An Introduction to Quantum Computing and Quantum Error-Correction

Charles A. Cusack

November 1999

1 Introduction and Preliminaries

1.1 Introduction to Quantum Computing

Quantum computing was hinted at by Feynman [27, 26] in 1982, and made more formal by David Deutsch in 1985 [18]. It was thought that perhaps quantum computers could perform certain operations faster than any classical[1] computer, but nothing concrete was known, so for a decade progress in the field was slow. Then in 1994, Peter Shor gave a polynomial-time quantum algorithm for factoring integers [49], a problem for which the best known classical algorithm has exponential complexity. Later came Grover's search algorithm [32, 34], which is capable of unstructured search in time $O(\sqrt{N})$, whereas the best possible classical algorithm has complexity $\Theta(N)$. This was the first real "proof" that a quantum computer is a more powerful device than a classical computer, at least in some sense.

In the 5 years since Shor's result, much has been done. The field of quantum computing is very broad, with research being done on algorithms, complexity theory, cryptography, information theory, error-correcting codes, and physical implementation. A major thrust of the work has been in the area of quantum error correction. The reason for this is the fact that quantum states are very susceptible to inaccuracy and *decoherence*. Without the use of error-correcting codes, quantum computation will be all but impossible. In fact, because of the no-cloning theorem [58], it was thought that perhaps error correction would be impossible. The plethora of papers on quantum error-correcting codes (see [14, 31, 15, 48, 47, 54, 52, 13, 41, 29, 30, 31, 42, 43], for a few) show that nothing could be further from the truth.

[1]In this paper, the term classical, when applied to compters and algorithms, means non-quantum, non-relativistic. In other words, present-day computers and the algorithms which operate on them.

Most researchers are optimistic about the future of quantum computing, although they differ on the question of *when*. Some researchers think we will see a fully functional quantum computer within 10 years, while the more realistic estimate seems to be that it will be at least 30 years, and perhaps longer.

In this paper, I attempt to give a very brief introduction to quantum computing. As a mathematician, I feel the plight of those non-physicists trying to wade through the quantum mechanics to understand the concepts involved. In light of this, I have attempted to make this paper as physics-free as possible. My hope is that it doesn't oversimplify too much. Those interested in a more in-depth introduction should consult one of [2, 45, 53, 56, 57]. For those with a good physics background, the much heftier [44] may be appropriate.

I will proceed as follows. In Section 1.2 I will define the basic unit of storage in a quantum computer, the *qubit*. I will then talk about multi-qubit systems, or *registers*. The idea of *entanglement* will be briefly discussed. In Section 1.3 I will talk about the quantum analog to the classical gate logic. I will discuss some of the key operations that are performed on qubits and registers.

In Chapter 2 I will discuss some of the "neat" properties of quantum systems, and one limitation.

Chapter 3 will deal with the two "breakthrough" quantum algorithms: Shor's factoring algorithm (Section 3.1) and Grover's search algorithm (Section 3.2).

Chapter 4 will discuss the idea of quantum error-correcting codes.

1.2 Qubits and Quantum Registers

The study of quantum computing involves physics, mathematics, and computer science. Most researchers have a lack of knowledge in at least one of these fields, making it difficult to write about the subject to a general audience. Most of the introductions on quantum computing [2, 53, 11, 56, 9] are written from the perspective of a physicist, making it difficult for mathematicians and computer scientists to understand the subject. Two references do a better job: [57] is a good non-technical introduction, and [45] is a good but more complete introduction for non-physicists.

The approach I will take in this subsection is perhaps a little non-standard. It is an attempt to make the concepts of quantum computing understandable with very little knowledge of quantum mechanics.

In a classical computer, data are stored as bits in a two-state system. Thus, each bit of data can be in either state '0' or state '1'. A collection of n bits is called a register. One can think of the possible states of a register as lying in $\mathbb{Z}_2^n$. The operations which a classical computer performs can be thought of as mappings from $\mathbb{Z}_2^n$ to $\mathbb{Z}_2^n$.

The situation in quantum computing is quite different. A *quantum bit*, or *qubit*, is a quantum two-level system which can be in state '0' or '1', or a *superposition* of both. That is, a qubit can be in state $a \cdot 0 + b \cdot 1$, where a and b are complex numbers satisfying $|a|^2 + |b|^2 = 1$. From the physical point of view, a qubit can be any quantum two-level system, such as a spin-1/2 particle. We shall not concern ourselves with the *hows* of implementation qubits. Instead we will view the qubits, and the operations performed on them, in terms of mathematics. There are a few difficulties in viewing quantum computers in this way. We shall discuss these as we come to them.

The state of a qubit can be thought of as a unit vector in a complex Hilbert space of dimension 2. The state of an n-qubit *quantum register* can be thought of as a unit vector in a Hilbert space of dimension 2^n.

Before we go further, we shall look more closely at the complex Hilbert spaces of dimension 2^n.

Let $\mathcal{H}_1$ be a 2-dimensional complex Hilbert space, and $\{(1,0)^T, (0,1)^T\}$ a basis for $\mathcal{H}_1$. A generic vector in $\mathcal{H}_1$ is of the form

$$a\begin{pmatrix}1\\0\end{pmatrix} + b\begin{pmatrix}0\\1\end{pmatrix},$$

where a and b are complex numbers with $|a|^2 + |b|^2 = 1$.

Let $\mathcal{H}_n$ be a complex Hilbert space of dimension 2^n. We can define $\mathcal{H}_n$ recursively by

$$\mathcal{H}_n = \mathcal{H}_{n-1} \otimes \mathcal{H}_1,$$

where $\otimes$ is the *tensor product*.

Thus, a basis for $\mathcal{H}_2$ is

$$\left\{\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix},\begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}\right\}$$

$$=\left\{\begin{pmatrix}1\\0\\0\\0\end{pmatrix},\begin{pmatrix}0\\1\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\1\\0\end{pmatrix},\begin{pmatrix}0\\0\\0\\1\end{pmatrix}\right\},$$

and a generic vector in $\mathcal{H}_2$ is written as

$$a_0\begin{pmatrix}1\\0\\0\\0\end{pmatrix}+a_1\begin{pmatrix}0\\1\\0\\0\end{pmatrix}+a_2\begin{pmatrix}0\\0\\1\\0\end{pmatrix}+a_3\begin{pmatrix}0\\0\\0\\1\end{pmatrix},$$

where $|a_0|^2+|a_1|^2+|a_2|^2+|a_3|^2=1$.

