Ausarbeitung Quantenrechner
Grundlagen und Anwendungen

Jan Kinder
24.03.2003

1. Inhalt

2. Einleitung

Im Gegensatz zu den bisherigen Kryptografischen Verfahren benutzen sogenannte Quantenrechner physikalische Gesetze der "Ungenauigkeit" um Sicherheit zu gewährleisten und riesige Rechenleistungen zu vollbringen. Im folgenden möchte ich die Prinzipien und den Aufbau dieser Rechner beschreiben und einen kurzen Überblick über die Leistungsmöglichkeiten geben. Einen tieferen Blick in die sichere Kommunikation mittels quantenphysikalischer Techniken gibt der Vortrag und die Ausarbeitung von Björn Karge.
Vielfach läßt sich das Verständnis für eine neuartige Technologie nur verstehen, wenn man Vergleiche zu klassischen Modellen aufzeigt. Dies vermittelt zum einen die grundlegenden Unterschiede als auch die gemeinsamen Grundlagen. Daher werde ich stets einen Bezug zu den heute geläufigen Rechnersystemen behalten.

3. Grundlagen

3.1. der klassische Rechner

Die heutige Rechnerarchitektur geht auf Johann von Neumann zurück. Er verfaßte in den 1950ger Jahren grundlegende Arbeiten über den inneren Aufbau von Rechenmaschinen. Dabei ist sowohl bei der Software wie bei der Hardware eine Unterteilung sinnvoll. Die Software ist in Programm und Daten zu unterteilen. Diese Trennung wurde in frühen Rechnern auch durch die Hardware festgelegt, als es noch getrennte Programm- und Datenspeicher gab. Ein Grund lag sicher in den technischen Möglichkeiten, da der Programmspeicher möglichst schnell sein sollte. Später faßte man beide Speicher zusammen und überließ es dem Programmierer des Betriebssystems Programme und Daten in getrennten Arealen des Speichers unterzubringen. Die Grundeinheit stellt dabei das Bit dar. Ein Bit kann exklusiv die Werte 0 oder 1 annehmen. Eine wichtige Größe der Beschreibung des Speicher ist die sogenannte Wortgröße. Sie gibt an wie viele Bits zu einer Dateneinheit zusammengefaßt werden. In den Anfangsjahren des Computerzeitalters schwankte dies stark, so gab es 4, 8, 10 oder 12-Bitrechner. Moderne Rechner arbeiten mit 32 oder 64 Bit für ein Register. Damit haben wir auch schon den ersten wichtigen Hardwarebaustein beschrieben, den Speicher. Der zweite wichtige Baustein ist der Prozessor. Ihn kann man wiederum in eine Controling-Einheit und eine Computing-Einheit zerlegen. Letztere ist für die eigentlichen Rechenleitungen verantwortlich. Sie enthält die Gatter für Addition, Multiplikatio und die Logikoperatoren, wie zum Beispiel ADD, NOT oder NAND. Dabei lassen sich die arithmetischen Gatter als spezielle Logikgatter interpretieren. So ist ADD als XOR zu verstehen. Die Controling-Einheit ist für die Programmabfolge zuständig. Sie lädt die Befehle, füttert die Computing-Einheit mit den Operanden, initiiert die Ausführung der richtigen Operation und schreibt die Ergebnisse zurück, um dann den nächsten Befehl zu laden. Die dritte wichtige Komponente ist die Peripherie: über sie kann mit dem Rechensystem kommuniziert werden. Im allgemeinen gehören hierzu ein Bildschirm, eine Tastatur, ein Drucker und seit einigen Jahren eine Maus. Im allgemeinen werden externe Speicher wie Festplatten und Floppy-Disk ebenfalls zur Peripherie gezählt.

3.2. der Quantenrechner

Auch der Quantenrechner besitzt die 3 Hardwarekomponenten Speicher, Prozessor und Peripherie. Ihr innerer Aufbau und die Funktion unterscheiden sich jedoch.

