State Machine (FSM) type of circuit. The flow chart of the states is shown in Fig. 3.9 and these are now briefly explained:

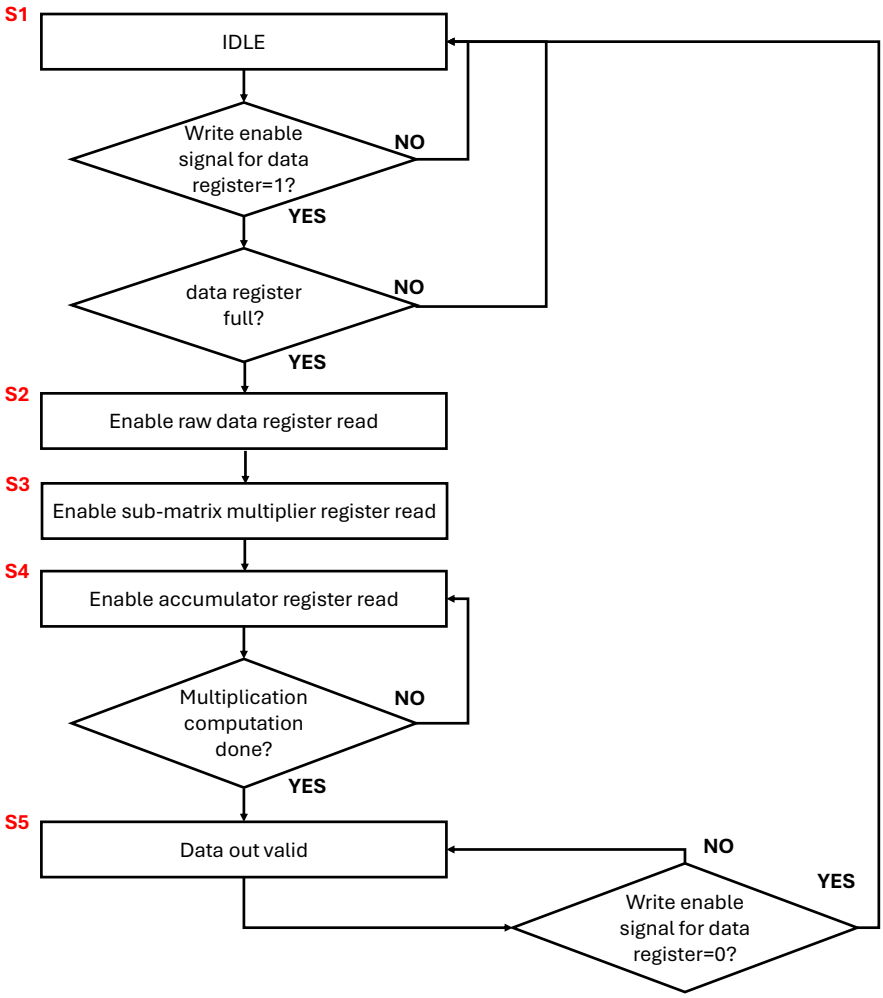


Figure 3.9: Computational module FSM flow chart.

S1) The idle state in which the registers of the pipeline are reset. The system waits for the write enable signal of the raw data register to be high. If it is active, it means that valid data is sent and the register is being filled. As mentioned, the number of clock cycles needed to fill it are nC/jS. Therefore a counter is used to understand at which clock cycle the register is full and it is possible to start the multiplication process.

S2) In this state the read enable signal is activated for the raw data register and the multiplier register is ready to be written.

S3) In this state the read enable signal for the multiplier register is activated and the accumulator register is ready to be written.

S4) In this state the read enable signal for the accumulator register is activated and the accumulation process is started. As mentioned previously, this process lasts for n/G clock cycles. Again a counter is used to keep track on the amount of accumulations already performed and if the correct amount is reached, the next state is asserted.

S5) In this state the output data is considered valid, since the accumulation process is done. Therefore a data-valid signal is asserted. The state returns to the idle one in case the write enable signal for the data register is 0. If this is not the case, the state remains S5.

Extractor FSM

This architecture manages the parallel execution of the computational modules using a FSM type of circuit. A total number Sj/CG of parallel modules is instantiated together with a single Toeplitz sub-matrix register. The purpose of this FSM is to decide the correct raw data register to write in order to obtain an efficient parallel execution. The flow chart of the states is shown in Fig. 3.10. As mentioned before, this module is slightly different in the free-running configuration with respect to the offline one. This minor change is in the fact that, if data is coming from the BRAM, there is an extra clock cycle delay before the data is valid for reading. For this reason, additional states are present but the core functionality remains unchanged.

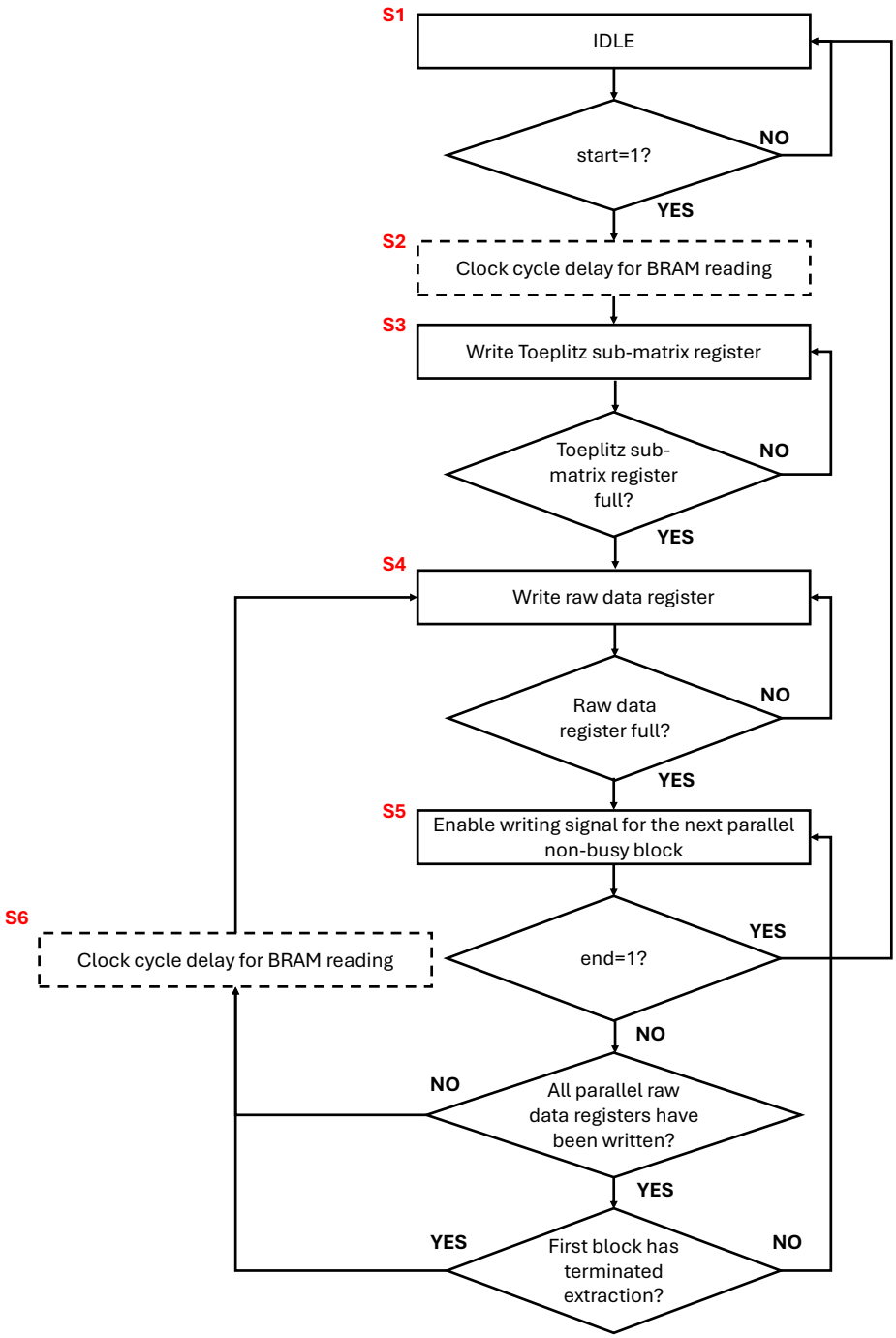


Figure 3.10: Extractor FSM flow chart. States represented by a rectangle with a dashed outline are present only in the offline configuration.

The states work as follows:

S1) In the idle state the circuit does not perform any action and waits for the start signal to be asserted.

S2) This state is present only in the offline configuration. A BRAM read enable signal is asserted and notifies the BRAM controller that the extractor wants to receive data. Since a one clock delay is present before any valid data can be read from the memory element, this state is necessary to request data from the BRAM a clock cycle before anything is actually written on the extractor registers.

S3) The write enable signal is activated for the Toeplitz sub-matrix register, in order for it to be filled. This operation requires $\lceil (n+m-1)C/Sj \rceil$ clock cycles. As usual the track of the number of writings performed is kept by a counter.

S4) One of the parallel raw data registers is now written. The operation takes $\lceil nC/Sj \rceil$ clock cycles, computed again with a counter.

S5) This state is reached once one of the parallel raw data registers has been written. It is now necessary to start filling with data the register relative to the successive block. In order to do this, the write enable signal to the new register is set high, while the one to the previous goes to 0. This mechanism proceeds in a loop to have the most amount of data processed at the same time, up until a end signal is activated. Once this happens, it means that the circuit has to return to the idle state.

Some care has to be taken in the management of the write enable signals for the raw data registers. In fact, it is possible that every block has been filled with raw data but the first block has not finished yet computing the result of the randomness extraction. To take into account this issue some controls are present: initially a check is made to understand if the next block to be written is the first one. If this is not the case, the algorithm proceeds by simply writing the next block. If instead the loop has to restart, a further control is present. This one checks if the first block has completed the computation by analyzing the state of the output data-valid signal of the first computational module block. If this signal is active it means that the loop can start over, otherwise the system remains idle.

S6) Like S2, this state i present to account for the clock delay in the BRAM reading, before starting to write the new raw data register.

In addition to the just described processes, this module has the function to communicate to the blocks external to the extractor, the instant in which a valid

result is given as output. To do so, a data-valid-toggle signal is present, that reads the data-valid signals coming from the computational modules, and toggles as soon as one of them is set to high. This type of control is useful also to determine which of the output strings of the computational modules can be considered a valid and fresh output to the extractor top-level architecture, in order for it to be communicated to the external world.

3.4 Integration of the Randomness Extractor Circuit into the SoC

Now that the circuit for randomness extraction has been described, it is possible to delve into the complete system that is loaded on the System on Chip (SoC). The board supplied for the realization of this work is mounted with a SoC containing both a FPGA and a CPU. Therefore, the complete systems are realized by exporting the hardware platform designed on Vivado on the Vitis software. This application is used to program the CPU with C scripts.

Moreover, it is important to mention, that every component written in VHDL has been defined with the "generic" directive, meaning that each parameter is customizable. This allows for easy testing and flexibility on the realization of a system that is compliant with the user requirements. Before starting to describe the two different configurations implemented, a brief description of the important components that are present in both of them is given:

- Processing system: the component representing the CPU of the SoC available in the Xilinx's Intellectual Property (IP) catalog. Its instantiation is necessary to allow the correct communication between the processor and the FPGA. It is performed using the Advanced eXtensible Interface (AXI) protocol, that will be obviously used in all the other Xilinx's IPs present in the design. For a easier management of the communications a AXI Smart Connect is used.

- Processor System Reset: present to centrally manage the synchronous reset signals across the design. It has to be instantiated for each of the different clock domains present.

- AXI GPIO: these components manage the correct use of the General Purpose Input/Output (GPIO) ports of the processing system. They are the principal way in which the CPU sends and receives signals with the FPGA domain.

- AXI BRAM Controller: allows for easy customization of the BRAM, in terms of width and depth of the memory block instantiated. Uses the AXI protocol to communicate with the CPU, consequently allowing it to write and read data.

- Block Memory Generator: represents the BRAM in the design. This IP permits easy customization on the type of memory needed (RAM, ROM, single port, dual port,...)
- BRAM writer: this custom sequential block is described in VHDL and handles the writing of the data processed by the randomness extractor onto the BRAM. It works by being sensible to the data-valid-toggle signal of the extractor FSM. When a change in the logical value of this signal is registered, the write enable signal for the BRAM is set high. The processed bits are forwarded as output and the correct address of the memory is computed, to cyclically write each cell with a new value. Both a full and a half-full flags are set high, respectively when the last and the middle addresses have been written.
- Bits selector: a simple custom combinatorial block, that takes as input a bit array of a certain length and gives as output a shorter array containing the lest significant bits of the original data.

3.4.1 Online implementation

The Vivado block design is shown in Fig. 3.11. This configuration has been implemented to give proof of the possibility of having an online randomness extraction. However, given the unavailability of a high performance ADC to interface with the FPGA, it is not used in the result analysis, however the correct functioning has been tested.

