Survey: Quantum Computing Algorithms and Programming Model

Md Maruf Hossain
Dept. of Computer Science
University of North Carolina at Charlotte
Charlotte, USA
mhossa10@uncc.edu

Abstract

Quantum computing is now one of the most popular and growing sectors in modern science. The main reason behind this is the performance capability of the quantum computer. But quantum programming is a relatively young subject and it is not so straight forward in comparison to the others. On the other hand, there is no ideal *Quantum Computer* exists so far. So, it is very important to analyze a problem before solving it in the quantum machine. The main purpose of this survey is to discuss the existing *quantum computer* model and its programming language. I am also going to talk about some future challenges of the *quantum programming* language.

I. Introduction

Sometimes, it is very difficult to solve quantum physics problem in the traditional computer, the main reason is the large size of the problem matrix. That is why scientists made a lot of effort to build up quantum computers that can solve the large problem within the acceptable time complexity. Whenever we talk about any computing system, that means it has the ability to solve the problem at a cost of time. And when we talk about *Quantum Computer*, it comes in our mind that it must have the power to solve the classic problem. So, this is very important for *Quantum Computing* and for some other problems, it can give a superior result that the traditional computer system failed so long. One example can be *Shor's* [1] algorithm for the prime factorization of an integer. Where traditional computers require $O(2^n)$ time complexity to solve the factorization problem, *Shor's* algorithm can solve it in $O((log\ N)^3)$ time and $O(log\ N)$ space in *Quantum Computer*. Another important algorithm regarding the significant achievement of quantum computers is *Gall's* [2] *Triangle Finding* in an unweighted graph. It can give you a solution in a $O(n^{5/4})$ time where the traditional computers require $O(n^3)$ time. These algorithms are boosted up the significance of the *Quantum Computer*. Now, the challenging part is to implement these algorithms in the quantum computer. Initially, we mentioned that it is not so straight forward like other programming languages, the main reason for this is the maturity of the language. So, different problems require different insight to implement in the quantum machine. On the other hand, there is no ideal quantum computer exist right now. So, in this survey, I am going to discuss the little bit about the quantum hardware or the simulator and then the programming models that can help us to solve the different problems in the quantum machine. This is a class projects of *Quantum Computing* course and I got most of the information from the course materials, websites and from some research works [3].

II. Hardware Model

Quantum Computer is mainly based on the superposition and entanglement of quantum physics, but it hard to build an ideal quantum computer. But, nowadays IBM, Microsoft, and other companies are trying to construct different types of prototype and virtual machine for quantum computing. Hardware is very important for any kind of languages because based on the hardware you have to design the compiler. In this section, I am going to discuss the different type of virtual hardware model for quantum computing.

A. Virtual Hardware Model

Most of the virtual hardware models are almost same, so one can choose any programming model they want.

1) Quantum Circuit Model: The *Quantum Circuit* [4] is made using quantum gates like traditional logic gates. The most important facts of *Quantum Circuit* is that the quantum gates are reversible and unitary transformation over a complex vector space. The *measurements* quantum operation is always carried out at the very last step in a computation. *Quantum Circuit* model is one of the most popular virtual hardware models in quantum computing.

2) QRAM: Another important virtual hardware model called QRAM [5] and this model will give you the freedom of *unitary transformation* and *measurements*. In this model, quantum device is controlled by a traditional computer. The quantum device contains a large number of addressable quantum bits but the number is finite. It acts like a RAM chip, that is why it is call QRAM. The traditional controller device send instruction to the quantum device, either the *unitary transformations* or the *measurements*.

III. Quantum Programming Language

Every programming models try to assemble the instructions and process them, in quantum computing the assembling instructions are processed in *Quantum Computer*. Different compilers have a different parser that is why the coding structure varies based on the languages. But primarily, we can divide the quantum programming model into two different categories.

A. Imperative Quantum Programming Language

Initially, the quantum programming language was kind of pseudo-code structure and started by Knill [5]. Later the more complete version of imperative programming language presented by Omer [6], Sanders and Zuliani [7], and Bettelli et al. [8]. Here I am going to talk about some imperative programming language in the below.

1) QCL: According to the quantum programming history and wiki, *Quantum Computation Language*(QCL) is one of the first quantum programming languages. QCL is similar to the traditional C programming language and it supports the user-defined operators and functions. It supports the classic data type like int, real, complex, boolean, vector, matrix, tensor, etc. as well quantum data types like qureg, quvoid, quconst, quscratch, qucond, etc. It can give a user more freedom in programming.

2) Q#: Q sharp developed and maintained by Microsoft and initially released as a quantum development kit. It can create and use qubits for algorithms. As a result, it can capable to produce the key feature of quantum computing "entangle" and "superposition". It has more similarity to C#, that can help the developer who already familiar with the C#.

3) Quantum Pseudocode: In this language, the algorithm is presented by pseudo-code and compiled onto the QRAM. This language is mainly dedicated to the QRAM virtual programming model.

4) qGCL: Quantum Guarded Command Language (qGCL) is based on Guarded Command Language created by Edsger Dijkstra. This kind of language is guarded by the proposition, if the proposition is true then the statement is going to execute.

B. Functional Quantum Programming Language

Scientists are trying to build the functional programming language for quantum computing because it can give the user more freedom. There are lots of functional programming languages are proposed by lots of researchers. Here I am going to discuss some of them.

1) QFC and QPL:

2) QML: QML is Haskell like programming language who first support the classical and quantum control operators.

3) Quipper: Quipper is implemented as an embedded language using Haskell as the host language. That is why the programmer need to write code in Haskell.

IV. Conclusion

References

[1] Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. Ieee, 1994.

[2] François Le Gall. Improved quantum algorithm for triangle finding via combinatorial arguments. *arXiv preprint arXiv:1407.0085*, 2014.

[3] Peter Selinger. A brief survey of quantum programming languages. In *International Symposium on Functional and Logic Programming*, pages 1–6. Springer, 2004.
[4] A Chi-Chih Yao. Quantum circuit complexity. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 352–361. IEEE, 1993.
[5] Emmanuel Knill. Conventions for quantum pseudocode. Technical report, Citeseer, 1996.
[6] Bernhard Ömer. A procedural formalism for quantum computing. 1998.
[7] Jeff W Sanders and Paolo Zuliani. Quantum programming. In *International Conference on Mathematics of Program Construction*, pages 80–99. Springer, 2000.
[8] Stefano Bettelli, Tommaso Calarco, and Luciano Serafini. Toward an architecture for quantum programming. *The European Physical Journal D-Atomic, Molecular, Optical and Plasma Physics*, 25(2):181–200, 2003.