Getting started coding quantum computers with QASM
and Qiskit
Initial research report
Dylan Miracle
ICS 698-02
Spring 2021
Mar 23, 2021
Dr. Jigang Liu

Getting started coding quantum computers with QASM and Qiskit

Dylan Miracle
Department of Computer Science
Metropolitan State University
St. Paul, Minnesota, USA
dylan.miracle@my.metrostate.edu

March 23, 2021

Contents

1 Introduction

In the early days of digital computing a programmer needed a deep understanding of electrical engineering to program computers. The programming was low level involving digital logic gates and binary code. Abstractions were slowly built in to allow programmers to manipulate abstract data types so that now you can program a computer in a human readable language that uses high level concepts such as object inheritance and lambda calculus. Quantum computing is in its early phase and will likely progress to human human readable abstractions quickly. Developers who want to learn to program quantum computers will be served well to learn a programming framework to experiment with as opposed to taking a PhD in physics before learning the basics of quantum computing. It is not necessary to understand how a computer stores a string to build a web page, we should expect the same kinds of abstractions from quantum computing in the coming decade.

Frameworks such as qiskit and others already give a programmer access to many tools for interpreting and experimenting with quantum computers. This work aims to show an interested developer how to get started programming and interpreting quantum results. After reading this paper, for example, a developer would be able to create a webpage to display quantum results because they could understand how the result data is returned. Without needing to know the ins-and-outs of quantum computing they would be able to manipulate result data with a classical computer. This is a valuable tool because quantum computers will always be used with classical computers.

Quantum computing has a rich theoretical background but practical quantum computers have only recently become available to researchers and business users. These computers are still rare and operated by a small number of hardware vendors. The dominant paradigm for programing quantum computers is the gate model of quantum computing that defines a sequence of operations on a set of qubits. In this paper we introduce a fundamental quantum circuit and demonstrate its implementation using an intermediate representation, QASM, and a higher level quantum computing framework Qiskit.

In this paper we will explore how quantum computing will fit into a standard computing workflow.

This work gives an overview, from mathematical development of a simple quantum circuit to implementation with a high level quantum framework. Readers will get a taste of the complete quantum pipeline. This is valuable because much work focuses on one area of the ecosystem instead of walking those new to quantum mechanics through the complete process of how to implement some quantum code. For example, there are many papers detailing the use of a particular language, or that go into depth on specific quantum algorithms. Even though we focus on a specific framework the model of understanding quantum development proposed by this paper will be useful to anyone who wants to delve further into quantum computing.

2 Background

A basic discussion of quantum computing will involve developing two fundamental concepts in quantum computing, namely superposition and entanglement. Specifically we will use the example of the Bell states. These are the most basic states that exhibit the main features of quantum computing, namely superposition and entanglement of qubits. The development of this basic quantum circuit will start by exploring qubits and some basic quantum operations.

2.1 Qubits

Qubits are represented as two element column vectors. The elements are complex numbers. Since we measure qubits to be either 0 or 1 we will talk about them as being in the computational basis. This is not the only basis we can measure qubits. A single qubit can be visualized as a vector inside a unit sphere. This sphere is known as the Bloch sphere.
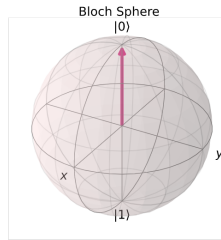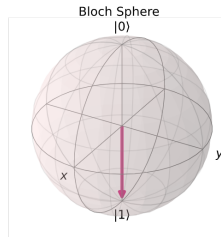

Figure 1: Bloch vector for qubit $|0\rangle$.
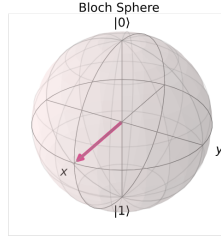

Figure 2: Bloch vector for qubit $|1\rangle$.

Figure 3: Bloch vector for qubit $(|0\rangle + |1\rangle)/\sqrt{2}$.

2.2 Bell state

We achieve a bell state by using a CNOT gate to combine a $|0\rangle$ with a superposition like $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. This creates what is known as an entangled state. This combination results in a state represented as

$$\frac{|00\rangle + |11\rangle}{2}$$

With the property that any individual measurement of on of the qubits is random, but the measurement of the second qubit is entangled with the measurement of the first. So a measurement of 1 for qubit 0, means we will get a 1 for qubit 1 as well. In the next section we will simulate such a state and visualize it with a histogram.

3 Preliminary findings

3.1 Quantum and classical workflows

3.2 Program bell state in QASM

QASM is the basic assembly level intermediate representation for quantum code. It allows us to create two types of registries: quantum registries and classical registries. The quantum registries hold the information in qubits while the classical registries hold information in bits. Higher level languages use python libraries to represent the different parts of the quantum program.