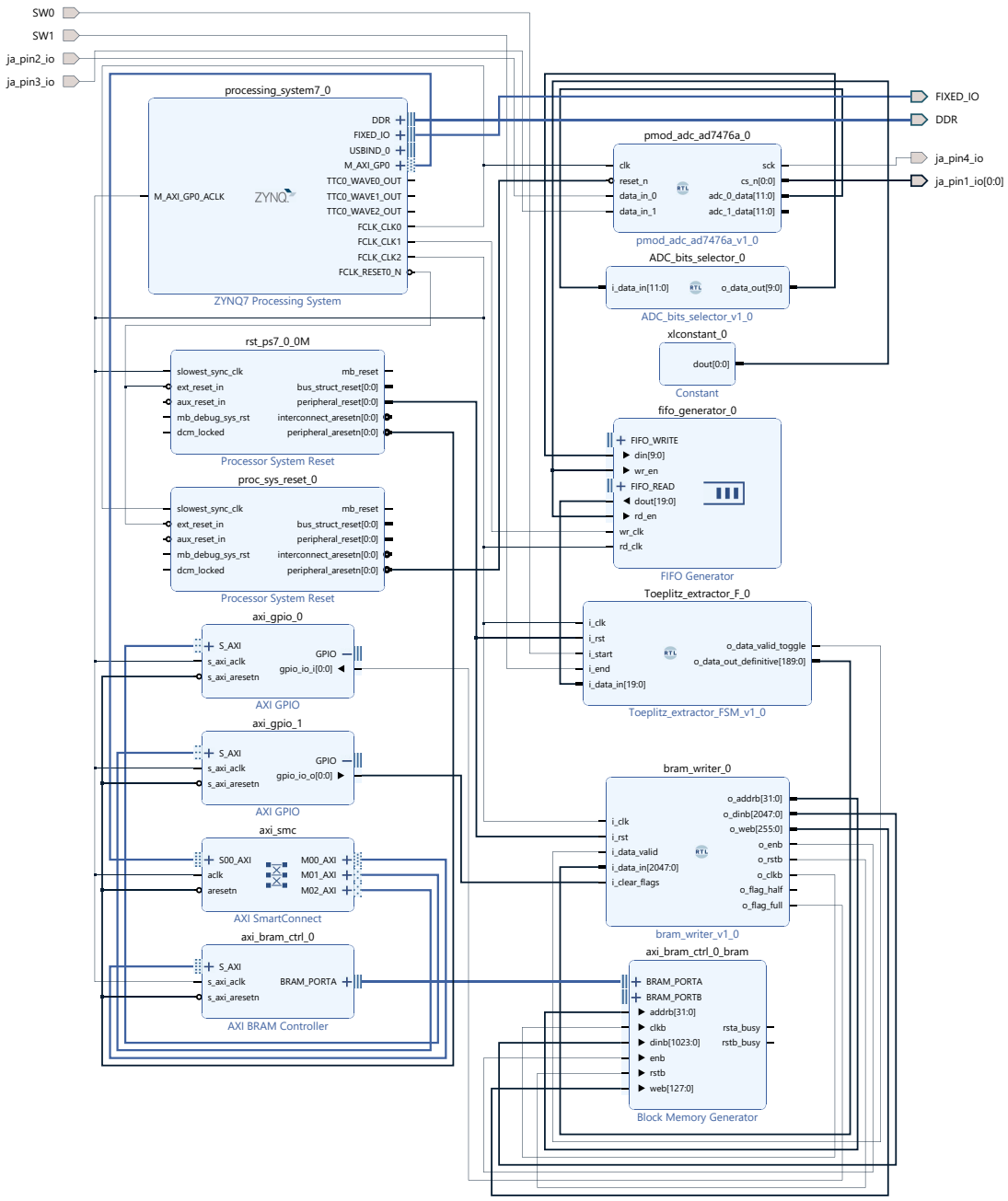


Figure 3.11: Online configuration Vivado block design.

As it can be seen in Fig. 3.11, there are several IPs that have yet to be described.

- ADC compatibility module: the module is connected to the pin bank of the board, in which the ADC is plugged in. The block, either with a AXI interface or not, is supposed to read the digitally converted voltages and giving it to the output port. The block shown in the Vivado design refers to the specific slow ADC module available. In general it is necessary to configure it correctly, depending on the board and the ADC used.

- FIFO generator: a Xilinx IP allowing to customize a FIFO and easily implement the asynchronous version to bridge the fast clock domain of the ADC and the slow one of the FPGA. Since this implementation is a proof of concept, the clock signals are both provided by the processing system.

The system works by having the ADC continuously acquiring data from the QRNG. The bit selector is present in case the whole ADC resolution range is not filled. The converted values are then stored in the FIFO and read from the extractor block. The process of extraction, in this case, starts by flipping a switch on the board but, if it is possible to use more slices of the FPGA, this can be done with a GPIO port of the processing system as well. The same principle holds for the end signal.

The system begins the processing of data and a flag of the BRAM writer is set high once the BRAM has been completely written with data (full flag). At this point the system starts printing the extracted random numbers. It is possible to read these on the PC, by plugging it to the serial communication port of the board and using software such as PuTTY [38]. Once this procedure has been terminated the flags on the BRAM writer are cleared, by sending a signal from the processing system through the GPIO1 port. The signal is then cleared and the process can start again. This particular configuration has been implemented in order for the user to easily read separate data batches of a fixed dimensions. The system can be easily converted to support a continuous stream of precessed data.

Notice that printing on a video terminal millions of characters is the most straightforward but least efficient way to read the processed data. This is because the serial port does not support particularly fast communications and the printing itself is very slow. However, with some modifications on the CPU script, it is possible to send the read data from the BRAM through a faster connection, like Ethernet or even saving it in a file on a SD card.

3.4.2 Offline implementation

The Vivado block design is shown in Fig. 3.12.

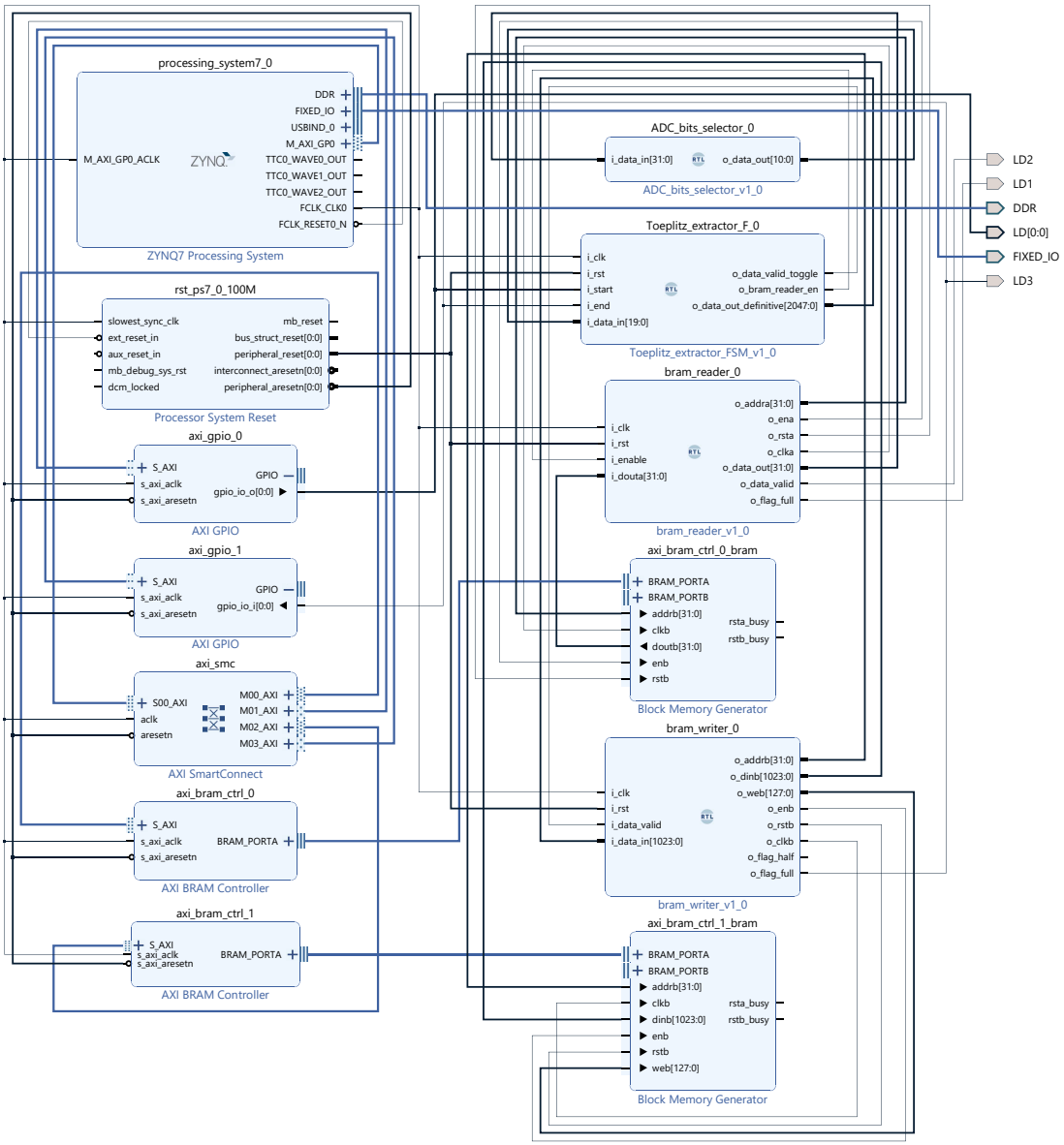


Figure 3.12: Offline configuration Vivado block design.

Besides the already described blocks, another component is present:

- BRAM reader: a custom sequential block that has the complementary function with respect to the writer component. In other words, it is able to read each cell of the memory element and give their content as input to the extractor.

The working principle is having the CPU reading the text file containing the QRNG voltage values. This is done using the AMD Fat File System (FFS) library called XilFFS. These numbers are then written on the BRAM0 in 32 bit binary format. Once each cell of the memory has been written, a high logical level is set for the bit correspondent to GPIO0. This port is connected to work as the start signal of the extractor FSM. At this point BRAM0 is read. Notice that a bit selector is used, since the values saved are not big enough to be represented in 32 bits. Therefore a smaller portion of the least significant bits is considered as input data for the extractor. The system then waits for the GPIO1 to go to 1. This port is connected to the full-flag of the BRAM writer module. In the instant in which this event happens, the entire content of BRAM1 is written on a text file on the SD card.

In simpler terms, the system reads the data batch of QRNG voltage values from the SD card and saves it on a portion of the BRAM. At this point the processing starts and the extracted random values are saved on another portion of the BRAM. When the memory is full, the random numbers are saved back on the SD card.

Chapter 4

Results

This chapter presents the results obtained from the QRNGs. Sec. 4.1 discusses the phase fluctuations-based QRNG, Sec. 4.2 examines the vacuum state heterodyne measurement-based QRNG, and Sec. 4.3 provides an overview of the performance of the FPGA implementation.

4.1 Phase Fluctuations-Based QRNG

4.1.1 Experimental Setup Analysis

The value regarding the bias current of the laser diode as well as its temperature, have been set after careful analysis of the optical spectrum of the emission. As expected, the spectrum widens with a lower bias current. However, a tradeoff has to be made between having a poorly correlated emission and a signal on the photodetectors that is high enough to be detected. As mentioned, the working point has been set at 13.5 mA of bias current, with a temperature of 24°C.

Another free parameter is the length of the delay line. It has been explained how it has to be long enough to have no correlation between the photons reaching the beam splitter. The experiment has been carried out with 2 m, 5 m and 7 m of delay line. The intermediate length has proven to be sufficient.

The absence of correlation can be verified simply by moving the optical fiber: in case there is still significant coherence in the electromagnetic radiation coming to the beam splitter, it is possible to observe and abrupt change in the optical power measured. This is because a movement of the optical fiber changes the phase at which the electromagnetic waves arrive at the beam splitter, therefore the interference produces a change in the instantaneous optical power. On the other hand, if coherence has been lost, no change is observed.

4.1.2 Raw Data Analysis

Using the offline configuration a total of 16 M voltage values have been saved on the hard disk. These values are signed integers on 12 bits, which is the resolution of the ADC. This means that the data batch is composed of 192 Mb.

The probability distribution of the digitally converted voltage values measured are plotted in Fig. 4.1.

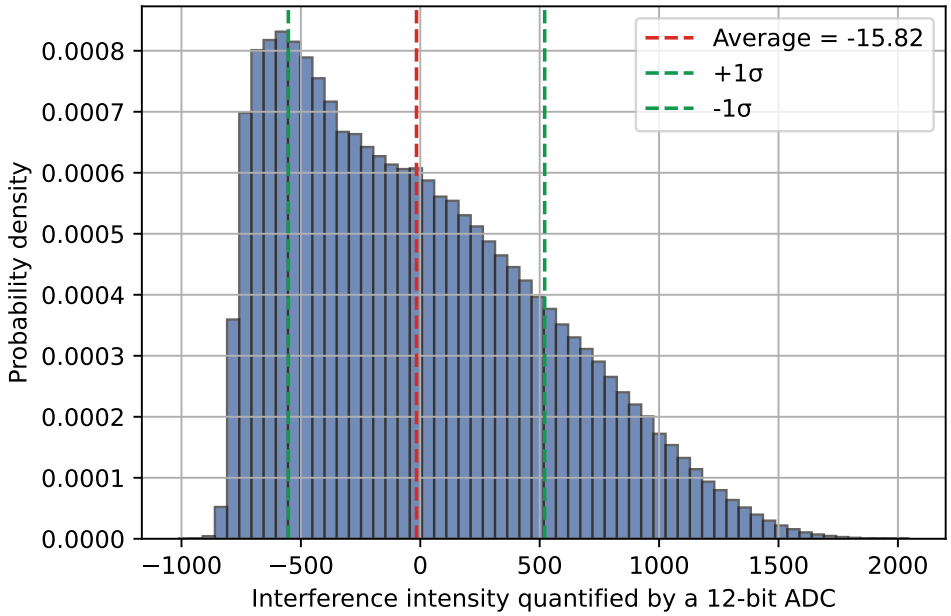


Figure 4.1: Probability distribution of the digitally converted voltage values measured from the phase fluctuations-based QRNG.

