Ultra-fast Quantum Random Number Generator

Shi Yicheng

A thesis submitted for the degree of
Master of Science

Department of Physics

National University of Singapore

2016

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously

Shi Yicheng

March 17, 2016

Acknowledgements

I would like to express my deepest gratitude to professor Christian Kurtsiefer, Ms Chng Mei Yuen Brenda and all the friends of the CQT Quantum Optics group for their support and advice on the project, and for patiently teaching me all the interesting I learned in the laboratory.

Despite the nature of this project, I am in fact a big fan of the idea *Superdeterminism*. In brief it states that *everything is completely deterministic* and denies the existence of *free will* and *randomness*. Unfortunately this hypothesis cannot be logically proved or experimentally tested, and it may never be. We may never know for sure how mysterious nature really is, but that's exactly what makes science so attractive.

Again, thank everyone for such a wonderful research experience.

Abstract

We describe a series of Randomness Extractors for removing bias and residual correlations in random numbers generated from measurements on noisy physical systems. The structures of the randomness extractors are based on Linear Feedback Shift Registers (LFSR). This leads to a significant simplification in the implementation of randomness extractors.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

People have been fascinated and puzzled by the concept of randomness since the earliest days of history. From ancients' reading of the innards of birds for divination to the tossing of dice in the game of gambling, all the way to modern day risk assessments and market investments, we have put ourselves into the hands of chance. The ancient Greek philosophers were among the first to discuss the concept of randomness and chance and link it to divinity [1]. Alongside the establishment of probability theory, randomness has been a subject receiving continuous attention. In the meantime, its usefulness was also gradually found in various fields of science and industry.

The study of randomness became crucial with the development of modern cryptography, when people realized that generating true randomness is a fundamental task in essentially all information security protocols [2, 3]. However, this seemingly easy task is in fact surprisingly challenging. Plenty of methods have been proposed and implemented, both arithmetical and physical, and yet few of them are considered truly satisfying. While the quality of randomness generation is of concern, faster generation speed is also demanded by most modern communication protocols. Hence, having a reliable, fast random number generator that produces true random numbers becomes one of the keystones of building a secure and efficient communication system, and this inspired various researches in the field.

1.1 Definition Of Random Numbers

The idea of a ‘random number’ is so simple that it can be understood by literally anyone: the toss of a coin gives you one random bit with two equal possible outcomes. We call it ‘random’ because we have no way of telling the result before it happens.

However, despite the simplicity of the idea, it is in fact difficult to give a truly satisfactory definition of random numbers. A comment made by D. H. Lehmer in 1951 addressed the issue as such [4]:

“A random sequence is a vague notion embodying the idea of a sequence in which each term is unpredictable to the uninitiated and whose digits pass a certain number of tests, traditional with statisticians and depending somewhat on the uses to which the sequence is to be put.”

The comment points out two major properties that should be possessed by a sequence of random numbers: unpredictability and a lack of structure in its statistics. No one should be able to predict the elements in a random number sequence: knowing the existing numbers should not leak any information about the next number in the sequence. On the other hand, the sequence should 'look random': it should (at least with a high probability) have similar statistics as one that is truly random. Even more strict arguments can be made indicating that the second property mentioned above is implied by the first: finding certain statistical structure in the data would increase the predictability of the sequence.

The description above is by no means mathematically rigorous, but a random number sequence having these properties should at least satisfy most real life applications, as will be introduced in the following session.

It is worth noting that other attempts are made in the field of computer science and information theory. Related works are done by Kolmogorov, Martin-Löf and Chaitin [5,6] interpret randomness in terms of computational complexity.

1.2 Applications Of Random Numbers

Random numbers are useful in many different applications, ranging from computer science to gaming industries.

The Monte-Carlo simulation is one of its well adopted applications in science. This method rely on using random sampling to obtain numerical results of a problem [7]. It is often used to simulate complex systems that are beyond the capabilities of analytical methods. The Monte-Carlo method usually requires a large amount of random numbers to seed the sampling process hence it relies a lot on the speed of random number generation.

In computer science, it is found that certain randomized algorithms, which require inputs of random numbers, perform better than their corresponding deterministic algorithms [4, 8]. Random number generators are required when running these algorithms. In fact, it is hard to find a standard compiler without a build-in random number generator.

1.2.1 Application in Cryptography

Perhaps the one field that relies on random numbers most is cryptography. The usage of random numbers in cryptography is a good reflection of the definition of random numbers and also provides some guide line for the designing of random number generators.

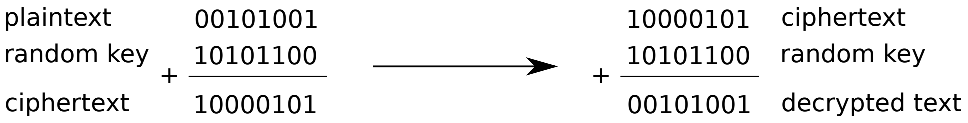

Figure 1.1: The One-Time Pad encryption protocol. The ciphertext is generated by XORing the plaintext with a randomly generated key, and is decrypted by XORing the same key once more.

One-Time Pad

As a first example, the One-Time Pad (OTP) is a cryptographic system developed in the early 20th century by Gilbert Vernam and Joseph Mauborgne [3]. The protocol is very simple:

1, Encode the plaintext in binary numbers;

2, Generate a key of random bits of the same length as the plaintext;

3, Obtain the ciphertext by performing a bitwise XOR between the plaintext and the key;

4, The ciphertext and the key are transmitted seprately to the receiver;

5, On the receiving side, the ciphertext is decrypted by performing another bitwise XOR between the ciphertext and the key.

An example is shown in Fig. 1.1. It is obvious that the OTP is an encryption system that largely depends on the generation of random numbers(bits). In fact, the OTP is considered unbreakable for ciphertext attacks if the random bits used for the encryption key are truly unpredictable random bits: the encrypted message will appear completely random and gives no information about the plaintext(well, except the length of the message) [2], and an eavesdropper will take forever to guess the content of the message. It was rumoured that a ‘hot line’ was established during the cold war between Washington D.C. and Moscow using the OTP for highest level of security.

However, the disadvantages of the one-time pad are fairly obvious: The protocol requires on a large amount of random bits being reliably generated. The rate of generation is the bottleneck of the entire scheme. Moreover, if the random number generator is jeopardised and becomes predictable, the system is no longer secure.

Key Generation

The one-time pad is a somewhat extreme protocol designed for ultimate security. The protocol is safe but not practical for most real life scenarios.

Most modern cryptographic protocols make use of pairs of encryption and decryption algorithms instead of directly XORing plaintext to the key. As shown in Fig. 1.2, these algorithms usually operate together with a pair of much shorter keys, whose values affect the algorithms: only people with the correct pair of keys can successfully encrypt or decrypt messages. Eavesdroppers who do not have access to the keys will be forced to guess their content in order to crack the ciphertext.

The security of a protocol rest in the design of the algorithms and more importantly, the choice of keys. It is in general a good practice to use completely random bits as the encryption key such that the system can only be attacked by brute force. For cryptographic system, it is vital that the key generation is truly unpredictable: not just statistically random, but truly uncorrelated to any outside information and takes maximum effort to be guessed.

1.3 Methods Of Generating Random Numbers

Much effort have been devoted into finding ways of generating good random numbers. Some very early work of the field include [9, 10]. There are two

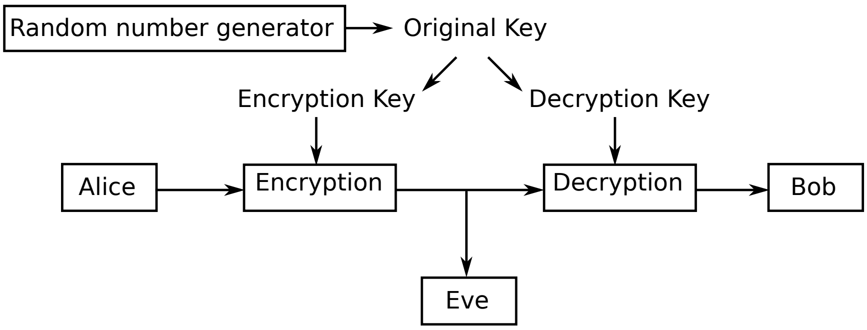

Figure 1.2: Usage of random number generator for key generation. The original key is usually a random string from a random number generator. The original key is manipulated and a pair of encryption/decryption keys are generated and distributed to the users (Alice and Bob). With the correct pair of keys, the encrypted messages can be safely transmitted and the eavesdropper (Eve) who does not have access to the keys cannot breach the protocol.

main types of random number generators, Pseudo-random number generator and Physical random number generator.

1.3.1 Pseudo-Random Number Generator

A pseudo-random number generator is in principle a deterministic algorithm producing an output stream that appears statistically random, i.e., the stream contains roughly the same number of 1s and 0s, does not seem to have correlations between the bits, etc. Some of the famous algorithms used for pseudo-random number generation include linear congruential generator[1] [2–4], Mersenne Twister [11] and the Blum-Blum-Shub(BBS) generator [2,3,12]. Well designed pseudo-random number generators usually have fast generation rates since they are completely based on arithmetic methods. These methods are most commonly used for Monte-Carlo simulations and general programming purposes.

[1] the linear congruential generator is in fact a family of generators based on similar methods.

However, the deterministic nature of the pseudo-random number generators suggests that their output can in principle be predicted and hence are not suitable for cryptographic purposes. On strict point of view, pseudo-random numbers are not random, or as John Von Neumann commented: *"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin"*.

1.3.2 Physical Random Number Generator

A physical random number generator (also known as Hardware random number generator) generates random numbers from physical processes. Instead of using deterministic algorithms, they make measurements on noisy physical systems and convert the results into random bits.

Various physical phenomena have been used as the source of randomness, ranging from radioactive decay events to atmospheric noise. Human behaviour related events can also be useful sources of randomness. In fact, in many Unix-like operating systems, a special device file named */dev/random* serves as a random number generator by collecting randomness from keystroke timings, mouse movements and other possible environmental noises [13].

