Compiling OpenQASM on a Neutral Atom Quantum Computer

Thesis Proposal

Author: Xavier Spronken (i6225376)

x.spronken@student.maastrichtuniversity.nl

Supervisor: Claire Blackman

Total Word Count:1998
Text:1813, Captions:165, Mathematical expressions: 20

Academic Year 2022-23

Contents

1 Introduction

1.1 Quantum Computers

While quantum computers can be used for analog computation or simulating quantum systems, it is also possible to make them perform quantum information processing (QIP). The carriers of information in QIP are qubits. These differ from classical qubits in that they make use of quantum superposition. Thus a qubit's state (either $|1\rangle$ or $|0\rangle$) can only be known precisely upon measurement. Before this, the qubit is said to be in a superposition of both states, where the probability of measuring either one of those states is $\mathrm{P}(|1\rangle) + \mathrm{P}(|0\rangle) = 1$. Mathematicaly a qubit is represented as $|a\rangle = \alpha|1\rangle + \beta|0\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix}$, where $\alpha^2 + \beta^2 = P(|1\rangle) + P(|0\rangle) = 1$

A method to view the state of a qubit before measuring is the Bloch sphere. In this representation, the $|0\rangle$ state and $|1\rangle$ state are respectively set to the north and south poles of the sphere. In addition to the probability of getting either state upon measurement, the Bloch sphere shows the phase of the qubit thanks to the latitudinal direction of the arrow in the sphere.

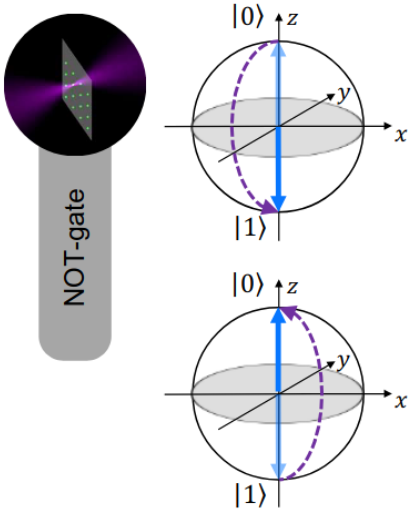

Figure 1: Bloch sphere representation of the effect of a not (or Pauli X) gate[1]

Before qubit measurement, one can change the state of a qubit using quantum logic gates, these are the quantum computing equivalents of logic gates that make up classical electronic circuits. When applied to a qubit, a quantum gate will set it to a specific superposition of states. This corresponds to a specific rotation of the arrow on the Bloch sphere. Since a gate corresponds to a rotation a does not place the qubit in a specific state, the resulting superposition depends both on the gate and the state of the qubit before the gate was applied. Some well-known quantum gates are the Pauli X Y and Z gates, each corresponding to a rotation of 180 degrees around the x, y and z axes of the Bloch sphere. Gates are mathematically represented as matrices that can be applied to the qubit state vector. The Pauli X gate for example would be $\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$.

Quantum gates are applied to single or multiple qubits at the same time, in general, these are control gates, which will only affect the target qubit if the control qubit is already in state $|1\rangle$. While there are multiple different types of gates, it is possible to create a universal set. Gates in this set can be combined to create any other gate. The set of arbitrary rotation gates $R_x(\theta), R_y(\theta), R_z(\theta)$, combined with a phase shift gate $P(\varphi)$ and a control-X gate are such a set.

