Implementation of Square Root Function Using Quantum Circuits

Computer Science Junior Independent Work

Pranav Gokhale
Princeton University Class of 2015
Advised by Dr. Iasonas Petras

January 12, 2014

Abstract

The design of quantum circuits that implement elementary functions has foundational value for the field of quantum computing and is also important for the development of other quantum algorithms. Here, we present the first design for a quantum circuit that computes the square root of any number $v > 1$. We include error analysis and asymptotic estimates for complexity and resource requirements. Specifically, if a total error of at most E is desired, then the procedure requires order $O(\log_2 \log_2 \frac{6v^{1.5}}{E})$ iterations and a maximum of $O(\log_2 \frac{v^{1.5}}{E})$ qubits at any particular stage. A preliminary design for a circuit extension computing v^z for other fractional powers (besides $z = 1/2$) is also presented, as well as for the special case where z is a power of 2.

1 Introduction

Quantum computing is an emerging field that presents a fundamentally new model of computation. Although no scalable quantum computers have been built yet, it is important to study the theory of quantum computing to understand what the capabilities of a quantum computer will be. In certain instances, quantum algorithms can provide exponential speedups over classical algorithms. For example, a quantum computer can run Shor's algorithm [4] to factor an integer in polynomial time with the input size, whereas the best known classical algorithms require at least subexponential time. More recently, quantum algorithms have been proposed that can efficiently solve problems that suffer from the "curse-of-dimensionality" under certain assumptions. For instance, numerically solving a Poisson equation problem has complexity that is exponential with dimension for a classical computer but there is a quantum algorithm with cost linear in the problem's dimension [2].

As the field of quantum computing grows, it is becoming increasingly important to design quantum circuits that implement elementary functions. Apart from the intrinsic value in being able to perform the most fundamental operations, the implementations of more complicated quantum algorithms are often predicated on the existence of circuit modules for elementary functions. For example, a quantum circuit design for elementary arithmetic operations [6] is vital for the implementation of the numerous quantum algorithms

involving addition, multiplication, and exponentiation. Similarly, the quantum algorithm for various partial differential equation problems [2] requires circuits for reciprocals, sines, and cosines. Last but not least, there has also been focus on developing standardized quantum circuit modules from the IARPA (Intelligence Advanced Research Projects Agency) Quantum Computer Science Program [3], which seeks to catalog the precise circuit resource costs and error analyses of quantum algorithms.

2 Background

In this paper, we present the design for a quantum circuit that takes input $v > 1$ and outputs an approximation of its square root, $\sqrt{v}$. Square roots can be efficiently computed on a classical computer with the Babylonian method (a special case of Newton's method). The recurrence formula is

$$x_{i+1} = \frac{x_i + v/x_i}{2}$$

so that $\lim_{i\to\infty} x_i = \sqrt{v}$. However, this does not translate into a practical quantum circuit, because the v/x_i term involves a new division at each stage of the iteration. Because quantum circuits for division are generally avoided, the Babylonian method cannot be directly used for a quantum circuit for division. Similarly, approximating the square root function with a Taylor series does not work, because the error grows rapidly away from the expansion

point and the formula also requires unwieldy divisions. It is also not sufficient to merely input v into a classical computer, because if $|v\rangle$ is a superposition of more than one state, the classical computer would collapse it into just one of the states of the superposition. Thus, a proper quantum circuit for approximating the square root should be able to accept a superposition state and output the correct superposition state.

Another challenge for quantum circuits is that they must be built with reversible logical components so that the quantum circuit involves a bijective mapping from input to output. This means that the logical gates can only be represented by unitary matrices [1]. While there is a reversible square root circuit design for 8-bit input [7], the design is specific to the input size and cannot be generalized to input of any size. Moreover, rigorous error analysis has not been carried out for this reversible circuit. Another reversible square root circuit proposal [5] claims to be extendible to arbitrary input size, but specifics of the requisite internal modules have not been published and no error analysis has been completed.

The approach used here involves first approximating $1/v$ via an inversion circuit and then applying an inverse square root circuit to approximate $\sqrt{v}$. The Newton method iterations for approximating these terms require only division by two, multiplication, and addition, which are all manageable.

3 Inversion Circuit

We first compute $\hat{x}_{s_1}$, which approximates $1/v$. The relevant quantum circuit design and error analysis has been completed as a step in a quantum algorithm for solving the Poisson equation [2]. The Newton iteration is shown in Figure 1. The total error after s_1 iterations, when using b_1 bits of accuracy (bits after the binary point), is bounded as

$$|\hat{x}_{s_1} - 1/v| \leq 2^{-2^{s_1}} + s_1 2^{-b_1} \qquad [2, \text{Thm B.1}]$$

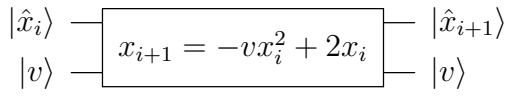

Figure 1: Circuit implementing each iteration step of the Newton method for the inversion stage

