HSK

Grundlagen Quantum Computing

Skript V1.2

Wolfgang Maier
1.7.2024

Inhaltsverzeichnis

Grundlagen Physik

Um ein solides Verständnis über die Funktionsweise eines Quanten-Computers zu erarbeiten ist es erforderlich einige grundlegende Methodiken der Physik zu beleuchten. Zunächst wollen wir uns vor Augen führen dass das Ziel physikalischer Betrachtungen darin liegt die durch unsere Sinne wahrgenommene Umgebung auf eine möglichst einfache und nachvollziehbare Weise zu beschreiben. Die Methodik um dies zu erreichen lässt sich kurz zusammenfassen. Die beobachtbaren physikalischen Entitäten, also alle Gegenstände die unsere Welt ausmachen, werden zunächst durch naheliegende Abstraktionen die ihre wesentlichen Grundzüge wiederspiegeln, angenähert. Dann wird ein mathematischer Formalismus entworfen der die Eigenschaften der gewählten Abstraktion möglichst quantifizierbar beschreibt. Dieser Ansatz ist auch in anderen Wissensgebieten eine gängige Methode und begegnet uns in unserem täglichen Leben auf Schritt und Tritt. So wird ein einfaches Metallstück durch entsprechende Prägung zu einem Zahlungsmittel abstrahiert und durch ein formales Währungssystem geregelt welche Kaufkraft diesem Metallstück zugeschrieben wird. Genau so gut könnte man dieses Metallstück aber auch als Zufallsgenerator abstrahieren um mithilfe geeigneter Elemente der Wahrscheinlichkeitsrechnung zu beschreiben welche Ergebnisse bei einem oder mehreren Münzwürfen zu erwarten sind. Mit genau dieser Vorgehensweise werden wir uns auch an das Thema Quantum Computing annähern. Wie nicht anders zu erwarten ist die physikalische Entität mit der wir uns beschäftigen werden ein Quantenobjekt, ein sogenanntes qBit, welches wir als eine digitale Informationseinheit abstrahieren. Unsere Aufgabe wird es also sein den erforderlichen Formalismus zu verstehen der dieses qBit umfassend beschreibt und seine Eigenschaften in quantifizierbarer Weise zu neuen Rechenansätzen verfügbar macht. Da qBits Quantenobjekte sind gilt es also zunächst einmal eine Ahnung zu bekommen was denn ein Quantenobjekt überhaupt ist. Dazu müssen wir uns zunächst ein wenig mit der Entwicklung der Physik in den letzten Jahrhunderten auseinandersetzen.

Klassische Physik vs Quantenphysik

Wie oben bereits beschrieben geht es in der Physik seit jeher um eine mathematische Beschreibung unserer Umgebung. Eine der anschaulichsten Teilgebiete der Physik ist dabei die Mechanik, die die Bewegung von Gegenständen unter der Einwirkung von Kräften beschreibt. Bereits im 17. Jahrhundert gelang es Newton die Grundlagen der Mechanik in einen Formalismus zu fassen. Dabei stützte er sich auf die von Leibniz entwickelte Differentialrechnung. Unserem Schema folgend werden dabei Gegenstände vereinfacht als Massepunkte abstrahiert welche durch auf sie einwirkende Kräfte auf eine Bahn gezwungen werden. Diese Bahn lässt sich, zumindest theoretisch, durch das zweite Newton'sche Axiom eindeutig berechnen. Newtons Formel lautet:

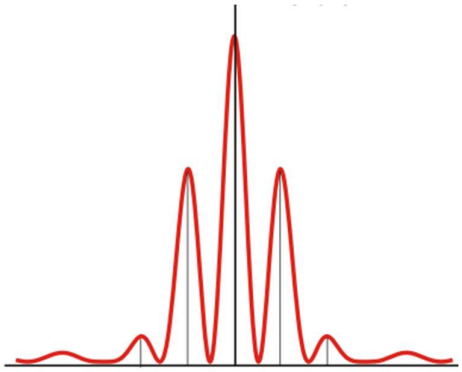
$$F(\vec{r}) = m\frac{d^2\vec{r}}{dt^2}$$

Dies ist eine Differentialgleichung 2.Ordnung die unter vernünftigen Startbedingungen mathematisch beweisbar immer eine eindeutige Lösung hat. Zumindest für dieses Teilgebiet ist damit im klassischen Sinne ein deterministisches Weltbild definiert, das zum Beispiel auch die Bewegungsbahnen unseres Planetensystems erklärt. Ein einfacher Spezialfall dieser Gleichung für eine konstant einwirkende Kraft F ist dabei die bekannte Formel **F=ma.** Mit diesen Betrachtungen war aus klassischer Sicht das physikalische Weltbild bezüglich der Bewegungsmechanik prinzipiell

abgeschlossen. Allerdings konnte ein weiteres relevantes Phänomen der Mechanik, nämlich die Ausbreitung von Licht, damit nicht beschrieben werden. Hierzu war es nötig einen anderen Ansatz als den von Massepunkten zu wählen. Die Lösung lag in der von Huygens formulierten Wellentheorie die als zweiter Eckpfeiler der klassischen Physik betrachtet werden kann. Die Wellengleichung lautet:

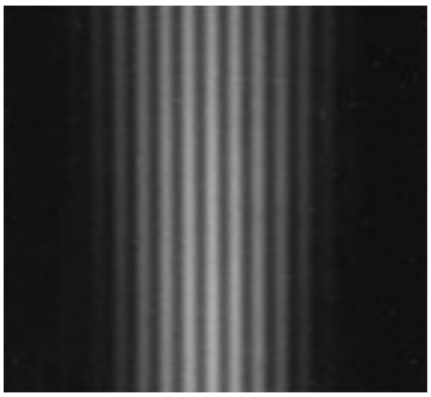
$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right)$$

Dies ist eine partielle Differentialgleichung in Raum und Zeit. Das interessanteste Experiment zur Wellenausbreitung ist die Brechung einer ebenen Wellenfront an einem Doppelspalt. Sie ergibt ein von der Wellenlänge und dem Spaltabstand charakteristisches Interferenzmuster das nach dem Huygensschen Prinzip bestimmt werden kann.



Damit war seit dem 17. Jahrhundert das physikalische Weltbild zunächst abgeschlossen. Es existieren prinzipiell 2 unterscheidbare natürliche Phänomene. Zum einen Teilchen, zum anderen Wellen, die durch die hier kurz skizzierten Formalismen beschrieben werden können.

Nachdem im 19. Jahrhundert immer mehr Erkenntnisse zur Elektrizität und zum Magnetismus erlangt worden waren konnten aufgrund der damit verbundenen technischen Möglichkeiten Experimente realisiert werden die sich auf mikroskopische Strukturen fokussierten. Es wurde immer klarer dass neue Ansätze erforderlich waren um passende Beschreibungsmodelle zu finden die die neuen Beobachtungen erklären konnten. Die in den chemischen und physikalischen Experimenten untersuchte Materie wurde in ersten Ansätzen durch kleinste atomare Teilchen modelliert deren Ausdehnung unterhalb der Auflösungsgrenze eines Lichtmikroskops liegt. Erstaunlicherweise zeigten diese Objekte sowohl Teilcheneigenschaften als auch Welleneigenschaften. Eines der Experimente die diesen Sachverhalt beweisen ist das Doppelspalt Experiment mit Elektronenstrahlen das in den 1950er Jahren von Möllenstedt und Jönsson durchgeführt wurde. Dabei ergab sich wie auf der folgenden Abbildung einer hinter dem Spalt positionierten Photoplatte eine vergleichbare Intensitätsverteilung wie sie oben für Licht gezeigt wurde.



Ebenso wie für Elektronenstrahlen Welleneigenschaften beobachtet werden konnten war es in anderen Experimenten möglich Effekte zu beobachten die Anlass dazu gaben dass es sich bei Licht aus einem Teilchenstrom aus so genannten Photonen handelt. Der Photoeffekt beschreibt dabei wie ein Photon aufgrund seines Impulses durch einen elastischen Stoß ein Elektron aus einer Materieoberfläche herauslösen kann. Dieser Umstand wird in der Physik als Welle-Teilchen Dualismus bezeichnet und durch den Formalismus der Quantenmechanik aufgelöst. Die grundlegende Gleichung der Quantenmechanik ist die Schrödinger Gleichung:

$$i\hbar\frac{\partial}{\partial t}\psi = -\frac{\hbar^2}{2m}\Delta\psi + V\psi$$

Diese ist eine partielle Differentialgleichung in Raum und Zeit. Die komplexe Funktion $\Psi(\vec{r},t)$, die diese Gleichung auflöst wird gedeutet als eine Materiewelle, die die Aufenthaltswahrscheinlichkeit eines Teilchens beschreibt.

Wir können nun den sehr anschaulichen Fall des harmonischen Oszillators betrachten. Und zwar einmal für den Fall des klassischen Ansatzes nach Newton und einmal für den Fall der quantenmechanischen Betrachtung nach Schrödinger. Ein eindimensionaler harmonischer Oszillator wird charakterisiert durch seine Potentialfunktion:

$$V(x) = \frac{1}{2}\,kx^2$$

Dies entspricht im klassischen Fall einem Massepunkt der mit einer linearen Kraft zu seinem Ursprung zurückgezogen wird wenn er sich von diesem entfernt. Seine potentielle Energie entspricht dann der parabelförmigen Funktion V(x) wie oben beschrieben. Für einen Massepunkt der Masse m ergibt sich im klassischen Fall für eine maximale Auslenkung u die Lösung:

$$x(t) = u\,sin(\omega_0 t + \varphi)$$

$$\omega_0^2 = \frac{k}{m}$$

Also eine harmonische Schwingung deren Frequenz von der Masse m und der Konstante k abhängig ist.

In der quantenmechanischen Betrachtung sieht die Lösung wesentlich komplizierter aus. Die Materiewellenfunktionen lauten:

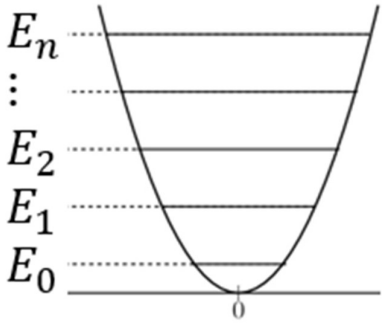
$$\psi_n(x) = \left(\frac{1}{\pi b^2}\right)^{\frac{1}{4}} \frac{1}{\sqrt{2^n n!}}\,H_n(x)e^{-\frac{x^2}{2}}$$

$$H_n(x) = (-1)^n \, e^{x^2} \frac{d^n}{dx^n} \left(e^{-x^2}\right)$$

Aus dieser komplizierten Lösung lassen sich nun nur bestimmte, also sozusagen gequantelte Energiewerte erzeugen:

$$E_n = \hbar\omega \left(n + \frac{1}{2}\right)$$

Man sieht also, dass im Gegensatz zum klassischen Fall bei dem durch entsprechende Wahl der Auslenkungsamplitude u jede kontinuierlich wählbare Energie in das System gesteckt werden kann im quantenmechanischen Fall nur ganz fest definierte Energieniveaus möglich sind:



qBits

Wie wir bereits erwähnt haben sind qBits, die im Zentrum unserer Betrachtungen stehen, Quantenobjekte. Durch die vorangegangenen Erläuterungen haben wir erörtert welche Erkenntnisse dazu geführt haben einen quantenmechanischen Formalismus zu definieren. Wir wollen nun etwas spezifischer auf die Eigenschaften von qBits schauen. Prinzipiell können qBits durch verschiedenste physikalische Implementierungen realisiert werden. Was aber alle diese Implementierungen eint verrät bereits der Name. In der klassischen Informationstechnologie wird ein Bit als die kleinste Informationseinheit betrachtet. Sie erlaubt es zwei gleich wahrscheinliche Alternativen zu unterscheiden und wir bezeichnen diese Alternativen üblicherweise mit 0 und 1. Mithilfe des Leibniz'schen Binärsystems lassen sich beliebige Zahlen binär formulieren. Außerdem können logische Verknüpfungen durch Boole'schen Operatoren auf einem solchen binären System realisiert werden. Ein qBit wird durch 2 unterscheidbare Quantenzustände realisiert. In unserem Beispiel des quantenmechanischen Oszillators könnten dies zum Beispiel die Zustände mit den Energieniveaus E_0 und E_1 sein. Da es sich um quantenmechanische Zustände handelt müssen wir uns etwas genauer anschauen wie diese Zustände beschrieben werden. Für die Darstellung von qBits gibt es zwei gleichwertige Möglichkeiten, die je nach Problemstellung von Vorteil sind.

Bra Ket Darstellung

In der Quantenmechanik ist es üblich die sogenannte Bra Ket Schreibweise zu benutzen. Sie wird in Anlehnung an ihren Erfinder auch als Dirac Notation bezeichnet. Ein Zustand eines qBits x wird dabei

durch ein sogenanntes Ket dargestellt. Eine aus den quantenmechanischen Eigenschaften des qBits resultierende Besonderheit ist der Umstand dass sich das qBit mit normierten Wahrscheinlichkeiten gleichzeitig in seinen beiden Basiszuständen aufhalten kann:

$$|x\rangle = \alpha|0\rangle + \beta|1\rangle \quad \text{mit} \quad |\alpha|^2 + |\beta|^2 = 1 \quad \alpha,\beta \in \mathbb{C}$$

Man nennt diese Eigenschaft Superposition. Im Gegensatz zu einem klassischen Bit ist somit eine Bestimmung des qBits nicht eindeutig. Man kann den Zustand des qBits nur durch Messung bestimmen. Bei dieser Messung wird das qBit allerdings in einen der beiden Zustände |0> oder |1> übergehen. Man misst dann |0> mit der Wahrscheinlichkeit $|\alpha|^2$ und |1> mit der Wahrscheinlichkeit $|\beta|^2$. Will man also die Wahrscheinlichkeit bestimmen sind mehrere Messungen notwendig.

Beispiele:

A: $|x\rangle = \frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}\,|1\rangle$

Messung:
|0> Wahrscheinlichkeit $^1/_3$
|1> Wahrscheinlichkeit $^2/_3$

B: $|x\rangle = \frac{1}{\sqrt{3}}|0\rangle - \sqrt{\frac{2}{3}}\,|1\rangle$

Messung:
|0> Wahrscheinlichkeit $^1/_3$
|1> Wahrscheinlichkeit $^2/_3$

C: $|x\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

Messung:
|0> Wahrscheinlichkeit $^1/_2$
|1> Wahrscheinlichkeit $^1/_2$

D: $|x\rangle = 1|0\rangle + 0|1\rangle$

Messung:
|0> Wahrscheinlichkeit 1
|1> Wahrscheinlichkeit 0

E: $|x\rangle = \frac{1+2i}{3}|0\rangle + \frac{2}{3}|1\rangle$

Messung:
|0> Wahrscheinlichkeit $^5/_9$
|1> Wahrscheinlichkeit $^4/_9$

Man sieht an den Beispielen A und B dass trotz unterschiedlicher Vorzeichen identische Messergebnisse auftreten. Obwohl diese sogenannten globalen Phasen für die Messungen nicht signifikant sind können sie während des Berechnungsprozesses eine entscheidende Rolle spielen.

Neben dem Ket Vektor wird der sogenannte Bra Vektor definiert. Jedem Ket $|x\rangle$ entspricht ein Bra $\langle x|$ mit:

$$\langle x| = \begin{pmatrix} \alpha^* \\ \beta^* \end{pmatrix}^T = (\alpha^*, \beta^*)$$

Das Skalarprodukt in einem Vektorraum über $\mathbb{C}$ lässt sich dann schreiben als:

$$\langle x|y\rangle = \alpha_x^* \alpha_y + \beta_x^* \beta_y$$

Die Multiplikation eines Bra $\langle y|$ mit einem Ket $|x\rangle$ ergibt eine Matrix:

$$|x\rangle \cdot \langle y| = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \cdot (y_1^*, \dots., y_n^*) = \begin{pmatrix} x_1 \cdot y_1^* & \cdots & x_1 \cdot y_n^* \\ \vdots & \ddots & \vdots \\ x_n \cdot y_1^* & \cdots & x_n \cdot y_n^* \end{pmatrix}$$

Insbesondere ergeben die Einheitsvektoren $|1\rangle$ …. $|N\rangle$ als Linearkombination die Einheitsmatrix:

$$I = \sum_{i=1}^{N} |i\rangle \cdot \langle i| = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

Damit ergibt sich ein beliebiger Vektor $|a\rangle$ zu:

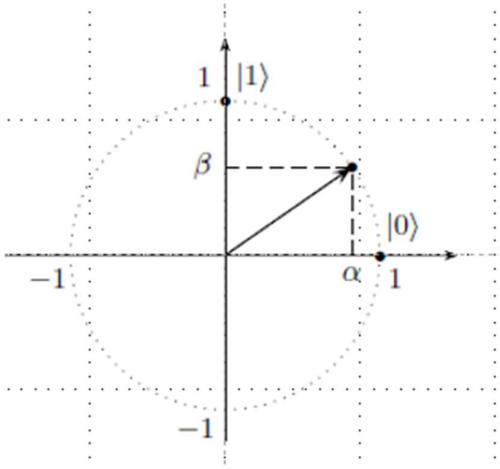
$$|a\rangle = I|a\rangle = \sum_{i=1}^{N} |i\rangle \cdot \langle i||a\rangle = \sum_{i=1}^{N} |i\rangle \cdot \alpha_i = \sum_{i=1}^{N} \alpha_i \cdot |i\rangle$$

Vektor Darstellung

Die beiden unterscheidbaren Zustände eines qBits werden durch zwei Basisvektoren repräsentiert.

$$|x\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \qquad \alpha, \beta \in \mathbb{C}$$

$|x\rangle$ ist also ein Vektor im komplexen Vektorraum $\mathbb{C}^2$.
Für den Spezialfall reeller Koeffizienten α,β lässt sich dieser Vektor in einem kartesischen Koordinatensystem darstellen:



Für Koeffizienten deren Imaginärteil ungleich null ist lässt sich der Vektor als Punkt auf der Bloch Kugel darstellen. Aus einem generischen Zustand $|x\rangle = \alpha|0\rangle + \beta|1\rangle$ lässt sich der oben angegebene Ausdruck durch folgende Überlegung erzeugen.

