B A C H E L O R T H E S I S

Grover's Algorithm

for the lecture
Projektpraktikum Theoretische Physik - Computational Physics,

executed at
Theoretische Physik - Computational Physics der Technischen Universität Graz

instructed by Univ.-Prof. Dr.rer.nat. Enrico Arrigoni

by
Manuel Auer
Matr.-Nr. 0530802

in
WS 2007/8

Contents

1 Introduction

Grover's algorithm is one of the first conceived algorithms designed to take advantage of the features a quantum computer offers. It is one of the algorithms often used in examples to show the potential of quantum computers in certain areas. This bachelor thesis aims at providing a broad insight into the workings of grovers algorithm using various examples, which should make it very easy to understand of whats really going on. This bachelor thesis relied heavily on the very well written, although in some parts lacking paper by C. Lavor [9] and the quite interesting paper by E. Borbely [2]. Details about what was done by the writer and what was copied or modified is to be found in the begining of every section.

2 Basics for Grover's Algorithm

This section of the bachelor thesis aims at providing some of the required mathematical tools and physical background used later in this bachelor thesis. The template for this section was served by [9], Sec. 3-4. The matrix representations of the required quantum gates were taken from [4]. One part that will not be covered are tensor products. For more information about tensor products, please refer to appropriate literature.

2.1 Information representation

Compared to a classical computer, a quantum computer can use quantum states instead of the classical ones. So, instead of talking about bits in quantum computing one usually speaks of a qubit (quantum bit). Like the classical counterpart, a qubit can have the state 0 or 1. The only difference is the notation, as in quantum mechanics quantum states are usually written using the so called Dirac notation. We therefore speak of $|0\rangle$ and $|1\rangle$. Now, that we have pointed out the similarities between a qubit and a classical bit, we may focus a bit in the way both differ. namely, a qubit can be in a linear combination $|\psi\rangle$ of the states $|0\rangle$ and $|1\rangle$ which looks like

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

having the normalization constraint

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

This is a so called superposition of the states $|0\rangle$ and $|1\rangle$ with the amplitudes α and β, where α and β are in general complex numbers. The states $|0\rangle$ and $|1\rangle$ represent the computational basis. The vector representations of the states look like

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \text{and} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

2.2 Multi-qubit states

To accomplish computational tasks on a quantum computer it is necessary to have more than one qubit at once. Now we want to tackle how these states are represented mathematicaly. We want to show this using a general two qubit state

$$|\Psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \tag{3}$$

with the normalization constraint

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$$

Looking at Eq.3 we see that the states of the two qubits are binary numbers. One may replace these binary numbers

$$|00\rangle, |01\rangle, |10\rangle, |11\rangle$$

with the decimal representations

$$|0\rangle, |1\rangle, |2\rangle, |3\rangle$$

This representation allows us to write a general n-qubit state $|\sigma\rangle$, which is nothing less than a superposition of the 2^n states $|0\rangle$, $|1\rangle$, ..., $|2^n - 1\rangle$ (these states also represent the computational basis). Mathematicaly this is written as

$$|\sigma\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$$

having the generalized normalization constraint

$$\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$$

One last aspect to show that a general 2 qubit state is not the product of two 1 qubit states. Lets assume we have a qubit in the state $|\psi\rangle$ and another one in $|\mu\rangle$ of the form

$$\begin{aligned} |\psi\rangle &= a|0\rangle + b|1\rangle \\ |\mu\rangle &= c|0\rangle + d|1\rangle \end{aligned}$$

The state arising from the two is represented by the tensor product

$$\begin{aligned} |\psi\rangle \otimes |\mu\rangle &= (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle) \\ &= ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle \end{aligned} \tag{4}$$

A general 2 qubit state Eq.3 is of the form Eq.4 only if

$$\begin{aligned} \alpha &= ac \\ \beta &= ad \\ \gamma &= bc \\ \delta &= bd \end{aligned}$$

These equalities point out, that a general 2 qubit state (Eq.3) is of the form Eq.4 if

$$\alpha\delta = \beta\gamma$$

This result shows that the general 2 qubit state (Eq.3) is not a product of two 1 qubit states. Two qubit states that are not a product of one-qubit states are called entangeled states.

2.3 Computation and measurement

In this section we want to briefly discuss how a quantum computer actually executes calculations and how to get to the result.

Measurement Let's think about a general 1-qubit state introduced in Eq.1. We previousely introduced α and β as the amplitudes, but what do they actually mean? In quantum mechanics a measurement of the state $|\psi\rangle$ gives either $|0\rangle$ or $|1\rangle$ with the respective probabilities $|\alpha|^2$ and $|\beta|^2$. After a measurement the superposition is effectively destroyed and either one of the two basis states is returned with respective probability. In quantum physics this process is called non-deterministic collapse. In many qubit environments its commonplace to measure the quantum register (which consists of n-qubits) qubit by qubit which yields the result of the computation with respective probability given by the squared norms of the corresponding amplitudes in $|\psi\rangle$.

Computation As seen in the previous paragraph any attempt to see whats going on in the quantum register leads to a non-deterministic collapse of the state to one of its computational base vectors, thus preventing any further attempt to manipulate the quantum register in and advantageous way. The only way to do quantum computing is done using so called unitary operations. The reason is that a unitary operation doesn't lead to a collapse of the wave function as an application of one of these operators only manipulates the amplitudes while keeping the normalization. That's also the reason why unitary operations are reversible, as theres always a way back to the state the system was in before application of the operator.

2.4 Required gates

Because of the chosen quantum network model (instead of a Turing machine model), we require gates, which perform the unitary evolution from the initial state to the one that contains the solution. Consequently you will be introduced to all required Gates in terms of the way they work and their matrix representation. [4] was used as template for this subsection. The following quantum gates will be required:

Hadamard Gate As mentioned in the first chapter, one reason, why quantum computers can work significantly faster than classical ones, leads back to a quantum property called 'Quantum Parallelism' also refered to as 'superposition'. In order to move a quantum register into a superposition, the so-called Hadamard-Gate is used. The Hadamard Gate is a one-qubit Gate. The Hadamard Gate works by moving a Qubit in the state $|0\rangle$ to

$$H|0\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

On the other hand the application of the Hadamard Gate on $|1\rangle$ yields

$$H|1\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Once a superposition is created it can also be destroyed by applying the Hadamard Gate to the qubit whose state is in a superposition. Therefore

$$H\tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |0\rangle$$

The matrix representation of H looks like

HADAMARD GATE

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

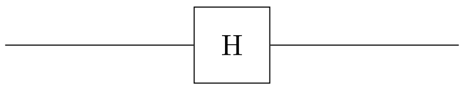

Figure 1: Circuit symbol for Hadamard Gate

Controlled-Not Gate The Controlled-Not Gate is a 2-qubit gate. It expects two input qubits, one target qubit and one control qubit. The target qubit is only flipped, if the control qubit is set to one. The Controled-Not Gate will be represented by the capital letter C.
In order to get an idea, how the Controlled-Not gate works, watch the following example,

$$\begin{aligned} C|00\rangle &= |00\rangle \\ C|01\rangle &= |01\rangle \\ C|10\rangle &= |11\rangle \\ C|11\rangle &= |10\rangle \end{aligned}$$

The Controlled-Not Gate posseses the following matrix representation

CONTROLLED-NOT GATE

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

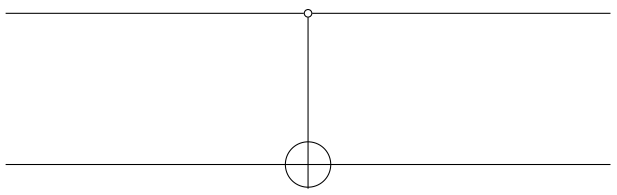

Figure 2: Circuit symbol for Controlled-Not Gate

Toffoli-Gate The Toffoli-Gate is a 3 qubit gate. It requires 3 input qubits, one being the target qubit and the other two act as control qubits.
From now on the Toffoli-Gate is represented by the capital letter T. The Toffoli-Gate works in the following way,

$$\begin{aligned}
T|000\rangle &= |000\rangle \\
T|001\rangle &= |001\rangle \\
T|010\rangle &= |010\rangle \\
T|011\rangle &= |011\rangle \\
T|100\rangle &= |100\rangle \\
T|101\rangle &= |101\rangle \\
T|110\rangle &= |111\rangle \\
T|111\rangle &= |110\rangle
\end{aligned}$$

The Toffoli-Gate has the following matrix representation

Toffoli Gate

$$T = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \end{pmatrix}$$

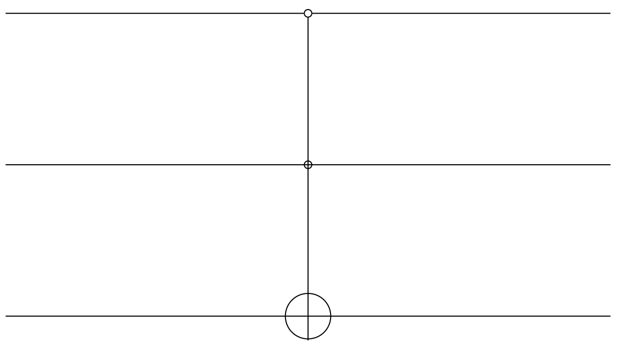

Figure 3: Circuit symbol for Toffoli

Not-Gate When one considers the action of the one qubit Not-Gate, it is found that the gate has the following effect on the qubit under consideration:

$$X|0\rangle = |1\rangle$$
$$X|1\rangle = |0\rangle$$

Thus it is easily found, that the matrix that represents this gate is of the following form

NOT GATE

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

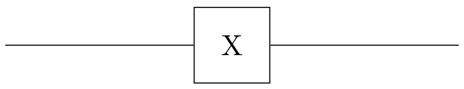

Figure 4: Circuit symbol for NOT Gate

3 Grovers'Algorithm

This section had its foundation in [9]. The subsection dealing with *Inversion about average* is based on the work by E. Borbely [2] and Lov K. Grover [5].

3.1 Introduction

Consider the following problem:
One has an unstructured database with N elements, which in turn are not ordered. How many attempts would it take on average until the element of ones desire is found? Classically it would take $\frac{N}{2}$, as the classical approach is limited to look for only one element at once. It shall later be demonstrated that the quantum mechanical approach, also commonly refered to as Grover's algorithm, is able to find the desired element in O($\sqrt{N}$) trials, when $N = 2^n$

3.2 Grover Interation

Now we want to list the steps taken during the execution of Grover's Algorithm:

1. Initialize the quantum register in a uniformly distributed superposition
2. Being called Oracle, this operator (or this set of quantum logic) marks the desired element by changing the sign of the probability amplitude, while leaving the others in their respective state
3. This step is commonly called "Inversion about average" for reasons we shall later discuss in detail. It increases the amplitude of the searched element and at the same time decreases the others amplitude

These 3 steps are generally called *Grover Iteration.* One has to perform this iteration approximately $\sqrt{N}$ times, as we are going to see later, to be able to meassure the searched element with probability close to 1.

3.2.1 Creating a superposition

Now, that the proceedings in general are known, we shall delve deeper into the inner workings of Grover's Algorithm. We start with generating a uniformly distributed superposition. In order to do that, one must know, that the algorithm requires two registers to execute properly. The first contains n qubits and the second one a single qubit. From Sec. 2.1 we know, that each single qubit can encode 2 states at a given time. Therefore, n qubits can represent $N = 2^n$ states. So, back to generating the superposition.
This can easily be achieved by initializing the first register in the state $|\Psi\rangle = |0\rangle \otimes |0\rangle \otimes \ldots \otimes |0\rangle$ and afterwards by applying the Hadamard operator H to each single qubit (or equivalently by applying the operator $H^{\otimes n}$). This results in

$$\begin{aligned}|\Psi\rangle &= H|0\rangle \otimes \ldots \otimes H|0\rangle \\ &= (\frac{|0\rangle + |1\rangle}{\sqrt{2}}) \otimes \ldots \otimes (\frac{|0\rangle + |1\rangle}{\sqrt{2}}) \\ &= (H|0\rangle)^{\otimes n} \\ &= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle\end{aligned}$$

As required this superposition is truely equally distributed with a probability amplitude of $\frac{1}{\sqrt{N}}$ for each element that can be encoded (pay attention, that n is the number of qubits in the first register, whereas N is the amount of encodeable elements).

Now let's turn to the second register (it's purpose is going to be uncovered later on).

The second register starts being in the state $|1\rangle$. By applying a hadamard gate, the resulting state is

$$|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

By having the second register in a superposition, we can now discuss the oracle in more detail.