There is a compact notation, called *ket notation*, to write vectors in a complex Hilbert space, developed by Dirac [21]. In addition to being more compact, it is much more suited for discussion of quantum states, as we will see. In $\mathcal{H}_1$, we write

$$|0\rangle=\begin{pmatrix}1\\0\end{pmatrix}\text{ and }|1\rangle=\begin{pmatrix}0\\1\end{pmatrix}.$$

In $\mathcal{H}_2$, we write

$$|00\rangle=|0\rangle\otimes|0\rangle=\begin{pmatrix}1\\0\\0\\0\end{pmatrix},\ |01\rangle=|0\rangle\otimes|1\rangle=\begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

$$|10\rangle=|1\rangle\otimes|0\rangle=\begin{pmatrix}0\\0\\1\\0\end{pmatrix},\quad\text{and }|11\rangle=|1\rangle\otimes|1\rangle=\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Thus, a basis for $\mathcal{H}_n$ is given by

$$\{|0\rangle,|1\rangle,\ldots,|2^n-1\rangle\},$$

where the numbers in the kets are thought of in terms of their binary expansions. Finally, we can write a vector in $\mathcal{H}_n$ as

$$\sum_{i=0}^{2^n-1}a_i|i\rangle,$$

where each of the a_i are complex numbers, and $\sum_{i=0}^{2^n-1} |a_i|^2 = 1$.

At times we will use the shorthand $|a\rangle|b\rangle = |a\rangle \otimes |b\rangle$.

We use a complex Hilbert space to represent the states of a n-qubit register because n qubits can be in a superposition of 2^n states. Recall that a classical register can store just one of the possible 2^n states. Thus, it seems that a quantum register can store exponentially more information than a classical one.

Unfortunately, things are not as nice as I have described so far. Although it is true that a quantum register can store a superposition of 2^n states, the state of a quantum register is not known unless we make a *measurement* of the register. When we make a measurement, the state of the register is *collapsed* into one of the base states, and which one it collapsed to is known to us. The probability that the state collapses to $|k\rangle$ is $|a_k|^2$. In the measurement process, the original state of the register is destroyed, and can not be reconstructed.

It is important to understand that this is not the same thing as we have with a classical probabilistic computer. In the case of probabilistic computer, at each step only 1 of the possible states is stored. Which one depends on the probability distribution. With a quantum computer, we can actually store *all* of the possible states and operate on them all *at the same time.* It is only when a measurement is performed that the state is reduced to one of the base states, which we can view as a classical state.

Example 1.1 *Consider a qubit in state* $1/\sqrt{2}(|0\rangle + |1\rangle)$. *If we measure the qubit, it will collapse to state* $|0\rangle$ *with probability 1/2, and to state* $|1\rangle$ *with probability 1/2. If we measure the state again, we will get the same result, because the state has collapsed to the state which we measured it to be in.*

Another difference between quantum and classical registers is the fact that two (or more) bits in a quantum register can be *entangled.* This is best illustrated with an example.

Example 1.2 *Consider a 2-qubit register in state* $1/\sqrt{2}(|00\rangle + |11\rangle)$. *Two particles in this state are called an* EPR pair *(See [23, 6, 26] for the history of the EPR* *paradox.). If we measure the register, it will collapse to state* $|00\rangle$ *with probability 1/2, and to state* $|11\rangle$ *with probability 1/2. In fact, if we just measure the first qubit of the register, it will have the same result.*

The two qubits in this register are said to be *entangled* because the value of one is not independent of the other. If we measure one of the qubits, it effects the other one. Essentially, a collection of qubits is entangled if it cannot be expressed as a tensor product of single qubits. It is not hard to see that we can't express $1/\sqrt{2}(|00\rangle + |11\rangle)$ as a tensor product.

Example 1.3 *A register in the state* $1/\sqrt{2}(|00\rangle + |01\rangle)$ *is not entangled, since it can be written as* $|0\rangle(1/\sqrt{2}(|0\rangle + |1\rangle))$.

It is entanglement that gives quantum computing an edge over classical computing. We will see two examples of how entanglement can be used to speed up classical algorithms in Chapter 3.

1.3 Quantum Transformations

Now that we have an understanding of how data is stored (at least theoretically) in a quantum computer, we need to discuss how it can be processed. As in classical computing, there are many computational models for quantum computing. The most popular model for quantum computation is the notion of quantum gates. I will not discuss quantum gates in detail. The interested reader can consult one of many papers dealing with this issue [3, 51, 20, 22, 19, 55, 4, 1]. Since the idea of quantum gates is an extension of the idea of reversible logic, Bennet's paper on reversible logic [8] may also be of interest.

Quantum registers are transformed by *unitary transformations*. A matrix U is unitary if $UU^\dagger = U^\dagger U = 1$, where $U^\dagger$ is the conjugate transpose of U. It is an important fact that unitary transformations are linear. In fact, unitary transformations are the linear transformations which map unit vectors to unit vectors.

We can think of the unitary transformations as gates, much like classical boolean logic gates. They have inputs and outputs, the outputs depending on the inputs and the type of gate. There are several very important differences, however.

First, since we are dealing with qubits, not boolean bits, the gates don't have outputs restricted to '0' and '1'. In fact, the outputs of a gate can be in any superposition, and can, more importantly, introduce entanglement between two or more qubits.

Second, unitary transformations are linear, so a transformation is described fully by its effect on the basis states. This makes it easy to describe what logic gates will do, but also results in the non-cloning theorem (See Section 2.1). In addition, it means that quantum gates operate on each basis state of a qubit independently. In other words, we can compute a function on many inputs at the same time. As stated previously, we can only read one of the outputs. We shall see in Chapter 3 how it is still possible to exploit the extra computation.

