PHOTON ARRIVAL TIME QUANTUM RANDOM NUMBER GENERATION

BY

MICHAEL ALAN WAYNE

B.S., Washington State University, 2003

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2009

Urbana, Illinois

Adviser:

Professor Paul G. Kwiat

ABSTRACT

A quantum random number generator (QRNG) is one which relies on a physical process, extracting randomness from the inherent uncertainty in quantum mechanics. This is to be contrasted with current pseudo-random number generators (PRNG), which although useful, are in fact deterministic: they always yield the same output sequence given the same input seed. This is unacceptable for some applications, such as quantum cryptography, which promise unconditional security. Unfortunately, the rate of QRNGs is still too slow for practical commercial quantum key distribution systems (which presently run at speeds over 1 GHz).

Previous QRNGs have been implemented, with the most common relying on the behavior of a photon at a beam-splitter, producing a random bit dependent on which of the two paths in which the photon is detected. However, these are totally limited by detector saturation rates, typically in the low MHz range. We previously proposed that by instead using the time interval between detections, much more random information could be extracted per detection event. Specifically, instead of only one bit per detection, in principle one could extract as many bits as the measurement time resolution would allow.

Over the past two years, we have been exploring this approach and have constructed a functional QRNG operating at rates up to 130 Mbit/s. The random output has been tested and has passed all common cryptographic random number tests.

For my family

ACKNOWLEDGMENTS

This work was initially funded in part by the DTO-funded U.S. Army Research Office Grant No. DAAD19-03-1-0282.

In my life I have come to the conclusion that there are three professions that I consider to be the most honorable: a teacher, who instructs and inspires young minds; a doctor, who cares for and nurtures these minds; and a soldier, who fights for the freedom that allows such minds to flourish. Luckily for us all, a mother and father are all of these. The most thanks, unequivocally and without reserve, go to mine.

Additionally, I would like to thank several others. My brother David, who in his own quiet way, has always been there for me. My extended family, who have always believed in me. My advisor Paul Kwiat, who has supported and helped me more than I deserve. My graduate advisor and friend, Nannapaneni Narayana Rao, who saw something in the boy sitting in his chair and gave him the biggest chance of his life. My lab-mates and friends, who put up with my songs and antics and push me to succeed. Finally, my savior, God, for being kind enough to give me these people, and for seeing, for whatever reason, the potential inside us all.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The need for randomness arises frequently in a broad spectrum of applications, ranging from numerical simulations and statistical analysis to encryption. Methods for achieving this randomness have advanced as the applications increase, evolving from simple mathematical techniques that generate *pseudo*-random numbers to physical sources of *true* random numbers. Dependent on the constraints of the application involved, it is often sufficient to use numbers that are not actually random, but random *enough*. These can be realized, e.g., by a pseudo-random number generator which performs mathematical operations on data based on a seed, typically a very large number. Unfortunately, such schemes are deterministic: the same initial seed will always create the same sequence of pseudo-random numbers. As computing power increases, this is no longer sufficient for many cryptographic applications, as it is possible for these algorithms to be compromised. In particular, one of the promises of the recent area of quantum cryptography is unconditional security based on laws of physics [1]. For such applications it is critical to have true sources of randomness.

A quantum random number generator (QRNG) exploits the inherent randomness present in quantum processes to create random numbers. Common implementations are based on the interaction of photons with a beam-splitter [2, 3, 4], where a random bit is determined by which path a photon takes. More recently, it was realized that one can use the *time* between successive photons in a single path to generate randomness [5, 6, 7]. We have used this method to achieve random number generation rates in excess of 100 MHz.

In the following thesis we start by exploring different types of

random number generators. Our current QRNG implementation is then described and evaluated. Further quantization of the amount of available entropy is introduced, and techniques to maximize that entropy are discussed. Possible detrimental issues, such as photon-number squeezed light, are introduced, and our methods for eliminating or compensating for them are presented. Finally, several future improvements and modifications are brought forth and evaluated.

CHAPTER 2

RANDOM NUMBER GENERATORS

A random number generator (RNG) is any device that is designed to generate data that appear to have no pattern. Having existed for centuries, RNGs have steadily evolved from simple mechanisms such as the flipping of a coin or the shuffling of cards into much more complex computational and physical methods. The driving force behind this evolution has primarily been the parallel development of methods to compromise the security afforded by an RNG. Given that RNGs are typically used for security purposes, when they are broken, the consequences can be very costly. The purpose of this chapter is to introduce the reader to several different types of random number generators, outline the characteristics of each, and provide enough background so that the research presented can be easily and thoroughly understood.

2.1 Pseudo-Random Number Generators

With the advent of modern computing came a revolution in the way random numbers were generated. Instead of using an unpredictable physical process, RNGs began to harness the newly available computational power and shifted into highly complex mathematical algorithms. These algorithms can produce very long strings of data that appear to be random but are in fact completely deterministic and chosen by some initial state, or "seed." Because their output is not truly random, these are referred to as pseudo-random number generators (PRNGs). Two of the most commonly used PRNGs today are variations of linear feedback shift registers (LFSRs) or linear congruential generators (LCGs). As illustrated in Figure 2.1, an LFSR is simply a register

whose input is a linear function of its input state. Because the exclusive-or and its inverse are the only linear logical operations on single bits, an LFSR is a shift register whose output is some combination of the xor of its register value. An LCG, on the other hand, is a PRNG adhering to the recurrence relation shown in Equation (2.1):

$$X_{n+1} = (aX_n + c) \pmod{m} \tag{2.1}$$

In this model X_n is the sequence of random values, m is the modulus, a is the multiplier, c is the increment, and X_0 is the seed [8]. Additional constraints are placed on the choices of these parameters, such as the requirement that c and m be relatively prime. This is one of the reasons for the large effort being put into finding very large prime numbers.

(a) (b)

Figure 2.1: Examples of LFSR implementations with a seed state size of $n = 2$ bits. Bits in (a) have been generated using a seed state of "00," resulting in completely periodic results while bits in (b) have been generated using a seed state of "01," resulting in the maximum period of $2^n - 1$.

Although these models are straightforward and easy to implement, they come with disadvantages. Most importantly, every type of PRNG is inherently periodic. Given a seed state of n bits, then the maximum period of any given PRNG cannot exceed 2^n. As illustrated in Figure 2.1(a), if a bad seed is chosen, the period of the output can be very short. LCGs suffer from this problem more than other PRNGs, as their lower bits typically have exceptionally short periods. Because of this periodicity, PRNGs cannot pass many of the statistical suites in use today. Any exception is referred to as a cryptographically secure pseudo-random number generator (CSPRNG) and must undergo rigorous testing before being accepted by the cryptographic community.

As mentioned above, the output of a PRNG only *appears* to be random. Given the same initial seed, every successive run will

produce exactly the same result. Although this fact is seemingly catastrophic from a security standpoint, this is not the case. Although the seed mechanism makes the PRNG completely deterministic, as long as the seed is kept "secret," then the computational power needed to break most PRNGs is sufficient. Despite the periodic nature, PRNGs with seeds in excess of 128 bits are commonly used (as searching through 2^{128} possibilities in a reasonable amount of time is computationally infeasible), and they are currently the most widely used and commercially viable option.

2.2 Physical Random Number Generators

Physical random number generators (PhRNGs) rely on complex physical processes to generate randomness. It is commonly believed that PhRNGs are *true* random number generators, as the processes that drive them are thought to be unpredictable. Unlike their PRNG counterparts, which are designed to produce virtually no bias (equal amount of 0s and 1s), PhRNGs occasionally do not possess this quality. Instead they generate a certain amount of *entropy*, which is then compensated for at a later time. In information theory, this is referred to as Shannon entropy [9], and the amount associated with a given distribution of N events is given by $S = -\sum_{i=0}^{N} P_i \log_2 P_i$, where P_i is an individual probability, corresponding to an outcome i, in the probability mass function of P. This entropy is measured in bits, so an entropy of 1 random bit per output bit is ideal. To reduce any inherent bias, compensation techniques such as hash functions are used; however, these are in effect pseudo-random number generators themselves.

Physical random number generators come in many forms, ranging from the simple (such as roulette wheels and lottery balls) to the more complex, such as atmospheric noise. A quantum random number generator, the focus of the research presented here and the subject of the next section, is a type of PhRNG as well. However, instead of relying on a *complex* physical process, a QRNG relies on the uncertainty present in relatively *simple* quantum processes to generate randomness.

2.3 Quantum Random Number Generators

As stated above, a QRNG exploits the inherent randomness present in quantum processes to create random numbers. Several QRNGs based on the quantum properties of light have been proposed and implemented. Most previous systems [2, 3, 10, 11], such as the one depicted in Figure 2.2 rely on the behavior of an incoming photon at a beam-splitter to generate data.

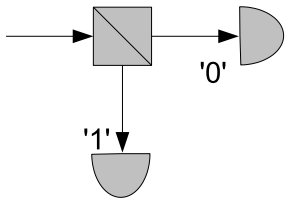


Figure 2.2: Example of a common QRNG implementation. Photons are directed along one of two paths by a beam-splitter and are registered by the corresponding detector.

