Cirq: A python framework for creating, editing, and invoking Quantum circuits

Design Document

Team 8
Victory Omole
Akhilesh Tyagi
Calista Carey
AJ Hanus
Andrew Hancock
Austin Garcia
Jake Shedenhelm
Jordan Cowen
sdmay20-08@iastate.edu
https://sdmay20-08.sd.ece.iastate.edu

Development Standards & Practices Used

For this project, we will be using python and conducting py tests, and we will be coding in VSCode. We will be working in Agile sprints starting in the second half of the semester when we begin designing the algorithm for Cirq.

Summary of Requirements

- Implement the functionality to output a cirq.Circuit in QUIL
    - https://github.com/quantumlib/Cirq/issues/2386
- Study Quantum Computing
- Complete smaller issues assigned to group

Applicable Courses from Iowa State University Curriculum

- CS 309 - Agile practices, GitHub experience
- CS 104 - Python course

New Skills/Knowledge acquired that was not taught in courses

- Quantum Computing
- Python

Table of Contents

1 Introduction

1.1 Acknowledgement

We would like to thank Victory Omole, our client, and the rest of the Cirq developers for providing us with the Cirq repository and a good base to begin our implementation. Our client has also provided us with significant assistance for gaining knowledge in Quantum Computing by providing us many lecture notes and other material.

1.2 Problem and Project Statement

Cirq is a programming language for quantum computing. QUIL is an instruction set architecture being developed by Rigetti Computing for quantum processors. Our main goal is to translate quantum circuits designed in Cirq to QUIL. This would allow people to use Cirq-designed quantum circuits in programs/machines that only run QUIL such as the QVM (Quantum Virtual Machine) designed by Rigetti. Cirq already has the ability to translate to other quantum protocols such as QASM.

Using QASM as an example, our group can see that we will need to implement translation for each gate and a parser for the Cirq circuit. We will also need to find out what QUIL supports and what it doesn't in order to make sure all gates translate well.

We plan on completing a well working implementation with clean code and thorough documentation. During this process we plan to gain a deeper knowledge of Cirq, QUIL, and quantum computing in general.

1.3 Requirements

1. Implement issue #2386 in the Cirq repository (https://github.com/quantumlib/Cirq):
   For this requirement, we need to implement the file, quil_output.py. Cirq allows of conversions between different Quantum Computing circuit specifications, and they want to allow a circuit to be exported as QUIL. This will be similar to how Cirq already outputs QASM.

   We will be using python and coding in Visual Studio.

2. Study quantum computing:

Group members are required to read up on quantum computing and get comfortable with the concept before getting the main issue assigned to us. In order to fulfill this requirement. We must watch and read all lecture slides that our client, Victory, assigned to us as well as looking up more material. We will also attend lectures suggested by our adviser, Professor Tyagi.

3. Complete smaller issues:

For the first part of the first semester, our group members will be completing smaller issues in the Cirq repository in order

1.4 Intended Users and Uses

The intended user for our algorithm is anyone who uses quantum computers. Many scientists use quantum computers to conduct virtual experiments [3].

Quantum computing is also used to search through large amounts of data, so anyone working with a database would find great use in using Cirq and our implementation of the decomposition algorithm [3].

1.5 Assumptions and Limitations

Assumptions:

- The current code in the repo does not break.
- Current code can withstand large data tests.

Limitations:

- Right now, it is hard to make a quantum computer on a large scale [3].
- Currently have basic knowledge of Quantum Computing.
- In experience with Cirq database
- Not a 1-to-1 relationship as to how circuits in Cirq are used and how QUIL must be exported.

1.6 Expected End Product and Deliverables

We are working for the google repository, Cirq, so it is important that we keep our client and other developers up to date with the work our group does throughout the entire process. Because of this, we will test and push often, but we will be working in monthly sprints. All work for the sprint should be completed by the last day of the month (except December), but we should always be updating the repository and documentation throughout each month.

First Semester:

September 30, 2019:

By the end of September, all group members should have started researching and studying Quantum Computing. They all should have watched and read all lecture notes that our client provided us. The members that have not had previous experience with python need to start learning the language. Cirq works exclusively in python, so it's crucial all group members know the language prior to starting the actual implementation.

