Theoretical Physics

Mathematical Analysis and Simulation of Shor's Algorithm and the Quantum Fourier Transform

Marcus Ahlström (881221-0717)
marcusah@kth.se

Peda Dizdarevic (900201-5890)
pedad@kth.se

SA104X Degree Project in Engineering Physics, First Level
Department of Theoretical Physics
Royal Institute of Technology (KTH)
Supervisor: Patrik Henelius

May 9, 2012

Abstract

In 1994, Peter Shor presented an algorithm for integer factorization that used exponentially less operations than the most efficient known algorithm. His algorithm requires the use of a quantum computer, a theoretical computational device using quantum mechanical effects not utilized in contemporary computers. In this paper we have analysed the mathematics behind Shor's algorithm and the quantum circuits on which it operates. We have also studied the Quantum Fourier Transform, a central component of Shor's Algorithm. Furthermore we have written a program in C++ to simulate a quantum circuit performing Shor's algorithm and the quantum Fourier transform. We were able to understand the critical parts of Shor's algorithm that contribute with the great increase in efficiency compared to classical algorithms.

Contents

Chapter 1

Introduction

The RSA-algorithm is essential in our daily lives. It encrypts our bank information and e-mails, and protects our personal information when we are connected to the internet. The safety of the RSA-encryption is largely based on the difficulty for a computer to factor very large integers in reasonable time. However, Shor's algorithm, working on a quantum computer (as opposed to a *classical* computer), may be able to factorize even integers of orders exceeding 1000 digits in reasonable time.

Shor's Algorithm was named after its creator Peter Shor, who first published the algorithm in 1994. Quantum computers had already been discussed for many years by researcher and popular media, but it with Shor's discovery that the potential of quantum computers was beginning to be understood. What was most remarkable was not the class of problems Shor's algorithm solved, nor the implications it has on our daily lives, but the speed at which it does it. Shor's algorithm solves operates with exponentially faster than the fastest known classical algorithm. Shor's algorithm has thus awoken a curiosity among physicists, mathematicians and computer scientists alike. After all, could other problems that are classically very difficult, or require an excessive amount of resources, be easily solvable on a quantum computer?

Chapter 2

Background Material

In this paper we will analyse quantum computational operations utilizing a heavily mathematical approach. To help readers unused to such a theoretical approach we will first give a short overview of the mathematical concepts we use. Thereafter follows a presentation of the basis of quantum mechanics before we wrap up the chapter by explaining the fundamentals of quantum circuits and quantum computation.

2.1 Mathematics

2.1.1 Dirac notation

In this thesis, elements of a complex n-dimensional vectorspace V will be represented using the Dirac-notation, by a *ket* denoted

$$|\psi\rangle = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}, \tag{2.1}$$

where α_i is a complex number. To this vector space V there is an associated dual vector space, V^*. Elements in V^* are represented by a *bra*, denoted

$$\langle\psi| = \begin{bmatrix} \alpha_1^* & \alpha_2^* & \dots & \alpha_n^* \end{bmatrix}, \tag{2.2}$$

where α_i^* is the complex conjugate of α_i.

We define a n-dimensional Hilbert space $\mathcal{H}$ as a complex vector space with an *inner product*, called a *bracket* (bra-ket) $\langle . \mid . \rangle$

$$\langle . \mid . \rangle : \mathcal{H} \times \mathcal{H}^* \to \mathbb{C}. \tag{2.3}$$

The following relations are true for the inner product [2]:

$$\langle \psi \mid \phi \rangle = \overline{\langle \phi \mid \psi \rangle} \text{ (conjugate symmetry).} \tag{2.4}$$

$$\langle \psi \mid (\alpha + \beta)\phi \rangle = \alpha \langle \psi \mid \phi \rangle + \beta \langle \psi \mid \phi \rangle, \forall \alpha, \beta \in \mathbb{C} \text{ (linearity).} \tag{2.5}$$

$$\langle \psi \mid \psi \rangle \geq 0 \text{ and } \langle \psi \mid \psi \rangle = 0 \Leftrightarrow |\psi\rangle = 0 \text{ (positive definiteness).} \tag{2.6}$$

2.1.2 Linear Operators

A linear operator is an function between vector spaces V, W, $\mathcal{L} : V \to W$ such that:

$$\mathcal{L}\left((\alpha + \beta)\right)|\psi\rangle = \alpha\mathcal{L}|\psi\rangle + \beta\mathcal{L}|\psi\rangle . \tag{2.7}$$

Using the fact that $(\sum_{k=1}^{n} |i\rangle\langle i|) = I.$, we also get:

$$\mathcal{L} = I\mathcal{L}I = \left(\sum_{i}^{n} |i\rangle\langle i|\right)\mathcal{L}\left(\sum_{j}^{n} |j\rangle\langle j|\right) = \tag{2.8}$$

$$\sum_{i,j=0}^{n} |i\rangle\langle i|\mathcal{L}|j\rangle\langle j| = \sum_{i,j=0}^{n} \left(\langle i|\mathcal{L}|j\rangle\right)|i\rangle\langle j| = \tag{2.9}$$

$$\sum_{i,j=0}^{n} \mathcal{L}_{ij}|i\rangle\langle j| , \tag{2.10}$$

where $\mathcal{L}_{ij}$ is the matrix element on the i:th row and j:th column.

2.1.3 Binomial fraction

It will sometimes be of use to represent a vector by its binomial expansion. We will write this as

$$|j\rangle = |j_1 j_2 ... j_n\rangle . \tag{2.11}$$

Where j_n is either 0 or 1. It will also be convenient to denote the binomial fraction by $0.j_1 j_2 ... j_n$ where:

$$0.j_1 j_2 \ldots j_n = \frac{j_1}{2} + \frac{j_2}{4} + ... + \frac{j_n}{2^n}. \tag{2.12}$$

2.1.4 Tensor Product

Having two matrices V and W, being of sizes $n \times m$ and $p \times q$ respectively, we define the tensor product $\otimes$ as the $mp \times nq$ matrix:

$$V \otimes W = \begin{bmatrix} v_{11}W & v_{12}W & \ldots & v_{1n}W \\ v_{21}W & v_{22}W & \ldots & v_{2n}W \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ v_{m1}W & v_{m2}W & \ldots & v_{mn}W \end{bmatrix} . \tag{2.13}$$

2.1.5 The Discrete Fourier Transform

The *discrete fourier transform* is a powerful mathematical tool with many application in mathematics and computer science. It is a mapping of an element in a set of complex numbers onto another complex number according to:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k/N}. \tag{2.14}$$

We will utilize the discrete fourier transform later on in this paper when presenting quantum algorithms for integer factorization on a quantum computer.