4 Inverse Square Root Circuit

4.1 Initial Approximation

Next, we compute $\hat{y}_{s_2}$, which approximates $1/\sqrt{\hat{x}_{s_1}}$. As an initial approximation for $1/\sqrt{\hat{x}_{s_1}}$, we take $\hat{y}_0 = y_0 = 2^{\lfloor (p-1)/2 \rfloor}$, where $2^{-p} \leq \hat{x}_{s_1} < 2^{-p+1}, p \in \mathbb{Z}$. The initial approximation circuit finds the most significant bit of $\hat{x}_{s_1}$ that is 1 (this bit is in the 2^{-p} position of $\hat{x}_{s_1}$) and sets a 1 in the $2^{\lfloor (p-1)/2 \rfloor}$ bit position of y_0. An ancilla qubit acts as a flag for whether a 1 has been set in

y_0. A small circuit for two-qubit input is demonstrated in Figure 2. When the ancilla qubit is $|1\rangle$, the circuit has no effect, $|0\rangle|1\rangle|r_1r_0\rangle \rightarrow |0\rangle|1\rangle|r_1r_0\rangle$. When the ancilla qubit is $|0\rangle$, the action of the circuit is $|0\rangle|0\rangle|r_1r_0\rangle \rightarrow |r_1 \text{ OR } r_0\rangle|r_1 \text{ OR } r_0\rangle|r_1r_0\rangle$. Thus, $|s_0\rangle$ is set to $|1\rangle$ if and only if the ancilla is 0 and r_1 or r_0 is 1.

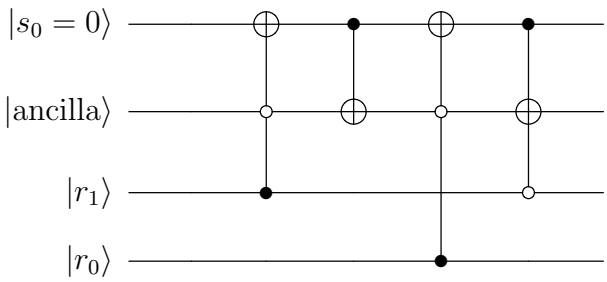

Figure 2: Small example of a circuit computing the initial approximation $\hat{y}_0 = 2^{\lfloor (p-1)/2 \rfloor}$ of Newton iteration for approximating $1/\sqrt{\hat{x}_{s_1}}$, where $2^{-p} \leq \hat{x}_{s_1} < 2^{-p+1}, p \in \mathbb{Z}$. In this example, $\hat{x}_{s_1}$ is the two qubit input $0.r_1r_0$ and $y_0 = s_0.0$

The circuit in Figure 2 is the building block for the general circuit in Figure 3 which is extendible to arbitrary input size. Because this circuit is composed of reversible logic gates, it can act on superpositions of inputs and produce superposition output states as well.

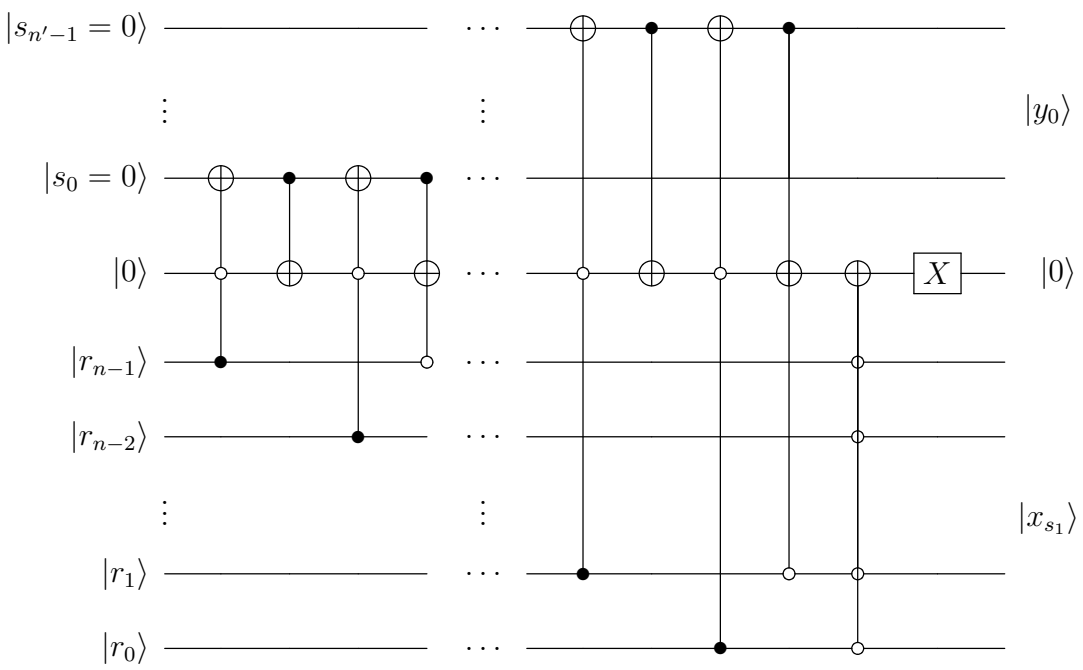

Figure 3: The quantum circuit computing the initial approximation $\hat{y}_0 = 2^{\lfloor (p-1)/2 \rfloor}$ of Newton iteration for approximating $1/\sqrt{\hat{x}_{s_1}}$, where $2^{-p} \leq \hat{x}_{s_1} < 2^{-p+1}$. Here, $\hat{x}_{s_1} = 0.r_{n-1}r_{n-2}...r_1r_0$ and $y_0 = s_{n'-1}s_{n'-2}...s_1s_0.0$. The length of y_0 is $n' = \lfloor (n-1)/2 \rfloor + 1 = \lfloor (n+1)/2 \rfloor$, where n is the length of x_{s_1}.

