Quantum Random Number Generation

Liran Dor, Jack Dorsey, Finnegan Rush, & Trin Wasinger 2023-09-26

Overview

We will be building a simple Optical Quantum Random Number Generator (QRNG) using polarization of photons. The goal is to have a functional—although not necessarily perfected—computer circuit capable of printing true random values to a console interface.

What's Special About a QRNG?

"Random" numbers in classical computers may appear random; however, they are actually anything but that. Almost all current implementations of computer generated random numbers use the idea of a Pseudorandom Number Generator (PRNG). In the most basic sense, PRNGs are wild—but deterministic—functions. Effectively, each *n*th call to generate a random number is *f*(*n*) where *f* is the PRNG function[1].

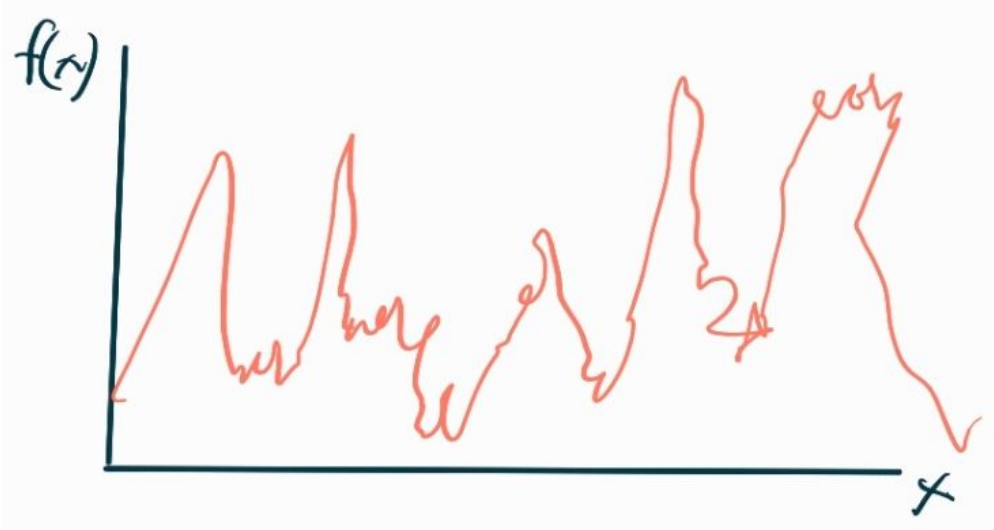

Figure 1: Sketched graph of a PRNG function

[1] This is an oversimplification. Most random functions take in the previously generated value rather than x; however, the implications are the same since *f* could be written recursively to capture this behavior.

```
fast(seed) {
    let [state] = Random.splitSeed(seed, 1, 53);
    return function() {
        const x = Math.sin(state++) * 10000;
        return (x - Math.floor(x));
    }
},
xoroshiro64s(seed) {
    let [a, b] = Random.splitSeed(seed, 2);
    return function() {
        var r = Math.imul(a, 0x9E3779BB);
        b = b ^ a; a = b ^ (a << 26 | a >>> 6) ^ b << 9;
        b = b << 13 | b >>> 19;
        return (r >>> 0) / 4294967296;
    }
},

splitmix32(seed) {
    let [a] = Random.splitSeed(seed);
    return function() {
        a |= 0; a = a + 0x9e3779b9 | 0;
        var t = a ^ a >>> 15; t = Math.imul(t, 0x85ebca6b);
        t = t ^ t >>> 13; t = Math.imul(t, 0xc2b2ae35);
        return ((t = t ^ t >>> 16) >>> 0) / 4294967296;
    }
},
```

Figure 2: Several common PRNG functions
(Implemented in TypeScript)[2]

The deterministic input-output nature of PRNGs can be useful but also has drawbacks. The pure-function nature of PRNGs is the reason Minecraft worlds with the same seed[3] always look the same. Again referring to Minecraft, this functional behavior is the reason why it is possible to predict the next “random” result if the current *n* is known or can be determined by fitting several *f*(*n*) to the known *f*. This predictability combined with an ability to force a call to the random function via some means allows one to manipulate *f*(*n*) for a particular future *n*th call. While these are all fun tricks for a videogame, the exact same ideas apply to all uses of PRNGs: cryptography, the lottery, etc.… A source of unpredictable, random numbers would be greatly beneficial in encryption and authentication especially as other quantum technology threatens security.

---

[2] Source available at:
https://github.com/SteveBeeblebrox/stevebeeblebrox.github.io/blob/main/assets/ts/random.ts

[3] Think of a seed as an initial offset added to *n*.