The merit of using a physical random number generator is that the generated numbers are practically unpredictable: a real life physical system used for such purposes is simply too complex for anyone to predict. However, question still remains about whether all these systems can be considered truly random. It is often argued that most marcoscopic systems described by classical mechanics are still considered deterministic and are not truly random [14]. Apart from that, physical random number generators usually suffer from slow generation rates due to the limitation in detection and data processing.

1.4 Quantum Random Number Generator

Quantum Random Number Generators (QRNG) belong to a special class of physical random number generators where the source of randomness is the unpredictable outcome of a quantum measurement.

The concept of randomness is embedded in the basic postulates of quantum mechanics. A quantum system under measurement will randomly fall into one of its possible eigenstates and yield the corresponding outcome. It is certified by experiments that these measurement results can be truly non-deterministic [15, 16] and hence make them idea candidates for random number generation.

Randomness sources based on radioactive decays were used in many early implementations of QRNGs [17,18]. A decay event (typically α decay) occurs spontaneously in the nucleus and does not depend on external condition. The decay statistics of a radioactive sample thus can be recorded and used to generate good random numbers.

Quantum optical system is another popular choice of randomness. Different schemes use the randomness of a single photon scattered by a partially reflective mirror into either of two possible ports [19, 20], as demonstrated in Fig. 1.3. Since the transmission/reflection of the photon is intrinsically random due to the quantum nature of the process, the unpredictability of the generated numbers is ensured.

There are other implementations of QRNGs measuring the vacuum fluctuations of electromagnetic field [21, 22] or the intensity and phase noise of different light sources [23–25].

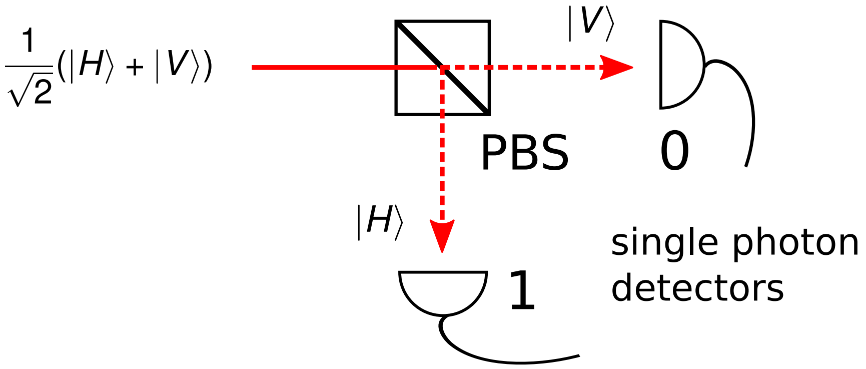

Figure 1.3: A quantum random number generator based on single photon events. A photon is prepared in a 45 degree polarization state and is sent through a polarisation beam splitter. The photon will have 50% chance being transmitted and another 50% being reflected. Two single photon detectors are used to count the photons coming from each port. Since the scattering of a photon is a quantum process, each event produces exactly 1 random bit

1.5 Statistical Tests Of Randomness

Besides the generation of random numbers, it is also important to have ways of testing them. Although in principle one cannot tell whether a recorded number sequence is truly random or not, the theory of probability and statistics does offer some quantitative measures for the likelihood of randomness, i.e. whether a given sequence "looks random" or not.

The tests of randomness follow a general procedure. The sample of numbers under test is first collected and manipulated to calculate certain statistics. These statistics could be standard ones such as mean, variance, autocorrelation, or some other self defined properties. In the mean time, one calculates the expectation value of the same statistics by assuming the numbers being truly random. These statistics obtained from the sample are compared with their expectation values and a final decision will be made: obviously, if the sample statistics are so far away from their expectations, then the numbers under test is most likely to be non-random; if the statistics fit closely to the expectation values, then one may choose to conclude that such a test cannot distinguish these numbers from truly random ones.

Lots of statistical tests have been created to assist the testing of random number generators. Some of the most famous test suites include the NIST Statistical Test Suite [26], Diehard/Dieharder Suite by Robert G. Brown [27], etc. Each of these test suites consists of dozens of carefully designed tests trying to probe possible statistical anomalies in the subjects, and they are often used to certify the performance of newly designed random number generators.

1.6 Thesis Outline

The purpose of this thesis is to document one specific design of quantum random number generator based on a homodyne measurement of the vacuum field.

The theory of vacuum fluctuation of electromagnetic fields and homodyne detection will be introduced in chapter 2, followed by implementation details and performance described in chapter 3. In chapter 4, I will introduce the method of quantifying the randomness from the source and introduce ways of extracting uniformly distributed random bits from the source. The thesis will be concluded by chapter 5 in which remarks and outlooks will be provided.

Chapter 2

Vacuum Fluctuations And Its Detection

2.1 Light As A Quantum Harmonic Oscillator

Light propagating in free space has long been recognized as an oscillating electromagnetic field. As an oscillation phenomenon, its mathematical description shares great resemblance to that of a harmonic oscillator. This resemblance is extended to the quantum regime and motivates the theory of quantization of the electromagnetic field.

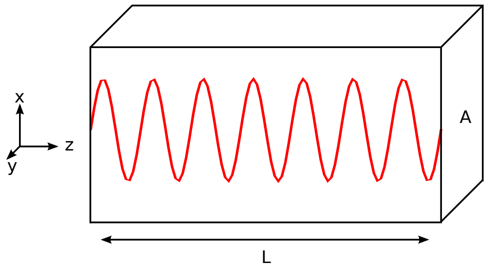

Figure 2.1: A linearly polarized EM wave in a cavity. The wave propagates along z axis and is linearly polarized along the x axis. The length of the cavity is L and its volume is $V = L \times A$. Such a cavity only supports discrete number of frequency modes.

A simple scenario to consider is a linearly polarized electromagnetic field enclosed in a cavity of length L and volume V, as illustrated in Fig. 2.1. It is apparent that such a cavity only supports EM field oscillations of certain angular frequencies $\omega = \frac{n\pi}{L}c$, and their corresponding wave vectors are $k = \frac{\omega}{c}$.

For a specific frequency mode ω, the total electromagnetic energy stored in the cavity has contribution from the electrical part and the magnetic part of the oscillation, and is expressed as:

$$E = \frac{1}{2}(\underbrace{\frac{V}{2}\epsilon_0{\mathcal{E}_0}^2\sin^2\omega t}_{\omega^2 q^2} + \underbrace{\frac{V{B_0}^2}{2\mu_0}\cos^2\omega t}_{p^2}) = \frac{1}{2}(p^2 + \omega^2 q^2) \tag{2.1}$$

For convenience, we regroup the terms and rename two variables p and q, where:

$$p = (\frac{V}{2\mu_0})^{\frac{1}{2}} B_0 \cos\omega t = (\frac{\epsilon_0 V}{2})^{\frac{1}{2}} \mathcal{E}_0 \cos\omega t \tag{2.2}$$

$$q = (\frac{\epsilon_0 V}{2\omega^2})^{\frac{1}{2}} \mathcal{E}_0 \sin\omega t \tag{2.3}$$

To see the merit of such arrangements, observe that:

$$p = \dot{q} \tag{2.4}$$

$$\dot{p} = -\omega^2 q \tag{2.5}$$

A comparison between the expressions above and the equation of motion of a harmonic oscillator is given below:

Harmonic Oscillator	Optical Field
$p = m\dot{x}$	$p = \dot{q}$
$\dot{p} = -m\omega^2 x$	$\dot{p} = -\omega^2 q$
$E = \frac{p^2}{2m} + \frac{1}{2}m\omega^2 x^2$	$E = \frac{1}{2}(p^2 + \omega^2 q^2)$
$\ldots$	
$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{1}{2}m\omega^2 \hat{x}^2$	$\hat{H} = \frac{1}{2}(\hat{p}^2 + \omega^2 \hat{q}^2)$

A important step here is to extend the similarity to quantum regime. The pair of variables p and q are replaced by operators $\hat{p}$ and $\hat{q}$ similar to the momentum and position operators of a quantum harmonic oscillator. The most direct consequence due to this is the quantization of energy spectrum:

$$E_n = (n + \frac{1}{2})\hbar\omega \tag{2.6}$$

At this point, the concept of photon is introduced: an energy quanta of electromagnetic field that equals to $\hbar\omega$. A quantum state of light can now be expressed as a linear combination of number states $|n\rangle$, each representing the existence of n photons in the optical mode.

Similarly, the uncertainty relation is also inherited in this case:

$$\Delta p \Delta q \geq \frac{\hbar}{2} \tag{2.7}$$

In many literatures [28, 29], a new pair of dimensionless variables known as the field quadratures are defined for the convenience of discussion. The field quadratures are defined in terms of the p and q variables in Eq. 2.2:

$$X_1 = (\frac{\omega}{2\hbar})^{\frac{1}{2}}q, \quad X_2 = (\frac{1}{2\hbar\omega})^{\frac{1}{2}}p \tag{2.8}$$

and naturally, the quadrature operators are defined for the quantum case:

$$\hat{X}_1 = (\frac{\omega}{2\hbar})^{\frac{1}{2}}\hat{q}, \quad \hat{X}_2 = (\frac{1}{2\hbar\omega})^{\frac{1}{2}}\hat{p} \tag{2.9}$$

and the Hamiltonian now can be written as:

$$\hat{H} = \hbar\omega(\hat{X_1}^2 + \hat{X_2}^2) \tag{2.10}$$

and the uncertainty relation is now expressed as:

$$\Delta X_1 \Delta X_2 \geq \frac{1}{4} \tag{2.11}$$

2.2 Coherent State And Vacuum State

2.2.1 Coherent State Of Light

The coherent state of light is a superposition of photon number states:

$$|\alpha\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_n \frac{\alpha^n}{(n!)^{\frac{1}{2}}} |n\rangle \tag{2.12}$$

and has a poissonian photon number distribution:

$$P(n) = |\langle n|\alpha\rangle|^2 = e^{-\frac{|\alpha|^2}{2}} \frac{|\alpha|^{2n}}{n!} \tag{2.13}$$

This state minimizes the uncertainty relation in the field quadratures:

$$\Delta X_1 = \Delta X_2 = \frac{1}{2} \tag{2.14}$$

Fig. 2.2 is a phasor diagram of the coherent state in Eq. 2.12. A coherent state is characterized by a complex number α defined as:

$$\alpha = X_1 + iX_2 = |\alpha|e^{i\phi} \tag{2.15}$$

and the square of its magnitude is given by:

$$|\alpha|^2 = {X_1}^2 + {X_2}^2 = \bar{n} \tag{2.16}$$

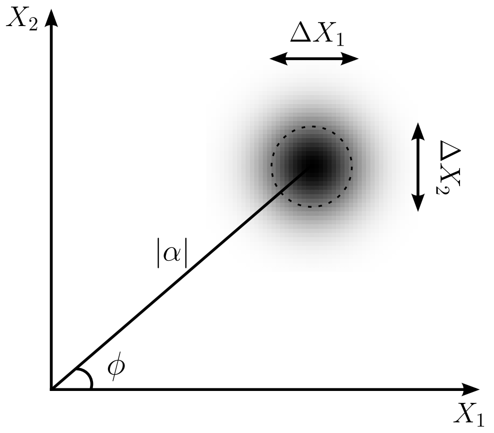

Figure 2.2: Representation of a coherent state in the phasor diagram. The state has a mean photon number of $|\alpha|^2$. The uncertainty of the field quadratures are $\Delta X_1 = \Delta X_2 = \frac{1}{2}$. Like a classical wave, it has a well defined average phase ϕ.

where $\bar{n}$ is the average photon number of the state.