4.2 Newton's Method Iteration

Next, we apply Newton's method with the function $g(y) = 1/y^2 - \hat{x}_{s_1}$ (which has root $y = 1/\sqrt{\hat{x}_{s_1}}$ as desired). The recurrence is

$$\varphi(y_i) = y_{i+1} = y_i - \frac{g(y_i)}{g'(y_i)} = y_i - \frac{1/y_i^2 - \hat{x}_{s_1}}{-2/y_i^3} = \frac{3y_i - \hat{x}_{s_1} y_i^3}{2}.$$

Thus, we apply s_2 iterations of the circuit in Figure 4 to attain $\hat{y}_{s_2}$, an approximation for $1/\sqrt{\hat{x}_{s_1}}$. This recurrence requires quantum circuits for addition

and multiplication, which have been previously studied [6]. The recurrence also requires division by 2 which could, for example, be implemented using a form of right shifting in a quantum register extended by one qubit set to $|0\rangle$.

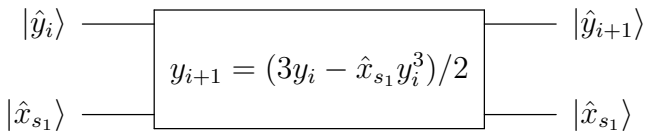

Figure 4: Circuit implementing each iteration step of the Newton method for the inverse square root stage.

4.3 Error Analysis

Lemma 1. *The only possible equilibrium values for the Newton's method recurrence are* $-1/\sqrt{\hat{x}_{s_1}}, 0, 1/\sqrt{\hat{x}_{s_1}}$.

Proof. The recurrence reaches an equilibrium when $y_{i+1} = \varphi(y_i) = y_i$. Thus,

$$\varphi(y_i) = \frac{3y_i - \hat{x}_{s_1} y_i^3}{2} = y_i.$$

This equation has solutions $y_i = -1/\sqrt{\hat{x}_{s_1}}, 0, 1/\sqrt{\hat{x}_{s_1}}$. □

Lemma 2. *For* $y_0 \in (0, 1/\sqrt{\hat{x}_{s_1}}]$*, the recurrence will maintain the invariant,* $y_i \in (0, 1/\sqrt{\hat{x}_{s_1}}]$.

Proof. We have the Newton iteration function $\varphi(y_i) = y_{i+1} = (3y_i - \hat{x}_{s_1} y_i^3)/2$. Since φ is continuous, it must have a maximum over the domain $[0, 1/\sqrt{\hat{x}_{s_1}}]$. The derivative, $\varphi'(y_i) = (3 - 3\hat{x}_{s_1} y_i^2)/2$, is 0 over this domain at $y_i = 1/\sqrt{\hat{x}_{s_1}}$.

Evaluating $\varphi(y_i)$ at $y_i = 0$ and $y_i = 1/\sqrt{x}$, we find by the Extreme Value Theorem that φ has a maximum value on this domain of $1/\sqrt{\hat{x}_{s_1}}$ (at endpoint $y_i = 1/\sqrt{\hat{x}_{s_1}}$).

Since $\varphi(y_i)$ is always positive for $y_0 \in (0, 1/\sqrt{\hat{x}_{s_1}}]$, this means that $0 < y_{i+1} \leq 1/\sqrt{\hat{x}_{s_1}}$ and thus we inductively know that the property holds for all subsequent y_i. Moreover, we are assured by Lemma 1 that for y_0 in this range, the recurrence can only converge to $1/\sqrt{\hat{x}_{s_1}}$ (which is the only point in this range where $y_{i+1} = y_i$). □

Lemma 3. *The Newton iteration error, $e_i = |y_i - 1/\sqrt{\hat{x}_{s_1}}|$, satisfies*

$$e_{i+1} = \frac{3e_i^2\sqrt{\hat{x}_{s_1}} - \hat{x}_{s_1}e_i^3}{2}$$

Proof. By Lemma 2, $e_i = |y_i - 1/\sqrt{\hat{x}_{s_1}}| = 1/\sqrt{\hat{x}_{s_1}} - y_i$. The error of the next term is $e_{i+1} = |y_{i+1} - 1/\sqrt{\hat{x}_{s_1}}| = 1/\sqrt{\hat{x}_{s_1}} - (3y_i - \hat{x}_{s_1}y_i^3)/2$. Indeed, we have,

$$\frac{3e_i^2\sqrt{\hat{x}_{s_1}} - \hat{x}_{s_1}e_i^3}{2} = \frac{3(1/\sqrt{\hat{x}_{s_1}} - y_i)^2\sqrt{\hat{x}_{s_1}} - \hat{x}_{s_1}(1/\sqrt{\hat{x}_{s_1}} - y_i)^3}{2} =$$

$$\frac{3(1/\hat{x}_{s_1} + y_i^2 - 2y_i/\sqrt{\hat{x}_{s_1}})\sqrt{\hat{x}_{s_1}}}{2} \text{ (contd.)}$$

$$-\frac{\hat{x}_{s_1}(1/(\hat{x}_{s_1}\sqrt{\hat{x}_{s_1}}) - y_i^3 + 3y^2/\sqrt{\hat{x}_{s_1}} - 3y/\hat{x}_{s_1})}{2} =$$

$$\frac{1}{\sqrt{\hat{x}_{s_1}}} - \frac{3y_i - \hat{x}_{s_1}y_i^3}{2} = e_{i+1}.$$

