itp cp

TU Graz

Projektpraktikum

Institut für Theoretische Physik
Computational Physics

Quantencomputer
9 Qbit Fehlerkorrektur nach Shor

Betreuer:
Univ. Prof. Dr. rer. nat. Enrico Arrigoni

Matthias Kaschutnig

Mat.Nr. 0631111

Wintersemester 2010/11

Inhaltsverzeichnis

Abstract

In der Arbeit geht es um den 9 Qbit Shoralgorithmus. Dieser dient zur Korrektur von Fehler die durch die Übertragung von quantenmechanischen Zustände entstehen. Der Algorithmus ist nicht die effizienteste Möglichkeit. Meine Wahl fiel trotzdem darauf weil die Korrektur Schritt für Schritt durchgeführt wird. Der Hauptteil der Arbeit war die Umsetzung einer Simulation, dafür benutzte ich Matlab. In der Arbeit selbst sind die wichtigsten theoretischen Grundlagen und deren Umsetzung im Programm beschrieben.

Motivation

Wie bei allen technischen Anwendungen treten auch im Bereich der Informationsverarbeitung Fehler auf. Da diese nicht verhindert werden können, gibt es Methoden diese zu erkennen und gegebenenfalls zu korrigieren. Im Allgemeinen verwendet man dafür redundante Informationen. Beim klassischen Computer können Daten kopiert werden, dies ist bei Quantencomputern nicht möglich. In diesem Fall nutzt man verschränkte Zustände um Redundanz zu erzeugen.
Die von mir verwendete Datenstruktur hat das Ziel, Übersicht zu gewährleisten und so die Schritte im Programm nachvollziehbar zu machen. Weiters ist sie so strukturiert das sie möglichst flexibel einsetzbar ist.

1 Einführung

Die Ausführungen in diesem Abschnitt beruhen hauptsächlich auf dem Buch Quantum Computing From linear Algebra to physical Realisations. [1]. Dieses Buch bezieht sich auf [3] und [5]

1.1 Mehrteilchensysteme und verschränkte Zustände

Ein wichtiges und häufig genutztes Element von Quantencomputern sind verschränkte Zustände, dieser Abschnitt sollte einen Einblick geben was darunter zu verstehen ist. Quantenmechanische Teilchen werden mittles Hilberträumen beschrieben. Man benötigt für jedes Teilchen einen Hilbertraum. Wenn man Mehrteilchensysteme beschreiben will werden diese Hilberträume mit dem Tensorprodukt verknüpft.

$$H = H_1 \otimes H_2$$

Allgemein wird ein Vektor wie folgt beschrieben.

$$|\varphi\rangle = \sum_{i,j} c_{i,j} |e_{1,i}\rangle \otimes |e_{2,j}\rangle \tag{1}$$

mit $|e_{a,i}\rangle$ (a = 1,2) als orthonormal Basis in H_a und $\Sigma_{i,j}|c_{i,j}|^2 = 1$
Der Vektor $|\varphi\rangle \; \epsilon \; H$ ist das Tensorprodukt aus 2 Vektoren $|\varphi\rangle = |\varphi_1\rangle \otimes |\varphi_2\rangle$ ($|\varphi_a\rangle \; \epsilon \; H_a$).
Dieser Zustand wird Tensorproduktzustand genannt.
Aus klassischer Sicht existieren die Zustände $|\varphi_a\rangle$ nebeneinander. So das die Dimension von H aus der Summe der Teilräume bestehen müsste.

$$dimH = dimH_1 + dimH_2$$

Wenn man nun die Koeffizienten in der Gleichung 1 zählt, ergibt sich für die Dimension H:

$$dimH = dimH_1 \cdot dimH_2 \geqq dimH_1 + dimH_2$$

Man sieht das vor allem bei höher dimensionalen Hilberträumen viele Zustände fehlen.
Beispiel:
Ausgehend vom einen Zustand für Spin $\frac{1}{2}$ - Teilchen.

$$|\varphi\rangle = \frac{1}{\sqrt{2}}(|\uparrow\rangle \otimes |\uparrow\rangle + |\downarrow\rangle \otimes |\downarrow\rangle)$$