There are classical ways to generate “cryptographically secure” “random” numbers that are much harder to predict; however, even those are not truly random. Not even flipping a coin or rolling a die is really random. With enough information on position, velocity, geometry, and so on, the result is predictable; with enough practice one can influence the outcome. The only truly random things in the universe that we know of are Colorado’s weather[4] and Quantum Mechanics[5]. Using a QRNG circuit like the one described herein allows computers to take advantage of Quantum Mechanics provable randomness to generate true, unpredictable, random numbers.

In lecture, the team was intrigued by the discussion of the applications of Quantum Mechanics to national and personal security. Although there are many applications of Quantum Mechanics in the field of cybersecurity, developing a QRNG was chosen due to the time constraints, a desire to make something physical and beyond theory/simulation, and the ability of a QRNG to address a fundamental problem in modern computers. Additionally, error in a QRNG is not nearly as crippling of a problem as in other projects which made this an ideal beginning task.

QRNG Design

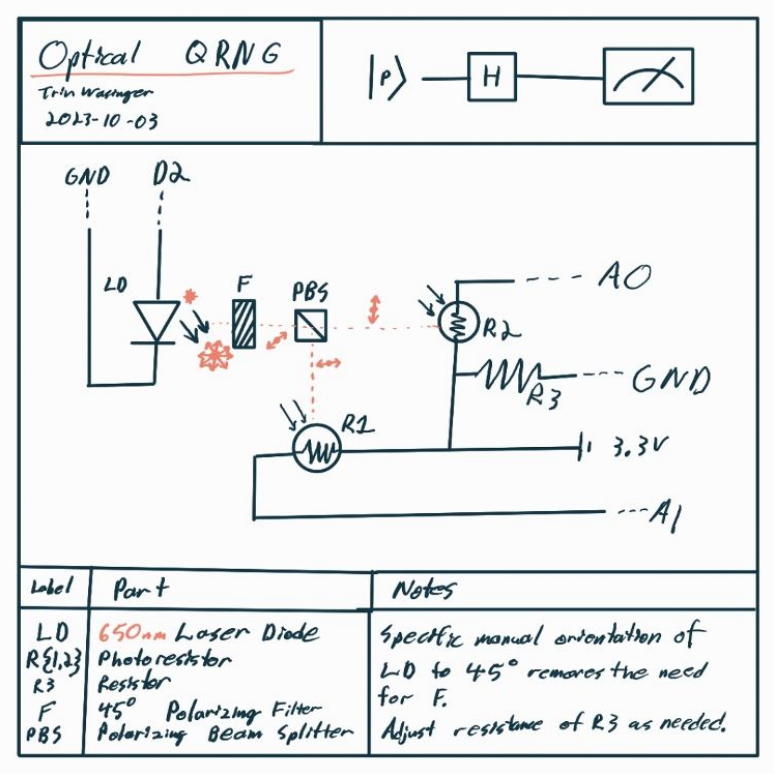

Figure 2: Diagram for an Optical QRNG

[4] Just joking, even that’s predictable.
[5] There is no “missing detail” responsible for the “randomness” like when rolling dice. At the quantum level, the universe is truly random. See the EPR Paradox and Bell Inequalities.

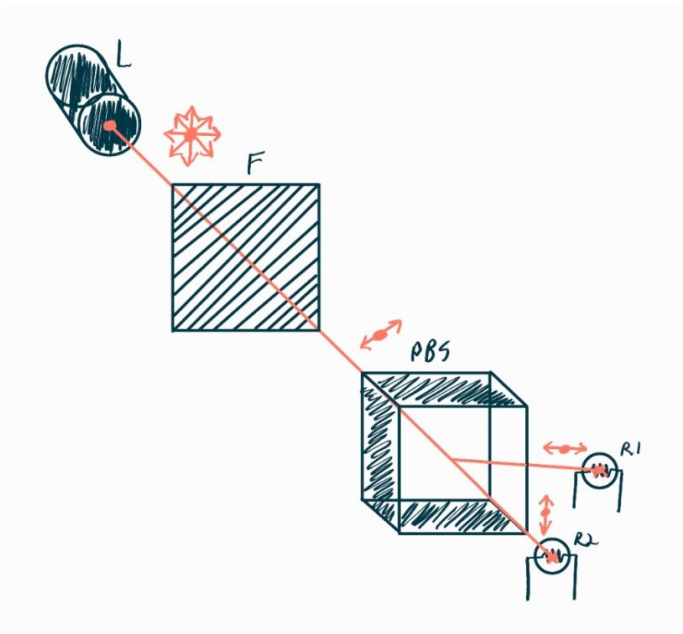

Figure 3: Isometric model of Optical QRNG

The photons coming from the Laser Diode (LD) may already be polarized; however, to be sure only the ideal values are obtained later on, a 45° polarizing Filter (F) is used. Any of the photons that make it through will now be polarized in a classical superposition[6] equally between vertical and horizontal.

$$|\nearrow\rangle = \frac{1}{\sqrt{2}}\left(|\leftrightarrow\rangle + |\updownarrow\rangle\right)$$

Equation 1: Mathematical representation of polarization

[6] Like a quantum superposition, a classical superposition is a combination of two base states; however, the coefficients are the weight of the respective states and *not* probability. The state can be measured to be in this combined state.