3.2.2 The oracle

As indicated prevoiusely, the oracle's responsibilities are to detect the desired element and insert it's probability amplitude. It's action is best described by the following function

$$f : \{0, \ldots, N-1\} \to \{0, 1\} \quad \text{with} \quad f(i) = \begin{cases} 1 & \text{if i is the searched element} \\ 0 & \text{otherwise} \end{cases}$$

This is a classical function capable of evaluating one element at once. In a quantum computer on the other hand, we want to deploy a "function" which makes good use of the previousely introduced quantum parallelism.
At this point we try to build a linear unitary operator which in turn depends on f so that

$$U_f(|i\rangle|j\rangle) = |i\rangle|j \oplus f(i)\rangle \quad \text{where } \oplus \text{ means addition modulo 2}$$

Here we introduced the so called oracle operator U_f. The equation includes $|i\rangle$ which represents the first register that was previousely initialized in a evenly distributed superposition. $|j\rangle$ was used to represent the second register. Now we shall verify the outcome, when one applies the operator U_f to both registers.
First we have to denote the fact that

$$1 \oplus f(i) = \begin{cases} 0 & \text{for } i = i_0 \\ 1 & \text{for } i \neq i_0 \end{cases} \tag{5}$$

which is obvious, as only when both bits are 1 the bit can be marked with a negative sign. Having Equation 5, we can now easily obtain a more useful form of U_f.

$$\begin{aligned} U_f(|i\rangle|-\rangle) &= \frac{U_f(|i\rangle|0\rangle) - U_f(|i\rangle|1\rangle)}{\sqrt{2}} \\ &= \frac{|i\rangle f(i) - |i\rangle|1 \oplus f(i)\rangle}{\sqrt{2}} \\ &= (-1)^{f(i)}|i\rangle|-\rangle \end{aligned}$$

Having U_f in this form we can now finally show the action of U_f on the two registers

$$\begin{aligned} |\Psi_F\rangle|-\rangle &= U_f(|\Psi\rangle|-\rangle) \\ &= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} U_f(|i\rangle|-\rangle) \\ &= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{f(i)}|i\rangle|-\rangle \end{aligned}$$

$|\Psi_F\rangle$ is the resulting state. $|-\rangle$ doesn't change.
Truely, we achieved our final goal of having the searched state marked by a negative sign.

As previousely indicated, we have made use of quantum parallelism, which allows us to see all elements at once. At this point one may be reminded that this information is not available classically, as any measurement at this point will with a high probability result in a wrong outcome.

3.2.3 The Grover operator

Now, that theres the oracle operator in place, its time to define the grover operator.

We start recalling what the grover operator is supposed to do: It should increase the amplitude of the searched element and at the same time decrease all other elements amplitudes.

It turns out that this operator is represented by

GROVER OPERATOR

$$G = 2|\Psi\rangle\langle\Psi| - I \tag{6}$$

We are not deriving the operator directly, as this would go beyond this bachelor thesis, but for further reading please refer to [6], hence we try to get more insights in the inner working of the grover iteration.

In order to do so, there are two major ways of explaining the grover iteration.

- Inversion about average
- Rotating the vector $|\Psi\rangle$ towards $|i_0\rangle$ (which is the solution) in a real vector subspace of the Hilber space

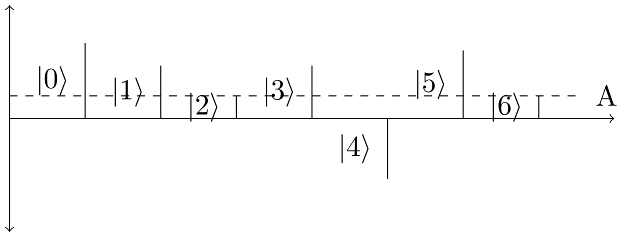

Figure 5: Before inversion about average operation is applied. Note that the only state with negative amplitude is $|4\rangle$

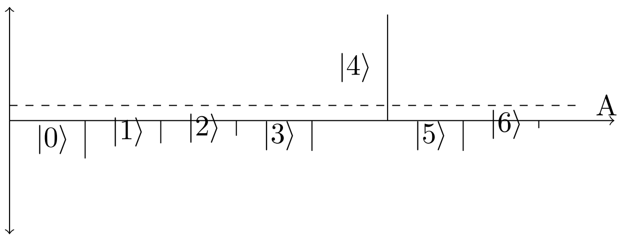

Figure 6: After inversion about average operation is applied. Note that now $|4\rangle$ is the only state whoese amplitude increased while the others have decreased in magnitude. Its also noteworthy, that the amplitudes of all states have changed sign.

Inversion about average We are now going to start first with "Inversion about average". As seen in Fig.5, A denotes the average amplitude over all states, so that, if a_i is the amplitude of the i^{th} state, the average is $\frac{1}{N}\sum_{i=1}^{N} a_i$ Therefore the operation of the grover operator results in an increased (decreased) amplitude of each state by the amount it was below (above) the average before the grover operator was applied. Now we prove that the grover operator really can be seen as an inversion about average. We know the grover operator is of the form Equation 6. According to [5] and [2], one can rewrite the operator to be

$$G = 2P - I$$

where P is the projection operator and I is the identity operator.

and its matrix representation [2] is

$$D = \begin{bmatrix} \frac{2}{N}-1 & \frac{2}{N} & \dots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N}-1 & \dots & \frac{2}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N} & \frac{2}{N} & \dots & \frac{2}{N}-1 \end{bmatrix}$$

To proceed further we should state an important property of the projection operator: P acting on any vector $|\Psi\rangle$ gives a vector $|\tilde{\Psi}\rangle$, whose components are individually equal to the average of all components.

Now we see what happens after applying G to a vector $|\Psi\rangle$

$$\begin{aligned} G|\Psi\rangle &= (2P-I)|\Psi\rangle = 2P|\Psi\rangle - |\Psi\rangle \\ &= \sum_{i=0}^{N} a_i'|i\rangle \qquad \text{with} \qquad a_i' = (\frac{2}{N}-1)a_i + \sum_{i\neq j}\frac{2}{N}a_j \\ a_i' &= -a_i + \sum_{j=0}^{N}\frac{2}{N}a_j \\ a_i' + a_i &= \sum_{j=0}^{N}\frac{2}{N}a_j \\ a_i' + a_i &= 2A \qquad \text{with} \qquad A = \frac{1}{N}\sum_{j=0}^{N} a_j \\ a_i' &= 2A - a_i \end{aligned}$$

by using $a_i = A + \Delta$ we get $a_i' = A - \Delta$ which indeed shows, that the vector was inverted about the average.

