Random Number Generation using boson sampling

In this challenge you will build an application that can generate truly random numbers using a quantum computing device called a boson sampler. You will be working on an application that could be one of the first commercial uses of quantum computation. You will have the opportunity to run it on the only publicly accessible quantum computer that has demonstrated quantum advantage.

Motivation

Random number generation (RNG) underlies a lot of the technology we use everyday. The most important of these is cryptography, where random numbers are used to generate keys to encrypt private information that must be completely unguessable to an attacker. This is like when you choose a password, if your password must be a string of 3 letters there are 52 possible characters (including upper and lower case) so there are 52^3 = 140,608 possible options for an attacker to guess. But if the attacker knows that your password is not completely random, for example if they know it starts with a capital letter and the rest are lower case then that reduces the number of options to 26^3 = 17,576, this makes their job of guessing 8x easier!

Exploiting weaknesses like this is called a *random number generator attack*, victims of these include have included Sony and early bitcoin users. More recently there have been concerns that one of the most widely used cryptographic standards, RSA, may be vulnerable to this attack. This technology underpins the secrecy of everything on the internet from your online banking information to securing the internet of things.

Random numbers also have applications beyond cryptography. Scientific experiments and surveys that use random sampling require them, as do lotteries and other chance-based games.

From Hardware-RNG to Quantum-RNG

The physical world includes lots of systems that can be harnessed to generate random numbers. For example, rolling a pair of dice or pulling numbers out of a hat. This has led to variety of physical measurements being used for RNG, such as thermal noise. While these chaotic systems are close to random, they do not have the true randomness of quantum mechanics.

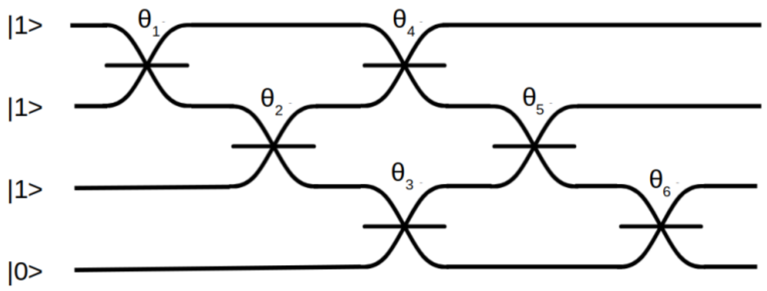
Recently there have been a number quantum random number generation (QRNG) experiments harnessing this fundamental randomness. You can go online and get a collection of random numbers generated from a vacuum state. However, this technique does not leverage the power of non-local interactions in quantum mechanics that can be used to amplify randomness, increasing the entropy of the numbers generated.

There is some cutting edge research that has proposed a new approach for QRNG that doesn't suffer from these problems. It uses a technology called *boson sampling* (explained below) that has attracted a lot of attention recently. Nearly two years ago boson sampling was able to demonstrate quantum advantage. Since then, boson sampling has become a commercially viable platform with ORCA's PT-series computer being released and sold to multiple users including the UK government. Xanadu have also demonstrated quantum advantage with their device that is now available for anyone (including you!) to use on Amazon's Braket service. Because boson sampling is cheaper to implement than fault tolerant quantum computers, and is already being deployed in a variety of businesses, it is the ideal platform to build the next generation of quantum random number generators.

Technical Background

Boson sampling

In boson sampling, identical single photons are sent into an optical interferometer, consisting of a network of beam splitters, which produces a complex probability distribution over photon numbers in the outputs. While seemingly simple, the quantum nature of photon interference makes the output of even relatively small interferometers with only a few tens of photons classically difficult to simulate within a reasonable amount of time.



This method of quantum computation replaces qubits with modes of light or 'qu-modes'. And instead of each qubit being only a $|0\rangle$ or $|1\rangle$, we count how many photons are in each mode, which can be any number from 0 to the total number of photons in the system. In this diagram, we send in 1 photon into each of the first 3 qu-modes, which interfere with each other at 6 beam splitters (similar to qubit rotations) with a tuneable angle (theta) and measure the outputs at the right. To run an experiment we choose an initial state (the number of photons in each mode) and set the angle of each beam splitter. After this we count the number of photons in each output mode, this is called a *sample.* We repeat this process many times to build up a distribution of the samples, this process is hard to simulate classically.

You can implement a boson sampling experiment with a few of lines of python using the Perceval or Strawberry fields packages. In this challenge you should familiarize yourself with one of these. If you want to use the Xanadu device to run your code, we recommend using Strawberry Fields.

RNG using boson sampling

To use one of these simulators to generate an unbiased random string of 1s and 0s you will need to implement the following algorithm:

1. Write code to draw samples from either a simulated boson sampler or a boson sampler available on the cloud.
2. Take two samples from this boson sampler S_1 and S_2
3. Apply von-Neuman post-processing (see below) to the two samples.

More details on this process can be found in this paper. To perform Von-Neuman post processing compare the two samples, if they both have either no photons or *any* number of photons we ignore that mode. If S_1 has no photons and S_2 has 1 or more photons output a '1', for the opposite scenario output a '0'. An example of this for an experiment with 5 modes is demonstrated in the table below.

Mode	*1*	*2*	*3*	*4*	*5*
No. of photons measured in S_1	0	0	1	1	2
No. of photons measured in S_2	0	2	1	0	3
Coding	-	1	-	0	-
Final String	10				

The Challenge

Using the algorithm described above design a product that uses a QRNG to solve a problem that is currently only addressed with classical approaches. In your solution you should aim to address one or more of these questions:

1. Can you simulate a QRNG using one of the packages above?
2. What is the best performance you can achieve on the tests above?
3. How does the rate at which you can generate random numbers compare to current applications? How does the rate change when you run it on real quantum hardware?
4. Why is this algorithm well suited to the business use case you have defined?

Using and analysing your random numbers

Once you have generated some random numbers you need quantify how random they are. There are some simple metrics like the entropy of the bit string and some more complicated industry standards like NIST SP 800-22. You should also consider how you can improve the performance of your QRNG. Could it be used as a *seed* for a pseudo-random number generator (PRNG) to increase the rate at which random numbers are generated?

If you are happy that you are creating genuinely random numbers at a high rate we would like to see what real world problems you can apply your QRNG to. You should consider the limitations of your algorithm when proposing a use case. You may also run your algorithm on a real boson sampler on AWS. What difference does this make to the quality of your random bit strings?

If you have any questions, head over to the Womanium hackathon discord to meet some of the ORCA team, get some advice, and discuss all things boson sampling!