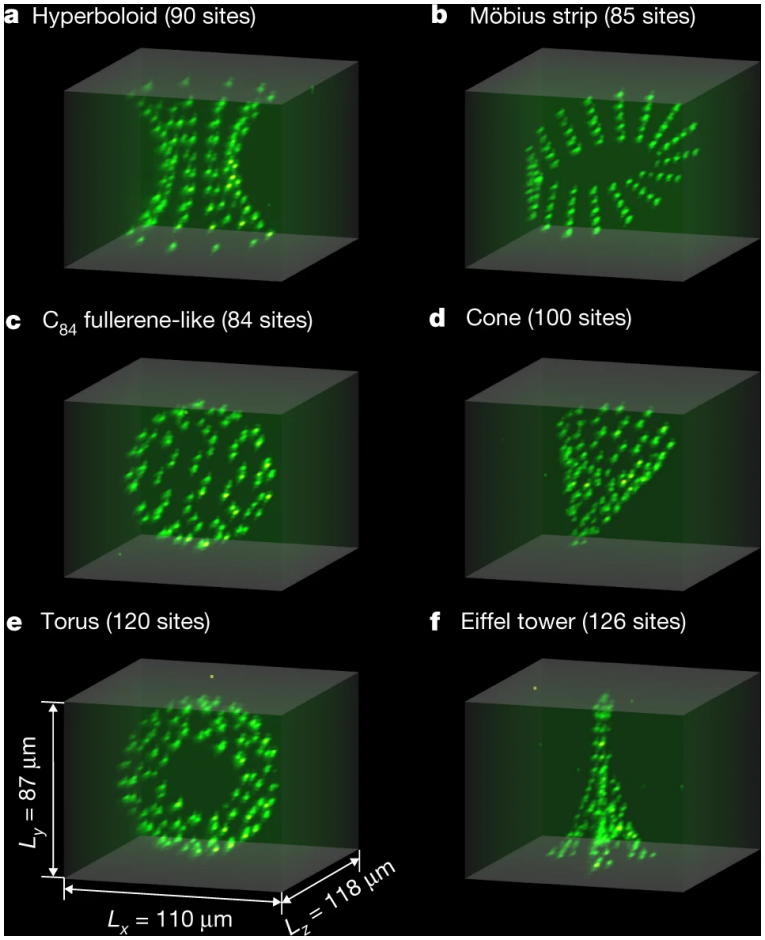

Figure 2: Different qubit configurations in a 3D register

Finally, these qubits are stored in a register. Depending on the type of quantum computer the registry can be fixed, such as for superconducting qubits, where all the qubits are pre-engraved onto a chip and used as needed. Other quantum computers such as the Neutral atom quantum computer built by Pasqal, have a dynamic registry, where a chosen number of qubits can be placed, at the user's will, into a 2D or 3D grid. This extends the ability of the device to be able to perform simulations of quantum systems and analog computations that cannot be represented in a traditional quantum circuit.

1.2 QASM

OpenQASM (Open Quantum Assembly Language) is a programming language for quantum circuits that allows the user to create a register of qubits and comes with a preselection of gates, as well as the universal set of arbitrary rotation, phase shift and CX gate. OpenQASM has become the standard for programming quantum circuits, this is partly due to it being machine-independent, meaning it can be compiled on any quantum computer. Along with the preset gates, custom user-created gates can be defined as well and saved into the circuit saved to the circuit before compilation and execution. OpenQASM 3 also incorporates timing statements as well as classical computing interactions within a circuit, allowing the user to precisely define when gates or measurements need to be applied, but also allowing the quantum circuit to perform classical operations and interact with a regular processor during the execution of the circuit.

1.3 Pasqal's Neutral Atom Quantum Computer

1.3.1 Register

This device uses optical beams to create multiple μm^3 sized traps, due to their size, these traps can only contain a single atom. The optical beams are then pointed at a vacuum chamber containing atomic vapor and used to single out and reposition atoms into a grid.