3.2.1.Der Speicher

Kommen wir zuerst zum Speicher. Bestand der Speicher im klassischen System aus einzelnen Bits, so wird er jetzt durch QBits gebildet. Primär kann ein QBit ebenfalls die Werte 0 und 1 annehmen. Die Werte können sich allerdings, solange das QBit unbeobachtet ist, überlagern. Das heißt, dass das QBit sich in Beiden Zuständen befinden kann. Erst eine Messung legt es auf einen der Zustände fest. Beschreiben läßt sich der Zustand des QBits durch eine komplexe Zahl, deren Betrag auf 1 normiert wird. Der reelle bzw. komplexe Anteil der Zahl gibt dann die Wahrscheinlichkeit an, mit der eine 1oder 0 gemessen wird. (Siehe Abb. 1)

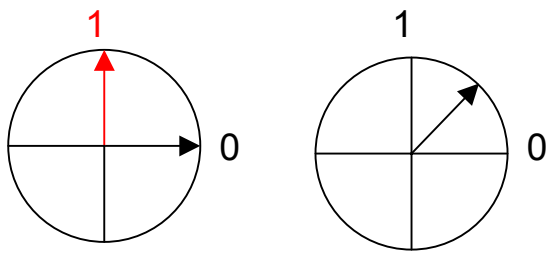

Abb. 1

Die Werte 0 und 1 stehen dabei senkrecht aufeinander . Für die Notation wählen wir dabei folgende Schreibweise:

$$\begin{pmatrix}1\\0\end{pmatrix} = |0> \qquad \begin{pmatrix}0\\1\end{pmatrix} = |1>$$

Damit läßt sich jeder Zustand Z des QBits als $Z = c_0 * |0> + c_1 * |1>$ schreiben. Wobei $c_0^2 + c_1^2 = 1$ gilt.

Ein QBit reicht für eine Berechnung natürlich nicht aus, sondern wir benötigen eine Kombination mehrerer QBits zu einem QBit-Register. Dieses besteht aus mehreren miteinander verketteter QBits, so daß eine Veränderung/Manipulation an einem der QBits eine Veränderung aller anderen QBits zur Folge hat. Dabei können sich alle QBits überlagern. Den Zustand Z eines solchen Quantenregisters beschreibt dann eine Gleichung der folgenden Art. Hier für ein 3 QBit-Register:

$Z = c_{000} |000> + c_{100} |100> + c_{010} |010> + c_{110} |110> + c_{001} |001> + c_{101} |101> + c_{011} |011> + c_{111} |111>$

3.2.2. Prozessor und Operatoren

Kommen wir nun zum "Prozessor". Die eigentlichen Berechnungen werden durch Warten erreicht, da uns die Quantengesetze die Lösung "frei Haus" liefern. Die einzige und wichtige Aufgabe des Prozessors ist also die Programmablaufsteuerung. Dabei ist zu beachten, daß die Berechnungen vorwärts und rückwärts gleichermaßen geschehen und die Ergebnisse erst im letzten Schritt vorliegen. Dazu ist dem Datenteil des Quantenregisters noch einen Programmzählerteil hinzuzufügen. Erst wenn der Programmzähler den letzten Schritt erreicht hat, kann das Ergebnis ausgelesen werden.
Stellt sich die Frage welche Operatoren können verwendet werden. Die in klassischen Rechnern verwendeten Gatter sind im allgemeinen nicht reversibel, das heißt aus ihrem Ergebnis läßt sich nicht auf die Ausgangsdaten schließen z.B. das AND-Gatter:

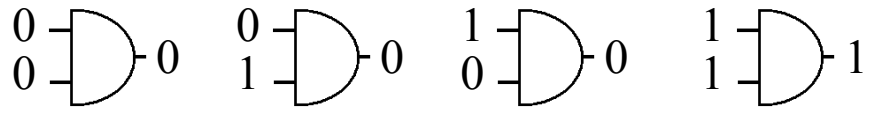

Abb.2

Der Ergebniswert 0 kann nicht eindeutig einer Ausgangskonfiguration zugeordnet werden. Dies liegt natürlich daran, das die Anzahl der Ausgangsbits nicht der Anzahl der Eingangsbits entspricht. Da Berechnungen in Quantencomputern sowohl vorwärts als auch rückwärts vor sich gehen, ist ein solches Gatter nicht so einfach verwendbar. Wir benötigen also reversible Gatter. Ein klassisches reversibles Gatter ist das NOT (Abb. 3). Ein anderes reversibles Gatter ist "Copying-XOR". Dabei wird ein Bit kopiert und das andere Bit bildet das Ergebnis (Abb. 4). Da zusätzliche Bits ignoriert werden können, ist es nicht schlimm, daß wir mehr Informationen behalten.

