University of Texas at Dallas

Department of Physics

Quantum Information Research Report

Stephen Aranda

Supervisor: Dr. Bei Zeng

A report submitted in partial fulfilment of the requirements of
the University of Texas at Dallas for the Certificate of
Quantum Information and Bachelors of Science in Computer Science.
December 7, 2024

Declaration

I, Stephen Aranda, of the Department of Physics, University of Texas at Dallas, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made available more widely to members of UTD and public with interest in teaching, learning and research.

Stephen Aranda
December 7, 2024

Abstract

In this paper, a brief summary of the paper, Quantum Algorithms with Qualtran, will be given along with an analysis of key points in the paper. Another aspect of this paper will be a discussion of some of the problems that exist in computer science. The paper will then proceed with an argument of how quantum computing is necessary and how the problems discussed in this paper can be resolved with quantum computers. And, finally, the paper will end with a conclusion of the findings along with their implications.

Keywords: Quantum Algorithms, Qualtran, Quantum Computing

Acknowledgements

In this acknowledgement, I would like to thank Dr. Raymond Benge; astrophysicist teaching at TCCD at northeast campus that was my professor when taking Physics 1. His passion for physics sparked my excitement and motivation to learn more about physics. I'd also like to thank Dr. Michael Kolodrubetz; theoretical physicist teaching at UTD. When I gained an interest in learning more about Quantum Mechanics I took his class Quantum Mechanics for Engineers and Programmers. It was a great class, but it was also the hardest class I ever took. It was because of him and this class that I became motivated to get into quantum computing to try to eventually enter the industry to work on these machines.

Contents

Chapter 1

Introduction

Qualtran is a Python library that provides a series of tools and abstractions that are meant to help the engineer or physicist deal with the complexity of quantum algorithms and programs. In the first portion of this paper, a summary and analysis of the paper, Quantum Algorithms with Qualtran, will be given.

1.1 Summary: Quantum Algorithms with Qualtran

This paper can best be described as a detailed walk-through of the Qualtran library and its respective features that help deal with the complexity of quantum algorithms, programs, and subroutines.

1.2 Bloqs

Bloq is the first feature discussed. Its use is to encapsulate quantum operations, subroutines, and algorithms as composable data structures. Bloqs represent quantum algorithms in a hierarchy. This helps in making Bloqs modular and reduce computational overhead.The feature also supports correctness guarantees aligned with physical laws, such as no-cloning and no-deletion theorems.

1.3 Resource Estimation

Qualtran automatically tabulates gate and qubit counts for quantum algorithms. This is crucial for estimating execution costs on hardware.

1.4 Qualtran Data Types

Introduces customizable data types for quantum registers, such as integers and fixed-point numbers, with tunable precision to optimize resource usage.

1.5 Protocol for Analysis

The protocol for analysis allows gate counting, qubit counting, and simulations (both classical and tensor-based). Call graphs are used to represent the decomposition and modularity of algorithms.

1.6 Standard Library of Quantum Subroutines:

Includes advanced primitives like rotations, quantum variable rotations, unary iteration, lookup tables, and state preparation. There is also a highlighting of resource-efficient implementations for applications in chemistry, cryptography, and Hamiltonian simulation.

1.7 Block Encodings and Arithmetic:

Provides tools to encode complex matrices as unitary operators and perform arithmetic on them. It also supports applications in linear algebra, optimization, and Hamiltonian simulation.

1.8 Visualization

Automatically generates detailed diagrams and graphs to aid algorithm validation and communication.

Chapter 2

Analysis of Quantum Algorithms with Qualtran

In this chapter, an analysis of Quantum Algorithms with Qualtran will be provided:

2.1 Strengths of Qualtran

Usability: Since Python is used, Qualtran is more accessible to researchers who have experience with programming; because of the built-in abstractions provided by Python, it also produces a faster development cycle.

Reproducibility: Offers a structured and standardized way to analyze algorithms, which encourages collaboration among different researchers.

Efficiency: Automates otherwise manual and error-prone tasks, such as resource estimation and circuit decomposition.

Scalability: Supports analysis for teraquop-scale algorithms, a necessity as quantum systems grow in size.

2.2 Limitations

The framework assumes sequential execution in its default resource estimates, which may not always reflect real world parallelism.

Limited support for advanced quantum data types like strings or lists.

2.3 Potential Impact

Qualtran bridges a critical gap in the development of quantum algorithms by providing a unified platform for design, testing, and analysis. It could accelerate the transition of theoretical quantum computing research to practical, real-world applications.

2.4 Future Directions:

Expanding the standard library with community contributions.

