Quantum Circuit Implementation: Represented as a Jupyter Notebook

Mohamed Ali[*]
University of Washington, Seattle WA
Physics 575

(Dated: 12/11/202)

This paper we will start of with small introduction of quantum computing, quantum circuits and some popular gates used within quantum mechanics. I have made a IBMQ account with https://quantum-computing.ibm.com and have used some remote servers to display some of these gates functionality and also created some plots to show the output of my code. I have done this by creating a jupyter notebook, I will go through what I have done and explain lines of my code and expected results.

Usage: For physics 575's project 2 requirements. .

I. INTRODUCTION TO QUANTUM COMPUTING

a quantum computer is made of qubits while a classical computer is mad of bits. Let's talk about what a qubit is, to understand this we have to talk about what a bit is. A bit is a binary value between 0 and 1. Sometimes 0 is referred to as a low signal and the 1 is referred to as a high signal. A qubit is a little different, it isn't as binary. A qubit can be a 0 or a 1, none of these two or even both! How would this be possible you ask? Well in Quantum Mechanics there are two principles that make this a possibility; they are superposition and entanglement and we will introduce these here briefly. Quantum superposition stipulates that some state can be in multiple arrangements and that the probability of each one of these configurations describes the actual state of a system. This may seem odd however lets use an example; suppose we have a mug that may contain tea, coffee, water, etc. Only when we look into the mug, or make a measurement, do we have a definitive understanding of that state of the mug in this case. The same thing applies to qubits. Initially the system may be some combination of 0 and 1, but once we make a measurement the value collapses to some definitive o or 1. Quantum entanglement, however, is a scenario where two or more qubits are so close or interact in such a way that the description of their state can cannot be done individually. So when we describe the state of a qubit as we have done in our description of what quantum superposition is, this cannot be done for each qubit in this group separately and in fact they are described as one state. So a change in one qubit results in the instantaneous change in the other qubits within that system. Now that we have an understanding of the difference between a qubit and a classical bit, let's look at what that means for the difference that causes in classical computing vs. quantum computing.

A classical computer can represent $n * bits * 2$ while a quantum computer can represent $2^(n * bits)$ what a difference in representation! I doubt that quantum computers will completely replace classical computers in the near future however, I believe that quantum computers will be used to solve problems classical computers simply cannot. Some great examples would be optimization problems, or encryption problems (breaking, or securing encryption) that simply take up an unrealistic amount of time or is vulnerable due to simplicity classically is perfect for a quantum computer. Other types of problems like logistics problems, where a plan has many possible paths or modelling problems are also a great example of the possible utilization of quantum computers. A hybrid computational world between classical and quantum computers will, however, be an immanent future.

II. BUILD QUANTUM CIRCUITS USING QISKIT

Before we talk about what I did in my code lets introduce what logic gates are. To manipulate qubits, we must use logic gates to either flip them to 0 or 1, another way of saying this is to use gates and then make a measurement to collapse that qubit into a classical bit. The most common gates are AND, OR, NOT, XOR and NAND gates. Let's discuss some common quantum gates, the most famous perhaps, is the Hadamard Gate. This gate takes in a single qubit and outputs the 50/50 probability that the qubit will collapse to a 0 or a 1. Another important gate is the Pauli-x Gate. This gate takes in a single qubit and performs the exact same operation as a classical NOT gate. This means that the qubit is flipped so if it is a 0 is then it collapses into a 1 and vice versa. There is also a Controlled NOT Gate which takes in 2 qubits and only flips the second qubit from a 0 to a 1 or from a 1 to a 0 if the first qubit is a 1 otherwise it leaves it unchanged. Lastly let us talk about measurement. Measurement is an operation done on qubits at the end of a circuit to get the final values of the qubit after an operation has been conducted. The difference between a measurement and a gate is a measurement is non reversable.