Dependent on which detector registers the incoming photon, a 0 or 1 is generated. This approach has the significant drawback that each photon can create at most only one bit of data, and in practice much less, since it is only *detected* events that contribute. Thus, the scheme is limited by the detection speed: one can generate random bits only at rates substantially below the detector saturation limit. Implementations with *multiple* detectors, while somewhat mitigating this problem, can suffer from bias created by differing detection efficiencies. Recently it was shown [7] that such bias could be eliminated by using a single detector, and comparing the time intervals between three successive detection events. However, this method is limited to a maximum of one-half bit of randomness per detection, so it is even more constrained by detector saturation.

Our implementation [6] also uses only a single detector to generate the data, but it uses the photon arrival time itself as the quantum random variable, as shown in Figure 2.3.

As originally proposed [5, 6], the time between successive photons is divided into time-bins, which are created by a high resolution counter operating in parallel with the detector (in principle this could also be combined with a beam-splitter and two

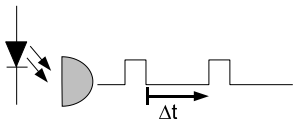


Figure 2.3: Example of our QRNG implementation. Time intervals between successive registered photons are translated into time-bin values. Because the average number of time-bins between detections can be quite large, one can distill multiple random bits per detection event.

detectors to obtain an extra random bit per detection). A given detection time interval can therefore provide multiple random bits per detection event. If we had a *constant* waiting time distribution, for which every time-bin had the same probability of occurring, then given n bins, we would generate a perfect $\log_2(n)$ random bits per detection event. However, the stimulated emission of photons from a semiconductor device is believed to be a process in which events occur continuously and *independent* of each other, i.e., a Poissionan process. The number of events that have occurred up to time t is given as $N(t)$, and the number of events in the time interval $[t, t+\tau]$ is characterized by
$P[N(t+\tau) - N(t) = k] = e^{-\lambda t}(\lambda t)^k/k!$, where λ is the average number of events per unit time. Consequently, the time between arrivals is the same as the time until the *first* arrival (since they are all independent) and is therefore given by
$P[N(t) - N(0) = 0] = e^{-\lambda t}(\lambda t)^0/0! = e^{-\lambda t}$.

As shown in Figure 2.4, the waiting-time distribution is a decaying exponential, with average value λ. As such, the entropy associated with each detection will be less than that if the distribution were simply uniform. In order to compensate for this lack of randomness, a data hashing technique must be used to "whiten" the random number string, thereby preparing a shorter but more random string, with randomness approaching one random bit per bit.

A hash function is simply a mathematical technique that converts a large amount of input data into a smaller fixed-length string. A *cryptographic* hash function, such as SHA [12], has more stringent constraints suited for tailoring its output to meet the demands required for secure communication. Among these, and of

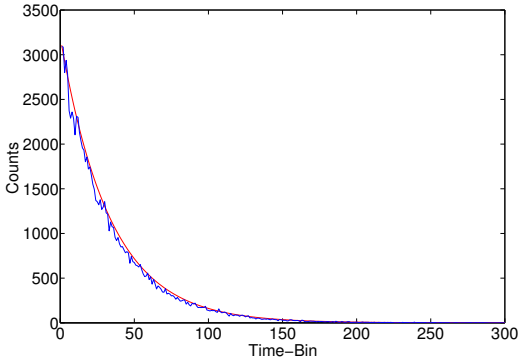


Figure 2.4: Actual (jagged trace) vs. theoretical (smooth curve) waiting-time distribution of raw data given an average detection rate of 6 MHz and a time-bin resolution of 5 ns. Here we have chosen the first time-bin to be *after* the 45-ns dead-time of the detector. Deviation from the expected decaying exponential is due to Poissonian noise and timing effects within the system.

additional importance when applied to an RNG, is the requirement that all possible outputs occur with the same probability. An example of whitened data from our QRNG is shown in Figure 2.5. This data has passed rigorous statistical tests, including the NIST test suite [13] and the DIEHARD statistical tests [14].

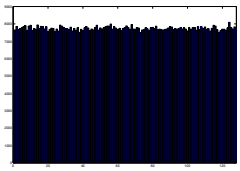


Figure 2.5: Final whitened data output from our QRNG.

Unfortunately, as is the case with all hash functions, sometimes collisions occur: cases where different inputs map to the same output. Because of this possibility we need to supply some extra entropy into the input buffer to make sure each hashed output has approximately the same probability. Further quantization of the amount of excess entropy needed is explored in Chapter 3.

2.4 Implementation Details

The implementation detailed here has four major components: photon production, optical attenuation, photon detection, and

data processing and storage. Although our initial experiments used an LED light source, we now employ an attenuated laser diode, for two primary reasons. First, the light emitted from a typical LED is in principle in a thermal state (i.e., with photon bunching over a small interval), whereas laser emission is ideally in a coherent state; this advantage is somewhat spurious, however, as the amount of bunching in an LED is minuscule, and in any event, our dead-time deletion (the time after a detection during which the detection device is disabled) automatically rejects such closely spaced photons. Second, it has been shown that a simple current-limited LED circuit can, under certain conditions, actually produce photon-number squeezed light [15]. The resulting sub-Poissonian waiting-time distribution would display lower random fluctuations than that of a laser diode. As discussed below, the random deletion afforded by our strong attenuation should completely eliminate any such squeezing correlations. Nevertheless, to avoid all such concerns, we deemed it preferable to use the system with the simplest characteristics (the laser diode).

We have used a variety of techniques to attenuate the light to the desired photon flux. If we imagine that the photons are emitted in a perfect coherent state, governed by Poisson statistics, then in principle any random deletion process will not alter the statistics. In fact, even if the photons are produced, e.g., in a squeezed state [15], the correlations will be washed out by the large amount of attenuation required [16] — we typically operate with over 120 dB of optical attenuation, so that a given photon from the light source has less than a 10^{-6} chance of making it to the detector. Given this level of attenuation, numerical simulations show no noticeable difference in the amount of entropy per detection between a coherent state and a heavily attenuated initially perfect number-squeezed state.

Since we rely on the independence of the photon arrival times, it is important that the physical process used to control the flux not *introduce* unwanted (or unknown) correlations. For example, periodically gating the laser diode so that it is operating only for a short time would not be appropriate, even though the average flux might be as desired; in that case the entropy would be greatly

reduced. We have implemented the necessary attenuation using three methods: spatial-mode selection (only collecting a small fraction of the emitted light), standard reflection neutral density filters, and a series of crossed polarizers. As predicted, in all cases we observed no significant difference in the final photon statistics, i.e., the waiting-time distribution was unaffected. Note that in the case of polarization filtering, we are essentially relying on the same intrinsic quantum mechanical randomness that is assumed for many quantum cryptography implementations [1]; the security of the latter depends on the fact that a photon's transmission through a polarization analyzer is a truly random quantum event. Similarly, the reflective neutral density filters are an extreme limit of a simple beam-splitter.

The transmitted photons are detected by a single-photon counter, in our case an avalanche photodiode (APD; id Quantique 100-MMF50-ULN). Although this device's 45-ns dead-time implies a saturation rate of over 22 MHz, in practice the device can only sustain a continuous count rate of 11 MHz, resulting in random number generation rates up to 130 MHz. We have also run successfully using a Perkin-Elmer APD (SPCM-AQR-13). In this case, however, we were further limited to rates of 5 MHz (to avoid damaging the detector); after all processing the SPCM-based system reached random number generation rates up to 20.1 MHz.

The detector pulse is read by an ACAM TDC-GPX Time-to-Digital-Converter [17]. This relatively inexpensive ($300) device is able to resolve detections with up to 27-ps accuracy. The APD pulses are directed toward the "stop" channel, while the start is periodically reset with a 100 kHz pulse to reduce any drift effects within the device. The time interval data is read by a field programmable gate array (Xilinx Spartan 3 FPGA) and input into a register. The number of bits assigned to each detection is dependent on our expected entropy per detection, and consequently on the average detection rate (e.g., it would not be appropriate to assign 8 bits to each interval if the average rate corresponds to only 6 bits of entropy per detection). The data string for each detection event is truncated to this calculated length, and concatenated in a register until enough data is present

to input into the hash function.

Finally, the hashed data is output through the PCI port on the FPGA and stored on a desktop PC, where further manipulation and testing can occur. Figure 2.6 below shows the data flow of our implementation.

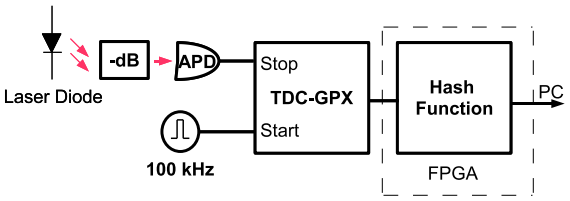


Figure 2.6: Data flow diagram for our implementation. Photons emitted from the laser diode cause the APD to output pulses which are registered by the counter. For each new detection, depending on the time elapsed between subsequent detection events (as determined by the TDC-GPX), the counter produces a random string. These counts are then accumulated until enough random data is present to "whiten" using the SHA-256 hash function. The final data is then output through the PCI bus.