What is expected is a Gaussian distribution of values. In fact, in the mathematical model explanation, it has been said that the phase term relative to the spontaneous emission has a Gaussian distribution. This behavior appears to be respected for the positive voltage values but the same cannot be said for the negative ones. The negative tail of the Gaussian appears to be compressed in a certain range of values. In order to investigate the origin of this phenomenon, an oscilloscope has been used to observe the photodetector's signal. This behavior has still been observed, even when varying parameters like the bias current of the laser diode. It has been concluded that the reason the negative values are squeezed is the AC coupling of the photodetector. This element, by eliminating the DC component, is probably distorting the negative valued voltages.

The autocorrelation function of the obtained bit sequence, can be seen in Fig. 4.2.

It can be observed that numerous peaks are present and that they appear with a certain periodicity. This is a hint on the presence of a fairly high degree of

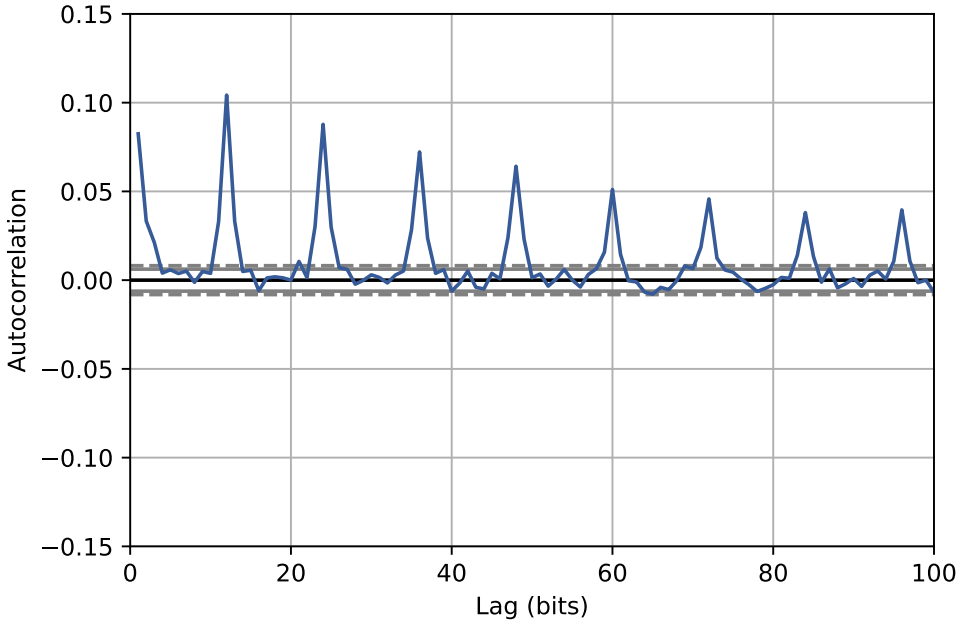


Figure 4.2: Autocorrelation function of 10^5 raw data bits acquired from the phase fluctuations-based QRNG. The dashed and full horizontal lines respectively represent the 99% and 95% confidence bands. These are the ranges in which the the values are expected to fall, if the sequence is uncorrelated with a confidence equal to the respective percentage.

correlation between the bits of the sequence and a periodic recurrence of certain patterns. It is also evident that the values fall outside of the confidence bands. This is due to the imperfections in the experimental apparatus, as well as the behavior previously described of negative voltage values. It also possible that these phenomena are imputable to the presence of some degree of residual correlation between the photons.

These characteristics are supposed to be fixed thanks to the post-processing implemented on the FPGA. In order to perform it, it is necessary to know the min-entropy related to the sequences produced by the QRNG. In this case, this value has to be estimated directly on the raw data. The min-entropy on sequences of 12 bits of the entire data batch under test is 9.928. This value has been computed using Eq. 3.3. The length of the sequences considered for the min-entropy computation is the effective number of bits of the digital conversion, which in this case equals the ADC resolution. It can be concluded that the percentage of bits that can be extracted is $(9.928/12) \cdot 100 \approx 82.68\%$. This value sets a higher bound on the ratio between rows and columns of the Toeplitz matrix during the post-processing.

4.1.3 Extracted Data Analysis

The raw data just described has been processed with the FPGA. Even though the estimated min-entropy indicates that it is technically possible to extract about 80% of the bits, it has been decided to extract 50% of them with the algorithm explained in Sec. 3.3. This is because this type of statistical characterization of raw data is unreliable, since the data batch is not infinite in dimension. By considering half of the bits as extractable, it is safer to say that the resulting random numbers have good characteristics.

The autocorrelation has been plotted in Fig. 4.3.

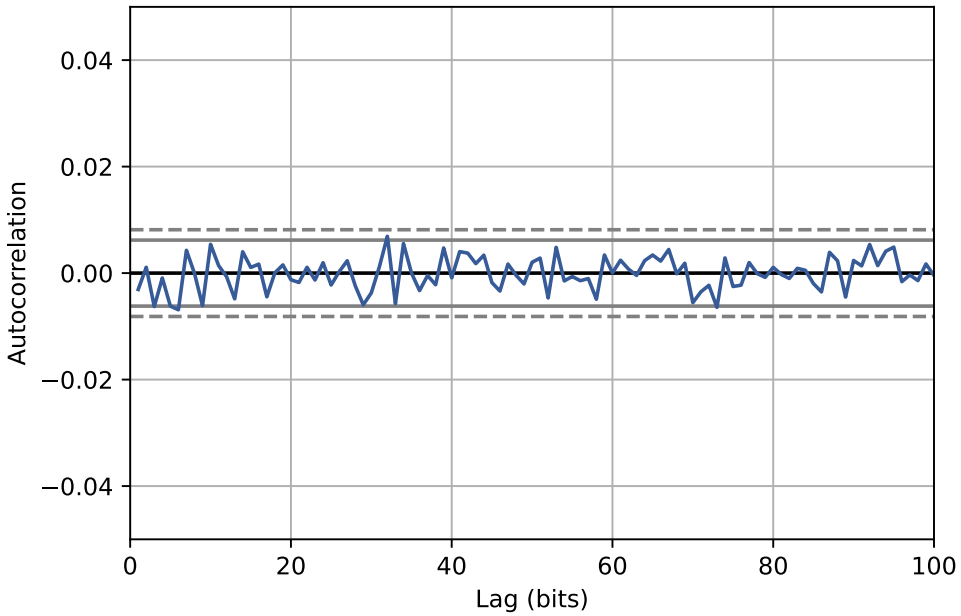


Figure 4.3: Autocorrelation function of 10^5 extracted data bits from the phase fluctuation-based QRNG.

It is possible to observe that the correlation between the bits has significantly decreased and the function's values lie inside the 99% confidence interval. While this is a good measure of the effect of randomness extraction, it is also a good practice to submit the sequence to a test battery. For this reason the NIST statistical test suite, which is described in Appendix A, has been used.

The test has been carried out on 80 sequences of 1 M bits each, with a level of significance $\alpha = 0.01$. According to the documentation the portion of times each test should be passed with these conditions is at least $0.99 - 3\sqrt{0.99 \cdot 0.01/80} \approx 0.99 - 0.033 = 0.957$, therefore with a minimum of 96% of sequences that passed each test, there is no evidence to assert the extracted bit string is not random.

The results are shown in Fig. 4.4.

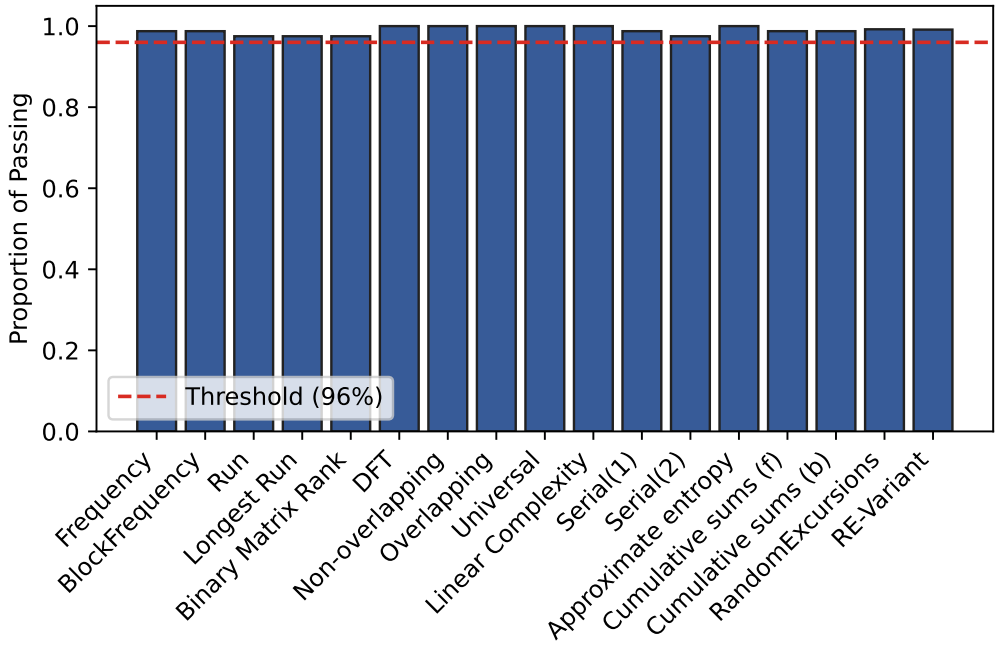


Figure 4.4: Proportion of passed test of the NIST statistical test suite for phase fluctuations-based QRNG after randomness extraction.

It can be observed that each of the 15 tests of the suite has been passed at least 96% of times.

According to the NIST guidelines, it is also necessary to make a check on the uniformity of the $P-values$ obtained by every test. The distribution is plotted in Fig. 4.5.

In this graph it can be seen that the Random Excursion Test and the Random Excursion Tariant Test present many more $P-values$, this is because they are composed of multiple sub-tests and here they are all plotted at the same time. It appears that the $P-values$ are uniformly distributed. However, this type of analysis is not rigorous. Then the Kolmogorov-Smirnov test is applied to have a further proof of uniformity. As a result, a final $P-value$ is obtained for each test, by aggregating the multiple ones present. The result can be seen in Fig. 4.6.

The $P-values$ lie between the level of significance $\alpha = 0.01$ and 1. It can be concluded that the extracted dataset obtained with the phase fluctuations-based QRNG appears to be random.

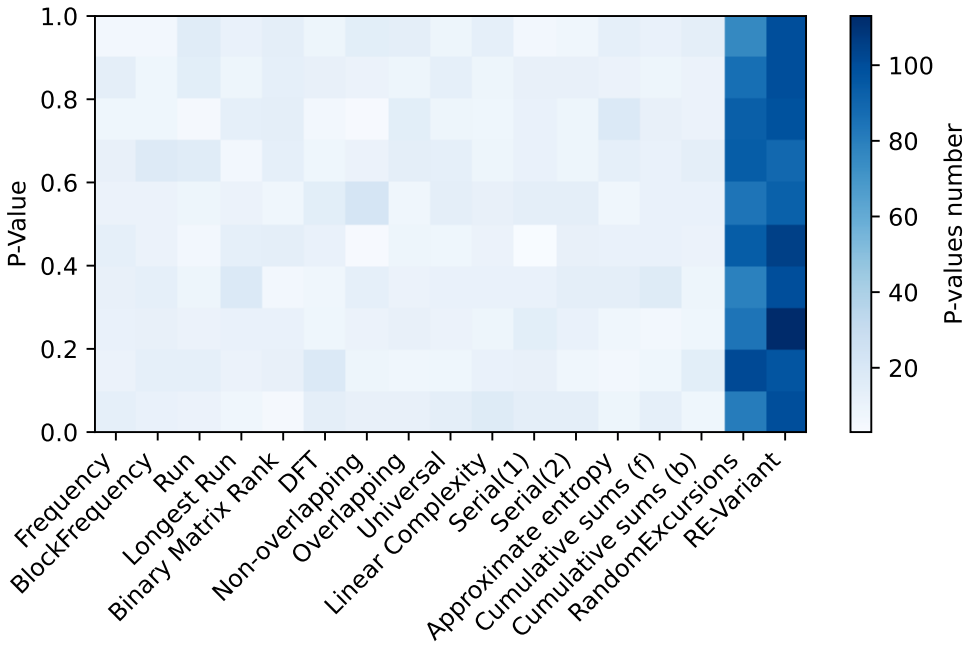


Figure 4.5: $P-values$ distribution for uniformity analysis for phase fluctuations-based QRNG after randomness extraction.

