Random number generation using IBM quantum processors

C Strydom and M S Tame

Department of Physics, SU, Matieland 7602, RSA

E-mail: conradstryd@gmail.com

Abstract. Random numbers are used extensively in both cryptography and simulation, but are difficult to generate reliably using classical methods. We investigate random number generation on the *ibmq_16_melbourne* quantum processor, a 15-qubit superconducting quantum computer. By applying simple post-processing techniques to the random bits generated, we were able to extract a sample of random bits which passed the NIST Statistical Test Suite. This shows that, with some post-processing, solid-state quantum computers such as IBM quantum processors can be used to generate random numbers of sufficient quality for cryptographic applications.

1. Introduction

Random numbers are used extensively in cryptography, the simulation of economic, traffic and agricultural models, as well as coordination in computer networks [1]. True random numbers are hard to generate using classical methods, as the unpredictability relies on an incomplete knowledge of a system, which can introduce ordered features. On the other hand, the inherent randomness central to quantum mechanics makes quantum systems ideal for generating true random numbers. A number of quantum random number generation schemes have been realised experimentally over the past two decades, many of which rely heavily on photonics [2, 3, 4, 5, 6]. Several photonic integrated circuits [7, 8, 9], as well as implementations on superconducting systems [10], have also recently been demonstrated.

In this paper, we further investigate random number generation on superconducting quantum computers. To this end, we generated a sample of random bits using the *ibmq_16_melbourne* quantum processor. Just as previous samples generated using IBM quantum processors [10], our sample showed a small bias towards zero. However, once we removed the bias by employing the von Neumann scheme [11, 12], the processed bits passed all 15 NIST tests [13], allowing us to show the successful generation of high quality randomness. The bits of Ref. [10] did not pass any NIST tests due to a lack of post-processing. After completion of the study, we were made aware that IBM recently released their own quantum random number generator with post-processing available via the University of Cambridge, which has also passed the NIST Statistical Test Suite.

2. Implementation

Bits of a classical computer can take only two values, namely 0 or 1. In contrast, qubits of a quantum computer can be prepared in either of two computational basis states, namely $|0\rangle$ or $|1\rangle$, or in any linear combination or superposition of these states. A random bit can be

generated using a qubit, by first applying a Hadamard to the qubit and then measuring it in the computational basis. Since all qubits are initialised in the state $|0\rangle$ on IBM processors, applying a Hadamard to a qubit prepares the qubit in the state $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. The state $|+\rangle$ has the property that measurements in the computational basis result in $|0\rangle$ with probability $\frac{1}{2}$ and in $|1\rangle$ with probability $\frac{1}{2}$. Thus, in the absence of noise, computational basis measurements generate uniformly distributed random bits.

The *ibmq_16_melbourne* quantum processor is a 15-qubit superconducting quantum computer and used to be accessible via the IBM cloud service [14], but has recently been decommissioned. By executing a circuit with Hadamards applied to all 15 qubits and computational basis measurements on all 15 qubits 200 times on the *ibmq_16_melbourne* quantum processor, with 8000 shots each, we were able to generate a sample of 24 million random bits. In what follows, we will refer to this sample of 24 million bits as the raw sample.

3. Results

3.1. Quality analysis

To determine if the raw sample has the expected properties of a true random bit sequence, we applied a number of standard tests [2, 3]. What follows is a brief description of each test as well as the results obtained for the raw sample.

In a true random bit sequence, the bits should be uncorrelated. To detect short-ranged correlations and periodicity in the raw sample, we calculate the Pearson correlation coefficient [15] of the full bit sequence with 1-bit to 40-bit delays of the sequence. The results are shown in figure 1.