Angenommen der Zustand kann faktorisiert werden

$$|\varphi\rangle = (c_1|\uparrow\rangle + c_2|\downarrow\rangle) \otimes (d_1|\uparrow\rangle + d_2|\downarrow\rangle) = c_1 \cdot d_1|\uparrow\rangle|\uparrow\rangle + c_1 \cdot d_2|\uparrow\rangle|\downarrow\rangle + c_2 \cdot d_1|\uparrow\rangle|\downarrow\rangle + c_2 \cdot d_2|\uparrow\rangle|\uparrow\rangle$$

Dies ist nicht möglich da für die Koeffizienten gelten müsste

$$c_1 \cdot d_2 = c_2 \cdot d_1 = 0, c_1 \cdot d_1 = c_2 \cdot d_2 = \frac{1}{\sqrt{2}}$$

Solche Zustände werden verschränkte Zustände genannt. Es sei noch kurz erwähnt das zur Unterscheidung zwischen separierbaren und verschränkten Teilchen die Schmidtdecomposition herangezogen wird (siehe [1] Seite 37 ff).

1.2 Shor Algorithmus

Als Grundlage für den Shor Algorithmus dient der 3 QBit Bitflip und Phaseflip Korrektur Code. Beschreibungen dieser befinden sich in [1]. Der 9 Qbit Algorithmus wird aus diesen Schaltungen aufgebaut. Wie beim klassischen Computer wird zur Fehlerkorrektur redundante Information verwendet. Da quantenmechanische Zustände auf Grund des No Cloning Theorems nicht kopiert werden können, werden zur Realisierung der Redundanz verschränkte Zustände verwendet. Auf die Elemente des Shorcodes gehe ich in dem Abschnitt Methoden 1.4 ein. Eine Gesamtübersicht der simulierten Schaltung ist in Abschnitt 3 .

No Cloning Theorem:
Quantenmechanische Zustände können nicht kopiert werden da ansonsten Ort und Impuls genau gemessen werden könnten, dies würde aber die Heisenberg'sche Unschärferelation verletzten[2].

1.3 Dekohärenz

Kurz erwähnt sei die Dekohärenz welche, neben den Fehlern die durch die Quantengattern selbst entstehen, die unten behandelten Fehler verursacht. Unter Dekohärenz versteht man die Wechselwirkung des betrachteten Systems mit seiner Umgebung. Eine genaue Betrachtung befindet sich in [1] S.173 ff.

1.4 Methoden

1.4.1 Kodierung und Datenstruktur

Das zu übertragende Qbit $|\varphi\rangle = a|0\rangle + b|1\rangle$ ist im Programm wie folgt dargestellt:

$$\begin{bmatrix} a & b \\ 0.0 & 1.0 \end{bmatrix} \tag{2}$$

Die weiteren berechneten Matrizen beziehen sich auf den Zustand $|\varphi\rangle$ mit den Koeffizienten a = 0.240 und b = 0.942. Die nun verwendeten Funktionen (Cnot,Hadamard) werden unten erläutert. An dieser Stelle möchte ich nur auf die verwendete Datenstruktur, mit Hilfe der Kodierung, eingehen. Der erste Schritt bei der Kodierung wird für die Korrektur des Bitflips benötigt (siehe Abb. 1).

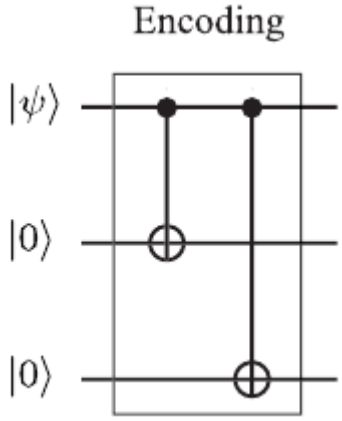

Abbildung 1: Kodierung für die Korrektur des Bitfehlers [1]

Nach den in Abbildung 1 durchgeführten Operationen entsteht ein verschränkter Zustand

$$|\varphi'\rangle = 0.240|000\rangle + 0.942|111\rangle$$

Dieser Zustand wird GHZ-Zustand [1] [4] (Greenberger-Horne-Zeilinger) genannt. Im Programm wird der Zustand wie in der Matrix 3 dargestellt:

$$|\varphi'\rangle = 0.240 \cdot |000\rangle + 0.942 \cdot |111\rangle = \begin{bmatrix} 0.240 & 0.942 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \tag{3}$$