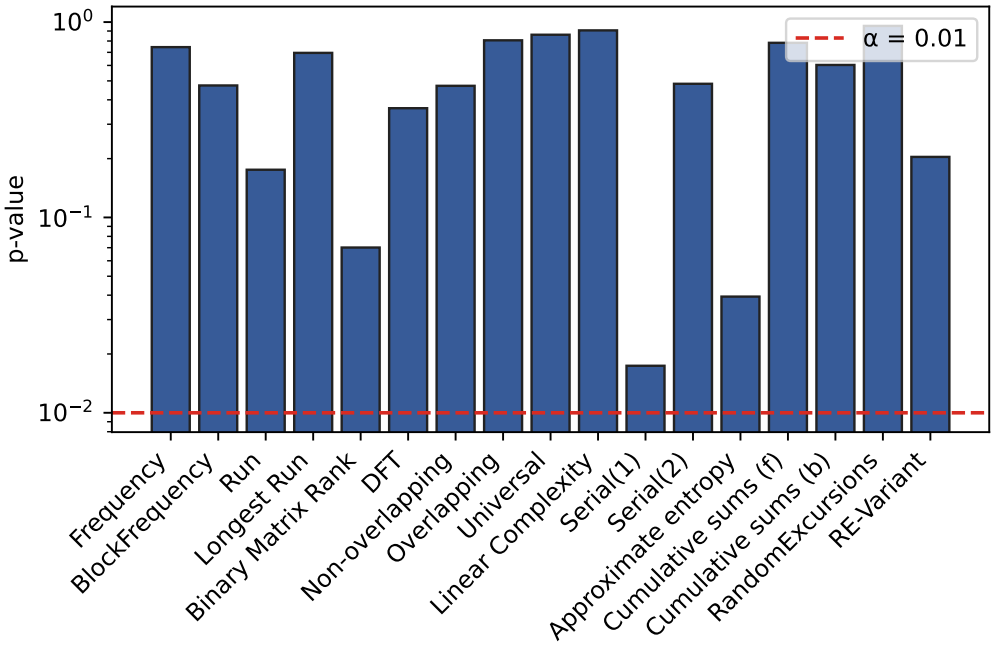


Figure 4.6: $P-values$ for the phase fluctuations-based QRNG after randomness extraction, obtained with the Kolmogorov-Smirnov test.

4.2 Vacuum State Heterodyne Measurement-Based QRNG

4.2.1 Experimental Setup Analysis and Calibration

The values regarding the bias current and the temperature of the local oscillator have been set to the typical operational regime. The optical power is set with the VOA at 20.1 mW.

For what concerns the data acquisition, since the photodetectors used work with 100 MHz of bandwidth, a Butterworth band-pass filter has been digitally applied to eliminate the frequencies outside the 1 MHz to 100 MHz window. Moreover, to avoid oversampling, decimation has been used.

In the previous chapters, it has been said that it is necessary to estimate the resolution of the two photodetectors. This is done to compute the lower bound of the conditional min-entropy and be able to extract a sufficient amount of randomness. For this reason a calibration phase is done before the detection stage: the optical power of the local oscillator is swept from a few mW to around 23 mW. The signal of the photodetectors is sampled for certain values of optical powers in this range. It is then possible to compute the variance of the signal in these points and plot it against the optical power.

The power sweep is performed with the VOA, by piloting it with the electronic driver. These actions are programmed with a Labview script that sets up the whole calibration stage, including the data acquisition and the signal variance computation.

The calibration curve can be seen in Fig. 4.7.

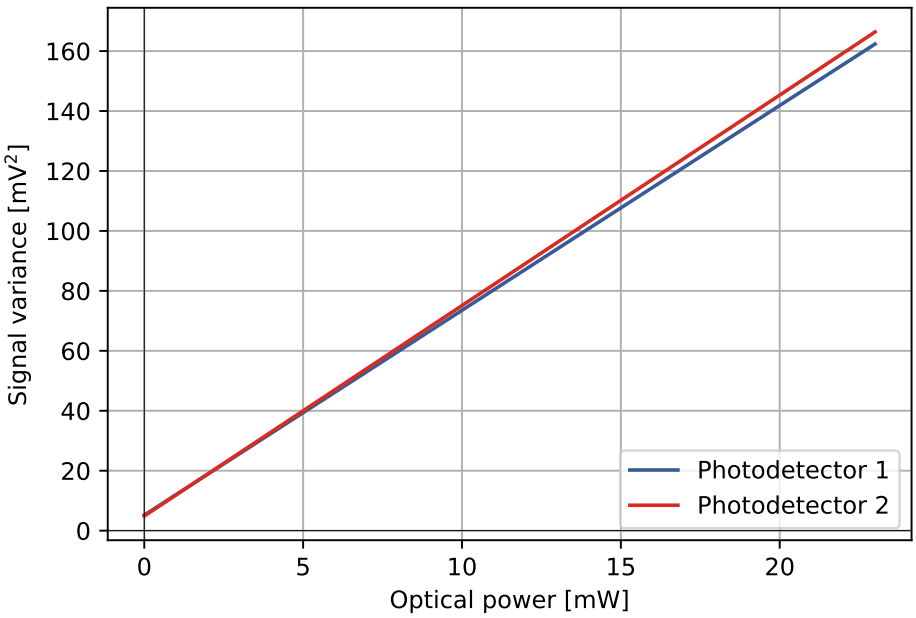


Figure 4.7: Linear fit of the signal variance as a function of the optical power for the two photodetectors for the vacuum state heterodyne measurement-based QRNG.

Since the relationship between the two variables is linear, a linear fit has been performed. Notice that if the sweep had been performed on a wider range of optical powers, the variance would saturate at a certain level.

The fit is fundamental for the characterization of the measurement device. In fact, two important parameters are extracted, namely the slope and the intercept of the fits, respectively called s_i and q_i, with $i = 1{,}2$ to distinguish between the two photodetectors. According to the heterodyne scheme, each photodetector measures a different quadrature of the electromagnetic field. The results are $s_1 = 6.833$ mV^2/mW, $q_1 = 5.178$ mV^2, $s_2 = 7.021$ mV^2/mW and $q_2 = 4.878$ mV^2.

The slope is useful to convert variances in the physical regime (measured in V) to variances in shot noise units: the conversion factor is $2s_iP_{LO}$, with $P_{LO} = 0.0201$ W being the optical power of the local oscillator. Since it is necessary to compute the resolution of the measurement apparatus in shot noise units, its resolution has to be converted from the physical units. In order to do so, the scale of the ADC, which is equal to 200 mV, has to be divided by the amount of numbers that can be represented with the effective bits available. Since in this case the effective bits are 11, this quantity is equal to 2^{11}.

The result is $\delta_V = (0.2 \text{ V})/2^{11} = 9.766 \cdot 10^{-5}$ V. Now that the resolution in physical units has been computed, it is possible to obtain the one in shot noise units:

$$\delta_{1,2} = \frac{\delta_V}{\sqrt{2s_{1,2}P_{LO}}} \tag{4.1}$$

The results are $\delta_1 = 5.891 \cdot 10^{-3}$ and $\delta_2 = 5.812 \cdot 10^{-3}$. Finally it is possible to compute the lower bound of the conditional min-entropy:

$$H_{min}(X_\delta|\mathcal{E}) \geq \log_2\left(\frac{\pi}{\delta_1\delta_2}\right) = 16.485 \tag{4.2}$$

This is the lowest number of bits that can be extracted from double the amount of effective bits of the ADC (the double has to be considered because the measure is made contemporarily on the two quadratures). This means that the percentage of extractable bits is $16.485/(2 \cdot 11) \cdot 100 = 74.9\%$

4.2.2 Raw Data Analysis

Using the offline configuration, a total amount of 10 M of values per photodetector has been saved on the hard disk. Each number is a signed float and gets later converted to a signed integer on 11 bits, in a single file containing both the photodetectors measurements. The total amount of data is then 220 Mb.

The fluctuations are reported in Fig. 4.8 and Fig. 4.9.

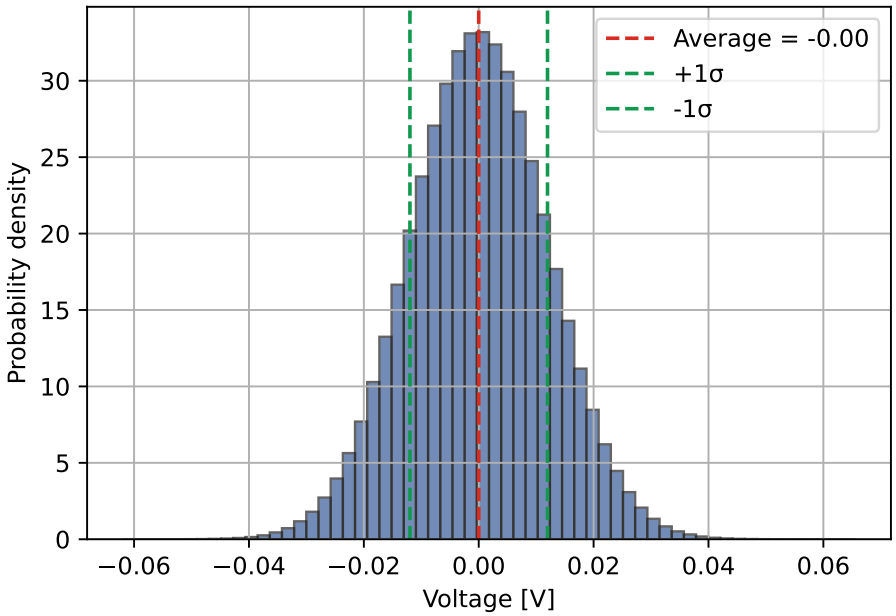


Figure 4.8: Probability distribution of the voltage values measured on photodetector 1 in the vacuum state heterodyne-measurement based QRNG.

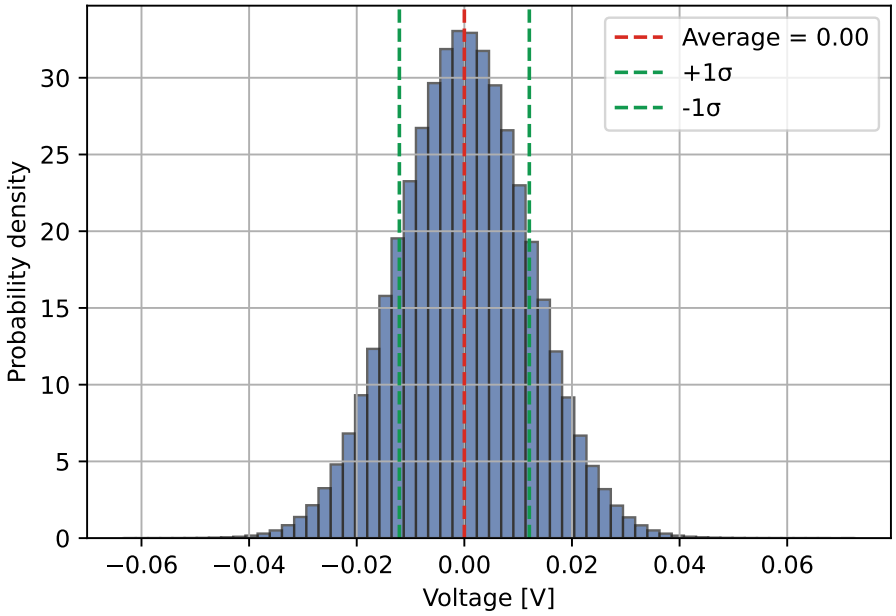


Figure 4.9: Probability distribution of the voltage values measured on photodetector 2 in the vacuum state heterodyne-measurement based QRNG.

These plots represent the quadratures distributions in physical units, which are quantities proportional to the Husimi function projected onto the respective electromagnetic field quadratures. By extrapolating the variances from the plots, the values obtained are $\sigma_1^V = 0.0015$ mV^2 and $\sigma_2^V = 0.0014$ mV^2, which converted in shot noise units become $\sigma_1 = 0.516$ and $\sigma_2 = 0.522$. The expected value for the variance of the vacuum state is $1/2$ but it can be seen that the measured ones are greater. This happens because of the electronic noise that widens the distribution.

The autocorrelation is plotted in Fig. 4.10.

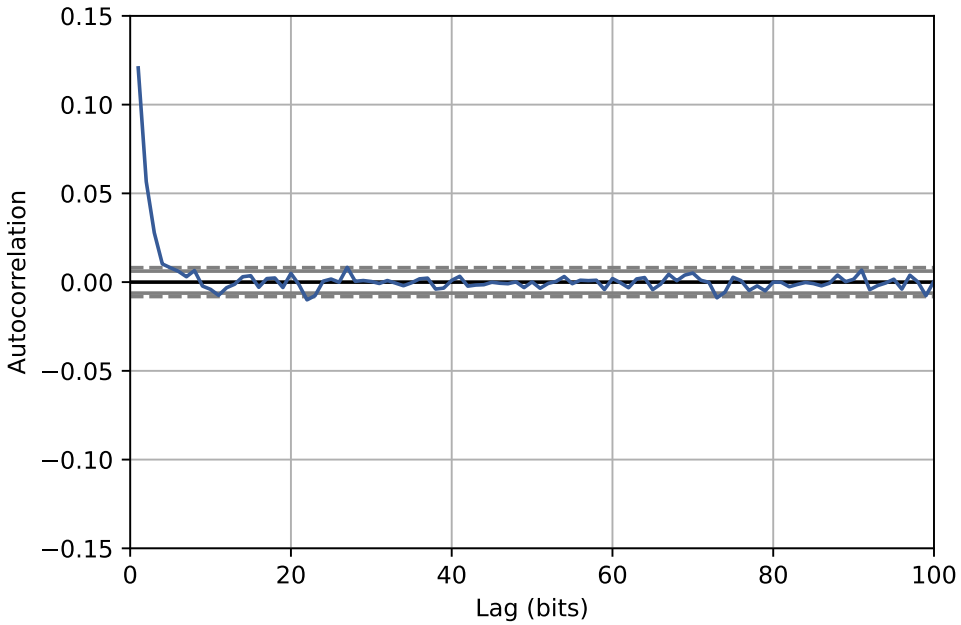


Figure 4.10: Autocorrelation function of 10^5 raw data bits acquired from the vacuum state heterodyne-measurement based QRNG.