Für eine komplexe Zahl z mit $z = a + ib$ gilt $z = re^{i\varphi}$ mit $r = \sqrt{a^2 + b^2}$ und $\tan\varphi = \frac{b}{a}$

Somit gilt für $\alpha, \beta \in \mathbb{C}$:

$$|x\rangle = r_0 e^{i\varphi_0}\begin{pmatrix}1\\0\end{pmatrix} + r_1 e^{i\varphi_1}\begin{pmatrix}0\\1\end{pmatrix}$$

Eine Multiplikation mit $e^{-i\varphi_0}$ bewirkt lediglich eine Veränderung der Phasen, hat jedoch keinerlei Auswirkungen auf jegliche messbaren Größen. Es ergibt sich:

$$|x\rangle = r_0\begin{pmatrix}1\\0\end{pmatrix} + r_1 e^{i(\varphi_1-\varphi_0)}\begin{pmatrix}0\\1\end{pmatrix} = r_0\begin{pmatrix}1\\0\end{pmatrix} + r_1 e^{i(\varphi)}\begin{pmatrix}0\\1\end{pmatrix}$$

mit $r_0, r_1 \in \mathbb{R}^+$ und $\varphi = \varphi_1 - \varphi_0$. Aufgrund der Normierungsbedingung gilt $r_0^2 + r_1^2 = 1$ und somit lässt sich definieren:

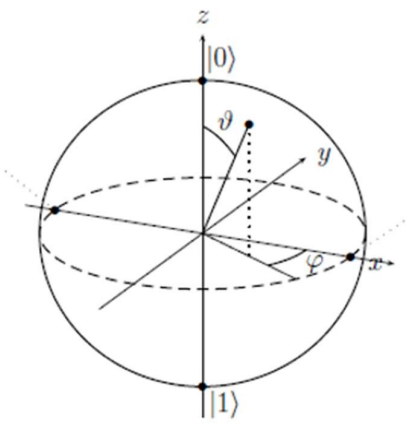
$$r_0 = \cos(\vartheta/2) \qquad\qquad r_1 = \sin(\vartheta/2)$$

Man erhält dann:

$$|x\rangle = \cos(\vartheta/2)\binom{1}{0} + e^{i\varphi}\sin(\vartheta/2)\binom{0}{1}$$

mit $\vartheta \in [0, \pi]$ und $\varphi \in [0, 2\pi]$.

Dabei ist der erste Koeffizient auf reele Wert eingeschränkt. Der Winkel ϑ beschreibt also die Aufteilung zwischen dem Zustand $|0\rangle$ und $|1\rangle$ und der Winkel φ beschreibt den Phasenwinkel in der xy Ebene.



Im späteren Verlauf werden wir sehen dass die beiden Zustände

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \quad \text{und} \quad |-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

für viele Anwendungen eine wichtige Rolle spielen. Im kartesischen Koordinatensystem sind diese beiden Zustände gerade die Winkelhalbierenden, während sie bei der Blochkugel auf der in der xy Ebene liegende Äquatorpunkte des Einheitskreises diametral gegenüber liegen.

Modifikation von qBits

Um den Zustand eines qBits modifizieren zu können müssen die speziellen Eigenschaften von Quantenobjekten berücksichtigt werden. Aus diesem Grund sind nur unitäre Transformationen erlaubt.

Unitäre Transformationen

In einem komplexen Vektorraum $\mathbb{C}^{\boldsymbol{n}}$ ist das Skalarprodukt zweier Vektoren $\vec{a}, \vec{b}$ Ɛ $\mathbb{C}^{\boldsymbol{n}}$ definiert als:

$$\vec{a} = \begin{pmatrix} a_1 \\ \dots \\ a_n \end{pmatrix}, \vec{b} = \begin{pmatrix} b_1 \\ \dots \\ b_n \end{pmatrix}$$

$$\langle \vec{a}, \vec{b} \rangle = a_1^* b_1 + \cdots + a_n^* b_n$$

Die Norm eines Vektors a ergibt sich damit zu:

$$\|\vec{a}\| = \sqrt{a_1^* a_1 + \cdots + a_n^* a_n} = \sqrt{|a_1|^2 + \cdots + |a_n|^2} = \sqrt{\langle \vec{a}, \vec{a} \rangle}$$

Zu einer Matrix A Ɛ $\mathbb{C}^{\boldsymbol{n}\times\boldsymbol{n}}$ ergibt sich die adjungierte Matrix $A^\dagger$ durch:

$$A^\dagger = (A^*)^T$$

Die adjungierte Matrix entsteht also sowohl aus der Transponierung als auch aus der komplexen Konjugation der Matrixelemente.

Es gilt:

$$(AB)^\dagger = B^\dagger A^\dagger$$

Eine Matrix U Ɛ $\mathbb{C}^{n\times n}$ heißt unitär wenn gilt:

$$U^\dagger = U^{-1}$$

Unitäre Matrizen erhalten das Skalarprodukt und damit auch die Winkel zwischen Vektoren. Bei rein reellen Einträgen sind die unitären Matrizen genau die orthogonalen Matrizen, deren Zeilen und Spalten paarweise orthogonal und normiert sind. Ist U unitär und a Ɛ $\mathbb{C}^{\boldsymbol{n}}$ dann gilt:

$$\|U\vec{a}\| = \|\vec{a}\|$$

Das heißt also insbesondere dass ein Zustandsvektor bei der Transformation U wieder auf einen Zustandsvektor abgebildet wird.

Mit diesen mathematischen Grundlagen können wir alle benötigten Modifikationen von qBits beschreiben. Zusammenfassend können wir also sagen:

- Ein qBit wird durch einen komplexen Spaltenvektor beschrieben
- Bei einer Messung geben die Betragsquadrate der komplexen Koeffizienten die Wahrscheinlichkeit an das qBit in einem der beiden möglichen Basiszustände vorzufinden
- Nach der Messung befindet sich das qBit mit Wahrscheinlichkeit 1 in dem zuvor gemessenen Zustand
- Befindet sich ein qBit im Zustand $|x\rangle$ wird es durch eine unitäre Transformation U in einen Zustand $|y\rangle$ überführt

Quanten Gatter

Die zur Realisierung von Quantenalgorithmen relevanten unitären Transformationen werden auch als sogenannte Gatter bezeichnet. Wir beschränken uns zunächst auf Gatter die lediglich ein singuläres qBit modifizieren. Solche Gatter werden beschrieben durch eine unitäre Matrix UƐ$\mathbb{C}^{\mathbf{2\times 2}}$, also durch eine quadratische Matrix mit komplexen Koeffizienten und 2 Spalten bzw. Reihen.

„Einfache“ Gatter

Typische Matrizen dieser Art sind zum Beispiel die Identitätsmatrix oder die sogenannten Pauli Matrizen die wir im weiteren Verlauf verwenden werden:

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

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Betrachten wir zunächst ein sehr einfaches Beispiel bei dem wir die Identitätsmatrix I_2 auf einen Zustand $|x\rangle = \alpha|0\rangle + \beta|1\rangle$ anwenden. Die Multiplikation der Matrix mit dem Vektor ergibt:

$$I\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$$

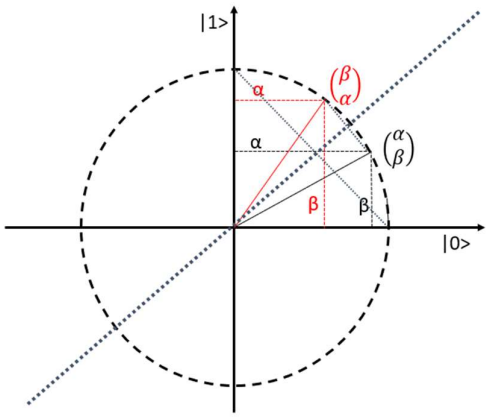
Betrachtet man die Anwendung von σ_x auf die Basisvektoren $|0\rangle$ und $|1\rangle$ so werden sie ineinander überführt. Damit entspricht σ_x aussagenlogisch auf Bitebene betrachtet einem NOT.

$$\sigma_x\begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}0\\ 1\end{pmatrix} \qquad\qquad \sigma_x\begin{pmatrix}0\\ 1\end{pmatrix} = \begin{pmatrix}1\\ 0\end{pmatrix}$$

Wenden wir die Paulimatrix σ_x an auf den Zustand $|x\rangle = \alpha|0\rangle + \beta|1\rangle$ an ergibt sich:

$$\sigma_x\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\beta\\ \alpha\end{pmatrix}$$

Hier werden die Koeffizienten α,β also gerade vertauscht. Diese Vertauschungen entsprechen graphisch einer Spiegelung an der Winkelhalbierenden des xy Quadranten:



Wenden wir die Paulimatrix σ_y an auf den Zustand $|x\rangle = \alpha|0\rangle + \beta|1\rangle$ an ergibt sich:

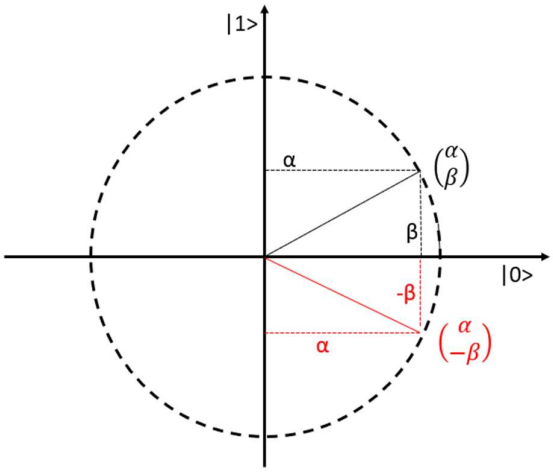
$$\sigma_y\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}-i\beta\\ i\alpha\end{pmatrix}$$

Hier werden die Koeffizienten α,β vertauscht und mit einer globalen Phase multipliziert. Auf der Bloch-Kugel entspricht dies einer 180◦-Rotation um die y-Achse.

Wenden wir die Paulimatrix σ_z an auf den Zustand $|x\rangle = \alpha|0\rangle + \beta|1\rangle$ an ergibt sich:

$$\sigma_z\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\alpha\\ -\beta\end{pmatrix}$$

Hier wird der Zustand im Prinzip an der x-Achse gespiegelt.



Hadamard Gatter

Ein in Quantenalgorithmen sehr häufig genutztes Gatter ist das nach dem französischen Mathematiker Jacques Hadamard benannte Gatter. Es ist folgendermaßen definiert:

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

Es lässt sich leicht nachrechnen dass die Matrix unitär ist.

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

$$H \cdot H^{\dagger} = I$$

Es gilt damit ausserdem:

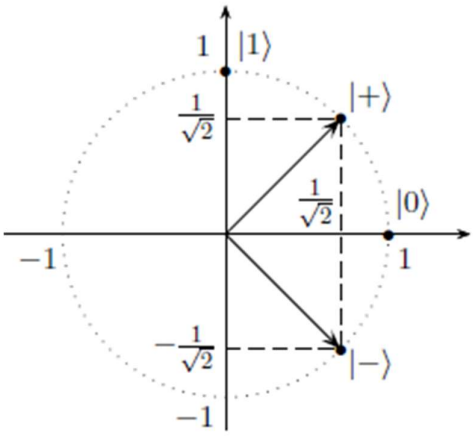
$$H^{-1} = H$$

D.h. eine zweimalige Anwendung der Hadamard Transformation entspricht der Identitätsmatrix I .
Wendet man H auf die Basiszustände |0> und |1> an so erhält man die oben bereits erwähnten Zustände |+> und |->:

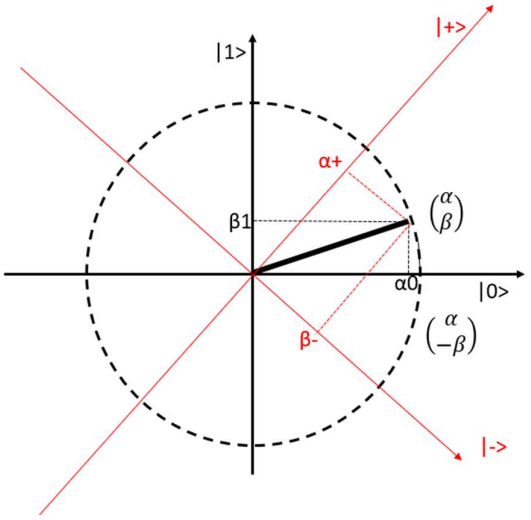
$$H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

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

Graphisch dargestellt sehen die beiden Zustände wie folgt aus:



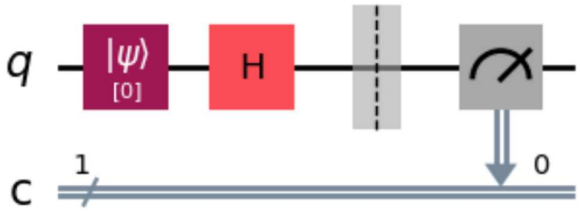
Wie bereits erwähnt werden bei einer Messung eines Zustandes die Beträge der Projektionen auf die Basiszustände |0> und |1> bestimmt. Wie man sieht ergeben sich dadurch für die Zustände |+> und |-> jeweils die selben Messergebnisse |0> Wahrscheinlichkeit $^1/_2$ und |1> Wahrscheinlichkeit $^1/_2$. Das liegt daran dass sich die beiden Zustände lediglich in ihrer globalen Phase unterscheiden. Wie bereits erwähnt kann diese Phase jedoch im Laufe des Rechenprozesses von Bedeutung sein. Man kann den Messprozess dahingehend erweitern dass eine Messung nicht bezüglich der Basiszustände |0> und |1> durchgeführt wird sondern als Projektionsachsen zum Beispiel die Zustände |+> und |-> verwendet werden. Misst man also zum Beispiel den Zustand |+> bezüglich dieser Basis ist das Ergebnis 1 für den |+> Zustand und Null für den |-> Zustand. Praktisch wird dieser Basiswechsel dadurch realisiert dass vor der Messung eine dann zur Messung gehörende Hadamard Transformation durchgeführt wird.



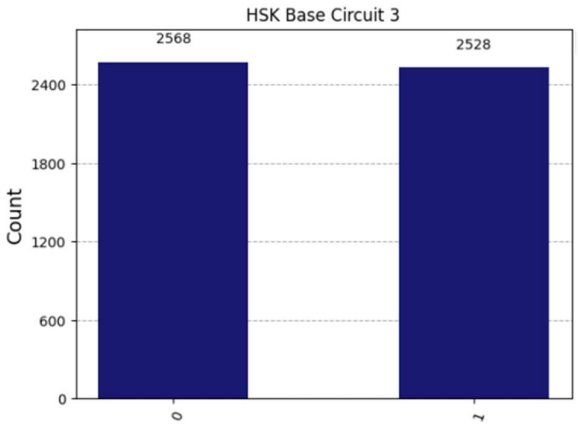
Zufallsgenerator

Mit unseren ersten Grundlagenkenntnissen sind wir nun in der Lage einen allerersten, sehr simplen Quantenalgorithmus zu generieren. Aufgrund der Tatsache dass ein qBit ein Quantenobjekt ist stellt eine Messung seines Zustandes einen stochastischen Prozess dar. Dies ist in der Natur der Quantenmechanik begründet. Bringen wir also ein qBit in einen Superpositionszustand so wird die Natur dafür sorgen dass wir ein stochastisch bedingtes Messergebnis erhalten. Wie wir gesehen haben können wir mithilfe der Hadamard Transformation einen gleichmäßig über die Basiszustände

|0> und |1> verteilten Zustand erzeugen. Unser erster Quantum Circuit sieht dann einfach folgendermaßen aus:



Wir initialisieren ein qBit q in den Basiszustand |0> und führen dann eine Hadamard Transformation durch. Bei einer Messung mit 5096 Wiederholungen erhalten wir das folgende Ergebnis:



Quantenregister

Um Operationen auf mehreren qBits durchzuführen werden wir uns mit Quantenregistern und mehrdimensionalen unitären Matrizen beschäftigen. Dazu machen wir uns zunächst mit dem sogenannten Tensorprodukt vertraut.

Tensorprodukt

Um eine geeignete Darstellung eines Quantenregisters zu haben bedienen wir uns des sogenannten Tensorprodukts. Mit dessen Hilfe wird der Zustand eines Quantenregisters aus n qBits aus den Zuständen der einzelnen qBits zusammengesetzt. Wir gehen dabei nicht auf eine allgemeine Definition des Tensors und seines Produktes ein sondern verwenden lediglich die für uns relevanten Konventionen die uns erlauben Vektoren und Matrizen auf eine bestimmte Art miteinander zu multiplizieren.