An dieser Stelle sei zu erwähnen das die Matrix nur der Ausgabe entspricht. Im Programm selbst werden, aus Gründen der Flexibilität, Zellen verwendet. Der in Matrix 3 angeführte Zustand hat folgenden Aufbau:

$$\begin{bmatrix} [2x1double] & [2x1double] \\ [0] & [1] \\ [0] & [1] \end{bmatrix} \tag{4}$$

Diese Struktur kann mit dem Matlabbefehl "cell" erzeugt werden. Dies ist eine Datenstruktur in der unterschiedlich große Matrizen zusammengefasst werden können. Im konkreten Beispiel verbirgt sich hinter [2x1 double] der Zustand $0.240 \cdot |0\rangle$ (1. Spalte) bzw. $0.942 \cdot |1\rangle$ (2. Spalte). Bei Elementen mit der Größe n * n, mit $n > 1$, wird nur die Anzahl der Elemente und der verwendete Datentyp angezeigt. In den Zeilen 2 und 3 sind die verschränkten Zustände abgebildet. So das der Zustand 3 repräsentiert wird.

Die Schaltung in Abbildung 2 zeigt die vollständige Kodierung des 9 Qbit Shoralgorithmus. Der 1. Teil der Schaltung repräsentiert die Bitfehlerkorrektur (vergleiche Abbildung 1). Der Teil ab den Hadamardgates dient zur Kodierung die für die Phasenfehlerkorrektur benötigt wird.

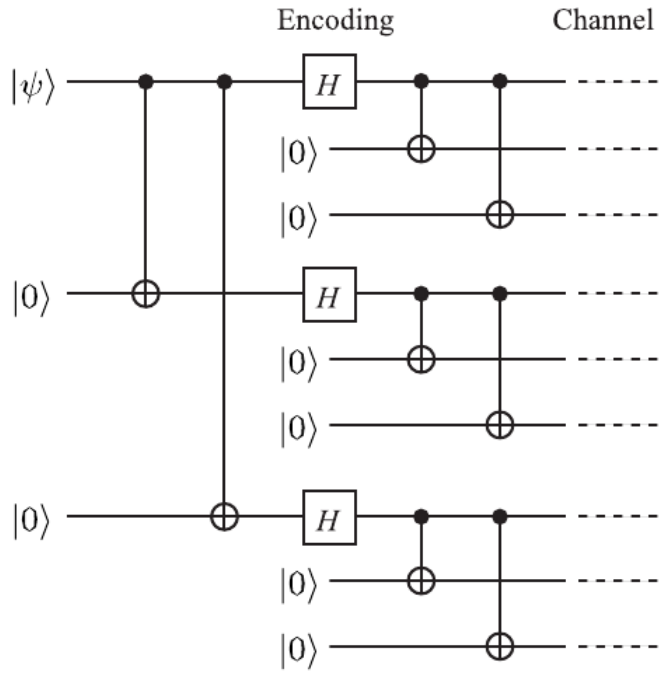

Abbildung 2: Kodierung für Bit und Phasenfehler [1]

In der Matrix 5 dargestellte Zustand ist, im Sinne der Schaltung in Abbildung 2, vollständig kodiert. d.h. die Information ist redundant und die Basis ist nun $|\pm\rangle$. Analytisch nimmt dieser folgende Form an:

$$|\varphi\text{“}\rangle = \frac{0.240}{\sqrt{8}}|+++\rangle + \frac{0.942}{\sqrt{8}}|---\rangle$$

Das in Abbildung 2 vorkommende Hadamard Gate verursacht die in Matrix 5 zu sehenden zusätzlichen Spalten.

$$\begin{bmatrix} 0.170 & 0.170 & 0.666 & -0.666 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.707 & 0.707 & 0.707 & -0.707 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.707 & 0.707 & 0.707 & -0.707 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \end{bmatrix} \tag{5}$$

1.4.2 Beschreibung der Funktionen

Die Funktionen entsprechen den verwendeten Quantenmechanischen Operatoren

1.4.3 controlled-Not

Das controlled Not Gatter (Cnot) ist ein 2 Qbit Gatter. Wobei das Qbit welches als Controllbit (i) vorgesehen ist, auf das Target (j) wirkt.
Definition[1]:

Allgemein:

$$|i\rangle|j\rangle \rightarrow |i\rangle|i \oplus j\rangle$$

mit i,j ϵ {0,1} und $|i \oplus j\rangle = |(i+j)_{mod2}\rangle$
z.B.: $0 \oplus 1 = 1, 1 \oplus 1 = 0$
Für die Fehlerkorrektur ist die folgende Betrachtung ausreichend

$$|j0\rangle \rightarrow |jj\rangle$$

mit j ϵ {0,1}

Der Funktionsaufruf in Matlab ist:

cnot(qbit,controll,target)

qbit $\cdots$ Matrix
controll $\cdots$ Steuerungsbit
target $\cdots$ zu verändernde Bit

Wie aus der obigen Beschreibung der Datenstruktur ersichtlich wirkt das Cnot auf die gesamte Zeilen mit Ausnahme der Koeffizienten. Die Zeilen werden beim Funktionsaufruf festgelegt (controll,target). Die Berechnung wird Spaltenweise der Definition entsprechend durchgeführt.

Beispiel:
Ausgehend von den Zustand: 6.

$$|\varphi\rangle = 0.240 \cdot |000\rangle + 0.942 \cdot |100\rangle = \begin{bmatrix} 0.240 & 0.942 \\ 0.0 & 1.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \end{bmatrix} \tag{6}$$

wird die Berechnung durchgeführt

$$\mathrm{Cnot}_{13}\mathrm{Cnot}_{12}|\varphi\rangle = 0.240 \cdot |000\rangle + 0.942 \cdot |111\rangle$$

Die Darstellung im Program ist in Matrix 7 ersichtlich.

$$\begin{bmatrix} 0.240 & 0.942 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \tag{7}$$

1.4.4 Hadamard

Die Definition des Hadamardgates [1]

H :

$$|0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$|1\rangle \to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Bei meiner Wahl der Datenstruktur wirkt die Operation auf die Spalten der einzelnen Matrizen. In Abhängigkeit des ursprünglichen Bits werden zusätzliche Spalten eingefügt. Die Vorzeichen werden den Koeffizienten zugeordnet.
Beim Funktionsaufruf gestaltet sich folgend:

hadamard(qbit,pos)

qbit $\cdots$ das zu bearbeitende Qbit
pos $\cdots$ Vektor der angibt wo der Hadamardoperator angewendet werden soll z.B: [1:9]

Beispiel: Ausgehend von den Zustand:

$$|\varphi\rangle = 0.240 \cdot |000\rangle + 0.942 \cdot |111\rangle = \begin{bmatrix} 0.240 & 0.942 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \tag{8}$$

wird nun folgende Operation durchgeführt

$|\bar{\varphi}\rangle = (H \otimes I \otimes I)|\varphi\rangle = \frac{0.240}{\sqrt{2}} \cdot (|0\rangle + |1\rangle)|00\rangle + \frac{0.942}{\sqrt{2}} \cdot (|0\rangle - |1\rangle)|11\rangle$

Dies wird im Programm wie folgt abgebildet.

$$\begin{bmatrix} 0.170 & 0.170 & 0.666 & -0.666 \\ 0.0 & 1.0 & 0.0 & 1.0 \end{bmatrix} \tag{9}$$
$$\begin{bmatrix} 0.0 & 1.0 \end{bmatrix}$$
$$\begin{bmatrix} 0.0 & 1.0 \end{bmatrix}$$

Dahinter steht die schon oben erwähnte Zellenstruktur

$$\begin{bmatrix} [2x2double] & [2x2double] \\ [0] & [1] \\ [0] & [1] \end{bmatrix} \tag{10}$$

Hier zeigt sich eine Stärke der Datenstruktur, diese ermöglicht es auf Teile des Zustandes zu zugreifen und mathematische Operationen durchzuführen. Dabei muss nicht darauf geachtet werden, in wie weit der Zustand geändert wird da die Teile in getrennten Matrizen aufgeteilt ist, somit sind sie in ihrer Größe (n x n) unabhängig. Durch die übergeordnete Zelle wird der gezielte Zugriff ermöglicht.