The function lies mostly inside the 99% confidence band, expect for an accentuated initial peak and some other minor ones. This behavior shows that the data presents correlation only with the immediately subsequent bits and that it is quickly lost. Differently from the phase-fluctuations based QRNG the periodic behavior in the sequences appears to be absent but this does not mean that a bias in the appearance of certain numbers is not present. Anyway, randomness extraction is always necessary to eliminate any possible correlation, due to the experimental apparatus imperfections.

Before proceeding with the analysis of the processed data, it is interesting to evaluate the min-entropy of the acquired data and compare it to the lower bound set during the calibration phase. This value has been computed using Eq. 3.3. The result is $H_{min} = 16.553 > H_{min}(X_\delta|\mathcal{E}) = 16{,}485$, which reveals that the bound is tightly respected.

4.2.3 Extracted Data Analysis

The raw data has been processed with the FPGA, by considering a ratio between the rows and the columns of the Toeplitz matrix of 70%.

The autocorrelation function is plotted in Fig. 4.11.

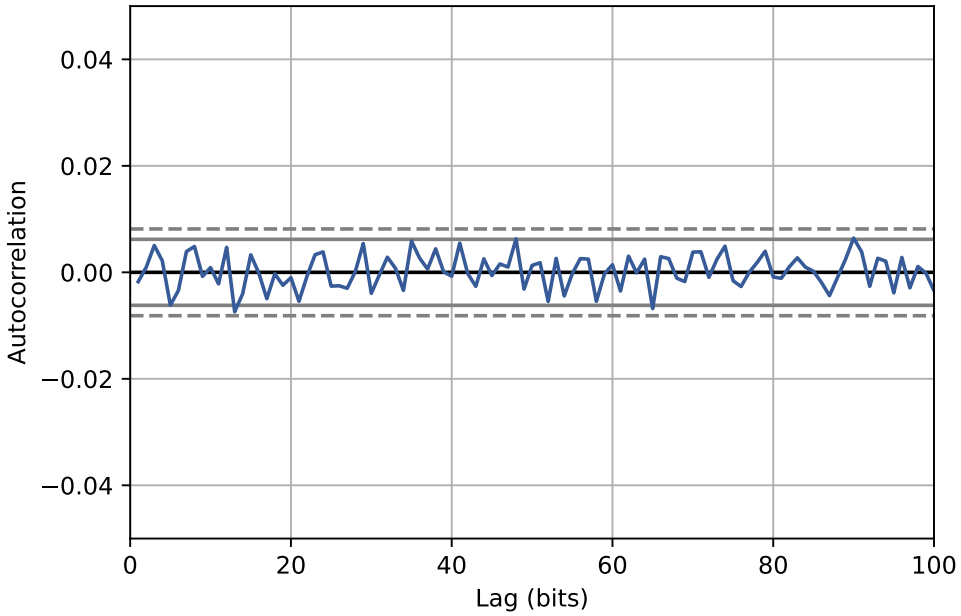


Figure 4.11: Autocorrelation function of 10^5 extracted bits from the vacuum state heterodyne-measurement based QRNG.

It can be observed that now the function lies entirely inside the 99% confidence interval and that the initial peak has been eliminated.

The bit string can now be submitted to the NIST test battery, described in Appendix A . The tests have been carried out on 150 sequences of 1 M bits. Again the level of significance has been considered to be $\alpha = 0.01$, therefore the proportion of the times in which each test should be passed is $0.99 - 3\sqrt{0.99 \cdot 0.01/150} \approx 0.99 - 0.024 = 0.966$. Consequently with at least 97% of correct tests, there is no evidence to conclude that the sequence is not random.

The results are shown in Fig. 4.12

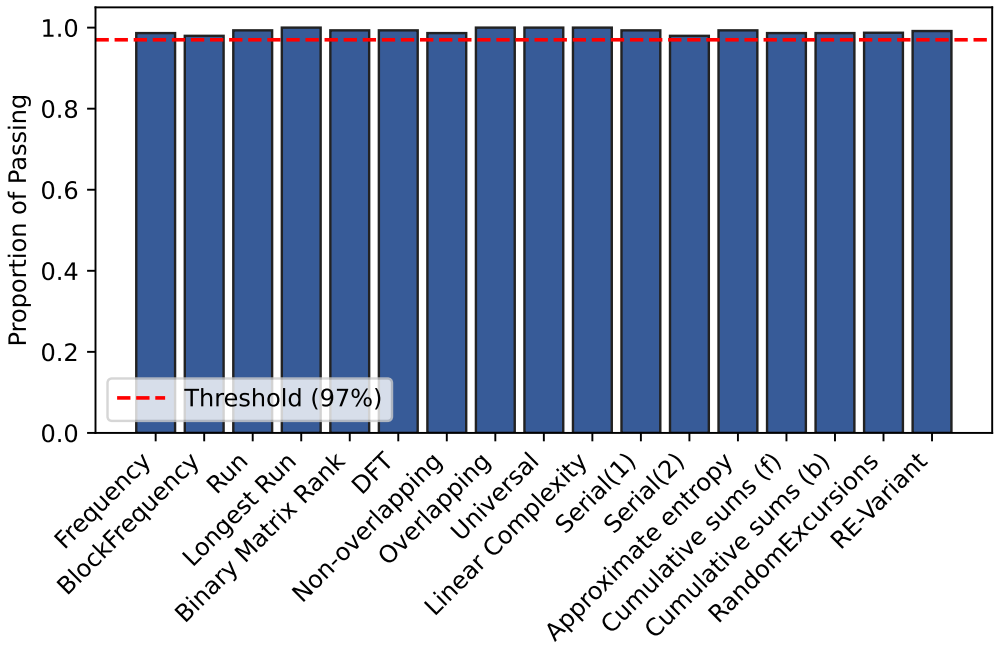


Figure 4.12: Proportion of passed test of the NIST statistical test suite for the vacuum state heterodyne-measurement based QRNG.

It is possible to observe that for every test of the suite the proportion of passing is at least 97%. Like in the result discussion for the phase fluctuations-based QRNG, it is good practice to estimate the uniformity of the $P - values$ obtained from the tests. A diagram, containing the distributions is shown in Fig. 4.13.

The histogram shows a good degree of uniformity in the distribution of $P-values$. Again here it is evident the higher number of $P - values$ obtained with certain tests.

Also in this case it has been decided to submit the $P-values$ to the Kolmogorov-Smirnov test to further verify the uniformity and obtain a final aggregated $P-value$ for each case. The results are shown in Fig. 4.14.

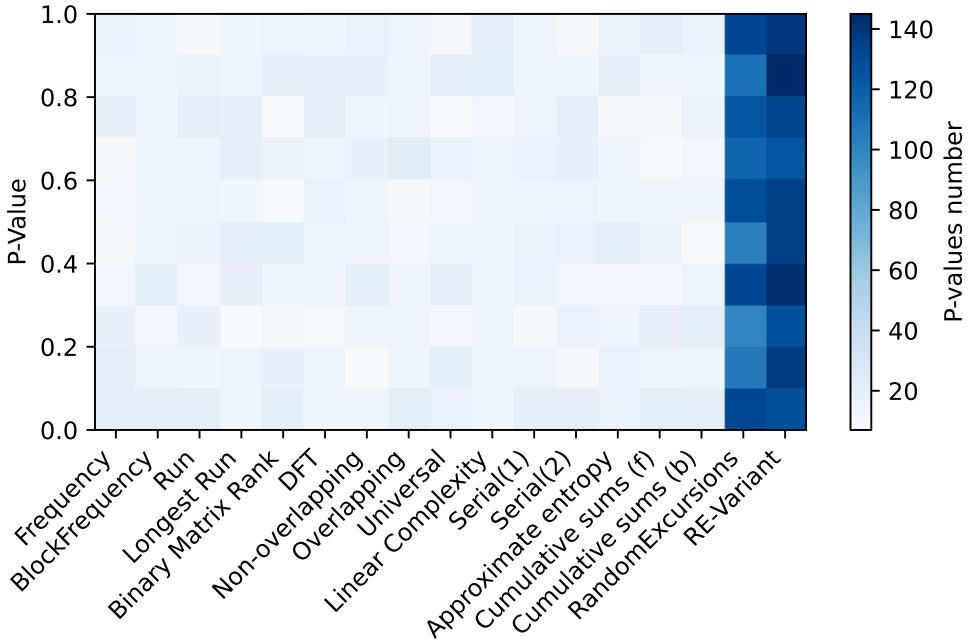


Figure 4.13: $P - values$ distribution for uniformity analysis for vacuum state heterodyne-measurement based QRNG.

The resulting $P - values$ are between $\alpha = 0.01$ and 1, therefore it can be concluded that there is no evidence to state that the numbers produced are not random.

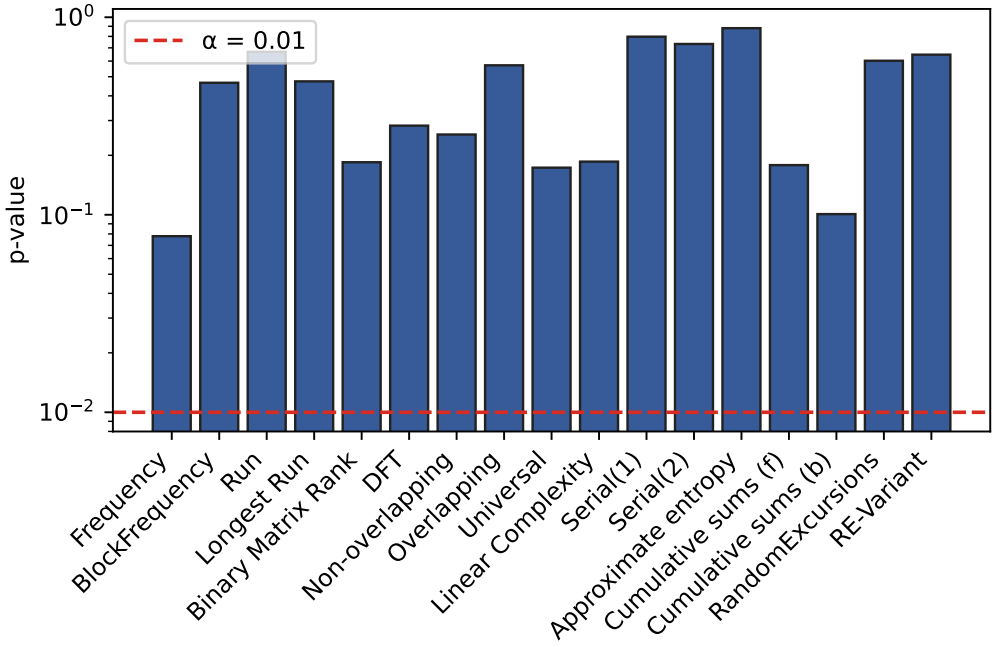


Figure 4.14: $P - values$ for vacuum state heterodyne-measurement based QRNG after randomness extraction, obtained with the Kolmogorov-Smirnov test.

4.3 Circuit Performance

The target of the circuit implementation is to provide a solution that is efficient from the computational standpoint, providing high-rate randomness extraction, and that occupies the least amount of slices on the FPGA device. The first point is tackled intrinsically by the algorithm that the circuit performs, which computes the matrix multiplication in a fixed amount of clock cycles (provided that the dimensions of the matrix and the number of sub-blocks instantiated has been decided). The occupation of FPGA resources, instead, is completely up to the definition of the blocks of the circuit and requires a careful analysis of the operations that have to be performed.

4.3.1 Slice Occupation and Security Parameter

A particular attention has to be paid for the first pipeline stage. In fact the implementation of its functionality, by means of registers that are written and read in a sliding window manner, produces a deep level of logic. After synthesis, a significant occupation of the FPGA slices is obtained. It has been estimated that, compared to the shifting register, this type of implementation occupies about triple the amount of slices.

As mentioned in the previous chapter, another optimization factor has been found in the instantiation of a single shifting register for the pipeline stage containing the bits for the Toeplitz matrix.

The most optimal version found for the circuit, which is the one described in the previous chapter, has been synthesized with different dimensions of the Toeplitz matrix, to test the slice occupation. Notice that these results are strictly bound to the FPGA used, since the number of slices available changes with different devices. Moreover a more capacious device may allow for a more efficient synthesis strategy, given the higher number of resources available, that further reduces the occupation rate. The results for the FPGA used in this work can be observed in Fig. 4.15.