The coherent state is important because it is a quantum-mechanical equivalent to a classical monochromatic EM wave and is often used as a model to describe laser light. However, unlike a classical EM wave, the phase and photon number of a coherent state have uncertainty in them and are always fluctuating about their average values.

2.2.2 Vacuum State And Vacuum Fluctuations

One counter-intuitive result due to the quantization of EM fields is the presence of energy in the vacuum state. In Eq. 2.6, even at the ground level of an optical mode (n=0) the system still possesses a non-zero energy of $\frac{1}{2}\hbar\omega$.

An explanation was suggested [30] that that this energy originates from a randomly fluctuating electromagnetic field. This field, often referred to as the vacuum field, is present everywhere with or without the presence of photons. The magnitude and direction of the vacuum field fluctuates about

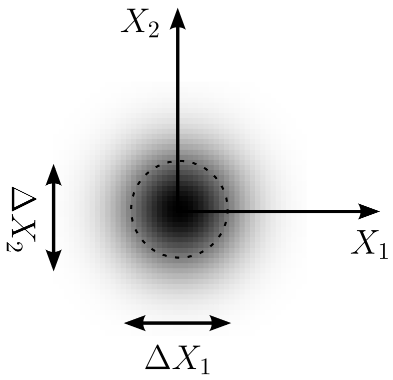

Figure 2.3: Representation of the vacuum state in the phasor diagram. The uncertainty of the field quadratures are $\Delta X_1 = \Delta X_2 = \frac{1}{2}$. However, the uncertainty of phase cannot be well defined in this case.

zero, and its variance contributes to the non-zero energy of vacuum.

A phasor diagram of the vacuum state is given in Fig. 2.3. The vacuum state can be treated as a special coherent state with $\bar{n} = 0$. However, since the vacuum field is a randomly fluctuating field, it does not have a well defined phase.

2.3 Detection Of Vacuum Fluctuations

The effect of vacuum fluctuations can be observed in phenomenon such as spontaneous parametric down-conversion and the Casimir effect [31]. In this thesis, we use a technique known as the balanced homodyne detection to measure the quantum noise induced by vacuum fluctuations of the electromagnetic field.

2.3.1 Balanced Homodyne Detection

The scheme of a balanced homodyne detector for vacuum fluctuation is illustrated in Fig. 2.4. The detector requires two optical inputs: a relatively weak input optical state under measurement, known as the signal and a

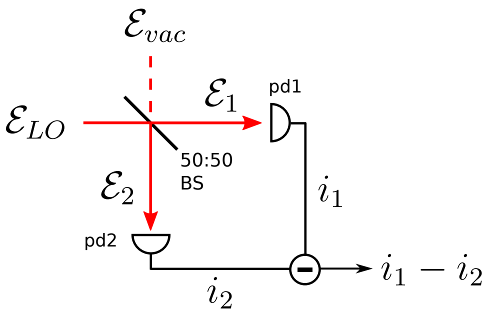

Figure 2.4: The balanced homodyne detection scheme. A coherent laser beam (the local oscillator) is split into two equal intensity beams by the beam splitter. The other input port has no input which corresponds to the vacuum state input. Two identical photodiodes pd1 and pd2 are used to generate photocurrents that are subtracted at the output.

strong coherent state, known as the local oscillator (LO). In the case of detecting vacuum fluctuations, the signal input is simply blocked to provide a vacuum state input.

The two input states are sent through the two ports of a 50:50 beam splitter, and the two output beams are received by two identical photodiodes pd1 and pd2 and converted to photocurrents. The measured photocurrents are then subtracted to provide the final output.

In the phasor diagram, the local oscillator and the vacuum field at arbitrary time t are be expressed as:

$$\alpha_{lo} = \overline{\alpha}_{lo} + \delta X_{1,lo}(t) + i\delta X_{2,lo}(t) \tag{2.17}$$

$$\alpha_{vac} = \delta X_{1,vac}(t) + i\delta X_{2,vac}(t) \tag{2.18}$$

As shown here, the coherent local oscillator is represented by a complex number α, which is decomposed as the sum of its average value $\overline{\alpha}$ plus the uncertainty $\delta X_1 + i\delta X_2$. The square of its magnitude, $|\alpha|^2$, equals to its average photon number $\overline{n}$. A similar expression is used for the vacuum field, except that $\overline{\alpha}_{vac} = 0$ since the average magnitude of vacuum field is zero.

After mixing the two input states at the PBS, we obtain two output fields β_1 and β_2:

$$\beta_1 = \frac{1}{\sqrt{2}}(\alpha_{lo} + \alpha_{vac}) \tag{2.19}$$

$$\beta_2 = \frac{1}{\sqrt{2}}(\alpha_{lo} - \alpha_{vac}) \tag{2.20}$$

The photocurrents measured at the detectors are proportional to the received photon number measured over a detection period τ:

$$i_1 = \frac{S\hbar\omega}{\tau} \cdot |\beta_1|^2 \, , \, i_2 = \frac{S\hbar\omega}{\tau} \cdot |\beta_2|^2 \tag{2.21}$$

where $\hbar\omega$ is the energy per photon and S is the sensitivity of the photodiode. Substituting the expressions in Eq. 2.17, the difference in the photocurrents is:

$$\begin{aligned} i_1 - i_2 &= \frac{S\hbar\omega}{\tau} \cdot (|\beta_1|^2 - |\beta_2|^2) && (2.22)\\ &= \frac{S\hbar\omega}{\tau} \cdot [\frac{1}{2}|(\alpha_{lo} + \alpha_{vac})|^2 - \frac{1}{2}|(\alpha_{lo} - \alpha_{vac})|^2] && (2.23)\\ &= \frac{S\hbar\omega}{\tau} \cdot (\alpha_{lo}\alpha^*_{vac} + \alpha^*_{lo}\alpha_{vac}) && (2.24)\\ &= \frac{S\hbar\omega}{\tau} \cdot \{\underbrace{(\overline{\alpha}_{lo} + \delta X_{1,lo} + i\delta X_{2,lo})}_{\alpha_{lo}}\underbrace{(\delta X_{1,vac} - i\delta X_{2,vac})}_{\alpha^*_{vac}} + C.C\} && (2.25) \end{aligned}$$

Since the local oscillator is a strong coherent state, $\overline{\alpha}_{lo} \gg \delta X$, an approximation is taken here to neglect any higher order terms such as $\delta X_{1,lo} \cdot \delta X_{1,vac}$, $\delta X_{1,lo} \cdot i\delta X_{2,vac}$, etc. As such, the current difference is given as:

$$\begin{aligned} i_1 - i_2 = \Delta i &= \frac{S\hbar\omega}{\tau} \cdot (|\beta_1|^2 - |\beta_2|^2) && (2.26)\\ &\approx \frac{S\hbar\omega}{\tau} \cdot 2\overline{\alpha}_{lo} \cdot \delta X_{1,vac} && (2.27) \end{aligned}$$

The square of the current fluctuation is:

$$\begin{aligned}\Delta i^2 &\approx (\frac{S\hbar\omega}{\tau}\cdot 2\overline{\alpha}_{lo}\cdot\delta X_{1,vac})^2 && (2.28)\\ &= \frac{4S\hbar\omega}{\tau}\cdot(\frac{S\hbar\omega|\overline{\alpha}_{lo}|^2}{\tau})\cdot(\delta X_{1,vac})^2 && (2.29)\\ &= \frac{4e}{\tau}\cdot\overline{i}\cdot(\delta X_{1,vac})^2 && (2.30)\end{aligned}$$

One can see that by rearranging the terms, the noise power is in fact proportional to the total photocurrent generated ($\overline{i}=\overline{i_1}+\overline{i_2}, \overline{i_1}=\overline{i_2}$ since the optical power is balanced). According to Eq. 2.14, $\langle\delta X_{1,vac}{}^2\rangle=\frac{1}{4}$, and if we take the average noise power, we have:

$$\begin{aligned}\langle\Delta i^2\rangle &= \frac{4e\overline{i}}{\tau}\langle\delta X_{1,vac}{}^2\rangle && (2.31)\\ &= \frac{e\overline{i}}{\tau} && (2.32)\end{aligned}$$

Which describes the shot noise power generated by a photodiode with average photocurrent $\overline{i}$. Alternatively, in the frequency domain, the noise power over a bandwidth Δf is:

$$\langle\Delta i^2\rangle = 2e\overline{i}\Delta f \tag{2.33}$$

It is apparent that the output signal is a white noise (independent of frequency) whose magnitude depends on the detector bandwidth. To obtain a large enough noise signal, fast photodiodes are desirable as they provide a larger bandwidth.

An interesting feature of homodyne detector is that it cancels the classical noise in the local oscillation. In practice, the intensity of the local oscillator cannot be perfectly constant due to possible interference from the environment (fluctuations of supply current, mechanical vibrations, etc). These classical noises resides in the laser beam and is simultaneously by the photodiodes. Unlike shot noise, the classical noises detected by the photodiodes are correlated and get cancelled out upon current subtraction.

