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 3-qubit unitary gate breakdown (main requirement)
- 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

Cird is a programming language for quantum computing. As of right now, the gates in the repository are 3-bit unitary gates. Our job is to create and algorithm to decompose these gates into a 2-qubit unitary gate and a 1-qubit unitary gate. This would allow ControlledGates made from 2-qubit KnownMatrixGates to automatically run on the simulator which causes the simulator to run faster and act more efficiently than it does currently. It will also cause the cost of production to be cheaper because everything in Cirq will be created from many of the same component.

Upon researching decomposition algorithms, it is important that we create a higher number of smaller *n*-qubit gates that are uniformly controlled The reason this is important is because the program will run much faster.

During this project, our group hopes to complete the implementation of the decomposition algorithm and make it as efficient as possible. We also hope to gain more knowledge of Quantum Computing and the benefits of using Cirq.

1.3 Requirements

1. Implement issue #451 in the Cirq repository (https://github.com/quantumlib/Cirq):

    For this requirement, we need to decompose 3-qubit unitary gates into 2-qubit unitary gates and a 1-bit unitary gate. This issue will not be assigned to us until the next semester and it will be split into three requirements.
    - Second half of first semester we must create the design of the decomposition algorithm we propose.
    - It is required that we spend the first half of second semester to implement the proposed algorithm in python that decomposes 3-qubit gates into a 2-qubit gate and a 1-qubit gate.
    - The second half of the semester will require heavy testing and making improvements to the algorithm we implemented

For this requirement, 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].
- Quantum computing is often used to search large amounts of data, so we need to ensure our gate composition algorithm can withstand [1].
- Must only use 1-qubit and 2-qubit gates.
- Currently have basic knowledge of Quantum Computing.

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 gate decomposition and optimization algorithms. We need to research these extensively so that we know which is the best choice for our design. During this month we will complete Weekly Logs 2 and 3.

November 30, 2019:

By the end of November, the decomposition algorithm we plan to implement will be chosen. We will choose this based off of the research we've completed and studying the repository more thoroughly. We will then finish the initial design for an efficient algorithm for the 3-qubit decomposition. Weekly Logs 4 and 5 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 the algorithm done by this point. The initial algorithm 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 algorithm 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. We will use the knowledge we gained from researching optimization algorithms first semester to choose the best possible way to do so. We will need to test our updates along the way and change any documentation as needed.

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 algorithm 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

So far, we have identified areas of the code that have similar algorithms implemented. For two qubit decomposition, Cirq has implemented the KAK decomposition because it works better for Cirq. Cirq is for near-term quantum computers which KAK works great with. Our plan is to take a similar method and apply it to three gate decomposition. There are several papers on quantum gate decomposition which we have been looking into. So far, we have worked on identifying the key features that will be required in creating the three gate decomposition. We have also identified the files that we will need to create. These include a “three_qubit_decomposition.py” file and an accompanying “three_qubit_decomposition_test.py” file. We have created these files in a common repository for all members to begin work on and pull from. 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.	Process Data- One of the main requirements for our project is that the decomposition of the gates can process large circuits to process a large amount of data
Scalability- Right now, the current algorithm works well with Cirq, but it doesn't support large circuits. Our algorithm should be able to be scalable, so that the developers for Cirq don't need to keep implementing a new algorithm as their circuits get larger.	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 constraints within our project are to find the "best" algorithm for three gate decomposition. This could require the testing multiple algorithms and comparing the results of each. The most important use-case of our project is that the algorithm is fast 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.

3. Statement of Work

3.1 Previous Work And Literature

There has already been a lot of work regarding the decomposition of 3-qubit unitary gates. Our objective is to find the ideal algorithm to implement within the Cirq framework. Identifying a new algorithm for decomposing 3-qubit gates is not within the bounds of this project. Thus one of our immediate tasks is to

search through papers such as [3, 4, 5] and pick an algorithm that best aligns with our goals for this project. Once an algorithm is selected, we will then implement the algorithm into our own work.

