UR-294

A Quantum Arithmetic-Logic Unit (ALU)

Abstract

We show that a quantum version of a classical Arithmetic Logic Unit (ALU) can be implemented on a quantum circuit. It would perform the same functions as a classical ALU, with the possibility of adding quantum functions in conjunction. To create the quantum ALU, we utilized IBM's Qiskit Python package and JupyterLab. We believe that a quantum ALU has the potential to be faster than its classical counterpart and the ability to calculate quantum specific operations. The simple classical functions translated to a quantum circuit show a promising future for the development of a full quantum ALU with unique quantum operations.

Background

Our work was inspired by *Quantum Computing based Implementation of Full Adder* by Sohel et al. They showcased a full adder implemented with IBM's Quantum Composer.

Technology

For development on this project, we setup local environments using Anaconda to maintain consistency within our codebase. On our local machines we also used JupyterLab to help create and test our code. The circuits were developed with Python 3.10 and IBM's Qiskit, their quantum computing package. We also utilized their Quantum Lab API to run the circuits.

Version Control

To manage version control we used Git, with its corresponding JupyterLab extension, and remotely hosted our code on GitHub. Every time a commit was made a Git hook was run which formatted our code using the Python formatter Black. Each member was responsible for their own branch, only pushing to a staging branch when a task was completed. We would then code review and push our updated final product to our main branch.

Results

Our final ALU is capable of correctly computing AND, OR, NOT, and ADD operations for 2 bits and an additional carry bit for the addition operation. The architecture of our circuit is designed with minimal parallel computation and is mostly serial. This is done to conserve qubits (quantum bits), which are a more limiting factor than circuit depth when it comes to resources on an actual quantum computer. The depth of our circuit is 30, indicating that at least 30 serial gate operations must be done back-to-back for any single qubit.

Operational Structure

The circuit is organized to compute outputs for all four operations and ignore the ones that aren't implied by the 2-bit opcode. 3 qubits are used for data input, 2 are used for the opcode, and 4 are used for the output for a total of a 9 qubit ALU.

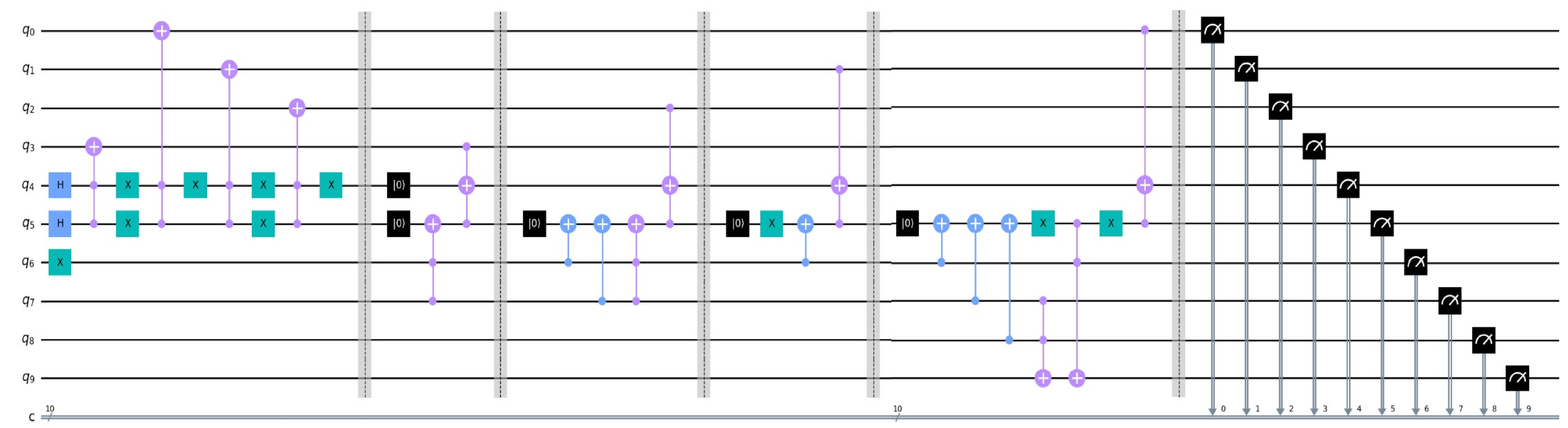

Fig.1 The full quantum Arithmetic Logic Unit, image created with IBM's Qiskit Virtualization module part of the Qiskit Python package. Qubits 1-3 are the data input, 4 & 5 are the opcode bits, and 6-9 are for recording the output.