1.4.5 U Gate

Das U Gate dient zum Einbau eines kontinuierlichen Fehlers. Dabei handelt es sich um einen Bitfehler der durch einen kontinuierlichen Parameter α charaterisiert ist [1]. Definition:

$$U_\alpha := e^{i\cdot\alpha\cdot X} = cos(\alpha \cdot I) + i \cdot sin(\alpha \cdot X)$$

Funktionsaufruf in Matlab:

ualpha(qbit,alpha,pos,bit)

qbit
alpha $\cdots$ Winkel
pos $\cdots$ Vektor der den zu bearbeitenden Teil des Qbits angibt z.B.: [1:3]
bit $\cdots$ Bitflip z.B: $1 \rightarrow (U_\alpha \otimes I \otimes I), 2 \rightarrow (I \otimes U_\alpha \otimes I)$

Beispiel:

Ausgehend von

$$|\varphi\rangle = 0.170 \cdot |+\rangle + 0.666 \cdot |-\rangle = \begin{bmatrix} 0.170 & 0.170 & 0.666 & -0.666 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \end{bmatrix} \tag{11}$$

wird nun $(I \otimes U_\alpha \otimes I)$ mit $\alpha = \frac{\pi}{5}$ auf $|\varphi\rangle$ angewandt führt dies zum Ergebnis:

$$\begin{bmatrix} 0.137 & 0.137 & 0.539 & -0.539 & 0 + 0.0998\,i & 0 + 0.0998\,i & 0 + 0.392\,i & 0 - 0.392\,i \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{12}$$

2 Berechnungen und Ergebnisse

2.1 Mögliche Fehler und deren Auswirkung auf die Matrix

In diesem Abschnitt (Basierend auf [1] bzw. [6]) möchte ich auf die möglichen Fehler eingehen bzw. anhand von Beispielen zeigen wie sich Fehler auf die Matrix auswirken. Zum Vergleich kann die Matrix 5 herangezogen werden. Die Fehler werden in dem Programm manuell eingebaut.

2.1.1 Bitfehler

Unter einen Bitfehler versteht man ein Änderung des Zustandes von zB.: $|0\rangle \rightarrow |1\rangle$ dies entspricht der Paulimatrix σ_x der durch die Übertragen von einen Qbit mit der Wahrscheinlichkeit p auftritt. In Abbildung 3 [1] ist die Korrekturschaltung zu sehen. Eine Gesamtübersicht befindet sich im Anhang in Abbildung 7.

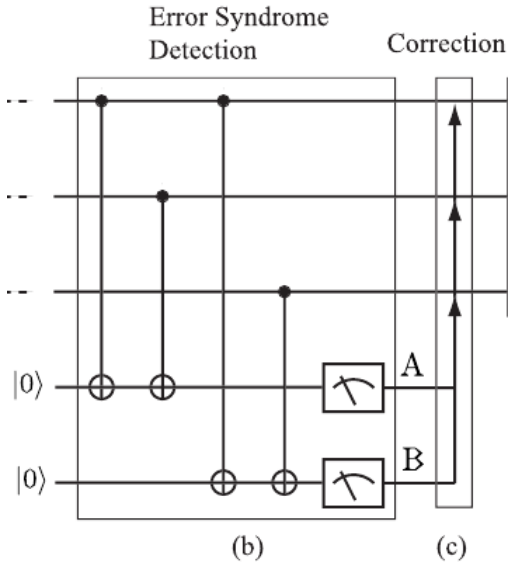

Abbildung 3: Bitkorrekturschaltung [1]

Die mit A bzw. B bezeichneten zusätzlichen Qubits werden zur Fehlererkennung verwendet. Auf Grund der 4 Cnot Operationen ergibt sich die in Abbildung 4 gezeigte Abhängigkeit.

States after error extraction is made and the probabilities with which these states are produced.