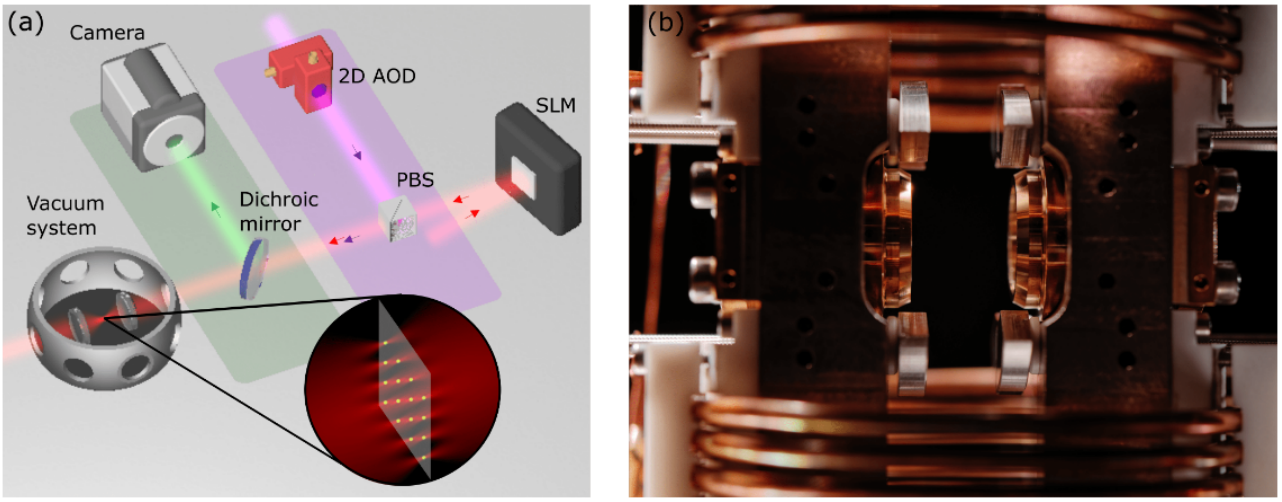

Figure 3: (a) "OMain hardware of a quantum processor. The trapping laser light (in red) is shaped by the spatial light modulator (SLM) to produce multiple microtraps at the focal plane of the lens (see inset). The tweezers (in purple), rearrange the atoms into the register, and are controlled by a 2D acousto-optic laser beam deflector (AOD) and super- imposed on the main trapping beam with a polarizing beam-splitter (PBS). The fluorescence light (in green) emitted by the atoms is split from the trapping laser light by a dichroic mirror and collected onto a camera. (b) Neutral-atom quantum co-processor heart containing the register."

1.3.2 Gates

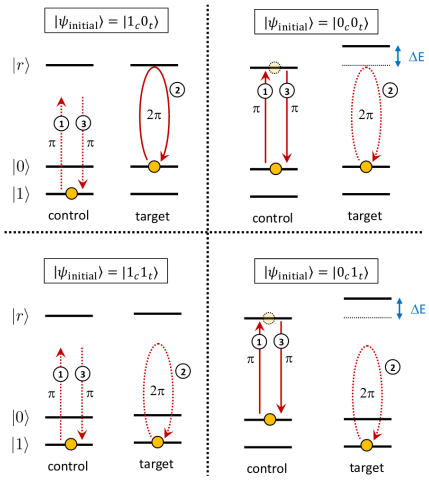

Figure 4: Sequence of optical pulses needed for a Control Z gate

Gates are applied using optical pulses from two different channels: the Raman and Rydberg channels, which can target a single qubit (local) or all of them (global). These pulses are used to attain two different excitation states from the ground state $|g\rangle \equiv |0\rangle$. The Raman pulses are used in single qubit gates to access the hyperfine state $|h\rangle \equiv |1\rangle$. Multi-qubit gates on the other hand require the use of quantum coherence, this is achieved using a phenomenon called the rydberg blockade: If an atom is excited to the Rydberg state $|r\rangle$(using the Rydberg channel of the quantum device), any atom situated within a certain radius will be blocked from achieving the Rydberg state. This interaction is used, for example in the implementation of a CZ gate.

1.3.3 Pulser

Pasqal has published a python library named Pulser that simulates their quantum device. It can create a registry with any kind of pattern as well as the required optical pulses. The pulses have a configurable waveform, amplitude Ω, detuning δ phase shifting φ and the duration τ of the pulse. Manipulating these parameters, it is possible to recreate any single-gate rotation on the Bloch sphere with angles :

$$(\Omega\tau\cos\phi, \Omega\tau\sin\phi, \delta\tau)$$

2 Project Objective

The objective of this thesis is to compile a logical OpenQASM circuit into a physical circuit on Pasqal's neutral atom quantum computer. This is achievable in four steps, the first is to parse the QASM code to retrieve the number of qubits used and which gates are applied to them. Once the code has been parsed, the qubits are mapped into a register that satisfies the constraints imposed by the multi-qubit gates. Following the creation of the register the gates can be translated into adequate optical pulses. Finally, the compiled quantum circuit can be tested through simulation, or on a real quantum device if available.