Pure quantum computing involves superposition and entanglement. We will use a hybrid approach, preparing our problem on a classical computer, then submitting a job to a quantum computer, finally gathering and displaying our results using a classical computer.

In this program we will implement the Bell state by preparing two quantum registers and two classical registers. The quantum registers will then be put into a Bell state by sending one qubit through a Hadamard gate to create a superposition, then both qubits through a CNOT gate to entangle them. Finally a measurement will be performed on each qubit independently.

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[2];
h q[0];
cx q[0],q[1];

measure q[0] -> c[0];
measure q[1] -> c[1];
```

Once we have defined the circuit using the intermediate representation of QASM, we can import the circuit into a higher level language that will be used to submit the circuit to a simulator or to quantum hardware. In the following example we will use Qiskit to load our circuit and submit it to the builtin simulator.

3.3 Load circuit into Qiskit and submit job

The Qiskit program that follows will expand our workflow. We use standard methods of python programming familiar to anyone used to using the language to load our libraries, import the circuit, then submit the circuit to the simulator backend. The only change we would need to make to submit this program to actual quantum hardware is the change the backend to point at a quantum processor.

```python
import qiskit
from qiskit.visualization import import plot_histogram

qc = qiskit.QuantumCircuit.from_qasm_file("bell.qasm")
qc.h(0)