State after error syndrome extraction	Probability
$(a\|000\rangle + b\|111\rangle)\|00\rangle$	$(1-p)^3$
$(a\|100\rangle + b\|011\rangle)\|11\rangle$	$p(1-p)^2$
$(a\|010\rangle + b\|101\rangle)\|10\rangle$	$p(1-p)^2$
$(a\|001\rangle + b\|110\rangle)\|01\rangle$	$p(1-p)^2$
$(a\|110\rangle + b\|001\rangle)\|01\rangle$	$p^2(1-p)$
$(a\|101\rangle + b\|010\rangle)\|10\rangle$	$p^2(1-p)$
$(a\|011\rangle + b\|100\rangle)\|11\rangle$	$p^2(1-p)$
$(a\|111\rangle + b\|000\rangle)\|00\rangle$	p^3

Abbildung 4: Fehlersyndrome und Wahrscheinlichkeiten für die auftretenden Fehler [1]

Beispiel:
Zum Vergleich kann Matrix 5 herangezogen werden.
Fehlerbehafteter Zustand Analytisch:

$$|\varphi\rangle = 0.085 \cdot |+\rangle|+\rangle(|010\rangle + |101\rangle) + 0.333 \cdot |-\rangle|-\rangle(|010\rangle - |101\rangle)$$

Fehlerbehafteter Zustand im Programm:

$$\begin{bmatrix} 0.170 & 0.170 & 0.666 & -0.666 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.707 & 0.707 & 0.707 & -0.707 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.707 & 0.707 & 0.707 & -0.707 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \end{bmatrix} \tag{13}$$

2.1.2 Phasenfehler

Der Phasenfehler ist ein Vorzeichenwechsel (σ_z). Zur Korrektur wird der in Gleichung 14 beschriebene Zusammenhang genutzt.

$$H\sigma_z H = \sigma_x \tag{14}$$

Dies bedeutet das der Phasenfehler in der Basis $|0\rangle$ und $|1\rangle$ ein Bitfehler in der Basis $|+\rangle$ und $|-\rangle$ ist.
mit $|\pm\rangle = \frac{1}{\sqrt{2}} \cdot (|0\rangle \pm |1\rangle)$

In Abbildung 5 ist die Phasenkorrektur für eine 3 Qbit Schaltung zu sehen. Es werden wie auch beim Shorcode 2 zusätzliche Qbits zu Fehlererkennung verwendet. Die Abhängigkeiten sind in Abbildung 6 dargestellt.

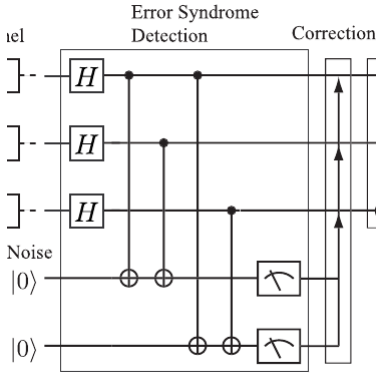

Abbildung 5: 3 Bit Phasenkorrektur [1]

States after error extraction is made.

Input to the detecting circuit	Ancillas state $\|A_4, B_4\rangle$
$a\|++\rangle + b\|--\rangle$	$\|00\rangle$
$a\|-+\rangle + b\|+-\rangle$	$\|11\rangle$
$a\|+-\rangle + b\|-+\rangle$	$\|10\rangle$
$a\|++\rangle + b\|--\rangle$	$\|01\rangle$
$a\|--\rangle + b\|++\rangle$	$\|01\rangle$
$a\|-+\rangle + b\|+-\rangle$	$\|10\rangle$
$a\|+-\rangle + b\|-+\rangle$	$\|11\rangle$
$a\|--\rangle + b\|++\rangle$	$\|00\rangle$

Abbildung 6: Fehlersyndrome für die auftretenden Phasenfehler [1]

Beispiel:

Fehlerbehafteter Zustand Analytisch

$$|\varphi\rangle = 0.240 \cdot |-\rangle|+\rangle|+\rangle + 0.941 \cdot |+\rangle|-\rangle|-\rangle$$

Fehlerbehafteter Zustand im Programm

Der entsprechende Eingriff ist in der Matrix 15 in den Zeilen 1 bis 4 zu entnehmen. Dabei zu beachten ist das sich nur die Vorzeichen bei den Koeffizienten ändern, die Basisvekoren bleiben unberührt.

$$\begin{bmatrix} 0.170 & -0.170 & 0.666 & 0.666 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.707 & 0.707 & 0.707 & -0.707 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.707 & 0.707 & 0.707 & -0.707 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \end{bmatrix} \tag{15}$$

Es kann auch ein Bit und Phasenfehler gleichzeitig auftreten dies entspricht der Matrix $-i\sigma_y$. Auch dieser Fehler kann korrigiert werden.
Da gilt:

$$-i\sigma_y = \sigma_x\sigma_z$$

Mit Hilfe des Algorithmus kann jeder 1 Qubit Fehler korrigiert werden.