Now we want to pay attention to the case, when all except one component are equal to a value l, which is approximately $\frac{1}{\sqrt{N}}$. The other single remaining component is negative. Thanks to the fact that only one amplitude is different the average is close to bein equal to l. Now it becomes clear, that all those components, which are close to being equal to l won't change significantly as a result of the inversion about average, whereas the one component with negative sign now in turn becomes positive and its magnitude increases by approximately 2l.

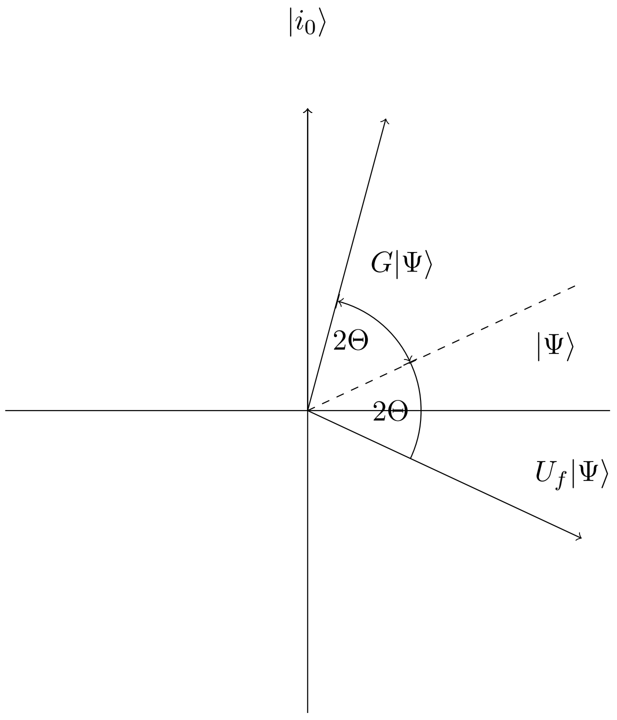

Figure 7: Rotation

Rotation of the vector towards the solution We are now going to examine the geometric interpretation of the actions of the grover operator.
Given the fact that all operators and amplitudes occuring in a grover iteration are real, we know that all states of the quantum computer are located in a real vector subspace of the Hilbert space. This allows us to nicely show the rotation of a vector $|\Psi\rangle$ towards the solution $|i_0\rangle$ by taking $|i_0\rangle$ and $|\Psi\rangle$ as base vectors (non orthogonal basis).
In Fig. (to be included afterwards) it can be seen, that the base vectors $|\Psi\rangle$ and $|i_0\rangle$ form an angle smaller than 90 since the following condition $\langle\Psi|i_0\rangle = \frac{1}{\sqrt{2^n}}$ with $0 < \langle\Psi|i_0\rangle < 1$ holds true. Thus, its obious thaht when theres a large amount of states that the angle will be close to 90.
Before we can start, we need some equations to derive the actions of the operators occuring during one iteration.

The resulting state after application of the oracle can be written as

$$\begin{aligned}|\Psi_o\rangle &= |\Psi\rangle - \frac{2}{\sqrt{2^n}}|i_0\rangle \\ &= |\Psi\rangle - \frac{2}{\sqrt{N}}|i_0\rangle\end{aligned}$$

where $|i_0\rangle$ denotes the searched element. Now $|i_0\rangle$ is a state of the computational basis and as such the following relation holds true

$$\langle\Psi|i_0\rangle = \frac{1}{\sqrt{2^n}} = \frac{1}{\sqrt{N}} \tag{7}$$

[9]

The last thing we need is the effect of the grover operator on the state $|\Psi_1\rangle$. We shall call the resulting state $|\Psi_G\rangle$

$$\begin{aligned}|\Psi_G\rangle &= (2|\Psi\rangle\langle\Psi| - I)|\Psi\rangle \\ &= \frac{2^{n-2}-1}{2^{n-2}}|\Psi\rangle + \frac{2}{\sqrt{2^n}}|i_0\rangle\end{aligned}$$

Before we can finally start our analysis, we have to stress that both, the oracle and the grover operator are unitary operators. They therefor keep the unit norm. From the first two equations we can see, that $|\Psi\rangle$ rotates 2Θ degrees clockwise, as

$$\begin{aligned}\langle\Psi|\Psi_o\rangle &= \langle\Psi|(|\Psi\rangle - \frac{2}{\sqrt{2^n}}|i_0\rangle) \\ &= 1 - \frac{2}{2^n} = 1 - \frac{1}{2^{n-1}} \\ \cos 2\Theta &= \langle\Psi|\Psi_o\rangle = 1 - \frac{1}{2^{n-1}} \end{aligned} \tag{8}$$

Thanks to the second equation one can calculate the angle $2\Theta'$ after applying the grover operator

$$\begin{aligned}\langle\Psi|\Psi_G\rangle &= \langle|(\frac{2^{n-2}-1}{2^{n-2}}|\Psi\rangle + \frac{2}{\sqrt{2^n}}|i_0\rangle) \\ &= \frac{2^{n-2}-1}{2^{n-2}} + \frac{2}{2^n} \\ &= 1 - \frac{1}{2^{n-1}} \\ \cos 2\Theta' &= \langle\Psi|\Psi_G\rangle = 1 - \frac{1}{2^{n-1}}\end{aligned} \tag{9}$$

We see that $2\Theta' = 2\Theta$ and therefore Ψ_o rotates 4Θ counterclockwise towards $|i_0\rangle$.

This result means, that the amplitude of $|i_0\rangle$ increases, while the other amplitudes decrease with respect to their original values in $|\Psi\rangle$. At this point a measurement will return $|i_0$ more likely than it originally would have. One can also see, why you have to apply the grover iteration repeatedly, as the angle 2Θ becomes smaller the more states there are available (as seen in Fig.7).

3.3 Ideal amount of iterations to attain the max probability

Now we want to examine how many times the grover operator needs to be applied to reach the answer with highest possible probability.

According to Fig.7 we get

$$\langle\Psi|i_0\rangle = \frac{1}{\sqrt{2^n}} = \frac{1}{\sqrt{N}} = \cos(\frac{\pi}{2} - \Theta) = \sin\Theta$$

for $|\Psi\rangle$ and $|i_0\rangle$.