According to [2], as of July 2018 there were already "four major gate-model quantum software platforms: Forest by Rigetti, QISKit by IBM, ProjectQ by ETH Zurich, and the Quantum Developer Kit by Microsoft" before Google announced Cirq, which our current project involves contributing to. Cirq may not offer the same level of performance as some of the other options but it is constantly being updated, and is fairly easy to work with. According to initial research, none of the competing software is able to decompose a 3-qubit unitary gate. Thus our project will provide a unique feature to Cirq for users to implement that other software competitors have yet to provide.

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 Decomposition

Our project can be split into the following tasks:

- Learning Quantum Computing and Python
- Complete initial issues to familiarize ourselves with Cirq
- Research and choose a decomposition algorithm
- Assign official roles to everyone
- Complete initial implementation of algorithm
- Test and edit algorithm
- 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.
- Choose a decomposition algorithm; debate over which algorithm to use had ceased.
- Complete official role assignment; each team member has an assigned role.
- Complete initial implementation of algorithm; algorithm has been translated into Cirq but still requires testing and debugging.
- Complete testing and revision of the algorithm; 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 algorithm in python that decomposes 3-qubit gates into a 2-qubit gate and a 1-qubit gate. 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, specifically quantum gate decomposition. 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 researched decomposition methods for 3 qubit gates. At this point we will decide upon the decomposition method that we will begin implementation on. By December 13th - the end of the semester - we will have all official project roles assigned specifically pertaining to implementing the initial algorithm we chose for decomposing 3 qubit gates in Cirq. By the end of January we will have a rough outline and potentially some working pieces of the initial algorithm implemented. By the end of February we will have this initial algorithm finished. We will then begin thorough testing of the algorithm 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 decomposition algorithm. These will include functional tests, as well as speed tests and stress tests. At this point we will begin optimization while holding the algorithm 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 decomposition of a 3 qubit quantum gate into 1 and 2 qubit gates. 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 the decomposition of a 3 qubit quantum gate into 1 and 2 qubit gates. 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

Include a detailed estimate in the form of a table accompanied by a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be based on the projected effort required to perform the task correctly and not just "X" hours per week for the number of weeks that the task is active

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.
Research and choose decomposition algorithm: 11/1 - 11/30	All team members required to find and read around 10 articles on gate decomposition and present to the team 2 plausible decomposition algorithms from these articles.
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.
Work on first implementation of decomposition algorithm: 12/14 - 2/29	During this time, we will expect team members to push code at least once a week and we will meet weekly to verify everyone's code is working together properly.
Testing Phase: 3/1 - 3/31	Everyone will be required to write at least one test 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.

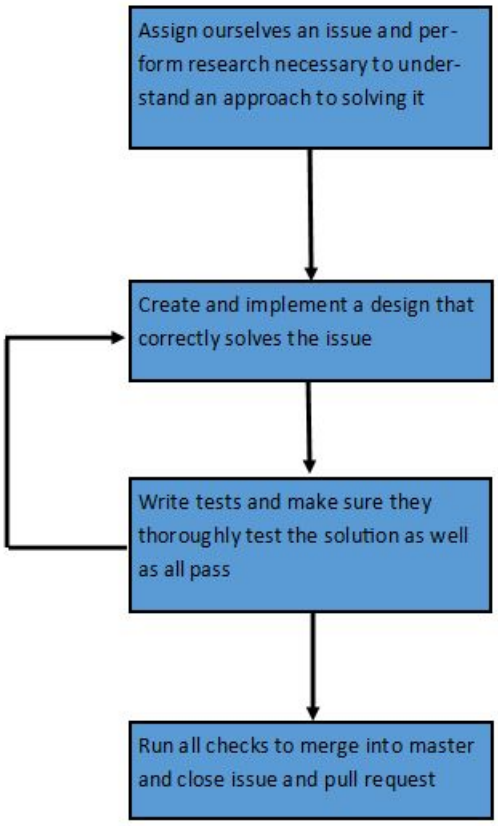
Software wise, we will be adding four files: three_qubit_decompositions.py, three_qubit_decompositions_test.py, two_qubit_decompositions.py, and two_qubit_decompositions_test.py. The two test files will be used to gain full coverage and thoroughly test each of our implementations. The actual decomposition files will be used to contain our implementation of the requested functionality.

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 #451 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.

6.3 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/