Third, since quantum transformations are unitary, the quantum gates must be reversible. Thus means that given the outputs, the inputs should be determined uniquely. One obvious implication of this is that the number of outputs must be the same as the number of inputs. Most quantum computations are thought of in the following framework: If I wish to compute the function f on an input x, I transform the state $|x\rangle|0\rangle$ to $|x\rangle|f(x)\rangle$. Since I have saved the input x, the computation can be reversed.

An important consideration when talking about quantum gates is whether or not there is an efficient implementation of the gate. For instance, the quantum Fourier transform in Example 1.8 below can be implemented efficiently in base q, if q has small prime power factors [24], but may not be efficiently computable for all bases q.

I will conclude this subsection with examples of common gates used in quantum computation.

Example 1.4 *Below is a list of transformations on single qubits. Because the transformations are linear, we need only show their effect on basis states. We also give the matrix associated with each.*

Identity	$I:$	$\begin{array}{lcl}\|0\rangle & \mapsto & \|0\rangle \\ \|1\rangle & \mapsto & \|1\rangle\end{array}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
NOT (negation)	$X:$	$\begin{array}{lcl}\|0\rangle & \mapsto & \|1\rangle \\ \|1\rangle & \mapsto & \|0\rangle\end{array}$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
phase shift	$Z:$	$\begin{array}{lcl}\|0\rangle & \mapsto & \|0\rangle \\ \|1\rangle & \mapsto & -\|1\rangle\end{array}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
phase shift and negation (ZX)	$Y:$	$\begin{array}{lcl}\|0\rangle & \mapsto & \|1\rangle \\ \|1\rangle & \mapsto & -\|0\rangle\end{array}$	$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$

These are often referred to as the Pauli matrices. *These transformations are important for the study of quantum error-correcting codes.*

Example 1.5 *The Hadamard transformation is the single-bit transformation defined by*

$$H: \begin{array}{lcl} |0\rangle & \mapsto & \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle & \mapsto & \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{array} \quad \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Example 1.6 *The* controlled-NOT *gate, C_{not} negates the second bit if the first bit is a '1', and does nothing otherwise:*

$$C_{not}: \begin{array}{lcl} |00\rangle & \mapsto & |00\rangle \\ |01\rangle & \mapsto & |01\rangle \\ |10\rangle & \mapsto & |11\rangle \\ |11\rangle & \mapsto & |10\rangle \end{array} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Example 1.7 *The Walsh or Walsh-Hadamard transformation, which performs the Hadamard transform on each qubit of a quantum register, can be defined recursively as follows:*

$$W_1 = H, \quad W_n = H \otimes W_{n-1}.$$

Notice that when applied to an n bit register whose qubits are each initially in state $|0\rangle$, it results in an equal superposition of all possible states:

$$\begin{aligned} W_n|0\rangle &= (H \otimes H \otimes \cdots \otimes H)(|0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle) \\ &= H|0\rangle \otimes H|0\rangle \otimes \cdots \otimes H|0\rangle \end{aligned}$$

$$= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \cdots \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$= \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle.$$

The last example we give will be useful when we discuss Shor's factoring algorithm.

Example 1.8 *The quantum Fourier transform with base 2^n is given by*

$$U_{QFT} : |a\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{c=0}^{2^n-1} e^{\frac{2\pi i c a}{2^n}} |c\rangle.$$

Like the discrete Fourier transform, the quantum Fourier transform maps from the time domain to the frequency domain. Thus, it maps a function of period r to a function which has non-zero values only at multiples of $1/r$. We will discuss this more with respect to Shor's algorithm. There are many papers that discuss the quantum Fourier transform in detail [24, 25, 49, 36, 38].

2 Quantum Effects and Limitations

In this section I will very briefly survey a few of the interesting abilities and limitations of quantum effects. Most of the details will be omitted, but can be found in various places, including [53, 45, 57]. I will not discuss several interesting topics, including random number generation, quantum cryptography and key distribution. See [57, 53] for more on these topics.

As is standard in the literature, I will refer to two individuals who wish to communicate as Alice and Bob.

2.1 No Cloning

An important question that needs to be addressed is whether or not it is possible to make a copy of a quantum register. That is, given a quantum register in state $|\phi\rangle$, can we make a copy of this state in another register, resulting in two registers with state $|\phi\rangle|\phi\rangle$. This is referred to as *cloning* a state.

If the state of a register is known, there are methods for cloning it. Given an unknown quantum state, there is certainly a transformation that will clone it, but the transformation may be unknown.

More generally, is there a single tranformation that will clone every quantum state? That this cannot be done was shown by Wootters and Zurek [58]. The proof given below is an adaptation of the proofs in [45, 53].

Theorem 2.1 *An unknown quantum state cannot be cloned.*

Proof. Let U be a unitary transformation that can clone arbitrary states. Then $U(|a\rangle|0\rangle) = |a\rangle|a\rangle$, and $U(|b\rangle|0\rangle) = |b\rangle|b\rangle$, where $a \neq b$. By linearity,

$$\begin{aligned} U(1/\sqrt{2}(|a\rangle + |b\rangle)|0\rangle) &= U(1/\sqrt{2}|a\rangle|0\rangle) + U(1/\sqrt{2}|b\rangle|0\rangle) \\ &= 1/\sqrt{2}(|a\rangle|a\rangle + |b\rangle|b\rangle) \\ &\neq (1/\sqrt{2}(|a\rangle + |b\rangle)(1/\sqrt{2}(|a\rangle + |b\rangle) \end{aligned}$$

Thus, U cannot clone $1/\sqrt{2}(|a\rangle + |b\rangle)$. Thus, the theorem is proven. □

We will discuss the implications of the no-cloning theorem again in Chapter 4.

2.2 Dense Coding

If Alice and Bob each have half of an EPR pair, then Alice can send Bob only one qubit q, and communicate two classical bits of information to Bob. Alice accomplishes this by entangling q with her half of the EPR pair in such a way that Bob can, upon receiving q, perform operations on q and his half of the EPR pair to determine which bits Alice wants to communicate. The EPR pair is destroyed in the process.

