Final report

Title: Quantum Calculator by using ‘Qutrit’

Team: Sogang’s Cat

Teammates: 김현우(Hyunwoo Kim), 남유정(Yoojung Nam), 이혜린(Hyerin Lee), 진승원(Seungwon Jin)

1. Problem

IBM Challenge topic #1: Build your minimal quantum circuit - making a quantum calculator

* Build a minimal quantum circuit to make 4-digit numbers adders including carry-up of digits with any technique you want to use.
* After building the adder, try to reduce the error of the adder on the real device, IBM Quantum Systems, as much as possible.
* Additional challenge: Try to make the UI of a quantum calculator.

\* The number of qubits available is limited to 7 for this challenge.

2. Background

At first, we tried to tackle ionQ challenge #4, which includes the problem of defining more efficient new data types (for example, int, float, char). Trying to solve it, it was needed to represent integers through the number system other than binary. To do so, We thought it would be useful to make a qubit to represent more states other than just |0> and |1> but we have soon realized that in qubit system only two states are allowed to be measured so it is not possible to simple make represent more.

While having stuck at ionQ challenge, we found out that that ‘IBM’s minimum quantum calculator building problem’ is quite similar to the ones we were thinking through; both problems required to minimize the number of qubits in use. In order to concentrate on our interest of defining multiple states in one qubit, we changed our challenge problem to IBM one.

In hindsight, we were confused with the concept of Qubit. Qubit is one of the quantum information unit realizations. To specify, qubit is a quantum information unit that has two states defined. Therefore, what we needed to do is constructing multi-level quantum information unit like Qutrit or Qudit.

In the following section, we built 4-bit quantum adder machine with 8 input qubits at first. Our team expected minimizing the number of quantum information units in the circuit would be possible by using Qutrit system.

3. Issues and how to overcome the limitations (solution)

If we construct 4 digit adder with qubit quantum circuit, it goes like the circuit below. We have used Half and Full Adders[1] to make a ripple carry adder below.

Diagram, schematic

Description automatically generated

And if we build n-digit carry adder, it would look like the one below. For convenience, 2-digit adder is constructed. (just as the above circuit, 3 or more digit adder could be built by adding extra full adders linearly) All input value a and b are acted with Hadamard gate to show all possible outcomes and the result appeared as expected (with noise). If we place X gates on certain inputs to make them 1s and leave the rest 0s, then the addition results come for such inputs.

Diagram

Description automatically generated Chart, bar chart

Description automatically generated

To make 4 digit adder, at least 12 qubits are required while the maximum allowed amount of qubits is 7. Since 8 qubits are assigned to set input values for the above case, it seemed really difficult to reduce those qubits circuit-wise. Therefore, we had to implement other methods and application of qutrit seemed highly appropriate.

In qutrit system, defining the three states are necessary. Our team chose pulse to define assign states. To minimize the error, we chose gaussian function as drag pulse and get the most reasonable frequency for ground state by frequency sweeping method from the backend’s default value.

To use pulse method, defining all three states are required, so we got 0->1 frequency at first. Then, by sideband method we tried to get the 1->2 frequency. However, while applying those method, all pulse supporting systems were blocked till the end of the hackathon, it was not able to find the local results. Therefore, we chose to discuss the further steps by getting through some reference codes’ results. [3], [4]

Chart, scatter chart

Description automatically generated Chart, scatter chart

Description automatically generated

While two – level systems boundary between states were quite clear and general, three-level states were not separable well. Each state was overlapped, and the mean data distance were so close that it was hard to get the good boundary between them.

Despite many disadvantages, three level system, aka qutrits are so powerful to represent data in small volume. Binary system with qubits can represent data with n-qubits. However, three level system can represent data with n-qutrits.

Anyway, since qutrit system is not stable and clear, pulse gates are not existing now. We proposed the truth table of CNOT gate. [5] The reason why CNOT gate is important is because it is used during the quantum error correction.

Qubit CNOT gate’s definition is

We induced Qutrit CNOT gate equation as

Trying all possible cases,

Based on the result above, we mathematically calculated the three CNOT gate, which is 9\*9 matix like this.

So CNOT Qutrit gates could be defined as above in matrix form.

If we have Qutrits defined and single/multiple gates in possession, then we can try building qutrit adding machine. To do so, we have to first find the pattern of addition in radix-3 calculation.

|  |  |  |  |
| --- | --- | --- | --- |
| SUM | 0 | 1 | 2 |
| 0 | 0 | 1 | 2 |
| 1 | 1 | 2 | 0 |
| 2 | 2 | 0 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| CARRY | 0 | 1 | 2 |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 1 |

After finding pattern of this operation, then we should try constructing this machine with qutrit resources. The sequence is building half adder first then full adder and finally we can build ripple carry adder in qutrit version. Unfortunately, we did not succeed getting all the gates in qutrit system so we didn’t have a chance to try so.

4. Further research and existing applications

- Error Correction [7] [8]

In usual cases, in order to fix the error, identical operations are executed. For the error correction, each bit is checked whether it has the same value from the other and correction is done according to ‘majority rule’. The state collapses if we compare bits to check their difference. However, bit comparing can be done without incurring state collapse if we use CNOT gate. Therefore, error correction is executed by first comparing the bits and secondly fixing the minority different valued bits.

Due to lack of available backends supporting pulse, qutrit realization have not been completed. Thus further implementations including whole gate construction, arithmetic machine building and error correction trial could be attempted.

Furthermore, by comparing the speed between qubit and qutrit, we can quantitatively analyze their efficiency.

**\*\*\* We issued our github repository**

[1] R. Feynman (1986). Foundations of Physics. Springer Science and Business Media

LLC. 16 (6): 507–531 . (<http://www.quantumdynamic.eu/doc/feynman85_qmc_optics_letters.pdf>)

[3]<https://github.com/0sophy1/Oct2021HKUST/blob/main/Lecture2/Part3%20Qtrit%20system%20with%20the%20Qiskit%20Pulse.ipynb>

[4] <https://github.com/CDL-Quantum/Hackathon2021/blob/main/Opacity%20Quantum/notebooks/accessing_higher_energy_states.ipynb>

[5] <https://www.iosrjournals.org/iosr-jap/papers/Vol10-issue6/Version-2/D1006021619.pdf>

[6] qutrit으로 iris data 분석한 예 <https://github.com/alejomonbar/Classification-Iris_using-Qutrits/blob/main/Iris%20Classifier.ipynb>

[7] Peter Shor, "Scheme for reducing decoherence in quantum computer memory", Phys. Rev. A 52, R2493(R) (1995).

[8] Dorit Aharonov and Michael Ben-Or, "Fault Tolerant Quantum Computation with Constant Error", arXiv:quant-ph/9611025 (1996).