2.2 Quantum Physics

The four postulates of quantum mechanics are [3]:

1. With every quantum system is an associated Hilbert space $\mathcal{H}$, called the *state space*. The system is completely described by its *state vector*; a unit vector in $\mathcal{H}$.

2. Evolution of a *closed* system is described by unitary transformations.

   A closed system is a system which do not interact with the rest of the universe. A unitary operator is a described by a matrix U such that $UU^\dagger = U^\dagger U = I$ where I is the identity matrix. This is to say, two states $|\psi\rangle$ and $|\phi\rangle$ at two different times t_1 and t_2 are related by
   $$|\psi\rangle = U\,|\phi\rangle\,. \tag{2.15}$$
   A closed system serves very little purpose to science, as the scientist will want to look at the quantum system to see what happens. When the laboratory equipment interacts with the closed system, it no longer evolves unitary. This act of *measuring* the system, is explained by the next postulate

3. Performing a *quantum measurement* is done by applying a series of measurement operators $\{M_k\}_{k=0}^n$ where n is the number of outcomes that can come from measuring the quantum system.

   A system being in the state $|\psi\rangle$ right before the measurement will have a probability of yielding the i:th measurement, determined by:
   $$P(k=i) = \left\langle \psi \mid M_i^\dagger M_i \mid \psi \right\rangle. \tag{2.16}$$
   After the measurement is done, the system has been put in the new state
   $$\frac{M_i\,|\psi\rangle}{\sqrt{\left\langle \psi \mid M_i^\dagger M_i \mid \psi \right\rangle}}. \tag{2.17}$$

4. If we combine multiple quantum systems, the composite state space is given by the tensor product of each individual system's state space. If our composite space is built up by n spaces, and if the k:th space is the state $|\psi_k\rangle$, the state for a composite state space is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle = |\psi_1, \psi_2, ..., \psi_n\rangle = |\psi_1\psi_2...\psi_n\rangle\,.$

2.3 Classical Computer Science

2.3.1 Computer Operations

In this thesis, we will distinguish between a quantum computer, and a computer that we think in a daily sense. The latter we will frequently address as a *classical* computer. In a classical computer the smallest component is the *bit*. A bit can be thought of as a switch, that is either off or on. This is normally represented by the bit assuming either of the discrete values (states) 0 or 1 respectively. A state can be operated on by a computer

operation, a so called gate. Examples include the *not*, *and* and the *or*. For two states a and b:

$$\neg a = \begin{cases} 1 & \text{if } a = 0 \\ 0 & \text{if } a = 1. \end{cases} \tag{2.18}$$

$$a \wedge b = \begin{cases} 1 & \text{if } a = b = 1 \\ 0 & \text{if } a \text{ or } b \neq 1 \end{cases} \tag{2.19}$$

$$a \vee b = \begin{cases} 1 & \text{if } a \text{ or } b \neq 0 \\ 0 & \text{if } a = b = 0. \end{cases} \tag{2.20}$$

These operation can be combined, and be used in larger systems to create complicated circuits. A combined operation could be the *nand* (not-and).

2.3.2 Integer Factorization and RSA-Cryptography

One of the greatest problems in arithmetics is that of factorizing a composite integer into prime factors. That is, for an integer N find integers p and q such that $N = pq$. One of the main focuses of this paper will be to show how to solve this problem on a quantum computer.

The solution of the factorization problem has a very significant application in the field of cryptography, more precisely in regards to RSA-cryptography.

RSA-cryptography works as follows. Given an integer x that we want to encrypt we choose large integers p and q and form $N = pq$. From N we then calculate a so-called public key c and a private key d. The keys are integers calculated from p,q and N, and used to form encryption and decryption functions $e(x)$ and $d(x)$.

$$e(x) = x^c, \text{ (mod N)} \tag{2.21}$$

$$d(x) = x^d, \text{ (mod N)} \tag{2.22}$$

$$d(e(x)) = x. \tag{2.23}$$

As long as the decryption key d is kept secret anyone can use the encryption key c to send information, but only those who possess d can read it. The integer factorization problem becomes an issue here because the task of decrypting information without knowing d by reverse engineering it from n and e can be reduced to factorizing $n = pq$, with the rest of the steps being trivial in this context. [4]

It is of interest to review the ability of classical computers to factorize large integers.

We shall do so by considering the most efficient classical algorithms for factorizing integers in terms of the number of bit operations they require to factorize an integer of a certain size. Later on we will compare this with the corresponding cost in bit operations for quantum computational algorithms factorizing the same numbers. With bit operation we mean preforming the kind of operations mentioned in 2.3.1 once on one bit.

The most efficient classical factorization algorithm is the *number field sieve* [5]. This algorithm requires ν bit operations to factorize the integer N, where ν grows in size with N according to:

$$\nu(N) \approx \Omega(\exp\left[(1.90)(\log_2(N))^{1/3}(\log_2(\log_2(N)))^{2/3}\right]). \qquad [5] \tag{2.24}$$

2.4 Quantum Computation

The quantum analogy of the bit is the quantum bit, or *qubit*. Just as its classical counterpart the qubit may assume the states $|0\rangle, |1\rangle$, but may also assume a linear combination of them. [3]. The qubit is the simplest state space, with two basis vectors $|0\rangle$, and $|1\rangle$.

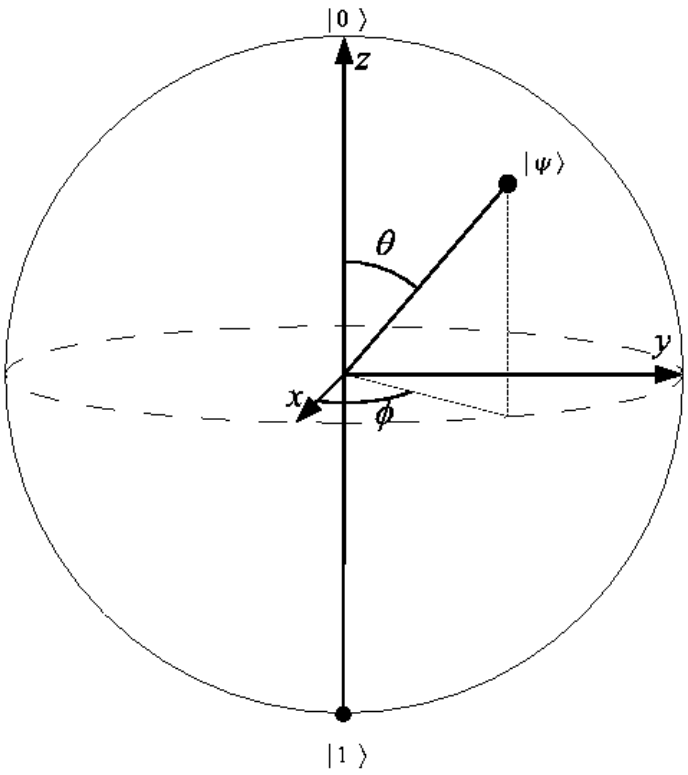

Figure 2.1: *The qubit can visually be represented by the* Bloch sphere*, where the pure states $|\psi\rangle$ are points on the sphere's hull. [6]*