Seien $\vec{a}, \vec{b} \in \mathbb{C}^n$ definiert als:

$$\vec{a} = \begin{pmatrix} a_1 \\ \dots \\ a_n \end{pmatrix} = a_1\overrightarrow{e_1} + \cdots + a_n\overrightarrow{e_n}, \vec{b} = \begin{pmatrix} b_1 \\ \dots \\ b_n \end{pmatrix} == b\overrightarrow{e_1} + \cdots + b_n\overrightarrow{e_n}$$

Dann ist ihr Tensorprodukt definiert als:

$$\vec{a}\otimes\vec{b} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i b_j \left(e_i \otimes e_j\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i b_j \, |ij\rangle$$

Für den Fall n=2 ergibt sich dann für die Zustände $|x\rangle = \alpha_x|0\rangle + \beta_x|1\rangle$ und $|y\rangle = \alpha_y|0\rangle + \beta_y|1\rangle$:

$$\begin{pmatrix}\alpha_x\\ \beta_x\end{pmatrix} \otimes \begin{pmatrix}\alpha_y\\ \beta_y\end{pmatrix} = \begin{pmatrix}\alpha_x\alpha_y\\ \alpha_x\beta_y\\ \beta_x\alpha_y\\ \beta_x\beta_y\end{pmatrix}$$

Für 2 qBits aus den jeweiligen Vektorräumen mit den Basisvektoren |0> und |1> erhalten wir dann 4 Basisvektoren:

$$|0> \otimes |0> = |00>$$

$$|0> \otimes |1> = |01>$$

$$|1> \otimes |0> = |10>$$

$$|1> \otimes |1> = |11>$$

Dies kann auch in Vektorschreibweise dargestellt werden:

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

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

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

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

Man nennt diese Sortierung lexikographisch. Da es bei vielen Berechnungen auf das Sortierungsschema ankommt empfiehlt es sich diese konsequent anzuwenden.
Für die Zustände $|x\rangle = \alpha_x|0\rangle + \beta_x|1\rangle$ und $|y\rangle = \alpha_y|0\rangle + \beta_y|1\rangle$ ergibt sich dann auch:

$$|x\rangle\otimes|y\rangle = (\alpha_x|0\rangle + \beta_x|1\rangle) \cdot \left(\alpha_y|0\rangle + \beta_y|1\rangle\right)$$

$$= \alpha_x\alpha_y|00> + \alpha_x\beta_y|01> + \beta_x\alpha_y|10> + \beta_x\beta_y|11>$$

Man erhält also den Zustand des Registers durch Ausmultiplizieren der beiden Zustände.

Diese Vorgehensweise lässt sich auch auf Register mit n qBits verallgemeinern. Für ein Register mit n qBits ergeben sich dann die 2^n Basiszustände:

$$|0 \dots \dots .0>, |0 \dots \dots 01>, \dots \dots, |11 \dots .10>, |11 \dots \dots . .11>$$

Für n beliebige qBits mit $|x_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ ……..$|x_n\rangle = \alpha_n|0\rangle + \beta_n|1\rangle$ ergibt sich dann das Tensorprodukt durch Ausmultiplizieren:

$$|x_1\rangle \otimes |x_2\rangle \dots . . \otimes |x_n\rangle = \alpha_1\alpha_2 \cdots \alpha_n|00 \cdots 0> + \alpha_1\alpha_2 \cdots \beta_n|00 \cdots 1> \cdots \cdots + \beta_1 \cdots \beta_n|11 \cdots 11>$$

Wie für die Zustandsvektoren lässt sich das Tensorprodukt auch für Transformationen definieren. Für die Matrizen A,B Ɛ $\mathbb{C}^{n \times n}$ ergibt sich:

$$A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \dots & a_{nn}B \end{pmatrix}$$

Wenden wir dies auf die uns bereits bekannten Transformationen an ergeben sich folgende Matrizen.

$$I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$I_2 \otimes I_2 = \begin{pmatrix} 1 \cdot I_2 & 0 \\ 0 & 1 \cdot I_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

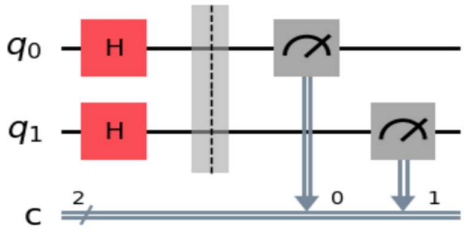
Und für die Hadamard Transformation.

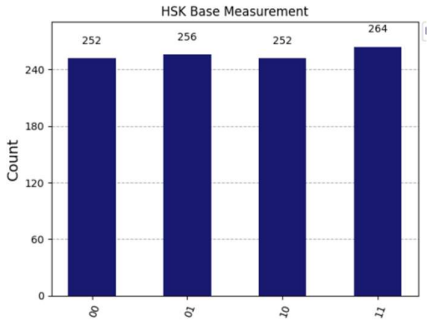
$$H = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$H_2 = H \otimes H = \frac{1}{\sqrt{2}} \begin{pmatrix} H & H \\ H & -H \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

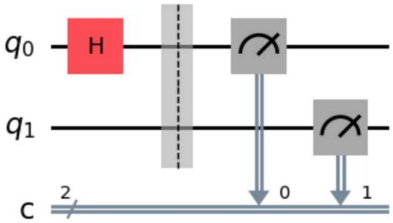
Wendet man einen n qBit Operator auf ein entsprechendes Quantenregister an so ist das Resultat dasselbe wie wenn die Einzeloperatoren auf die individuellen qBits angewendet werden:

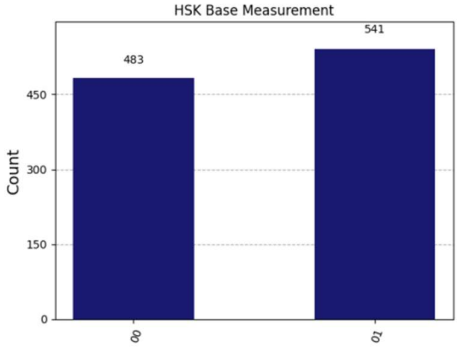
$H_2 = H \otimes H$ entspricht der Anwendung von H auf |x> und |y>



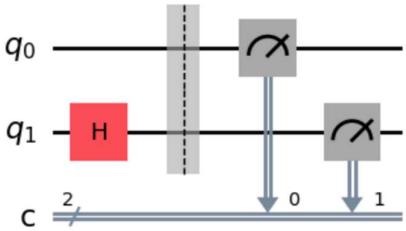


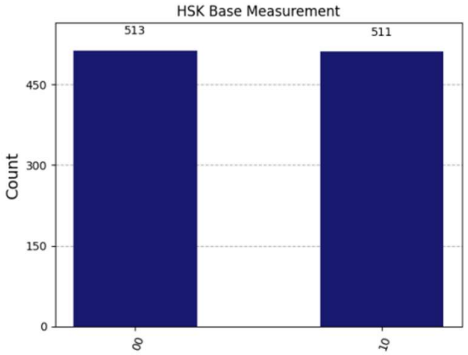
$H \otimes I_2$ entspricht der Anwendung von H auf |x>





$I_2 \otimes H$ entspricht der Anwendung von H auf |y>





Während die Messung für H_2 zeigt dass alle Basiszustände besetzt sind zeigt sich für $H \otimes I_2$ dass nur die Registerzustände $|00>$ und $|10>$ besetzt sind, während für $I_2 \otimes H$ die Registerzustände $|00>$ und $|01>$ besetzt sind.

Für das Tensorprodukt gelten folgende Rechenregeln:

- **Nicht kommutativ** $a \otimes b \neq b \otimes a$
- $a \otimes (b + c) = (a \otimes b) + (a \otimes c)$ und $(a + b) \otimes c = (a \otimes c) + (b \otimes c)$
- $(\lambda a) \otimes b = \lambda(a \otimes b) = a \otimes (\lambda b)$

- $a \otimes (b \otimes c) = (a \otimes b) \otimes c$

Eine weitere wichtige Eigenshaft des Tensorproduktes ist die Invarianz des Skalarprodukts. Das bedeutet für die Zustände |x >, |y >, |a >, |b > ε $\mathbb{C}^{2n}$:

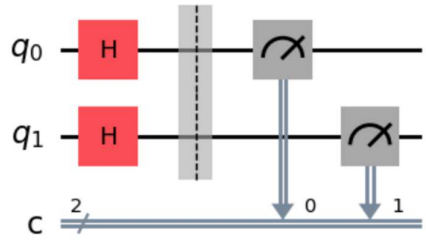
$$\langle x \otimes y | a \otimes b \rangle = \langle x | a \rangle \cdot \langle y | b \rangle$$

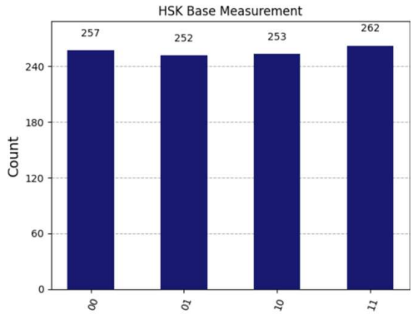
Messung eines Quantenregisters

Wie oben bereits angedeutet entstehen aus der Nichtkommutativität des Tensorprodukts für Operatoren Konsequenzen für den Messvorgang. So haben $H \otimes I_2$ und $I_2 \otimes H$ für unterschiedliche Messresultate gesorgt. Wir wissen dass die Messung eines qBits bezüglich der Normalbasis $|0>$ und $|1>$ den Zustand des qBits nach der Messung festlegt. Ähnlich wie wir oben gesehen haben erhalten wir für unterschiedliche Messansätze unterschiedliche Ergebnisse abhängig davon welches qBit gemessen wird.

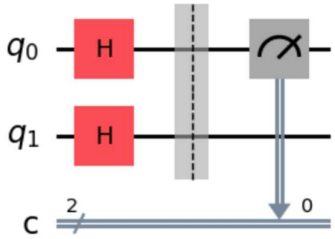
Wir wenden in allen Fällen die Hadamard Transformation auf beide qBits an.

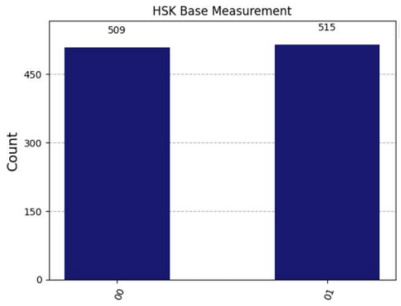
Messung beider qBits:



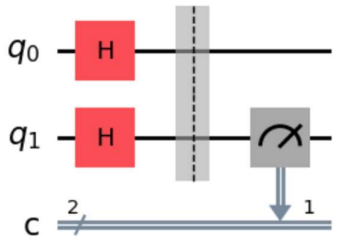


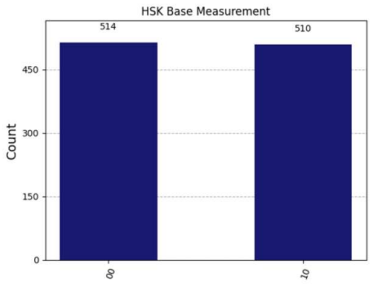
Messung qBit 0:





Messung qBit 1:





Man sieht dass auch hier bei den Einzelmessungen die Registerzustände $|00>$ und $|10>$ bzw. die Registerzustände $|00>$ und $|01>$ besetzt sind. Das bedeutet wenn beide qBits gemessen werden ergeben sich für die einzelnen Basiszustände:

$$|\mathrm{xy}>= \alpha_x\alpha_y|00> +\alpha_x\beta_y|01> +\beta_x\alpha_y|10> +\beta_x\beta_y|11>$$

Messung beide qBits(Verteilung auf alle 4 Basiszustände):

|00> $\|\alpha_x\alpha_y\|^2$

|01> $\|\alpha_x\beta_y\|^2$

|10> $\|\beta_x\alpha_y\|^2$

|11> $\|\beta_x\beta_y\|^2$

Messung qBit 0(gruppiert nach x):

|00> |01> $\|\alpha_x\alpha_y\|^2 + \|\alpha_x\beta_y\|^2$

|10> |11> $\|\beta_x\alpha_y\|^2 + \|\beta_x\beta_y\|^2$

Messung qBit 1(gruppiert nach y):

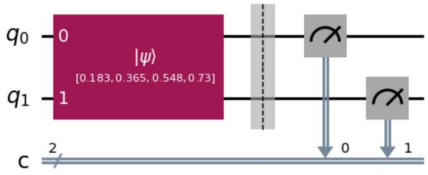
|00> |10> $\|\alpha_x\alpha_y\|^2 + \|\beta_x\alpha_y\|^2$

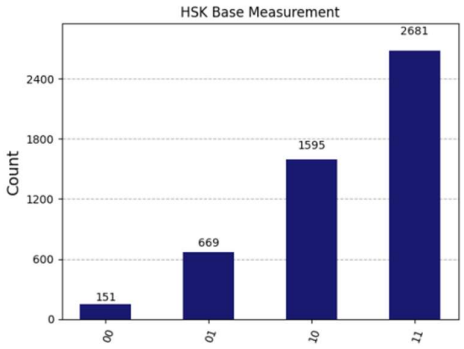
|01> |11> $\|\alpha_x\beta_y\|^2 + \|\beta_x\beta_y\|^2$

Dies lässt sich demonstrieren wenn wir die qBits inhomogen initialisieren und dann messen. Die Initialisierung ist folgendermaßen gewählt:

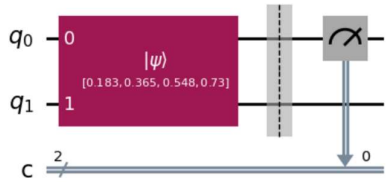
$$|\mathrm{xy}>= .183|00> +.365|01> +.548|10> .73|11>$$

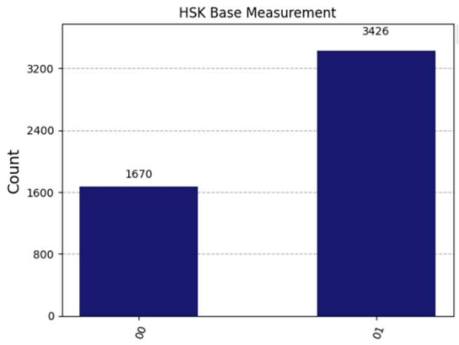
Messung beider qBits:





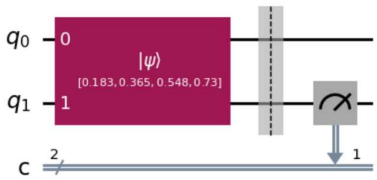
Messung qBit 0:

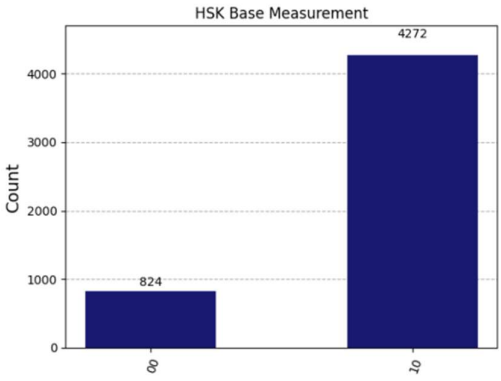




$$\left\|\alpha_x\alpha_y\right\|^2+\left\|\alpha_x\beta_y\right\|^2 \approx 5096 \cdot ((.183)^2+(.548)^2) \approx 1701$$
$$\left\|\beta_x\alpha_y\right\|^2+\left\|\beta_x\beta_y\right\|^2 \approx 5096 \cdot ((.365)^2+(.73)^2) \approx 3394$$

Messung qBit 1:





$$\left\|\alpha_x\alpha_y\right\|^2+\left\|\beta_x\alpha_y\right\|^2 \approx 5096 \cdot ((.183)^2+(.365)^2) \approx 849$$
$$\left\|\alpha_x\beta_y\right\|^2+\left\|\beta_x\beta_y\right\|^2 \approx 5096 \cdot ((.548)^2+(.73)^2) \approx 4246$$

Man sieht also wie sich bei Messung von qBit0 $|00>$ und $|01>$ bzw $|10>$ und $|11>$ addieren, während bei qBit1 $|00>$ und $|10>$ bzw $|01>$ und $|11>$ sich addieren.