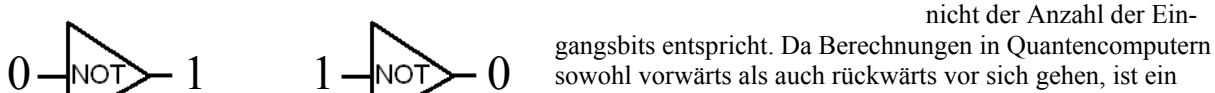

Abb. 3

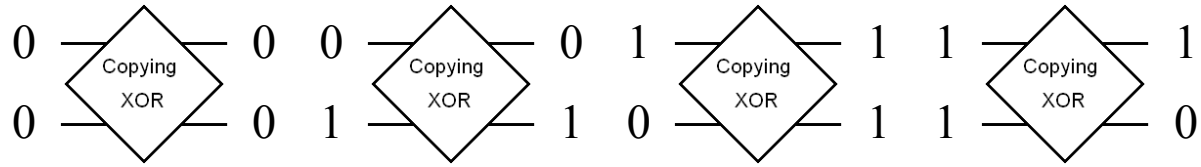

Abb. 4

Auch die anderen klassischen Gatter lassen sich ebenfalls in reversiblen Gattern interpretieren bzw. aus solchen Gattern zusammensetzen. Hier die wichtigen AND + NAND in Abb. 5.

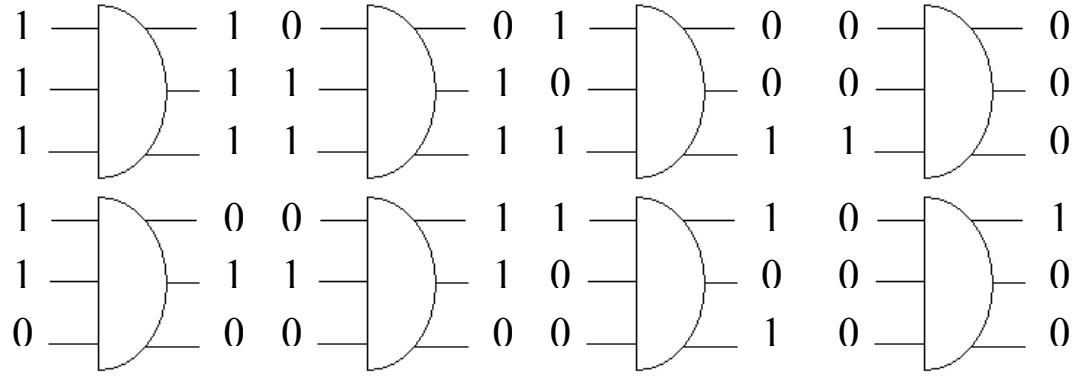

Abb. 5

Die beiden oberen Eingänge sind dabei die Operanden A und B der untere Eingang legt die Operation fest: eine 1 für AND und eine 0 für NAND. Das Ergebnis ist dann im obersten Ausgangsbit zu finden. Die unteren Ausgangsbits sind nur für die Eineindeutiglkeit und könnten verworfen werden. Damit stehen für Quantenoperationen alle klassischen Gatter zur Verfügung.

Jedes reversible Gatter läßt sich als unitäre Matrix schreiben. Das heißt die durch das Gatter ausgeführte Funktion ist identisch der durch die Matrix dargestellten linearen Funktion. Die Eigenschaft unitär ist Folgerung aus der Invertierbarkeit und der Normierung auf Betrag 1. Konkret bedeutet dies, das die Matrix M und ihre komplex-konjugiert-transponierte Matrix M* zueinander invers sind, also als Produkt die Einheitsmatrix ergeben. Da im klassischen Fall die Koeffizienten c_i stets 0 oder 1 sein müssen, ergibt sich eine sehr eingeschränkte Anzahl von Operationsmatrizen. Es können ja nur die ganzzahligen unitären Matrizen eingesetzt werden. Als Beispiel möchte ich die Matrix von NOT angeben:

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

Und hier die Anwendung auf 1= $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ und 0 = $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$

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

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

Abb. 6

Für QBits gilt diese Einschränkung nicht. Damit stehen dem Quantenrechner unendlich mehr Operatoren zur Verfügung. Davon sind die meisten allerdings nicht besonders sinnvoll.
Eine der zusätzlichen Matrizen möchte ich mit (Wurzel NOT) bezeichnen. Sie hat folgende Gestalt:

$$\begin{pmatrix} \frac{1+i}{2} & \frac{1-i}{2} \\ \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix}$$

Und es gilt:

$$\begin{pmatrix} \frac{1+i}{2} & \frac{1-i}{2} \\ \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix}\begin{pmatrix} \frac{1+i}{2} & \frac{1-i}{2} \\ \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix}=\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \text{NOT}$$

Abb. 7

Für diese Operation gibt es kein klassisches Äquivalent.

3.2.3. Peripherie

Da sich aktuell Quantenrechner noch in der Phase der Pilotprojektierung befinden, ist die Peripherie nicht so umfangreich notwendig wie beim klassischen Rechner. Die wichtigste Peripherie-Komponente ist der externe Speicher. Er ist von klassischer Gestalt und enthält neben der Ausgangsdaten auch das "Programm". Es wird in Form einer Manipulationsmatrix auf einem klassischen Rechner berechnet und durch Einwirkung mittels geeigneter physikalischer Apparaturen auf die als Quantenregister fungierenden Objekte ausgeführt. Siehe auch Kapitel 5.

4. Eine "einfache" Berechnung

4.1. Vorüberlegungen

Zuerst benötigen wir eine leichte einfach zu überprüfende Aufgabe. Dazu wählen wir (Wurzel NOT) mal (Wurzel NOT) = NOT. Das einzige benötigte Gatter ist unter Abb. 7 dargestellt. Da wie oben schon beschrieben noch ein Programmzähler benötigt wird, müssen wir dieses Gatter noch anpassen. Zuerst ist die benötigte QBit-Anzahl zu berechnen. Ein- und Ausgabe benötigen ein QBit, da uns die Eingabe zum Schluß nicht mehr interessiert, können wir das Eingabe-Bit auch als Ausgabe-Bit nutzen. Da es 2 Berechnungsschritte sind, haben wir 3 Programmzustände: Initial, nach der ersten Berechnung und nach der zweiten Berechnung. Damit muß unser Quantenregister 4 QBits umfassen. Das unter Abb7. Dargestellt Gatter arbeitet aber nur auf einem 1-QBit-Register. Daher müssen wir die Matrix so verändern, daß das Gatter auf einem 4-QBit-Register arbeitet, aber die ersten 3 QBits nicht verändert. Die Lösung wird mathematisch durch das Tensorprodukt beschrieben. Wir erhalten:

$$\sqrt{NOT}[4,4] = id_2 \otimes id_2 \otimes id_2 \otimes \sqrt{NOT}$$

Abb. 8

Wobei id_2 die Identitätsmatrix = Einheitsmatrix für 2-wertige Vektoren ist. Das Ergebnis ist die Matrix aus Abbildung 9.

$$\begin{pmatrix}
\frac{1+i}{2} & \frac{1-i}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1-i}{2} & \frac{1+i}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2}
\end{pmatrix}$$

Abb. 9

4.2. Berechnung des Hamilton-Operators H

Bis jetzt haben wir die statische Betrachtungen gemacht. Nun müssen wir Dynamik ins Spiel bringen. Dazu betrachten wir zunächst die Schrödinger Gleichung (Abb. 10):