This is referred to as *dense coding*, since 2 bits of information are communicated by sending only 1 qubit. Of course, Alice and Bob had to somehow each obtain half of an EPR pair, so in some sense, one bit was already exchanged. The advantage here is that the EPR pair can be exchanged when "rates are cheap", and can be used later when "rates are expensive" to communicate 2 bits by sending only one.

2.3 Teleportation

As with dense coding, here we assume Alice and Bob share and EPR pair. Teleportation is a method that Alice can use to transmit the state of a quantum particle q to Bob using only classical communication. Again, she entangles q with her half of the EPR pair. She then sends classical information that Bob can use to determine what transformation(s) to perform on his half of the EPR pair in order to transform it to state q.

Since quantum data cannot be cloned (See Section 2.1), it is clear that the state of Alice's qubit will be destroyed in the process. The EPR pair will also be destroyed in the process.

3 Quantum Algorithms

Some researchers [17] have argued that most currently known quantum algorithms are, at least in some sense, the same. Most known quantum algorithms start by placing the register in an equal superposition of all states, compute a function on the inputs, perform a quantum Fourier transform, and finish with a measurement. This is true of Simon's algorithm [50], Shor's factoring and discrete logarithm algorithms [49], Kitaev's algorithm for the Abelian stabilser problem [40], and several others.

The next two sections outline the two most celebrated quantum algorithms – Shor's factoring algorithm and Grover's search algorithm.

3.1 Shor's Algorithm

Since the supposed intractability of factoring is the basis for cryptosystems like RSA, a polynomial-time quantum factoring algorithm is of interest to many people. A detailed explanation of Shor's factoring algorithm can be found in either Shor's original paper [49], or the slightly more accessible paper by Ekert and Jozsa [24]. Several of the introductions to quantum computing [45, 56, 53, 57] give shorter explanations of varying degrees of detail.

A well known probabalistic classical algorithm to find factors of a number N goes as follows:

1. Pick a random number y such that $0 < y < N$. If $gcd(y, N) \neq 1$, we have found our factor, and we can quit. Otherwise, proceed.

2. Find the order of y modulo N. That is, the smallest number a such that $y^a \equiv 1 \bmod N$. If a is odd, go back to step 1. Otherwise, continue.

3. Notice that since a is even,

$$y^a \equiv 1 \bmod N \Longleftrightarrow y^a - 1 \equiv 0 \bmod N \Longleftrightarrow (y^{a/2} - 1)(y^{a/2} + 1) \equiv 0 \bmod N$$

   Thus, as long as $y^{a/2} \neq \pm 1 \bmod N$, then $gcd(y^{a/2} \pm 1, N)$ are non-trivial factors of N. If $y^{a/2} = \pm 1 \bmod N$, go back to step 1.

The probability that a random integer y has even order modulo N and that $y^{a/2} \neq \pm 1 \bmod N$ is at least 1/2. Thus, this algorithm does not have to be repeated too many times before one expects to find a non-trivial factor. The problem with this algorithm classically is that there is no known efficient (polynomial-time) algorithm to find the order of an element modulo N (Step 2).

Shor's algorithm isn't actually a factoring algorithm. It is a polynomial-time quantum algorithm for finding the order of an element modulo N. Thus, the actual algorithm is the above with Shor's algorithm as Step 2.

Here is how Shor's algorithm works.

1. Given the number N to factor, choose a number $q = 2^b$ such that $N^2 \leq q < 2N^2$. The choice of q a power of 2 assures that the quantum Fourier transform can be computed efficiently. The fact that we choose $q \geq N^2$ is for more subtle reasons dealing with the accuracy of the quantum operations.

2. Choose at random an integer x, $0 < x < N$, such that $gcd(x, N) = 1$. This will be the element whose order we will find.

3. We start with $2m$ qubits, which we will think of as two registers a and b. Both a and b will start out in state $|0\rangle$. Thus, the initial state is

$$|0\rangle|0\rangle.$$

4. Apply W_m to a. This puts the first register in an equal superposition of all integers from 0 to $2^m - 1$. The current state is now

$$\frac{1}{\sqrt{2^m}} \sum_{a=0}^{2^m-1} |a\rangle|0\rangle.$$

5. Compute $x^a \bmod N$ and place the result in b. Notice that since a is in an equal superposition of all integers from 0 to $2^m - 1$, b is now in a superposition of x^a for all integers a from 0 to $2^m - 1$. In addition, the two registers are now entangled accordingly. The current state is now

$$\frac{1}{\sqrt{2^m}} \sum_{a=0}^{2^m-1} |a\rangle|x^a \bmod N\rangle.$$

6. Measure b. This will yield some value y, which we really don't care about. More importantly, b will collapse to the state y, and a will collapse to an equal superposition of all values k such that $x^k \equiv y \bmod N$. Let r be the order of x, and let l be the smallest value such that $x^l = y$. It is not hard to see that $x^k \equiv y \bmod N$ if and only if $k = l+rj$ for some j. Notice that there are $A \approx q/r$ values of j such that $0 \leq l + rj \leq q$. Thus, our current state is now

$$\frac{1}{\sqrt{A}} \sum_{j=0}^{A} |l + rj\rangle|y\rangle.$$

We will no longer deal with register b, so I won't write it anymore. Thus our state is

$$\frac{1}{\sqrt{A}} \sum_{j=0}^{A} |l + rj\rangle.$$

7. Notice that if we could measure the register twice, we could extract the period r by a simple subtraction. Unfortunately, as previously stated, one measurement

will collapse the state, and a second measurement will yield the same result. Thus, we need some trick to extract r. That is where the quantum Fourier transform comes in. We compute the Fourier transform of a and get

$$\frac{1}{\sqrt{r}}\sum_{j=0}^{r-1} e^{2\pi i l j/r)}|j\frac{q}{r}\rangle.$$

The reader can check this, or read all of the gory details in [24]. Notice that the value l is not seen in this state. The Fourier transform extracted the period of the function (actually the inverse of the period), wiping away the "shift" l. Thus, it didn't matter which state was measured in step 6.