October 31, 2019:

Throughout the month, smaller issues will be assigned to group members in order to familiarize themselves with the Cirq repository. All group members should have all smaller repository issues that were assigned completed. Along with this, all group members should research QUIL and gain a thorough understanding of the similarities and differences it shares with Cirq.

November 30, 2019:

By the end of November, we will have begun writing a basic parser and working on converting the gates to QUIL. We will continue to look into the commonalities between Cirq and QUIL. Weekly Logs 5 and 6 will also be written and uploaded to our team website.

December 13, 2019:

This month is shorter due to the fact that there are finals and winter break beginning this month. In December, all official roles will be assigned to the project. Everyone will know which specific part they are working on in the project. After completing the design, we will update the documentation in the repository to include our design choices and diagrams we created. The last Weekly Log will also be completed and uploaded.

Second Semester:

January 31, 2020:

By January 31, we will have begun the first implementation of our design. We should have a good base of quil_output.py complete. The initial program should be almost ready to be tested.

February 29, 2020:

By the end of the month, our first implementation should be completed. We will begin testing the file and simulating test cases. We will make improvements to our initial code as needed, and get ready to optimize the solution.

March 31, 2020:

By the end of March, we should begin optimizing our solution. As well, if our work on quil_output.py is complete, we will work on other issues within the repo. Having the experience that we will gain throughout the previous months while working on Cirq will be very beneficial at this stage.

April 30, 2020:

By the end of April, we should be ready for the release of the final product. The first part of the month will be spent heavily testing the end program and fine tuning our changes. We will push our final product on April 30th including any changes to the documentation that needs to be made.

2. Specifications and Analysis

2.1 Proposed Design

For our proposed design, we are modeling it off of a comment that was made about the implementation of outputting to QASM format. The general structure is the following:

- Each gate has a private function that returns itself as it would be written in QUIL syntax.
- There is a parser for the Cirq circuit that will iterate through each gate and call the function that should return QUIL syntax.

The parser should live in it's own file with other necessary methods. Each gate's translation function should live in that gates class. We are currently in the stage of identifying what is supported by QUIL so that we know what will translate from Cirq and we can adjust for what currently will not translate.

The standards required for our project are that all tests in the Cirq repository pass and that all the code that we create has full code coverage with tests. We also signed a contributor agreement license which means that all the code we submit to the Cirq repository is open-source and that Cirq has permission to use and redistribute your contributions as part of the project.

Non-functional Requirements	Functional Requirements
Performance- Our algorithm should be able to work efficiently with larger circuits.	Correct translation- Our implementation must correctly translate Cirq circuits. They must be usable by programs or machines that accept QUIL
Supports all gates- We should be able to handle any gates that Cirq could use in a circuit and if not gracefully report an error in translation	Be used for virtual experiments- Quantum Computing is often used for simulation of experiments, so our implementation will need to be able to withstand any sort of experimentation that is conducted.

2.2 Development Process

Our team plans to work with an Agile mindset. Each week we will give each team member the tasks they need to complete for the week and we will communicate through a meeting or group message if we can not find a meeting time for that week. Section 1.6 outlines the deliverables we will be working towards throughout the semester and our tasks each week will be assigned to lead us towards each month's goal.

2.3 Design Plan

Our design plan is to model the architecture of our code off of the code already present within the Cirq code base. This will provide easier understanding to the current people working on Cirq as well as new people joining the project. The largest constraint within our project is to implement as much cross conversion functionality as possible between Cirq and QUIL. We will have to come up with a large number of tests covering a variety of quantum circuit types to be converted. The most important use-case of our project is that we consistently output quil specification that can be regenerated in Cirq and can be implemented wherever the Cirq repository could be used. Because Cirq is meant for near-term quantum computing, we intend to ensure that the algorithm we choose works well for this technology.

Markdown file can be viewed here.

3. Statement of Work

3.1 Previous Work And Literature

Cirq contributors have already done a decent amount of work on a similar task of converting between Cirq circuits and QASM spec. We will be modeling our work closely after the QASM conversion code. Obviously the output will be different but we will structure the code similarly.