The shot noise is usually a limiting factor in classical optical detections. Unlike other noises, the shot noise originated from the quantum uncertainty of the coherent itself and in principle cannot be eliminated unless squeezing techniques are applied to modify the state of the light [32]. However, in our case it becomes a suitable choice as a source of randomness.

2.4 Vacuum Fluctuation As Source Of Randomness

Measurement of vacuum fluctuations turns out to be a suitable source of randomness for several reasons. Firstly, it is a purely quantum mechanical phenomenon that does not have a classical correspondence. Measurements on such a quantum system should yield outcomes that are intrinsically random: they are neither predetermined [33], nor are they correlated to any other systems. This meets the requirement of unpredictability of a random number generator.

From a practical point view, vacuum fluctuation measurement is a suitable choice of randomness source because of its efficiency and simplicity. The balanced homodyne detection scheme makes use of fast photodiodes which usually have high cut-off frequencies. This generates a random signal with larger bandwidth and allows faster data acquisition speed that results eventually in a higher random number generation rate. A homodyne detector is also a relatively simple construct and can be made compact and portable.

Chapter 3

Hardware Implementation

3.1 Experimental Setup

Fig. 3.1 shows the schematic of our random number generator implementation. The local oscillator of the homodyne detector is provided by a continuous wave laser diode (LD) of 780 nm wavelength. The laser diode is powered by a constant current source with a 5V supply voltage.

The output optical power of the laser diode is split by a polarizing beam splitter (PBS). Since the laser diode output is a linearly polarized light, one can use a PBS to tune the power distribution of the two split beams by simply rotating the laser diode. The two split beams are directed to two identical photodiodes (Hamamatsu S5972) and are converted to photocurrents i_1 and i_2. The photocurrent difference $i_1 - i_2$ at the middle point between the two diodes is measured.

The DC component of the photocurrent difference is measured across a resistor R_{DC}. Since the scheme is a balanced detection, the DC component should be ideally kept at 0V such that the photodiodes receive equal input power. This can be achieved by rotating the polarization of the laser diode to

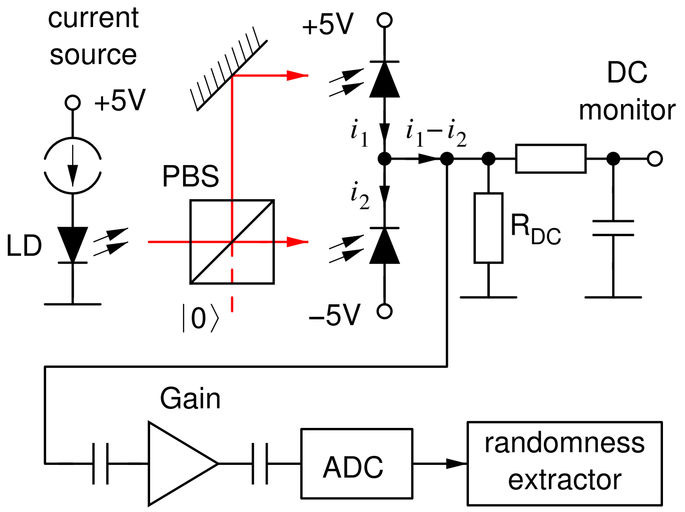

Figure 3.1: Schematic of the quantum random number generator. A polarizing beam splitter (PBS) distributes the optical power from the laser diode equally on two photodiodes. The photodiodes generate photocurrents i_1 and i_2. The photocurrent difference $i_1 - i_2$ is measured. Its fluctuating AC component is amplified, digitized and sent to a randomness extractor to generate true random bits.

adjust the optical power at the output ports of PBS. Since the optical power received by the diodes is balanced, classical power fluctuations in the local oscillator will be simultaneously detected and cancelled in the photocurrent difference [34, 35]. An alternative interpretation is that the laser beam is generating photocurrents i_1,i_2 with a noise power proportional to the average optical power, which is the shot noise of the local oscillator.

Gain Stage

The AC component of the photocurrent difference is the shot noise of local oscillator, which is used as our source of randomness. However, since the shot noise level is too weak to be directly used, amplification of signal is needed.

The AC component is sent through a gain stage, as shown is Fig. 3.2.

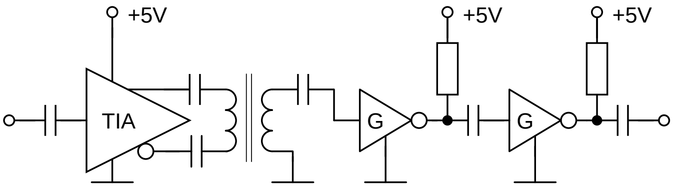

Figure 3.2: Gain stage used to amplify the shot noise signal. A transimpedance amplifier (TIA) forms the first stage to convert photocurrent difference $i_1 - i_2$ into a low impedance voltage signal. The voltage signal is further amplified by the second and third stage amplifiers, which are both transistor amplifiers (Mini Circuits MAR 6 SM+). The gain stage is designed to have an effective transimpedance $R_{eff} \approx 590k\Omega$.

A transimpedance amplifier (Analog Devices AD8015) is used as the first stage amplifier to amplify the photocurrent difference into a voltage signal. The amplifier is inductively coupled to the next stages and has a effective transimpedance of $7.07 \pm 1.41k\Omega$. The second and third stages are two transistor amplifiers (Mini Circuits MAR 6 SM+), with a gain of 20 dB each. Due to insertion losses and other factor, we concluded the effective transimpedance of the gain stage to be $R_{eff} \approx 590 \pm 118k\Omega$.

The gain stage is designed to have a wide bandwidth for amplification, ranging from 20 MHz to 120 MHz, determined by the DC block capacitors and the cut-off frequency of the transimpedance amplifier. This will avoid most RF noises and acoustic/mechanical noises from entering the circuit.

Digitization And Post-processing

The amplified signal is digitized into signed 16-bit numbers at a sampling rate of 60 MHz set by the Analog-to-Digital Converter (ADC) unit used. The sampling rate is set to be lower than the cut-off frequency of the amplifier in order to avoid temporal correlation between samples.

The digitized 16-bit numbers follow a non-uniform distribution and cannot be directly used as good random numbers. An additional step known

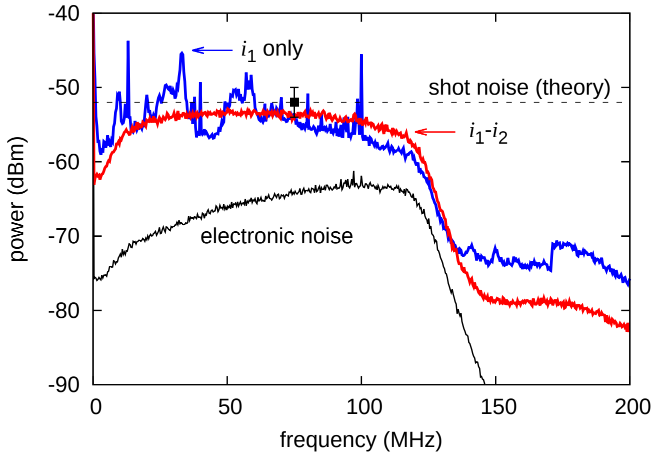

Figure 3.3: Noise levels measured after the gain stages with a resolution bandwidth $B =$ 1 kHz. Between 20 and 120 MHz, the total noise (red) is the amplified photocurrent difference $i_1 - i_2$ with a balanced optical power impinging on both photodiodes. The measured total noise comes close to the expected shot noise level calculated in Eq. 3.1. The AC current of a single photodiode i_1 is also recorded (blue) and shows classical noises at various frequencies. The electronic noise (black) is measured without any optical input.

as randomness extraction is need to process the imperfect random numbers. The details of randomness extractors will be elaborated in chapter 4.

3.2 Performance

In operation, the local oscillator laser supplies about 10 mW optical power. After the PBS, 3.1 mW is coupled to each photodiode. After the gain stages, one would expect an amplified noise power of:

$$P = \frac{4e\bar{I}B{R_{eff}}^2}{Z} \approx -52 \pm 2 \text{ dBm} \tag{3.1}$$

where $\bar{I} \approx 1.34$ mA is the average DC photocurrent generated by each photodiode and e is the electron charge. The measurement bandwidth chosen to be $B = 1$ kHz and the load resistance is $Z = 50\ \Omega$.

The actual output shot noise signal is measured with a spectrum anal-

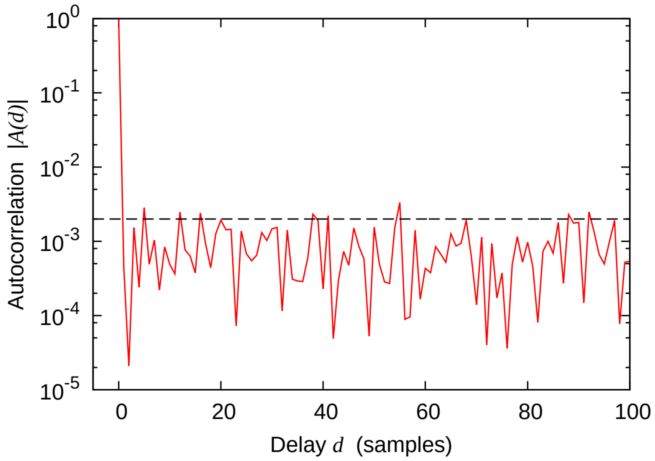

Figure 3.4: Autocorrelation of the total signal sampled at 60 MHz (solid line), compared with the 2σ confidence level (dashed line).

yser with a resolution bandwidth of 1 kHz. The spectral power density is displayed in Fig. 3.3. The measured total noise signal (red) is about 1.5 dB lower than the theoretically calculated shot noise level (dashed trace), which is still with the error bar caused by the uncertainty in the gain of the amplifier chain. The noise signal has a relatively flat power density in the range of 20 to 120 MHz.

To illustrate the effectiveness of removing classical noise using balanced homodyne detection, the power spectral density of photocurrent generated by a single diode is also displayed (blue). Strong spectral peaks at various radio frequencies appear in the signal possibly through modulating the laser supply current and this may reduce the randomness available in the signal. For completeness, the electronic noise of the amplifier is also recorded (black) which appears to be at least 10 dB below the total noise level, thus it is safe to conclude that the total output noise is dominated by quantum fluctuations.

