|  |
| --- |
| **­­­­Various Applicated Adder implementation and their performance with Qiskit** |
| Team QAS Seunghoon Choi(Team leader) Electronical Engineering and Computer Science, GIST undergraduate student , Hyejin Yoon, Electronical Engineering and Computer Science, GIST undergraduate student , Se-bin Kim, Physics, GIST undergraduate student, Sumin Jin, Development of Software and Communications Engineering, Hongik university undergraduate student , Youshin Chung, Electronic Engineering, Korea University graduated student. |
| **Abstract**  As part of IBM challenge topic 1, we have built and tested several adders including full adder, ripple-carry adder on Qiskit. Then, we checked their performances by checking the propriate output of 4000 input. By this process, we have demonstrated transpiled CCX gate changed adder has most high performance among 4 adders in both cases. | | |

**1.Introduction**

**1.1 About us**

Our team is beginner for Quantum Computing, it was big challenge for us to use Qiskit and utilize Quantum gates. But our curiosity lead us to attend IBM Quantum Challenge topic 1: Build your minimal quantum circuit - making a quantum calculator. We realized ‘adder’ is one of the most important calculators on quantum computing, so tried to optimize the numbers of quantum gate composing adders.

On first day, we tried to minimize the number of gates of various adders featured in IBM challenge. But passing several hours, we learned that it is not efficient to come up with a way to reduce size of the circuits simply by using imagination. And by getting advice from mentors, we realized that QFT adder now we are using has not so much part to minimize.

So, what we can do was narrowed by two: 1.Error correction and 2.implementing various adders. We decided to concentrate on composing various type of adders on Qiskit. Taking one step forward from the first day, we try using Qiskit ‘Unroller’, ‘Transpiler’ for decomposing and so minimizing circuits. Further, we compared their performance with original adders.

**1.2 background**

**1.2.1 Full Adder**

Full adder is a circuit that gives following operation in Table 1 for given input A, B, Cin.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Cin | Sum | Count |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Table 1. Truth table of full adder

For a reversible machine, a full adder using quantum gate can be described in the following way in Fig.1. [1] We used this circuit when testing simple optimization and transpiler.

Chart

Description automatically generated

Fig 1. Full adder circuit using cx and ccx gate.

**1.2.2 Ripple Carry Adder**

Ripple Carry adder is an extension of full adder. This adder is for larger data (Fig.2). A quantum ripple-carry addition circuit is consisted of 2 steps. Adding each input bit unit and acquiring sum for each unit and Intitailizing each resistor for further usage.

Diagram, schematic

Description automatically generated

Fig 2. Ripple-Carry Adder for n=6

MAJ is a circuit that gives carry of ai and bi. After applying UMA circuit, we get sum of each unit and final carry. The architecture of MAJ and UMA is on Fig 3 and Fig 4.[2]

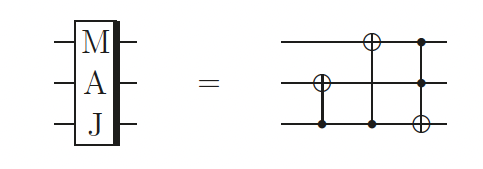


Fig 3. MAJ architecture

Diagram

Description automatically generated

Fig 4. UMA architecture

**1.2.3 QFT Adder**

Earlier, quantum addition algorithms mirrored their classical computers with necessary extensions for a reversible computation, but ideal addition algorithm for a quantum computer may not be similar to its classical counterpart. That’s why the quantum adder on the base of quantum Fourier Transform was invented. Fig 5. is the Diagram of QFT adder. Also, each gate can be depicted as a matrix form like Fig 6.[3]

Diagram

Description automatically generated Fig5. The quantum addition scheme based on QFT[4]

Diagram, schematic

Description automatically generated

Fig 6. QFT scheme and it’s mathematical form

**2. Method and Result**

As mentioned above, our project took three steps. The driving force was that CCX gate is fatal to efficiency of a circuit. First, we tried to think of an optimized unit without any devices. Since this attempt didn’t work well, we tried unroller to convert CCX gate to real device’s basis gates; ‘id’, ‘rz’, ‘sx’, ‘x’, ‘cx’ in Qiskit expression. The reason why basis gates are used is because they support immediate operation.[4] Using unroller, mentors advised us to use transpiler. Therefore, we made an applicated adder in 3 types. Doing so, we used decomposition and transpiler. After all, comparing each adder and applicated adders, we had a result which would bring out better performance.

**2.1 Full Adder**

We also diversified full adder in 4 version.

**(1) Original Full adder**

We brought full adder source from Feynman’s article[1] and composed it on Qiskit. By giving 4000 times input of ‘10’, we get 2669 times ‘10’(which is propriate).

Diagram, schematic

Description automatically generated

Fig 5. original full adder

**(2) CCX changed Full adder**

From (1), we changed CCX gate into CH-CZ-CH gates on Qiskit, and this time gived 4000 times input of ‘00’ and get 2741 times ‘00’(whick is propriate).

Chart, box and whisker chart

Description automatically generated

Fig 6. ccx gate changed original full adder

**(3) Transpiled original full adder**

This time, we transpiled original full adder(2) on Qiskit. By giving 4000 times input of ‘10’, we get 3023times ‘10’(which is propriate).

Diagram, schematic

Description automatically generated with medium confidence

Fig 7. Transpiled original full adder

**(4) Transpiled CCX changed full adder**

Lastly, we transpiled CCX changed adder(1) on Qiskit. By giving 4000 times input of ‘00’, we get 3173 times ‘00’(which is propriate).