8. Now, a measurement of a will yield some number $j(q/r)$, where both j and r are unknown. The value of r can be extracted with high probability using the classical technique for continued fraction expansion. (See appendix A of either [24, 45]).

9. Repeat the algorithm until it is successful. The procedure fails when one of the following happens:

   (a) r is odd.

   (b) $x^{r/2} \equiv \pm 1 \bmod N$, so a trivial factor is found.

   (c) j and r have a common factor. In this case, the period is a multiple of the number found. This case can be salvaged some of the time. Since we know a divisor of the period, we can determine if small multiples of the returned number are the period.

   (d) The continued fraction expansion can return an answer that is close, but not close enough.

   Repeating the process $O(\log r) < O(\log n)$ times will amplify the success probability arbitrarily close to 1.

That Shor's algorithm is polynomial is not absolutely clear. The complexity depends on the time complexity of performing the Walsh-Hadamard Transform, the quantum

Fourier transform, and the computation of $x^a \bmod N$. There are methods for performing each of these in time polynomial in $O(\log N)$ [5]. Since the computation must be repeated at most $O(\log r) < O(\log n)$ times, the total complexity is still polynomial in $O(\log N)$.

3.2 Grover's Algorithm

Many problems in computer science can be thought of as search problems. Given a list of elements, one wishes to pick out an element with a certain property. Often problems of this type have some sort of structure that can be exploited to make searching rather quick. However, some problems have no structure that can be exploited. These problems are often referred to as unstructured search problems.

For an unstructured search problem, the best a classical algorithm can do is to search through the list one by one. Whether this is done randomly or in some order, the complexity will be $O(n)$, where n is the size of the list. Grover's algorithm solves the problem in time $O(\sqrt{n})$ on a quantum computer. This is an example of a quantum algorithm that is provably better than what can be done on a classical computer. (It should be noted that it hasn't been proven that Shor's algorithm is more efficient than any classical factoring algorithm. It is exponentially faster than any *known* classical algorithm.)

In Grover's original paper [32], he assumed that only one element has the given property. When there are t elements with the given property, the complexity is $O(\sqrt{n/t})$ [12]. Grover's algorithm is discussed in detail by various researchers [32, 33, 34, 39] and has been shown to be optimal [10, 32, 35, 59].

We will state the unstructured problem more precisely. Given an unstructured list of n elements and some property P, we wish to find an element x from the list such that $P(x)$ is true. That is, P will return 1 if x satisfies the property and 0 otherwise. For the algorithm, we will make several assumptions. First, the items on the list correspond to the 2^m possible base states in a m-qubit quantum register. Second, we have an efficient method for determining whether or not $P(x)$ is true for a given x

without measuring. In other words, we have a quantum gate which acts as follows:

$$U_P : |x\rangle|0\rangle \mapsto |x\rangle|P(x)\rangle$$

Lastly, we will assume just one value x has $P(x) = 1$. If there are more than one, the only modification needed is to change the number of times the algorithm is repeated [12].

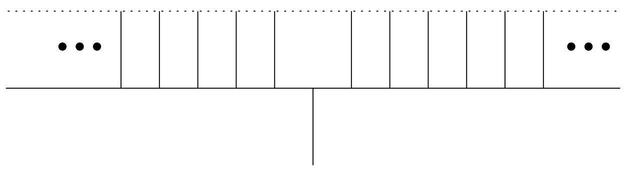
The algorithm is as follows:

1. As in Shor's algorithm, we start with $2m$ qubits, which we will think of as two registers a and b. Both a and b will start out in state $|0\rangle$. Thus, the initial state is
$$|0\rangle|0\rangle.$$

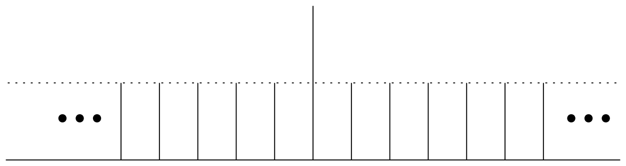
2. Apply W_m to a. This puts the first register in an equal superposition of all integers from 0 to $2^m - 1$. The current state is now
$$\frac{1}{\sqrt{2^m}} \sum_{a=0}^{2^m-1} |a\rangle|0\rangle.$$

3. Compute the transformation U_P. For the first iteration, we obtain
$$\frac{1}{\sqrt{2^m}} \sum_{a=0}^{2^m-1} |a\rangle|P(a)\rangle.$$
It seems at this point we could measure the second register. If we measure 1, we know the first register is in a superposition of those x such that $P(x) = 1$, so we can then measure the first register and obtain one of the desired x. Unfortunately, if t is the number of values of x such that $P(x) = 1$, then the probability of measuring 1 in the second register is $t/2^m$, which is small if t is small. Even if t is large, this would be no better than the classical algorithm.

4. Apply a transformation that changes the sign of the amplitude of the x for which $P(x) = 1$. There is an efficient method for doing this [32]. This will result in the amplitudes of the states looking something like the following:



5. Apply inversion[2] about the average to the first register. This means that the amplitudes of each state will be as much above (below) the average as they were below (above). The method of performing this operation is given in [32, 45]. This will result in the amplitude of the x for which $P(x) = 1$ to be about twice as large, and the amplitudes of the other states to be just slightly lower. It will look something like the following



6. Repeat steps 3 through 5 $\frac{\pi}{4}\sqrt{2^m}$ times.

7. Measure the second register. Now with a high probability, a 1 will be obtained, and a measurement of the first register will yield the correct value of x.

4 Quantum Error-Correction

4.1 Introduction and Brief History

So far the picture I have painted of quantum computing looks too good to be true. In a sense it is. So far in the discussion, I made three assumptions:

- All of the transformations discussed can be implemented perfectly,
- no errors occur when applying transformations to quantum states, and
- the state of a quantum register is independent of the environment.