2.1.3 Kontinuierlicher Fehler

Wenn ein kontinuierlicher Fehler auftritt hat der Zustand z.B: die Form

$$(U_\alpha \otimes I \otimes I)(a|000\rangle + b|111\rangle) = \cos\alpha(a|000\rangle + b|111\rangle) + i\sin\alpha(a|100\rangle + b|011\rangle)$$

Die Korrektur derartiger Fehler geschieht bei der Messung der Fehlersyndrome im Rahmen der Bitfehlerkorrektur. Vor der Messung gestaltet sich der Fehler wie folgt:

$$\cos\alpha(a|000\rangle + b|111\rangle)|00\rangle + i\sin\alpha(a|100\rangle + b|011\rangle)|11\rangle$$

Das Ergebnis der Messung ist entweder 00 oder 11. Die Wahrscheinlichkeit für 00 ist mit p $\cos^2\alpha$ gegeben. In diesem Fall kollabiert der Zustand zu $a|000\rangle + b|111\rangle$. Im Fall 11 zerfällt der Zustand zu $a|100\rangle + b|011\rangle$. Wie in der Abbildung 4 ersichtlich wird nun das erste Qbit korrigiert.
Anschließend muss der Zustand noch dekodiert werden. Die entsprechende Schaltung befindet sich im Anhang (Abbildung 8).

3 Zusammenfassung

Der Algorithmus kann Bit- und Phasenfehler korrigieren. Diese können auch gleichzeitig auftreten. Wobei die Korrektur auf einen Bitfehler beschränkt ist, dieser Umstand stellt hohe Anforderungen an die Umsetzung von Quantencomputern, da die Wahrscheinlichkeit für einen Bitflip gering gehalten werden muss. Weiters können auch kontinuierliche Fehler im Rahmen der Bitfehlerkorrektur korrigiert werden.

Literatur

[1] Quantum Computing From linear Algebra to physical Realisations
Mikio Nakahara and Tetsuo Ohmi
A Taylor and Francisbook

[2] Vorlesungsskript Quantencomputer Eine Einführung
Verfasst von: Enrico Arrigoni, Roland Peckl, Stefan Rossegger

[3] Quantum Computation and Quantum Information
Michael A. Nielsen and Isaac L. Chuang
Cambridge University Press

[4] D. M. Greenberger, M. A. Horne and A. Zeilinger, in 'Bell's Theorem, Quantum Theory, and Conceptions of the Universe, ed. M. Kafatos, Kluwer, Dordrecht (1989). Also avilable as arXiv:0712.0921 [quant-ph].

[5] E. Riefell and W. Polak, ACM Computing Surveys (CSUR) 32, 300 (2000).

[6] A. M. Steane, eprint, quant-ph/0304016 (2003).

Anhang

In Abbildung 7 [1] ist eine Gesamtübersicht der Korrekturschaltung zu sehen.