Verschränktheit

Neben der Superposition stellt die Verschränkung von qBits eine spezielle Eigenschaft von Quantenobjekten dar. Die bekanntesten verschränkten Zustände sind die sogenannten Bell States:

$$|\phi^+>=\frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$$

$$|\phi^->=\frac{1}{\sqrt{2}}(|00\rangle-|11\rangle)$$

$$|\psi^+>=\frac{1}{\sqrt{2}}(|01\rangle+|10\rangle)$$

$$|\psi^->=\frac{1}{\sqrt{2}}(|01\rangle-|10\rangle)$$

Prinzipiell kann man jeden beliebigen Zustand danach klassifizieren ob er separierbar oder verschränkt ist. Ein aus 2 qBits gebildeter Zustand $|\psi>$ ist separierbar wenn man ihn als Tensorprodukt zweier 1 qBit Zustände $|x>, |y>$ darstellen kann:

$$|\psi\rangle = |x\rangle\otimes|y\rangle$$

Man kann zeigen dass ein 2 qBit Zustand

$$|xy>=\alpha_x\alpha_y|00>+\alpha_x\beta_y|01>+\beta_x\alpha_y|10>+\beta_x\beta_y|11>$$

separierbar ist wenn:

$$\alpha_x\alpha_y\cdot\beta_x\beta_y=\alpha_x\beta_y\cdot\beta_x\alpha_y$$

Ist ein 2 qBit Zustand nicht separierbar so nennt man ihn verschränkt: Betrachtet man die Bell States so zeigt sich zum Beispiel für $|\phi^+>$:

$$\alpha_x\alpha_y\cdot\beta_x\beta_y = {}^1/_2$$

$$\alpha_x\beta_y\cdot\beta_x\alpha_y = 0$$

Das heißt die Separabilitätsbedingung ist nicht erfüllt und $|\phi^+>$ ist daher verschränkt. Wendet man zum Beispiel H_2 auf $|11>$ an so ergibt sich:

$$H_2|11>=\frac{1}{2}(|00>-|01>-|10>+|11>)$$

Damit ist:

$$\alpha_x\alpha_y\cdot\beta_x\beta_y=\frac{1}{4}$$

$$\alpha_x\beta_y\cdot\beta_x\alpha_y=\frac{1}{4}$$

Somit ist der Zustand separierbar und lässt sich als Tensorprodukt darstellen:

$$H_2|11> = \frac{1}{2}(\,|00> - |01> - |10> + |11>) =$$

$$\frac{1}{\sqrt{2}}(\,|0> - |1>) \otimes \frac{1}{\sqrt{2}}(|0> - |1>) =$$

$$|-> \;\otimes\; |-> =$$

$$(H|1>) \otimes\; (H|1>)$$

Wie stark ein Zustand verschränkt ist kann man durch die sogenannte *Concurrence* quantifizieren. Sie ist definiert als:

$$C(|\mathrm{xy}>) = 2 \cdot \left|\left(\alpha_x\alpha_y \cdot \beta_x\beta_y\right) - \left(\alpha_x\beta_y \cdot \beta_x\alpha_y\right)\right|$$

$$0 \leq C(|\mathrm{xy}>) \leq 1$$

Für $C(|\mathrm{xy}>) = 0$ ist der Zustand seperierbar, da die Separabilitätsbedingung wie oben angegeben erfüllt ist. Für $C(|\mathrm{xy}>) = 1$ ist der Zustand maximal verschränkt. Für $|\phi^+>$ ergibt sich zum Beispiel:

$$C(|\phi^+>) = 2 \cdot \left|\alpha_x\alpha_y \cdot \beta_x\beta_y\right| = 2 \cdot \frac{1}{2} = 1$$

Dieser, wie auch die anderen Bell States zeigen maximale Verschränkung. Man kann verallgemeinert sagen dass Zustände die durch:

$$|\mathrm{xy}> = \frac{1}{\sqrt{k}}|00> + \frac{\sqrt{k-1}}{\sqrt{k}}|11>,\; k \geq 2$$

gegeben sind verschränkt sind. Dabei ist $C(|\mathrm{xy}>) = 1$ für k=2. Für wachsendes k konvergiert $C(|\mathrm{xy}>)$ gegen 0 da:

$$C_k(|\mathrm{xy}>) = 2 \cdot \left(\frac{\sqrt{k-1}}{k}\right)$$

Die Concurrence ist invariant gegen eine unitäre Transformation. Wendet man also auf eines der beiden qBits des Zustandes |xy> eine unitäre Transformation U an so gilt:

$$C(|\mathrm{xy}>) = C(|\mathrm{x}(Uy)>) = C(|(\mathrm{Ux})\mathrm{y}>)$$

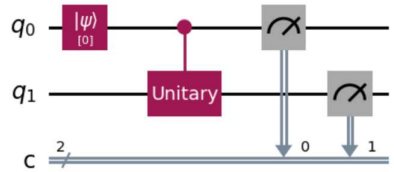
Insbesondere gilt damit für die Bell States dass sie bei einer unitären Transformation eines einzelne qBits maximal verschränkt bleiben.

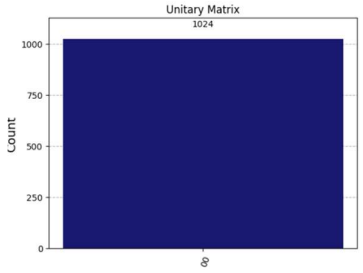
Kontrollierte 2 qBit Gatter

Wir haben am Beispiel der Hadamard Transformation bereits gesehen dass sich mithilfe des Tensorprodukts unitäre Transformationen erzeugen lassen die auf mehrere qBits einwirken. Während H_2 durch eine Hadamardtransformation auf jedes einzelne qBit realisiert werden kann gibt

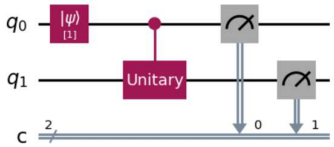
es auch sogenannte kontrollierte Gatter. Diese Gatter werden nur dann ausgeführt wenn ein Kontrollbit gesetzt ist.

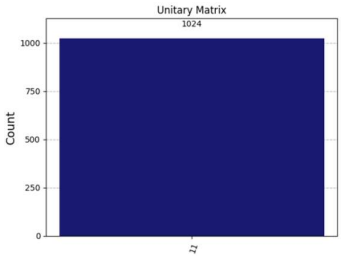
Beispiel Not Funktion:





In diesem Fall wird die Unitäre Transformation auf $|q_1>$ nicht ausgeführt da $|q_0>=0$ ist. Man misst $|00\rangle$. Setzt man dagegen $|q_0>=1$ wird U, in diesem Fall eine Not Operation ausgeführt und auf $|q_1>$ angewendet, man misst dann $|11\rangle$:





Man kann ganz allgemein zu einer beliebigen Transformation U, die auf ein Qubit wirkt das entsprechende durch ein anderes Qubit kontrollierte Gatter betrachten. Dies bedeutet zum Beispiel für die Basiszustände:

$$|00> \rightarrow |00>$$
$$|01> \rightarrow |01>$$
$$|10> \rightarrow |1> \otimes U|0>$$
$$|11> \rightarrow |1> \otimes U|1>$$

Die entsprechende Transformationsmatrix für diese 2 qBit Transformation hat dann die Struktur:

$$\begin{pmatrix} I & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & U \end{pmatrix}$$

Im obigen Beispiel hatten wir die Not Funktion betrachtet. Die unitäre Matrix für diese Funktion ist:

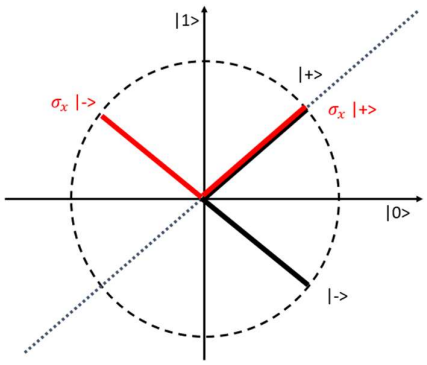
$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Anzumerken ist dass diese Matrix der Pauli σ_x Matrix entspricht. Wie wir oben gesehen haben entspricht diese Transformation graphisch betrachtet einer Spiegelung an der Winkelhalbierenden. Wie wir weiter oben gesehen haben bildet die Hadamard Transformation die Basiszustände $|0>$ und $|1>$ auf die sogenannte Hadamard Basis $|+>$ und $|->$ ab. Diese Basis ist wie die Winkelhalbierende gerade um 45° gegenüber den Standardbasisachsen verdreht. Wendet man σ_x auf den Zustand $|+>$ an so ergibt sich

$$\sigma_x|+>=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}=\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$$

$$\sigma_x|->=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix}=\begin{pmatrix}\frac{-1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$$

Dies entspricht wiederum einer Spiegelung an der Winkelhalbierenden. Dabei bleibt der $|+>$ Zustand invariant während der $|->$ Zustand um 180° auf dem Einheitskreis gedreht wird:



Die unitäre 2 qBit Matrix ergibt sich dann zu:

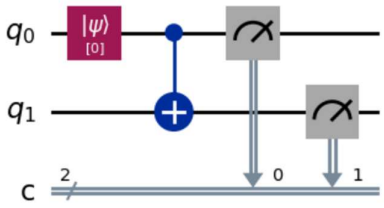
$$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$$

Sie stellt also die Operation CNOT dar, die wie wir sehen werden dem klassischen XOR entspricht. Dabei ist das zweit qBit das Ergebnisbit:

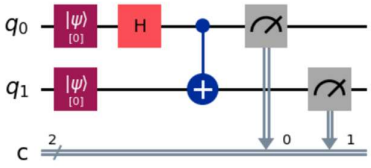
$$x_1x_2 \overset{\oplus}{\rightarrow} x_1(x_1 \oplus x_2)$$

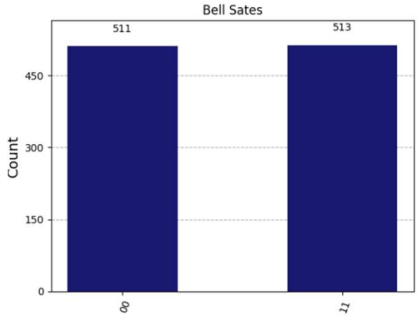
CNOT Gate

Das CNOT gate ist ein sehr wichtiges kontrolliertes Gatter. Es negiert wie bereits angesprochen ein qBit in Abhängigkeit von seinem Kontrollbit und hat deshalb sein eigenes Gatter Symbol:

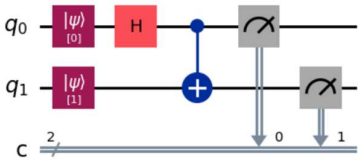


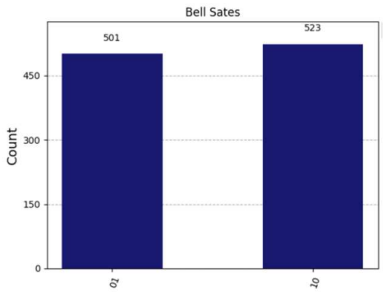
Das CNOT Gate spielt eine wichtige Rolle bei der Erzeugung der Bell States. Mit folgendem circuit können abhängig von der Initialisierung die 4 Bell States erzeugt werden:



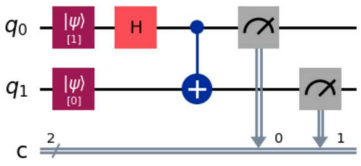


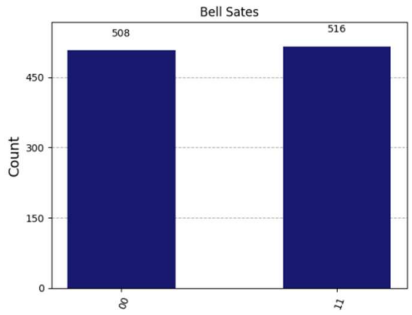
Bei Initialisierung $|00\rangle$ ergibt sich $|\phi^+>$



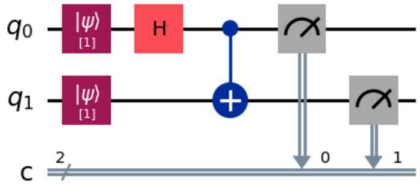


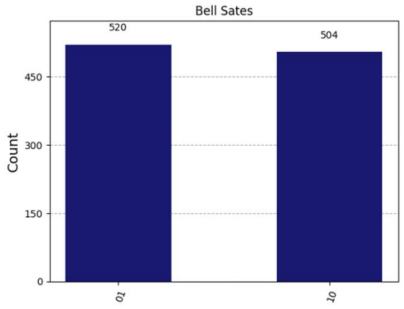
Bei Initialisierung $|01\rangle$ ergibt sich $|\psi^+>$





Bei Initialisierung $|10\rangle$ ergibt sich $|\phi^- >$





Bei Initialisierung $|11\rangle$ ergibt sich$|\psi^- >$

Man sieht dass die Phasen sich nicht im Messergebnis auswirken. Das heißt die Ergebnisse für $|\psi^+ >$ und $|\psi^- >$, sowie für $|\phi^+ >$ und $|\phi^- >$ sind jeweils gleich.

No Cloning Theorem

Eine Besonderheit von qBits ist die Einschränkung dass ein qBit nicht kopierbar ist. Mathematisch formuliert bedeutet dies:
Es gibt keine unitäre Transformation U mit:

$$\exists U, \phi \quad \text{so dass} \quad \forall \psi \text{ gilt} \quad U(|\psi > \otimes |\phi >) = (|\psi > \otimes |\psi >)$$

Es würde bereits ausreichen wenn Cloning für $|\phi >= |0 >$ möglich wäre. Man kann also ohne Beschränkung der Allgemeinheit $|\phi >= |0 >$ setzen. Gäbe es ein solches U dann folgt zum Beispiel für den $|+>$ Zustand:

$$U(|+> \otimes |0 >) = (|+> \otimes |+>) = \frac{1}{2}(|00 > +|01 > +|10 > +|11 >)$$

Aufgrund der Linearität von U gilt jedoch gleichzeitig:

$$U(|+> \otimes |0 >) = U\left(\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \quad\right) \otimes |0 >\right)$$

$$= \frac{1}{\sqrt{2}} U(|0 > \otimes |0 >) + \frac{1}{\sqrt{2}} U(|1 > \otimes |0 >)$$

aufgrund der Kopiereigenschaft von U wäre dann:

$U(|0> \otimes |0>) = (|0> \otimes |0>) = |00>$

$U(|1> \otimes |0>) = (|1> \otimes |1>) = |11>$

und somit:

$$U(|+> \otimes |0>) = \frac{1}{\sqrt{2}}|00> + \frac{1}{\sqrt{2}}|11>$$

Dies ist ein offensichtlicher Widerspruch. Zusammenfassend halten wir fest: Der Zustand eines qBits kann nicht gecloned werden.

Deutsch Algorithmus

Mittels unserer in den vorherigen Abschnitten erarbeiteten Grundlagen können wir uns nun einem weiteren Quantenalgorithmus zuwenden. Es handelt sich um den sogenannten Deutsch Algorithmus der in erster Linie theoretische Bedeutung hat. Der Deutsch Algorithmus beschäftigt sich mit der folgenden Problemstellung. Angenommen es gäbe eine Antwortmaschine die wir befragen könnten. Wir können unserer fiktiven Maschine allerdings nur 2 verschiedene Fragen stellen, die wir mit Frage 0 und Frage 1 bezeichnen wollen. Unsere Antwortmaschine kann genau wie wir auch nur 2 Antworten geben, die wir mit Antwort 0 und Antwort 1 bezeichnen wollen. Die Maschine antwortet immer nach einem fest determinierten Schema. Das bedeutet wir bekommen auf die Frage 0 von der Maschine stets die selbe Antwort und auf die Frage 1 ebenso. Gegeben der Konstellation gibt es 4 unterscheidbare Schemata wie die Maschine antworten kann:

Frage		AM 1	AM 2	AM 3	AM 4
0		0	1	0	1
1		0	1	1	0

Man sieht es gibt also prinzipiell 2 unterschiedliche Arten der Maschine. AM1 und AM2 antworten immer konstant mir der selben Antwort. AM3 und AM4 jedoch variabel in Abhängigkeit von der Frage. Die Antwortart von AM1 und AM2 nennt man konstant. Die Antwortart von AM3 und AM4 nennt man balanciert. Wir wollen nun durch eine geeignete Fragestellung herausfinden ob unsere Antwortmaschine konstanter oder balancierter Natur ist. Intuitiv ist uns klar dass wir dazu 2 Anfragen benötigen um dies beantworten zu können.

Der Deutsch Algorithmus beschäftigt sich nun damit ob es möglich ist auf Basis der Eigenschaften eines Quantenobjektes mit nur einer Anfrage herauszufinden ob die Antwortmaschine balanciert oder konstant ist. Um den ganzen Sachverhalt formalisieren zu können führen wir zunächst folgende Funktion f ein:

$$f: \{0,1\} \rightarrow \{0,1\}$$

Wie wir bereits oben gesehen haben gibt es entsprechend der 4 Maschinen genau 4 solcher Funktionen.

Wir können nun mithilfe dieser Funktionen eine unitäre Transformation definieren die auf 2 qBits operiert:

$$U_f: |\mathrm{xy}> \rightarrow |\mathrm{x}, \mathrm{y} \oplus \mathrm{f}(x)>$$

Die Operation $\oplus$ ist das bekannte XOR. Je nachdem welche Funktion f angewandt wird werden also die 4 Basiszustände folgendermaßen transformiert:

$\|\mathrm{xy}>$		$f(x)=0$	$f(x)=1$	$f(x)=x$	$f(x)=\neg x$
00		00	01	00	01
01		01	00	01	00
10		10	11	11	10
11		11	10	10	11

Für die einzelnen Funktionen f ergeben sich dann folgende Transformationsmatrizen:

$$f(x)=0 \qquad U_f = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$f(x)=1 \qquad U_f = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

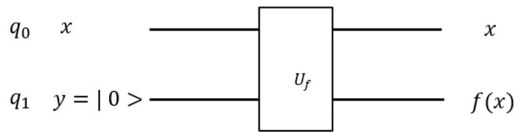
$$f(x)=x \qquad U_f = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$f(x)=\neg x \qquad U_f = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Diese Transformation angewandt auf die **Basiszustände** lässt also den Zustand $|\mathrm{x}>$ für alle f unverändert während sie den Zustand $|\mathrm{y}>$ in Abhängigkeit von f, $|\mathrm{x}>$ und $|\mathrm{y}>$ ändert. Dies gilt allerdings lediglich bei der Anwendung auf die Basiszustände. Man kann $|\mathrm{y}>$ im Prinzip als Target qBit der Operation U_f bezeichnen:

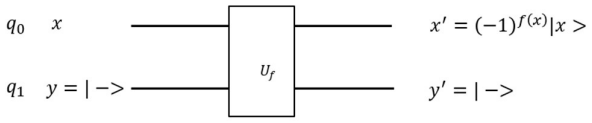
$$|\mathrm{x}>|\mathrm{y}> \quad \overset{U_f}{\rightarrow} \quad |\mathrm{x}>|\mathrm{y} \oplus \mathrm{f}(x)>$$

Für $|\mathrm{y}>=|0>$ erhält man als Ergebnis dann $|\mathrm{y} \oplus \mathrm{f}(x)> = |\mathrm{f}(x)>$. Das bedeutet man hat folgende Abbildung:



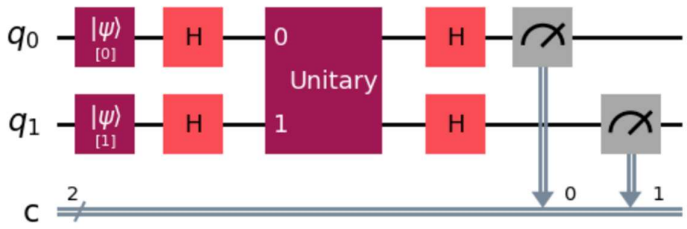
Das Target qBit spiegelt also den Funktionswert $f(x)$ wieder.