If no photons are being observed to make it through the filter, LD must be rotated 90° so that it is not perpendicular to the filter. The photons then travel through a Polarizing Beam Splitter (PBS) which sends vertically polarized photons down one path and horizontal ones along another:

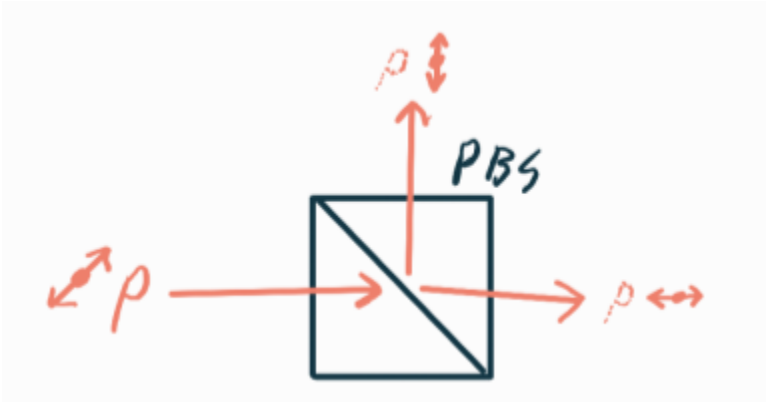

Figure 4: Polarizing Beam Splitter depiction

As it enters the PBS, a photon is in the above-mentioned classical superposition of horizontal and vertical, but a lone photon is “quantum” and cannot be divided. Thus, since it is horizontal and vertical, the photon takes *both* paths. However, it is impossible to measure the photon on both paths at one time. After moving through the PBS, the photon’s classical polarization superposition has been converted into a quantum superposition given by the same equation as the classical form (Equation 1). In this quantum superposition, the state can now only be measured to be in a basis state—path one or path two, and the coefficients of the new state, and thus the probabilities it appears in either basis state, come from the weight of the polarization to horizontal and vertical polarization states. If the photon passed through was more horizontal than vertical, it would be more likely to appear in the horizontal path; however, since the incoming photons to the PBS are at 45°, there is now a 50/50 chance that they are measured on one of the paths versus the other. Using photoresistors and a microcontroller or simple computer like an Arduino or Raspberry Pi, it is possible to measure which path was taken. This allows a computer to use the fundamentally random nature of quantum mechanics to generate true random numbers.

Converting Readings to Numbers

For random numbers with n digits, they are evenly distributed across $\{0,1,....,b^n\text{-}1\}$ in base 10 if every digit is evenly distributed across $\{0,1,...,b\text{-}1\}$ where b is the basis of the number (e.g., 10 for decimal or 16 for hexadecimal)[7]. The 50/50 measurement of getting a photon on a specific path is used to create a random digit in {0,1} where 0 and 1 correspond to the different detectors being hit. With repeated measurements of different photons, this process can be used to get an n-bit unsigned binary integer number evenly distributed in the base 10 range range $\{0,1,...,2^n\text{-}1\}$.

[7] The odds of getting a 3-digit, base-10 number xyz from random digits is $P(xyz) = P(x) * P(y) * P(z)$. With evenly weighted digits {0,1,….,9}, the odds of getting a specific digit are 1/10. Thus $P(xyz) = (1/10)^3 = 1/1000$ for any choice of x, y, and z.

Calculations

This optics project is light[8] on calculations. The classical superposition and quantum superposition can both be represented by Equation 1—although interpreted differently in each case. In reality, it won't be easy to detect a single photon with photoresistors; instead, the board can check which input pin gets slightly more voltage (corresponding to several extra photons on one path). If the amounts are equal or nearly equal—a reasonable result, it polls again. Otherwise, it should interpret the one with a higher analog reading to be the hit detector. Combining bits to obtain random bytes, words, double-words[9], etc… can be done with bit shifts and logical OR. No further preliminary computations were needed.

Tentative Assignments

Assignee	**Tasks**
Trin Wasinger *(Comp. Engineering)*	Circuit design and coding
Finn Rush *(Physics)*	Presentation and leadership
Liran Dor *(Physics)*	Circuit assembly and testing
Jack Dorsey *(Materials Science)*	Research and assembly

Table 1: Task Assignments

[8] Get it? "optics" and "light"?
[9] 8-bits, 32-bits (like int32_t in C++), and 64-bit (like int64_t in C++) respectively