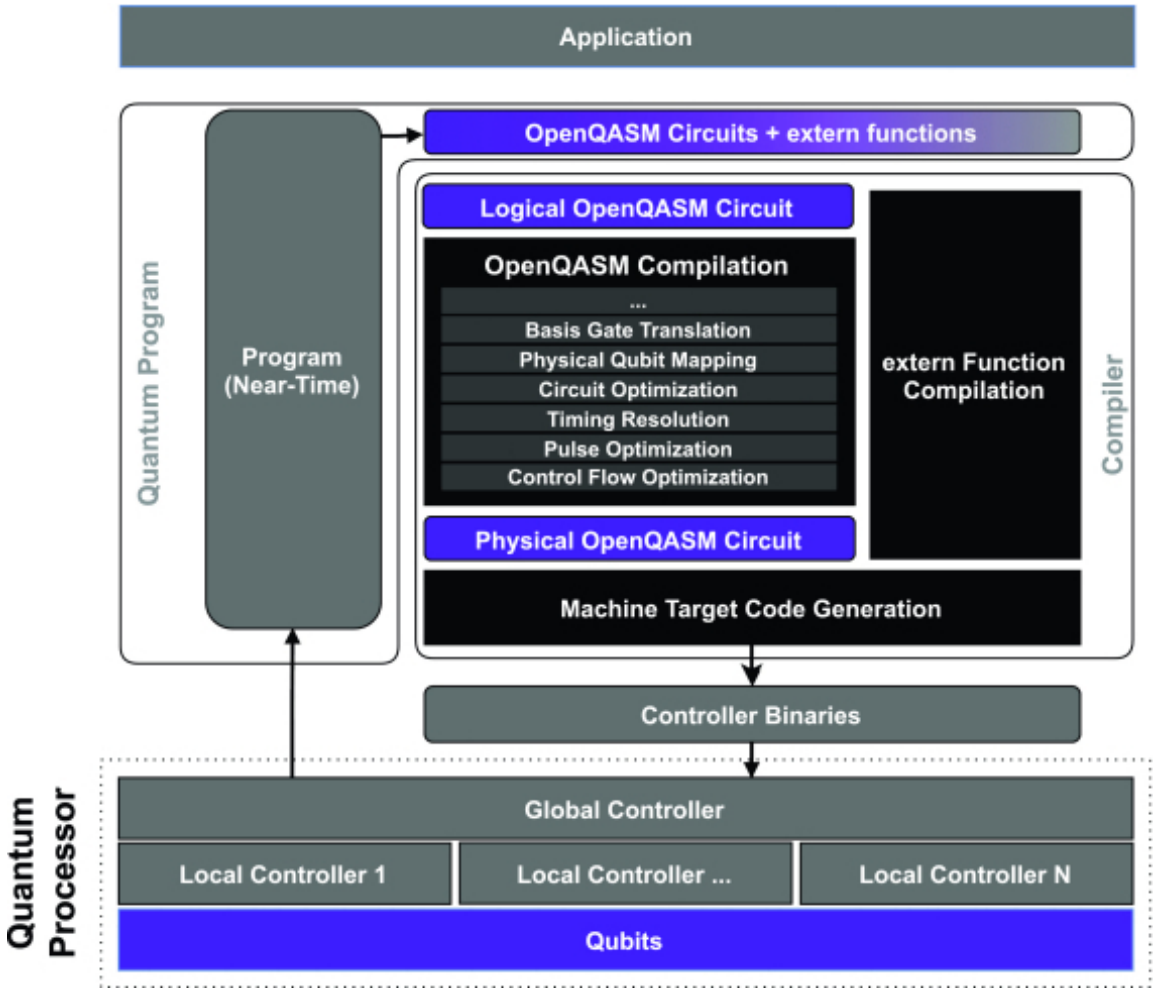

Figure 5: Flowchart detailing compilation steps for OpenQASM on a generalised quantum computer. Note that the optimization steps in the OpenQASM compilation are not in the scope of this project.