□

Lemma 4. *The Newton iteration error satisfies the bound*

$$e_i < \left(\frac{3}{2} e_0 \sqrt{\hat{x}_{s_1}}\right)^{2^i}$$

Proof. Since $e_i > 0$ and $\hat{x}_{s_1} > 0$, we have

$$e_{i+1} = \frac{3e_i^2\sqrt{\hat{x}_{s_1}} - \hat{x}_{s_1} e_i^3}{2} < \frac{3e_i^2\sqrt{\hat{x}_{s_1}}}{2}.$$

Unfolding this recurrence, we have

$$e_i < \left(\frac{3}{2}\right)^{2^i-1} (e_0)^{2^i} (\sqrt{\hat{x}_{s_1}})^{2^i-1} < \left(\frac{3}{2} e_0 \sqrt{\hat{x}_{s_1}}\right)^{2^i}.$$

This error will converge to 0 if $(\frac{3}{2} e_0 \sqrt{\hat{x}_{s_1}}) < 1$, or equivalently if $e_0 < 2/(3\sqrt{\hat{x}_{s_1}})$. □

Lemma 5. *For the stated initial approximation, the error of the Newton method after i iterations satisfies*

$$e_i < (.75)^{2^i}$$

Proof. We use the initial approximation, $\hat{y}_0 = y_0 = 2^{\lfloor (p-1)/2 \rfloor}$, where $2^{-p} \leq \hat{x}_{s_1} < 2^{-p+1}, p \in \mathbb{Z}$. Thus, $2^{p/2-1} \leq y_0 \leq 2^{(p-1)/2}$ and $2^{-p/2} \leq \sqrt{\hat{x}_{s_1}} <$

$2^{(-p+1)/2}$. Now, we find an upper bound for

$$e_0\sqrt{\hat{x}_{s_1}} = |y_0 - 1/\sqrt{\hat{x}_{s_1}}|\sqrt{\hat{x}_{s_1}}.$$

Since $y_0 = 2^{\lfloor (p-1)/2 \rfloor} \leq 2^{(p-1)/2} \leq 1/\sqrt{\hat{x}_{s_1}}$, we have

$$e_0\sqrt{\hat{x}_{s_1}} = (1/\sqrt{\hat{x}_{s_1}} - y_0)\sqrt{\hat{x}_{s_1}} = 1 - y_0\sqrt{\hat{x}_{s_1}}.$$

By the bounds on y_0 and $\sqrt{\hat{x}_{s_1}}$, we have

$$2^{p/2-1} \cdot 2^{-p/2} = 1/2 \leq y_0\sqrt{\hat{x}_{s_1}} \leq 2^{(p-1)/2} \cdot 2^{(-p+1)/2} = 1.$$

Thus, $e_0\sqrt{\hat{x}_{s_1}}$ is at most $1 - 1/2 = 1/2$. Plugging in to the inequality in Lemma 4, we have

$$e_i < \left(\frac{3}{2}e_0\sqrt{\hat{x}_{s_1}}\right)^{2^i} \leq \left(\frac{3}{2} \cdot \frac{1}{2}\right)^{2^i} = (.75)^{2^i}.$$

□

Lemma 6. *For this stage, the total round off error, $|\hat{y}_{s_2} - y_{s_2}|$, is at most*

$$(9/8)^{s_2}2^{3-b_2}$$

where s_2 is the number of iterations performed and b_2 is the number of bits of accuracy in $\hat{y}_{s_2}$.

Proof. Since we use fixed precision arithmetic, there is a truncation error at each step, $\epsilon_i = \hat{y}_i - \varphi(\hat{y}_{i-1})$, with $\epsilon_0 = 0$. Thus,

$$|\hat{y}_{s_2} - y_{s_2}| = |\epsilon_{s_2} + \varphi(\hat{y}_{s_2-1}) - \varphi(y_{s_2-1})| \leq |\epsilon_{s_2}| + |\varphi(\hat{y}_{s_2-1}) - \varphi(y_{s_2-1})|$$

Next, we note that $y_i \in (1/(2\sqrt{\hat{x}_{s_1}}), 1/\sqrt{\hat{x}_{s_1}})$ because $e_0 \leq 1/(2\sqrt{\hat{x}_{s_2}})$ and all further error terms are decreasing. Therefore,

$$|\varphi'(y_i)| = |(3 - 3\hat{x}_{s_1} y_i^2)/2| \leq 9/8$$

and so

$$|\varphi(\hat{y}_{s_2-1}) - \varphi(y_{s_2-1})| \leq \max(|\varphi'(y_i)|)|\hat{y}_{s_2-1} - y_{s_2-1}| \leq \frac{9}{8}|\hat{y}_{s_2-1} - y_{s_2-1}|.$$

Thus, we have the relation,

$$|\hat{y}_{s_2} - y_{s_2}| \leq |\epsilon_{s_2}| + \frac{9}{8}|\hat{y}_{s_2-1} - y_{s_2-1}|.$$