Quil is "an abstract machine architecture for classical/quantum computations---including compilation---along with a quantum instruction language called Quil for explicitly writing these computations. With this formalism, we discuss concrete implementations of the machine and non-trivial algorithms targeting them. The introduction of this machine dovetails with ongoing development of quantum computing technology, and makes possible portable descriptions of recent classical/quantum algorithms."[6].

3.2 Technology Considerations

Since Cirq is built with Python, the only possible option to complete our project involves building our solution with Python. All work towards deliverables will be in Python using the current Cirq features.

3.3 Task: Export as QUIL

Our project can be split into the following tasks:

- Learning Quantum Computing and Python
- Complete initial issues to familiarize ourselves with Cirq
- Study the QASM output, and determine how we can model QUIL closely to this sort of export function
- Assign official roles to everyone
- Implement file quil_output.py
- Test and edit the output file
- Optimize work up to this point
- Release final result for implementation into Cirq

3.4 Possible Risks And Risk Management

Skill Training Time: Since our team is unfamiliar with quantum computing concepts and some individuals are unfamiliar with Python, the time before being able to fully jump into development on the main deliverable may be significant. A way to reduce this risk may be to implement checkpoints for all members to have viewed a certain number of lectures by.

Software Updates: Any updates to the Cirq platform or any third party software being used by our team may trigger setbacks that could halt development. This is an accepted risk with any software project.

Scheduling Conflicts: This project spans across two semesters with six college seniors attempting to finish their degrees. Thus, varying schedules and course loads could have a significant impact on the project, especially around midterms and final exams. The best way to reduce this risk is to ensure all team members are in constant communication regarding their capability to contribute to the project at any one time.

3.5 Project Proposed Milestones and Evaluation Criteria

Our key milestones are as follows:

- Complete Quantum Computing and Python learning objectives; evaluation in an as needed basis.
- Complete initial issues to familiarize ourselves with Cirq; initial issues have been closed.
- Study qasm_output.py and determine how QUIL can be modeled similarly; pick the gates we need to include in our design

- Complete official role assignment; each team member has an assigned role.
- Complete initial implementation of quil_output.py; new circuit export function has been translated into Cirq but still requires testing and debugging.
- Complete testing and revision of the file; the project is fully functional.
- Complete optimization; code has been cleaned, commented, and performance is optimized for implementation in Cirq.
- Release final result for implementation into Cirq; the project is fully functional and running within the Cirq framework.

3.6 Project Tracking Procedures

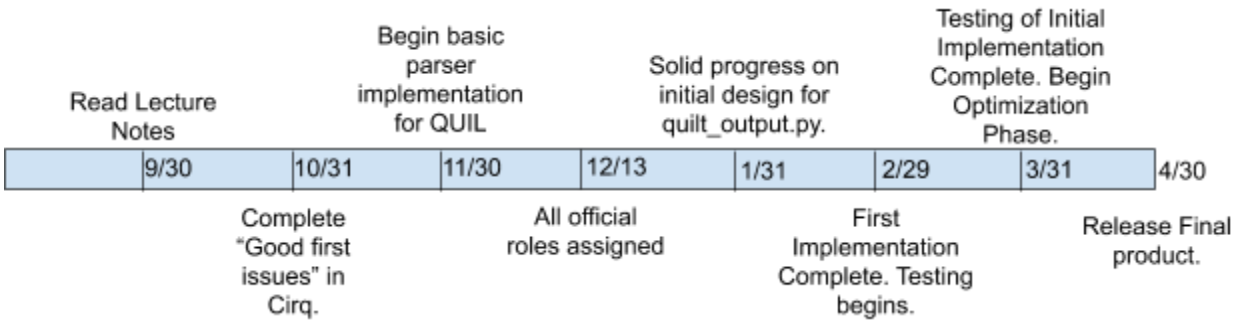
Our group is using GitHub to track deliverables and assign issues to team members. Github will also allow team members to view the progress of other members and ease the collaboration process.