Developing heuristics to automatically choose optimal implementations (e.g., for rotations).

Improving qubit allocation strategies to handle parallelism more accurately.

Chapter 3

Problems in Computer Science

Currently, I am an undergraduate that is majoring in computer science. Therefore, I do not have a research area at this moment. During this semester, I have made the personal decision to work towards a masters and, eventually, towards a P.H.D. This is due to my interest in quantum computers. Taking this into consideration, then, I would say, my research area is quantum computing. Specifically, in the areas of quantum algorithm development, quantum software development, and quantum error correction. Now, the requirements for this paper make the assumption that a student's research area is not quantum computers. Because of this, for the sake of argument, I will specify problems that are associated with classical computers in computer science rather than problems in quantum computers even though I stated above that I want to research quantum computers.

3.1 The CPU Size Problem

The CPU of a computer, as it was being developed during the second half of the 20th century and up to the present time, had a pattern of experiencing a boost in performance and the size of the CPU would be reduced. The issue is the CPU is reaching its physical limit in how small we can shrink the size of the chip. The reason for this is because once a CPU reaches the nanometers scale in terms of its size, it starts experiencing quantum mechanical phenomenon like quantum tunneling. At the present time, there is no solution in dealing with this issue, and as a result, the physical limit is about to be reached.

3.2 The CPU Power Wall

This particular issue is associated with the limit of how much the clock rate of a CPU can be increased. As the clock rate of a single CPU is increased, the amount of heat produced increases. This heat causes damage to the CPU itself. The amount of heat that a CPU can sustain is approximately 100W to 120W.

There are other issues, but for the sake of keeping this report at a reasonable length, these are the only problems that will be discussed in the report.

Chapter 4

Necessity of Quantum Computers

This chapter will discuss how quantum computing can resolve the problems discussed in chapter 3 and, as a caveat, why it is necessary.

4.1 Necessity

My argument as to why time, energy, money, and effort should be placed into the development of quanutum computers that meet the DiVicenzo criteria is not because of all the benefits that the machine offers. Even though these benefits are reason enough in my opinion. The reason why we should invest in the development of quantum computers is because it is necessary. We have to do this in order to continue advancement into next-generation cutting-edge computer technology.

Let's take the two problems from the previous chapter as an example. Currently, there is no equipment or classical computer components that exist to deal with quantum tunneling in a CPU that is a few nanometers in scale. A fully implemented quantum computer would easily solve this problem because the components of this machine are built to deal with quantum mechanical phenomenon.

The power wall issue is another problem that quantum computers solve. With the use of superposition and entanglement, you can have many computational tasks performed almost instantaneously. This solves the power wall problem because you don't have to worry about clock rate issues. You've also increased the speed at which computational tasks are done without having to increase the clock speed of the CPU, thereby eliminating the risk of the CPU being damaged by excess heat.

One could make the argument that multi-core processors along with parallel programming have resolved these two problems. In my opinion, it's a band-aid, not a solution. Multi-core processors, just like single core processors, will reach their physical limit and will produce excess heat that will damage them. Not only that, but there's added complexity with multi-core parallel programming. For those not familiar with parallel programming, the way it works is that you have to implement a mutex, which is essentially a lock. This lock is then acquired by a process or a thread which will execute it's own instructions in it's own core, then releases the lock so that another process or thread can execute their instructions. Complexity is a problem but the main problem is that, based on the above description, parallel programming isn't actually parallel, even though you have multiple cores. It's sequential because only the process or thread that has the lock can execute their instructions. Which means you're not getting a performance increase when using multi-core processors. You are, however, getting a performance increase and true parallel execution with superposition and entanglement.

Chapter 5

Conclusions

5.1 Conclusions

In this paper, there was a summarization and analysis of the paper Quantum Algorithms with Qualtran. There was a discussion on its strengths, its limitations, its impact on quantum computing, and future direction in how the library is developed.

The paper then proceeded to discuss two problems in computer science.It then proceeded to discuss not only why it is necessary to develop quantum computers, but how quantum computers solve the two problems that were discussed. This was the power wall problem and the size problem; both problems are CPU problems. Although there are other problems in computer science, these problems, along with the complexity of multi- core processors and parallel programming, were the only problems discussed to keep the report at a reasonable length.

The development of full-fledged quantum computers that meet the DiVicenzo criteria will be difficult, mainly because of what is the best way to deal with quantum noise. Hopefully, the argument presented in this paper about the necessity of the development of these machines if we want to continue to advance in next-generation computer technology was enough to convince the reader that we must continue on this path to develop these machines. This concludes my quantum information research white paper. Thank you for reading.