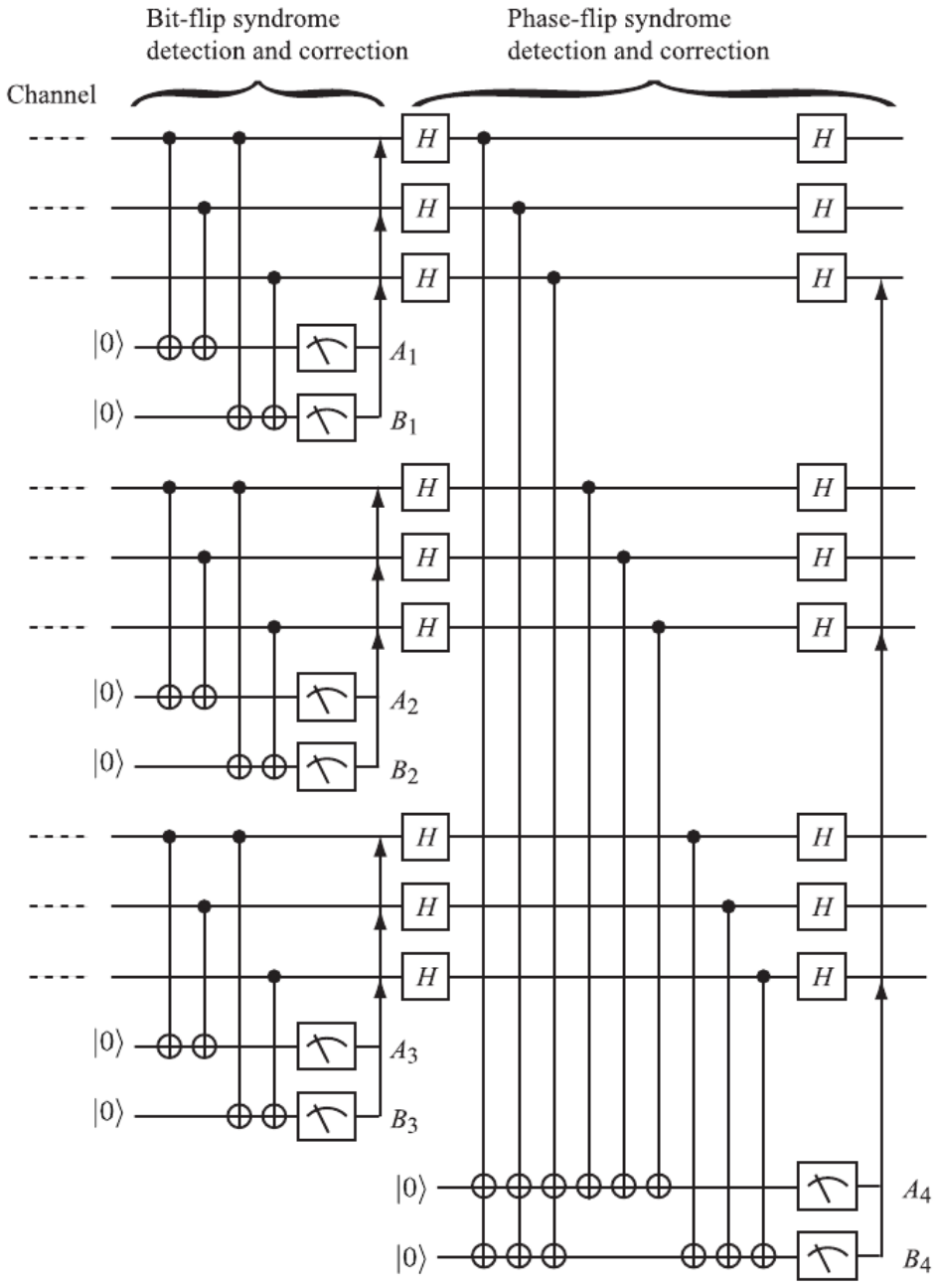

Abbildung 7: Korrektur [1]

Die in Abbildung 8 [1] gezeigte Schaltung dekodiert den zuvor korrigierten Zustand. Es sei zu beachten das es sich nur um die invertierte Kodierung handelt

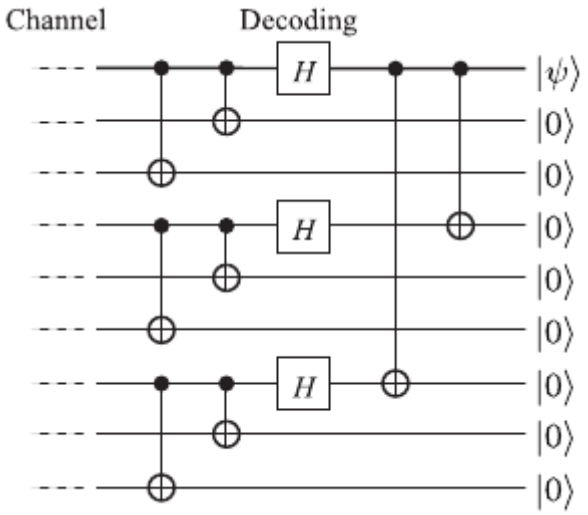

Abbildung 8: Dekodierung [1]