It is possible to see that in the conditions described, the FPGA used is capable of synthesizing a circuit for the multiplication of a Toeplitz matrix of dimensions $n = 2920$ and $m = 2048$, with two parallel blocks and an occupation around 53%. Notice that the dimension of the Toeplitz matrix has important consequences in the randomness extraction. In fact, the Leftover Hash Lemma explained in Sec. 3.2 says that the length of the extracted string can be computed as $m = n \cdot H_{min}/j - 2\log_2(1/\varepsilon)$, where j is the resolution in bits of the ADC, H_{min} the min-entropy and ε is the security parameter. Therefore, by considering an extraction ratio around 70%, like in the data reported in Fig. 4.15 and by considering a min-entropy equal to the lower bound found in the vacuum state heterodyne-measurement-based QRNG (74.9%), it is possible to compute for each case the

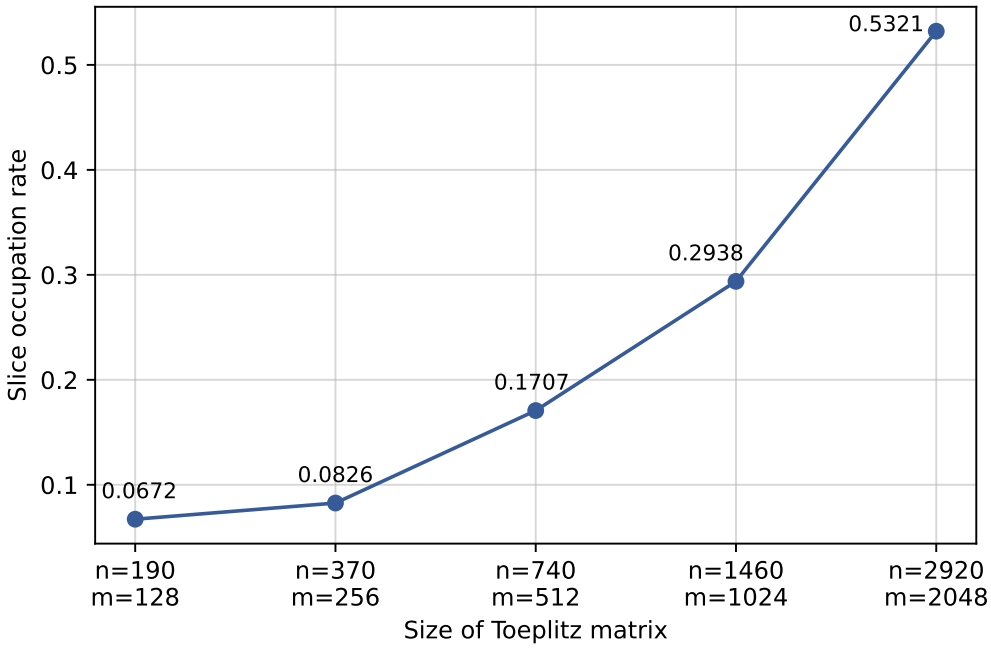


Figure 4.15: Slice occupation rate for two randomness extraction blocks instantiated in parallel and $G = 10$, the number of sub-matrices used for the multiplication in each block.

security parameter as

$$\varepsilon = 2^{(m-n\cdot H_{min}/j)/2} \tag{4.3}$$

The results can be seen in Fig. 4.16.

The security parameter is an important variable to consider during randomness extraction, since at each extraction cycle it increases. This means that it is more convenient to use higher dimensions matrices.

An analysis on the slices occupation has been carried out also by varying the amount of parallel blocks instantiated and maintaining fixed the Toeplitz matrix dimensions. The results can be seen in Fig. 4.17. It can be observed that when the number of parallel blocks is doubled, the slice occupation increases by slightly less than a factor of two.

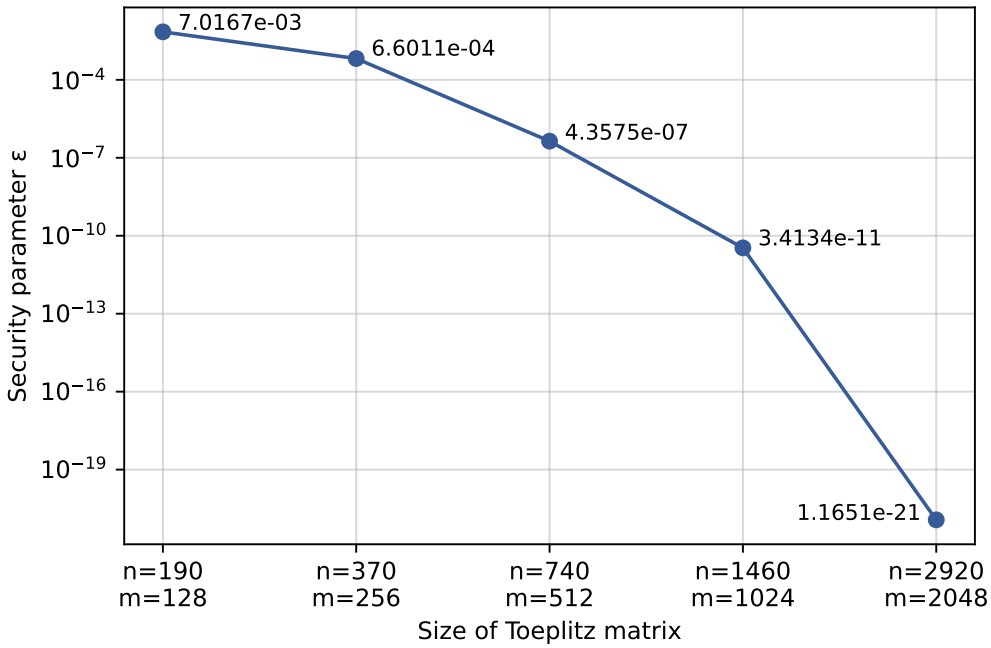


Figure 4.16: Security parameter as a function of the Toeplitz matrix dimensions, when the extractable randomness percentage of the input raw data is 74.9%.

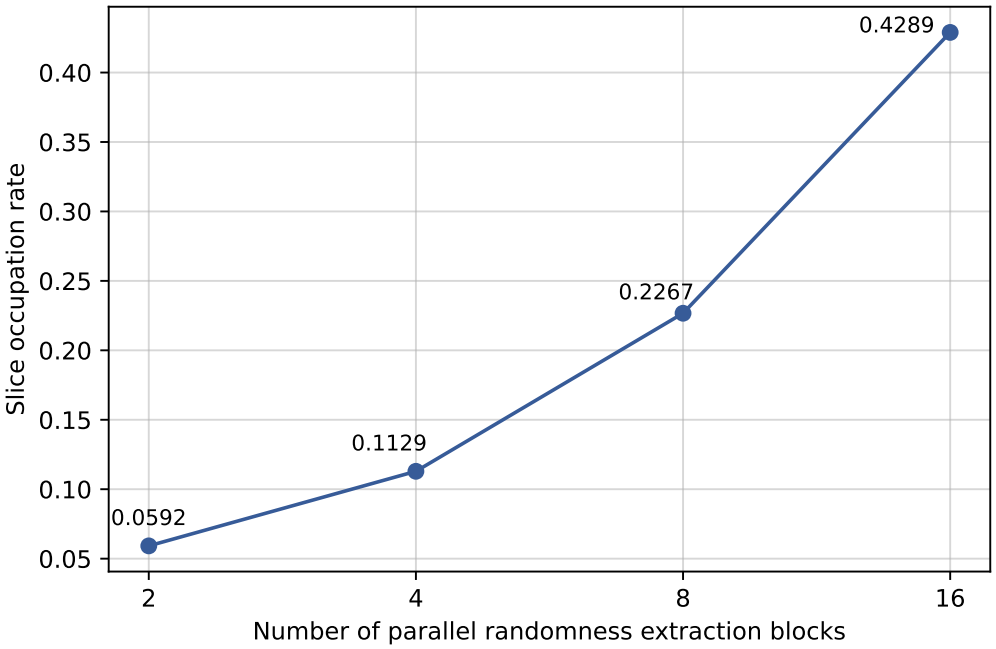


Figure 4.17: Slice occupation rate for a Toeplitz matrix of dimensions $m = 192$ and $n = 390$ with $G = 10$ the number of sub-matrices used for the multiplication in each block.

4.3.2 Throughput

In this section the circuit performance is analyzed, in terms of rate of extracted bits. In order to do so, the number of clock cycles elapsed in each state of the FSMs are analyzed.

Notice that this analysis refers to the online implementation, as throughput loses its meaning in an offline configuration.

The first action that is done by the circuit, after the start signal has been asserted, is to write the register containing the Toeplitz matrix. Consider that the amount of bits written at every clock cycle is $(Sj)/C$, with S the sampling frequency, C the FPGA clock frequency and j the ADC resolution in terms of bits. Since the Toeplitz matrix register has a width of $m + n - 1$ bits, the amount of clock cycles needed is:

$$\left\lceil \frac{C(n+m-1)}{Sj} \right\rceil \tag{4.4}$$

with $\lceil x \rceil$ the smallest integer greater or equal to the number x. Next, the extractor FSM is designed to immediately write the raw data registers of the parallel blocks instantiated. Recall that the system is designed to implement a total of $(Sj)/(CG)$ parallel extractors. The amount of clock cycles needed for each register is equal to:

$$\left\lceil \frac{Cn}{Sj} \right\rceil \tag{4.5}$$

Once a block has been written, the write-enable signal has to be routed to the successive module. This operation requires one clock cycle. Therefore considering as a computational cycle a period in which every parallel module has computed a result, a total of Sj/CG clock cycles are used to change the write-enable signal. The last missing stage for the extractor FSM is the one in which the machine waits for the first computational module to produce a result, before starting again to write the raw data registers. The amount of clock cycles that this waiting stage lasts, strictly depends on the time needed for the computational module to produce a result.

As mentioned in the previous chapter, the computational module starts working as soon as the raw data register is filled. Once this happens, the registers relative to the other two stages of the pipeline are activated. Therefore two clock cycles are employed before starting the computation.

The multiplication lasts a total of n/G clock periods. After this amount of time, one clock cycle is employed to assert the data valid signal. As a consequence, the result of the multiplication is available after $n/G + 3$ clock periods subsequent to the raw data register filling. However, before starting again to write the first block data register, the data valid signal has to be sampled, therefore an extra clock cycle is necessary.

Now that these durations are known, it is possible to compute the amount of clock periods the circuit waits before starting over a computation cycle. It will be equal to:

$$\frac{n}{G} + 3 + 1 - \left[\left(\frac{Sj}{CG} - 1\right)\left\lceil\frac{Cn}{Sj}\right\rceil + \frac{Sj}{CG}\right] \tag{4.6}$$

At this point every element to compute the throughput is present: let's not consider the period in which the Toeplitz matrix register is written, since it is a procedure performed one time at the start, and consider that a computation cycle has been already done once. This last aspect is relevant because in the first cycle, while the first block is being written, the others are idle so the parallelization is not being exploited. In normal operational conditions, the other blocks are performing the multiplication. Anyway, with these conditions a total of:

$$\frac{Sj}{CG}m \tag{4.7}$$

bits are given as output in a period of:

$$\begin{aligned}&\left\{\frac{n}{G} + 3 + 1 - \left[\left(\frac{Sj}{CG} - 1\right)\left\lceil\frac{Cn}{Sj}\right\rceil + \frac{Sj}{CG}\right]\right\} + \left(\frac{Sj}{CG}\right)\left(\left\lceil\frac{Cn}{Sj}\right\rceil + 1\right) \simeq \\ &\simeq \left\{\frac{n}{G} + 3 + 1 - \left[\left(\frac{Sj}{CG} - 1\right)\left\lceil\frac{Cn}{Sj}\right\rceil + \frac{Sj}{CG}\right]\right\} + \frac{n}{G}\end{aligned} \tag{4.8}$$

corresponding to the time in which the circuit waits before starting the new computational cycle summed to the time needed to fill every data register and change the write-enable signal. The approximation considers $\lceil (Cn)/(Sj) \rceil$ an integer number much greater than 1.

This complicated formula can be simplified if a large enough number of parallel blocks is instantiated. If this is the case, the waiting period between computational cycles can be neglected and the final formula for the throughput becomes:

$$\frac{Sj}{CG}m \cdot \frac{G}{n} \cdot C = Sj\frac{m}{n} \tag{4.9}$$

It is possible to observe that, with these appropriate parameters, the throughput depends only on the characteristics of the ADC and the ratio of extracted bits.

While on one hand this result proves the possibility of using the FPGA to build a high-rate randomness extraction system, on the other hand it has to be taken with care. This is because the amount of parallel blocks instantiated is directly proportional to the ratio between the sampling frequency of the ADC and the clock frequency of the FPGA. This last variable cannot be increased indefinitely. In fact a very fast clock, if available, may induce a failing in the timing requirements

of the circuit. Given this fact, the solution may seem to instantiate the largest amount possible of parallel blocks. However the FPGA capacity, in terms of slices, is limited and the occupation of them becomes greater both with an increase of the parallel blocks and with an increase of the dimensions of the Toeplitz matrix (refer to the plots in the previous discussion).

