Resources Estimation for Quantum Computing Algorithms in Multiple Physical Platforms

Cesar Zaragoza Cortes

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2021

Reading Committee:

Program Authorized to Offer Degree:

Physics

Chapter 1

INTRODUCTION

*Write something

Limitations on current NISQ[1] technology...

1.1 The Purpose of This Thesis

This thesis...

1.2 Conventions and Notations

In this thesis...

[1]See, for example, John Preskill[1] for a recent discussion.

BIBLIOGRAPHY

[1] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, Aug 2018.

Appendix A

BERNSTEIN-VAZIRANI Q# IMPLEMENTATION

The following code presents a simple implementation of the Bernstein-Vazirani algorithm in the Q# programming language:

```
namespace Quantum.BernsteinVazirani {

    open Microsoft.Quantum.Arrays;
    open Microsoft.Quantum.Canon;
    open Microsoft.Quantum.Intrinsic;

    function ArrayToString<'T> (array : 'T[]) : String
    {
        mutable first = true;
        mutable itemsString = "[";
        for item in array
        {
            if (first)
            {
                set first = false;
                set itemsString = itemsString + $"{item}";
            }
            else
            {
                set itemsString = itemsString + $", {item}";
```

```
        }
    }

    set itemsString = itemsString + "]";
    return itemsString;
}

@EntryPoint()
operation BernsteinVazirani () : Unit {
    Message("Bernstein-Vazirani");
    let secret = [One, Zero, One, One, Zero];
    use (qubits, aux) = (Qubit[Length(secret)], Qubit()) {
        X(aux);
        H(aux);
        ApplyToEach(H, qubits);

        // Oracle.
        for index in 0 .. Length(qubits) - 1 {
            if (secret[index] == One){
                CNOT(qubits[index], aux);
            }
        }

        ApplyToEach(H, qubits);
        let results = ForEach(M, qubits);
        Message(ArrayToString<Result>(results));
        ResetAll(qubits);
```

```
                Reset(aux);
            }
        }
    }
```