Unfolding to the next term gives

$$|\hat{y}_{s_2} - y_{s_2}| \leq |\epsilon_{s_2}| + \frac{9}{8}|\epsilon_{s_2-1} + \frac{9}{8}|\hat{y}_{s_2-2} - y_{s_2-2}|| =$$

$$= (9/8)^0|\epsilon_{s_2}| + (9/8)^1|\epsilon_{s_2-1}| + (9/8)^2|\hat{y}_{s_2-2} - y_{s_2-2}|.$$

Fully unfolding the recurrence, we have

$$|\hat{y}_{s_2} - y_{s_2}| \leq \sum_{i=1}^{s_2} (9/8)^{s_2-i} |\epsilon_i|.$$

With b_2 bits of accuracy, $|\epsilon_i| \leq 2^{-b_2}$ and we further have

$$|\hat{y}_{s_2} - y_{s_2}| \leq 2^{-b_2} \sum_{i=1}^{s_2} (9/8)^{s_2-i} = 2^{-b_2} \sum_{i=0}^{s_2-1} (9/8)^i =$$

$$= 2^{-b_2} \frac{1 - (9/8)^{s_2}}{1 - 9/8} \leq (9/8)^{s_2} 2^{3-b_2}$$

□

5 Total Error

We can bound the total error by

$$|\hat{y}_{s_2} - \sqrt{v}| \leq |\hat{y}_{s_2} - 1/\sqrt{\hat{x}_{s_1}}| + |1/\sqrt{\hat{x}_{s_1}} - \sqrt{v}|.$$

Now suppose we desire a total error of at most E. We can split the error across these two terms so that each contributes an error of at most $E/2$.

By Lemma 5 and Lemma 6, we know that the first term is bounded by the sum of the Newton iteration and round off errors for the inverse square root stage,

$$|\hat{y}_{s_2} - 1/\sqrt{\hat{x}_{s_1}}| \leq (.75)^{2^{s_2}} + (9/8)^{s_2} 2^{3-b_2}.$$

Next, we further split the desired error $E/2$ across these two terms, so that each contributes an error of at most $E/4$. Thus, we first have

$$(.75)^{2^{s_2}} \leq \frac{E}{4}$$

or

$$s_2 = \lceil \log_2 \log_{4/3} \frac{4}{E} \rceil$$

For the next term, we have

$$(9/8)^{s_2} 2^{3-b_2} = (9/8)^{\lceil \log_2 \log_{4/3} \frac{4}{E} \rceil} 2^{3-b_2} \leq \frac{E}{4}$$

so that

$$b_2 = 3 + \lceil \log_2 \frac{4}{E} + \log_2 \frac{9}{8} \lceil \log_2 \log_{4/3} \frac{4}{E} \rceil \rceil.$$

Now, we also desire an error of at most $E/2$ for the second absolute value term, $|1/\sqrt{\hat{x}_{s_1}} - \sqrt{v}|$. By Lemma 7 below, setting $|\hat{x}_{s_1} - 1/v| \leq E/(3v^{1.5})$ is sufficient so that $|1/\sqrt{\hat{x}_{s_1}} - \sqrt{v}| \leq E/2$.

Since $|\hat{x}_{s_1} - 1/v| \leq 2^{-2^{s_1}} + s_1 2^{-b_1}$ [2, Thm B.1], we can split an error of $E/(3v^{1.5})$ between these two terms. Thus,

$$2^{-2^{s_1}} \leq \frac{1}{2} \frac{E}{3v^{1.5}}$$

and

$$s_1 = \lceil \log_2 \log_2 \frac{6v^{1.5}}{E} \rceil.$$

For the final term, we have

$$s_1 2^{-b_1} = \lceil \log_2 \log_2 \frac{6v^{1.5}}{E} \rceil 2^{-b_1} \leq \frac{1}{2} \frac{E}{3v^{1.5}}$$

so that

$$b_1 = \lceil \log_2 \frac{6v^{1.5}}{E} + \log_2 \lceil \log_2 \log_2 \frac{6v^{1.5}}{E} \rceil \rceil.$$

Lemma 7. *If*

$$|\hat{x}_{s_1} - 1/v| \leq \frac{E}{3v^{1.5}}$$

then

$$|1/\sqrt{\hat{x}_{s_1}} - \sqrt{v}| < \frac{E}{2}.$$

Proof. We start with

$$|\hat{x}_{s_1} - 1/v| \leq \frac{E}{3v^{1.5}}.$$

Multiplying the left hand side by $v/\left(\sqrt{\hat{x}_{s_1}}(1 + \sqrt{v\hat{x}_{s_1}})\right)$ yields

$$|\hat{x}_{s_1} - 1/v| \cdot \frac{v}{\sqrt{\hat{x}_{s_1}}(1 + \sqrt{v\hat{x}_{s_1}})} = \left| \frac{1 - \sqrt{v\hat{x}_{s_1}}}{\sqrt{\hat{x}_{s_1}}} \right| = \left| \frac{1}{\sqrt{\hat{x}_{s_1}}} - \sqrt{v} \right|$$

so we have

$$\left| \frac{1}{\sqrt{\hat{x}_{s_1}}} - \sqrt{v} \right| \leq \frac{E}{3v^{1.5}} \cdot \frac{v}{\sqrt{\hat{x}_{s_1}}(1 + \sqrt{v\hat{x}_{s_1}})} = \frac{E}{3\sqrt{v}} \cdot \frac{1}{\sqrt{\hat{x}_{s_1}}(1 + \sqrt{v\hat{x}_{s_1}})}.$$