The total noise signal is digitized into signed 16-bit words x_i at a sampling rate of 60 MHz and the autocorrelation function is computed to check for temporal correlation between samples. As shown in Fig. 3.4, the normalized autocorrelation function:

$$A(d) = \langle x_i x_{i+d} \rangle_n / \langle {x_i}^2 \rangle_n \tag{3.2}$$

is computed over $n = 10^6$ samples and the autocorrelation coefficient falls into the expected confidence interval (dashed line) which indicates no significant correlation between samples.

3.3 Entropy Estimation

The random noise signal we obtained at the output of the gain stages is a combination of amplified shot noise and the electronic of the amplifiers. Although we can achieve a signal to noise ration of 10 dB, the electronic noise may still affect the randomness of the signal. At this point, it would be helpful if we can quantify the amount of randomness we can safely extract from the total noise such that the randomness is still considered originating from a quantum mechanical source.

To quantify the amount of randomness, we use the concept of Shannon entropy. The Shannon entropy of a random variable X is defined as:

$$H(X) = \sum -p(x) \log_2 p(x) \tag{3.3}$$

where $p(x)$ is the probability distribution of the random variable X.

To estimate the amount of Shannon entropy of shot noise $H(X_s)$, we follow an approach used in [25, 36]. We assume that the measured total noise signal X_t is the sum of the shot noise X_s and the electronic noise X_e, such that:

$$X_t = X_s + X_e \tag{3.4}$$

Here, X_t, X_s and X_e are random variables with discrete distributions over the digitization interval from 2^{-15} to $2^{15} - 1$. Further more, we assume that

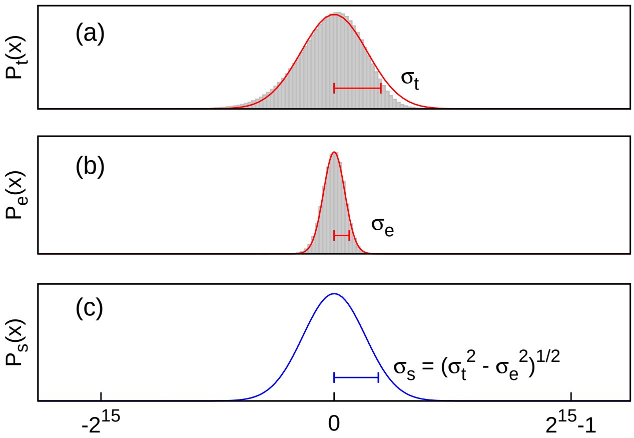

Figure 3.5: Probability distribution of the measured total output noise with variance ${\sigma_t}^2$ (a), electronic noise with variance ${\sigma_e}^2$ (b), and the estimated shot noise with calculated variance ${\sigma_s}^2$ (c). The filled areas in (a) and (b) show the actual measurements over 10^9 samples. The solid lines approximate the related Gaussian distributions.

X_s and X_e are independent to each other. Since we are uncertain of the origin of electronic noise, we consider the worst case scenario that the electronic noise is completely untrustworthy, i.e., an adversary is able to gain full knowledge of electronic noise and predict the exact outcome of variable X_e at any moment. In this case, the accessible amount of randomness in the acquired total noise signal is quantified by the conditional entropy $H(X_t|X_e)$, i.e., the amount of entropy left in the total noise, given full knowledge of the electronic noise X_e.

As we have assumed the random variables to be additive and independent, the conditional entropy may be calculated as:

$$H(X_t|X_e) = H(X_s + X_e|X_e) = H(X_s|X_e) = H(X_s) \qquad (3.5)$$

Which is the Shannon entropy of the shot noise itself.

However, since the shot noise signal cannot be directly measured, we can only approximately estimate its distribution. The variance of the total noise ${\sigma_t}^2$ is given by the sum of the variance of shot noise ${\sigma_s}^2$, and electronic noise ${\sigma_e}^2$. We recorded an ensemble of 10^9 samples and the distribution is displayed in Fig. 3.5. The total noise and electronic appear to follow Gaussian distribution with $\sigma_t = 4504.41$ and $\sigma_e = 1481.8$. Note that for the total noise, the observed distribution is slightly skewed compared to a Gaussian fitting (solid line in Fig. 3.5(a)), which is believed to be effect from amplifier distortions.

We assume here that the shot noise has a Gaussian distribution [30], such that we may compute:

$$ {\sigma_s}^2 = {\sigma_t}^2 - {\sigma_e}^2 \approx 4253.7^2 \tag{3.6}$$

At this point, we estimate that X_s is a random variable with a Gaussian distribution of variance 4253.7^2 over the digitization interval 2^{-15} to $2^{15} - 1$. The Shannon entropy is then estimated:

$$ H_s = \sum_{x=1}^{2^{16}} -p_s(x) \log_2 p_s(x) \approx 14.1 \text{ bits} \tag{3.7}$$

with $p_s(x)$ being the corresponding distribution function. This suggests that out of every 16-bit sample, one should be able to extract 14.1 bits of uniformly distributed random bits.

It should be noted that this numerical estimation only serves as an upper bounder of randomness, i.e., the maximum possible amount of entropy one can extract from the total noise with the assumptions of a Gaussian distribution of the independent random variables X_s and X_e. In other literatures [21, 37], entropy is estimated under more strict assumption that an adversary not only can monitor the electronic noise, but is also able to change its value. Fewer random bits can be extracted under this scenario.

Chapter 4

Post-processing And Evaluation

From the physical setup introduced in Chapter 3. We obtained a noise signal that can be used as our source of randomness. We concluded that our signal resembles white noise over a large bandwidth (Fig. 3.3), shows no significant temporal correlation (Fig. 3.4) and follows a Gaussian distribution (Fig. 3.5). We also estimated the amount of extractable entropy from the noise signal and concluded an upper bound of 14.1 bits entropy out of every 16-bit sample.

Our noise signal originates from quantum fluctuations and is in principle unpredictable. However, in many applications, random numbers are required to be not only unpredictable, but also uniformly distributed. This means that our raw date from the digitized noise signal cannot be directly used since they are non-uniformly distributed. To form a complete random number generation scheme, post-processing is required.

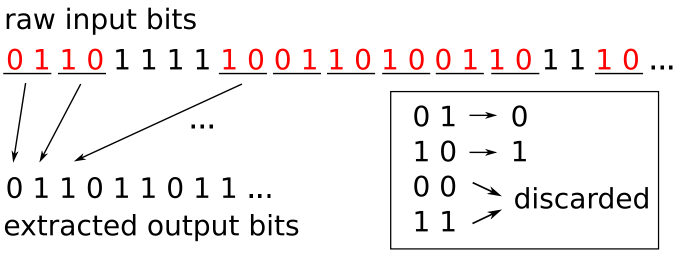

Figure 4.1: The Von Neumann randomness extractor. A stream of biased bits are divided into pairs. The algorithm discards all the 00 and 11 pairs, and makes the mapping 01 → 0, 10 → 1. The resulted stream will have equal probability of 0s and 1s.

4.1 Randomness Extraction

Randomness extraction is the essential process required to generate high quality random numbers that are uncorrelated and uniformly distributed. The central part of randomness extraction is usually an algorithm known as the randomness extractor. The randomness extractor receives a statistically weak binary stream as input, and generate uniformly distributed random bits at its output.

The Von Neumann Extractor is perhaps one of the earliest randomness extractors proposed [10]. The algorithm is used to extract uniform random bits from a stream of independent, but biased random bits ($P(0) \neq P(1) \neq \frac{1}{2}$).

Fig. 4.1 illustrates the Von Neumann extraction scheme. The algorithm receives the biased stream as input and divides the stream into pairs of bits. All the 00 and 11 pairs are discarded, and the remaining 01 and 10 pairs are mapped into 0s and 1s accordingly. Since the probability of having a 01 pair is the same as having a 10 pair, the output stream of such algorithm is guaranteed a uniform distribution between 0s and 1s.

It is worth noting that the Von Neumann extractor is only suitable for independent , biased random bit stream. If correlations exist between consecutive bits in the stream, this method is no longer applicable. Also, the algorithm is not considered efficient as more than 75% of the bits from the input are lost while there are still remaining entropy available.

Many other implementations of randomness extractors have been reported, such as Trevisan's extractor and Toeplitz-hashing extractor in [36], random-matrix multiplication used in [25, 38]. For implementations of random number generators, various families cryptographic hashing functions are often adopted such as Secure Hashing Algorithms (SHA) in [22], and Advanced Encryption Standard hashing (AES) in [37]. These algorithms are usually carefully designed and have good performance. However, most cryptographic hashing functions are complicated and required lots of computational resources. This could be a limiting factor when one is pushing for higher rate of random number generation.

In this document we report an implementation of a randomness extractor based on Linear Feedback Shift Register (LFSR). The implementation appears effective against various randomness tests and the construct is compact. Details of the implementation will be introduced in the remains of the chapter.

4.2 Randomness Extractors Based On Linear Feedback Shift Registers

4.2.1 Linear Feedback Shift Registers

Linear Feedback Shift Register (LFSR) is a arithmetic method known for quickly generating long pseudo-random streams with very little computational resources. It is widely used in communication applications for spectrum whitening and other purposes [38, 39].

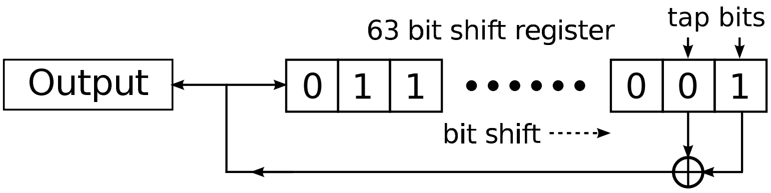

Figure 4.2: Schematic of a 63-bit LFSR. Tap bit in the shift register is XORed to generated a new bit and is fed back to the shift register. This specific choice of tap bits can maximize the period of the output stream, making it repeat itself after $2^{63} - 1$ bits.