The QRNG presented here has achieved a final output rate of 130 MHz. Additional methods of increasing this rate will be discussed in Chapter 5. This rate is well above that of currently commercially available QRNGs, which have random number generation rates of 4 MHz per detector [18].

CHAPTER 3

ENTROPY

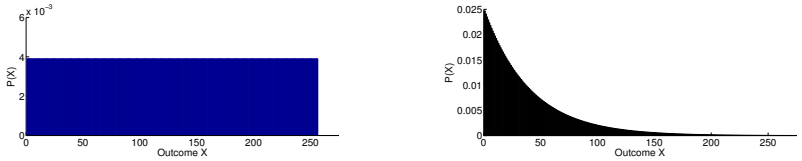
As discussed in the previous chapter, the processes that drive a PhRNG may be random, but not necessarily ideal. Instead of every possible output value having an equal probability of occurring, an inherent amount of bias may be present. In the case of the QRNG detailed in this thesis, the output probabilities take the form of a decaying exponential (instead of the ideal flat distribution). When this is the case, it becomes increasingly important to properly quantify the amount of randomness that is generated. In information theory this is also analogous to the amount of information contained in a message, and is referred to as *entropy*.

When used in this context, the term *entropy* typically refers to Shannon entropy [9] and will be referred to as such for the remainder of this thesis. There are, however, related measures of entropy that are of special interest in RNGs. In particular, the *min-entropy* [19] is a measure of the largest amount of information an attacker could gain from a single guess; it is often used in evaluating randomness in cryptographic systems. In this chapter, we will explore both types of entropy, as well as ways of maximizing the amount output in relation to our QRNG implementation.

3.1 Shannon Entropy

As discussed in Section 2.2, the Shannon entropy of a message with N possible outcomes is given by the equation $S = -\sum_{i=0}^{N} P_i \log_2(P_i)$, where P_i is the probability of the ith outcome. For random number generation, an ideal distribution, as shown in Figure 3.1(a), would have equal probability for each

outcome, giving its Shannon entropy the maximum value of exactly $\log_2(N)$ bits. Our QRNG, however, has an output waiting-time distribution of the form $e^{-\lambda t}$, where λ is the mean value and t is the time until the next photon is detected ($t = 0$ corresponds to the detection of the previous photon), as shown in Figure 3.1(b).



(a) (b)

Figure 3.1: Probability mass functions for (a) uniform and (b) Poissonian waiting-time distributions.

For our implementation, a possible outcome X corresponds to a time-interval measurement between two successive photon detections. The amount of entropy that can then be extracted depends primarily on the rate of incoming detections and the time resolution with which they can be measured. By increasing the resolution, more possible values can "fit" into the decaying exponential probability distribution, and thus the amount of available entropy is increased. Since the available entropy increases as more time-bins become resolvable, a reasonable approximation is that the average amount of entropy per detection increases by one bit every time N increases by a factor of two, as shown in Table 3.1.

Table 3.1: Example average entropy per detection for a rate of 11 MHz.

Time Resolution (ns)	Average Entropy (bits)
50	2.05
25	3.05
10	4.45
5	5.51
0.1	11.26
0.05	12.27

In the limit that infinitely small time resolution can be obtained, an infinite amount of bits can be extracted from any detection. It is important to mention, however, that having a resolution less than jitter of the equipment could introduce unwanted bias. This

argument is somewhat spurious, however, as the random deletion afforded by our detector's dead-time (time after a detection during which the detector will not register incoming photons) eliminates this effect from the final data.

3.2 Detection Speed vs. Shannon Entropy

While increasing the resolution results in a logarithmic increase in the rate of entropy generation, increasing the source rate has a somewhat different effect. In actuality, although increasing the rate results in more detections per *second*, it also lowers the available entropy per *detection*, as the average time-bin value gets lower. As shown in Figure 3.2, increasing the detection rate causes the waiting-time distribution to "shrink;' and become more predictable. In the limiting case, all detections fall into the first time-bin and no entropy is generated.

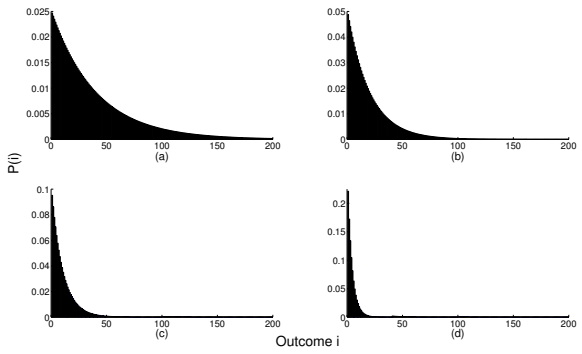
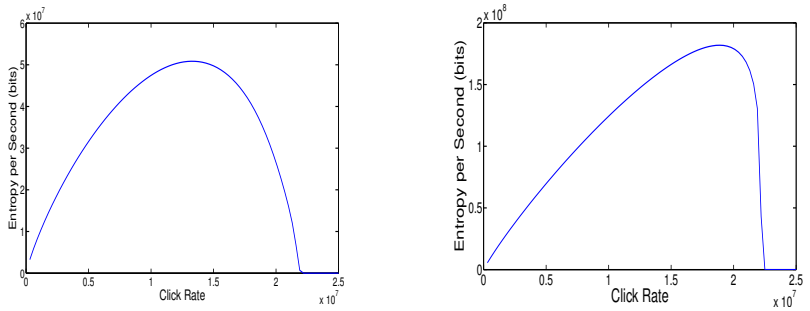


Figure 3.2: Theoretical Poissonian waiting-time distributions for our QRNG implementation. Average detection rates of (a) 5 MHz, (b) 10 MHz, (c) 20 MHz, and (d) 50 MHz result in curves of decreasing entropy.

To calculate the maximum amount of entropy generated per second, we must first calculate the available entropy per detection. For an average incoming photon rate R and time-bin size Δt, the probability of a photon falling into time-bin i is given by the equation $P_i = R\Delta t e^{-R\Delta t i}$. The total entropy per detection is then calculated by summing over all available P_i according to the Shannon entropy equation $S = -\sum_{i=0}^{N} P_i \log_2(P_i)$. Given an APD with dead-time DT, the "click-rate" (or rate at which photons are

registered) is given by $\mathrm{CR} = 1/(\mathrm{DT} + 1/R)$. Therefore, given an entropy value per click and a rate at which they are registered, we can multiply the two together to get the available entropy per *second*. Assuming that our time-bin resolution is fixed,[1] we can then calculate the entropy per second for varying detection speeds.



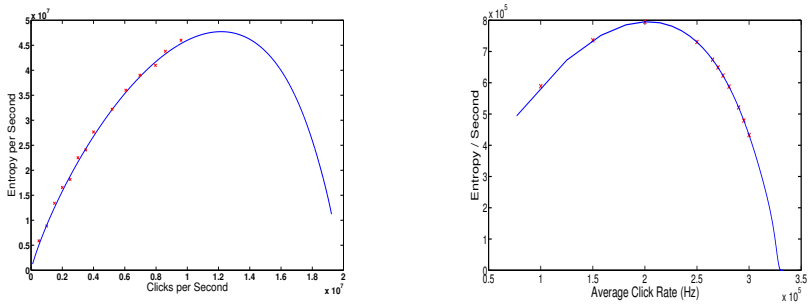
(a) (b)

Figure 3.3: Entropy per second vs. detection speed given a 45-ns dead-time and (a) 5-ns and (b) 27-ps time-bin resolutions. Note the different vertical scales on the two plots.

As shown in Figure 3.3, the maximum detection rates vary slightly with differing time-bin resolutions, with the peak approaching 1/DT as resolution increases. For very high rates, however, it is prudent to operate slightly below the maximum detection speed, as slight fluctuations can quickly saturate the detector, giving zero entropy per detection. Specifically, if the incoming detections occur immediately after every dead-time period, then all detections will fall into the first time-bin. The onset of saturation occurs when the detection rate approaches 1/DT.

Verifying the above curves was experimentally infeasible as the optimum operating speed was well above the maximum safe continuous count rate of our detector (11 Mc/s). However, by artificially increasing the dead-time by gating off the APD after a successful detection, we can lower the optimal detection speed, allowing for a direct comparison and verification of our theory, as shown in Figure 3.4.

[1] We have also considered variable bin widths and alternate bin ordering to reduce the amount of bias in the waiting-time distribution.



(a) (b)

Figure 3.4: Theoretical (curve) and actual (points) entropy per second vs. detection speed, given a 50-ns (a) and 3-μs (b) dead-time.

3.3 Min-Entropy

Of special interest in RNGs, particularly in security applications, is a measure known as *min-entropy*. A special case of Rényi entropy [19], it is given by the equation $S_{\min} = -\log(\max\{P_i\})$, where $\max(P_i)$ is the probability of the most likely event. For example, in our Poissonian waiting-time distribution, the most likely time-bin value is always the first. In some sense, the min-entropy is a measure of the "worst-case scenario," or the maximum amount of information that can be gained from a single attack. Therefore, in applications where privacy is important, the min-entropy can be considered of more importance than the Shannon entropy. In the optimal case, however, the min-entropy equals the Shannon entropy, as all N possible time-bin values would have the same probability $1/N$; in this case $S = \log_2(N)$.