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.25}$$

where it follows from the first postulate, that

$$|\,\alpha\,|^2 + |\,\beta\,|^2 = 1. \tag{2.26}$$

The states $|0\rangle$ and $|1\rangle$ can be interpreted as the traditional spin of a particle, or as a switch that is turned on or off, just as the classical bit. The ability to superposition two states $|0\rangle, |1\rangle$to create a new state makes the interpretation somewhat different than the classical case. Instead of thinking of a switch as was done in the classical case, the qubit is often thought of as a *Bloch sphere* (Figure 2.1), a unit sphere, with the general state written

$$|\psi\rangle = \cos\theta\,|0\rangle + e^{i\phi}\sin\theta\,|1\rangle\,. \tag{2.27}$$

As in the classical case, operations can be done on qubits just like they were on bits. In the quantum computer, the second postulate will demand that the evolution is unitary. This is equivalent to demanding the norm should be conserved [7]. Because of the close relationship with electrical engineering, the operators will frequently be called *quantum gates* , or just gates, in this paper. Some of the most important gates will be:

The *Pauli spin-matricies*

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

From these we can create other important quantum gates such as [7]:

The *Hadamard Gate*

$$H = \frac{1}{\sqrt{2}}(\sigma_x + \sigma_z) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{2.28}$$

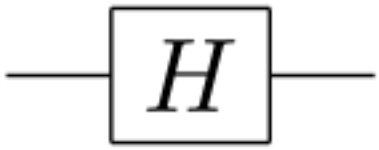

Figure 2.2: *Circuit representation of the Hadamard gate. [8]*

and the *Phase Shift Gate*:

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix}. \tag{2.29}$$

Notice that the Pauli Z-spin gate is the Phase shift gate with $k = 1$. Having set up a one qubit system, it is easy to create a multi-dimensional system with a tensor product of n separate qubit systems, as expected by the fourth postulate of quantum physics. Using the tensor product, the two qubits $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \in \mathcal{H}^\alpha$, and

$|\phi\rangle = \beta_0 |0\rangle + \beta_1 |1\rangle \in \mathcal{H}^\beta$.

$$|\psi\rangle \otimes |\phi\rangle = \begin{pmatrix}\alpha_0\\ \alpha_1\end{pmatrix} \otimes \begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix} = \begin{pmatrix}\alpha_0 |\phi\rangle\\ \alpha_1 |\phi\rangle\end{pmatrix} = \begin{pmatrix}\alpha_0\beta_0\\ \alpha_0\beta_1\\ \alpha_1\beta_0\\ \alpha_1\beta_1\end{pmatrix}. \tag{2.30}$$

In this system, the possible states we can measure are $|00\rangle, |01\rangle, |10\rangle,$ and $|11\rangle$, which are elements in the state space $\mathcal{H}^\alpha \otimes \mathcal{H}^\beta$. These states correspond to both qubits having spin down, the first having spin up and the second having spin down, and so forth. This gives us the general state $|\psi\rangle$ for the two-qubit system as [5]

$$|\psi\rangle = \tag{2.31}$$
$$\gamma_{00} |0\rangle \otimes |0\rangle + \gamma_{01} |0\rangle \otimes |1\rangle + \gamma_{10} |1\rangle \otimes |0\rangle + \gamma_{11} |1\rangle \otimes |1\rangle = \tag{2.32}$$
$$\gamma_{00} |00\rangle + \gamma_{01} |01\rangle + \gamma_{10} |10\rangle + \gamma_{11} |11\rangle. \tag{2.33}$$

It follows from the definition of the tensor product that γ_{ij} are normalized by

$$| \gamma_{00} |^2 + | \gamma_{01} |^2 + | \gamma_{10} |^2 + | \gamma_{11} |^2 = 1. \tag{2.34}$$

Having a system of two qubits, we can do measurement on the first qubit without affecting the second [3]. This will collapse the state-vector. The resulting vector will be a state vector $|\phi\rangle \in \mathcal{H}^\beta$. If measurement of the first qubit of $|\psi\rangle$ yields $|0\rangle$ then:

$$|\phi\rangle = \frac{\gamma_{00} |00\rangle + \gamma_{01} |01\rangle}{\sqrt{| \gamma_{00} |^2 + | \gamma_{01} |^2}}. \tag{2.35}$$

From this, generalization to a n-qubit system, the composite Hilbert space $\mathcal{H}^{\alpha_1} \otimes \mathcal{H}^{\alpha_2} \otimes \ldots \otimes \mathcal{H}^{\alpha_n}$, is completely analogous.

$$|\psi\rangle = \gamma_{00\ldots0} |00\ldots0\rangle + \gamma_{00\ldots1} |00\ldots1\rangle + \ldots + \gamma_{11\ldots1} |11\ldots1\rangle. \tag{2.36}$$

2.4.1 Controlled Gates

A special class of quantum gates important for the purpose of this paper are the so called controlled gates. These gates operate on two qubits, one of which is referred to as the target qubit and one called the control qubit. Controlled gates either operate on the target qubit or do nothing depending on the value of the control qubit. For example, the controlled not gate below flips the target qubit if the state of the control qubit is $|1\rangle$, and does nothing if the state is $|0\rangle$

$$CNOT = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}. [7] \tag{2.37}$$

The controlled phase shift gate, which either performs or does not perform the phase shift operation (see eq.(2.29)) is of particular interest in this paper and will be of use later when discussing the quantum analogy of the Discrete Fourier Transform.

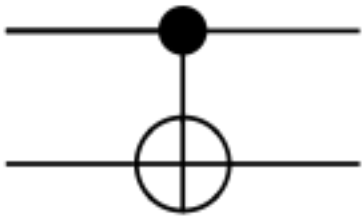

Figure 2.3: *Circuit representation of a CNOT-gate. The top line depicts the control qubit, with the gate acting on the target qubit. [9]*

2.4.2 Implementations

In this paper we discuss quantum circuits in an abstract sense and do not focus to much upon the technical details of how to construct them. We will however mention a few suggestions that have been made as regards to the actual construction of quantum gates.

Earlier we mentioned the possibility of using photon beams as our qubits. In this case a controlled not gate can be made with a polarization interferometer. Here the entire quantum circuit will consist of qubits in the form of beams of light being split and polarized in different sequences.