Fig. 4.2 shows an example of a LFSR. Its main body is a shift register of 63 bits length. Bits stored in the two rightmost cells are called tap bits and are involved in the logic gate operations of LFSR. At each clock cycle, an XOR operation is performed between the tap bits and produce one new bit, which is then sent to the output; the very same result bit is also sent to the left most cell of the shift register with the remaining bits shifted to the right by one bit (right most bit discarded). At each clock cycle the LFSR produces on bit output and update the internal state of the shift register.

It is obvious that a LFSR has a deterministic output: the binary stream will repeat itself after a certain period since there is only a finite number of internal states possible. This period can be maximized by specific choices of tap bit positions. For a n-bit LFSR, the maximal period achievable is $2^n - 1$ bits, which corresponds to all possible possible internal state of a n-bit word except the all zero state[1]. In the example in Fig. 4.2, choosing the tap bits at two right most cells produces an output stream of maximal length of period $2^{63} - 1$ bits. We chose a 63 bit LFSR for the simplicity of the structure: a 63 bit LFSR only needs 2 tap bits to produce a maximal length output, which reduces the number of XOR gates required to the minimum.

The output streams of such maximal-length LFSRs have nice statistical

[1] If all the bits in the shift register are 0, the LFSR will produce an output stream of 0s.

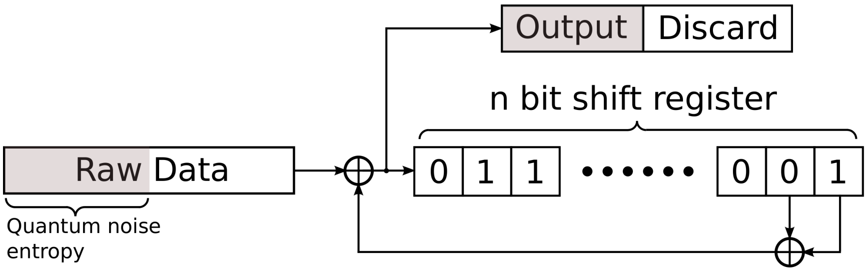

Figure 4.3: Randomness extractor based on a LFSR structure. The tap bits are XORed together with the input to generate a new bit. The new bit is sent to output and also fed back to the LFSR.

properties. They have uniform distribution of bits and their spectrum is white, and for this reason the LFSRs are sometimes used to generate pseudo-random streams for communication purposes [38, 40].

LFSRs can be implemented using simple digital circuits and reach very high speed in real operations. This motivates us to construct a randomness extractor based on LFSR structure.

4.2.2 LFSR-Based Randomness Extractor

Our implementation of randomness extractor is illustrated in Fig. 4.3. The main body of the extractor is a 63-bit linear feedback shift register of maximal period, similar to the example in Fig. 4.2. The raw data is sent to the extractor in serial. At each clock cycle, a XOR operation is performed between the two tap bits of the LFSR. The result bit is then XORed with the input bit of the raw data, generating one new bit. The new bit is then sent to output and also fed back into the LFSR and update its internal state by shifting all bits to the right. The initial internal state of the extractor can be set to any random bit string of 63 bit length, and the first 63 bits output should be discarded to ensure that the extraction process is irreversible.

Since the extractor is a deterministic process, the entropy in the output

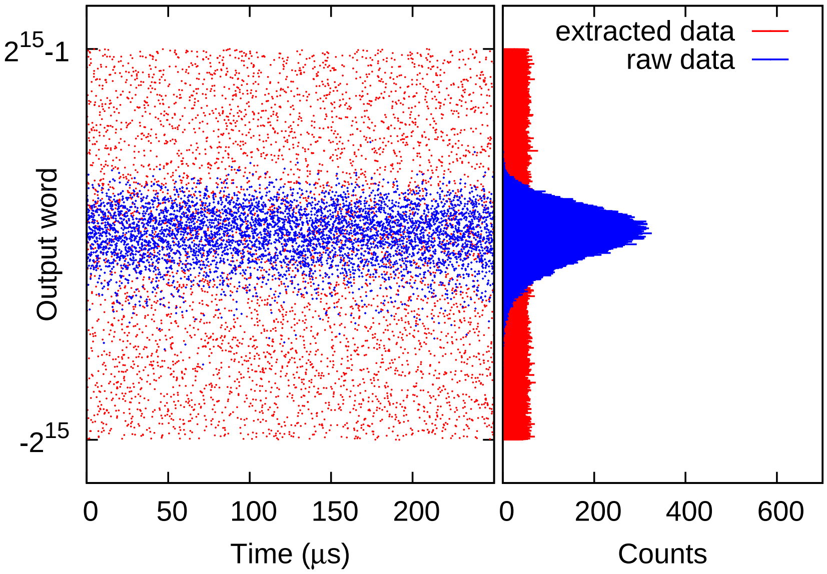

Figure 4.4: Effect of randomness extraction. Raw data (blue) has obvious non-uniform distribution. After extraction, the extracted data are grouped in 16-bit numbers and shows a uniform distribution over the 2^{16} interval.

should be less than or equal to the entropy in the raw data. It is previously calculated in Eq. 3.7 that the raw data contains 14.1 bits of Shannon entropy per 16-bit sample. To ensure that we are extracting less than the Shannon entropy bound, for every 16-bit output, we discard 8 bits[2] and keep 8 bits to the final data for the sake of programming simplicity.

Fig. 4.4 demonstrates the effect of randomness extraction. The raw data samples (red) are displayed in both time domain (left) and histogram (right). The raw data obviously follows a Gaussian distribution. The extracted random bits are grouped into signed 16-bit numbers and displayed in the same figure (red). The extracted numbers now distribute uniformly over the entire interval and appears more random.

[2] This can be done by simply discard bits at fixed positions of output, since the extracted numbers are already randomly distributed.

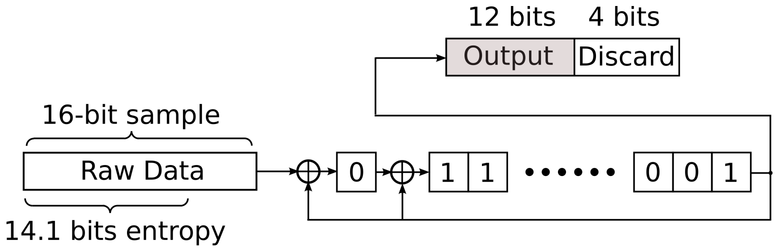

Figure 4.5: A Galois type LFSR-based extractor (63-bit). The XOR operations can be performed simultaneously, which shortens the processing time.

This implementation shares some nice features of a LFSR. It can operate at very high speed as processing each bit only takes one clock cycle. Unlike many other randomness extractors, our method will not be a limiting factor of the overall generation rate. This extractor is also very compact as a LFSR requires very little computational resources. It can in principle be implemented in cell-phones, smart cards or other mobile applications where only limited computing power is allowed.

Variations Of Randomness Extractor

A few variations can be made to our LFSR-based randomness extractor to even further improve the efficiency of randomness extraction.

Fig. 4.5 shows an extractor based on a slightly different LFSR known as the Galois type LFSR. The merit of such a design is that all the XOR operation in the Galois type LFSR can be done simultaneously, thus reduce the actual time needed to process one input bit. Extractor based on a Galois type LFSR can achieve the same uniformizing effect as the one introduced in the previous section.

A parallel version of randomness extractor is shown in Fig. 4.6. This version takes parallel input/output and provides a boost in the processing speed. However, the trade-off is that the parallel extractor requires more

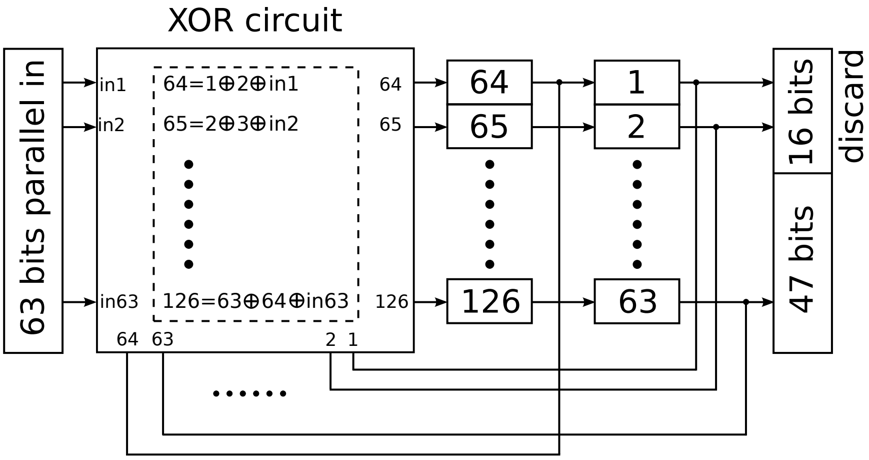

Figure 4.6: A parallel LFSR-based extractor. This version of extractor has parallel input/output and significantly improves the processing speed of randomness extraction. The trade-off is an increased number of memory cells and XOR gates.

memory cells and XOR gates in its circuitry.

4.3 Testing Random Numbers

Till this point we have completed the process of quantum random number generation. It is natural at this stage to consider about testing the quality of the generated random numbers.

Testing random numbers is in fact a non-trivial task and research on this matter is still active in related fields in mathematics and statistics. One frustrating fact regarding random number tests is that there is no way to perfectly certify randomness. Given a number sequence from unknown source, one can never tell whether it is truly random or not because there is always possibility of having a random number generator generating the same sequence.

However, it is possible to determine the likely-hood of a sequence being random. This can be achieved through various statistical tests, as introduce

Compression software	Original file size (Bytes)	Compressed file size	Compression ratio
Lzma	2,147,438,647	2,176,693,974	1.0136
WinRAR		2,153,045,093	1.0026
Gzip		2,147,831,450	1.0001
Bzip		2,156,976,807	1.0044

Table 4.1: 2 GByte random bits compressed using commercial compression software. None of the software can compress the file to a smaller size.

in Chapter 1. The tests probe the statistical properties of the number sequence and compare them to the expected values of true random number sequences. The comparison is usually done in the form of a chi-square test and the likely-hood is expressed in terms of P-values.

Luckily, there are well organized sets of randomness tests, or randomness test suites, available for the testing purposes. We adopted two famous test suites to evaluate the performance of our prototype: the statistical test suite from NIST (National Institution of Standards and Technology), and the Die-harder randomness test suite [26,27]. Reports from two test suites show that our random number generator passed both test suites. A conclusive test report from the Die-harder test suite can be found in the Appendix.