$$ih\frac{\partial|\psi(t)\rangle}{\partial t} = H|\psi(t)\rangle$$

Abb. 10

$$|\psi(t)\rangle = e^{-\frac{iHt}{h}}|\psi(0)\rangle = U(t)|\psi(0)\rangle$$

Abb. 11

Dabei sind: i die Imaginäre Einheit (Wurzel aus -1), h die normierte Plank'sche Konstante ≈ 10-34 Js, $|\psi(t)\rangle$ der Zustand des Systems zum Zeitpunkt t und H der Hamiltonsche Operator. Die Lösung dieser Differentialgleichung lautet (Abb. 11):
Dabei ist U(t) die Überführungsmatrix. U ist dabei zeitabhängig. Das Problem ist herauszufinden, wann U der geforderten Matrix $\sqrt{NOT}[4,4]$ entspricht. Dazu haben wir die Programmzähler-QBits eingefügt. Wenn wir die Matrix so anpassen, daß erstens stets nur eines der Programmzähler-QBits 1 ist und zweitens der Programmzähler bei jedem Rechenschritt um eins nach rechts geshiftet wird, haben wir das Problem auf das Auslesen der Programmzähler-QBits zurückgeführt. Da die Berechnung vorwärts und rückwärts erfolgt, muß dieses Bit in beiden Richtungen erzeugt und vernichtet werden. Damit ergibt sich für H folgende Matrix:

$$H = \sum_{i=0..k}\left(c_{i+1}a_iM_{i+1}\right)+\left(c_{i+1}a_iM_{i+1}\right)^*$$

Abb. 12

Dabei sind: c_i der Erzeuger des i-ten Bits (Abb. 13) , a_i der Vernichter der i-ten Bits (Abb. 14) und M_i die für den i-ten Berechnungsschritt benötigte Matrix. In unserem Fall also $\sqrt{NOT}[4,4]$.

Das Sternchen steht dabei für die komplex konjugierte Matrix. Die Festlegung $\left(e^{-\frac{iH}{\hbar}}\right)^t = U(t)$ aus Abb.11 läßt uns damit U(t) berechnen.

$$\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\otimes\begin{pmatrix}0 & 0\\ 1 & 0\end{pmatrix}\otimes\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\otimes\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$$

Abb. 13 Erzeuger c_2

$$\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\otimes\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\otimes\begin{pmatrix}0 & 1\\ 0 & 0\end{pmatrix}\otimes\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$$

Abb. 14 Annulator a_3

4.3. Anwendung des "Programms"

Die bisherigen Berechnungen sind dem Programmieren eines klassischen Rechners äquivalent. Nun muß das Programm zur Ausführung gebracht werden. Dazu muß das Quantenregister mit dem Startwert geladen werden. Dies ist entweder |1001> falls NOT 1 berechnet werden soll oder |1000> falls NOT 0 berechnet werden soll. Dabei stellen die ersten drei QBit den Startwert für den Programmzähler und das letzte QBit das eigentliche Datum dar. Nun wird mittels geeigneter Instrumente die Umgebung des Quantenregisters so eingestellt, daß es sich nach dem berechneten Hamilton-Operator entwickelt. Dann muß "gewartet" werden. Die benötigte Zeit ist allerdings extrem kurz, so daß trotz der heute technisch möglichen Zeitspanne von nur wenigen Microsekunden, in denen das System im Quantenzustand gehalten werden kann bereits tausende Operationen möglich sind. "Von Zeit zu Zeit" müssen die Programmzähler-QBits ausgelesen werden. Hier gibt es 3 Möglichkeiten:

1. Es wird 100 ausgelesen, dann befindet sich das System im Ausgangszustand,
2. Es wird 010 ausgelesen, dann befindet sich das System zwischen der Ersten und 2. Berechnung. Das Daten-QBit befindet sich in Superposition, da es nicht ausgelesen wurde, kann die Berechnung ohne Einschränkungen fortgesetzt werden. Verallgemeinert ist dies ein Zwischenzeitpunkt. Falls mehr als ein Daten-QBit verwendet wird, befinden sich alle diese in einer Superposition, die mit dieser Cursor-Position vereinbar ist.
3. Es wird 001 ausgelesen, dann ist die Berechnung beendet und das Daten-Qbit enthält bzw. die Daten-QBits enthalten die Antwort und kann ausgelesen werden.

Wie man leicht erkennt, werden für diese Berechnung recht viele QBits benötigt. Um ein Programm mit n Schritten auszuführen, benötigt man schon n+1 QBits für die Cursor sowie weitere m Daten-QBits. Daher ist es wichtig die Programme so kurz wie möglich zu halten.

