Qiskit Fall Fest ’24 Hackathon Report

Hrishikesh Makwana Chaitanya Gaur Anadi Goyal

October 19, 2024

Contents

1 Introduction

Quantum Computing in general, is exploited on the fundamental level of **superposition property of qubits**. A number of applications show prospects of being driven by quantum logic, much faster and efficient. However, as Feynman pointed out, a quantum algorithm can only be implemented by a quantum circuit and not a classical logic, we have a totally new strain of problems to concern ourselves with. Measurement of the qubits and resultant error introduced due to the required hardware is a major field of research. However, that does not stop us from exploring what all facets of life can be 'quantumly' tried.

2 Problem Statement

The hackathon presents us with a classic and familiar problem of *Random Number Generation.* The various combinations of qubits, which give rise to various **probabilistic states**, can be utilised to represent numbers in a particular range. A simple and intuitive approach to generate these numbers is to create superposition states that is implemented with the use of **H-gates** in the circuit.
However, superposition chain beyond the limit of 10 qubits can be dicey, and won't give satisfactory results (as we will see). Thus, some tweaks in the design help us through to achieve our objective of generating more uniformly distributed random numbers though at the cost of time of execution and other factors too. Lastly, there is also the factor of measurement error as mentioned earlier which tends to increase as the measurement process becomes more repetitive (which also will be evident later).
Here, is my approach for this problem attempting to solve these complexities.

3 Approach

3.1 Circuit design

We focuse on generating and analyzing random numbers using different random number generators - Bitwise, Chunked and Normal.

1. Bitwise - In this approach we use just a single qubit H-gate circuit to generate any n-bit random number through iterative measurement cycle, generating the n-bit number bit-by-bit.

2. Chunked - In this approach, we use chunks of qubits (like 3 or 4), to generate the complete number, also through an iterative process of measurement but lesser iterations than bitwise generation.

3. Normal - Lastly, the normal case as the name suggests, is where we have as many qubits in the circuit as required to generate the maximum number of the range. It requires measurement only once.

The output time and statistics are also noted for all three of the cases, in terms of mean, std. dev., min., and max. of the numbers generated by each. Also, we bin the range into intervals of equal length, to track the uniformity of number generation better. For eg. for a range of 0-10000, the time taken are:

1. Bitwise - 2m 31s
2. Chunked - 55 s
3. Normal - 10 s

However, the random number generation was dismal for Normal QRNG. Hence, there is a tradeoff between time taken/computation required and result efficiency. Now what about the error generated in this other than the measurement process? We bypassed much of the possible error by using a simulator and not an actual QPU. We need some error mitigation techniques in the case of actual quantum hardware.

3.2 Error Mitigation

We design a basic Normal QRNG of four qubits to demonstrate the effectiveness of error mitigation. Employing the least_busy server in QiskitRuntimeService, we utilise the free available run time to port our circuit on a quantum hardware. In the process, we employ Dynamic decoupling(DD), Measurement error mitigation(TXE), Gate Twirling and Zero-noise extrapolation(ZNE). We obtain the following output after deploying the circuit over the QPU and comparing the measurement results among the different cases.

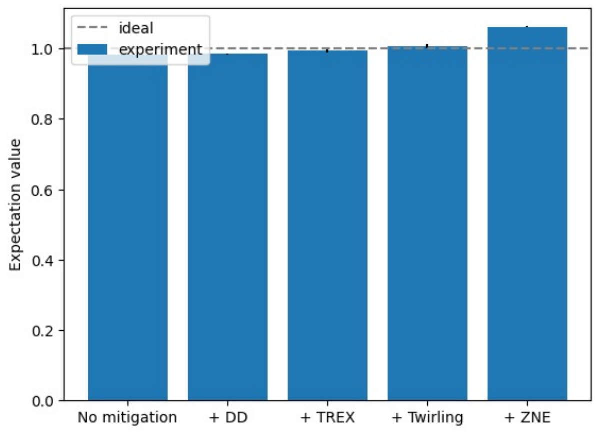

Figure 1: Error mitigation results

4 Conclusion

Thus, through these approaches we attempted to tackle the dual problem of obtaining our use case and minimizing the error encountered in the process. There are better and also more complex error mitigation techniques including tomography techniques but at the scope of current requirement we leave it to the future work. Refer the code files for better insight into the applied methodology.