Table

Description automatically generated with medium confidence

Fig 8. Transpiled ccx changed full adder

Table 2 is the table showing all the result from full adder section.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | (1) | (2) | (3) | (4) |
| rank | 4 | 3 | 2 | 1 |
| performance | 2669 | 2741 | 3023 | 3173 |

Table 2. Performance comparison in 4 cases of full adder

We have done 5 times of simulation for each adder with same input every time. And we plotted performances of each desired case for every trials. Fig 9 is the plotted graph.

**Chart, line chart

Description automatically generated**

Fig 9. Comparing 4 types of full adder performance

**2.2 Ripple-Carry Adder**

Before looking at the result, we want to explain path to acquiring final circuits that we are comparing. At first, using unroller, we constructed CCX with CX and unitary gate (Fig 10).

Chart

Description automatically generated

Fig 10. ccx replaced into u3 gate in ripple-carry adder

Though this is expected to be more efficient than circuit in (Fig 13), we wanted to change unitary gate into basis gate and replace CCX in ripple-carry adder. Since it was unable to decompose u3 gate into basis gate in unroller, we attempted to change it by hand. Fig 11. shows this process.

Diagram, engineering drawing

Description automatically generated

Fig 11. Attempt to decompose u3 gate

Transpiler made it easier to decompose ripple-carry circuit into basis gates. This function selects more efficient decomposed circuit than hand-written circuit. Therefore, we decided to use Transpiler for the final circuits compared.

Timeline

Description automatically generated

Fig 12. Rewriting steps of transpiler

The circuit that is a subject to transpiler process was following ripple-carry circuits: original circuit (Fig 13), modified ripple-carry circuit (Fig 14), transpiled original ripple-carry circuit circuit (Fig 15), transpiled modified ripple-carry circuit (Fig 16). We compared efficiency of these four circuits with 6 qubit input. Especially, the fourth circuit was expected to perform better than the other two, because this circuit is a circuit that has already gone through pass 2 in Fig 12.

**(1) Original ripple-carry circuit**

Chart, line chart

Description automatically generated

Fig 13. Original Ripple-carry circuit

**(2) CCX changed ripple-carry circuit**

Chart

Description automatically generated

Fig 14. Modified Ripple-carry circuit

**(3) Transpiled Original ripple-carry circuit**

A picture containing chart

Description automatically generated

Fig 15. Transpiled Original Ripple-carry circuit

**(4) Transpiled modified ripple-carry circuit**

A picture containing chart

Description automatically generated

Fig 16. Modified Ripple-carry circuit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | (1) | (2) | (3) | (4) |
| rank | 4 | 3 | 2 | 1 |
| performance | 1234 | 1092 | 1594 | 2501 |

Table 3. Performance comparison in 4 cases of full adder

Chart, line chart

Description automatically generated

Fig 17. Compating Ripple-Carry Adder Circuit

**2.3 QFT Adder**

**(1) QFT Adder circuit**

Chart

Description automatically generated with medium confidence

Fig 18. QFT Adder Circuit

The QFT adder consists of QFT, rotation part, and inverse QFT. Rotation part consists of 6 rotation gates. We implementation the Draper’s algorithm which is quantum adding operation based on QFT.

As a result of the QASM simulation, it was confirmed that was measured when = and =. However, as a result of implementation on the ibmq-jakarta system, the results were not measured correctly. Instead, we use an AER simulator that simulates with noise, and we confirm that is measured in the same way as QASM.

**(2) Transpiled QFT Adder circuit**

**A picture containing diagram

Description automatically generated**

Fig 19. Transpiled QFT Adder Circuit

As a result of transmitted, although the number of gates has increased, it is meaningful to try transfile and convert it into a basic gate.

|  |  |  |  |
| --- | --- | --- | --- |
| **Gate in Original circuit** | **# of gates** | **Gate in transpiled circuit** | **# of gates** |
| **Cp** | **12** | **Cx** | **60** |
| H | 6 | Rz | 42 |
| Swap | 2 | Sx | 12 |
| **x** | **1** | **x** | **7** |

Table 4. QFT Adder Circuit

**3. Conclusion**

Through our work, we obtain the sight of Circuit Optimization. We can now decompose the 3+ Quantum gate and replace them with more efficient circuit. Further, we will try to recombine the decomposed circuit into basis vectors. In the Future, we are planning to make web calculator that provides a service to calculate various applicated quantum adders.

**(1) Out big first step for Quantum Computing**

Thankfully, we get used to utilizing Qiskit by this challenge. More than just visualizing quantum circuit in jupyter notebook, we could use IBM cloud and access to backend to simulate our circuits.

**(2) Transpiler**

At first, we have tried Qiskit ‘Unroller’ to decompose the circuit so we can get optimized circuit. But We realized Transpiler is more efficient way to get minimun circuit. The transpiler function can analyze real device that we want to use. It was also crucial to well utilize the parameters of function. This will be our further challenge.

**(3) Changing CCX gate**

**The fact we noticed on Qiskit community was that we can change ‘CCX’ gate into ‘CH-CZ-CH’. And by out implementing this to ripple carry adder and full adder, we found out CCX changed version has less noise on output and even better when we transpile them.**

**4. References**

**[1]R. Feynman (1986). Foundations of Physics. Springer Science and Business Media LLC. 16 (6): 507–531.**

**[2]V. Vedral, A. Barenco, A. Ekert (1995). Quantum Networks for Elementary Arithmetic Operations.**

**[3]A.V.Cherkas and S.A. Chivilikhin, Quantum Adder of Classical Numbers, 2016 J.Phys.: Conf. Ser. 735.**

**[4] Qiskit Development Team, Transpiler, 2022.06.28.,** **https://qiskit.org/documentation/apidoc/transpiler.html**

**[5]Qiskit Development Team, Transpiler, 2022.06.28.,**

**https://qiskit.org/documentation/apidoc/transpiler.html**