5. Technische Umsetzung

Aktuell gibt es mehrere Ansätze. Erfolgversprechend sind dabei die NMR-Technologie, dabei wird der Spin eines Elektrons als QBit genutzt, die Ionen-Falle, hierbei werden Ionen auf knapp über Null Kelvin abgekühlt und können so ein QBit repräsentieren. Oder das Polymolekülverfahren , das hier etwas näher beschrieben werden soll.

Betrachtet man einzelne "Elektronenbahnen" in Atomen oder Molekülen stellt man fest, das bestimmte Bahnen stabil und andere instabil sind. Zu zwei stabilen niederenergetichen Bahnen sucht man sich eine hochenergetische instabile Bahn so heraus, das ein auf dieses instabile Niveau gehobenes Elektron entweder auf die eine oder die andere stabile Bahn zurückfällt. (siehe Abb. 15). Das Anheben des Elektron kann mittel Laserlicht gesteuert

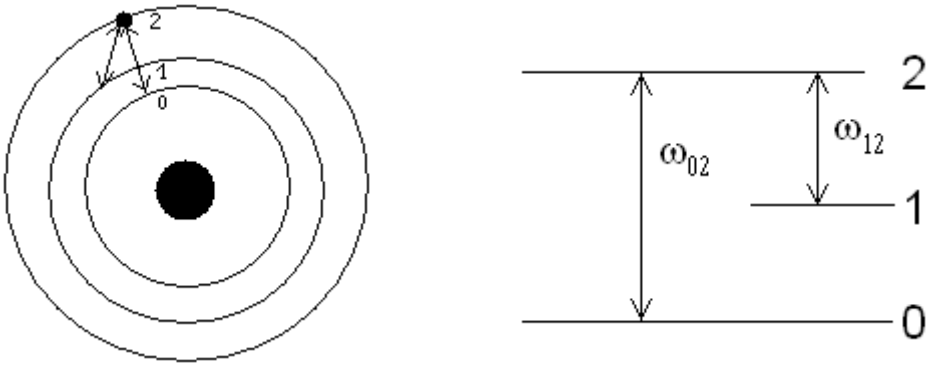

Abb. 15

werden. Unter Laserlicht der Wellenlänge ω_{02} wird ein auf das Niveau 0 zurückfallendes Elektron sofort wieder auf das Niveau 2 angehoben. Damit bleibt als einzig stabiler Zustand der Status 1 übrig. Ist der Impuls jedoch kurz genug, oder wird kurzzeitig Laserlicht beider Wellenlängen ausgesandt und dann gleichzeitig abgestellt, ist der Zustand des Elektrons ungewiß - es befindet sich in Superposition. Da sich die Orbitale der Elektronen in Molekülen überlagern, ändert sich das Energieniveau des Elektrons, sobald sich das Niveau eines Elektrons in

benachbarten Atomen verändert, ebenfalls minimal. Durch genaue Messung lassen sich diese Unterschiede erfassen. Durch geeignete Laser lassen sich auch diese feinen Unterschiede adressieren. Damit können die QBits miteinander interagieren. Für ein 2-QBit-Register gibt Abbildung 15 die Energieniveaus an.

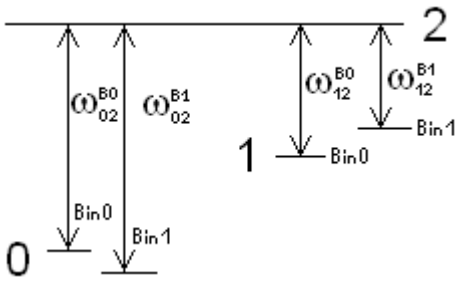

Abb. 15

a =	Log Fkt	Pulsfolge:
0	0	$[\omega_{A21}^{B0}, \omega_{A21}^{B1}]$
1	1	$[\omega_{A20}^{B0}, \omega_{A20}^{B1}]$
a	Identität	No Pulse
b	Lade mit b	$[\omega_{A20}^{B1}, \omega_{A21}^{B0}]$
NOT (a)	Not	$[\omega_{B20}^{A1}, \omega_{B21}^{A0}]$; $[\omega_{A20}^{B1}, \omega_{A21}^{B0}]$
a+b	XOR	ω_{A20}^{B1}
a*b	AND	ω_{A21}^{B0}

Die bis jetzt maximale Anzahl der auf diese Art miteinander verschränkten QBits ist 5. Damit stößt diese Methode auch schon fast an ihre Grenzen. Hauptproblem ist die Interaktion mit der Umwelt. Je mehr QBits miteinander verschränkt werden, desto größer wird der Einfluß der Umgebung und desto kleiner wird die zur Berechnung zur Verfügung stehende Zeit.