3.7 Expected Results and Validation

The desired outcome of the project is to implement an output function that translates circuits in Cirq to QUIL. The output should model the current possible QASM output, but containing information relevant to QUIL. We can validate this result with a set of tests with predetermined results.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline



To start the semester this year we began by researching. None of us have any background in the subject material required for our project - quantum mechanics and quantum computing. By September 30 we all read the lecture notes provided to us by our sponsor Victory and watched the accompanying videos. By the end of October we will all have finished our assigned first issues on GitHub in order to become accustomed to the Cirq framework and understand how to use it and develop within it. By the end of

November we will all have studied circuits in Cirq, QASM, and how QUIL should be exported. At this point we will decide upon the best way to implement the parser that will allow for exporting to happen. By December 13th, the end of the semester, we will have all official project roles assigned specifically pertaining to implementing the parser, and so that each member can begin their specific work over winter break. By the end of January we will should have a good base of quil_output.py complete, and it should be almost ready to be tested. By the end of February we will have this initial implementation of quil_output.py will be finished. We will then begin thorough testing of the function to verify its proper operation. By the end of March we should have all of the testing finished and have a large variety of tests that verify proper implementation of the QUIL parser. These will include functional tests, as well as speed tests and stress tests. At this point we will begin optimization while holding the function to the same functional tests developed in the previous deliverable. The goal of this phase will be to increase performance and potentially pass more speed and stress tests that were not feasible in the first implementation. Optimization will be finished by the end of April and the final product will be delivered.

4.2 Feasibility Assessment

Feasibly, this project will result in the successful python implementation within Cirq's framework of the translation of a circuit to QUIL. One challenge with this will be learning the material necessary. Nobody on the team has prior experience with quantum computing or quantum mechanics and therefore we will all need to do a lot of research. The second challenge will be working together on a project with no clear division of work. The entire project is based off a single GitHub issue created by our sponsor: implementing quil_out. The issue has not been subdivided into smaller issues and we do not know enough from the limited research we've had time for at this point to determine subissues and divide the work.

4.3 Personnel Effort Requirements

Read Lecture Notes: 8/21 - 9/30	Team members are required to read/watch accompanying videos for 5 lectures per week. This would translate to about an hour a day Monday-Friday.
Complete "Good First Issues" in Cirq: 10/1 - 10/31	Varying effort is required depending on the issue assigned and the familiarity with the style of Cirq's development cycle. Team members should at least be expected to spend a few hours a week until they have completed their issues.
Begin the design implementation of QUIL: 11/1 - 11/30	All team members required to study the QASM files, and bring one idea as to how we can write the basic QUIL parser. In order to ensure how our design will be best for Cirq, members should continue to look into the commonalities between Cirq and QUIL. Should spend 1 hour a day

	writing code for this basic parser
Assign official roles and delimit tasks: 12/1 - 12/13	Since this is before finals not much effort will be required. We will meet as a team and assign roles once we've determined the decomposition algorithm we're going to use. Will be an hour total.
Work on completing quil_output.py: 12/14 - 2/29	During this time, we will expect team members to push code at least once a week and we will meet every Sunday to verify everyone's code is working together properly. Each member should spend 1-2 hours each day working on creating the parser
Testing Phase: 3/1 - 3/31	Everyone will be required to write at least two tests a week so that we can generate a large number of quality tests.
Optimization Phase: 4/1-4/30	Effort requirements will be similar to the first implementation phase. Members will be required to push code once a week and we will meet to verify optimizations still pass all the tests we designed.

4.4 Other Resource Requirements

All team members need access to a personal linux machine or linux vm in order to do development on the project.

4.5 Financial Requirements

There are no financial requirements for this project.

5. Testing and Implementation

5.1 Interface Specifications

Our project will not involve any user interfaces. Rather, we will add functionality to the backend that processes user designed circuits. Since Cirq is a quantum computer programming language there is no

real user interface. It's an API that allows users to write python files that describe the circuit they would like to experiment with.

5.2 Hardware and software

There will be no hardware directly involved in our project. This makes sense since Cirq is used to simulate hardware. There's at least one real quantum computing chip already programmed into Cirq for simulation. It is Google's Bristlecone chip.

Software wise, we will be adding two files, one for testing and one for the actual program. The file for our issue is going to be quil_output.py and the file to test our issue will be quil_output_test.py.