[2]reflection

Unfortunately, none of these assumptions is valid. Some of transformations discussed can only be implemented within a certain threshold. When applied repeatedly, the transformations may lead to states that are not close enough to the desired result. The method of implementation of the quantum computer (Two promising implementations include the use of *cold trapped ions* [16] or *Nuclear Magnetic Resonance (NMR)* [28, 37, 46]) can possibly bring with it causes of different sources of errors. Different quantum systems are prone to different errors. There may also be restrictions on the amount of precision with which a transformation may be implemented or a measurement may be made. In short, the gates required for quantum computation are not perfect.

On top of this, the state of a register can become entangled with the environment. In other words, the state of the register and the state of the environment are not independent. When this happens, the state of the register starts to decay to a classical state. This process is referred to as *decoherence*.

One may ask how bad these problems are. Steane [53] estimates that the rate of decoherence is about 10^7 times too fast to perform factorization on a 130 digit number. Thus, it is unreasonable to expect that we can implement quantum computers on a large scale without some way of protecting against errors. In addition, when errors occur, there must be a method of *undoing* them. The goal of the field of quantum error-correction, much like its classical counterpart, is precisely this.

In some sense, classical codes seem to rely on the ability to copy the data which is being protected. Since copying (cloning) a quantum state is impossible (Theorem 2.1), it was thought that perhaps quantum error-correction was impossible. But quantum error-correcting codes do exist.

In 1995 and 1996, Shor [47], Steane [54, 52], and Calderbank and Shor [15] published the seminal results in the theory of quantum error correction. In 1996, Bennett et al. [7] related the idea of quantum error correction to entanglement purification protocols. The theory was more formalized by Knill and Laflamme [41], and Calderbank et al. [13, 14] in 1997. Some very recent papers transfer the ideas of cyclic codes [29], BCH codes [30] and Reed-Solomon codes [31] to quantum codes.

An important consideration when dealing with quantum error correction is that the methods that attempt to correct errors are not perfect, so they introduce errors. Fortunately, it is possible to implement error correction so that the amount of overall noise (error) after the process is reduced [48].

For the remainder of this paper, we will assume that the process of error correction acts perfectly – that is, it introduces no additional errors.

4.2 Quantum Error Model

We have discussed the idea of errors in quantum states, but have not defined what an error is. Below we state the error model we will work with. Many authors [13, 14, 53] work under this error model.

Recall the Pauli Matrices defined earlier: $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, and $Y = XZ$.[3] When applied to a qubit, X corresponds to a bit flip, Z to a phase error, and Y to both. Notice that the set $P = \{\pm I, \pm X, \pm Y, \pm Z\}$ forms a group isomorphic to the dihedral group D_4.

The error model we use is based on two assumptions. First, each error occurs randomly and effects only one qubit. Thus, if a quantum register experiences errors in t qubits, the errors are all independent of each other. Second, the possible error transformations are those described by the Pauli matrices.

Given these assumptions, we see that the possible errors are members of the group $\mathcal{E}$ of tensor products $\pm w_1 \otimes \cdots \otimes w_n$, where each $w_i \in \{I, X, Y, Z\}$. The group $\mathcal{E}$ is a subgroup of the unitary group $U(2^n)$. The *weight* of an error is the number of matrices $w_i \neq I$ in the tensor product.

In reality, errors may not act independently on each qubit, and may have non-discrete phase changes. It turns out that any code that corrects t errors under these assumptions will correct arbitrary errors on t qubits [7, 41, 13, 14, 43]. Thus, the restriction is not too severe.

[3]Some authors define $Y = iXZ$.

4.3 Quantum Error-Correcting Codes

Assume we have a k-qubit register we wish to encode. Let the state of the register be $|\phi\rangle$. The error correction scheme will use $k+2(n-k)$ qubits. We begin by encoding the first k qubits with an additional $n-k$ qubits, initially in state $|0\rangle$: $E(|\phi\rangle|0\rangle) = |\phi_e\rangle$. The set of encodings forms a subspace of the n-qubit register. From now on, we can imagine that the first n qubits form an n-qubit register. The last $n-k$ bits are called an *ancilla*, and are used to extract the *syndrome* as will be described shortly. They will be set to state $|0\rangle$ before the error detection is done.

Let $M \in \mathcal{E}$ be an error that effects our quantum state $|\phi_E\rangle$. The state of the system after the error is applied is $M|\phi_E\rangle$. We wish to be able to reconstruct $|\phi_E\rangle$ from $M|\phi_E\rangle$, where we do not know which error M occurred.

Each encoding scheme E has a set $\mathcal{S}$ of correctable errors. That is, if $M \in \mathcal{S}$, then the state $|\phi_E\rangle$ can be recovered from $M|\phi_E\rangle$. Each error $M_s \in \mathcal{S}$ has associated with it a *syndrome* s, and there is an operation A which will extract the syndrome state $|s\rangle$ given the state $M_s|\phi_E\rangle$. That is, $A(M_s|\phi_E\rangle|0\rangle) = M_s|\phi_E\rangle|s\rangle$. The syndrome states are mutually orthogonal so that they are distinguished by measurement. Given the syndrome $|s\rangle$, we know which error occurred, and we can apply the inverse operator M_s^{-1} to recover the state $|\phi_E\rangle$. We can then reset the ancilla back to $|0\rangle$ for use again.

Now assume that several errors from the set $\mathcal{S}$ occur. Then the state of the register will become

$$\sum_s |e_s\rangle M_s|\phi_E\rangle,$$

where $|e_s\rangle$ are states of the environment. If we apply A to the state, we obtain

$$\sum_s |e_s\rangle(M_s|\phi_E\rangle)|s\rangle.$$

Measuring the ancilla state will collapse the state to $|e_s\rangle(M_s|\phi_E\rangle)|s\rangle$ for some s, removing all but one error, which we can correct as before.