We know that we have to get $|\Psi\rangle$ close to $|i_0\rangle$, thus requiring a rotation by an angle of approximately $\frac{\pi}{2}$. Therefore we get a rotation of

$$\Theta + 2k\Theta = \frac{\pi}{2}$$

after k iterations. This leads right to

$$(2k+1)\Theta \cong \frac{\pi}{2}$$

and as the amount of required iterations has to be an integer we have to round the result

$$k = \text{round}(\frac{\pi}{4\Theta} - \frac{1}{2})$$

with $\sin\Theta \approx \Theta = \frac{1}{\sqrt{N}}$ we get

IDEAL AMOUNT OF ITERATIONS

$$k = \frac{\pi}{4}\sqrt{N} \tag{10}$$

iterations to find $|i_0\rangle$ with high probability.

4 Grovers algorithm for a database of size 8

This section is based upon [9]

Here we want to show how Grover's algorithm works for an imaginery database with 8 elements.

4.1 Prequesites

To encode all 8 entries, we need 3 qubits in the first register, whereas an additional qubit is required in the second one to mark the desired element. Using Eq.10, we know that

$$k = \frac{\pi}{4}\sqrt{8} \approx 2 \tag{11}$$

This means we need two iterations to get the searched element with high probability. In case that we would have used a classical computer instead, it would have required us at least 4 queries to attain the searched element with a probability greater than 1/2.

Our imaginery quantum computer will be described in every state (namely $|\Psi_0\rangle, |\Psi_{00}\rangle, |\Psi_1\rangle, |\Psi_{10}\rangle, |\Psi_2\rangle, |\Psi_{20}\rangle$

Initially, the state of the first register is

$$|\Psi_0\rangle = H^{\otimes 3}|000\rangle = \frac{1}{2\sqrt{2}}\sum_{i=0}^{7}|i\rangle \tag{12}$$

which means that the quantum computer is initialized in the required evenly distributed superposition.

4.2 Grovers algorithm at work

Suppose that the desired element is the 5th which would equal $|5\rangle = |101\rangle$ in binary encoding.

In order to evaluate the state of the quantum computer after every computational step, we introduce

$$|u\rangle = \frac{1}{\sqrt{7}}\sum_{i=0 \wedge i\neq 5}^{7}|i\rangle = \frac{|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |110\rangle + |111\rangle}{\sqrt{7}} \tag{13}$$

Using $|u\rangle$, we can rewrite $|\Psi\rangle$ as

$$|\Psi\rangle = \frac{\sqrt{7}}{2\sqrt{2}}|u\rangle + \frac{1}{2\sqrt{2}}|101\rangle$$

At the next step, we apply the oracle operator U_f, which has the following effect

$$U_f(|101\rangle|-\rangle) = -|101\rangle|-\rangle$$
$$U_f(|i\rangle|-\rangle) = |i\rangle|-\rangle, \quad \text{if} \quad i \neq 5$$

Applying the oracle operator on $|\Psi_0\rangle$ results in

$$|\Psi_{10}\rangle|-\rangle = U_f(|\Psi\rangle|-\rangle) = \frac{|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |110\rangle + |111\rangle - |101\rangle}{2\sqrt{2}}|-\rangle$$

Here we can see the oracles effet upon the state $|\Psi\rangle$ as now, theres one element ($|101\rangle$) with a minus sign. Reqriting $|\Psi_{10}\rangle$ using Eq.13 we get

$$|\Psi_{10}\rangle = \frac{\sqrt{7}}{2\sqrt{2}}|u\rangle - \frac{1}{2\sqrt{2}}|101\rangle = |\Psi\rangle - \frac{1}{\sqrt{2}}|101\rangle$$

At the next step we apply the grover operator

$$G = 2|\Psi\rangle\langle\Psi| - I$$

$$\begin{aligned}
|\Psi_{11} &= G|\Psi_{10}\rangle \\
&= (2|\Psi\rangle\langle\Psi| - I)(|\Psi\rangle - \frac{1}{\sqrt{2}}|101\rangle) \\
&= |\Psi\rangle - \frac{2}{\sqrt{2}}\frac{1}{\sqrt{8}}|\Psi\rangle + \frac{1}{\sqrt{2}}|101\rangle \\
&= |\Psi\rangle - \frac{1}{2}|\Psi\rangle + \frac{1}{\sqrt{2}}|101\rangle \\
&= \frac{1}{2}|\Psi\rangle + \frac{1}{\sqrt{2}}|101\rangle \\
&= \frac{\sqrt{7}}{4\sqrt{2}}|u\rangle + \frac{5}{4\sqrt{2}}|101\rangle
\end{aligned}$$

At the begining of our final iteration, we have yet again to apply the oracle operator (note that the second register is ommited, as it always remains in the state $|-\rangle$)

$$|\Psi_{20}\rangle = U_f(|\Psi_{11}\rangle|-\rangle) = \frac{\sqrt{7}}{2\sqrt{2}}|u\rangle - \frac{5}{4\sqrt{2}}|101\rangle$$

Rewriting this expression again by using Eq.13, we have

$$|\Psi_{20} = \frac{1}{2}|\Psi\rangle - \frac{3}{2\sqrt{2}}|101\rangle$$

The last application of the grover operator results in

$$|\Psi_f\rangle = G|\Psi_{20}\rangle = -\frac{\sqrt{7}}{8\sqrt{2}}|u\rangle + \frac{11}{8\sqrt{2}}|101\rangle$$

4.3 The result

Now, we want to check with which probability we would be measuring $|101\rangle$

$$P = \left|\frac{11}{8\sqrt{2}}\right|^2 \approx 0,945$$

We get $|101\rangle$, which corresponds to the 5th element with a probability of around $94,5\%$

5 Implementing grovers algorithm using quantum gates

This section has its foundations in [9]. After all previous considerations one might ask how to implement grovers algorithm on a physical quantum computer.

5.1 Generalized Toffoli Gate

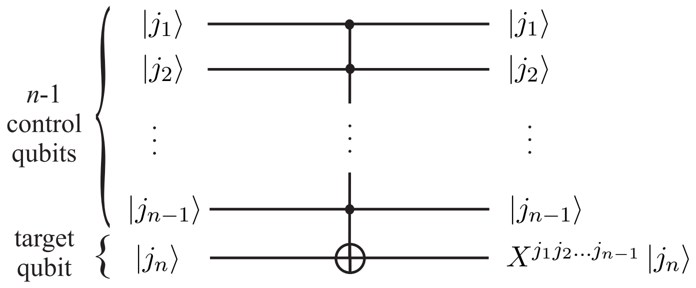

Figure 8: Generalized Toffoli gate.