Setzt man das Target Bit jedoch in einen superponierten Zustand $|y> = |->$ so ergibt sich ein andere Situation:



Das Target qBit bleibt nun also invariant, während $|x>$ eine von $f(x)$ abhängige Phasenverschiebung erfährt.

Mithilfe dieser Transformation wird nun der Deutsch Algorithmus folgendermaßen aufgesetzt:



Jetzt wird U_f also nicht mehr auf die Basiszustände sondern auf die nach der Hadamard Transformation superponierten Zustände angewandt. Der Algorithmus lässt sich in folgenden Schritten beschreiben:

1. Initialisierung $|xy>$ auf $|01>$
2. Anwendung des Hadamard Gatters auf $|x>$ und auf $|y>$
3. Anwendung von U_f
4. Anwendung des Hadamard Gatters auf $|U_f x>$ und auf $|U_f y>$
5. Messung auf $|x>$ und auf $|y>$

Die Messung ergibt dann:

- für konstante f $|01>$
- für balancierte f $|11>$

Das bedeutet dass der Zustand $|x>$ sich in Abhängigkeit davon ob f balanciert oder konstant ist ändert.

Es lässt sich schrittweise nachrechnen wie dieses Ergebnis zustande kommt:

1. Initialisierung auf $|01>$

2. Anwendung der Hadamardtransformation

$$H_2|01> = \frac{1}{\sqrt{2}}(|0> + |1>) \cdot \frac{1}{\sqrt{2}}(|0> - |1>)$$
$$= \frac{1}{2}(|00> - |01> + |10> - |11>)$$

3. Anwendung von U_f :

$$= \frac{1}{2}(|0> \cdot |0 \oplus f(0)> - |0> \cdot |1 \oplus f(0)> + |1> \cdot |0 \oplus f(1)> - |1> \cdot |1 \oplus f(1)>)$$

$$= \frac{1}{2}(|0> \cdot (|f(0)> - |1 \oplus f(0)>) + |1> \cdot (|f(1)> - |1 \oplus f(1)>))$$

mit $|f(x)> - |1 \oplus f(x)> = (-1)^{f(x)} \cdot (|0> - |1>)$

lässt sich schreiben:

$$= \frac{1}{2}((-1)^{f(0)}|0> + (-1)^{f(1)}|1>) \cdot (|0> - |1>)$$

somit ist:

$$|x> = \frac{1}{\sqrt{2}}((-1)^{f(0)}|0> + (-1)^{f(1)}|1>)$$

$$|y> = \frac{1}{\sqrt{2}}(|0> - |1>)$$

Man sieht dass für *f* konstant das Vorzeichen beider Terme gleich bleibt, während für *f* balanciert das Vorzeichen wechselt

4. Betrachtet man nun die 2 Fälle balanciert und konstant separat so gilt:

f konstant: $(-1)^{f(0)} = (-1)^{f(1)}$
f balanciert: $(-1)^{f(0)} = (-1) \cdot (-1)^{f(1)}$

damit ergibt sich für $|x>$:

konstant: $|x_k> = \frac{\pm 1}{\sqrt{2}}(|0> + |1>)$ bildet H auf $|0>$ ab

balanciert: $|x_b> = \frac{\pm 1}{\sqrt{2}}(|0> - |1>)$ bildet H auf $|1>$ ab

Desweiteren gilt: $|y> = \frac{1}{\sqrt{2}}(|0> - |1>)$ bildet H auf $|1>$ ab

5. Die Messung von $|\mathrm{xy}>$ ergibt also
   f konstant $|01>$
   f balanciert $|11>$
   Das heisst man kann die beiden Fälle konstant und balanciert anhand der Messung unterscheiden.

Um den Kern des Algorithmus besser zu verstehen betrachten wir noch einmal die Situation nach der ersten Hadamard Transformation der beiden qBits. Wir erhielten den Zustand:

$$H_2|01>= \frac{1}{\sqrt{2}}(|0>+|1>)\cdot\frac{1}{\sqrt{2}}(|0>-|1>)$$

Dies entspricht:

$$H_2|01>= |+>\cdot|->$$

Wendet man nun U_f an so ergibt sich:

$$U_f(|+>\cdot|->) = |+>\cdot(-1)^{f(x)}|->= (-1)^{f(x)}|+>\cdot|->= \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0>+(-1)^{f(1)}|1>\right)\cdot|->$$

Das heißt
für konstantes *f=a,* $a \in \{0,1\}$ gilt $c = (-1)^a$:

$$\frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0>+(-1)^{f(1)}|1>\right)\cdot|->= c\cdot\frac{1}{\sqrt{2}}(|0>+|1>)\cdot|->= c\cdot|+>\cdot|->$$

für balanciertes f gilt $(-1)^{\mathrm{f}(0)} = (-1)\cdot(-1)^{\mathrm{f}(1)}$ bzw für *f(0)=a,* $a \in \{0,1\}$ mit $c = (-1)^a$:

$$\frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0>+(-1)^{f(1)}|1>\right)\cdot|->= c\cdot\frac{1}{\sqrt{2}}(|0>-|1>)\cdot|->= c\cdot|->\cdot|->$$

Man sieht also dass U_f den Zustand von $|\mathrm{x}>$, das in unserem Algorithmus das Control Bit darstellt, in Abhängigkeit von f zwischen $|+>$ und $|->$ ändert. Man nennt diese Methode, die in späteren Algorithmen verwendet werden wird Phase Kickback.

Deutsch Josza Algorithmus

Die Verallgemeinerung des Deutsch Algorithmus auf mehrdimensionale Funktionen

$$f:\{0,1\}^n \to \{0,1\}$$

ist der Deutsch Josza Algorithmus. Das heißt die Funktion *f* bildet beliebige n-bit Bytes entweder auf 0 oder auf 1 ab. Nehmen wir als Beispiel n=4 dann gäbe es $2^4 = 16$ verschiedene Bitstrings (0,0,0,0).....(1,1,1,1) mit entsprechender Zuordnung. Also zum Beispiel für konstantes *f(x)*=0

x	***f*(x)**
0000	0
0001	0

….	0
….	0
1110	0
1111	0

f nennt man konstant wenn alle Funktionswerte gleich sind.
f nennt man balanciert wenn $(2^n/2)$ Funktionswerte gleich 0 sind und ebenso viele gleich 1.

Vergleichen wir nun diese Situation mit der von oben bedeutet dies wir hätten eine Antwortmaschine der wir 2^n Fragen stellen können, die wir durch Bitstrings mit n Bits unterscheiden. Unsere Maschine kann auf jeden Bitstring nur mit 0 oder 1 antworten und zwar entweder konstant oder balanciert. Wieder wollen wir mit nur einer Frage herausfinden ob unsere Maschine konstant oder balanciert ist. Im klassischen Ansatz würden wir im günstigsten Fall nur 2, jedoch im ungünstigsten Fall $(2^n/2) + 1$ Fragen benötigen um sagen zu können ob *f* konstant oder balanciert ist.

Nun lässt sich ähnlich wie oben auf Basis der Funktion f folgende unitäre Transformation definieren:

$$U_f: |x_n \dots x_1, \mathrm{y} > \rightarrow |x_n \dots x_1, \mathrm{y} \oplus \mathrm{f}(x_n \dots x_1) >$$

man beachte dass $\mathrm{f}(x_n \dots x_1) \in \{0,1\}$

Ohne alle rechnerischen Details skizzieren wir den Algorithmus:

1. Initialisierung auf $|0 \dots .0,1 >$

2. Anwendung der Hadamardtransformation

$$H_n|0 \dots .0,1 >= \mathrm{HNX} \cdot \frac{1}{\sqrt{2}}(|0> - |1>)$$

mit $$HNX = \frac{1}{\sqrt{2^n}} \cdot \sum_{x=1}^{2^n} |x>$$

3. Anwendung von U_f nach Umformung

$$\mathrm{HU} = \mathrm{HNXf} \cdot \frac{1}{\sqrt{2}}(|0> - |1>)$$

mit $$HNXf = \frac{1}{\sqrt{2^n}} \cdot \sum_{x=1}^{2^n} (-1)^{\mathrm{f}(x)} |x>$$

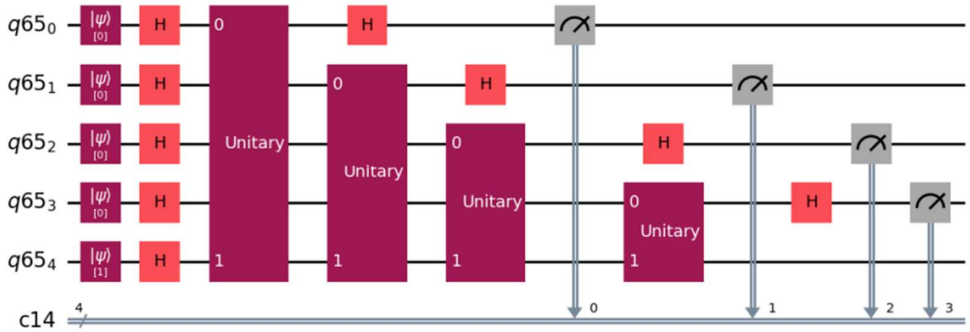
4. Anwendung der Hadamard Transformation auf $|\mathrm{x}>$

$$\mathrm{HUH} = \mathrm{HNXfH} \cdot \frac{1}{\sqrt{2}}(|0> - |1>)$$

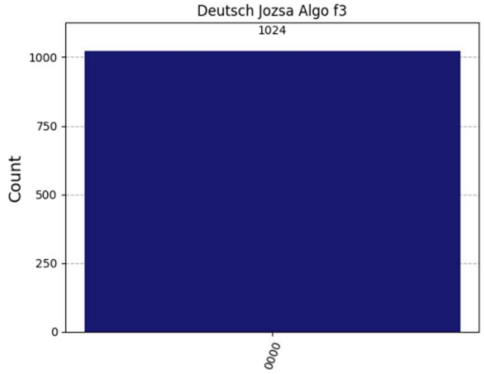
mit $$HNXfH = \frac{1}{\sqrt{2^n}} \cdot \sum_{z=1}^{2^n} \sum_{x=1}^{2^n} (-1)^{xz} (-1)^{\mathrm{f}(x)} |z>$$

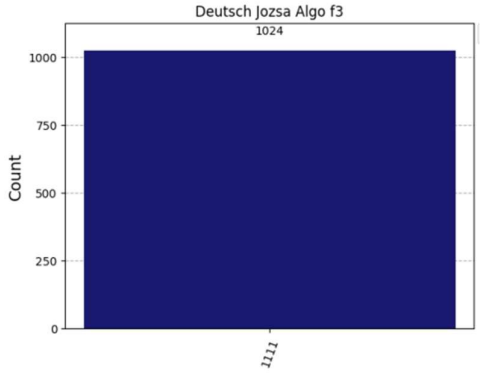
5. Die Messung von $|\mathrm{xy}>$ ergibt
   f konstant $|0 \dots \dots 0,1>$
   f balanciert $|\mathrm{xy}> \neq |0 \dots .0,1>$
   Das heisst man kann die beiden Fälle konstant und balanciert anhand der Messung unterscheiden.

Für n=4 sieht der Algorithmus folgendermaßen aus:



Messergebnisse konstant/balanciert





Anhand des Deutsch Josza Algorithmus sieht man bereits das man mit diesem Algorithmus im klassischen Fall eine in Abhängigkeit von n sehr schnell wachsende Anzahl von Abfragen stellen muss um zu entscheiden ob *f* konstant oder balanciert ist, während unser Quantenalgorithmus diese Frage mit einer Abfrage beantwortet.

Reversibilität

Wie wir bereits besprochen haben werden Operationen an qBits durch unitäre Transformationen dargestellt. Für diese gilt:

$$U^{\dagger} = U^{-1}$$

Dies bedeutet gleichzeitig dass die Transformationen reversibel sein müssen. Man muss also aus den Ergebnissen der Transformation rückschließen können welche Eingangswerte der Transformation zugrunde liegen. Die relevanten Operationen bei klassischen Rechnern sind teilweise nicht reversibel. So sehen wir zum Beispiel an der auf ein einzelnes Bit einwirkende NOT Operation dass sie reversibel ist:

$$\neg 0 = 1 \qquad\qquad \neg 1 = 0$$

Denn aus dem Ergebnis lässt sich auf die Eingabe schliessen. Im Fall der AND Operation ist dies nicht der Fall:

$$0 \wedge 0 = 0 \qquad 0 \wedge 1 = 0 \qquad 1 \wedge 0 = 0 \qquad 1 \wedge 1 = 1$$

Man sieht dass man die ersten 3 Fälle bezüglich ihrer Eingabewerte nicht unterscheiden kann. Um umkehrbare Abbildungen zu charakterisieren führen wir den Begriff der Bijektivität ein.

Eine Abbildung f, die von der Menge *X* in die Menge *Y* abbildet heißt bijektiv wenn für alle $y \in Y$ genau ein x$\in X$ existiert mit $y = f(x)$:

$$f: X \to Y \qquad \forall y \in Y : \exists! x \in X: f(x) = y$$

Mit dieser Definition können wir sagen eine Operation ist umkehrbar wenn sie eine bijektive Abbildung auf sich selbst ist:

$$f: X \to X \text{ , } \textit{bijektiv}$$

Wir haben am Beispiel der NOT Operation gesehen dass es klassische Abbildungen gibt die umkehrbar sind. Diese Abbildungen können auch als Quantenoperationen dargestellt werden. Für Operationen die im klassischen Fall nicht umkehrbar sind benötigt man eine geeignete Erweiterung um sie mithilfe zusätzlicher Hilfsbits reversibel zu gestalten. Wir haben dies bereits im Verlaufe unserer Betrachtungen am Beispiel der XOR Operation gesehen. XOR erzeugt folgende Transformationen:

$$0 \oplus 0 = 0 \qquad 0 \oplus 1 = 1 \qquad 1 \oplus 0 = 1 \qquad 1 \oplus 1 = 0$$

Wie man sieht ist die Abbildung nicht umkehrbar. Nun lässt sich eine reversible Abbildung gestalten indem man sich eines der beiden Bits merkt:

$$(x, y) \to (x, x \oplus y)$$

Diese Abbildung erzeugt folgende Transformationen:

$$\begin{aligned} |00> &\to |00> \\ |01> &\to |01> \\ |10> &\to |11> \\ |11> &\to |10> \end{aligned}$$

Wie man sieht ist diese Abbildung bijektiv auf sich selbst. Indem man sich also einen der beiden Input Werte merkt lässt sich die Umkehrbarkeit erzeugen. Die entsprechende unitäre Matrix ist:

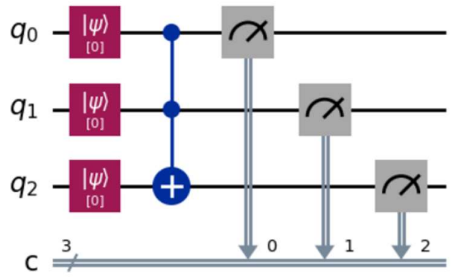
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

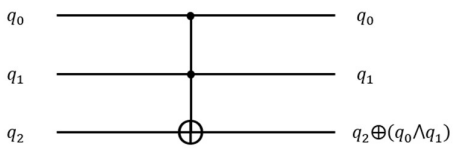
Sie entspricht genau dem CNOT Gate das wir bereits in einem früheren Abschnitt kennengelernt hatten und ergibt im 2. Argument die klassische XOR Wahrheitstafel. Uns sind im Verlaufe unserer Betrachtungen immer wieder Matrizen begegnet die sich sehr ähnlich sind. Sie haben in jeder Reihe sowie in jeder Spalte lediglich eine 1 und ansonsten 0. Man nennt solche Matrizen Permutationsmatrizen. Es lässt sich zeigen dass Permutationsmatrizen unitär sind. Werden sie auf einen Zustandsvektor angewendet so permutieren sie die den Basisvektoren zugeordneten Amplituden. Für das Beispiel CNOT:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_4 \\ \alpha_3 \end{pmatrix}$$

Toffoli Gatter

Wir wollen diese Vorgehensweise nun verallgemeinern. Bezüglich klassischer Gatter sind AND, OR und NOT bekannt. Mit der sogenannten *Disjunktiven Normalform* lässt sich zeigen dass jede beliebige Boole'sche Aussage durch die oben genannten Gatter AND, OR und NOT auf eine geordnete Weise dargestellt werden kann. Ein umkehrbares Gitter das universell ist und AND, OR und NOT realisiert ist das Toffoli Gatter. Es hat 3 Eingänge und 3 Ausgänge.

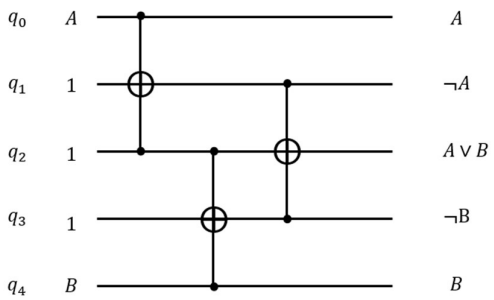




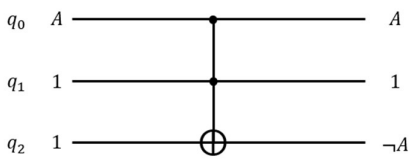
Das Toffoli Gatter entspricht einem CCNOT Gate. Das heißt die Negierung von q_2 wird von den beiden Kontrollbits q_0 und q_1 bedingt. Dies entspricht der logischen Verknüpfung $q_2 \oplus (q_0 \wedge q_1)$, die als Ergebnis auf q_2 ausgegeben wird.