One such sequence is the one making up the Mach-Zender interferometer. This can be seen as a Hadamard gate, followed by a phase shift gate and then another Hadamard gate. [7]

Chapter 3

Investigation

3.1 Problem

The goal of the thesis is to answer the following questions:

> *Shor's algorithm can factorize integers in polynomial number of operations, using unitary quantum transformations. How does this algorithm work, which operations does it depend on and why does it require a quantum computer?*

We will try to answer these questions by studying the quantum Fourier transform and other parts of Shor's Algorithm and by attempting to simulate a quantum circuit on a classical computer to factorize small integers with said algorithm.

3.2 Model

We will represent a quantum computer with Q-qubits as a circuit of quantum gates (a *quantum circuit*). This will be viewed in formal mathematical manner. All gates will be represented by matricies, or drawn in a circuit. To understand quantum computational algorithms, it will sometimes be of benefit to see them as a circuit, and sometimes as we normally view an algorithm in classical computer science. We will switch between these views whenever its appropriate.

The crucial step in Shor's Algorithm, is the use of the Quantum Fourier transform (QFT). We construct it analogously to the Discrete Fourier transform, for a system of $Q = 2^n$ qubits as

$$|a\rangle \longrightarrow \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i a k/2^n} |k\rangle . \tag{3.1}$$

The circuit model for the QFT will be as depicted in Figure 3.1 [**?**]. The top line depicts operations done on the first qubit, the second on qubit two, and so on. The boxes depicts unitary transformations on the qubits, with H being a Hadamard-gate, and R_k being a Phase-shift gate. The proof for this circuits validity will be presented in section 3.3. Since the QFT is built up solely by unitary transforms, the QFT is unitary.

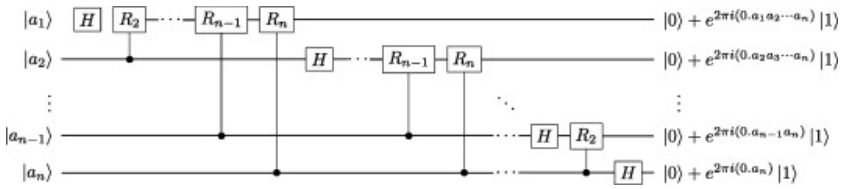

Figure 3.1: *A circuit model of the Quantum Fourier transform (QFT). The normalizing factor* $1/2^{n/2}$ *has been omitted.*

3.3 Analytical Calculations

In this section, we will present the analytical proof that Shor's algorithm factors integers faster than the known classical algorithms. We will first show how quickly the QFT can be computed with, and then show how the factoring problem will be solved using the QFT and its inverse.

The derivations below are largely based on those presented by M.Nielsen and I.Chuang in *Quantum Computation and Quantum Information*, [3].

3.3.1 Complexity of the QFT

The circuit model is justified here. We write our qubits in the computational basis, by doing a binomial expansion $|a\rangle = |a_1a_2\ldots a_n\rangle$. Performing the Hadamard gate (equation (2.28)) on the first qubit (top line in Figure 3.1)

$$H_1 |a_1a_2\ldots a_n\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.a_1}|1\rangle\right)|a_2\ldots a_n\rangle\,. \tag{3.2}$$

Following up with the controlled-R_2 gate, which produce the state

$$CR_{2,1}\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.a_1}|1\rangle\right)|a_2\ldots a_n\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.a_1a_2}|1\rangle\right)|a_2\ldots a_n\rangle\,. \tag{3.3}$$

Thus, after performing the series of controlled-R_k gates from 1 to n, we have obtained

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.a_1a_2\ldots a_n}|1\rangle\right)|a_2\ldots a_n\rangle\,. \tag{3.4}$$

Next we work on our second qubit, by first performing the Hadamard gate, followed up by the controlled R_k gates, where k range from 2 to n. The first qubit will be left unchanged, while using the same procedure as above

$$\begin{aligned}&\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.a_1a_2\ldots a_n}|1\rangle\right)|a_2\ldots a_n\rangle \longrightarrow \\ &\frac{1}{2}\left(|0\rangle + e^{2\pi i 0.a_1a_2\ldots a_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.a_2\ldots a_n}|1\rangle\right)|a_3\ldots a_n\rangle\,.\end{aligned} \tag{3.5}$$

Doing the same procedure recursively, the final state becomes

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i 0.a_1a_2\ldots a_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.a_2\ldots a_n}|1\rangle\right)\ldots\left(|0\rangle + e^{2\pi i 0.a_n}|1\rangle\right). \tag{3.6}$$

Lets us now instead start with the definition of the QFT. We will see that we will reach the same form as above.

$$
\begin{aligned}
|a\rangle \longrightarrow & \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i a k/2^n} |k\rangle = \\
& \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i a \left(\sum_{l=1}^{n} k_l 2^{-l}\right)} |k_1 \ldots k_n\rangle = \\
& \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i a k_l 2^{-l}} |k_l\rangle = \\
& \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left(\sum_{k_l=0}^{1} e^{2\pi i a k_l 2^{-l}} |k_l\rangle \right) = \\
& \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left(|0\rangle + e^{2\pi i a 2^{-l}} |1\rangle \right) = \\
& \frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i 0.a_1 a_2 \ldots a_n} |1\rangle \right) \left(|0\rangle + e^{2\pi i 0.a_2 \ldots a_n} |1\rangle \right) \ldots \left(|0\rangle + e^{2\pi i 0.a_n} |1\rangle \right). \qquad (3.7)
\end{aligned}
$$

To calculate the cost in bit operations of the QFT, we count the number of operations done on each qubit. On the first qubit we compute a Hadamard gate, followed by $n-1$ controlled-phase shifts. We follow up by doing a Hadamard, followed by $n-2$ controlled-phase shifts on the second qubit. The i:th qubit thus has a total of $n-i+1$ operations performed on it. The total number of operations is thus

$$\sum_{i=1}^{n} n - (i-1) = \sum_{i=0}^{n-1} n - i. \qquad (3.8)$$

The last sum is simply an arithmetic series

$$\sum_{i=0}^{n-1} n - i = \frac{n(n+1)}{2} = \frac{n^2+n}{2}, \qquad (3.9)$$

and here n^2 is the dominating term. The circuit performs the QFT for a n-bit integer at a cost of approximately $\Theta(n^2)$ bit operations.

3.3.2 Factoring

The algorithm for factorizing a composite number N of n-bits length is given by combining the two following theorems [3]:

Theorem 1. *Assume N is a $\log_2(N)$-bit composite number, and that x is a non-trivial solution to*

$$x^2 = 1 \ (mod\ N), \qquad (3.10)$$

such that $1 \leq x \leq N-1$, then at least one of gcd$(x-1, N)$ and gcd$(x+1, N)$ is a non-trivial factor of N, and can be computed in $\mathcal{O}(\log_2(N)^3)$ operations.