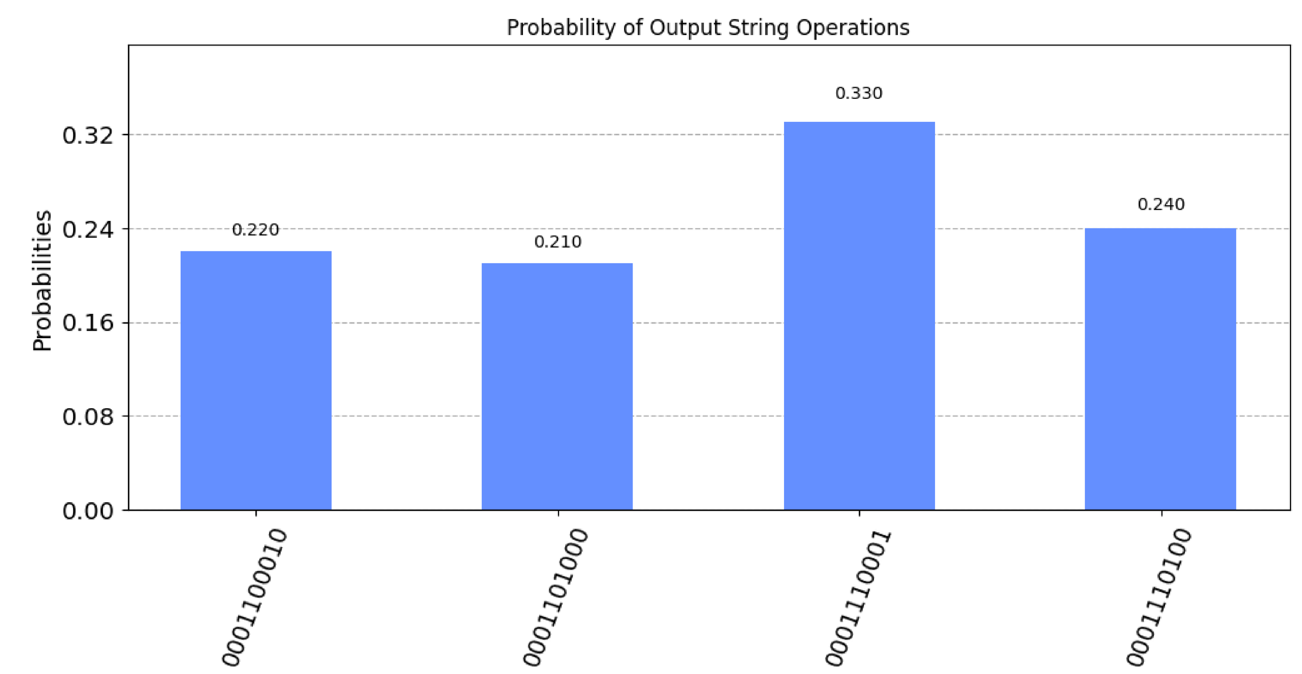

Fig. 2 Histogram showing probabilities for operations. Qubits are read right to left (i.e. the last 3 bits are the input)

Known Complications

Quantum computing is an emerging field which therefore means that quantum technology is also still in its infancy. Quantum computers produce "noise" which negatively influences the accuracy of the output. In addition to noise, the number of qubits needed is a limiting factor. Due to our circuit needing more than 7 qubits, we were only able to run this on a quantum simulator, and not on an actual quantum computer. This produces an output without noise. There are techniques to mitigate noise, but this will need to be covered in further research as it is outside the scope of our project.

Future Research

Quantum computing has a significant amount of progress that can be made. There are multiple things that can expand upon our initial work. Some examples are adding more classically analogous operations. This would make it more comparable to a standard classical ALU. In addition, implementing quantum operations specifically inside the ALU would be a novel approach that could possibly improve the efficiency to complete quantum computations.

Acknowledgments

Dr. Dan Lo – Acted as the project owner, research advisor, and gave instrumental guidance for completing our work.

Lecturer Sharon Perry – Acted as the project advisor and provided external resources for learning about quantum computing.

Website

Our project website was developed with Jekyll and GitHub Pages. More information about ourselves and the project can be found here: https://ksu-quantum-capstone.github.io/CS4850-DL1/

References

Quantum Computing based Implementation of Full Adder
https://ieeexplore.ieee.org/document/9298394

Qiskit Textbook
https://qiskit.org/textbook/

More extended references can be found at on our project website at https://ksu-quantum-capstone.github.io/CS4850-DL1/refs/

KENNESAW STATE UNIVERSITY
COLLEGE OF COMPUTING AND SOFTWARE ENGINEERING

Author(s): Ethan Butler, David Carroll, Bryson Phillip, Ben Ulrich
Advisors(s): Dr. Dan Lo, Lecturer Sharon Perry