5.3 Functional Testing

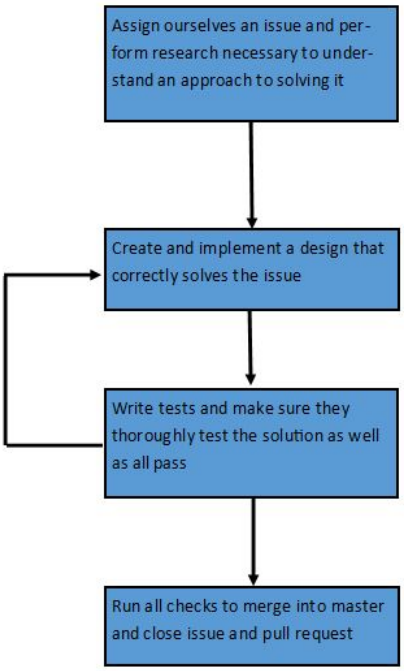
Each file in cirq has a test file in its folder that shares the name with "_test" appended to the end. This will test the full functionality of the file with the goal of getting 100% coverage besides where there are coverage exceptions. In order to run these tests we will use the pytest library.

5.4 Non-Functional Testing

In order to test for performance we will time the simulation of circuits. There will not be any testing in relation to security. We will run format and lint checks to make sure our code follows the Cirq standard.

5.5 Process

We haven't been able to test our approach through the main issue yet. We have been able to test our approach through introductory issues, though.



5.6 Results

Since our group has only been working on smaller tickets, no testing in relation to our main issue has been done. When it comes to testing for our introductory tickets, a lot of experienced failures have been due to improper setup of the Cirq development environment, improper syntax due to inexperience with python, and lack of knowledge when it comes to the structure of Cirq and how different parts of it are used. For successes, there aren't any metrics we can judge our current ticket solutions against besides if they all pass or fail.

The same process of testing will be used throughout the duration of our project. The contents of the test file may change depending on the code change for the issue.

6. Closing Material

6.1 Conclusion

Thus far, our team has been working hard to gain a fundamental understanding of quantum computing. Our sponsor, Victory, gave us lectures to read and watch. As stated prior, our primary goals/requirements are to implement issue #2386 in the Cirq repository (https://github.com/quantumlib/Cirq), study quantum computing and to complete smaller issues along the way. Our plan of action is spread out over two semesters, the first semester will be spent completing the smaller issues and learning more about quantum computing. Second semester, we will work together to complete our primary requirement, https://github.com/quantumlib/Cirq. Since our team has limited knowledge of quantum computing, this plan will work very well because it will allow us to gain ground work before attempting a difficult issue.

6.2 References

https://github.com/quantumlib/Cirq

[1] Ambainis, Andris, et al. "What Can We Do with a Quantum Computer?" *Institute for Advanced Study*, www.ias.edu/ideas/2014/ambainis-quantum-computing.

[2] LaRose, R. (2019). Review of the Cirq Quantum Software Framework. [online] Quantumcomputingreport.com. Available at: https://quantumcomputingreport.com/scorecards/review-of-the-cirq-quantum-software-framework/ [Accessed 8 Oct. 2019].

[3] Mottonen, Mikko, and Juha J. Vartiainen. *Decompositions of General Quantum Gates*. 2005, arxiv.org/pdf/quant-ph/0504100.pdf.

[4] Slepoy, Alexander. "Quantum gate decomposition algorithms." (2006).

[5] Vartiainen, Juha J. et al. "Efficient decomposition of quantum gates." Physical review letters 92 17 (2004): 177902.

[6] Smith, Robert S., et al. "A Practical Quantum Instruction Set Architecture." *ArXiv.org*, 17 Feb. 2017, https://arxiv.org/abs/1608.03355.

6.3 Appendices

Lecture Notes:

https://www.scottaaronson.com/blog/?p=3943

Lecture Videos:

https://www.youtube.com/playlist?list=PLm3J0oaFux3YL5qLskC6xQ24JpMwOAeJz

http://www.cs.cmu.edu/~odonnell/quantum15/