As mentioned in Section 2.3, our random data is extracted from the waiting-time distribution of a Poissonian process, characterized by a rate parameter λ. Previously, we have assumed a *homogeneous* Poissonian process, one for which λ is constant and time-independent. Since our light source is a constant-current-driven laser diode operating well above threshold, this is a reasonable assumption, as the photon flux is directly proportional to the input current. It has been shown [5, 6], however, that by shaping the photon flux, the counting statistics can be altered, and the waiting-time distribution can be tailored to approximate the ideal, uniform case.

To determine how we need to modify the rate parameter we must consider an *inhomogeneous* Poissonian process. In this case, our rate is dependent on time, and the expected number of events between time a and b is $\lambda_{a,b} = \int_a^b \lambda(t)dt$. Consequently, the waiting-time distribution is now given by $\lambda(t)e^{-\int_a^b \lambda(t')dt'}$[20].

Given a waiting-time distribution with T possible time-bins, the ideal case is one for which the probability of every bin is time-independent, and exactly $1/T$. Therefore, $\lambda(t)$ must be a solution to the equation

$$\lambda(t)e^{-\int_0^t \lambda(t')dt'} = \frac{1}{T}. \tag{3.1}$$

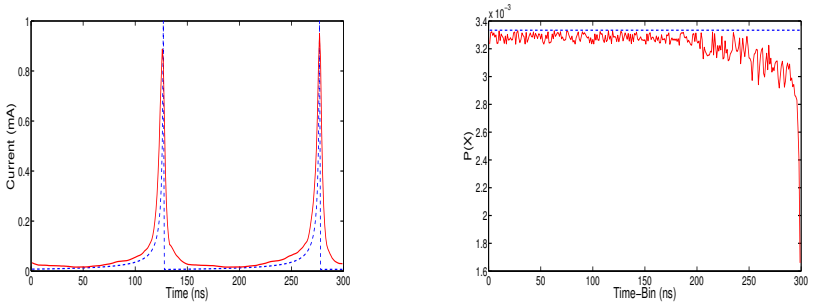
A rate parameter of the form $\lambda(t) = 1/(T-t)$ is a solution to this equation.[2]

Since $\lambda(t)$ represents the photon arrival probability, it is dependent on the photon flux of the laser diode, which in turn has a linear relationship with the input current. Therefore, if the current is equal to $I(t) = 1/(T-t)$, then the photon flux should be proportionally altered, achieving the ideal case.

As shown in Figure 3.5(a), this exact shape is impossible to produce, as the current grows rapidly, thus requiring a very high bandwidth and dynamic range and possibly damaging the diode. Approximating the shape, however, yields reasonable results, with a simulated min-entropy of approximately 0.96 random bits per bit. Additional details on circuit realization of this pulse shape can be found in Appendix B.

The choice of the reset period T depends on several factors. The entropy per detection increases on a logarithmic scale (i.e., to go from 6 to 7 bits requires 64 extra bins, while increasing from 7 to 8 requires 128), so an optimal reset period may not necessarily be the longest one. As seen in Figure 3.6, the optimal period is strongly dependent on the detector dead-time. If we had detectors with no dead-time, for example, the optimal reset period would be after 2 bins (or 4 bins, as the interval between 1 and 2 bits is the same). Unfortunately, however, the peak generation rate corresponds to a

[2] $-\int_0^t \lambda(t')dt' = -\int_0^t \frac{1}{T-t'}dt' = \ln(T-t) - \ln(T) = \ln((T-t)/T)$
. $\lambda(t)e^{-\int_0^t \lambda(t)dt} = \frac{1}{T-t}e^{\ln((T-t)/T)} = \frac{1}{T}$.



(a) (b)

Figure 3.5: Ideal (dashed) and simulated (solid) shaped pulse of $1/(T-t)$ (a) pulse-shape; (b) associated waiting-time distribution.

detection rate well above what our current APDs can sustain.

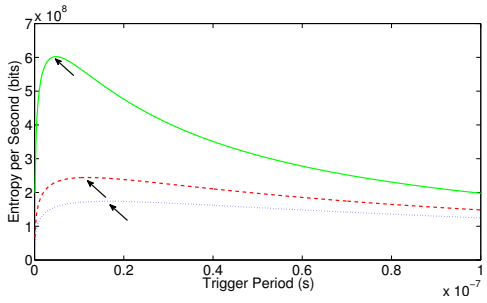


Figure 3.6: Peak min-entropy generation rate vs. trigger period T for 45-ns (short dash), 30-ns (long dash) and 10-ns (solid) dead-times. Optimal generation rates (as denoted by arrows) decrease with dead-time.

3.4 Min-Entropy vs. Shannon Entropy

Although min-entropy is arguably more critical for random number generation, earlier stages of our experiment were quantified according to Shannon entropy. Therefore, here we compare the two measures and their respective rates.

For the Shannon entropy "version," the input current is kept constant, and the detections typically come faster than in the shaped-current implementation (because of the higher probability of earlier time-bins). For the min-entropy version, however, this is not the case, as every time-bin is equally likely. Also, if no detection occurs before the reset period, then the counter is reset, and that time interval is wasted. For these reasons, the rate at

which Shannon entropy is generated is slightly higher than the min-entropy rate, as shown in Figure 3.7.

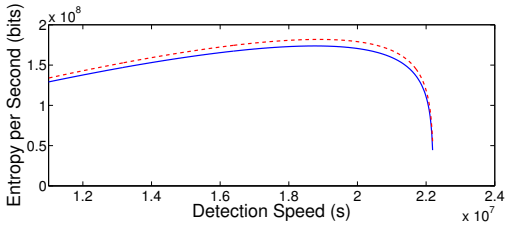


Figure 3.7: Shannon entropy (dashed) and min-entropy (solid) generation rates versus detection rate, given 27-ps resolution and 45-ns detector dead-time.

It is important to note, however, that although slower, the min-entropy version has several advantages. The amount of entropy input into the hash function is fixed at 266 bits (as explained in Chapter 4). Therefore, if each detection event contains more entropy, it will take fewer detections to reach the required amount. Since the hash is arguably the most computationally expensive component of our design, using the min-entropy allows for more FPGA resources to be used elsewhere. Additionally, from a security standpoint, the min-entropy version is more secure, as it relies less on a potentially insecure hash function.

CHAPTER 4

POST-PROCESSING

In order to ensure the quality of our QRNG output, a significant level of post-processing is required. Typically, when used in this context, *post-processing* refers to the process of reducing bias in the output, but for the purposes of this chapter it will also encompass testing the data for randomness. As uniformity (no bias of 0s or 1s) is only a single requirement for random numbers, examining the data with a series of widely accepted tests helps to identify other possible problems (i.e., absence of correlations and predictability). Although our current implementation carries out only a fraction of post-processing in real-time, future implementations will include the entire process.

4.1 Hashing

Because of the bias present in our waiting-time distribution, a data-hashing technique must be used to "whiten" the random number string, thereby preparing a shorter but more random string, with randomness approaching one random bit per bit (alternatively, as discussed in Section 3.3, one could use a shaped optical pulse to approximate a constant waiting-time distribution, though here too it is likely that *some* residual hashing would be needed). The hash function used in our QRNG implementation is the SHA-256 hash, of the SHA series created by NIST [12]. The hash takes as input a 512-bit block, some of which is predetermined and some of which contains the actual message. The output is a 256-bit hash string, with each of the 2^{256} possible strings having approximately the same probability.

A hash function, is by definition, a mathematical procedure that

converts a large, possibly variably sized, amount of data into a smaller hash value. Typically, when used in whitening applications, hash functions must satisfy two criteria: determinism and uniformity. Determinism is to the property that the same input *always* maps to the same output. While harder to prove, uniformity requires that the outputs be spread across all possible values with equal probability. Testing this completely would be computationally infeasible (e.g., 2^{256} is more than the number of electrons in the universe), but we have verified uniformity to a lesser degree by separating the output into smaller 16-bit blocks, as shown in Figure 4.1.

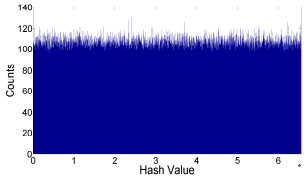


Figure 4.1: Final whitened data output from our QRNG.

Although 256 bits of data are output from the hash, it is imperative to realize that this does not necessarily correspond to 256 bits of *entropy*. A common mistake is to "seed" a hash with a smaller random bit string and still assume all of the output is random (if the seed is only 8 bits long, only 2^8 of the 2^{256} hash values will ever occur). Therefore, it is prudent to assume that the net amount of entropy output will always be less than the net input. A reasonable assumption, and one by which our QRNG operates, is that by providing 10 "extra" bits of entropy (i.e., 266 random bits input), the hash function will be sufficiently saturated. It can be shown [21] that 266 random input bits corresponds to an output entropy of 255.9999, or a Shannon entropy per bit of 0.999996. The corresponding output min-entropy is 255.28 bits, or 0.9972 random bits per bit.