Theorem 2. *Assume $N = p_1^{\eta_1} p_2^{\eta_2} \dots p_m^{\eta_m}$ is the prime factorization of an odd composite positive integer. Let x be an integer chosen uniformly at random, with the requirement that $1 \leq x \leq N-1$ and x is co-prime to N. Let r be the order of x mod(N), then*

$$P(r \text{ is even and } x^{r/2} \neq -1 \pmod{N}) \geq 1 - \frac{1}{2^{m-1}} \tag{3.11}$$

We have introduced the concept of the *order* of an integer. We will find out what this is and how we find it in section 3.3.4.

The algorithm for factorizing a number N can be summarized as [3]:

Input A composite number N,

Output A non-trivial factor of N.

1. If N is even, return 2.
2. Check if $N = a^b$ for integers $a \geq 1$ and $b \geq 2$, and if so return a.
3. Choose a random $1 \leq x \leq N-1$. If $\gcd(x, N) > 1$ then return $\gcd(x, N)$.
4. Find the order r of $x \pmod{N}$.
5. If r is even and $x^{r/2} \neq -1 \pmod{N}$ then compute $\gcd(x^{r/2} \pm 1, N)$ and $\gcd(x^{r/2} + 1, N)$, and test if any of these are non-trivial factors of N and, if so, return it.
6. Repeat from step 3.

Theorem 1 will guarantee that if r is even, $\gcd(x^{r/2} - 1, N)$ or $\gcd(x^{r/2} + 1, N)$ will factor N. This will happen with probability greater than $\frac{1}{2}$ by Theorem 2. All steps, except step 4, can be computed on a classical computer in a polynomial number of operations, or less. The problem that is left for us is to find the order of x.

3.3.3 The Phase Estimation Subroutine

Before we continue to find the order of x, we observe another problem we will need in our final solution. Given two states $|\psi\rangle$ and $|\overline{\psi}\rangle$ such that

$$|\overline{\psi}\rangle = e^{2\pi i\phi} |\psi\rangle . \tag{3.12}$$

The two states will upon measurement be identical, since

$$\begin{aligned} &\left\langle \overline{\psi} \mid M_i^\dagger M_i \mid \overline{\psi} \right\rangle \\ =&\left\langle \psi \mid e^{-2\pi i\phi} M_i^\dagger M_i e^{2\pi i\phi} \mid \psi \right\rangle \\ =&\left\langle \psi \mid M_i^\dagger M_i \mid \psi \right\rangle . \end{aligned} \tag{3.13}$$

Suppose that U is a unitary operator, such that $|\overline{\psi}\rangle = U |\psi\rangle = e^{2\pi i\phi} |\psi\rangle$.

We want to find an algorithm which estimates this ϕ to arbitrary precision. We con-

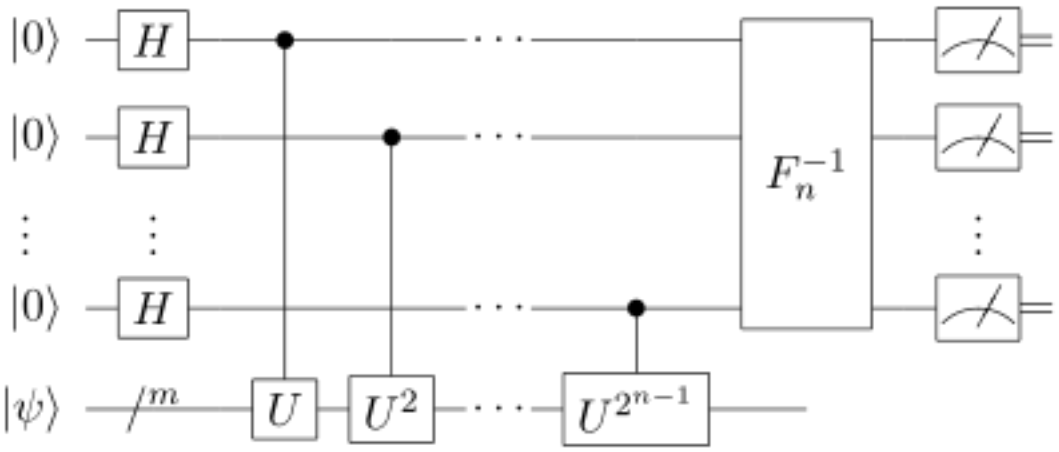

Figure 3.2: *The phase estimation subroutine, followed by the inverse QFT and measurement [10]*

struct the algorithm by setting up two separate registers. The first register has n-qubits, where n is determined by the precision we want to have on the phase estimation. We set all our qubits in the register to start in the state $|0\rangle$. The second register will store $|\psi\rangle$, and has m qubits, where m is the number of qubits needed to store all of $|\psi\rangle$. As shown in the circuit in Figure 3.2, we apply the Hadamard transform on all qubits in our first register, then apply U^{2^j} on the j:th qubit. Using the same type of calculations as we did in equation (3.6), we will get the output of the first register to be

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i 2^{n-1}\phi}|1\rangle\right)\left(|0\rangle + e^{2\pi i 2^{n-2}\phi}|1\rangle\right)\ldots\left(|0\rangle + e^{2\pi i\phi}|1\rangle\right)$$
$$\frac{1}{2^{n/2}}\sum_{j=0}^{2^n-1} e^{2\pi i j\phi}|j\rangle\,, \tag{3.14}$$

while the second register $|\psi\rangle$ will be unchanged [3]. If we use the *inverse QFT* on equation (3.14), denoted F_n^{-1} in Figure 3.2

$$\frac{1}{2^{n/2}}\sum_{j=0}^{2^n-1} e^{2\pi i j\phi}|j\rangle|\psi\rangle \underset{F_n^{-1}}{\longrightarrow} |\phi\rangle|\psi\rangle\,, \tag{3.15}$$

and then perform a measurement in the computational basis, we will obtain the phase $|\phi\rangle = |\phi_1\ldots\phi_n\rangle$. The inverse QFT is easily constructed by inverting each unitary operation in the QFT and applying them in reverse order.[3]

In the above calculation, we have assumed that ϕ can be written exactly as a binary expansion. In the general case it can be proven that the inverse QFT will provide an estimator $\widetilde{\phi}$ with accuracy of 2^{-k} and a success rate of $1-\epsilon$ by choosing

$$n = k + \left\lceil \log_2\left(2 + \frac{1}{2\epsilon}\right)\right\rceil \text{[3]}. \tag{3.16}$$

We have here not defined what the operators U are. The phase estimation subroutine is therefore not a complete quantum algorithm in itself, but can be included in a bigger algorithm.