6. Reale Anwendungen und Ausblick

Das diese Art zu rechnen nicht nur ein unsinniges Spielvergnügen sind, sondern reale Anwendungen bringen, beweisen die Algorithmen von Grover (schnelle Datenbanksuche) und Shor (Primzahlzerlegung). Letzteren werde ich jetzt kurz skizzieren:

6.1. Shor's Algorithmus zur Primzahlzerlegung

Zuerst ein paar mathematische Grundlagen. Wenn $N = n_1 n_2$ zu faktorisieren ist, und x eine zufällige zu N relativ prime Zahl ist (sonst finden wir leicht einen Faktor) so hat die Funktion $f(k) = x^k \bmod N$ eine Periode r, d.h. $f(k) = f(k+r)$. Es gilt also $x^r \equiv 1 \bmod N$. Für gerade r, können wir $y = x^{r/2}$ finden mit $y^2 \equiv 1 \bmod N$. Der Chinesische Restsatz gibt uns nun y_1, y_2, y_3 und y_4 die das Kongruenzsystem A:
$y_1 \equiv 1 \bmod n_1$, $y_1 \equiv 1 \bmod n_2$
$y_2 \equiv 1 \bmod n_1$, $y_2 \equiv 1 \bmod n_2$
$y_3 \equiv 1 \bmod n_1$, $y_3 \equiv 1 \bmod n_2$
$y_4 \equiv 1 \bmod n_1$, $y_4 \equiv 1 \bmod n_2$
erfüllen. Neben den Trivialen Lösungen +1 und -1 gibt es noch die nichttrivialen Lösungen a und -a. findet man nun eine dieser Lösungen, so hat a+1 oder a-1 einen gemeinsamen Teiler mit N, da $a^2 \equiv 1 \bmod N$ also $a^2 - 1 = (a+1)*(a-1) = c*N$. Da der ggT leicht in polynomieller Zeit zu berechnen ist, genügt es also eine solches x und die Periode r in polynomieller Zeit zu berechnen.
Shor fand dazu folgenden Algorithmus:

0. Es wird ein Quantenregister der Größe $(2^n, 2^n)$ mit $2^n > N$ benötigt.
1. Erzeuge eine Superposition aller Werte $0,\ldots,2^n$ im ersten Register. Dazu zum Beispiel die Hadamard-Transformation benutzen.
2. Die Funktion $x^k \bmod N$ auf den ersten Teil anwenden und das Ergebnis im 2. speichern.
3. Den 2. Teil messen, dadurch wird die Gesamt-Superposition gestört. Und es bleiben nur Werte der Form rj+s übrig derart das $x^{rj+s} \equiv x^s \bmod N$
4. Anwenden der diskreten Fourier-Transformation auf die erste Registerhälfte. Dadurch wird dieses in das Frequenzspektrum umgewandelt und der unbekannte Offset s entfernt. Nun sollte ein Vielfaches der Periode r in der ersten Registerhälfte stehen.
5. Wiederholtes Anwenden von 1-4 gibt ausreichend viele $j_i r$ um r in Polynomieller Zeit zu berechnen.

6.2. Ausblick

Die Quantencomputertechnologie steckt noch in den Kinderschuhen. Die technische Umsetzung erweist sich momentan noch als sehr schwierig. Sollten diese Probleme in den nächsten Jahrzehnten in den Griff bekommen werden, so wird sich das Bild dieser Erde ein weiteres mal ändern. Die Quantenrechner werden vermutlich nicht so schnell die Wohnzimmer und Büroräume erobern, für einfache Office-Anwendungen sind sie nicht geeignet, aber sie werden das Internet revolutionieren. Neue Sicherheitskonzepte erfordern aber auch liefern. (Siehe dazu auch den Vortrag von Björn Karge.)

7. Quellen

C. Williams und S.Clearwater: Explorations in quantum computing, Springer Verlag, 1998