It can also be noticed that the throughput increases with the ADC resolution. An increase of this parameter also implies a larger number of instantiated parallel blocks. The choice of an ADC with a larger resolution may be more convenient than one with a higher sampling rate, as a large S/C ratio requires the instantiation of cascaded asynchronous FIFOs to bridge the high speed gap.

As an example, consider a scenario in which high-end hardware is used, including an FPGA that is able to instantiate a circuit that performs randomness extraction with Toeplitz matrices of dimensions $m \times n$, with $m = 2048$ and $n = 4096$. The amount of sub-matrices used for the computation is $G = 12$, and the clock frequency is $C = 100$ MHz. Consider also that this FPGA is interfaced with an ADC that samples at $S = 1$ GHz with a resolution of $j = 12$ bits. According to the implementation of this work, the number of instantiated parallel blocks is $Sj/CG = 10$ and the real-time generation rate is $Sjm/n = 6$ Gbps.

It can be concluded that the system has to be carefully designed but makes it possible to obtain high randomness extraction rates that are compliant with real-world application requirements.

Chapter 5

Conclusions and Future Developments

This work aimed to design a high-rate randomness extraction circuit with an FPGA and to validate the results using a trusted-device phase fluctuations-based QRNG and a source-device-independent vacuum state heterodyne measurement-based QRNG. The two implementations have been designed to produce bit sequences, consistently with the mathematical models describing them.

The first one exploits the interaction of photons emitted with spontaneous emission in a laser diode, that are recombined in an incoherent manner through a beam splitter and a delay line. The resulting numbers appear to be compliant with the Gaussian distribution expected from the mathematical model but are affected by a compression of the negative voltage values in a limited range of values. Moreover, the autocorrelation function reveals a high degree of correlation between the bits and the occurrence of periodic patterns.

The second implementation uses a heterodyne measurement scheme to sense the fluctuations of the vacuum state, where a closed fiber cable and a local oscillator are connected to a 90° optical hybrid. Two balanced photodetectors are used for the measurement of each of the quadratures of the electromagnetic field. This type of implementation allows to set a lower bound in the conditional min-entropy, by a simple estimation of the variance of the signal as a function of the optical power of the local oscillator. The autocorrelation function on the data obtained with this implementation has shown a high degree of short-term correlations between the bits.

The negative characteristics present in the sequences produced by the two QRNGs are mainly caused by imperfections in the experimental setups, which induce correlations between the bits generated by the actual quantum mechanical phenomenon and classical noise. For this reason, randomness extraction is required,

which is a mathematical operation that takes a bit string and yields a shorter sequence with a higher degree of randomness. In this work, Toeplitz hashing has been chosen as the extractor, for its efficient hardware implementation. It consists of a matrix-vector product between the raw data and a Toeplitz matrix, whose ratio of rows to columns is determined by the min-entropy of the starting sequence.

This type of post-processing has been implemented on an FPGA. This device, together with a CPU, is included in a SoC, which is mounted on an evaluation board. The circuit for the randomness extraction is composed of a three-stage pipeline that allows for the efficient computation of the matrix-vector product between a Toeplitz matrix and a raw data bit string. This architecture is instantiated multiple times to deliver the parallel processing of multiple data blocks, achieving higher rates of extracted bits. FSMs are used to manage both the pipeline and the operation of the parallel blocks.

This circuit is used in two different implementations, supporting an online and an offline functionality. The first demonstrates the feasibility of using the circuit as high-rate and real-time randomness extractor. In fact, the throughput computation has shown that with an efficient ADC, the system is capable of reaching bit rates of the order of Gbps, which are compatible with real-world applications. In the second implementation, the system processes data stored on an SD card, which were previously acquired from the QRNG, using a high-sampling-rate ADC. After randomness extraction, the results are saved on a file on the same storage device.

In order to estimate the quality of randomness of the extracted bits, the autocorrelation function has been computed, showing the reduction of previously present correlations or recurring patterns in the sequences produced by both QRNGs. Moreover the bit strings have been analyzed with the NIST test suite and the results show uniformly distributed $P - values$ and an acceptable proportion of passed tests. This means that no claim can be made on the fact that the numbers produced are not random. These tests do not certify the true randomness of the generated bits, since they can also be passed by PRNGs. However, such conclusion cannot be drawn by any type of analysis, therefore they still provide a good measure of the statistical behavior of the sequences and represent a first step to the evaluation of their randomness.

Future developments of this work may include improvements in the management of the FSM, to reduce the use of clock cycles that are not directly employed for the computation algorithm. Another possibility would be to design a different and more efficient algorithm for randomness extraction, which would increase even more the bit rate capabilities.

From the security point of view an improvement can be made by adding a real-time seed updating functionality. The idea would be to change the seed that generates the Toeplitz matrix at each successive post-processing operation, in order for them to be considered as independent and thereby addressing a possible security

vulnerability.

An interesting future development would be also the integration of a QRNG and the post-processing stage described in this work with a QKD system. FPGAs are a powerful tool also for high-speed generation of signals for QKD protocols. Integrating such devices with the randomness extraction circuit, either on the same or a different FPGA, would allow for the realization of a theoretically proven communication protocol for key exchange, where the keys are generated in real time by a QRNG with all the security implications it guarantees.

It would also be interesting the employment of this system for simulations and in particular in the context of entropy generation for supercomputers or virtual machines, where the need for a high bit-rate of truly random numbers is crucial. With its use, the absence of correlations may produce more reliable simulation outcomes.

In conclusion, it is important to mention that the use of the circuit for randomness extraction is not limited to the field of QNRGs but it may find space in cryptographic applications that require high-rate hashing . In fact, hash functions are fundamental to ensure information integrity in authenticated channels and an efficient hardware-based solution may be crucial in critical contexts.

Appendix A

Randomness Testing

In the context of random number generation, several test batteries are present to evaluate the quality of the produced sequences and some of the most common, as can be seen for instance in [27] and [39], are NIST, Diehard and ENT.

In this work the former has been considered and in this section a brief explanation on each test will be given but before a general context on randomness testing is provided.

A.1 Statistical Testing Concept

Testing a sequence of random numbers is a matter to be taken with care, as obviously, the assertion of randomness is given by the absence of recurring patterns inside the numbers. However, the number of possible recurring sequences is infinite, meaning that a test battery can never be considered complete but it is still a good metric for the quality of a RNG [40].

A certain statistical test is formulated to evaluate a null hypothesis, which in this case is if the sequence is random, and it is associated with an alternative hypothesis, which states that the sequence is not random. Then, for each of the tests, a relevant statistics is determined to assert if the null hypothesis is rejected or not. This relevant statistic has a certain distribution of values that can be compared with a theoretical distribution (for instance standard normal or χ^2 distribution), from which a critical value is extracted. If the test statistics fall outside of the bounds set by this critical value, then the null hypothesis is rejected. The rejection of the null hypothesis for a sequence which was truly random is referred to as a type I error, while the acceptance of the null hypothesis when the sequence was in reality non-random, is called a type II error. The probability that a type I error occurs is called level of significance of the test, it is usually referred to as α and has commonly a value of 0.01. Notice that this parameter is set by the user and takes

values based on the level of confidence needed to state that the sequence under test is random. On the other hand, the probability that a type II error occurs is usually referred to as β but its value is often obtained by fixing the length of the sequence under test and its level of significance. This is because it is hard to fix this parameter, since there are infinite ways in which a sequence can be non-random [40].

After the test, a parameter called *P-value* is extracted based on the resulting statistical values of the test and it is defined as the probability that a perfect random number generator would produce a sequence that is less random than the one under test. What is done to state if the null hypothesis is confirmed or not, is to compare this *P-value* with the level of significance of the test: if $P\text{-}value \geq \alpha$ then the null hypothesis is accepted and viceversa. In other words, the *P-value* measures how anomalous the observed data is under the assumption that the sequence is random, while α quantifies the tolerance for mistakenly rejecting a truly random sequence (so the probability of a type I error). Therefore, a sequence is rejected if the *P-value* indicates that the data are more extreme than what it is accepted as likely under randomness, which is instead determined by α [40].

A.2 Test Battery

The NIST test suite is composed of 15 tests, which are briefly described below [40]:

1. Frequency (Monobit) Test: this test focuses on the proportion of zeros and ones in the sequence. The theoretical distribution of reference is composed of an almost equal number of zeros and ones, as they have the same probabilities of being generated. For this reason the closeness of the fraction of ones present should be close to 1/2 in the tested sequence.

2. Frequency Test Within a Block: this test focuses on the proportion of ones in M-bit blocks. The test is analogous to the Monobit one but considering bit blocks, instead of the entire sequence.

3. Runs Test: this test focuses on the number of runs in the sequence, with runs being an uninterrupted sequence of identical bits, and this number is compared with the theoretical value expected for a random sequence.

4. Test for the Longest Run of Ones in a Block: this test focuses, as the name says, on the identification of the longest run of ones within M-bit blocks and consequent comparison with the expected theoretical case.

5. Binary Matrix Rank Test: this test focuses on the rank of disjoint sub-matrices of the entire sequence, which falls to an identification of a linear dependence between sub-strings of the sequence.

6. Discrete Fourier Transform (Spectral) Test: this test focuses in the identification of periodic patterns inside the Discrete Fourier Transform of the sequence.

7. Non-Overlapping Template Matching Test: this test focuses in the number of occurrences of pre-specified strings. By analysis of blocks of bits, correspondence with the searched pattern is investigated. In case a match occurs the search starts again from the first bit after the found pattern (that is the reason why it is deemed as non-overlapping)

8. Overlapping Template Matching Test: this test has the same purpose as the Non-Overlapping Template Matching Test, with the difference that whether a match has occurred or not, the search always happens by shifting a window, of length in bits equal to the length of the target string, by one bit (that is the reason why it is deemed as overlapping).

9. Maurer's "Universal Statistical" Test: this test focuses on the number of bits present between matching patterns. The objective is to state if the sequence can be compressed without losing much information, which in information theory means that the sequence is non-random, since its content is highly predictable.

10. Linear Complexity Test: this test focuses on the length of a LFSR. What this means is that, given the sequence under test, the length of the shortest LFSR needed to reproduce such sequence is computed and this parameter is referred to as linear complexity. A longer LFSR means that the sequence appears to be more random than one with a shorter LFSR.

11. Serial Test: this test focuses on computing the number of occurrences of overlapping patterns of a fixed length and stating if that number is in line with what would be expected in a random sequence, which is uniformity or in other words that each pattern should appear with the same frequency as any other.

12. Approximate Entropy Test: this test still focuses on the number of occurrences of overlapping patterns but now blocks whose lengths varies of one bit are considered and are compared with the theoretical random sequence.

13. Cumulative Sums Test: This test focuses on the maximal excursion from zero of a random walk defined by cumulatively summing the digits in the sequence, which, instead of being zeros and ones, are now (–1, +1) respectively. By cumulative sum random walk, in this context, it is meant the random process in which the decision to either add +1 or –1 to the current state is made with

probability 1/2 [41]. A random sequence should have the excursion of the random walk around zero.

14. Random Excursion Test: this test still focuses on a cumulative sum random walk. Now cycles are considered, which are defined as sequences of steps that start and return to the origin, and the number of visits to certain states are counted within a cycle and are compared to the theoretical expected value. The test consists in eight tests, one for each state from -4 to 4 (excluding 0).

15. Random Excursions Variant Test: this test focuses on the amount of times a state is visited in a cumulative sum random walk. It is composed of eighteen tests, one for each state from -9 to 9 (excluding 0).

A.3 Results Interpretation

According to the NIST documentation, it is necessary to analyze a certain amount of sequences with the test battery. As a consequence, numerous $P-values$ are obtained for each test. In order to understand if claims can be made on the fact that the bits produced are not random, it is necessary to understand the ratio of sequences that passed each test. It is indicated that such ratio should be at least:

$$(1-\alpha) - 3\sqrt{\frac{\alpha(1-\alpha)}{m}} \tag{A.1}$$

where α is the level of significance and m is the number of sequences.

It is also necessary that the $P-values$ obtained for each test are distributed uniformly between [0,1]. In order to verify it, these $P-values$ can be plotted with an histogram of 10 bins across that range or they can be tested with a χ^2 test [40]. An alternative can also be the Kolmogorov-Smirnov test [25].

Bibliography

[1] Charles H. Bennett and Gilles Brassard. «Quantum cryptography: Public key distribution and coin tossing». In: *Theoretical Computer Science* 560 (Dec. 2014), pp. 7–11. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2014.05.025. URL: http://dx.doi.org/10.1016/j.tcs.2014.05.025 (cit. on p. 1).

[2] Vaisakh Mannalatha, Sandeep Mishra, and Anirban Pathak. «A comprehensive review of quantum random number generators: concepts, classification and the origin of randomness». In: *Quantum Information Processing* 22.12 (Dec. 2023). ISSN: 1573-1332. DOI: 10.1007/s11128-023-04175-y. URL: http://dx.doi.org/10.1007/s11128-023-04175-y (cit. on pp. 1–6).

[3] Bing Qi, Yue-Meng Chi, Hoi-Kwong Lo, and Li Qian. «High-speed quantum random number generation by measuring phase noise of a single-mode laser». In: *Optics Letters* 35 (Jan. 2010), pp. 312–314. DOI: 10.1364/OL.35.000312 (cit. on pp. 1, 11).