3.3.4 Order-Finding

We will now finish our problem of factorizing our composite number. As was hinted in section 3.3.2, we need to find the order of our number x. The order is defined as the smallest r such that

$$x^r = 1 \ (\text{mod } N). \tag{3.17}$$

We will make use of the following theorem:

Theorem 3. *Let the ordered list of positive integers*

$$[s_0, s_1, ..., s_M] = s_0 + \cfrac{1}{s_1 + \cfrac{1}{s_2 + \cfrac{1}{... + \cfrac{1}{s_M}}}}, \tag{3.18}$$

be the continued fraction expansion. *Suppose p/q is a rational number such that*

$$| \frac{p}{q} - \phi | \leq \frac{1}{2q^2}. \tag{3.19}$$

Further on, suppose that ϕ is known up to $n = 2k+1$ bits, and that ϕ is a rational number. Then $\frac{p}{q}$ is a convergent of ϕ by the continued fraction expansion in $\mathcal{O}(k^3)$ operations. [3]

We will use the phase estimation subroutine to find the order, with our U as

$$U |y\rangle \equiv |xy \ (\text{mod } N)\rangle , \tag{3.20}$$

where $y = y_1 y_2 ... y_n$, and $y_i = \{0, 1\}$. We can disregard the cases when $N \leq y \leq 2^n - 1$ since then

$$xy = y \ (\text{mod } N), \tag{3.21}$$

and U will act trivially. For the cases $0 \leq y \leq N$, the eigenstates for U are for integers $0 \leq p \leq r - 1$:

$$|u_p\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i p k/r} \left| x^k \ (\text{mod } N) \right\rangle , \tag{3.22}$$

which we confirm with:

$$\begin{aligned} U |u_p\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i p k/r} \left| x^{k+1} \ (\text{mod } N) \right\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{k=1}^{r} e^{-2\pi i p (k-1)/r} \left| x^k \ (\text{mod } N) \right\rangle \\ &= \frac{1}{\sqrt{r}} e^{2\pi i p/r} \sum_{k=1}^{r} e^{-2\pi i p k/r} \left| x^k \ (\text{mod } N) \right\rangle \\ &= e^{2\pi i p/r} |u_p\rangle , \end{aligned} \tag{3.23}$$

where we used the fact that $e^{2\pi i r/r} |x^r \ (\text{mod } N)\rangle = e^{2\pi i 0/r} |x^0 \ (\text{mod } N)\rangle = |1\rangle$ in the last step. We might be discouraged by this, since the expression for $|u_p\rangle$ assumes r to be

known, however

$$\begin{aligned}
&\frac{1}{\sqrt{r}}\sum_{p=0}^{r-1}|u_p\rangle \\
=&\frac{1}{r}\sum_{p}^{r-1}\left(\sum_{k=0}^{r-1}e^{-2\pi ipk/r}\left|x^k \pmod N\right\rangle\right) \\
=&\begin{cases}|1\rangle & \text{if } k=0\\ \frac{1}{r}\sum_{k=0}^{r-1}\frac{\left(e^{-2\pi i0k/r}-e^{-2\pi irk/r}\right)}{1-r}\left|x^k \pmod N\right\rangle & \text{if } k\neq 0.\end{cases} \\
=&\delta_{0k}\left|1\right\rangle. \qquad (3.24)
\end{aligned}$$

Couple this with the procedure *modular exponentiation*, which is simply exponentiation over a modulus [3]. With this, coupled with Theorem 3 we can use the phase estimation subroutine to find the fraction p/r, and thus the order of r.

Using the phase estimation, we can with the result in eq. (3.24), we can just prepare our second register in the state $|1\rangle$ [3], and get a ϕ which is a good estimate to our p/r.

Inputs A unitary operation $U : |j\rangle|k\rangle \longrightarrow |j\rangle|x^j k \pmod N\rangle$, for $\gcd(x,N)=1$, and $n = 2k+1\left[\log_2\left(2+\frac{1}{2\epsilon}\right)\right]$ qubits initialized to $|0\rangle$, and k qubits initialized to $|1\rangle$.

Output The order r of $x \pmod N$.

1. Start with the two regiesters in their initial state $|0\rangle^n|1\rangle^k$.
2. Create a superposition $\frac{1}{2^{n/2}}\sum_{j=0}^{2^n-1}|j\rangle|1\rangle$ by a series of Hadamard gates.
3. Apply the Unitary operation U to create $\frac{1}{2^{n/2}}\sum_{j=0}^{2^n-1}|j\rangle|x^j \pmod N\rangle \approx \frac{1}{\sqrt{r2^n}}\sum_{p=0}^{r-1}\sum_{j=0}^{2^n-1}e^{2\pi ipj/r}|j\rangle|u_p\rangle$.
4. Apply the inverse Fourier transform to get $\frac{1}{r^{1/2}}\sum_{p=0}^{r-1}\left|\widetilde{p/r}\right\rangle|u_p\rangle$.
5. Measure the first register to get $\widetilde{p/r}$.
6. Apply the continued fraction algorithm to get r.

Using this order finding algorithm as our step 4 in our factoring algorithm, we have a complete factorization algorithm running on a quantum computer. Even though much focus is put on the QFT and its inverse, the most computationally demanding part is the *modular exponentiation*. The modular exponentiation operates in $\mathcal{O}(n^3)$, much higher than the QFT's $\Theta(n^2)$.

3.4 Numerical Analysis

As part of our work we have attempted to simulate a quantum circuit on a classical computer. This was done by writing an object oriented program in C++. Below follows a rough outline of how this program works and what we accomplished with it.

A desired N-qubit system is represented by a set of pairs of complex numbers. Each qubit is stored as an individual state vector, with coefficients representing the probability of finding that particulair qubit in a certain state regardless of the state of the rest of the qubits. Thus the state vector for the entire system, given by the system described in equation (2.36), is not formulated by default.

The decision for using this approach is based on the notion of seeing the whole system as a circuit. The quantum gates utilized in for example the QFT, as shown in figure 3.1, exclusively represent operations on one or two qubits. By storing each qubit as an individual state vector, these operations can be preformed with the simple matrices shown in section 2.4 rather than having to develop general versions depending on the number of qubits. In the system represented by equation 2.33, operations are of course computed for the relevant qubits when for example applying the controlled phase shift gate.

Storing each qubit individually contributes with both technical simplicity and easier code overview. The unitary transforms are represented by functions taking pairs of complex numbers as arguments, preforming the appropriate mathematics and returning pairs of complex numbers. In this way we have written a function preforming the operations corresponding to the Hadamard gate on a qubit, another one for the controlled phase shift gate and so on.

As can be seen in the UML diagram in figure 3.4, the simulation is handled by three objects. The quantum circuit objects holds the states of all qubits, and it is from here that the quantum circuits are initiated. A call is made to the quantum computer object to perform whatever quantum algorithm is necessary (observe that the names of the objects are arbitrary and are used for code structure purposes only. That those two objects are called quantum circuit and quantum computer has nothing whatsoever to do with how an actual physical implementation would be structured.).