We performed another interesting test by using compression software. A 2 GigaByte binary file was populated with random bits generated from our prototype. Four different compression software are applied to this file, shown in Table. 4.1. Since the file contains nothing but random bits, the compression algorithm of the software should not be able to find a more efficient way to encode the file, hence the compressed file size should be at least larger than or equal to the original file size.

4.4 Rate Of Random Number Generation

Another point of concern is the speed of random number generation. As mentioned in Chapter 3, the rate of sampling of the ADC is set to 60 MHz and each sample is a 16-bit signed integer. In the randomness extraction process, we discard 4 bits from each 16-bit sample and only keep 12 bits as final output. This in theory give us a generation rate of 720 Mbits/s. However, due to limitation of data transfer (USB2.0), the generation rate is currently limited to 480 Mbits/s. However, this result is still significantly faster compared to many commercial products in the market [41].

The implementation has the potential to reach even higher rate (>1 Gbits/s) by changing to faster ADC unit and wider bandwidth amplifiers. The randomness extractor will not be a limiting factor since it can be implemented very efficiently.

Chapter 5

Conclusion And Outlook

The thesis described an implementation of a quantum random number generator based on measurements of vacuum fluctuation of electromagnetic field. The importance and usefulness of random numbers motivates the pursuit of good random number generators [2–4]. This is introduced in chapter 1 along with a brief summary of existing methods of random number generation. For generating truly unpredictable random numbers for cryptographic purposes, physical random number generators based on quantum measurements are favoured due to their intrinsic randomness. Several different schemes of quantum random number generation have been reported in [17–25].

The theory of vacuum fluctuation is briefly introduced in chapter 2. For the vacuum state of the electromagnetic field, the number of photons fluctuates due to the energy-time uncertainty. Since the vacuum fluctuation is not correlated to any other state, it is a good randomness source. A way of probing the vacuum fluctuation is through a homodyne measurement with a coherent laser, in which the measured fluctuation is also interpreted as the quantum noise of the coherent laser [28].

The detailed implementation of the homodyne detector is described in chapter 3, and is based on a 50:50 polarisation beam splitter and a pair of fast photodiodes. The differential signal measured by the photodiodes is recognized as the quantum noise and is amplified by a transimpedance amplifier. The output is a white noise signal of 100 MHz bandwidth and has a noise power of -53.5 dBm. The signal appears to have very low autocorrelation and its amplitude follows a nearly Gaussian distribution and it is used as the source of randomness.

In chapter 4, we try to quantify the amount of randomness in the random signal by statistically calculate its Shannon entropy. The effect of electronic noise is also taken into account. This entropy estimation concludes 14.1 bits of entropy out of every 16-bit digital sample. The digitized signal is yet to be converted to a uniformly distributed random bit stream, for which we applied a randomness extractor based on linear feedback shift registers. Several designs of the randomness extractors are proposed and one is implemented. The final output after the extractor appears to be a correlation-free, uniformly distributed random bit stream. The generated bit stream passed two different statistical test suites and is shown to be incompressible against compression software. An overall generation rate of 480 Mbit/s is achieved.

This implementation has the potential to reach even higher generation rate by using faster data transfer protocols, amplifiers and ADC with wider bandwidth. The hardware can be miniaturized into a compact design to improve portability. These points may be addressed in future prototypes of the device. There are still room for improvement regarding the theoretical description of entropy estimation and randomness extraction and is subject to future effort.

The reported prototype can potentially be applied to various information security systems where secure key/password generations are required.

Given its compact design, the random number generator can be embedded to mobile communication systems and even satellites. The proposed randomness extraction algorithm can be efficiently implemented to systems where only limited computational resources are available such as cellphones or even smart cards. We are currently in process of commercializing the prototype into a robust and easy-to-use product.

Appendix A

Test Report Of Dieharder Randomness Test Suite

We display a randomness test report from the Dieharder Randomness Test Suite. It shows the various tests used against our random number output, and the P-value of each test. The pass/fail threshold of P-value is set to 0.05, which is a commonly adopted value.

The test result is Pass for majority of the tests, except for one of the marsaglia-tsang gcd test. However, we did not consider this to be an anomaly as the fail is not reproducible. In addition, a good random number generator should fail a test occasionally (1 out of 100 test, estimated by the author of the test suite [27]) as there is always non-zero possibility that a random number generator generates a sequence that does not appear random.

```
#=============================================================================#
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
#=============================================================================#
   rng_name    |rands/second|   Seed   |
stdin_input_raw|  5.64e+05  |3685956135|
#=============================================================================#
        test_name   |ntup| tsamples |psamples|  p-value |Assessment
#=============================================================================#
   diehard_birthdays|   0|       100|     100|0.42189983|  PASSED
      diehard_operm5|   0|   1000000|     100|0.83296102|  PASSED
  diehard_rank_32x32|   0|     40000|     100|0.48340646|  PASSED
    diehard_rank_6x8|   0|    100000|     100|0.30685907|  PASSED
   diehard_bitstream|   0|   2097152|     100|0.08570837|  PASSED
        diehard_opso|   0|   2097152|     100|0.86761981|  PASSED
        diehard_oqso|   0|   2097152|     100|0.87503398|  PASSED
         diehard_dna|   0|   2097152|     100|0.63984183|  PASSED
diehard_count_1s_str|   0|    256000|     100|0.80631795|  PASSED
diehard_count_1s_byt|   0|    256000|     100|0.03422649|  PASSED
 diehard_parking_lot|   0|     12000|     100|0.87504329|  PASSED
    diehard_2dsphere|   2|      8000|     100|0.97398772|  PASSED
    diehard_3dsphere|   3|      4000|     100|0.26366379|  PASSED
     diehard_squeeze|   0|    100000|     100|0.63934427|  PASSED
        diehard_sums|   0|       100|     100|0.01266915|  PASSED
        diehard_runs|   0|    100000|     100|0.97014427|  PASSED
        diehard_runs|   0|    100000|     100|0.29336373|  PASSED
       diehard_craps|   0|    200000|     100|0.73510155|  PASSED
       diehard_craps|   0|    200000|     100|0.40634955|  PASSED
 marsaglia_tsang_gcd|   0|  10000000|     100|0.00319964|   WEAK
 marsaglia_tsang_gcd|   0|  10000000|     100|0.35131881|  PASSED
         sts_monobit|   1|    100000|     100|0.14597918|  PASSED
            sts_runs|   2|    100000|     100|0.70212549|  PASSED
          sts_serial|   1|    100000|     100|0.07460339|  PASSED
          sts_serial|   2|    100000|     100|0.41882230|  PASSED
          sts_serial|   3|    100000|     100|0.53204182|  PASSED
          sts_serial|   3|    100000|     100|0.94640049|  PASSED
          sts_serial|   4|    100000|     100|0.93194562|  PASSED
          sts_serial|   4|    100000|     100|0.56535565|  PASSED
          sts_serial|   5|    100000|     100|0.26895674|  PASSED
          sts_serial|   5|    100000|     100|0.73262918|  PASSED
          sts_serial|   6|    100000|     100|0.76240849|  PASSED
          sts_serial|   6|    100000|     100|0.04844870|  PASSED
          sts_serial|   7|    100000|     100|0.41811741|  PASSED
          sts_serial|   7|    100000|     100|0.61719976|  PASSED
          sts_serial|   8|    100000|     100|0.32199931|  PASSED
          sts_serial|   8|    100000|     100|0.06904407|  PASSED
          sts_serial|   9|    100000|     100|0.17257992|  PASSED
          sts_serial|   9|    100000|     100|0.46246179|  PASSED
          sts_serial|  10|    100000|     100|0.56731759|  PASSED
          sts_serial|  10|    100000|     100|0.11145197|  PASSED
          sts_serial|  11|    100000|     100|0.97415867|  PASSED
          sts_serial|  11|    100000|     100|0.98259562|  PASSED
          sts_serial|  12|    100000|     100|0.68642366|  PASSED
          sts_serial|  12|    100000|     100|0.36017898|  PASSED
          sts_serial|  13|    100000|     100|0.64254801|  PASSED
          sts_serial|  13|    100000|     100|0.47286122|  PASSED
          sts_serial|  14|    100000|     100|0.59154759|  PASSED
          sts_serial|  14|    100000|     100|0.38707573|  PASSED
          sts_serial|  15|    100000|     100|0.95824892|  PASSED
          sts_serial|  15|    100000|     100|0.92350938|  PASSED
          sts_serial|  16|    100000|     100|0.09095919|  PASSED
          sts_serial|  16|    100000|     100|0.56182805|  PASSED
         rgb_bitdist|   1|    100000|     100|0.39639663|  PASSED
         rgb_bitdist|   2|    100000|     100|0.72602136|  PASSED
         rgb_bitdist|   3|    100000|     100|0.37324029|  PASSED
         rgb_bitdist|   4|    100000|     100|0.19810600|  PASSED
         rgb_bitdist|   5|    100000|     100|0.29363437|  PASSED
         rgb_bitdist|   6|    100000|     100|0.47719328|  PASSED
         rgb_bitdist|   7|    100000|     100|0.67125884|  PASSED
         rgb_bitdist|   8|    100000|     100|0.14444784|  PASSED
         rgb_bitdist|   9|    100000|     100|0.71522946|  PASSED
         rgb_bitdist|  10|    100000|     100|0.93546675|  PASSED
         rgb_bitdist|  11|    100000|     100|0.32744169|  PASSED
         rgb_bitdist|  12|    100000|     100|0.60879141|  PASSED
rgb_minimum_distance|   2|     10000|    1000|0.35819754|  PASSED
rgb_minimum_distance|   3|     10000|    1000|0.31634046|  PASSED
rgb_minimum_distance|   4|     10000|    1000|0.31847551|  PASSED
rgb_minimum_distance|   5|     10000|    1000|0.01436149|  PASSED
    rgb_permutations|   2|    100000|     100|0.99476937|  PASSED
    rgb_permutations|   3|    100000|     100|0.32529068|  PASSED
    rgb_permutations|   4|    100000|     100|0.92522581|  PASSED
    rgb_permutations|   5|    100000|     100|0.66868108|  PASSED
```

```
rgb_lagged_sum|   0|   1000000|     100|0.65341124|  PASSED
rgb_lagged_sum|   1|   1000000|     100|0.11967455|  PASSED
rgb_lagged_sum|   2|   1000000|     100|0.60587959|  PASSED
rgb_lagged_sum|   3|   1000000|     100|0.82562475|  PASSED
rgb_lagged_sum|   4|   1000000|     100|0.75646089|  PASSED
rgb_lagged_sum|   5|   1000000|     100|0.53671066|  PASSED
rgb_lagged_sum|   6|   1000000|     100|0.14519368|  PASSED
rgb_lagged_sum|   7|   1000000|     100|0.66749106|  PASSED
rgb_lagged_sum|   8|   1000000|     100|0.88940851|  PASSED
rgb_lagged_sum|   9|   1000000|     100|0.54639878|  PASSED
rgb_lagged_sum|  10|   1000000|     100|0.47756900|  PASSED
rgb_lagged_sum|  11|   1000000|     100|0.58962360|  PASSED
rgb_lagged_sum|  12|   1000000|     100|0.20041641|  PASSED
rgb_lagged_sum|  13|   1000000|     100|0.15039751|  PASSED
rgb_lagged_sum|  14|   1000000|     100|0.80833370|  PASSED
rgb_lagged_sum|  15|   1000000|     100|0.65139925|  PASSED
rgb_lagged_sum|  16|   1000000|     100|0.72767544|  PASSED
rgb_lagged_sum|  17|   1000000|     100|0.16736509|  PASSED
rgb_lagged_sum|  18|   1000000|     100|0.40254815|  PASSED
rgb_lagged_sum|  19|   1000000|     100|0.15101255|  PASSED
rgb_lagged_sum|  20|   1000000|     100|0.79959023|  PASSED
rgb_lagged_sum|  21|   1000000|     100|0.38269395|  PASSED
rgb_lagged_sum|  22|   1000000|     100|0.98203674|  PASSED
rgb_lagged_sum|  23|   1000000|     100|0.38378827|  PASSED
rgb_lagged_sum|  24|   1000000|     100|0.86744838|  PASSED
rgb_lagged_sum|  25|   1000000|     100|0.81447636|  PASSED
rgb_lagged_sum|  26|   1000000|     100|0.95293320|  PASSED
```

```
      rgb_lagged_sum|  27|   1000000|     100|0.89398631|  PASSED
      rgb_lagged_sum|  28|   1000000|     100|0.14228229|  PASSED
      rgb_lagged_sum|  29|   1000000|     100|0.17492848|  PASSED
      rgb_lagged_sum|  30|   1000000|     100|0.79591771|  PASSED
      rgb_lagged_sum|  31|   1000000|     100|0.30424708|  PASSED
      rgb_lagged_sum|  32|   1000000|     100|0.75532517|  PASSED
     rgb_kstest_test|   0|     10000|    1000|0.88354617|  PASSED
     dab_bytedistrib|   0|  51200000|       1|0.40187309|  PASSED
             dab_dct| 256|     50000|       1|0.04782332|  PASSED
Preparing to run test 207.  ntuple = 0
        dab_filltree|  32|  15000000|       1|0.80805904|  PASSED
        dab_filltree|  32|  15000000|       1|0.15756894|  PASSED
Preparing to run test 208.  ntuple = 0
       dab_filltree2|   0|   5000000|       1|0.35748968|  PASSED
       dab_filltree2|   1|   5000000|       1|0.90131637|  PASSED
Preparing to run test 209.  ntuple = 0
        dab_monobit2|  12|  65000000|       1|0.56350949|  PASSED
```