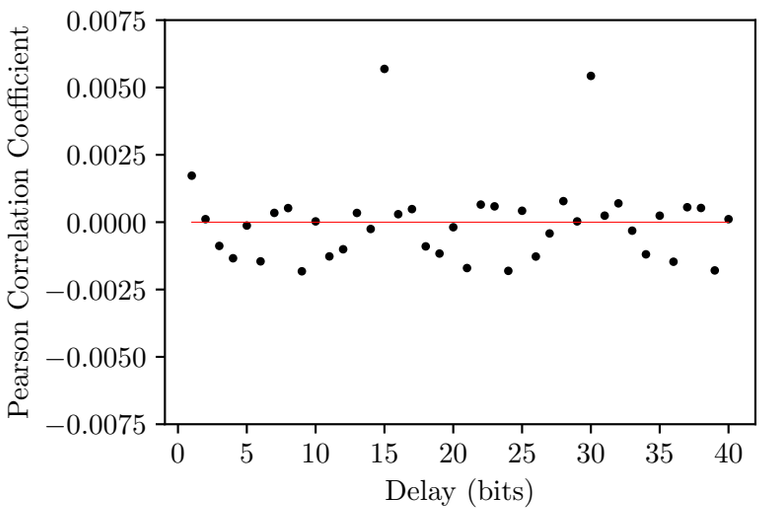

Figure 1. Pearson correlation coefficient of the full bit sequence of the raw sample (blue dots) and a true random sample (red line) with 1-bit to 40-bit delays of the sequence.

The Pearson correlation coefficient ranges from -1 to 1, where -1 implies perfect anti-correlation, 0 implies no correlation and 1 implies perfect correlation. Thus the results indicate that the bits in the raw sample are mostly uncorrelated, with small correlations present between bits at 15-bit intervals. This is likely because these bits were generated by the same qubit in the *ibmq_16_melbourne* quantum processor. Since each qubit has a unique error rate, bits generated by the same qubit show larger correlations than bits generated by different qubits.

All possible n-bit combinations should occur with equal probability in a true random bit sequence. We first consider single bits, for which we expect to find an approximately equal proportion of zeros and ones in a large enough sample. In the raw sample, the relative frequency of zeros and ones is 0.5262 and 0.4738 respectively. Hence the data shows a bias towards zero. As a result of decoherence of the qubits and errors that occur when gates are applied and

measurements are made, the two different computational basis measurement outcomes do not occur with a probability of exactly $\frac{1}{2}$ each. To investigate 8-bit blocks, we convert each 8-bit block in the bit sequence to an unsigned integer in the range $[0, 255]$. For a true random bit sequence, these integers should be uniformly distributed over the interval $[0, 255]$ and they should have an average of 127.5. Figure 2 shows the distribution of integers for the raw sample and their average is 120.87. The small average can be attributed to the bias towards zero.

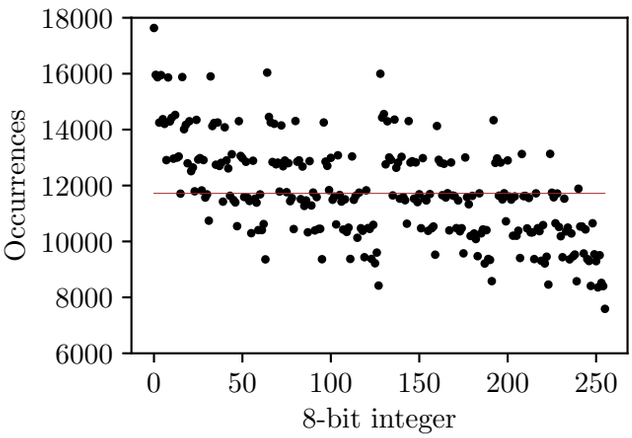

Figure 2. Distribution of integers for the raw sample (blue dots) and a true random sample (red line).

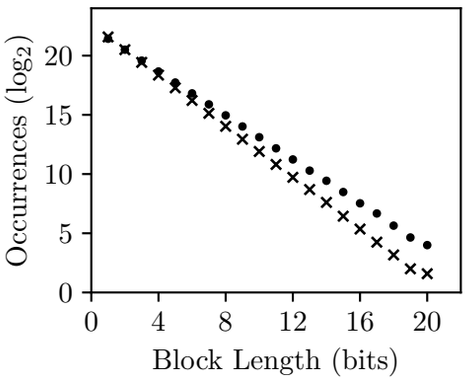

Figure 3. Run length distributions for zeros (blue dots) and ones (blue crosses) for the raw sample.