The quantum computer object holds information about how the algorithms, such as for example the QFC, are performed. These are stored as sequences of calls to unitary quantum transformations, which are performed on the qubits in the quantum circuit object. In this object we have a function for performing the QFC, and one for performing the inverse QFC. When the QFC function in the quantum computer object is called it in turn calls on functions in the unitary transformer object to act upon the the qubits in the quantum circuit, in the order indicated by figure 3.1.

The unitary transformer contains functions corresponding to the Hadamard gate and various phase shift gates. Also stored here are functions for performing base shifts on pairs of qubits to the form given by equation (2.36) and back to individual qubits. These are used when performing controlled operations.

In this way we have created virtual circuits capable of performing the QFT and the inverse QFT on a set of qubits.

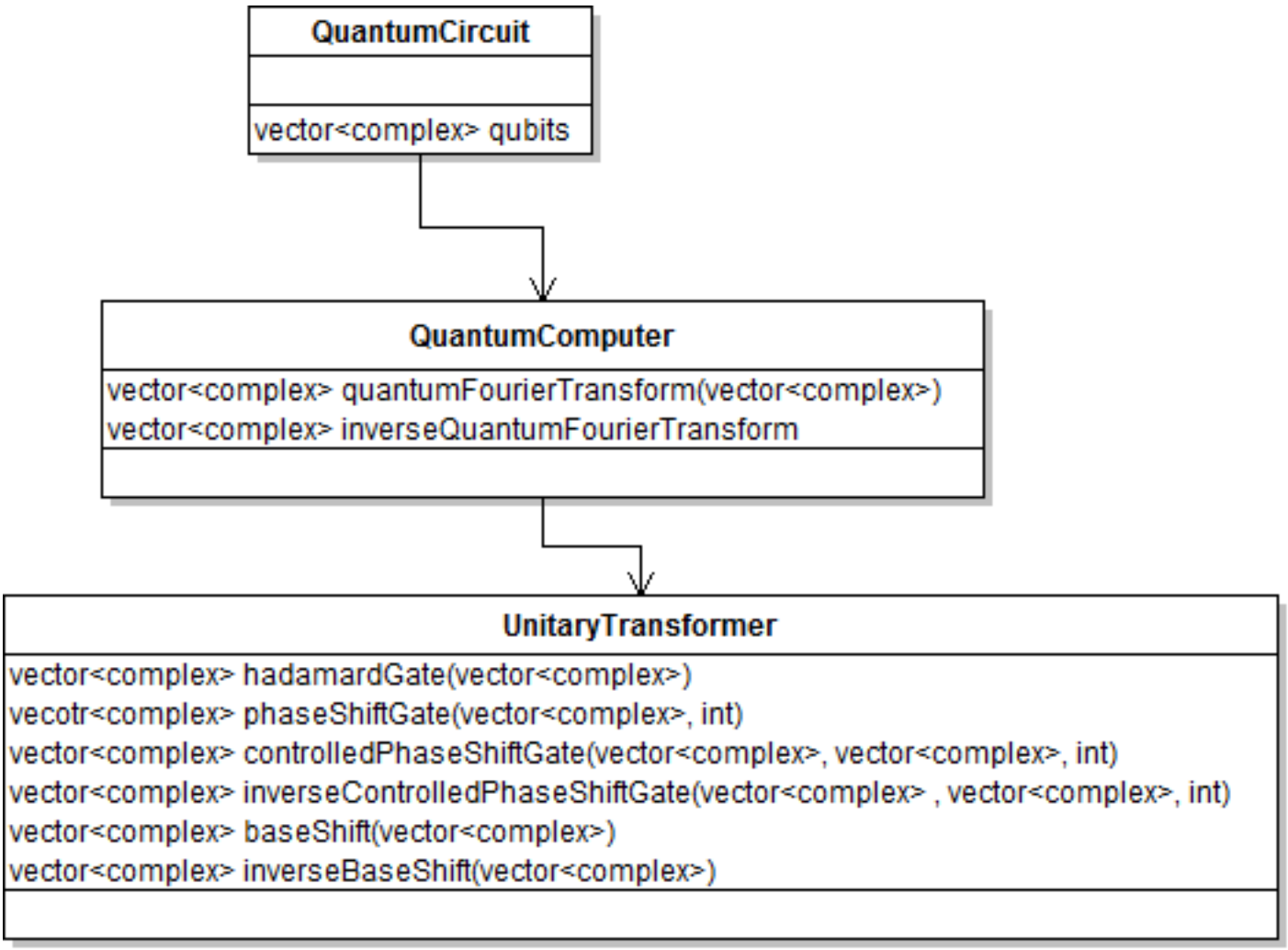

Figure 3.3: *A rough UML diagram for the simulation. All functions are expressed in the following syntax: returned datatype, function name, argument datatype.*

3.5 Results

The most time consuming part of Shor's algorithm is the modular exponentiation. The cost in bit operations of this part, as a function of the bit-size of the integer to be factorized, grows like $\mathcal{O}(k^3)$. This means that for large integers it is much more costly than the QFT, for which the required number of bit operations only grows like $\Theta(k^2)$.

Note that the algorithm may fail with a probability of ϵ to find a ϕ in the phase estimation which is a good estimator to p/r. However, as explained in section 3.3.3, this ϵ can be made neglible by having enough qubits in the first register. The second problem is if $\gcd(p, r) \neq 1$. However, this problem can be solved by doing the phase estimation twice, which will only increase the complexity with a constant number of operations. [3].

In addition to these analytic results we have managed to simulate quantum circuits performing unitary operations on qubits. We have combined such operations into sequences which performe the QFT, and the inverse QFT, on a number of qubits. This has