3 Methodology

3.1 Tools

The compiler will be written using the Python language. This choice was made due to the popularity of the language in the scientific community. Thanks to this there are many useful and well-documented toolbox libraries available to help simplify the project, three of these tools will be of particular importance. Firstly OpenQASM3 provides a Python library with parsing methods which will be used to retrieve all the required information from the user-input code. The NetworkX library will provide the optimization tools necessary to generate an adequate array of qubits to fill the register, based on the constraints imposed by the different multi-qubit gates in the input code. And finally, the Pulser Library, maintained by Pasqal, will allow for the simulation of their Quantum device. This library will be used to simulate the quantum computer the compiler will be written for. It allows the creation of different qubit registers and optical pulses and simulates the outcome of a quantum circuit on Pasqal's computer. If the simulations are successful, there may also be a possibility to test the compiler on Pasqal's device rather than simply using the simulations.

Some restrictions need to be taken into account regarding circuit simulation, notably that the hardware used to run the code will be a personal laptop. This will limit the complexity of quantum circuits that can be tested to about 5-10 qubits as anything greater would take large amounts of time to simulate completion of this project, it can be quite slow when it comes to computation

3.2 Approach

To maximize the chances of producing a functional compiler by the end of the project, a specific approach has to be used. Instead of trying to make each step (Parsing, Mapping, Translating, Simulating), a full compilation pipeline will be created for increasingly complex QASM algorithms. This means that the project will start by creating a simple parser, optimizer, translator and simulator which will work with one or two qubits and a small number of select single-qubit gates. As the project progresses, the pipeline will be modified to handle more complex algorithms, which involve more qubits and multi-qubit gates. Once the compiler can handle multi-qubit gates, the work will shift to translating as many gates as possible into their corresponding optical pulses and making sure the compiler can still work with these new gates.

3.3 Ethical considerations

The software tools used to produce this project are all free and published under open-source licenses, and the different resources used will be cited adequately. The code for the compiler will be stored in a GitHub repository and made public under an open-source license once it has acquired enough substance to be usable or useful to other researchers. Thus no further ethical considerations are thought to be required regarding the methodological approaches of this project.

3.4 Time planning

As mentioned above in the approach, time planning will be structured towards producing a limited but functional piece of software if the project is held up or problems occur. Thus the

project will be divided into three steps. First, the basic infrastructure will be created for simple algorithms that only involve single-qubit gates. Since these gates do not impose any constraints on how qubits should be stored in the registry, optimization will be handled in the next step. This second step involves adapting the compiler to be able to handle multi-qubit gates and the constraints that accompany them. Work on parsing should not be necessary if it was correctly written in step 1. Once the multi-qubit compiler is achieved, focus can be brought to translating all the remaining OpenQASM gates to optical pulses. At the end of each of these steps, the compiler will be tested using a known quantum algorithm and the results will be formatted and integrated into the final thesis paper. The timeline of this project is summarized in the Gantt chart below.