In order to determine the amount of entropy input into the hash, we must first calculate the expected entropy per detection. For a measured average time-bin λ, the probability of a photon falling into time-bin i is given by the equation $P_i = \lambda e^{-\lambda t}$. The total entropy per detection is then calculated by summing over all

available P_i according to the Shannon entropy equation $S = -\sum P_i \log_2(P_i)$. Given an average entropy of N bits, then the hash input buffer is filled with blocks of size $266/N$, until the 266-bit requirement is met.[1]

4.2 Randomness Tests

In order to test the amount of randomness present in our data, we have performed several tests. Some, while applicable to the uniform hashed data, are not appropriate for the raw unhashed data, and vice versa. Although there is no true test to determine whether a sequence of bits is random (as the simplest example, the next block of bits could duplicate all the data up to that point), there are several widely accepted tests that we have utilized.

Several of these tests are designed to test a specific *null hypothesis*. In this case, the hypothesis is that the bit-sequence under test is random. Each of the tests creates a test statistic, which is then used to calculate an associated *p-value*, which is related to the strength of the evidence against the null hypothesis. This p-value is a value on the interval [0,1], with a p-value of 1 denoting perfect randomness and a p-value of 0 denoting perfect nonrandomness. A *significance level* (α) is then chosen for the tests. If $p \geq \alpha$, then the null hypothesis is accepted; i.e., the sequence appears to be random. If $p < \alpha$, then the null hypothesis is rejected; i.e., the sequence appears to be nonrandom. Typically, α is chosen to be 0.01 [13], meaning that, assuming the test is passed, the sequence can be said to be random (or nonrandom) with a confidence of 99%.

4.2.1 Autocorrelation function

The measured waiting-time data (of Figure 2.4) does not reveal anything about the *order* in which the underlying data was produced; if that is not random, then obviously the random number output would not be, either. To detect any simple

[1] In the case of an average entropy of a fraction of a bit, the value is rounded down.

frequency patterns we have performed an autocorrelation analysis on the raw prehash data. The autocorrelation function is used in signal processing and statistics to measure how well a signal matches a shifted version of itself. The autocorrelation of a "white-noise" signal — a signal with no frequency patterns — would look like a flat line except for a single sharp peak in the center. Our raw data exhibited this behavior, as shown in Figure 4.2.

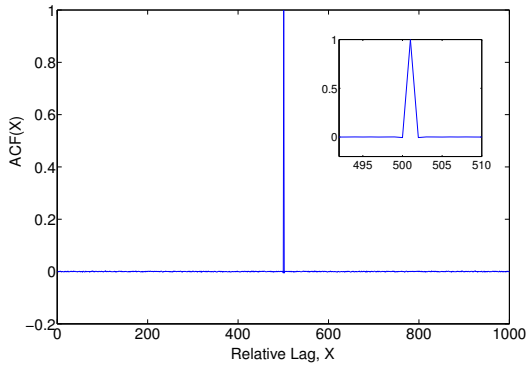


Figure 4.2: Autocorrelation function results from *prehash* data for a correlation distance of 500. Inset displays the solitary peak in the middle, an indication of random white-noise behavior.

Autocorrelation analysis was also performed on the hashed data, but is not directly relevant as frequency patterns would carry over through the hash.

4.2.2 χ^2 analysis

To test how well the data fits the expected distribution (both for raw and hashed data), a series of χ^2 tests were performed. For the raw data, the distribution was compared against a decaying exponential, while the hashed data was compared to the uniform distribution. Although testing the output of a widely accepted hash seemed somewhat unnecessary, it has proven useful in testing for various logical problems.[2]

The χ^2 *statistic* is calculated using the formula

[2]For example, a previous implementation had an error by which the last value in the buffer register was output twice. The χ^2 test was able to detect this, and subsequent autocorrelation analysis was able to pinpoint the cause.

$\chi^2 = (O_i - E_i)^2/E_i$, where O_i and E_i respectively are the observed and expected values of each outcome i. The resulting χ^2 statistic is then compared against a χ^2 *function* with the same degrees of freedom. For binned data, the number of degrees of freedom is equal to the number of bins [8]. Comparing the two values gives an associated p-value, that represents the probability that any deviation from the expected case was due to random fluctuations only. A generally accepted threshold for RNGs is a significance level of 0.01, which means that there is a 99% chance that observed deviations are due to chance alone.

For the whitened data, the average p-value across 100 trials was 0.505, with the lowest being 0.091 and the highest being 0.982. Therefore, our final QRNG data passed the χ^2 test with a 0.01 significance level. The raw counter data, however, failed, even when compared against a decaying exponential, which we attributed to after-pulsing.

After-pulsing, in the context of APDs, refers to the "false" detections caused by trapped carriers left over from the initial avalanche. For our detector, the peak after-pulse time is approximately $0.1\mu s$ after a "correct" detection, with a total probability of 3%. To verify that the deviation in the χ^2 test was, in fact, from this effect, additional analysis was performed. First, since the contribution to the waiting-time distribution due to after-pulsing should also be a decaying exponential (just shifted by 0.1 μs and at 3% of the original value), we superimposed the two together and did a χ^2 test again, for which the test passed with a significance level of 0.01. Therefore, we are fairly confident that at least most of the deviation is from after-pulsing. In any event, the nonrandomness from such after-pulsing is removed by the hash.

4.2.3 Overlapping serial test

Another common test for RNGs is the overlapping serial test (OST), or overlapping m-tuple test [22]. Although it is a variation of the χ^2 test against uniformity, the serial test is important as it can also test for correlations in the supposedly random data. The OST groups the random output into a cyclic string of tuples and,

after repeating the experiment n times, overlaps the outcomes and examines them for uniformity. For example, if after a particular time-bin x an electronic characteristic caused the next output to be time-bin y, then the tuple $[x, y]$ would be overrepresented and be detected by this test. The OST is also generally considered to be one of the most flexible and stringent RNG tests available, as adjusting the tuple size can reveal previously unknown characteristics.[3]

The result of the OST is a χ^2 value, which given a distribution with d possible values and tuple size t should have $d^t - d^{t-1}$ degrees of freedom. The p-value should be uniformly distributed between 0 and 1. If the tuples are too evenly distributed, the p-value will approach zero, while if they are too unevenly distributed, the p-value will approach 1. Unfortunately, given a random output string of length N, and t possible outcomes, if $k = \log_2(N)$, then the amount of memory required is on the order of 2^{kt+1} (e.g., assuming $k = 6$ and $t = 6$, we would need 36 GB of memory). Because of this constraint, we have performed this test for various combinations of k and t only up to a value of $kt = 24$.

Each tuple:output combination was tested 32 times, and each resulting p-value distribution was tested for uniformity using the Anderson-Darling goodness-of-fit test [24]. The QRNG is rejected if the p-value significance level is less than 0.01, meaning the distribution has a 1% chance of *not* being from the uniform distribution. For the tests where $kt = 24$, we tested the versions of $[k = 1, t = 24], [k = 2, t = 12], [k = 3, t = 8], [k = 4, t = 6], [k = 8, t = 3], [k = 12, t = 2]$, and $[k = 24, t = 1]$, for which all versions passed the test. An example p-value distribution for $[k = 4, t = 4]$ is shown in Table 4.1.

4.2.4 NIST and DIEHARD test suites

There are several test suites available for random number testing, but for this experiment we have chosen two of the most popular: the DIEHARD Test Suite [14] and the NIST Statistical Test Suite

[3]For example, the popularly used GNU Scientific Library contains 57 RNGs, 29 of which failed the OST for various tuple sizes [23].

Table 4.1: Example results from OST given tuple size of 4 and 2^4 possible values. The resulting χ^2 value should fluctuate around 61440.

χ^2	p-value
61017	0.8863
61483	0.4504
61443	0.4958
61011	0.8949
61170	0.7791
60968	0.9111
61044	0.8934
61510	0.3108
61524	0.2928

[13]. The DIEHARD suite contains 15 random number tests and requires approximately 3 million random samples. The majority of tests are run many times, and a resulting p-value is recorded. The p-value is obtained by $P = F(X)$, where F is the assumed distribution of the random sample variable X. The p-values should be uniform on the interval $[0, 1]$ if the input file contains truly random bits. The p-values are then tested for uniformity using the Kolmogorov-Smirnov (KS) Test [25]. Unfortunately, the approximation for $F(X)$ is asymptotic (i.e., uniform everywhere except approaching zero and one), for which the fit is the worst in the tails, so obtaining an accurate significance level is difficult.[4] In any event, the tests were run on 100 sets of 3-Mbit sets of our final whitened data, and the resulting p-values were tested for uniformity, as shown in Table 4.2. The lowest p-value recorded was 0.019 and the highest was 0.98, which is consistent with the expected results of the suite.

Similarly, NIST has released a statistical suite for pseudo-random and random number generators for cryptographic applications. As cryptographic applications require the highest-quality random numbers, this suite is especially stringent. As with the above, each test was run 100 times, and resulting p-values were tested for uniformity. The highest p-value across all tests was 0.994 and the lowest was 0.016, giving this RNG a passing value at a significance level of $\alpha = 0.01$. The results from this suite are shown in Table 4.3.