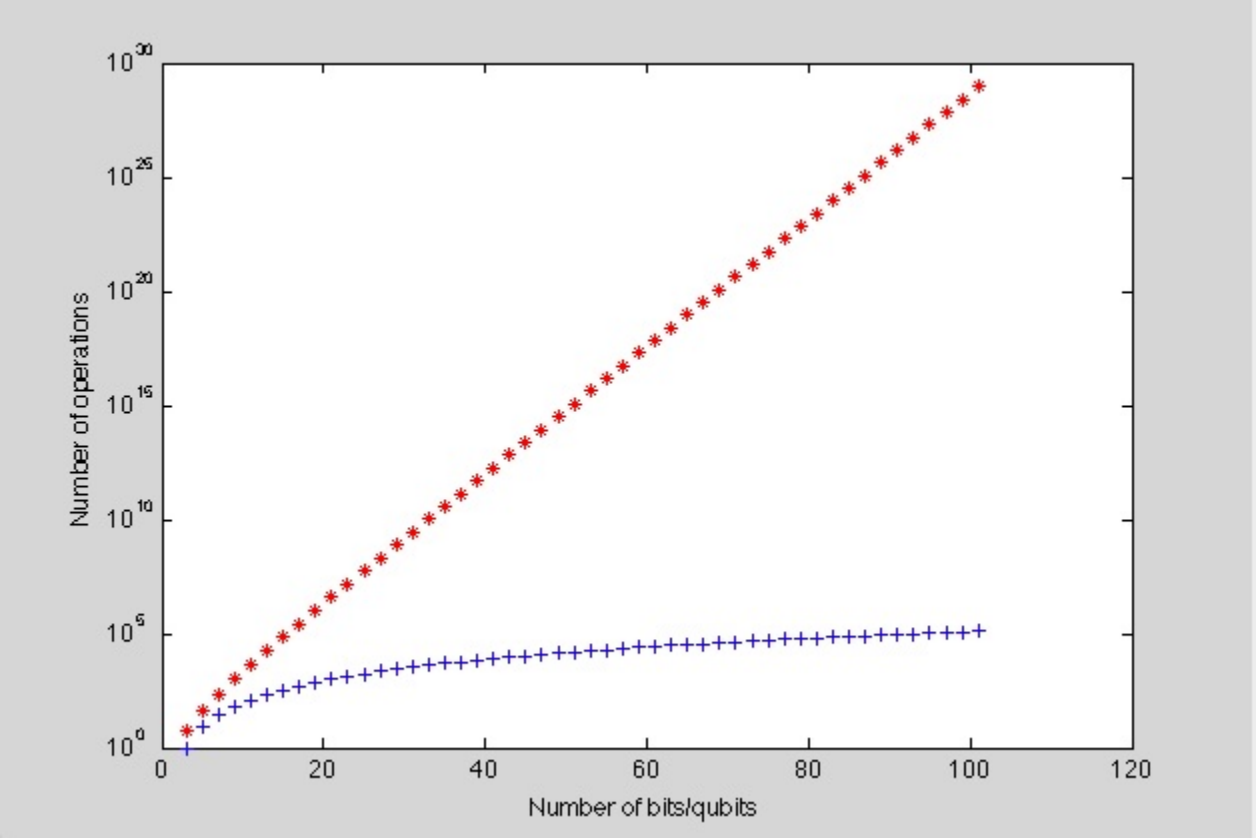

Figure 3.4: *Comparison of the number of operations required to factorize an integer of n bits, as functions of n with the quantum algorithm in blue, and the classical in red.* ***Observe*** *The scale is logarithmic on the y-axis*

given further insight into the circuit model of the QFT, and demonstrated the simplicity of said model.

3.6 Discussion

As we saw in section 2.3.2 there exists obvious applications of efficient factorization algorithms within the area of cryptography. Shor's algorithm is one such efficient factorization algorithm, as we showed in Figure 3.5. That plot clearly shows that when the integer one wishes to factorize grows today's best factorization algorithms will eventually become obsolete. Shor's algorithm however, when implemented on a quantum computer, will acording to the same plot be extremely efficient.

In this paper, we have presented the quantum mechanics from the perspective of having our quantum systems expressed with *pure states*. A generalization of quantum systems is to observe the system as being in a statistical ensemble of these states (a *mixed state*). There is ongoing research regarding how these mixed states are used to increase the efficiency of the quantum computer.[11][12]

Realisation of an actual quantum computer that can run Shor's algorithm for small composite integers have been made [13]. A great question for the realization for large

quantum computers that can perform a multitude of different tasks is what techniques and materials that can store large registers of qubits.

Chapter 4

Summary and Conclusions

We set out to answer the question of how Shor's algorithm can factorize integers in polynomial number of operations using unitary quantum transformations, how this algorithm works, which operations it depends on and why it requires a quantum computer.

The central part of the algorithm is the QFT. The algorithm operates on quantum states and is able to solve *the discrete logarithm problem*, a traditionally complex problem. The ability of the quantum computer to superposition quantum states results in the algorithm requiring far less operations to perform. Beyond the scope of this paper one can further analyse how quantum effects such as *quantum entanglement* plays a role in this.

We were able to emulate a quantum circuit on a classical computer. For this purpose our representation of quantum algorithms as circuits was essential. The simulation of quantum circuits helped us with gaining greater understanding of how one by combining unitary quantum transformations can perform more complex algorithms in an efficient manner utilizing circuit methods.

Our virtual quantum circuit performs the QFT and the inverse QFT, but as can be seen in section 3.3, Shor's algorithm consists of a few more steps. A natural extension of our work could be to construct a program that performs these steps as well, thereby completing a circuit performing Shor's algorithm.

Bibliography

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring", *Proc. 35nd Annual Symposium on Foundations of Computer Science* (Shafi Goldwasser, ed.), IEEE Computer Society Press (1994), 124-134.

[2] S. Gasiorowicz, *Quantum Physics, 3:rd Edition*, John Wiley & Sons Inc., (2003).

[3] M. Nielsen, I. Chuang, *Quantum Computation and Quantum Information, 1:st Edition*, Camebridge University Press, (2000).

[4] A.Björner, "Kryptografi och primalitet", *Institute of Mathematics, Royal Institute of Technology*, `http://www.math.kth.se/math/GRU/2009.2010/SF1630/TFYS/`, (2012-04-20) 17:45 CEST.

[5] E.Gerjouy, "Shor's factoring algorithm and modern cryptography. An illustration of the capabilities inherent in quantum computers", *Am. J. Phys.* **73** (6), 521 (2005).

[6] "Quantiki.org", `http://www.quantiki.org/wiki/File:Bloch.png`, (2012-04-22, 18:15 CEST).

[7] S. Auletta, M. Fortunato, G. Parisi, *Quantum Mechanics, 1:st Edition*, Cambridge University Press, (2009).

[8] "Wikipedia.org", `http://en.wikipedia.org/wiki/File:Hadamard_gate.svg`, (2012-04-22, 19:00 CEST).

[9] "Wikipedia.org", `http://en.wikipedia.org/wiki/File:CNOT_gate.svg`, (2012-04-22, 19:00 CEST).

[10] "Wikipedia.org", `http://en.wikipedia.org/wiki/File:Quantum_phase_estimation.svg`, (2012-04-21, 18:30 CEST).

[11] P. Zuliani, "Quantum programming with mixed states", *Electronic Notes in Theoretical Computer Science (ENTCS)*,**(170)**, (2007), 185-199.

[12] M. Siomau and S. Fritzhe, "Quantum computing with mixed states", *European Physical Journal D*, **62**, 449 (2011).

[13] L. Vandersypen†, M. Steffen, G. Breyta†, C. Yannoni†, M. Sherwood and I. Chuang, "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance", *Nature*, 414, 883-887 (20/27 Dec 2001)