backend = qiskit.Aer.get_backend('qasm_simulator')
result = qiskit.execute([qc], backend, shots=100)
count = result.result().get_counts()
plot_histogram(count)
```

3.4 Summary of findings

By running our simulation 100 times we can gather statistics on the results of the measurements. Here is a histogram of measurements from the simulation shown
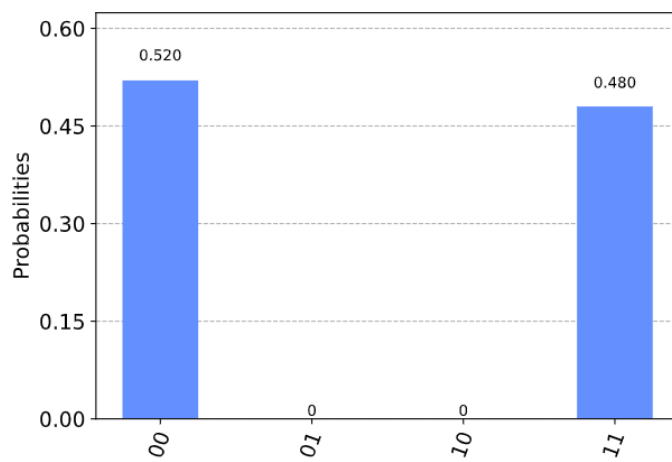
Figure 4: Simulations of measurements of Bell state.

While QASM and Qiskit are specific to IBM, the model they use of inter-mediate representation and higher level language is present with other vendors (see Rigetti) and a good way to understand and control a quantum computer. An intermediate language gives us a method to write out specific gates used in most quantum computing algorithms, while the higher level framework lets us use standard computing.

4 Charts or tables

4.1 Circuit diagram

This diagram shows the standard circuit representation for gate model quantum computing. This is a simple circuit that creates a Bell state with two starting qubits, then shows the measurements of those qubits being written into classical bits.
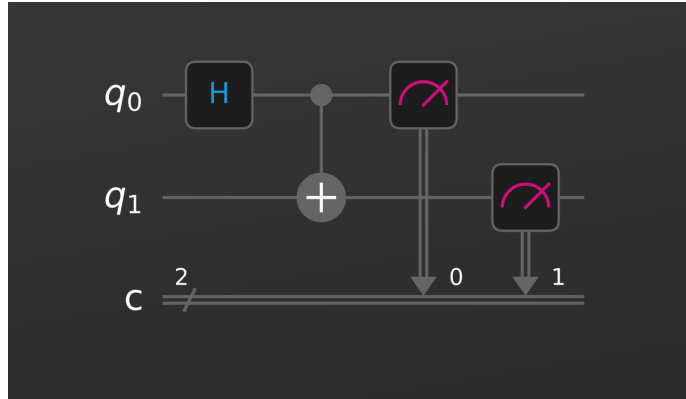
Figure 5: A quantum circuit that will create a Bell state with two qubits.

4.2   Quantum development flowchart

Developing quantum algorithms is found to follow a flow from the quantum mechanics, to a quantum circuit, to intermediate representations, to higher level frameworks. This flow illustrates the findings of this paper and will serve as an outline for our exploration of quantum computing.

Figure 6: The quantum development flow.

4.3 Quantum workflow flowchart

Developing quantum algorithms is found to follow a flow from the quantum mechanics, to a quantum circuit, to intermediate representations, to higher level frameworks. This flow illustrates the findings of this paper and will serve as an outline for our exploration of quantum computing.
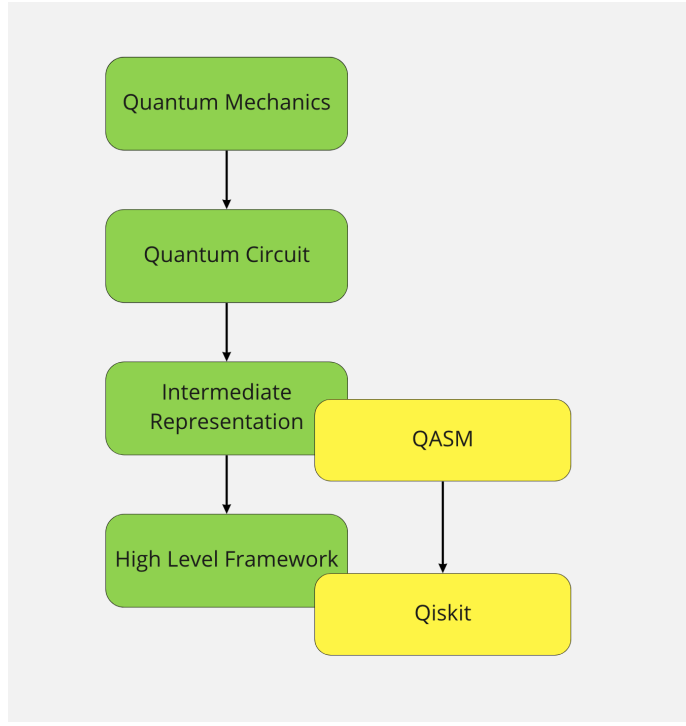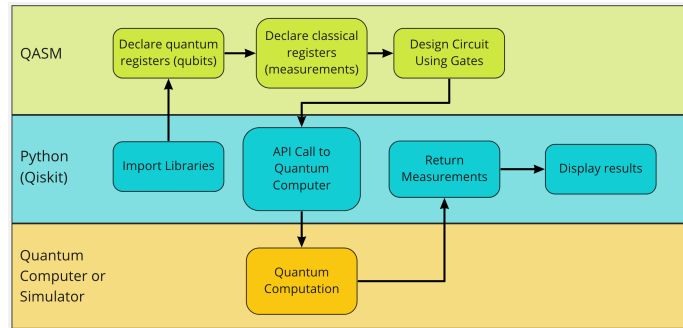

Figure 7: The quantum computing workflow.

5   Preliminary conclusion

We should understand quantum computation as a workflow between the classical computer and the quantum computer. In this example we will create a simple quantum program and follow its execution using standard open source tooling available for programming IBM quantum computers. At the highest level we can understand that the classical computer is used to define the computation we will do, then an API is used to communicate the quantum job to the chosen backend, in our case a simulator, though a similar method will be used to submit the job to actual quantum hardware.

Developers can write code for quantum computers by learning frameworks in already existing languages. Several frameworks studied in the literature review have python frameworks. Python is often used because it is broadly adopted by the scientific community and creates a lower barrier to entry than some other languages might. For example Microsoft's Q# language uses a combination of F# and C# that is less familiar to people not used to writing code in both of those languages. A good framework gives developers access to the tools needed to run quantum gates, but also standard computing tools that are helpful in interpreting results. The model used by qiskit to run jobs and return results means that programmers can predict what kind of data they will be dealing with before they fully understand the algorithms being used on the quantum side. As more software has a quantum components having well defined result data and job definitions means that software developers can create code for interpreting results and dealing with the output of quantum computers without becoming quantum computing experts.

Quantum computing will not stand on its own, but will always be employed in tandem with classical computing. This is as simple as using standard computers to create flow control or as advanced as using the fully featured APIs needed to connect to actual quantum hardware. As discussed the data returned by quantum algorithms will be interpreted and represented by classical computers. It is possible to currently develop software that will use quantum results in the future because well designed frameworks like qiskit have a standard workflow and results data structure.

Much work has been done to develop quantum computing frameworks. These frameworks are racing to increase functionality and capabilities for advanced users. This paper should give an overview of some of the fundamentals of quantum computing for an advanced undergraduate or software developer without a background in quantum mechanics. The audience for most quantum computing work is physics PhDs making it inaccessible to a typical software developer. This has a chilling effect on a bourgeoning industry. When we think about the great contributions to our modern information age we often end up thinking about hackers in a garage just as much as engineers in a clean room.

6 Next Steps

The next part of this study would be to extend our workflow understanding to encompass what

7 References