A run in a sequence of bits is a block of consecutive zeros or ones. In a true random bit sequence, a n-bit run of zeros or ones should occur with probability 2^{-n}, since zeros and ones each occur with probability $\frac{1}{2}$. The run length distributions for zeros and ones for the raw sample are shown in figure 3. The gradient of the best fit line is (-0.925 ± 0.002) and (-1.076 ± 0.005) for runs of zeros and ones respectively. For a true random bit sequence, we should have a gradient of $-\log_2(2) = -1$ for both zeros and ones. The deviations in the data can be explained as follows — since the zeros occur with a higher frequency than ones, runs of zero occur with a higher frequency than runs of one of the same length.

Entropy quantifies irregularity or randomness. The Shannon entropy is defined as

$$H = -\sum_i p_i \log_2 p_i \tag{1}$$

where $i \in \{0, 1\}^n$ is an n-bit string and p_i is the probability of obtaining i. The Shannon entropy should be n bits for a true random source of n-bit strings. The Shannon entropy for 8-bit strings in the raw sample is 7.98403 bits. This was calculated using relative frequencies obtained from the distribution in figure 2. The entropy is slightly smaller than for a perfectly random source as a result of order introduced by the bias towards zero and the 15-bit interval correlations.

Finally, we test the raw sample on a practical problem in simulation — estimating the value of π using a Monte Carlo method. Since the area of a circle of radius r divided by the area of a square of side length $2r$ is $\frac{\pi}{4}$, the value of π can be estimated by randomly placing points in a square of side length r which contains a quarter-circle of radius r. An estimate for π is calculated by dividing the number of points inside the quarter-circle by the total number of points used and multiplying by 4. Considering $r = 255$, and obtaining random positions for points using the unsigned integers extracted from the raw sample when investigating the distribution of 8-bit blocks, we obtain an estimate of 3.282 for π. This is larger than the true value of π, because the bias towards zero results in a larger proportion of points being placed inside the quarter-circle than for randomly placed points.

3.2. Post-processing

The quality analysis of the raw sample shows that the bias towards zero greatly reduces its quality. We therefore employ the von Neumann scheme [11, 12] to remove the bias and improve the quality. Given a biased sample of independent bits (say, in which 0 occurs with probability p and 1 occurs with probability q, where $p \neq q$), the von Neumann algorithm yields an unbiased sample of random bits. Provided that the bits are independent, the pairs 01 and 10 both occur with a probability pq in the biased sample and can thus be used as an unbiased source of random bits. Hence an unbiased sample is obtained from the biased sample by replacing occurrences of 01 by 0, occurrences of 10 by 1 and removing occurrences of 00 and 11. Applying the von Neumann algorithm reduces the size of the sample by a fraction pq of its original length. However, this large reduction can be avoided by applying the recursive von Neumann algorithm [12]. Instead of simply removing occurrences of 00 and 11, they are used to generate additional biased bit sequences, to which the von Neumann algorithm is then applied. This is repeated recursively, each time appending the output to the previous. The number of unbiased bits produced by the recursive von Neumann algorithm is arbitrarily close to the entropy bound [12], and so the algorithm is optimal in terms of output size.

Although some non-negligible correlations are present between bits in the raw sample, it is clear from figure 1 that these are generally small enough so that the bits can be considered independent and the von Neumann scheme can be applied. Applying the recursive von Neumann algorithm to the raw sample, we obtain a debiased sample of 23,795,395 bits. We applied the same five standard tests to this debiased sample. The Pearson correlation coefficients are much smaller for the debiased sample, as shown in figure 4, indicating that the 15-bit interval correlations have been removed by the rearrangement of bits which occurred when applying the recursive von Neumann algorithm.