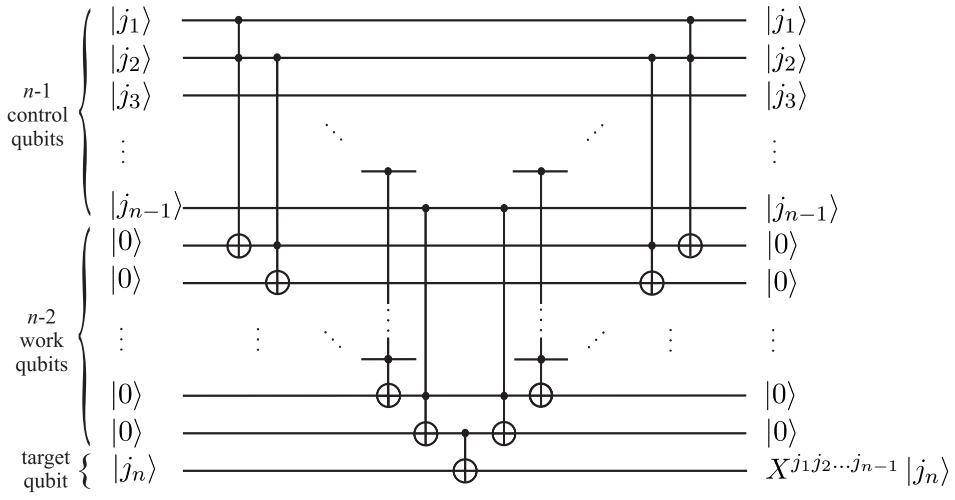

Figure 9: Decomposition of the generalized Toffoli gate in terms of Toffoli gates.

5.2 Circuit for oracle operator

Here one can find the circuit diagram for the oracle operator The given example is an operator working on a 3 qubit system. It selects the state $|101\rangle$. Please mind, that one has to apply two symetrical NOT-Gates, in case the ith binary digit of i_0 is 0. In this case it would be the second qubit, where we have to apply the NOT-Gates.

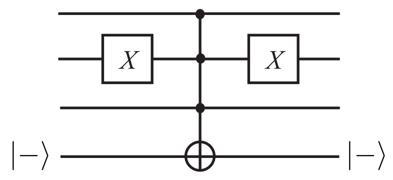

Figure 10: Decomposition of $I - 2\left|101\right\rangle\left\langle 101\right|$, which simulates U_f that searches number 5.

5.3 Circuit for grover operator

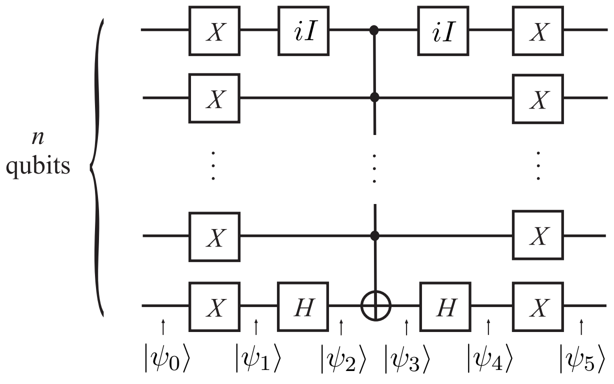

Figure 11: Circuit for $2\left|0\right\rangle\left\langle 0\right| - I$. Note the presence of the imaginary unit, which does not affect the real character of the operator.

5.4 Example using presented gates

Now at last we have a working example using the previousely presented physical implementation of the two operators. This example is a very basic one intended to be very easy to comprehend. The quantum system consists of $N = 2$ and the element we're looking for is $\left|11\right\rangle$.

In order to accomplish the task of easy understandability, we're going to examine the system after every application of a quantum gate.

The circuit diagram looks as

5.4.1 The actual computation

At step one we put the registers into a superposition

$$\left|\Psi_1\right\rangle = \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + \left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + \left|1\right\rangle\right) = \frac{1}{2}\left(\left|00\right\rangle + \left|01\right\rangle + \left|10\right\rangle + \left|11\right\rangle\right)$$

As expected, the two qubits are in a evenly wighted superposition.

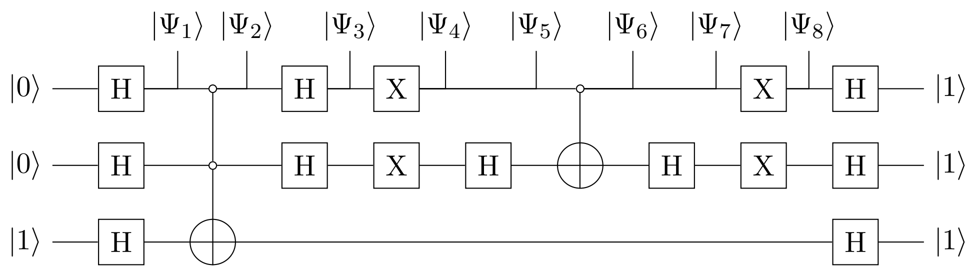

Figure 12: Searching state $|11\rangle$ using quantum gates

Next up is the application of the oracle

$$\begin{aligned}|\Psi_2\rangle &= \frac{1}{2}[(|00\rangle + |01\rangle + |10\rangle)(|0\rangle - |1\rangle) + |11\rangle \\ &\quad (|1\rangle - |0\rangle)] \\ &= \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle - |11\rangle\right)\end{aligned}$$

We see, that the oracle selected the searched element by negating it's amplitude.
Now, we have to apply Hadamard gates to the first 2 qubits

$$\begin{aligned}|\Psi_3\rangle &= \frac{1}{4}[(|0\rangle + |1\rangle)(|0\rangle + |1\rangle) + (|0\rangle + |1\rangle)(|0\rangle - |1\rangle) \\ &\quad + (|0\rangle - |1\rangle)(|0\rangle + |1\rangle) + (|0\rangle - |1\rangle)(|0\rangle - |1\rangle)] \\ &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle)\end{aligned}$$

After applying a NOT gate to $|\Psi_3\rangle$ we have

$$|\Psi_4\rangle = \frac{1}{2}\left(|11\rangle + |10\rangle + |01\rangle - |00\rangle\right)$$

A Hadamard gate now becomes applied to the second qubit

$$\begin{aligned}|\Psi_5\rangle &= \frac{1}{2\sqrt{2}}[|1\rangle(|0\rangle - |1\rangle) + |1\rangle(|0\rangle + |1\rangle) \\ &\quad + |0\rangle(|0\rangle - |1\rangle) - |0\rangle(|0\rangle - |1\rangle)] \\ &= \frac{1}{\sqrt{2}}\left(|10\rangle - |01\rangle\right)\end{aligned}$$