Mithilfe des Toffoli Gatters können die Operationen AND, OR und NOT auf folgende Weise dargestellt werden.

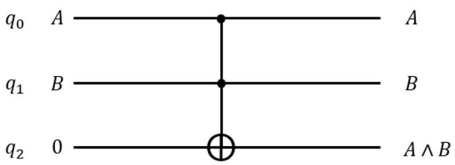
OR



NOT



AND



Mithilfe des Toffoli Gatters lässt sich jeder klassische auf der Boole'schen Aussagenlogik basierende Schaltkreis in einen unitären Quantenschaltkreis überführen, der nur in linearem Masse umfangreicher ist. Man sieht also dass sich alle logischen Operationen durch entsprechende kontrollierte Negierungen erzeugen lassen.

Fehlerbetrachtungen

qBits sind im Gegensatz zu bits nicht binär. Das heißt sie können sich aufgrund des Superpositionsprinzips in unendlich vielen Zuständen befinden. Da nach einer unitären Transformation ein modifizierter Einheitsvektor gebildet wird lässt sich nicht eindeutig sagen wie stark der erzeugte Vektor vom beabsichtigten Vektor abweicht. Um uns dieser Problematik anzunähern stellen wir uns einen Zustand $|\mathrm{x}>$ vor auf den wir sequentiell eine Reihe von n unitären Transformationen $U_1 \dots U_n$ anwenden. Wären alle Transformationen fehlerfrei würden wir aus $|\mathrm{x}>$ einen Zustand $|\mathrm{y}>$ erzeugen mit:

$$|\mathrm{y}>= U_1 \dots U_n|\mathrm{x}>$$

Sind die unitären Transformationen jedoch fehlerbehaftet erhalten wir einen von $|\mathrm{y}>$ abweichenden Zustand $|\mathrm{z}>$:

$$|\mathrm{z}>= U_1' \dots U_n'|\mathrm{x}>$$

Der Fehler ergibt sich dann zu:

$$\||\mathrm{y}> - |\mathrm{z}>\|$$

Da die Transformationen alle unitär sind erhalten sie die Einheitsnorm eines jeden Zwischenzustandes. Das heißt jede Transformation $U_i \in \{U_1 \dots U_n\}$ wird wieder auf einen Einheitsvektor angewendet. Die Einzelfehler multiplizieren sich also nicht auf sondern addieren sich lediglich. Betrachten wir n Transformationen mit einem angenommenen Fehler von 1% so ergeben sich folgende Abweichungen:

n	Multiplikation	Addition
10	1,1	1,1
20	1,2	1,2
50	1,6	1,5
100	2,7	2

Man sieht dass sich der additive Fehler bei steigendem n weniger auswirkt als der multiplikative. Zusammenfassend lässt sich also sagen dass man in Anlehnung an eine klassische auf der Boole'schen Aussagenlogik basierende Rechenmaschine eine auf qBits basierende Quantenrechenmaschine realisieren kann die innerhalb bestimmter Fehlergrenzen operiert.

Komplexitätsklassen

Wie wir anhand des Algorithmus von Deutsch Josza gesehen haben ist es entscheidend wie schnell die Anzahl der einzelnen erforderlichen Aktionen in Abhängigkeit von einer bestimmten input Größe anwächst. Im oben betrachten Fall galt für die Anzahl der gemachten Anfragen im ungünstigsten Fall $(2^n/2) + 1$. Intuitiv ist klar dass der entscheidende Faktor dabei 2^n ist. Wir werden uns im Folgenden ein weinig mit den sogenannten Komplexitätsklassen beschäftigen die diese Art von Abschätzungen umreißen. Ganz formal beschäftigen wir uns also mit einer Funktion f die die Eingabegröße eines Berechnungsproblems auf die Anzahl der benötigten Rechenschritte abbildet:

$$f: \mathbb{N} \to \mathbb{N}$$

Ohne *f* genauer zu kennen interessiert uns an dieser Funktion in erster Linie wie ihr Wachstumsverhalten in Abhängigkeit ihres input n aussieht.

O Notation

Es stellt sich also die Frage ob *f* asymptotisch unterhalb einer bestimmten Obergrenze bleibt. Um dies zu formalisieren führen wir die sogenannte O Notation ein. Mit dieser Notation sagen wir:

$$f(n) = O\big(g(n)\big)$$

Was bedeutet dass $f(n)$ asymptotisch nicht schneller wächst als $g(n)$. Für zwei Funktionen *f,g* gilt dass *f* nicht schneller wächst als *g* wenn es ein n_0 und eine Konstante *c* gibt so dass:

$$f(n) \leq c \cdot g(n) \quad \forall\, n > n_0$$

Man verwendet dann für $g(n)$ in der Regel Funktionen deren Wachstum gut bekannt ist und das sich signifikant in seinem Wachstumsverhalten unterscheidet:

n	$\boldsymbol{log_2 n}$	$\boldsymbol{\sqrt{n}}$	$\boldsymbol{n^2}$	$\boldsymbol{n^3}$	$\boldsymbol{2^n}$
8	3	3	64	512	256
1000	10	32	10^6	10^9	10^{301}
1000000	20	1000	10^{12}	10^{27}	10^{30100}

Man definiert weiter $f: \mathbb{N} \to \mathbb{N}$ heisst polynominal wenn:

$$f(n) = O(n^k) \text{ für } k\text{=konstant}$$

Ein Sonderfall stellt $O(1)$ dar. Es bedeutet dass der Aufwand eines Algorithmus unabhängig von der input Grösse n ist. Ein Beispiel hierfür wäre etwa ein Zugriff auf ein Array bei dem der Arrayindex bekannt ist.

Um Berechnungsprobleme anhand ihrer Schwierigkeit grob einzuteilen betrachtet man unterschiedliche Komplexitätsklassen. Wie beschränken uns bei den folgenden Betrachtungen auf sogenannte Entscheidungsprobleme. Dies sind Probleme deren Ergebnis 0 oder 1 ist. Ein typisches Beispiel wäre:

Eingabe: n
Ausgabe: 1 falls n eine Primzahl ist 0 falls n keine Primzahl ist

Auch das im Deutsch Algorithmus gezeigte Problem ist ein Entscheidungsproblem. Für konstante *f* war die Ausgabe 0, für balancierte *f* war sie 1.

Definition diverser Klassen

Man kann auf diesen Grundlagen nun folgende Komplexitätsklassen definieren:

P
Die Komplexitätsklasse P umfasst alle Probleme, die mittels eines Computers in polynomieller Zeit entscheidbar sind. Ein Problem ist polynomiell entscheidbar, wenn ein Algorithmus mit einer Laufzeit $O(n^k)$ mit *k=konstant* existiert, der das Problem determiniert entscheidet.

NP
Die Komplexitätsklasse NP umfasst alle Probleme für die mittels eines Computers in polynomieller Zeit beweisbar entschieden werden kann dass das Entscheidungsergebnis 1 ist.

BPP
Die Komplexitätsklasse BPP umfasst alle Probleme bei denen man mittels eines Computers in polynomieller Zeit mit einer festen Wahrscheinlichkeit $> {}^1/_2$ bestimmen kann ob die Lösung 0 oder 1 ist.

BQP
Die Komplexitätsklasse BQP umfasst alle Probleme bei denen man mittels eines Quantencomputers in polynomieller Zeit mit einer festen Wahrscheinlichkeit $> {}^1/_2$ bestimmen kann welche Entscheidungsfälle mit 1 entschieden werden können.

Zum Beispiel ist das Problem der Primfaktor-Zerlegung in NP, da man eine von einem geeigneten Algorithmus vorgegebene Faktorisierung in polynomieller Zeit überprüfen kann. Man kann allerdings nicht sagen ob das Problem in P ist. Der Shor-Algorithmus ist ein polynomieller Quanten-Algorithmus, der die Primfaktor-Zerlegung mit einer hohen Wahrscheinlichkeit berechnet und damit das Problem in BQP legt. Die Komplexitätstheorie ist ein weites Forschungsfeld bei dem es bislang noch zu keiner Einigung gekommen ist welche oder ob die verschiedenen Komplexitätsklassen ineinander liegen.

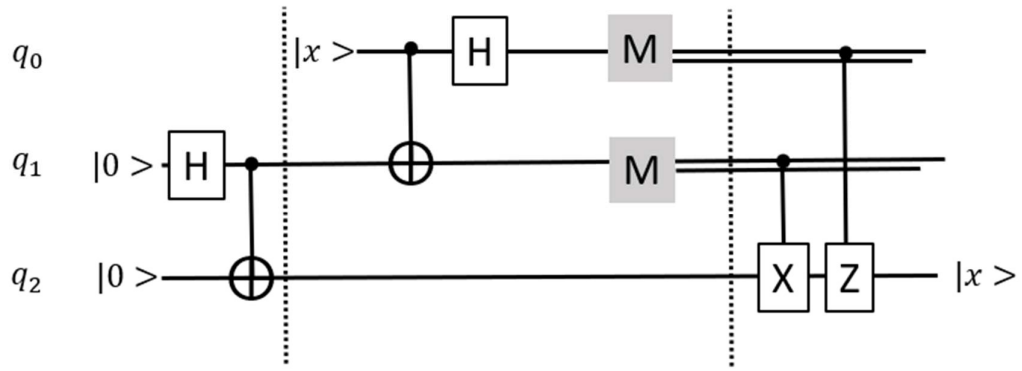
Teleportation

Wir haben bereits gesehen dass sich zwei qBits durch entsprechende Transformationen verschränken lassen. Insbesondere zeigen die sogenannten Bell States maximale Verschränkung.

Basierend auf dieser speziellen Eigenschaft von Quantenobjekten kann ein qBit auf ein anderes qBit übertragen werden indem eine Übertragung von klassischer Information stattfindet.

Algorithmus

Üblicherweise wird der Teleportationsprozess anhand dem Informationsaustausch zweier Personen die Alice und Bob genannt werden erläutert. Voraussetzung um diesen Prozess zu realisieren ist dass sich beide unabhängig von dem zu übertragenden Zustand $|\mathrm{x}>$ ein verschränktes qBit Paar, das sich zum Beispiel im Zustand $|\phi^+>$ befindet, teilen. Wir skizzieren zunächst den Algorithmus als Quantenschaltkreis:



Es geht darum den Zustand $|\mathrm{x}>$ der bereits vorliegt oder erzeugt wurde und sich bei Alice befindet an Bob zu übermitteln. Zunächst verschränken beide die qBits q_1 und q_2 um ein ein Bell Paar zu erhalten. Dann trennen sie sich räumlich wobei jeder eines der beiden qBits des Bell Paares mit sich nimmt. Danach verschränkt Alice ihren Zustand $|\mathrm{x}>$ mit dem ihr zugeteilten qBit des Bell Paares und misst beide qBits q_0 und q_1. Diesen gesamten Vorgang der Verschränkung und Messung nennt man auch Bell Messung. Das Ergebnis übermittelt sie dann auf klassische Weise an Bob, der in Abhängigkeit von den Ergebnissen eine Pauli σ_x beziehungsweise eine Pauli σ_z Transformation durchführt. Die Pauli Transformationen haben wir bereits in einem vorherigen Abschnitt kennengelernt.

Analyse

Für die Analyse betrachten wir für die einzelnen Transformationen jeweils das Register

$$|\psi>=|q_0\, q_1 q_2>$$

Der Zustand des zu übertragenden qBits sei

$$|x\rangle = \alpha|0\rangle + \beta|1\rangle$$

Wir erhalten also den Anfangszustand:

$$|\Psi_A>=(\alpha|0\rangle+\beta|1\rangle)\cdot\frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$$

$$|\Psi_A>=\frac{\alpha}{\sqrt{2}}(|000\rangle+|011\rangle)+\frac{\beta}{\sqrt{2}}(|100\rangle+|111\rangle)$$

Nun wendet Alice das CNOT Gate an und wir erhalten:

$$|\Psi_{CNOT}>=\frac{\alpha}{\sqrt{2}}(|000\rangle+|011\rangle)+\frac{\beta}{\sqrt{2}}(|110\rangle+|101\rangle)$$

Gefolgt von einem Hadamard Gate:

$$|\Psi_{HCNOT}>=\frac{\alpha}{2}(|000\rangle+|011\rangle+|100\rangle+|111\rangle)+\frac{\beta}{2}(|010\rangle+|001\rangle-|110\rangle-|101\rangle)$$

Dies lässt sich umsortieren so dass man die Messung der beiden qBits von Alice isoliert:

$$|\Psi_{Mess}>=\frac{1}{2}\Big(\big(|00\rangle\cdot(\alpha|0\rangle+\beta|1\rangle)\big)+\big(|01\rangle\cdot(\beta|0\rangle+\alpha|1\rangle)\big)+\big(|10\rangle\cdot(\alpha|0\rangle-\beta|1\rangle)\big)+\big(|11\rangle\cdot\beta|0\rangle-\alpha|1\rangle)\big)\Big)$$

Man erhält also 4 Terme die für die 4 unterschiedlichen Messergebnisse stehen:

Ergebnis Alice	qBit von Bob	σ_x	σ_z
$\|00\rangle$	$\alpha\|0\rangle+\beta\|1\rangle$	no	no
$\|01\rangle$	$\beta\|0\rangle+\alpha\|1\rangle$	yes	no
$\|10\rangle$	$\alpha\|0\rangle-\beta\|1\rangle$	no	yes
$\|11\rangle$	$-\beta\|0\rangle+\alpha\|1\rangle$	yes	yes

Das heißt aufgrund der Verschränkung wird auch Bob's qBit bei der Messung beeinflusst und in einen bestimmten Zustand der von $|x>$ abhängt gebracht. Man sieht an der Tabelle dass dann die kontrollierte Anwendung der σ_x und σ_z Transformationen dafür sorgt dass Bob's qBit im Zustand:

$$|x\rangle=\alpha|0\rangle+\beta|1\rangle$$

ist und somit von Alice an Bob übermittelt wurde. Dabei vertauscht σ_x die Koeffizienten und σ_z invertiert den zweiten Koeffizienten.

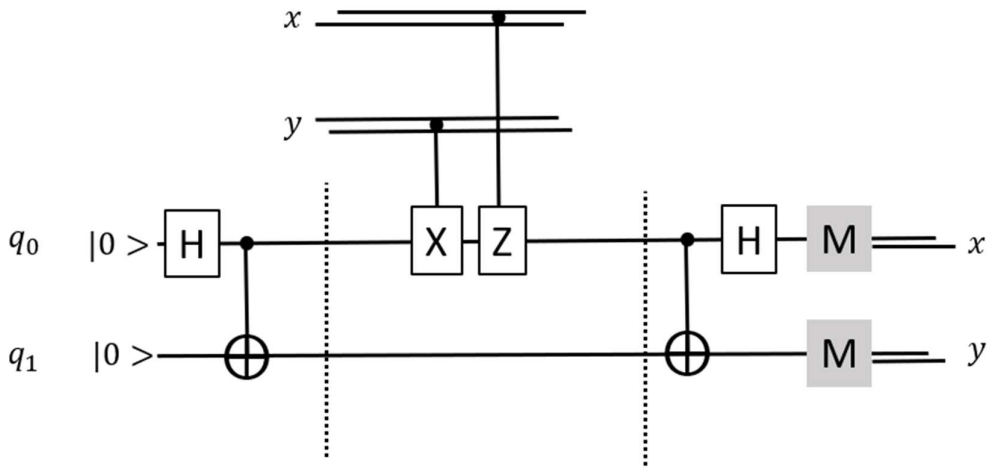
Nach der Messung von Alice hat sich der Zustand des qBits q_0 von Alice verändert und sie kann den Zustand $|x>$ nicht mehr rekonstruieren. Dies ist aufgrund des No Cloning Theorems evident.

Dichte Kodierung

Bei der „dichten Kodierung" werden 2 klassische Bits in ein qBit gepackt. Alice hat 2 klassische Bits die sie an Bob übertragen möchte. Sie teilen sich wie bei der Teleportation gemeinsam ein Bell Paar.

Algorithmus

Um die Information zu übertragen wird folgender Algorithmus benutzt:



Alice wendet also in Abhängigkeit von *x* und *y* die Transformationen σ_x und σ_z auf das qBit q_0 an das zuvor mit q_1 in einen Bell State verkoppelt wurde.

Analyse

Damit ergeben sich folgende Transformationen:

xy	**CX/CZ**	**Bell Messung**
00	$\frac{1}{\sqrt{2}}(\|00\rangle + \|11\rangle)$	00
01	$\frac{1}{\sqrt{2}}(\|10\rangle + \|01\rangle)$	01
10	$\frac{1}{\sqrt{2}}(\|00\rangle - \|11\rangle)$	10
11	$\frac{1}{\sqrt{2}}(-\|10\rangle + \|01\rangle)$	11

Man sieht dass die Messungen *x* und *y* reproduzieren. Die „dichte Kodierung“ hat keine praktische Bedeutung. Sie beleuchtet jedoch einen weiteren interessanten Aspekt der Verschränkung zweier qBits. Zusammen mit der Teleportation kann man also bezüglich Kommunikationsprozessen folgende Äquivalenz beobachten:

$$1\ qBit \iff 2\ klassische\ Bit$$

CHSH Game

Da die Verschränktheit zweier qBits irgendwie mysteriös zu sein scheint werden immer wieder Überlegungen gemacht wie man ihre Richtigkeit belegbar nachweisen könnte. Diese Überlegungen werden als Bell Test bezeichnet. Das CHSH Spiel ist ein Gedankenexperiment das sich um diesen Umstand dreht.

