Quantum Computing

Clodagh Walsh 19230737

Qubits can store a value of either 0, 1 or any value in between. They reside in a three-dimensional space referred to as the Bloch Sphere. Since the qubit can occupy any state between the basis states of 0 and 1 the state of the qubit at any instance must be accompanied by a probability. The computational basis states of the qubits are denoted by the vectors |0> and |1> ,with the former being the ground state of the qubit. This notation is referred to as Ket or Dirac notation which was developed by the physicist Paul Dirac in 1939. Since the qubit can occupy a state anywhere between 0 and 1 inclusive its state is thought of as a vector in a three-dimensional sphere.

This ability of qubits to occupy a state between the basis states is known as superposition. Exploiting this capability of qubits, a quantum computer can outperform a classical computer. Despite its benefits there is one serious flaw. Upon inspection, the qubit collapses out of superposition and hence the advantages of quantum computing are lost. By Heisenberg’s Uncertainty Principle the observation of a qubit alters its state. Programmers generally examine the state of their variables through the program, however this is not possible in quantum programming.

The property of entanglement alleviates this seemingly catastrophic situation. When a set of qubits are entangled, a change in any one qubit will prompt an immediate reaction in the other regardless of the distance between them. By measuring one of the qubits the properties of the other qubit can be determined without having to inspect it. This proves very useful in maintaining the superposition of the qubit.

Programming a quantum computer involves the manipulation of qubits. This is achieved through the use of quantum gates. Quantum gates take the input in a superposition apply the given operation and output a qubit with an updated superposition and therefore updated probability. These quantum gates are availed of to construct quantum circuits which in turn perform computations.

The Swap gate takes two qubits as input and alters their states. The quantum NOT gate has the same function as the classical NOT gate, it changes the state of the qubit. The quantum NOT gate is also referred to as the Pauli X gate, as the Not gate essentially rotates the qubit 180 degrees on the x-axis. Other rotation gates include the Pauli Y and Pauli Z gates. Where the qubit is rotated 180 degrees on the y-axis where the Pauli Y gate is applied and a rotation of 180 degrees is carried out on the qubit in the z-axis when the Pauli Z gate is implemented. The controlled Not (CNOT) gate is a two qubit version of the NOT gate composed of a control qubit and a target qubit. If the control qubit is set then a NOT operation is carried out on the target qubit. The CNOT gate produces entanglement a phenomenon crucial to the success of quantum programs. The Hadamard gate applies two rotations on the qubit. It first rotates the qubit 180 degrees on the x-axis, followed by a 90 degree rotation along the y-axis. This procedure produces the Hadamard matrix which places the qubit in superposition. Therefore this gate is frequently used .The Toffoli gate is an extension of the CNOT gate with three inputs. If both of its two control qubits are set then a NOT operation is performed on the third qubit. The Toffoli gate is essential as it can be used to generate all other logic gates.

It is important to note that the state of the qubit is found or measured at the very end of the program. When the state of the qubit is inspected it loses its superposition therefore inspection of the qubit is reserved until the end to obtain maximum benefit from quantum computing. When measured the result is stored in a classical register. This is possible as the result is either a 0 or 1, which is a traditional bit. However, there is a slight difference in that the result here comes with a probability attached as the state of the qubit can not be known for certain. The measurement gate accepts a qubit in a superposition of states and generates a classical bit as described above.

The IBM Q Experience is a cloud-based platform for designing and testing quantum programs. It is a visual editor where users build quantum circuits by selecting quantum gates represented graphically as blocks. However, access to the quantum processors is limited as they are in high demand. The user builds their quantum circuit in the Composer workspace. Completed circuits named Scores are then run on either on a simulator or an actual quantum processor. The number of qubits available to work with depends on the quantum processor selected.

Quantum programs can be written in Python with the aid of the QISKIT SDK for Quantum Computing. To set up the environment the programmer first imports the Quantum Computing library. Following that generate the required number of qubits and classical registers. There should be a classical register for each qubit. Set up a circuit in which to place necessary gates. Call quantum gates on chosen quantum register. Apply the measurement gate. As previously mentioned, the outcome will be stored in a classical register. Even though programming in a High Level Language the programmer relies entirely on the use of gates to execute their commands. Such details are abstracted from the programmer of a classical program.

QASM is the quantum assembly language. Programs written in Python and circuits designed on the IBM platform are translated into QASM and then run on a quantum processor. Not that it is more efficient to write a program using QASM than in Python.

Quantum supremacy is the term used to describe the ability of a quantum computer to carry out a task impossible to a classical computer. The term was originally proposed by physicist John Preskill in 2012. This concept has gained much interest and attention since. Quantum supremacy is defined as a super polynomial speedup. In reality very few algorithms produce super polynomial speed up. However, a quadratic speed up is achievable for search problems. Google in collaboration with NASA conducted tests providing experimental evidence of quantum supremacy. The program which randomly generated numbers ran on a 54-qubit quantum processor named Sycamore. It is estimated that it would take 10,000 years to simulate this program on a classical machine in comparison to the 3 minutes and 20 seconds it took for the quantum processor to execute the program. Despite this remarkable advancement in quantum technology, quantum supremacy could be disproved if there was an improvement in the classical algorithms. Though proving the existence of quantum supremacy is a challenge in its own right since it is difficult to simulate these quantum programs on classical machines. This bench marking is a necessary part of the process of demonstrating the arrival of quantum supremacy.

Quantum Computing is greatly impacting the field of Computer Science. It has put modern cryptography methods and cybersecurity defenses in peril. The development of quantum algorithms has prompted experts to review classical algorithms. Developments in this field contribute to advancements in other emerging fields in Computer Science namely Artificial Intelligence. Ultimately this technology is redefining storage capacities and processing time beyond what Computer Scientists once thought was possible.

References

Quantum Computing for Computer Scientists By Noson S. Yanofsky and Micro A. Mannucci Published by Cambridge University Press 2008

Practical Quantum Computing for Developers: Programming Quantum Rigs in the Cloud using Python, Quantum Assembly and IBM Q Experience By Vladimir Silva Published by Apress 2018

Quantum Computing and the Entanglement Frontier By John Preskill of Institute for Quantum Information and Matter California Institute of Technology Published 13 November 2012

Arute, F., Arya, K., Babbush, R. *et al.* Quantum supremacy using a programmable superconducting processor. *Nature* **574,**505–510 (2019) Published October 23

Information leaflet on Quantum Computing published by ICHEC (Irish Centre for High End Computing)