Since $|\hat{x}_{s_1} - 1/v| \leq E/(3v^{1.5})$, the denominator of the last term is equal to at least

$$\sqrt{1/v - E/(3v^{1.5})}\left(1 + \sqrt{v(1/v - E/(3v^{1.5}))}\right).$$

Moreover, since $E < \sqrt{v}$ (otherwise $y = 0$ would be an immediate and trivial approximation for $\sqrt{v}$ that would satisfy the desired error), this is at least

$$\sqrt{1/v - \sqrt{v}/(3v^{1.5})}\left(1 + \sqrt{v\left(1/v - \sqrt{v}/(3v^{1.5})\right)}\right) =$$

$$= \sqrt{1/v - 1/(3v)}\left(1 + \sqrt{v(1/v - 1/(3v))}\right) =$$

$$= \sqrt{\frac{2}{3v}}\left(1 + \sqrt{\frac{2}{3}}\right) = \left(\frac{2}{3} + \sqrt{\frac{2}{3}}\right) \cdot \frac{1}{\sqrt{v}}$$

Returning to the inequality, we have

$$\left|\frac{1}{\sqrt{\hat{x}_{s_1}}} - \sqrt{v}\right| \leq \frac{E}{3\sqrt{v}} \cdot \frac{1}{\sqrt{\hat{x}_{s_1}}(1 + \sqrt{v\hat{x}_{s_1}})} \leq \frac{E}{3\sqrt{v}} \cdot \frac{1}{(2/3 + \sqrt{2/3})/\sqrt{v}} \leq \frac{E}{2}.$$

And so we indeed have the final inequality,

$$\left|\frac{1}{\sqrt{\hat{x}_{s_1}}} - \sqrt{v}\right| < \frac{E}{2}.$$

□

6 Total Cost

Lemma 8. *The total number of iterations required is of order* $O(\log_2 \log_2 \frac{6v^{1.5}}{E})$.

Proof. To achieve a total error, $|\hat{y}_{s_2} - \sqrt{v}|$, of at most E, we require $s_1 = \lceil \log_2 \log_2 \frac{6v^{1.5}}{E} \rceil = O(\log_2 \log_2 \frac{6v^{1.5}}{E})$ iterations of the inverse circuit and $s_2 = \lceil \log_2 \log_{4/3} \frac{4}{E} \rceil = O(\log_2 \log_2 \frac{4}{E})$ iterations of the inverse square root circuit. s_1 is asymptotically greater than s_2 and the total number of iterations required to achieve an error of E is thus of order $O(\log_2 \log_2 \frac{6v^{1.5}}{E})$. □

Lemma 9. *The maximum number of qubits required in the circuit at any particular stage is of order* $O(\log_2 \frac{v^{1.5}}{E})$.

Proof. Initially, storing the input v requires $\lceil \log_2 v \rceil$ qubits. The inversion stage requires $b_1 = \lceil \log_2 \frac{6v^{1.5}}{E} + \log_2 \lceil \log_2 \log_2 \frac{6v^{1.5}}{E} \rceil \rceil = O(\log_2 \frac{v^{1.5}}{E})$ bits of accuracy. The final inverse square root stage requires $b_2 = 3 + \lceil \log_2 \frac{4}{E} + \log_2 \frac{9}{8} \lceil \log_2 \log_{4/3} \frac{4}{E} \rceil \rceil = O(\log_2 \frac{4}{E})$ qubits at the beginning and end of each iteration. Since the first term, $O(\log_2 \frac{v^{1.5}}{E})$ is asymptotically largest, we conclude that the maximum number of qubits required in the circuit at any particular stage is of order $O(\log_2 \frac{v^{1.5}}{E})$. □

Lemma 10. *The total number elementary operations (single and singly-controlled gates) required is of order* $O((\log_2 \frac{v^{1.5}}{E})^3 \log_2 \log_2 \frac{6v^{1.5}}{E})$, *a low degree polynomial in* b_1 *and* s_1.

Proof. The initial approximation circuits for the inversion stage requires one

doubly-controlled gate and one singly-controlled gate per bit of input [2, Fig. 3]. The same is true for the initial approximation circuit in Figure 3 for the inverse square root stage. Both circuits also require a multiply controlled X gate, which can be implemented with a number of single and singly-controlled gates that is linear to the input size [1, Lemma 7.2]. Therefore, both initialization circuits require a number of elementary operations that is linear with the input sizes.

Thus the asymptotically dominant factor in the number of elementary operations is the multiplication circuit, which for b bit input requires at most on the order of b^3 elementary operations [6, Section IV]. Since b_1 and s_1 are both asymptotically dominant over b_2 and s_2, this means that we require $O(s_1 b_1^3)$ elementary operations in total. We bound

$$b_1^3 = (\lceil \log_2 \frac{6v^{1.5}}{E} + \log_2 \lceil \log_2 \log_2 \frac{6v^{1.5}}{E} \rceil \rceil)^3 <$$

$$< (1 + \log_2 \frac{6v^{1.5}}{E} + \log_2 \lceil \log_2 \log_2 \frac{6v^{1.5}}{E} \rceil)^3 = O\left(\left(\log_2 \frac{v^{1.5}}{E} \right)^3 \right).$$

We also know that s_1 is of order $O(\log_2 \log_2 \frac{6v^{1.5}}{E})$. Thus, we can conclude that the total number of elementary operations required is of order $O((\log_2 \frac{v^{1.5}}{E})^3 \log_2 \log_2 \frac{6v^{1.5}}{E})$, a low degree polynomial in b_1 and s_1. □

The following theorem summarizes the asymptotic results for total costs.