Given the general idea of quantum error correction, the obvious question is which errors an encoding E correct. It is more useful to think of the code as a subspace rather than an encoding. Knill and Laflamme [41] (see also [43]) gave a simple characterization of the sets of errors correctable by a code:

Theorem 4.1 *Let Q be a subspace of an n-qubit quantum register of dimension at least three, and let $\mathcal{T}$ be any set of $2^n \times 2^n$ matrices. Then the following are equivalent:*

1. *Q allows correction of all errors in $\mathcal{T}$;*
2. *all errors $E_1^* E_2$ such that $E_1, E_2 \in \mathcal{T}$ are detectable relative to Q;*
3. *for all $E_1, E_2 \in \mathcal{T}$, and all $x, y \in Q$, if x and y are orthogonal, then $E_1 x$ and $E_2 y$ are orthogonal;*
4. *for all $E_1, E_2 \in \mathcal{T}$, there exists a scalar $\lambda_{(E_1,E_2)}$ such that for all $x \in Q$,*
$$\langle E_1 x, E_2 x\rangle = \lambda_{(E_1,E_2)} |x|^2;$$
5. *for each error $E = E_1^* E_2$ such that $E_1, E_2 \in \mathcal{T}$, there exists a scalar λ_E such that*
$$\Pi_Q E \Pi_Q = \lambda_E \Pi_Q,$$
where Π_Q is the orthogonal projection onto Q.

In practice, certain errors and types of errors occur more often than other types. The trick is to find quantum error-correcting codes E that will correct the most common errors. This is a difficult task. The most obvious, and most studied, types of codes are those which correct all errors affecting t or less qubits. These are called t-error correcting codes. There is an interesting method, developed by Shor and Calderbank [15] and Steane [54], for using classical codes and their duals for quantum error correction. I will conclude this section with a discussion of the ideas behind this.

Linear classical error-correcting codes will correct X-errors. Since X-errors are bit flips, we can use a classical code C to encode our k-qubit state to the n-qubit state corresponding to the codewords of C. Recall that the syndrome extraction of a linear code depends on only the error, not the codeword itself. Thus, no matter what state the register is in, the error can be corrected. Thus, we can use an $[n, k, d]$ code to correct up to $(d-1)/2$ X-errors.

Unfortunately, we cannot use these codes to correct Z-errors. However, notice that $Z = W_n X W_n$, where W_n is the Walsh-Hadamard transform discussed in example 1.5. So correcting Z-errors is equivalent to rotating the state of each qubit by H, correcting X-errors, and rotating back. Steane [54] noticed that

$$W_n \sum_{i \in C} |i\rangle = \frac{1}{\sqrt{2^k}} \sum_{j \in C^\perp} |j\rangle.$$

This fact can be used to correct Z-errors using $C^\perp$.

Since Y-errors are a combination of X- and Z-errors, it should be evident at this point that good quantum error-correcting codes can be obtained from good classical linear codes C for which $C^\perp$ is also a good code. In fact, linear classical codes which are contained in their dual (such codes are called *weakly self-dual*) work very well as quantum codes. Various authors have investigated weakly self-dual codes, and their error-correcting capability as quantum codes. Calderbank et al. discuss quantum codes over $GF(4)$ [14], Grassl and Beth discuss quantum BCH [30] and cyclic [29] codes, and Grassl, Geiselmann, and Beth discuss quantum Reed-Solomon codes [31].

5 Conclusions

We have seen that quantum computers are capable of performing some tasks faster than any classical computer. There seem to be two major hurdles in making quantum computing a reality. The first is implementation. Although many researchers are currently working on different methods of implementation, the largest experiments done thus far involve tens of qubits, whereas useful quantum computing will require at least thousands of qubits. One should keep in mind, however, that the field of quantum computing is in its infancy, and much progress is expected over the next 10 to 30 years in regards to this issue.

The second is the problem of decoherence. However, the fields of error-correcting coding theory and fault-tolerant quantum computing are moving forward rapidly.

Many researchers now believe that the errors can be dealt with sufficiently to allow quantum computations on the scale required to solve interesting problems.

Even if quantum computing never becomes a reality, the study of quantum computing has helped to bring better understanding of fields ranging from quantum mechanics and physical systems to information theory and computational complexity.

References

[1] A. Barenco. A universal two-bit gate for quantum computation. *Proc. R. Soc. Lond. A*, June 1995. quant-ph/9505016.

[2] A. Barenco. Quantum physics and computers. *Contemp. Phy.s*, 37:375, 1996. quant-ph/9612014.

[3] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P. Shor, T. Sleater, J. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 1996. quant-ph/9503016.

[4] A. Barenco, D. Deutsch, and Q. Ekert. Conditional quantum dynamics and logic gates. *Phys. Rev. Lett.*, 74:4083–4086, 1995. quant-ph/9503017.

[5] D. Beckman et al. Efficient networks for quantum factoring. quant-ph/9602016, 1996.

[6] J.S. Bell. On the einstein poldosky rosen paradox. *Physics*, 1:195–200, 1964.

[7] C.H. Bennet, D.P. DiVincenzo, J.A. Smolin, and W.K. Wootters. Mixed-state entanglement and quantum error correction. *Phys. Rev. A*, 54(5):3824–3851, Novermber 1996.

[8] C.H. Bennett. Logical reversibility of computation. *IBM J. Res. Devel.*, 17:525–532, Nov. 1973.

[9] C.H. Bennett and P.W. Shor. Quantum information theory. *IEEE Transactions on Information Theory*, October 1998.

[10] C.H. Bennett et al. Strengths and weaknesses of quantum computing. *Siam J. Comput.*, 26(5):1510–1523, October 1997.

[11] A. Berthiaume. Quantum computation. In *Complexity Theory Retrospective II.* Springer-Verlag, 1996.

[12] M. Boyer et al. Tight bounds on quantum searching. In *Proceedings of the Workshop on Physics of Computation: PhysComp '96*, Los Alamitos, CA, 1996. IEEE Society Press. quant-ph/9605034.

[13] A.R. Calderbank, E.M. Rains, P.W. Shor, and N.J.A. Sloane. Quantum error correction and orthogonal geometry. *Phys. Rev. Let.*, 78(3):405–408, January 1997.

[14] A.R. Calderbank, E.M. Rains, P.W. Shor, and N.J.A. Sloane. Quantum error correction via codes over gf(4). *IEEE Trans. Information Theory*, to appear 1997. quant-ph/9608006.