Application of the CNOT gate gives

$$|\Psi_6\rangle = \frac{1}{\sqrt{2}}\left(|11\rangle - |01\rangle\right)$$

Again we have to apply a Hadamard gate to the second qubit

$$\begin{aligned}|\Psi_7\rangle &= \frac{1}{2}[|1\rangle(|0\rangle - |1\rangle) - |0\rangle(|0\rangle - |1\rangle)] \\ &= \frac{1}{2}(|10\rangle - |11\rangle - |00\rangle + |01\rangle)\end{aligned}$$

Negating the qubits using NOT gates lets us have

$$|\Psi_8\rangle = \frac{1}{2}(|01\rangle - |00\rangle - |11\rangle + |10\rangle)$$

We have the final state after applying Hadamard gates to both qubits

$$\begin{aligned}|\Psi_9\rangle = &\frac{1}{4}[(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) - (|0\rangle + |1\rangle)(|0\rangle + |1\rangle) \\ &- (|0\rangle - |1\rangle)(|0\rangle - |1\rangle) + (|0\rangle - |1\rangle)(|0\rangle + |1\rangle)] \\ = &\frac{1}{4}(4|11\rangle) = |11\rangle\end{aligned}$$

5.4.2 The result

As expected for this setup, Grover's algorithm returned the desired element with 100% probability after only one iteration.

6 Grovers algorithm implemented using libquantum

One subject of this bachelor thesis was to show grovers algorithm on a classical computer. During the work on this topic, it was found, that a very good and solid implementation already came along with an open source quantum computation simulation library called libquantum [1]. It was thus chosen to deliver its source code altough a little bit modified to provide a better educational experience along with this bachelor thesis. These modifications consist of various comments in the source code to create a link between the computational implementation and the work done in Sec.5.4.1

Listing 1: Grovers algorithm implemented using *libquantum*

```
/* grover.c: Implementation of Grover's search algorithm

   Copyright 2003 Bjoern Butscher, Hendrik Weimer

   This file is part of libquantum

   libquantum is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published
   by the Free Software Foundation; either version 3 of the License,
    or (at your option) any later version.

    libquantum is distributed in the hope that it will be useful, but
    WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
    General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with libquantum; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
   MA 02110-1301, USA

*/

#include <quantum.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>

#ifdef M_PI
#define pi M_PI
#else
#define pi 3.141592654
#endif


/*
This function effectively represents the oracle operator.
*/
void oracle(int state, quantum_reg *reg)
{
  int i;


  for(i=0;i<reg->width;i++)
    {
        /* this if-conditional makes sure, that for example,
           in case we look for the element 101,
           we end up having 111, simply because
           we couldn't invert the amplitude,
           if this wasn't the case.
           For more informations,
           please refer to the bachelor's
```

```
            thesis Sec 4.2


         */
      if(!(state & (1 << i)))
         {
            quantum_sigma_x(i, reg);
         }
    }

         /* now we have to apply the generalized toffoli gate.
         See more in Sec. 4.1 */
  quantum_toffoli(0, 1, reg->width+1, reg);

  for(i=1;i<reg->width;i++)
    {
      quantum_toffoli(i, reg->width+i, reg->width+i+1, reg);
    }

  quantum_cnot(reg->width+i, reg->width, reg);

  for(i=reg->width-1;i>0;i--)
    {
      quantum_toffoli(i, reg->width+i, reg->width+i+1, reg);
    }

  quantum_toffoli(0, 1, reg->width+1, reg);

  /* again we have to apply the NOT-gate */
  for(i=0;i<reg->width;i++)
    {
      if(!(state & (1 << i)))
         quantum_sigma_x(i, reg);
    }

}


/*
This function represents the grover operator
*/
void inversion(quantum_reg *reg)
{
  int i;
```

```
// we have to apply NOT gates to all qubits
// see more in Sec. 4.4
for(i=0;i<reg->width;i++)
  quantum_sigma_x(i, reg);

//we have to apply a Hadamard gate
//to the last qubit in the first register
quantum_hadamard(reg->width-1, reg);

//here we merely distinguish if we
//have only 3 qubits in our register
//if thats the case we don't need to
//apply the generalized Toffoli gate
//and can use the Toffoli gate instead
if(reg->width==3)
  quantum_toffoli(0, 1, 2, reg);

else
  {
        //here we have to apply the generalized
        //Toffoli gate
    quantum_toffoli(0, 1, reg->width+1, reg);

    for(i=1;i<reg->width-1;i++)
      {
        quantum_toffoli(i, reg->width+i, reg->width+i+1, reg);
      }

    quantum_cnot(reg->width+i, reg->width-1, reg);

    for(i=reg->width-2;i>0;i--)
      {
        quantum_toffoli(i, reg->width+i, reg->width+i+1, reg);
      }

    quantum_toffoli(0, 1, reg->width+1, reg);
  }
//and at after the computation we have
//to make sure we apply the
//Hadamard gate to the last qubit again
quantum_hadamard(reg->width-1, reg);

//same for the NOT gate
for(i=0;i<reg->width;i++)
```

```
    quantum_sigma_x(i, reg);
}

/* this function represents a single
grover iteration */
void grover(int target, quantum_reg *reg)
{
  int i;
  //here we apply the oracle operator
  oracle(target, reg);
  //before we can apply the grover
  //operator itself
  //we have to apply hadamard gates
  //to all qubits first
  for(i=0;i<reg->width;i++)
    quantum_hadamard(i, reg);
  //here we apply the grover operator
  inversion(reg);
  //and here we have again to apply
  //hadamard gates to all
  //qubits
  for(i=0;i<reg->width;i++)
    quantum_hadamard(i, reg);

}
/* the entry C function */
int main(int argc, char **argv)
{
  quantum_reg reg; //this is the quantum register
  int i, N, width=0; //define required
  //variables for loops (i), the searched number
  //and register width

  srand(time(0));

  //get the command line arguments
  if(argc==1)
    {
      printf("Usage: grover [number] [[qubits]]\n\n");
      return 3;
    }
  //set the searched number
  N=atoi(argv[1]);