Appendix B

Patent Information

The Randomness extraction algorithm described in Chapter 4 has been registered as a Non-Provisional Application at the Intellectual Property Office of Singapore, as in the following:

SG Non-Provisional Application No. 10201509277U

Title: Efficient Randomness Extractor for Random Numbers Based on Physical Measurements

ILO Ref: 15014N-SG/PRV

Bibliography

[1] D.J. Bennett. *Randomness.* Harvard University Press, 2009.

[2] B. Schneier. *Applied cryptography: protocols, algorithms, and source code in C.* Wiley, 1994.

[3] W. Trappe and L.C. Washington. *Introduction to Cryptography: With Coding Theory.* Pearson Education. Pearson Prentice Hall, 2006.

[4] D.E. Knuth. *Art of Computer Programming, Volume 2: Seminumerical Algorithms.* Pearson Education, 2014.

[5] Per Martin-Lf. The definition of random sequences. *Information and Control*, 9(6):602619, Dec 1966.

[6] G.J. Chaitin. *Information, Randomness & Incompleteness: Papers on Algorithmic Information Theory.* Series in computer science. World Scientific, 1990.

[7] N. Metropolis. The beginning of the monte carlo method. *Los Alamos Science*, 15:125–130, 1987.

[8] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge International Series on Parallel Computation. Cambridge University Press, 1995.

[9] Francis Galton. Dice for statistical experiments. *Nature*, 42(1070):1314, May 1890.

[10] John von Neumann. Various Techniques Used in Connection with Random Digits. *J. Res. Nat. Bur. Stand.*, 12:36–38, 1951.

[11] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):330, Jan 1998.

[12] Lenore Blum, Manuel Blum, and Michael Shub. Comparison of two pseudo-random number generators. *Advances in Cryptology*, page 6178, 1983.

[13] Linus Torvalds. *Linux Kernel drivers/char/random.c comment documentation*, 04 2005.

[14] Max Born. Is classical mechanics in fact deterministic? *Physics in My Generation*, page 7883, 1968.

[15] Alain Aspect, Philippe Grangier, and Grard Roger. Experimental tests of realistic local theories via bells theorem. *Physical Review Letters*, 47(7):460463, Aug 1981.

[16] Alain Aspect, Philippe Grangier, and Grard Roger. Experimental realization of Einstein-podolsky-rosen-bohm gedankenexperiment: A new violation of bells inequalities. *Physical Review Letters*, 49(2):9194, Jul 1982.

[17] Michael Gude. Concept for a high performance random number generator based on physical random phenomena. *Frequenz*, 39:187, 1985.

[18] A. Figotin, I. Vitebskiy, V. Popovich, G. Stetsenko, S. Molchanov, A. Gordon, J. Quinn, and N. Stavrakas. Random number generator based on the spontaneous alpha-decay, June 1 2004. US Patent 6,745,217.

[19] Thomas Jennewein, Ulrich Achleitner, Gregor Weihs, Harald Weinfurter, and Anton Zeilinger. A fast and compact quantum random number generator. *Review of Scientific Instruments*, 71(4):1675, 2000.

[20] A. Stefanov, N. Gisin, O. Guinnard, L. Guinnard, and H. Zbinden. Optical quantum random number generator. *Journal of Modern Optics*, 47(4):595–598, 2000.

[21] T. Symul, S. M. Assad, and P. K. Lam. Real time demonstration of high bitrate quantum random number generation with coherent laser light. *Applied Physics Letters*, 98(23):–, 2011.

[22] 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. *Nature Photon*, 4(10):711715, Aug 2010.

[23] Ido Kanter, Yaara Aviad, Igor Reidler, Elad Cohen, and Michael Rosenbluh. An optical ultrafast random bit generator. *Nature Photon*, 4(1):5861, Dec 2009.

[24] C. Abelln, W. Amaya, M. Jofre, M. Curty, A. Acn, J. Capmany, V. Pruneri, and M. W. Mitchell. Ultra-fast quantum randomness generation by accelerated phase diffusion in a pulsed laser diode. *Optics Express*, 22(2):1645, 2014.

[25] Bruno Sanguinetti, Anthony Martin, Hugo Zbinden, and Nicolas Gisin. Quantum random number generation on a mobile phone. *Phys. Rev. X*, 4:031056, May 2014.

[26] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.* National Institute of Standards and Technology, April 2010.

[27] David Bauer Robert G. Brown (rgb), Dirk Eddelbuettel. Dieharder: A random number test suite, 2004.

[28] M. Fox. *Quantum Optics : An Introduction: An Introduction.* Oxford Master Series in Physics. OUP Oxford, 2006.

[29] U. Leonhardt. *Measuring the Quantum State of Light.* Cambridge Studies in Modern Optics. Cambridge University Press, 1997.

[30] Roy J. Glauber. Coherent and incoherent states of the radiation field. *Phys. Rev.*, 131:2766–2788, Sep 1963.

[31] R. L. Jaffe. Casimir effect and the quantum vacuum. *Phys. Rev. D*, 72(2), Jul 2005.

[32] R. E. Slusher, L. W. Hollberg, B. Yurke, J. C. Mertz, and J. F. Valley. Observation of Squeezed States Generated by Four-Wave Mixing in an Optical Cavity . *Physical Review Letters*, 55:2409, 1985.

[33] J.S. Bell. *Speakable and Unspeakable in Quantum Mechanics: Collected Papers on Quantum Philosophy.* Collected papers on quantum philosophy. Cambridge University Press, 2004.

[34] Bonny L Schumaker. Noise in homodyne detection. *Optics Lettes*, 9:189, 1984.

[35] Horace P Yuen and Vincent W.S. Chan. Nosie in homodyne and heterodyne detection. *Optics Letters*, 8:177–179, 1983.

[36] 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. *Phys. Rev. A*, 87:062327, July 2012.

[37] J. Y. Haw, S. M. Assad, A. M. Lance, N. H. Y. Ng, V. Sharma, P. K. Lam, and T. Symul. Maximization of extractable randomness in a quantum random-number generator. *Phys. Rev. Applied*, 3:054004, November 2014.

[38] Hugo Krawczyk. Lfsr-based hashing and authentication. *Advances in Cryptology CRYPTO 94*, page 129139, 1994.

[39] Elad Barkan, Eli Biham, and Nathan Keller. Instant ciphertext-only cryptanalysis of GSM encrypted communication. *J Cryptol*, 21(3):392429, Sep 2007.

[40] Manfred. Schroeder. *Number Theory in Science and Communication: With Applications in Cryptography, Physics, Digital Information, Computing, and Self-Similarity; Fifth Edition.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[41] idq. Id quantique white paper. Technical report, ID Quantique, 2010.