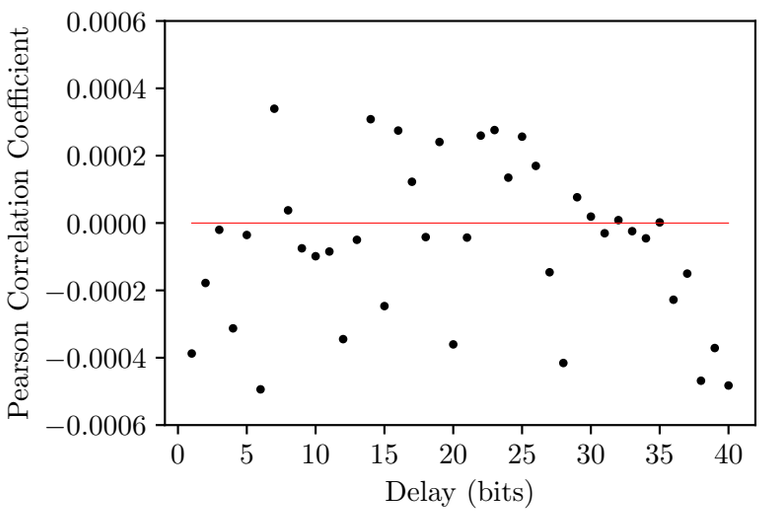

Figure 4. Pearson correlation coefficient of the full bit sequence of the debiased sample (blue dots) and a true random sample (red line) with 1-bit to 40-bit delays of the sequence.

The relative frequency of zeros and ones in the debiased sample is 0.5001 and 0.4999 respectively, which shows that the bias towards zero has indeed been removed. The distribution of integers is much closer to a uniform distribution, as can be seen in figure 5, and their average is 127.46 reflecting the removal of bias. The run length distributions for zeros and ones for the debiased sample are shown in figure 6 and also reflect the removal of bias. The gradient of the best fit line is (-1.004 ± 0.008) and (-1.007 ± 0.005) for runs of zeros and ones respectively, both of which are closer to the expected result for a true random bit sequence. Furthermore, we find that the Shannon entropy for 8-bit strings is 7.99994 bits and obtain 3.134 for an estimate of π, both of which show improvement in the quality of the random bits.

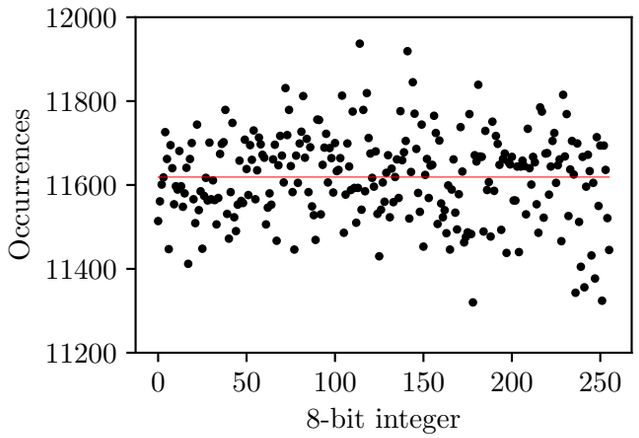

Figure 5. Distribution of integers for the debiased sample (blue dots) and a true random sample (red line).

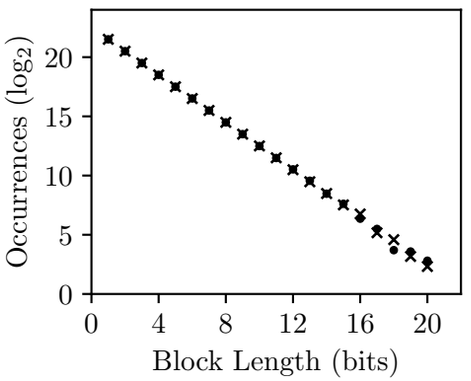

Figure 6. Run length distributions for zeros (blue dots) and ones (blue crosses) for the debiased sample.

Table 1. Summary of NIST test results for the debiased sample. A sequence passes a test if the p-value (the probability that a true random number generator would generate the given bit sequence or a less random bit sequence) is greater than 0.01. 'Required' is the required number of sequences which must pass a test for the sample to pass and 'Proportion' is the number of sequences which passed. The p-values quoted in the final column arise via applying a χ^2 test to the p-values obtained for the individual sequences and can be used to assess their uniformity. When a test consists of more than 5 separate tests (indicated with *) averages were taken.