```

```
//in case the user set the register width
//explicitly , we store that in qidth
if(argc > 2)
  width = atoi(argv[2]);

//now we have to check if the width of
//the quantum register
//allows complete encoding of the
//entered number if thats not the case w
//iden the register appropriately
if(width < quantum_getwidth(N+1))
  width = quantum_getwidth(N+1);

//create a new register initialized
//in the state 0
reg = quantum_new_qureg(0, width);
//invert the last bit so it is in
//the state 1
quantum_sigma_x(reg.width, &reg);

//here we create the evenly distributed
//superposition
for(i=0;i<reg.width;i++)
  quantum_hadamard(i, &reg);

//and apply a hadamard gate to the last
//anquilla qubit to have it in state
// |0> - |1>
quantum_hadamard(reg.width, &reg);

//now we let the user know how many
//iterations it will take to have the final
//result
printf("Iterating %i times\n", (int) (pi/4*sqrt(1<<reg.width)));

//execute the grover iteration the amount
//previously displayed to the user
for(i=1; i<=pi/4*sqrt(1 << reg.width); i++)
  {
    printf("Iteration #%i\n", i);
    grover(N, &reg);
  }

//again apply a hadamard gate to the last
```

```
    //qubit
    quantum_hadamard(reg.width, &reg);

    reg.width++;
    //execute a physical measurement
    quantum_bmeasure(reg.width-1, &reg);

    for(i=0; i<reg.size; i++)
      {
        if(reg.node[i].state == N)
          printf("\nFound %i with a probability of %f\n\n", N,
                 quantum_prob(reg.node[i].amplitude));
      }
    //delete the created quantum register
    //and free memory
    quantum_delete_qureg(&reg);

    return 0;
}
```

7 Possible types of quantum computers

7.1 Solid state quantum computing device

A very young but promising approach has been demonstrated back in May 2009 by the group of L. DiCarlo which in short is quantum computing on a solid state device. This approach is a rather complex one and requires a profound background in order to understand it. As this would go far beyond the scope of this bachelor thesis only a small summarising excerpt of the paper [3] is given:

This device is based on a quantum bus architecture quantum bus architecture which uses an on-chip transmission line cavity to couple, control, and measure qubits. We augment the architecture with flux-bias lines that tune individual qubit frequencies, permitting single-qubit phase gates. By pulsing the qubit frequencies to an avoided crossing where a $\sigma_z \otimes \sigma_z$ interaction turns on, we are able to realize a two-qubit conditional phase (c-Phase) gate. Operation in the strong-dispersive regime of cQED allows joint readout that can efficiently detect two-qubit correlations. Combined with single-qubit rotations, this enables tomography of the two-qubit state. Through an improved understanding of spontaneous emission and careful microwave engineering, we are now able to combine state-of-the-art $\approx 1\mu s$ coherence times into a two-qubit device. This allows sufficient time to concatenate $\approx$ 10 gates, realizing simple algorithms with fidelity greater than 80%

Further details are found in [3].

7.2 NMR quantum computing

This short summary is taken from Wikipedia [10]:
NMR quantum computing uses the spin states of molecules as qubits. NMR differs from other implementations of quantum computers in that it uses an ensemble of systems, in this case molecules. The ensemble is initialized to be in the thermal equilibrium state. Operations are performed on the ensemble through magnetic pulses applied perpendicular to a strong, static field created by a very large magnet.

For further reading, please refer to [8]

7.3 Trapped ion quantum computer

Implementing quantum computers using trapped ions is one of the oldest approaches. A short summary is taken from Wikipedia [11]:
A Trapped ion quantum computer is a type of quantum computer. Ions, or charged atomic particles, can be confined and suspended in free space using electromagnetic fields. Qubits are stored in stable electronic states of each ion, and quantum information can be processed and transferred through the collective quantized motion of the ions in the trap (interacting through the Coulomb force). Lasers are applied to induce coupling between the qubit states (for single qubit operations) or coupling between the internal qubit states and the external motional states (for entanglement between qubits). The fundamental operations of a quantum computer have been demonstrated experimentally with high accuracy (or "high fidelity" in quantum computing language) in trapped ion systems and a strategy has been developed for scaling the system to arbitrarily large numbers of qubits by shuttling ions in an array of ion traps. This makes the trapped ion quantum computer system one of the most promising architectures for a scalable, universal quantum computer.

An exeptional overview is given by the paper of the group of Haeffner [7].

References

[1] Bjoern Butscher, Hendrik W.: *libquantum - C library for quantum computing and quantum simulation.* 2009. – URL `http://libquantum.de/`

[2] Borbely, Eva: *Grover search algorithm.* 2007. – URL `http://www.citebase.org/abstract?id=oai:arXiv.org:0705.4171`

[3] DiCarlo, L. ; Chow, J. M. ; Gambetta, J. M. ; Bishop, Lev S. ; Johnson, B. R. ; Schuster, D. I. ; Majer, J. ; Blais, A. ; Frunzio, L. ; Girvin, S. M. ; Schoelkopf, R. J.: Demonstration of Two-Qubit Algorithms with a Superconducting Quantum Processor. In: *Nature* 460 (2009), S. 240. – URL `doi:10.1038/nature08121`

[4] Ekert, Artur ; Hayden, Patrick ; Inamori, Hitoshi: *Basic concepts in quantum computation.* 2000. – URL `http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0011013`

[5] Grover, Lov K.: *A fast quantum mechanical algorithm for database search.* 1996. – URL `http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/9605043`

[6] Grover, Lov K.: *From Schrdinger's Equation to the Quantum Search Algorithm.* 2001. – URL `http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0109116`

[7] Haeffner, H. ; Roos, C. F. ; Blatt, R.: Quantum computing with trapped ions. In: *Physics Reports* 469 (2008), S. 155. – URL `doi:10.1016/j.physrep.2008.09.003`

[8] Laflamme, R. ; Knill, E. ; Cory, D. G. ; Fortunato, E. M. ; Havel, T. ; Miquel, C. ; Martinez, R. ; Negrevergne, C. ; Ortiz, G. ; Pravia, M. A. ; Sharf, Y. ; Sinha, S. ; Somma, R. ; Viola, L.: *Introduction to NMR Quantum Information Processing.* 2002. – URL `http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0207172`

[9] Lavor, C. ; Manssur, L. R. U. ; Portugal, R.: *Grover's Algorithm: Quantum Database Search.* 2003. – URL `http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0301079`

[10] public, General: *Nuclear magnetic resonance quantum computer.* 2009. – URL `http://en.wikipedia.org/wiki/Nuclear_magnetic_resonance_quantum_computer`

[11] public, General: *Trapped ion quantum computer.* 2009. – URL `http://en.wikipedia.org/wiki/Trapped_ion_quantum_computer`