Set Up

Es gibt 2 Spieler die wir Alice und Bob nennen und einen Spielleiter den wir Richard nennen. Richard wird im Spiel fragen an Alice und Bob stellen. Alice und Bob können sich vor Beginn des Spiels über eine Spielstrategie einigen, sie können jedoch während des Spiels nicht miteinander kommunizieren. Um das Spiel zu formalisieren wählen wir folgende Bezeichnungen:

Richard hat 2 verschiedene Fragen. Frage an Alice ist x. Frage an Bob ist y. Alice und Bob können 2 verschiedene Antworten geben. Antwort von Alice ist a. Antwort von Bob ist b. Richard wählt seine Frage zufällig aus, entscheidet aber nach einem festen Schema eine bestimmte Kombination von xy und a, b ist ein win, sonst ein loss. Dieses Schema ist Alice und Bob bekannt.

Wir machen folgende Annahmen:

1. Fragen und Antworten sind alle bits $x, y, a, b \in \{0,1\}$
2. Die Fragen werden zufällig ausgewählt $p(0,0) = p(0,1) = p(1,0) = p(1,1)$
3. Win bedeutet $a \oplus b = x \wedge y$

Richard xy	A antwortet a	B antwortet b	$a \oplus b$	$x \wedge y$	Resultat	a=b
00	0	0	0	0	win	y
01	0	0	0	0	win	y
10	0	0	0	0	win	y
11	0	0	0	1	loss	y
00	0	1	1	0	loss	n
01	0	1	1	0	loss	n
10	0	1	1	0	loss	n
11	0	1	1	1	win	n
00	1	0	1	0	loss	n
01	1	0	1	0	loss	n
10	1	0	1	0	loss	n
11	1	0	1	1	win	n
00	1	1	0	0	win	y
01	1	1	0	0	win	y
10	1	1	0	0	win	y
11	1	1	0	1	loss	y

Analyse

Alice und Bob können sich vor dem Spiel nun überlegen ob sie eine deterministische Strategie wählen oder eine probabilistische. Bei der deterministischen Strategie überlegen sie sich ein eigenes Schema bei dem ihre Antworten jeweils eine Funktion in Abhängigkeit der an sie gestellten Frage ist. Also:

$$a = f_a(x)$$

$$b = f_b(y)$$

Es gibt 4 Möglichkeiten für jede der Funktionen f_a und f_b :

x		f_1	f_2	f_3	f_4
0		0	0	1	1
1		0	1	0	1

Man kann also 16 Möglichkeiten bilden die Funktionen zu kombinieren und erhält damit 16 unterscheidbare Antwortschemata:

$$(f_{a1}, f_{b1}), (f_{a1}, f_{b2}) \dots (f_{a4}, f_{b4})$$

Keine der Kombinationen aus diesen Funktionen ergibt jedoch eine sichere Gewinnchance. Dies sieht man wenn man die Bedingung 3 von oben, also $a \oplus b = x \wedge y$ analysiert. Man kann aus der Tabelle ablesen wie sich die einzelnen x, y, a, b Kombinationen die zu einem Win führen verhalten:

xy	win
00	$a = b$
01	$a = b$
10	$a = b$
11	$a \neq b$

Das heißt abhängig von der Fragenkombination xy reicht es aus zu entscheiden ob $a = b$ oder $a \neq b$ ist. Da aber Alice nicht weiß welche Frage an Bob gestellt wurde und Bob nicht weiß welche Frage an Alice gestellt wurde würde man ein Schema benötigen das ohne dieses Wissen in allen Fällen die richtige Entscheidung trifft. Keine der 16 Möglichkeiten erfüllt diese Bedingung. Als Beispiel wählen wir (f_{a1}, f_{b1}):

x	f_{a1}	f_{b1}
0	0	0
1	0	0

Dann ergibt sich:

xy	$f_{a1}(x)$	$f_{b1}(y)$	Comp
00	0	0	$a = b$
01	0	0	$a = b$
10	0	0	$a = b$
11	0	0	$a = b$

Dies lässt sich auch für die anderen 15 verbleibenden Kombinationen zeigen. Man kann zeigen dass mit einer deterministischen Strategie eine maximale Win Wahrscheinlichkeit von 0.75 erreichbar ist. Dies ist zum Beispiel der Fall wenn Alice und Bob immer mit 0 antworten wie es im oben betrachteten Fall skizziert wird.

Für eine probabilistische Strategie gilt dass sie als zufällige Wahl einer deterministischen Strategie betrachtet werden kann. Alice und Bob entscheiden sich zufällig für eine der 16 Kombinationen der oben angegebenen Funktionenpaare. Da aber der Durchschnitt der Win Wahrscheinlichkeit nicht grösser sein kann als das Maximum kann auch mit einer probabilistischen Methode keine Verbesserung erzielt werden. Wir folgern also dass 0.75 die beste erreichbare Win Wahrscheinlichkeit ist.

Nun stellt sich die Frage ob man diese Wahrscheinlichkeit mittels eines verschränkten qBit Paares verbessern kann. Wir werden sehen wie sich dies unter der Voraussetzung dass ein Bell Paar zur Verfügung steht tatsächlich realisieren lässt.

Wir haben an den Paulitransformationen gesehen dass sich diese als Drehungen des Einheitsvektors in der von den Basisvektoren $|0>$ und $|1>$ aufgespannten Ebene betrachten lassen. Dies lässt sich dahingehend verallgemeinern dass es zu einem beliebigen Winkel θ eine unitäre Transformation U_θ gibt die eine Drehung in der $|0>$ - $|1>$ Ebene bewirkt.

Alice und Bob verfolgen nun die folgende Strategie. Sie teilen sich wie bei den Betrachtungen zuvor ein Bell Paar.

Alice wendet auf ihr qBit folgende Transformation an:

U_0 für x=0
$U_{\pi/4}$ für x=1

Dann misst sie ihr qBit und schickt das Ergebnis an Richard.

Bob wendet auf sein qBit folgende Transformation an:

$U_{\pi/8}$ für y=0
$U_{-\pi/8}$ für y=1

Dann misst er sein qBit und schickt das Ergebnis an Richard. Wir betrachten zur Analyse exemplarisch den Fall x=0 und y=0 für den die Win Bedingung a=b ist. Unser Bell Paar befindet sich im Zustand:

$$|\phi^+>=\frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$$

Alice wendet nun U_0 auf das erste Bit an und Bob $U_{\pi/8}$ auf das zweite. Mit dem Tensorprodukt können wir schreiben:

$$(U_0 \otimes U_{\pi/8})\,|\phi^+>=\frac{1}{\sqrt{2}}\,(\alpha|00>+\beta|01>+\gamma|10>+\delta|11>)$$

mit:

$$\alpha=cos\left(\frac{-\pi}{8}\right) \qquad \beta=cos\left(\frac{-5\pi}{8}\right) \qquad \gamma=cos\left(\frac{3\pi}{8}\right) \qquad \delta=cos\left(\frac{-\pi}{8}\right)$$

Nun ergibt sich aus diesem Zustand für die Wahrscheinlichkeiten:

ab	**p**	**p**
00	$\alpha^2=\frac{2+\sqrt{2}}{8}$	0.42
01	$\beta^2=\frac{2-\sqrt{2}}{8}$	0.07
10	$\gamma^2=\frac{2-\sqrt{2}}{8}$	0.07
11	$\delta^2=\frac{2+\sqrt{2}}{8}$	0.42

Das heisst die Wahrscheinlich dass $a=b$ ist 0.42+0.42=0.84 und damit signifikant höher als 0.75! Die anderen Fälle lassen sich analog berechnen.

Für Nachweise dieser Art gab es 2022 für Alain Aspect, John Clauser und Anton Zeilinger den Physik Nobelpreis.

Grover Algorithmus

Im Abschnitt Komplexitätsklassen haben wir bereits die sogenannten Entscheidungsprobleme erwähnt die nur 2 Ergebnisse erlauben die wir mit 0 und 1 bezeichnet haben. Ein Suchproblem bei dem wir aus einer Menge an Datensätzen D ein Element x_0 suchen welches ein bestimmtes Kriterium erfüllt können wir als ein solches Entscheidungsproblem formulieren:

$$f: D \to \{0,1\} \qquad \forall x \in D : \exists! \, x_0 \in D: f(x_0) = 1$$

Es gibt also genau ein einziges Element x_0 in der Menge D welches unser Suchkriterium erfüllt und genau für dieses Element ist unser Funktion f gleich 1, während für alle anderen Elemente f gleich 0 ist. Geht man mit klassischen Mitteln vor wertet man die Funktion f in sequentieller Reihenfolge für die einzelnen Elemente $x \in D$ aus bis das Funktionsergebnis 1 beträgt. Gibt es in der Menge an Datensätzen N Elemente so braucht man statistisch $N/2$ Auswertungen von f. Man kann die Problemstellung auch dahingehend formulieren dass man bei gegebenem f versucht ein x_0 zu finden für das $f(x_0) = 1$ ist. Dabei kann f zum Beispiel durch Boole'sche Verknüpfungen gegeben sein. Ein triviales Beispiel wäre:

$$x \in \{0,1\}^n \qquad f(x) = x_1 \wedge x_2 \cdots \wedge x_n \qquad \text{Lösung: } x_0 = (1,1,\cdots,1)$$

In erster Linie stehen hier sogenannte SAT Probleme, also Erfüllbarkeitsprobleme der Aussagenlogik, im Vordergrund. Die hier zur Anwendung kommenden aussagenlogischen Formeln sind UND, ODER und NICHT Verknüpfungen von Boole'schen Variablen die in der disjunktiven oder konjunktiven Normalform vorliegen. Für komplexere Boole'sche Funktionen ist es sehr schwierig solche Lösungen zu finden.

Algorithmus

Zur Bearbeitung dieses Suchproblems mittels eines Quantenalgorithmus ist es notwendig die Menge D an Datensätzen zu enkodieren indem jeder Datensatz x durch einen Binärcode gekennzeichnet wird. Damit ist:

$$D \to \{0,1\}^n \quad \text{und} \quad f: D \to \{0,1\} \quad \text{mit} \quad \forall x \in D : \exists! \, x_0 \in D: f(x_0) = 1$$

Für einen Datensatz mit N Elementen ist dann $n = log_2 N$.
Wie wir bereits beim Deutsch Josza Algorithmus gesehen haben benötigen wir um die Funktion f auszuwerten eine geeignete unitäre Transformation:

$$U_f(|x> \otimes |y>) = |x> \otimes (|y \oplus f(x)>) \quad \text{mit} \quad x \in \{0,1\}^n \quad y \in \{0,1\}$$

Diese Transformation verändert unseren Gesamtzustand entsprechend der Methodik die wir als Phase Kickback bereits früher erläutert haben. In Superposition kann $|x>$ alle Datensätzen der Menge D gleichzeitig repräsentieren. Die Idee des Algorithmus ist nun eine Amplitudenverstärkung für ein bestimmtes ausgewähltes Element x_0 für das $f(x_0) = 1$ zu erzielen.
Es gilt für $x \in \{0,1\}^n$ und $y \in \{0,1\}$:

$$U_f(|x> \otimes |->) = (-1)^{f(x)} \cdot (|x> \otimes |->)$$

Da nur für das Element x_0 gilt $f(x_0) = 1$ wird also lediglich für dieses Element die Amplitude invertiert während alle anderen Amplituden gleich bleiben. Der Grover Algorithmus nutzt diesen Umstand indem er daraus sukzessive den gesuchten Zustand verstärkt. Nehmen wir an wir hätten 4 Elemente im Datensatz D und unser gesuchter Datensatz wäre der mit der Nummer 2. Mit 2 qBits können wir zunächst einen gleichverteilten Zustand herstellen. Da 4 Zustände vorliegen hat bei einer

Gleichverteilung jeder Zustand die Wahrscheinlichkeit $1/4$ mit der Amplitude $A_1 = 1/2$. Nach Anwendung von U_f gilt dann für A_2:

x	A_1	A_2	$A_3 = 2m - A_2$
00	.5	.5	0
01	.5	-.5	1
10	.5	.5	0
11	.5	.5	0
Σ	2	1	
m	.5	.25	

Da eine Messung für alle Datensätze das gleiche Resultat ergeben würde spiegelt der Algorithmus alle Werte am Mittelwert m und erhält damit die Amplitude A_3.

Betrachtet man größere Werte von N fällt die Amplitudenverstärkung schwächer aus. Die Gleichverteilung ergibt Anfangsamplituden $A_1 = \frac{1}{\sqrt{N}}$. Für $N = 100$ wäre also $A_1 = 0.1$ und damit $m = 0.08$ womit die Amplitude des gesuchten Zustandes nach der Spiegelung sich zu 0.26 (p=0.067) ergibt während die anderen Amplituden etwa 0.08 (p=0.0064) groß sind. Man sieht also dass unter Umständen mehrere Iterationen nötig sind bis der gesuchte Datensatz sich signifikant von den anderen Amplituden abhebt.

Nachdem wir die schematische Vorgehensweise des Grover Algorithmus betrachtet haben stellt sich nun zum einen die Frage wie dieser durch unitäre Transformationen realisiert werden kann und zum anderen wie viele Iterationen sinnvoll sind um ein vernünftiges Ergebnis zu erzielen. Die Erzeugung eines gleichverteilten Zustandes über n qBits haben wir bereits an verschiedenen Beispielen kennengelernt. Sie ergibt sich durch die Anwendung eines Hadamard Gatters auf jedes einzelne qBit.

Der Mittelwert einer Verteilung mit einer Amplitude a_0 und restlichen Amplituden a_1 ist:

$$m = \frac{1}{N}(-a_0 + (N-1) \cdot a_1)$$

Spiegelt man die Amplituden *a* einer Verteilung am Mittelwert *m* so ergeben sich die neuen Amplituden *b* zu:

$$b = 2m - a$$

Für den gesamten Datensatz den wir durch *N* unterscheidbare Zustände eines n qBit Registers enkodieren gilt dann:

$$\sum_{i=1}^{N} a_i|i> \rightarrow \sum_{i=1}^{N}(2m - a_i)|i>$$

Diese Transformation lässt sich durch die *NxN* Matrix D_N beschreiben, die wiederum aus elementaren Matrizen erzeugt werden kann die wir teilweise bereits kennengelernt haben:

$$D_N = -H_n \cdot R_N \cdot H_n$$

Wobei H_n die Hadamardmatrizen auf n qBits sind und R_N:

$$R_N = \begin{matrix} -1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{matrix}$$

R_N ändert also das Vorzeichen des Zustandes $|00 \ldots 0>$. Es stellt sich nun die Frage wie diese Operation dahingehend heruntergebrochen werden kann dass sie aus Operationen zusammengesetzt werden kann die „lokal" sind. Es lässt sich zum Beispiel zeigen dass R_4 nicht als Tensorprodukt von *2x2* Matrizen geschrieben werden kann. Beim Deutsch Algorithmus haben wir im Zuge der Phase Kickback Diskussion bereits den bedingten Vorzeichenwechsel kennengelernt der durch Einführung eines Hilfsbits $|y>$ im Zustand $|->$ durch eine entsprechende unitäre Transformation dazu führt dass für einen selektierten Zustand das Vorzeichen der Amplitude invertiert wird. Ein Vorzeichenwechsel in $|x>$ findet statt für $f(x) = 1$.

$$U_f(|x>\cdot\ |->) = \ |x>\cdot(-1)^{f(x)}|->= (-1)^{f(x)}|+>\cdot|->$$

Wir definieren allgemein als bedingten Vorzeichenwechsel V_f die folgende Transformation:

$$f: \{0,1\}^n \to \{0,1\} \qquad V_f: |x> \to (-1)^{f(x)} \cdot |x>$$

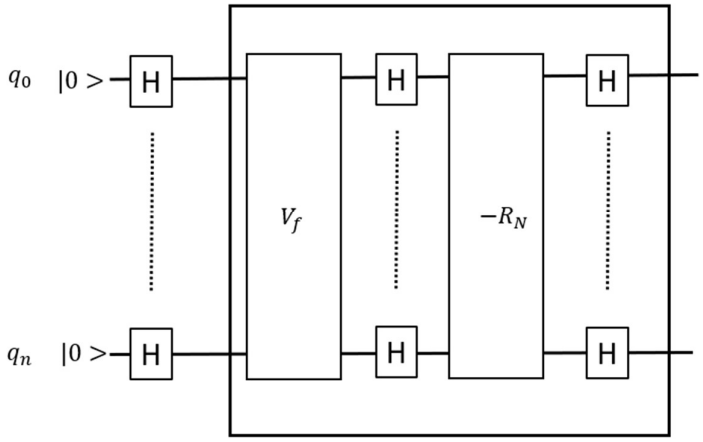
R_N ist nun einfach ein Spezialfall eines bedingten Vorzeichenwechsels. Er führt den Vorzeichenwechsel nur auf den Zustand $|00 \ldots 0>$ aus. Wählen wir unsere Funktion *f* also so dass sie nur für diesen Zustand gleich 1 ist und sonst 0 so lässt sich R_N durch U_f erzeugen:

$$U_f: |\mathrm{x}>\ |y> \to: |\mathrm{x}>\ |y \oplus f(x)> \qquad \text{mit} \qquad f(x_n \ldots \ldots x_1) = \begin{cases} 1 \text{ für } x_n = \cdots = x_0 = 0 \\ 0 \text{ sonst} \end{cases}$$

Diese unitäre Transformation lässt sich in einen Quantenschaltkreis umwandeln und damit lässt sich unsere Anwendung von R_N realisieren. Des weiteren lässt sich mit V_f der gesuchte Zustand markieren, der ja im Gegensatz zu den restlichen Zuständen vor der Spiegelung invertiert werden soll.