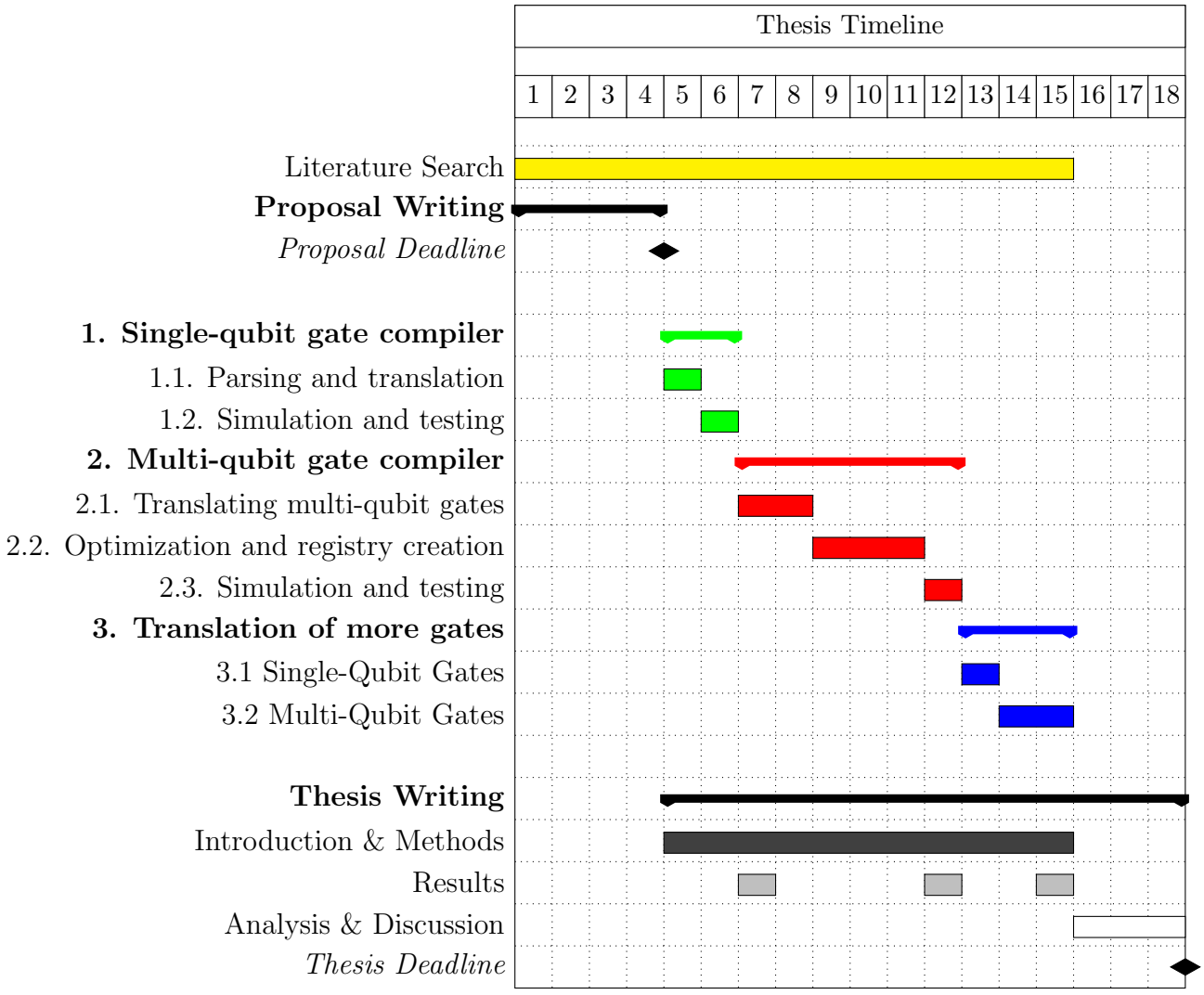

Figure 6: Gantt chart of the thesis timeline

4 Project Impact

4.1 Impact

Considering all the different approaches to implementing quantum computing, OpenQASM's position as a hardware-agnostic language may set it up to be a universal layer that all higher-

level quantum software languages could translate to and work on any quantum computer. This would allow quantum algorithms, such as Grover's algorithm, to work on any quantum computer, as long as it was written in OpenQASM. However, for this to be the case OpenQASM compilers need to exist for all the different types of quantum computers. This isn't the case yet for neutral atom quantum computers, or at least there is no open-source version available to the public. Thus this project aims to be the first open-source OpenQASM compiler for neutral atom quantum computers, more specifically Pasqal's device.

4.2 Ethical considerations

The main ethical concern for this project is that, while the project would produce an open-source solution to compiling OpenQASM, the main beneficiary of this code would be Pasqal, a for-profit start-up. However, they seem to be committed to open-sourcing most of their code as is the case with Pulser. Thus a compiler such as this one would help develop Pasqal's quantum technology further, while also keeping a majority of the knowledge open to the public and competition.

References

[1] Loïc Henriet, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond, and Christophe Jurczak. Quantum computing with neutral atoms. *Quantum*, 4:327, September 2020.