Theorem 1. *To achieve an error, $|\hat{y}_{s_2} - \sqrt{v}|$, of at most E, the approach in*

this paper has the following costs:

- *The total number of iterations is of order* $O(\log_2 \log_2 \frac{6v^{1.5}}{E})$.
- *At any particular stage, the maximum number of qubits required is of order* $O(\log_2 \frac{v^{1.5}}{E})$.

7 Extension to Arbitrary Powers

By repeated square root operations, the black box circuit in Figure 5 that computes v^z ($v > 1$ and $0 \leq z < 1$) can be implemented.

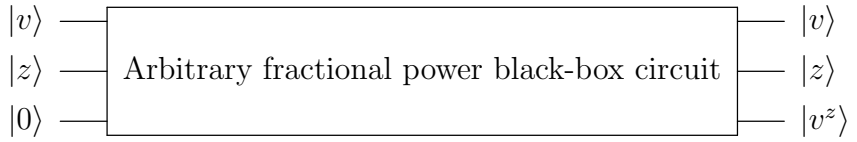

Figure 5: Black box for circuit computing v^z, with $v > 1$ and $0 \leq z < 1$.

First, the circuit in Figure 6 is used to compute $v^{1/2}, v^{1/4}, \ldots, v^{1/2^n}$, where the power z is an n bit fractional power $z = 0.z_{n-1}z_{n-2}...z_1z_0$. This stage requires n of the square root modules proposed in this paper.

Then, we use the n cascading controlled multiplication circuits in Figure 7 to multiply the correct combination of powers of v. For example, if we have $z = 0.1011_2$ ($11/16_{10}$), the circuit in Figure 6 first computes approximations for $v^{1/2}, v^{1/4}, v^{1/8}$, and $v^{1/16}$. Finally, the circuit in Figure 7 multiplies $v^{1/2} \cdot v^{1/8} \cdot v^{1/16} = v^{11/16}$.

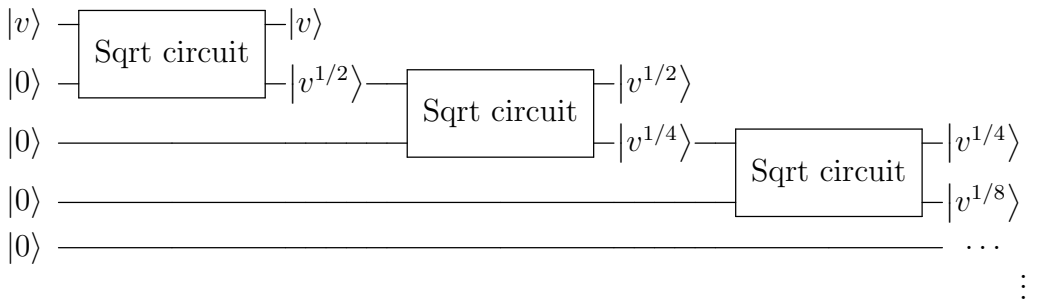

Figure 6: Cascading-style design for iterative square roots. Because the square root circuit is only an approximation, $v^{1/2}$ denotes an approximation for the square root of v. Similarly, $v^{1/4}$ denotes an approximation for the square root of the approximation for $v^{1/2}$ and so forth.

As an extension, if a client desires to take the "r^{th} root" of v, r can first be fed into an inversion circuit to attain a binary approximation for $1/r$. Up to such an approximation, this reciprocal can be treated as z and $v^z \approx v^{1/r}$.

8 Extension to Arbitrary Fractional Powers of 2

For the special case of computing $v^{1/z}$ where $z = 2^k$, the cascading style of the arbitrary power circuit can be avoided. This may lead to smaller errors and costs because the cascading style involves large error propogation in the successive approximations of $v^{1/2}$ to $v^{1/4}$ to $v^{1/8}$ and so forth.

Instead, we can just start by inverting as with the square root circuit to obtain $\hat{x}_{s_1} = \frac{1}{v}$. Next, we apply Newton's method with the function $g(y) = 1/y^z - \hat{x}_{s_1}$ (which has root $y = (1/\hat{x}_{s_1})^{1/z}$ as desired). The recurrence