[4]Instructions for the test suite advise that p-values above 0.975 and below 0.025 are not abnormal; however, p-values of 1 or 0 out to six decimal places indicate the random number generator has significant issues.

Table 4.2: Results from DIEHARD Test Suite [14] for final whitened data.

Test	Result of DIEHARD Test
Birthday Spacings	PASS
Overlapping 5	PASS
Binary Rank Test 31	PASS
Binary Rank Test 32	PASS
Binary Rank Test 6	PASS
Craps Test	PASS
RUNS Test	PASS
Overlapping Sums	PASS
Squeeze	PASS
3-D Spheres	PASS
Min Distance	PASS
Parking Lot	PASS
Count the 1-s	PASS
OPSO	PASS
BitStream	PASS

Table 4.3: Results from NIST Test Suite [13] for final whitened data.

Test	Result of NIST Test
Frequency	PASS
Block Frequency	PASS
Cumulative Sum	PASS
RUNS	PASS
Long RUNS	PASS
Rank	PASS
DFFT	PASS
Non-Overlapping	PASS
Overlapping	PASS
Universal	PASS
Approx Entropy	PASS
Serial	PASS
Linear Complexity	PASS
Random Excursions	PASS
Serial	PASS

CHAPTER 5

FUTURE IMPROVEMENTS

Our QRNG implementation has four major elements: photon flux production, single-photon detection, time-interval measurement, and data processing. In this chapter, techniques for increasing both the quality and rate of random number generation will be discussed. Some of these methods (such as shaped-pulse-driven diodes) are specific to this application, while others (such as better detectors), have a broader range of applications. For each improvement, a qualitative discussion on how random number generation will be affected is presented.

5.1 Photon Production

As discussed in Section 3.3, we are currently exploring driving our light source with a shaped pulse of the form $1/(T-t)$, so as to reduce the bias present in our Poissonian waiting-time distribution. While increasing the amount of min-entropy present in our raw counter data, this improvement will significantly reduce the amount of required hashing, freeing up additional resources on the FPGA for real-time analysis. Additional details on the actual circuit are given in Appendix B.

5.2 Detection Methods

The method by which photons are detected is one of the most important parameters for our QRNG system. While detection *efficiency* is typically an important criterion when evaluating a single-photon counter, for our application this is not the case. Instead, we are more interested in the maximum detection *rate* and

the detector dead-time. As shown in Figure 3.3(b), our current rate of detection is not optimal for our time-bin size of 27 ps. Increasing the detection speed will allow a faster rate of random number generation, while also allowing for further improvements in time-bin resolution and dead-time to offer additional benefit. Therefore, we have researched several detection methods and evaluated them with respect to their expected performance benefit to our QRNG system.

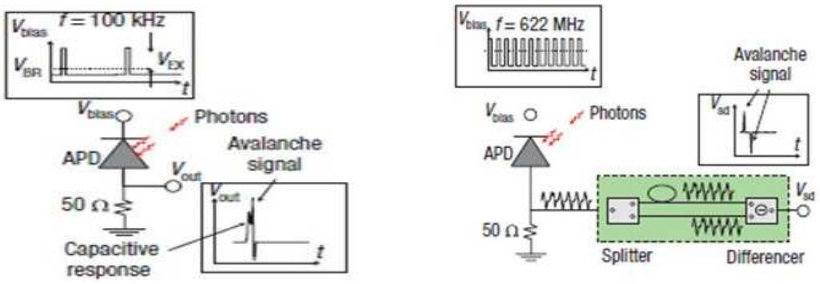
5.2.1 Photomultiplier tubes

A photomultiplier tube (PMT) is a high-efficiency photon-counting device used in many applications such as medical imaging, blood tests, and high-end imaging applications. A viable alternative to APDs, PMTs offer extremely low noise and very high efficiency. Of special interest to us is the fact that a PMT contains no Johnson noise, only quantum shot noise. This is overshadowed, however, by the requirement that when used in single-photon counting mode, a PMT has to be reset, thus limiting its continuous detection speed to rates well below the 1-GHz speed of the APD. Additionally, PMTs are extremely sensitive and can be burnt out easily, while APDs generally contain added circuitry to control the avalanche current. Therefore, at this time, APDs are a much more viable option.

5.2.2 Self-differencing APDs

Avalanche photodiodes (APDs) are widely used for single-photon detection because of their simplicity and robustness. When an APD is biased above the breakdown voltage, a single photo-excited carrier can quickly multiply as a result of impact ionization, producing an easily detectable current. Once triggered, the avalanche flows through the whole multiplication volume of the APD; because the avalanche is self-sustaining, it must be quenched by reducing the bias to a level below the breakdown voltage. This is typically done by operating the diode in gated Geiger mode, for which voltage pulses are periodically applied for several

nanoseconds to bias the APD above the breakdown voltage. While the pulses are high, the APD is enabled and can detect single photons; while they are low the APD is disabled, as shown in Figure 5.1(a). The resulting photon-induced signal is then compared against the APDs characteristic capacitive response. Since this capacitive response can also be quite large, this makes it difficult to detect much smaller avalanches.



(a) (b)

Figure 5.1: APDs operating in (a) conventional and (b) self-differencing Geiger modes, as shown in [26].

It has been shown, however, that by incorporating additional circuitry, it is possible to eliminate the capacitive response from the output signal, thus revealing much weaker signals and even enabling photon-number resolution [26]. As shown in Figure 5.1(b), the output signal is split into two paths, one of which introduces a delay of one period of an alternating bias voltage relative to the other path. The two signals are then subtracted from each other using a differencer circuit, allowing for the discrimination of avalanche currents more than 10 times weaker than what can be detected conventionally. Additionally, this method requires a much smaller applied voltage, allowing the detector to be operated at rates exceeding 1 GHz, after which after-pulsing becomes problematic.

Assuming *only* a maximum detection rate of 1 GHz (and not photon number resolution), this improvement alone would allow for our QRNG to operate at speeds exceeding 5 Gbit/s. If, however, photon number resolution were possible, other alternatives (such as using photon *number* as the quantum random variable) become available. Recently, these detectors were used in conjunction with

the beam-splitter approach, as shown in Figure 2.2, but because of resolution issues they were able to achieve only 4 Mbit/s [27].

5.2.3 APD arrays

Instead of a single APD per photon-counting module, currently available [28], multi-pixel photon counter arrays (MPPCs) can be used. Each array consists of multiple APD pixels operating in Geiger mode, with each pixel outputting a pulse signal when it detects a single photon. The total output from the MPPC is the sum of all the signals. While the largest device available has 1600 pixels, it currently affords no additional QRNG performance benefit over the single photon-number-resolving APD. If, however, it were possible to tell *which* pixel fired, this device could be treated as 1600 separate time-interval-based QRNGs operating in parallel and would allow the use of *spatial* information as well as temporal to generate random bits, as shown in Figure 5.2. Assuming a uniform beam profile and pixel efficiency (so that every pixel has equal probability of detecting a photon), and N pixels, each sample would provide $\log_2(N)$ random bits. This would be insignificant, however, next to the improvement from 1600 parallel QRNGs. However, one would need individual timing circuitry for each pixel. Additionally, different pixels would undoubtedly not have the same detection efficiency, potentially introducing unwanted bias into the final data.

5.3 Time Resolution Measurement

We are currently operating at a time-bin resolution of 27 ps, as afforded by our ACAM TDC-GPX Time-to-Digital Converter [17]. By relying on the known propagation times of inverters, this device offers a very simple implementation at a very low cost ($300). We recently evaluated a PicoQuant Picoharp 300 [29], which has a time-bin resolution of 4 ps. This device, however, is very expensive (approximately $20,000) and is infeasible for our current implementation. Therefore, at this time we have no immediate

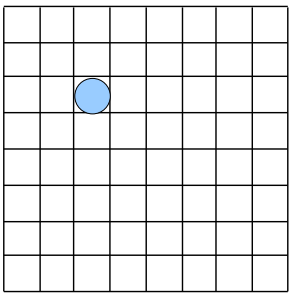


Figure 5.2: Example multi-pixel photon-counting array. Assuming 8×8 pixels, and the capability to resolve which pixel received a photon (as denoted by the circle), each detection event would provide up to 6 bits of random data. This assumes equal pixel efficiency and a uniform-intensity beamprofile, making this difficult to implement. Specifically, although one could measure the device characteristics and the light beam transverse intensity profile, and thereby quantify the randomness per detection (just as we did for our own time-based system), for secure operation one would need to monitor these characteristics every time the QRNG was used, in order to avoid undetected biases in the raw (unhashed) data.

plans to upgrade our time-bin resolution.

5.4 Data Processing

The data processing component of our QRNG has room for several improvements. The current FPGA (Xilinx Spartan-3) is an older model, and simply upgrading to a new version such as the Virtex-6 [30] would allow for more processing power and faster operation. Specifically, we plan to implement a system that will, in real-time, monitor the incoming time-interval data and perform several analysis operations.