To set up my project I created a folder and initialized

[*] http://www.Second.institution.edu/~Charlie.Author

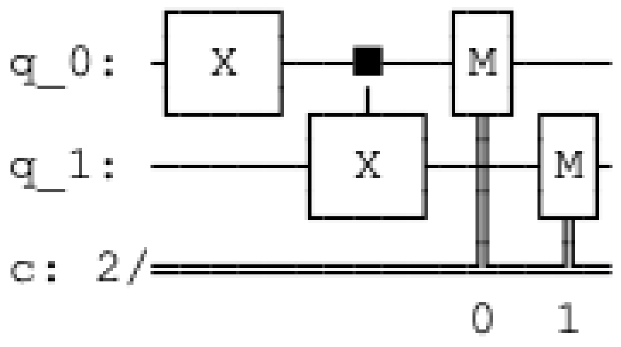

FIG. 1. ASCII Drawing

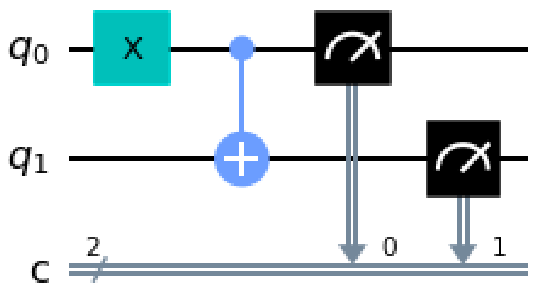

FIG. 2. Matplotlib Drawing

a jupyter notebook and here we go. Initially I imported qiskit and name is some variable, I used qiskit for simplicity sake. We are using qiskit here because qiskit is an open source platform that you can integrate with python to generate some of the quantum gates we will be using today. Qiskit allows for us to connect with remote quantum processors to do some of these things since we are using a classical computer. In my case I also had to make an account with IBM's IBMQ service which allows you to connect to IBM's remote quantum processors. I also import the use of matplotlib to generate some visuals down the line. After making my account at IBMQ, each user has a unique token, I took that token and saved that to a .txt file and placed that file in the same folder as my jupyter notebook folder.

We are finally set up and ready to go, we have our matplotlib and qiskit modules imported and we are ready to set up our first quantum circuit. We are going to define our circuit in python and then allow our remote quantum computer push our qubits through gates, make measurements and return bits. This is another example why an interaction between classical computers and quantum computers is the most feasible in the near future. We establish our quantum circuit and define the amount of qubits and bits we want, I picked two for each, we apply a NOT gate to the 0 bit. We then apply a CNOT gate which entangles the two qubits, we know that this gate flips the second qubit's value if the first qubit is a one. When we apply these two gates we measure our results, the input in the .measure function is two parameters; we want to input our qubits and classical bits, as you can see I have inputted two arrays and then have python's matplotlib feature draw the gates and output. The classical computer here builds the circuit, the quantum server will run the quantum circuit through gates and perform measurements and the results are communicated back to my classical computer. the first image is an ASCII representation of the circuit, the second image is a cleaner representation of the first image using matplotlib.

Next we connect code here to our IBMQ account by importing IBMQ, then using that imported module to open

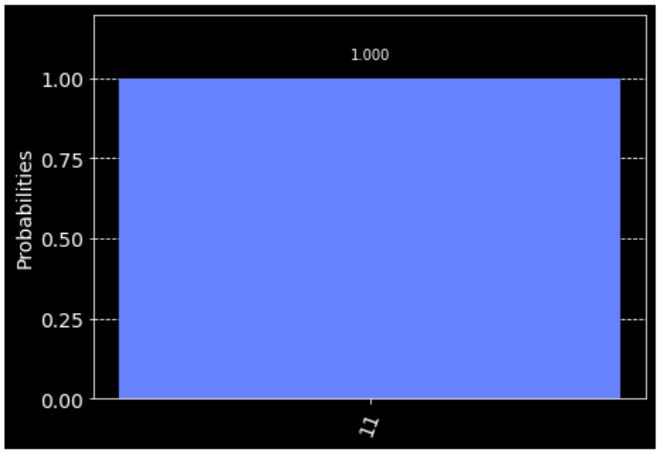

FIG. 3. Histogram

up our token.txt module and load it into our account. In my IBMQ account there are many remote back-ends that I can use, I pick one an set a provider variable to one of the available back-ends [2]. I think run a for loop to simply display the available back-ends, their jobs and the amount of qubits available to them. The outputted message simply keeps track of the traffic every back-end is having at the moment and when you can utilize a specific back-end.

After this I'll pick an available back-end and then use the job monitor module in qiskit.tools.monitor to make sure my job is run on that specific back-end when I'm next in the queue and to take a specific amount of shots, the amount of iterations you want to pass through gates to get an accurate representation, of my jobs so I can develop those shots into visuals after. To build these visualizations I import the plot histogram module in qiskit.visualization, for added effect I also import the style module to represent these histograms in dark mode. These portion of my work took some time now that we have defined a job, I had to wait till a back-end was available to take my job, our job should be over quick and not take to much time from people doing more intensive work. My histogram had a heavy leaning towards 11, but why wasn't it 100 percent? This is a result of noise, perhaps a change in temperature has caused this slight deviance towards other probabilities [1]. After this example I do the same thing as above but now I use a Hadamard gate which puts these gates in superposition. I do the same thing now; make a measurement and draw the ASCII representation. I know, like last time, define the amount of shots I want and plot the histogram.

Lastly I use a local simulator to do this work. I use a module Aer from qiskit and set my back-end simulator to "qasm simulator". Just to test this is working, I print the back-ends available and then define my results and plot my histogram just like I have done before.

III. REFERENCES

[1]
[2] Mark Fingerhuth, Tomáš Babej, and Peter Wittek. Open source software in quantum computing, Dec 2018.