Statistical Test	Required	Proportion	p-value
Frequency	230	234	0.231016
Block Frequency	230	234	0.735559
Cumulative Sums 1	230	233	0.236279
Cumulative Sums 2	230	232	0.043540
Runs	230	236	0.158711
Longest Run of Ones	230	235	0.247079
Binary Matrix Rank	230	234	0.547343
Discrete Fourier Transform	230	233	0.609979
Non-overlapping Template*	230	234.36	0.420221
Overlapping Template	21	22	0.392456
Universal Statistical	21	23	0.788728
Approximate Entropy	230	236	0.565084
Random Excursions*	10	12	0.358879
Random Excursions Variant*	10	11.89	0.344683
Serial 1	21	23	0.105618
Serial 2	21	23	0.484646
Linear Complexity	21	23	0.105618

3.3. NIST Statistical Test Suite

For a more stringent test of the quality, we applied the NIST Statistical Test Suite [13] to the debiased sample. This is an industry standard test suite for random number generators, aimed

at assessing their suitability for use in cryptographic applications. The debiased sample passed all 15 NIST tests at the required 1% significance level. The results are presented in table 1. The Overlapping Template, Universal Statistical, Random Excursions, Random Excursions Variant, Serial and Linear Complexity tests were run with 23 sequences, each consisting of one million bits. The other tests were run with 237 sequences, each consisting of one hundred thousand bits. The block length was adjusted to 1128 and 1000 in the Block Frequency and Linear Complexity tests respectively. Default values were used for the block length in the other tests.

The NIST test results for the debiased sample clearly demonstrate the importance of post-processing, since previous implementations on superconducting systems, in which no post-processing was done, did not pass any NIST tests [10]. Photonic quantum random number generators also generally require post-processing to pass the NIST Statistical Test Suite [1, 2, 9].

4. Conclusion

We generated a sample of 24 million bits using the *ibmq_16_melbourne* quantum processor. This raw sample showed a small bias towards zero, which was removed by applying the recursive van Neumann algorithm [12]. The resulting debiased sample passed the NIST Statistical Test Suite [13]. We therefore conclude that, with post-processing, solid-state quantum computers such as IBM quantum processors can be used to generate random numbers of sufficient quality for cryptographic applications.

Acknowledgments

We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team. We thank Taariq Surtee and Barry Dwolatzky at the University of Witwatersrand and Ismail Akhalwaya at IBM Research Africa for access to the IBM processors through the Q Network and Africa Research Universities Alliance. This research was supported by the South African National Research Foundation, the University of Stellenbosch, and the South African Research Chair Initiative of the Department of Science and Technology and National Research Foundation.

References

[1] Herrero-Collantes M and Garcia-Escartin J C 2017 *Rev. Mod. Phys.* **89** 015004
[2] Francis J T, Zhang X, Özdemir S K and Tame M S 2017 *Quantum Sci. Technol.* **2** 035004
[3] Jennewein T, Achleitner U, Weils G, Weinfurter H and Zeilinger A 2000 *Rev. Sci. Instrum.* **71** 1675
[4] Nie Y Q, Zhang H F, Zhang Z, Wang J, Ma X, Zhang J and Pan J W 2014 *Appl. Phys. Lett.* **104** 051110
[5] Nie Y Q, Huang L, Liu Y, Payne F, Zhang J and Pan J W 2015 *Rev. Sci. Instrum.* **86** 063105
[6] Shi Y, Chng B and Kurtsiefer C 2016 *Appl. Phys. Lett.* **109** 041101
[7] Abellan C, Amaya W, Domenech D, Muñoz P, Capmany J, Longhi S, Mitchell M W and Pruneri V 2016 *Optica* **3** 989–94
[8] Raffaelli F, Ferranti G, Mahler D H, Sibson P, Kennard J E, Santamato A, Sinclair G, Bonneau D, Thompson M G and Matthews J C 2018 *Quantum Sci. Technol.* **3** 025003
[9] Bai B, Huang J, Qiao G R, Nie Y Q, Tang W, Chu T, Zhang J and Pan J W 2021 *Appl. Phys. Lett.* **118** 264001
[10] Tamura K and Shikano Y 2021 *Int. Symp. on Mathematics, Quantum Theory, and Cryptography, Mathematics for Industry 2019 (Fukuoka)* vol 33 (Singapore: Springer) pp 17–37
[11] von Neumann J 1951 *Natl Bur. Stand. Appl. Math. Ser.* **12** 36–8
[12] Peres Y 1992 *Ann. Stat.* **20** 579–90
[13] `https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf` Accessed on 5 November 2020
[14] `https://quantum-computing.ibm.com/` Accessed on 5 November 2020
[15] Edwards A L 1976 *An Introduction to Linear Regression and Correlation* (San Francisco, CA: Freeman) chapter 4 pp 33–46