Our current implementation expects a fixed amount of entropy per detection. This parameter is set during the device initialization and cannot be changed unless the QRNG is reset. Consequently, if the detection rate were to change suddenly (as a result of a change in the light flux or in the detector efficiency) and the entropy per detection were to go down, the system would not detect this condition and would then be assigning more bits per detection than it should. By periodically sampling the time-interval data and adjusting the number of random bits assigned to each detection, the quality of our random numbers will be increased.

Additionally, we would like the capability to perform, in realtime, several statistical randomness tests on our data. This would require substantial processing power and would certainly require an upgraded FPGA.

APPENDIX A

ATTENUATION

As discussed in Section 2.4, we have used a variety of techniques to attenuate the light to the desired photon flux. We have also assumed that, given a high enough level of attenuation, any unwanted effects arising from a non-ideal light source will be washed out of the final data. To verify this we have run a variety of simulations, but for the purpose of this discussion we will assume the most extreme case - one for which the light initially contains *zero* randomness.

In previous sections, when referring to the waiting-time distribution, we have defined the *ideal* case as one for which every time-bin has equal probability of occurring. While important, this definition does not encompass the entire problem. If, for example, the final waiting-time distribution was flat, but the *order* in which these events occurred was deterministic - while seemingly having a maximum entropy value, the data would be totally non-random. Therefore, we will now assume that the ideal case is one for which every value in the waiting-time distribution is independent of every other, or Poissonian.

A.1 Poissonian Light

A Poissonian *process* is a continuous-time counting process characterized by a rate parameter λ. The rate parameter is the *expected* number of events per unit time, or in our case, the average number of single-photon detections per second. For simplicity, we assume a *homogeneous* process, one for which λ does not change over time. In this case, the number of events that have occurred up to time t is $N(t)$, and the number of events in the interval

$(t, t+\tau)$ is characterized by $P[N(t+\tau) - N(t) = k] = \frac{e^{-\lambda t}(\lambda t)^k}{k!}$. Consequently, the waiting-time *between* arrivals is the same as the time until the *first* (since they are all independent), and is therefore given by $P[N(t) - N(0) = 0] = \frac{e^{-\lambda t}(\lambda t)^0}{0!} = e^{-\lambda t}$. For this distribution, the mean number of occurrences λ is also its variance, and thus fluctuates with standard deviation $\sigma_k = \sqrt{\lambda}$. These fluctuations are referred to as Poissonian noise, and are an important measure of randomness in our waiting-time distribution.

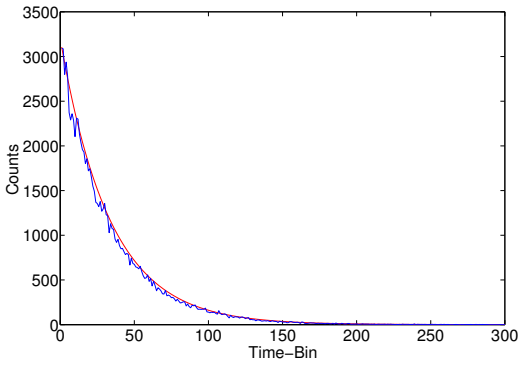


Figure A.1: Actual (blue) vs. theoretical (red) waiting-time distribution of raw data given an average detection rate of 6 MHz and a time-bin resolution of 5 ns. Deviation from the expected decaying exponential is due to after-pulsing and timing effects within the system.

As shown in Figure A.1, the measured waiting-time distribution does not exactly fit the expected $e^{-\lambda t}$ behavior. Given an infinite amount of time and samples, the two would approach each other. In the event that the noise is less than expected, the distribution is *sub-Poissonian*, and in the case that it is more, *super-Poissonian*.

A.2 Squeezed Light

It has been shown [15], that under certain conditions it is possible to generate light for which the number uncertainty is less than for a coherent state. Using an LED driven by current through a resistor, at sufficiently low temperatures the resistor's Johnson noise dominates over the quantum shot noise, giving the photon flux sub-Poissonian characteristics. Therefore, the resulting waiting-time distribution contains less uncertainty, a feature

undesirable for random number generation.

Fortunately, given any distribution, heavy enough attenuation (or random deletion), will cause the resulting distribution to approach a Poissonian [16]. To verify this we numerically modeled a totally periodic source, as shown in Figure A.2a. As every interval is the same, the waiting-time histogram contains only one value, which for simplicity has been assigned the first time-bin. Next, we simulate increasing the attenuation percentage. This was done using a RNG which outputs a number between 0 and 1. If the number was greater than the percent attenuation, that was counted as a detection. Otherwise, the time interval was increased and the RNG ran until the next detection.

As shown in Figure A.2, increasing levels of attenuation result in waiting-time distributions that, at higher levels of attenuation, approach the decaying exponential characteristic of a Poissonian. At an attenuation level of 95%, the difference between the two distributions decreased to less than one percent. Therefore, since we operate at an optical attenuation level of 99.9999%, it is reasonable to assume that any correlations are reduced to an insignificant level and may be ignored.

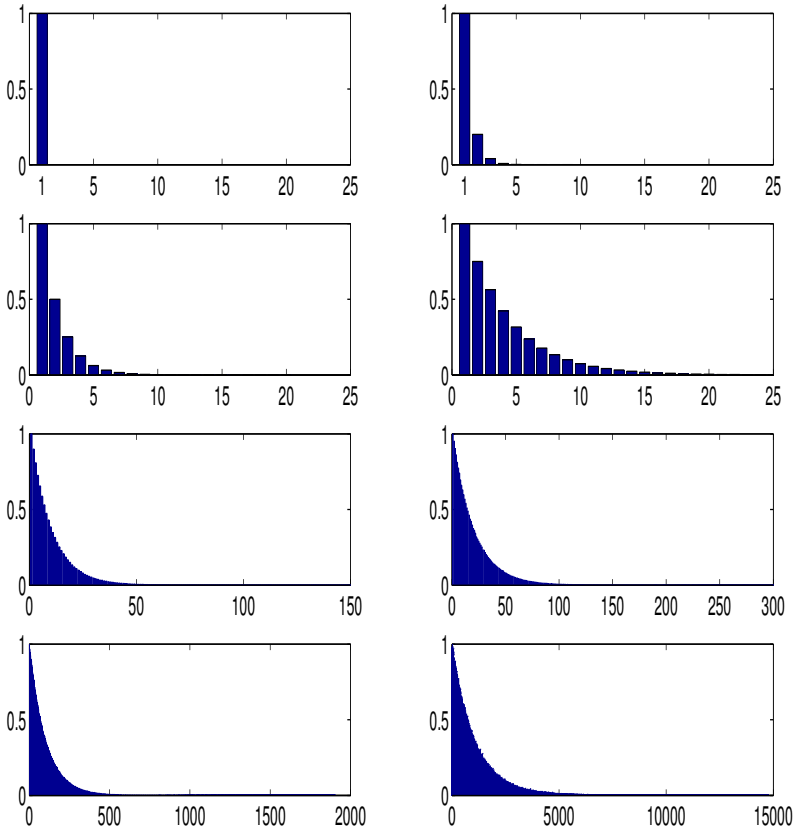


Figure A.2: Simulated waiting-time distributions of perfectly number-squeezed light for attenuation levels of 0% (a), 20% (b), 50% (c), 75% (d), 90% (e), 95% (f), 99% (g), and 99.9% (h).

APPENDIX B

SHAPED-PULSE CIRCUIT

As discussed in Section 3.3, in order to reduce the bias in our waiting-time distribution we have simulated a circuit which drives our laser diode in such a way as to make every time-bin occur with approximately equal probability. Given a desired uniform waiting-time distribution with reset period T, the output current must be of the form $\frac{1}{T-t}$, which is impossible to achieve. Therefore, we have approximated the pulse shape using the method detailed below.

The circuit in our design contains three major components: a sawtooth generator, logarithmic converter, and differentiator. Additional implementation specific components are required to achieve correct signal levels (gain stages, voltage followers, etc.), but the four main components can accurately approximate the pulse shape. The pulse is achieved by starting with a sawtooth (T-t), taking its natural log (ln(T-t)), and differentiating to achieve the final pulse shape of($\frac{1}{T-t}$).

B.1 Sawtooth Generator

The sawtooth generator is the first stage of our shaped pulse circuit. As shown in Figure B.1, the design for the sawtooth is fairly simple. Other components for impedance matching and voltage shifting will need to be added depending on the implementation specifics, but are not shown here. The left side of the circuit (the op-amp and resistors R3 and R4) form a bistable multi-vibrator, or a square wave. The square wave signal is then fed into the integrator formed by the right op-amp, forming a symmetric triangle wave. The symmetry is lost, however, due to

the values of the resistors R1 and R2. With a small R1, the capacitor charges fast, and with a large R2, charges slowly, thus forming the desired sawtooth shape, given by the equation $f = \frac{1}{2C(R_1+R_2)}\frac{R_3}{R_4}$.Due to the high bandwidth requirements of this circuit, several high-speed components had to be used. Specifically, OPA847 op-amps [31] and 1N4148 fast rectifying diodes [32], were used to achieve a bandwidth of approximately 3.9 Ghz.