[4] Christian Gabriel, Christoffer Wittmann, Denis Sych, Ruifang Dong, Wolfgang Mauerer, Ulrik L. Andersen, Christoph Marquardt, and Gerd Leuchs. «A generator for unique quantum random numbers based on vacuum states». In: *Nature Photonics* 4.10 (Oct. 2010), pp. 711–715. DOI: 10.1038/nphoton.2010.197 (cit. on pp. 1, 15).

[5] Bing Qi, Yue-Meng Chi, Hoi-Kwong Lo, and Li Qian. «High-speed quantum random number generation by measuring phase noise of a single-mode laser». In: *Opt. Lett.* 35.3 (Feb. 2010), pp. 312–314. DOI: 10.1364/OL.35.000312. URL: https://opg.optica.org/ol/abstract.cfm?URI=ol-35-3-312 (cit. on p. 1).

[6] Xiongfeng Ma, Feihu Xu, He Xu, Xiaoqing Tan, Bing Qi, and Hoi-Kwong Lo. «Postprocessing for quantum random-number generators: Entropy evaluation and randomness extraction». In: *Physical Review A* 87.6 (June 2013). ISSN: 1094-1622. DOI: 10.1103/physreva.87.062327. URL: http://dx.doi.org/10.1103/PhysRevA.87.062327 (cit. on pp. 1, 26–29).

[7] Anindya De, Christopher Portmann, Thomas Vidick, and Renato Renner. «Trevisan's Extractor in the Presence of Quantum Side Information». In: *SIAM Journal on Computing* 41.4 (Jan. 2012), pp. 915–940. ISSN: 1095-7111. DOI: 10.1137/100813683. URL: http://dx.doi.org/10.1137/100813683 (cit. on p. 1).

[8] Hugo Krawczyk. «LFSR-based Hashing and Authentication». In: *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*. Vol. 839. Lecture Notes in Computer Science. Springer, 1994, pp. 129–139. DOI: 10.1007/3-540-48658-5_15 (cit. on pp. 1, 28).

[9] Xiaomin Guo, Fading Lin, Jiehong Lin, Zhijie Song, Yue luo, Qiqi Wang, and Yanqiang Guo. *Parallel and real-time post-processing for quantum random number generators*. 2024. arXiv: 2403.19479 [quant-ph]. URL: https://arxiv.org/abs/2403.19479 (cit. on pp. 1, 8, 39, 40).

[10] Xiangyu Wang, Yichen Zhang, Song Yu, and Hong Guo. «High-Speed Implementation of Length-Compatible Privacy Amplification in Continuous-Variable Quantum Key Distribution». In: *IEEE Photonics Journal* 10.3 (June 2018), pp. 1–9. ISSN: 1943-0655. DOI: 10.1109/jphot.2018.2824316. URL: http://dx.doi.org/10.1109/JPHOT.2018.2824316 (cit. on p. 1).

[11] Xiaoguang Zhang, You-Qi Nie, Hao Liang, and Jun Zhang. «FPGA implementation of Toeplitz hashing extractor for real time post-processing of raw random numbers». In: *2016 IEEE-NPSS Real Time Conference (RT)*. 2016, pp. 1–5. DOI: 10.1109/RTC.2016.7543094 (cit. on pp. 1, 29, 33–35).

[12] R. Adami. «Lecture notes for the course "Quantum Cryptography"». 2024-2025 (cit. on p. 2).

[13] R Sri Durga, C K Rashmika, Oruganti N V Madhumitha, D G Suvetha, Bandaru Tanmai, and N Mohankumar. «Design and Synthesis of LFSR based Random Number Generator». In: *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. 2020, pp. 438–442. DOI: 10.1109/ICSSIT48917.2020.9214240 (cit. on p. 2).

[14] Kamalika Bhattacharjee and Sukanta Das. «A search for good pseudo-random number generators: Survey and empirical studies». In: *Computer Science Review* 45 (2022), p. 100471. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2022.100471. URL: https://www.sciencedirect.com/science/article/pii/S1574013722000144 (cit. on p. 2).

[15] Miguel Herrero-Collantes and Juan Carlos Garcia-Escartin. «Quantum random number generators». In: *Rev. Mod. Phys.* 89 (1 Feb. 2017), p. 015004. DOI: 10.1103/RevModPhys.89.015004. URL: https://link.aps.org/doi/10.1103/RevModPhys.89.015004 (cit. on pp. 3, 8).

[16] Manabendra Nath Bera, Antonio Acín, Marek Kuś, Morgan W Mitchell, and Maciej Lewenstein. «Randomness in quantum mechanics: philosophy, physics and technology». In: *Reports on Progress in Physics* 80.12 (Nov. 2017), p. 124001. ISSN: 1361-6633. DOI: 10.1088/1361-6633/aa8731. URL: http://dx.doi.org/10.1088/1361-6633/aa8731 (cit. on p. 4).

[17] Helmut Schmidt. «Quantum-Mechanical Random-Number Generator». In: *Journal of Applied Physics* 41.2 (Feb. 1970), pp. 462–468. ISSN: 0021-8979. DOI: 10.1063/1.1658698. eprint: https://pubs.aip.org/aip/jap/article-pdf/41/2/462/18352678/462_1_online.pdf. URL: https://doi.org/10.1063/1.1658698 (cit. on p. 4).

[18] S. Pironio et al. «Random numbers certified by Bell's theorem». In: *Nature* 464.7291 (Apr. 2010), pp. 1021–1024. ISSN: 1476-4687. DOI: 10.1038/nature09008. URL: https://doi.org/10.1038/nature09008 (cit. on p. 4).

[19] L. Columbo. «Lecture slides for the course "Quantum Photonics"». 2023-2024 (cit. on pp. 4, 10, 16).

[20] Thomas Jennewein, Ulrich Achleitner, Gregor Weihs, Harald Weinfurter, and Anton Zeilinger. «A fast and compact quantum random number generator». In: *Review of Scientific Instruments* 71.4 (Apr. 2000), pp. 1675–1680. ISSN: 0034-6748. DOI: 10.1063/1.1150518. eprint: https://pubs.aip.org/aip/rsi/article-pdf/71/4/1675/19183814/1675_1_online.pdf. URL: https://doi.org/10.1063/1.1150518 (cit. on pp. 4, 5).

[21] Xiongfeng Ma, Xiao Yuan, Zhu Cao, Bing Qi, and Zhen Zhang. «Quantum random number generation». In: *npj Quantum Information* 2.1 (June 2016). ISSN: 2056-6387. DOI: 10.1038/npjqi.2016.21. URL: http://dx.doi.org/10.1038/npjqi.2016.21 (cit. on pp. 6, 7).

[22] Giuseppe Vallone, Davide G. Marangon, Marco Tomasin, and Paolo Villoresi. «Quantum randomness certified by the uncertainty principle». In: *Phys. Rev. A* 90 (5 Nov. 2014), p. 052327. DOI: 10.1103/PhysRevA.90.052327. URL: https://link.aps.org/doi/10.1103/PhysRevA.90.052327 (cit. on pp. 6, 15).

[23] Zhu Cao, Hongyi Zhou, and Xiongfeng Ma. «Loss-tolerant measurement-device-independent quantum random number generation». In: *New Journal of Physics* 17.12 (Dec. 2015), p. 125011. DOI: 10.1088/1367-2630/17/12/125011. URL: https://dx.doi.org/10.1088/1367-2630/17/12/125011 (cit. on p. 6).

[24] Mario Stipčević and Çetin Koç. «True Random Number Generators». In: Nov. 2014, pp. 275–315. ISBN: 978-3-319-10682-3. DOI: 10.1007/978-3-319-10683-0_12 (cit. on pp. 7, 8).

[25] Ziyong Zheng, Yichen Zhang, Weinan Huang, Song Yu, and Hong Guo. «6 Gbps real-time optical quantum random number generator based on vacuum fluctuation». In: *Review of Scientific Instruments* 90.4 (Apr. 2019). ISSN: 1089-7623. DOI: 10.1063/1.5078547. URL: http://dx.doi.org/10.1063/1.5078547 (cit. on pp. 8, 32, 78).

[26] C. Henry. «Theory of the linewidth of semiconductor lasers». In: *IEEE Journal of Quantum Electronics* 18.2 (1982), pp. 259–264. DOI: 10.1109/JQE.1982.1071522 (cit. on p. 9).

[27] Jinlu Liu, Jie Yang, Zhengyu Li, Qi Su, Wei Huang, Bingjie Xu, and Hong Guo. «117 Gbits/s Quantum Random Number Generation With Simple Structure». In: *IEEE Photonics Technology Letters* 29.3 (2017), pp. 283–286. DOI: 10.1109/LPT.2016.2639562 (cit. on pp. 11, 12, 75).

[28] Jialiang Li, Zitao Huang, Chunlin Yu, Jiajie Wu, Tongge Zhao, Xiangwei Zhu, and Shihai Sun. «Quantum random number generation based on phase reconstruction». In: *Optics Express* 32.4 (Jan. 2024), p. 5056. ISSN: 1094-4087. DOI: 10.1364/oe.515390. URL: http://dx.doi.org/10.1364/OE.515390 (cit. on p. 11).

[29] Ivan B. Djordjevic. «Chapter 5 - Quantum detection and quantum communication». In: *Quantum Communication, Quantum Networks, and Quantum Sensing*. Ed. by Ivan B. Djordjevic. Academic Press, 2023, pp. 157–214. ISBN: 978-0-12-822942-2. DOI: https://doi.org/10.1016/B978-0-12-822942-2.00011-X. URL: https://www.sciencedirect.com/science/article/pii/B978012822942200011X (cit. on pp. 16, 17).

[30] R. Proietti. «Lecture slides for the course "Quantum Communications and Networks"». 2024-2025 (cit. on pp. 17, 18).

[31] Marco Avesani, Davide G. Marangon, Giuseppe Vallone, and Paolo Villoresi. «Source-device-independent heterodyne-based quantum random number generator at 17 Gbps». In: *Nature Communications* 9.1 (2018), p. 5365. ISSN: 2041-1723. DOI: 10.1038/s41467-018-07585-0. URL: https://doi.org/10.1038/s41467-018-07585-0 (cit. on pp. 17–20).

[32] Arti Badhoutiya, Zainb Jaffer, Heba Mohammed Hussein, Ashima Juyal, Manisha Mittal, and Rohit Anand. «Field Programmable Gate Array: An Extensive Review, Recent Trends, Challenges and Applications». In: *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*. 2024, pp. 1084–1090. DOI: 10.23919/INDIACom61295.2024.10498934 (cit. on p. 23).

[33] G. Turvani. «Lecture slides for the course "Quantum Hardware Design and Optimization"». 2024-2025 (cit. on pp. 23, 24).

[34] AMD (Xilinx). *7 Series CLB Features. 7 Series FPGAs Configurable Logic Block User Guide (UG474)*. Rev.1.9. AMD Inc. 2025. URL: https://docs.amd.com/r/en-US/ug474_7Series_CLB/7-Series-CLB-Features (cit. on p. 24).

[35] Xilinx Inc. *Vivado Design Suite User Guide: Design Flows Overview*. UG892 (v2025.1). Xilinx. 2025. URL: https://docs.amd.com/r/en-US/ug892-vivado-design-flows-overview/Design-Flows (cit. on pp. 24, 25).

[36] Rudolf J. Freund, William J. Wilson, and Donna L. Mohr. «CHAPTER 2 - Probability and Sampling Distributions». In: *Statistical Methods (Third Edition)*. Ed. by Rudolf J. Freund, William J. Wilson, and Donna L. Mohr. Third Edition. Boston: Academic Press, 2010, pp. 67–124. ISBN: 978-0-12-374970-3. DOI: https://doi.org/10.1016/B978-0-12-374970-3.00002-0. URL: https://www.sciencedirect.com/science/article/pii/B9780123749703000020 (cit. on p. 27).

[37] J.Lawrence Carter and Mark N. Wegman. «Universal classes of hash functions». In: *Journal of Computer and System Sciences* 18.2 (1979), pp. 143–154. ISSN: 0022-0000. DOI: https://doi.org/10.1016/0022-0000(79)90044-8. URL: https://www.sciencedirect.com/science/article/pii/0022000079900448 (cit. on p. 28).

[38] *PuTTY: a free SSH and Telnet client*. https://www.chiark.greenend.org.uk/~sgtatham/putty/ (cit. on p. 48).

[39] Hong Guo, Wenzhuo Tang, Yu Liu, and Wei Wei. «Truly random number generation based on measurement of phase noise of a laser». In: *Physical Review E* 81.5 (May 2010). ISSN: 1550-2376. DOI: 10.1103/physreve.81.051137. URL: http://dx.doi.org/10.1103/PhysRevE.81.051137 (cit. on p. 75).

[40] Andrew Rukhin et al. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. NIST Special Publication 800-22rev1a. Revised version 1a. Gaithersburg, MD: National Institute of Standards and Technology, 2010. URL: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf (cit. on pp. 75, 76, 78).

[41] Feng Xia, Jiaying Liu, Hansong Nie, Yonghao Fu, Liangtian Wan, and Xiangjie Kong. «Random Walks: A Review of Algorithms and Applications». In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 4.2 (Apr. 2020), pp. 95–107. ISSN: 2471-285X. DOI: 10.1109/tetci.2019.2952908. URL: http://dx.doi.org/10.1109/TETCI.2019.2952908 (cit. on p. 78).