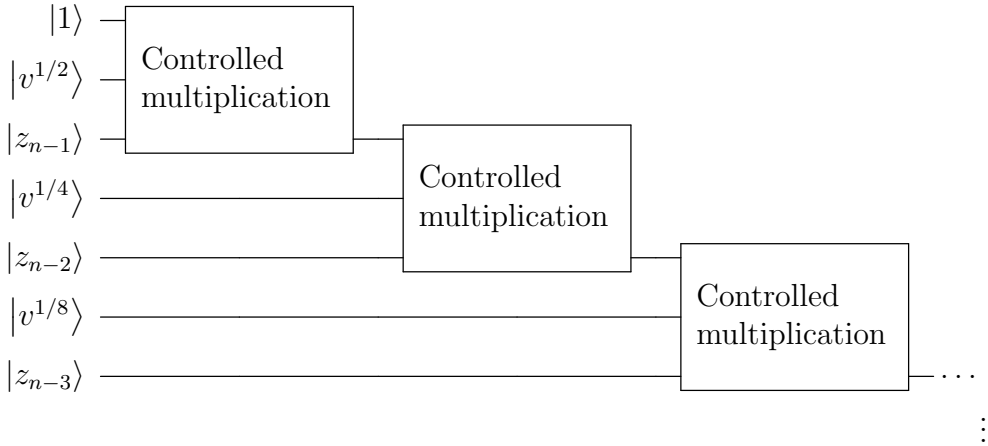

Figure 7: Circuit that multiplies appropriate powers of v to output v^z. Once $v^{1/2}, v^{1/4}, v^{1/8}, \ldots$ are computed, this circuit is applied to find v^z. Here, $z = 0.z_{n-1}z_{n-2}...z_1z_0$. Each controlled multiplication circuit outputs the product of the top two inputs if the control bit (the bottom input) is 1. Otherwise, it outputs the top input, unchanged. For brevity, the additional outputs (required for reversibility) of the controlled multiplication module are omitted in this diagram.

is

$$\varphi(y_i) = y_{i+1} = y_i - \frac{g(y_i)}{g'(y_i)} = \frac{(z+1)y_i - \hat{x}_{s_1} y^{z+1}}{z}.$$

Since z is a power of 2, the division by z can be implemented, again by using a form of repeatedly right shifting in a quantum register extended by one qubit set to $|0\rangle$. The numerator too can be implemented by arithmetic circuits.

The square root circuit in this paper is actually just the special case where $z = 2^1$. Error analysis for the other powers has not been carried out here and is an area for future research. Moreover, if there is a circuit that can implemented division by a fixed scalar that is not neccessarily a power

of 2, then this method can be extended for all $z \in \mathbb{N}$.

9 Conclusion

The procedure presented here allows the approximation of $\sqrt{v}, v > 1$ with a quantum circuit. If a total error of at most E is desired, then the procedure requires order $O(\log_2 \log_2 \frac{6v^{1.5}}{E})$ iterations and a maximum of $O(\log_2 \frac{v^{1.5}}{E})$ qubits at any particular stage.

Despite the dependence of these terms on v, these are practical rates of growth. For example, the number of iterations required grows very slowly, because it involves a double logarithm that acts on the already-low powers of v and E. Moreover, if we consider the fractional error, E/v, with $E \leq 1$, the order of growth can be written as a maximum of $O(\log_2 \frac{1}{E/v})$ qubits at any particular stage, which is reasonably small.

Finally, we should note that the error analysis in this paper is based on worst-case analysis and conservative estimates. For instance, the desired error E was split into four terms in the analysis with each contributing an error of at most $E/4$. However, each term will not contribute exactly $E/4$ error and thus the total error will always end up being lower than E. Thus, in practice, it would be reasonable to expect even lower errors or resource requirements. Further work could be done to find tighter cost estimates by formulating the error analysis as an optimization problem with an objective

function that minimizes required resources.

Future work is also necessary to obtain error analysis and cost estimates for the arbitrary power v^z circuit proposed in this paper. However, more precise cost estimates for arithmetic circuits are first required before such further work. Error analysis, including finding a sufficient initial approximation, is also required for the fractional power of 2 circuit proposed in this paper.

Since quantum computing is such a new field, it is difficult to predict exactly where and how a quantum square root circuit will be of use. Nonetheless, this paper represents an important step in the implementation of elementary functions. Moreover, since the resource requirements for the proposed circuit are reasonable, it will be feasible to implement when large-scale quantum circuit fabrication is possible. Until then, this paper provides a key step that can be used as a module for solving other problems with quantum computing.

10 Acknowledgements

I would like to thank my advisor, Dr. Iasonas Petras, for spending many hours guiding me through this research problem. This paper would not have been possible if not for his patience and dedication. I am also grateful for help from Dr. Anargyros Papageorgiou at Columbia University, who provided

helpful feedback at various steps throughout the research process.

References

[1] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, A52, March 1005.

[2] Yudong Cao, Anargyros Papageorgiou, Iasonas Petras, Joseph Traub, and Sabre Kais. Quantum algorithm and circuit design solving the poisson equation. *New Journal of Physics*, 15, January 2013.

[3] IARPA. Quantum computer science program. In *Broad Agency Announcement IARPA-BAA-10-02*, April 2010.

[4] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J.Sci.Statist.Comput*, 26:1484–1509, October 1997.

[5] Sayeeda Sultana and Katarzyna Radecka. Reversible implementation of square-root circuit. In *Electronics, Circuits and Systems (ICECS)*, pages 141–144, 2011.

[6] Vlatko Vedral, Adriano Barenco, and Artur Ekert. Quantum networks for elementary arithmetic operations. *Phys. Rev. A*, 54:147–153, July 1996.

[7] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler. RevLib: An online resource for reversible functions and reversible circuits. In *Int'l Symp. on Multi-Valued Logic*, pages 220–225, 2008.