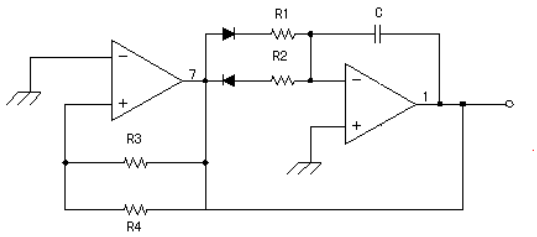


Figure B.1: Sawtooth generator circuit for our shaped-pulse implementation.

B.2 Logarithmic Converter

After the $(T-t)$ sawtooth shape has been generated, the next step in preparing the shaped pulse is to create the natural log of signal, or $ln(T-t)$. This is accomplished with a logarithmic converter [33], a device based on the precisely logarithmic relationship between collector current and emitter-base voltage in a bipolar transistor. Specifically, given a collector current I_c and emitter-base voltage V_{be}, the two are related by the equation $I_c = I_s e^{\frac{V_{be}}{V_t}}$, where I_s and V_t are device specific saturation current and thermal voltages. As shown in Figure B.2, transistor Q_1 is used as the non-linear feedback element around an op-amp. Negative feedback is applied to the emitter of Q_1 and the emitter base junction of $Q2$. This forces the collector current of Q_1 to be exactly equal to the current through the input resistor. Negative feedback forces the collector current of Q_2 to equal the current through $R3$. Since the collector current of Q_2 remains constant, the emitter-base voltage also remains constant. Therefore, only the V_{be} of Q_1 varies with a change of input current. The resulting

output voltage is given by $E_{out} = \frac{-kT}{q}\frac{R_1+R_2}{R_2} ln \frac{E_{IN}R_3}{E_{REF}R_{IN}}$.

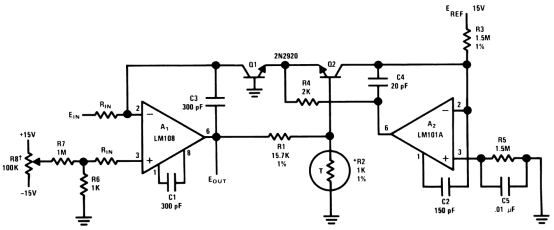
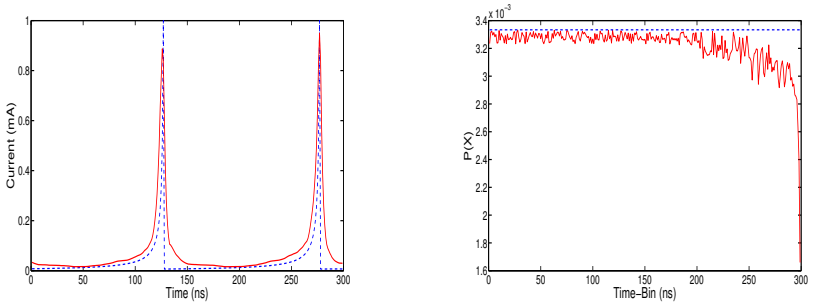


Figure B.2: Transistor based logarithmic converter design, as shown in [33]

B.3 Differentiator

In order to achieve the final pulse shape, a differentiator is required to convert the current from $ln(T-t)$ to $\frac{1}{T-t}$. This is one of the most elementary circuits built with op-amps, so greater detail will not be given to this component. As shown in Figure B.3, however, using the high-speed OP847 op-amp allows for a reasonable approximation to the final pulse shape.



(a) (b)

Figure B.3: Ideal (blue) and simulated (red) shaped pulse of $\frac{1}{T-t}$ pulse-shape (a) as well as associated waiting-time distribution (b).

REFERENCES

[1] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Review of Modern Physics*, vol. 74, no. 1, pp. 145–195, March 2002.

[2] A. Stefanov, N. Gisin, L. Guinnard, and H. Zbinden, "Optical quantum random number generator," *Journal of Modern Optics*, vol. 47, no. 4, pp. 595–598, March 2000.

[3] T. Jennewein, U. Achleitner, G. Weihs, H. Weinfurter, and A. Zeilinger, "A fast and compact quantum random number generator," *Review of Scientific Instruments*, vol. 71, no. 4, pp. 1675-1680, April 2000.

[4] P. Wang, G. Long, and Y. Li, "Scheme for a quantum random number generator," *Journal of Applied Physics*, vol. 100, no. 5, pp. 056107, September 2006.

[5] N. Lutkenhaus, J. Cohen, and H. Lo, "Efficient use of detectors for random number generation," U.S. Patent 7197523, March 27, 2007.

[6] P. Kwiat, E. Jeffrey, and J. Altepeter, "Quantum random number generator," U.S. Patent Application 20060010182, January 12, 2006.

[7] M. Stipceveć and B. M. Rogina, "Quantum random number generator based on photonic emission in semiconductors," *Review of Scientific Instruments*, vol. 78, no. 4, pp. 045104–4, April 2007.

[8] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C*. Cambridge, UK: Cambridge University Press, 1992.

[9] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp 379–423 and 623–656, 1948.

[10] W. Dultz and E. Hildebrandt, "Optical random-check generator based on the individual photon statistics at the optical beam divider," PCT Patent WO/98/16008, April 1998.

[11] J. G. Rarity, P. C. M. Owens, and P. R. Tapster, "Quantum random-number generation and key sharing," *Journal of Modern Optics*, vol. 41, no. 12, pp. 2435–2444, December 1994.

[12] National Institute of Standards and Technology, "Standard hash algorithm specification," April 2009. [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html.

[13] National Institute of Standards and Technology, "A statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," April 2009. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-22-rev1/SP800-22rev1.pdf.

[14] G. Marsaglia, Florida State University, "The Marsaglia random number CDROM including the Diehard battery of tests of randomness," April 2009. [Online]. Available: http://www.stat.fsu.edu/pub/diehard/

[15] P. R. Tapster, J. G. Rarity, and J. S. Satchell, "Generation of Sub-Poissonian light by high-efficiency light-emitting diodes," *Europhysics Letters*, vol. 4, pp. 293–299, January 1987.

[16] Bachor, H. A. Bachor and T. C. Ralph, *A Guide to Experiments in Quantum Optics*. Hoboken, NJ: Wiley, 2004.

[17] ACAM, "TDC-GPX time-to-digital converter," April 2009. [Online]. Available: www.acam-usa.com/Content/English/gpx/gpx_1.html.

[18] idQuantique, "Quantis – quantum random number generators," April 2009. [Online]. Available: www.idquantique.com.

[19] A. Rényi, "On measures of information and entropy," in *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability*, 1960, pp. 547-561.

[20] D. L. Snyder and M. I. Miller, *Random Point Processes in Time and Space*. New York, NY: Springer-Verlag, 1991.

[21] E. R. Jeffrey, "Advanced quantum communication systems," Ph.D. Dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 2007.

[22] I. J. Good, "The serial test for sampling numbers and other tests for randomness," *Proceedings of Cambridge Philosophical Society*, vol. 49, pp. 276–284, 1953.

[23] X. Xu and W. W. Tsang, "An empirical study on the power of the overlapping serial test," in *Proceedings of the Asia Simulation Conference*, 2007, pp. 298–306.

[24] T. Anderson and D. Darling, "Asymptotic theory of certain goodness of fit criteria based on stochastic processes," *The Annals of Mathematical Statistics*, vol. 23, no. 2, pp. 193–212, 1952.

[25] I. M. Chakravarti, R. G. Laha, and J. Roy, *Handbook of Methods of Applied Statistics*. New York, NY: Wiley, 1967.

[26] B. E. Kardynal, Z. L. Yuan, and A. J. Shields, "An avalanche-photodiode-based photon-number-resolving detector," *Nature Photonics*, vol. 2, pp. 425–428, June 2008.

[27] J. F. Dynes, Z. L. Yuan, A. W. Sharpe, and A. J. Shields, "A high speed, post-processing free, quantum random number generator," *Applied Physics Letters*, vol. 93, no. 3, p. 031109, 2008.

[28] Hamamatsu, "Multi-pixel photon counting module," April 2009. [Online]. Available: http://jp.hamamatsu.com/products/sensor-ssd/4010/index_en.html.

[29] PicoQuant, "PicoHarp 300," April 2009. [Online]. Available: http://www.picoquant.com/getfs.htm?products/picoharp300/picoharp300.htm.

[30] Xilinx, "Extended Spartan-3A FPGAs," April 2009. [Online]. Available: http://www.xilinx.com/products/spartan3a/.

[31] Texas Instruments, "Wideband, ultra-low noise, voltage-feedback operational amplifier with shutdown," April 2009. [Online]. Available: http://focus.ti.com/lit/ds/symlink/opa847.pdf.

[32] Fairchild Semiconductor, "1N4148 small signal diode," April 2009. [Online]. Available: http://www.fairchildsemi.com/ds/1N/1N4148.pdf.

[33] National Semiconductor, Appl. Note 30, pp. 1–5, September 2002.