[15] A.R. Calderbank and P.W. Shor. Good quantum error-correcting codes exist. *Phys. Rev. A*, 54(2):1098–1105, August 1996.

[16] J.I. Cirac and P. Zoller. Quantum computation with cold trapped ions. *Physical Review Letters*, 74(20):4091–4094, May 1995.

[17] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *Phil. Trans. R. Soc. Lond. A*, 1996. quant-ph/9708016.

[18] D. Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proc. R. Soc. Lond. A*, 400:97–117, 1985.

[19] D. Deutsch. Quantum computational networks. *Proc. R. Soc. Lond. A*, 425:73–90, 1989.

[20] D. Deutsch, A. Barenco, and A. Ekert. Universality in qunatum computation. *Proc. R¿ Soc. Lond. A*, 449:669, Jun3 1995. quant-ph/9505018.

[21] P.A.M. Dirac. *The principles of Quantum Mechanics.* The International Series of Monographs on Physics (27). Oxford University Press, New York, 4th edition, 1958.

[22] D.P. DiVincenzo. Quantum gates and circuits. *Phil. Trans. R. Soc. Lond. A*, December 1996.

[23] A. Einstein, B. Poldosky, and N. Rosen. Can quantum-mechanicaldescriptions of physical reality be considered complete? *Physicsl Review*, 47:777–780, 1935.

[24] A. Ekert and R. Jozsa. Quantum computation and shor's factoring algorithm. *Rev. of Mod. Phys.*, 68(3):733–753, July 1996.

[25] A. Ekert and R. Jozsa. Quantum algorithms: Entanglement enhanced information processing. *Phil. Trans. Roy. Soc. (Lond.)*, 1998. quant-ph/9803072.

[26] R.P. Feynman. Simulating physics with computers. *Iner. J. of Th. Phys.*, 21(6/7):467–488, 1982.

[27] R.P. Feynman. *Feynman Lectures on Computation.* Addison-Wesley Publishing Company, Inc, 1996. Chapter 6 reprinted from *Optics News*, February 1985, 11-20.

[28] N.A. Gershenfeld and I.L. Chuang. Bulk spin-resonance quantum computation. *Science*, 275:350–356, January 1997.

[29] M. Grassl and T. Beth. Cyclice quantum error-correcting codes and quantum shift registers. quant-ph/9908061, October 1999.

[30] M. Grassl and T. Beth. Quantum bch codes. quant-ph/9908060, October 1999.

[31] M. Grassl, W. Geiselmann, and T. Beth. Quantum reed-solomon codes. quant-ph/9910059, October 1999.

[32] L.K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings, 28th Annual ACm Symposium on the Theory of Computing (STOC)*, pages 212–219, May 1996. quant-ph/9605043.

[33] L.K. Grover. A framework for fast quantum mechanical algorithms. quant-ph/-9711043, 1997.

[34] L.K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.*, 78(2):325–328, July 1997. quant-ph/9706033.

[35] L.K. Grover. How fast can a quantum computer search? quant-ph/9809029, Sept 1998.

[36] P. Hoyer. Efficient quantum transforms. quant-ph/9702028, 1997.

[37] J.A. Jones and M Mosca. Implementation of a quantum algorithm to solve deutsch's problem on a nuclear magnetic resonance quantum computer. *Journal of Chemical Physics*, 109:1648, 1998. quant-ph/9801027.

[38] R. Jozsa. Quantum algorithms and the fourier transform. quant-ph/9707033, 1997.

[39] R. Jozsa. Searching in grover's algorithm. quant-ph/9901021, Jan 1999.

[40] A.Y. Kitaev. Quantum measurment and the abelian stabilizer problem. quant-ph/9511026, 1995.

[41] E. Knill and R. Laflamme. Theory of quantum error-correcting codes. *Phys. Rev. A*, 55(2):900–911, February 1997.

[42] E. Knill, R. Laflamme, and L. Viola. Theory of quantum error correction for general noise. quant-ph/9908066, August 1999.

[43] W. Martin. A physics-free introduction to quantum error correcting codes. unpublished?

[44] J. Preskill. Quantum computing lecture notes from physics 229 at caltech., 1997. Available at http://www.theory.caltech.edu/people/preskill/ph229.

[45] E. Rieffel and W. Polak. An introduction to quantum computing for non-physicists. quant-ph/9809016, August 1998.

[46] L.J. Schulman and U. Vazirani. Scalable nmr quantum computation. quant-ph/-9804060, 1998.

[47] P.W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52(4):R2493–R2496, October 1995.

[48] P.W. Shor. Fault-tolerant quantum computation. quant-ph/9705011, 1997.

[49] P.W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *Siam. J. Comput.*, 26(5):1484–1509, October 1997.

[50] D.R. Simon. On the power of quantum computation. *Siam J. Comput.*, 26(5):1474–1483, October 1997.

[51] T. Sleator and H. Weinfurter. Realizable universal quantum logic gates. *Phys. Rev. Lett.*, 74(20):4087–4090, May 1995.

[52] A. Steane. Multiple-particle interference and quantum error correction. *Proc. R. Soc. Lond. A*, 452:2551–2577, 1996.

[53] A. Steane. Quantum computing. *Rept. Prog. Phys.*, 61:117–173, 1998. quant-ph/9708022.

[54] A.M. Steane. Error correcting codes in quantum theory. *Phys. Rev. Let.*, 77(5):793–797, July 1996.

[55] V. Vedral, A. Barenco, and E. Ekert. Quantum networks for elementary arithmetic operations. *Phys. Rev. A*, 1995. quant-ph/9511018.

[56] V. Vedral and M.B. Plenio. Basics of quantum computation. *Porgress in Quantum Electronics*, 22, 1998. quant-ph/9802065.

[57] C.P. Williams and S.H. Clearwater. *Explorations in Quantum Computing.* Springer-Verlag, New York, 1998.

[58] W.K. Wootters and W.H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, October 1982.

[59] C. Zalka. Grover's quantum searching algorithm is optimal. quant-ph/9711070, 1997.