Quantenschaltkreis

Auf Gatter Ebene lässt sich der Grover Algorithmus ohne Hilfsbit nun so darstellen:



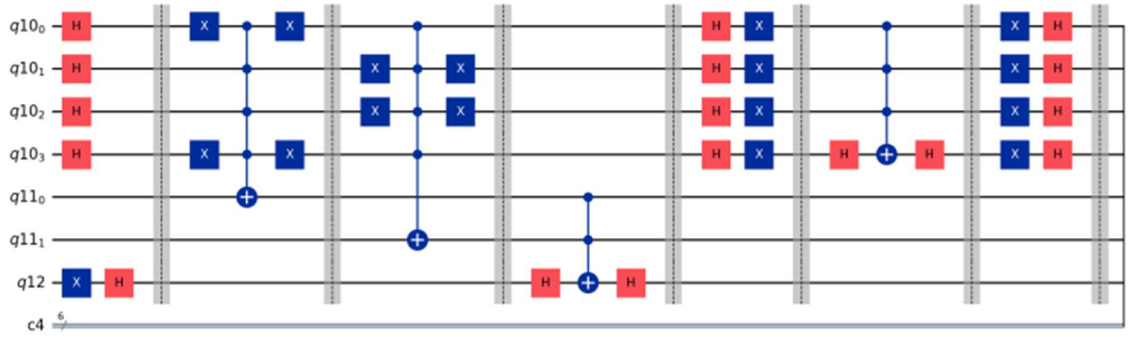
Die in der Box platzierten Elemente stellen eine Grover Iteration dar. Die Transformationen V_f und R_N zu implementieren führt zu komplexen Schaltkreisen wie wir sehen werden.

Wir haben für den Fall *N=4* gesehen dass bereits eine Iteration ausreicht um den gesuchten Fall zu isolieren. Für größere N haben wir festgestellt dass mehrere Iterationen notwendig sein können um ein signifikantes Ergebnis zu bekommen. Vor allem gibt es einen optimalen Zeitpunkt die Iteration zu beenden da sich die Signifikanz des Ergebnisses nach der Überschreitung dieses Optimums wieder verschlechtert. Man kann sich anhand geometrischer Überlegungen bei denen man die einzelnen Transformationen als Drehungen betrachtet überlegen welche Anzahl an Iterationen optimal ist. Dabei ergibt sich:

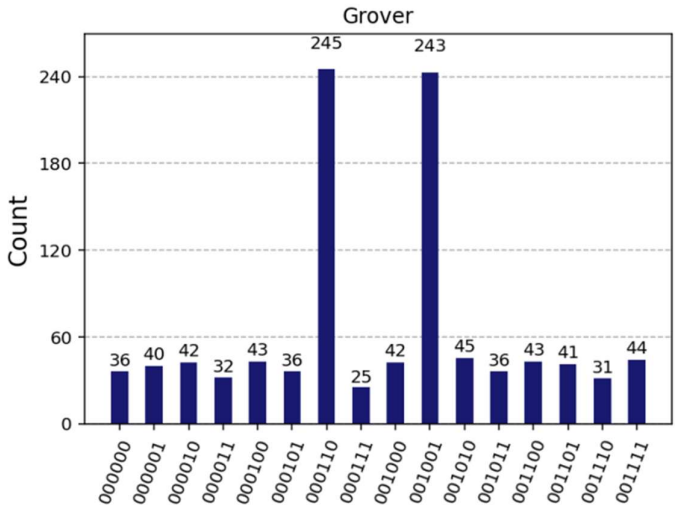
$$k = \frac{\pi}{4} \cdot \sqrt{N}$$

Für unser obiges Beispiel für *N=100* wären also etwa 8 Iterationen optimal.

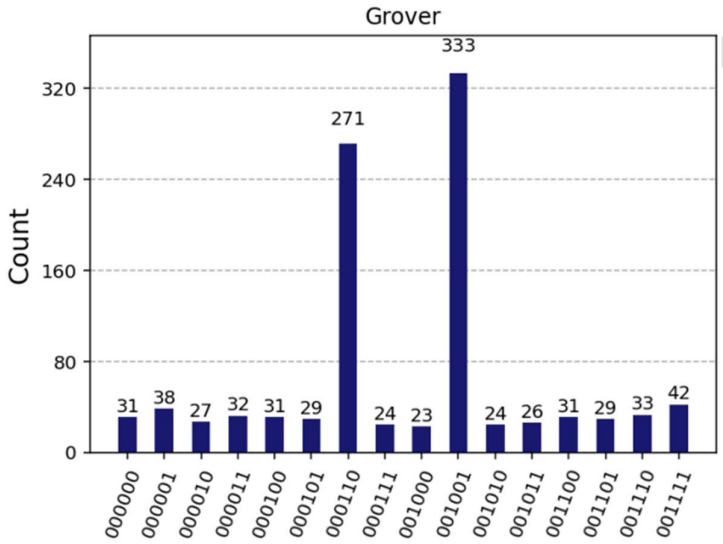
Der folgende Quantenschaltkreis zeigt die Implementierung einer Iteration für 4 qBits plus zwei Hilfs qBits für die markierten Werte 0110 und 1001. Die 0 markierten Werte werden aufgrund der Big Endian Konvention mittels X Gates markiert:



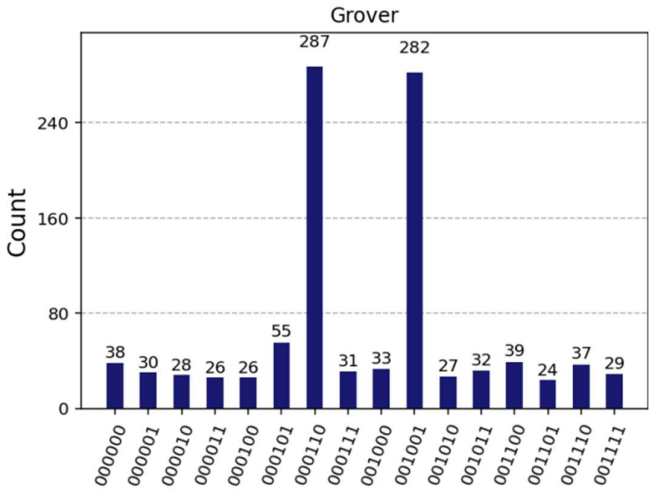
Die Ergebnisse nach aufsteigender Iteration sehen dann wie dargestellt aus:



Nach einer Iteration



Nach 3 Iterationen



Nach 7 Iterationen

Man sieht dass sich das Ergebnis nach circa 3 Iterationen die bei $n = 4$ etwa $\frac{\pi}{4} \cdot \sqrt{N} = \pi$ entsprechen wieder verschlechtert.

Quanten Kommunikation

Die Idee der sicheren Datenübertragung von Alice zu Bob ist dass Eve die übertragenen Daten zwar abhören kann, die Daten aber so verschlüsselt sind dass nur Alice und Bob die Daten entschlüsseln können weil sie den entsprechenden Ver- und Entschlüsselungscode besitzen. Dies entspricht dem Kerkhoffschen Prinzip welches in seinem Kern besagt dass die Sicherheit eines Verfahrens nicht durch die Geheimhaltung des Algorithmus sondern durch die Geheimhaltung des Schlüssels gewährleistet werden soll. Damit kommt dem Schlüsselcode und seiner Übertragung eine besondere Bedeutung zu. Aufgrund des No Cloning Theorems gibt der Austausch des Schlüsselcodes Hinweise darauf ob die Übertragung abgehört wurde oder nicht. Dadurch lassen sich Schlüssel sicher austauschen.

Messbasen

Wir hatten bereits erwähnt dass man qBits nicht nur in der Standardbasis $|0>$ und $|1>$ messen kann sondern alternativ auch die durch die Hadamard Transformation aus der Basis $|0>$ und $|1>$ erzeugten orthogonalen Zustände $|+> \text{ und } |->$ als Basis dienen können. In einer 2 dimensionalen Ebene kann man dabei einfach die Projektionen des Zustandsvektors auf die entsprechenden Basisachsen als Messergebnis interpretieren. Für die Zustände $|+> \text{ und } |->$ gilt umgekehrt dass die Hadamard Transformation sie wieder auf $|0>$ und $|1>$ abbildet. Man kann diese Vorgehensweise verallgemeinern indem man beliebige Drehwinkel bezüglich der $|0>$ und $|1>$ Achsen betrachtet. Dann gilt:

$$U_\alpha|\alpha_0> = |0> \qquad\qquad U_\alpha|\alpha_1> = |1>$$

Möchte man nun $|\alpha_0>$ bezüglich der Basis $\{|\alpha_0>, |\alpha_1>\}$ messen, so muss zunächst die Transformation U_α auf $|\alpha_0>$ angewendet werden und dann bezüglich der Standardbasis gemessen werden. Man misst dann sicher $|0>$ und mit der selben Begründung für $|\alpha_1>$ misst man $|1>$.

Misst man eines der beiden qBits eines Bell States so ergibt sich in der Standardbasis mit Wahrscheinlichkeit $1/2$ das Ergebnis $|0>$ mit einem anschliessenden Übergang des

Gesamtzustandes in $|00>$ und mit Wahrscheinlichkeit $1/2$ das Ergebnis $|1>$ mit einem anschliessenden Übergang des Gesamtzustandes in $|11>$. Bei Messungen von Bell States in beliebigen Basen gilt:

$$|\alpha_0> = \cos\alpha\ |0> + \sin\alpha\ |1> \qquad |\alpha_1> = -\sin\alpha\ |0> + \cos\alpha\ |1>$$

Hat man nun den auf die Basis $\{|\alpha_0>, |\alpha_1>\}$ bezogenen Bellstate:

$$|\phi^+> = \frac{1}{\sqrt{2}}(|\alpha_0\alpha_0> + |\alpha_1\alpha_1>)$$

$$= \frac{1}{\sqrt{2}}(|\alpha_0> \otimes |\alpha_0> + |\alpha_1> \otimes |\alpha_1>)$$

$$= \frac{1}{\sqrt{2}}\big((\cos\alpha^2 + \sin\alpha^2)|00> + (\sin\alpha^2 + \cos\alpha^2)\,|11>\big)$$

$$= \frac{1}{\sqrt{2}}(|00> + |11>)$$

$$= |\phi^+>$$

Aus dieser Betrachtung folgt dass der Bell State $|\phi^+>$ bei einer Messung in einer beliebigen Basis mit Wahrscheinlichkeit $1/2$ das Ergebnis $|\alpha_0>$ mit einem anschliessenden Übergang des Gesamtzustandes in $|\alpha_0\alpha_0>$ und mit Wahrscheinlichkeit $1/2$ das Ergebnis $|\alpha_1>$ mit einem anschließenden Übergang des Gesamtzustands in $|\alpha_1\alpha_1>$ ergibt. Man misst also bei der Messung des zweiten qBits stets das selbe Ergebnis wie beim ersten. Man sagt $|\phi^+>$ ist in allen Basen perfekt korreliert. Man kann diese Betrachtungen für alle Bell States anstellen und stellt fest dass sie sich unterschiedlich verhalten. Die Zustände mit gemischten Paaren $|01>$ oder $|10>$ sind zum Beispiel antikorreliert. Dies bedeutet das nach der Messung des ersten qBits mit $|0>$ das zweite qBit auf jeden Fall mit $|1>$ gemessen wird und umgekehrt. Man erhält folgendes Ergebnis:

State/Base	$\|0>$ $\|1>$	$\|+>$ $\|->$	$\|\alpha_0>\|\alpha_1>$
$\|\phi^+> = \frac{1}{\sqrt{2}}(\|00\rangle + \|11\rangle)$	perfekt korreliert	perfekt korreliert	perfekt korreliert
$\|\phi^-> = \frac{1}{\sqrt{2}}(\|00\rangle - \|11\rangle)$	perfekt korreliert	perfekt antikorreliert	partiell korreliert
$\|\psi^+> = \frac{1}{\sqrt{2}}(\|01\rangle + \|10\rangle)$	perfekt antikorreliert	perfekt korreliert	partiell korreliert
$\|\psi^-> = \frac{1}{\sqrt{2}}(\|01\rangle - \|10\rangle)$	perfekt antikorreliert	perfekt antikorreliert	perfekt antikorreliert

Quantenkanal

In der Quantenkommunikation wird das System über das Quanteninformationen übertragen werden als Quantenkanal bezeichnet. Er verfügt über spezielle Eigenschaften die aus seinen quantenmechanischen Prinzipien resultieren. Ein Quantenkanal wird mathematisch durch einen

linearen unitären Operator der auf das zu übertragende qBit angewendet wird dargestellt. Wie wir bereits erläutert haben ist dadurch die für quantenmechanische Prozesse geforderte Reversibilität der Übertragung gewährleistet. Quantenkanäle werden durch unterschiedliche Fehlerquellen verfälscht und unterliegen daher dem Einfluss von Quantenrauschen das die übertragene Information verändert. Sie können verwendet werde um Verschränkungszustände über große Distanzen herzustellen. Aufgrund der erwähnten Linearität und Unitarität des beschreibenden Operators gilt auch für die Übertragung durch einen Quantenkanal das No Cloning Theorem. Die geschilderten Eigenschaften sind entscheidend für die Realisierung von Quantenverschlüsselungsprotokollen.

BB84 Protokoll

Es wird angenommen dass zwischen Alice und Bob ein rauschfreier Quantenkanal existiert auf dem sie Informationen in Form von qBits verschicken können. Um ihre geheime Information auszutauschen verwenden sie eine Folge unabhängiger Zufallsbits der Länge m als Schlüssel. Beide sind im Besitz dieses Schlüssels. Dieser Schlüssel hat die gleiche Bitlänge wie die Information die sie austauschen wollen. Die Information selber ist also auch eine Folge von Bits. Die Bits die Alice an Bob verschicken möchten seien $a_1 \dots a_m$ und die Bits des Schlüssels seien $k_1 \dots k_m$. Um die Information zu verschlüsseln schickt Alice nicht die Bits $a_1 \dots a_m$ durch den Kanal sondern sie bildet zusammen mit den Schlüsselbits die Bitfolge $a_1 \oplus k_1 \dots a_m \oplus k_m$ die dann an Bob übermittelt werden. Bezeichnen wir die bei Bob angekommenen Bits als $b_1 \dots b_m$ dann kann Bob mittels seiner Schlüsselbitfolge die Bits von Alice entschlüsseln indem er wie Alice $b_1 \oplus k_1 \dots b_m \oplus k_m$ bildet. Denn für ein beliebiges Bit gilt:

$$b_i \oplus k_i = (a_i \oplus k_i) \oplus k_i = a_i \oplus 0 = a_i$$

Mit diesem Verfahren bleibt nun die Frage offen wie es gelingen kann den Schlüssel $k_1 \dots k_m$ so zu übermitteln dass er nicht gefunden werden kann. Der Schlüssel muss zufällig sein und für jede Bitfolge neu aufgesetzt werden.

Der Prozess zur Erstellung der Schlüsselbits sieht folgende Schritte vor. Zunächst erzeugt Alice ein zufälliges Bit a_{init} mit dessen Wert man ein qBit $|x>$ initialisiert. Ziel ist es dieses Bit an Bob zu übermitteln. Dann erzeugt Alice ein zweites zufälliges Bit a_H das bestimmt ob man auf $|x>$ eine Hadamard Transformation anwendet oder nicht. Für $a_H = 1$ wird *H* angewendet, sonst nicht. Durch nachrechnen kann man sehen dass $|x>$ sich jeweils mit der Wahrscheinlichkeit $1/4$ in einem der 4 Zustände $|0>$, $|1>$, $|+>$, $|->$ befindet. Dies lässt sich an der folgenden Tafel sehen:

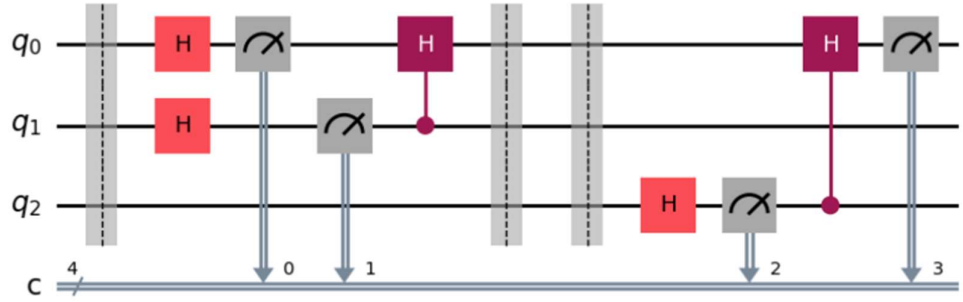
	a_H=0	a_H=1
a_{init}= 0	$\|0>$	$\|+>$
a_{init}= 1	$\|1>$	$\|->$

Die durch a_H bedingte Hadamard Transformation bestimmt in welchen der 4 Basisvektoren der Zustand $|x>$ sich transformiert hat. Dann wird $|x>$ durch einen Quantenkanal an Bob geschickt. Bob misst das erhaltene qBit. Da er nicht weiss was der Wert von a_H war kann er nicht sagen bezüglich welcher Basis er $|x>$ messen muss um a_{init} herauszufinden. War a_{init}= 0 und a_H= 0 und er misst bezüglich der Standardbasis so findet er den richtigen Wert. Misst er dagegen in der Hadamard Basis ist die Wahrscheinlichkeit dass er den richtigen Wert bekommt nur 0.5 , weshalb er seine Entscheidung über die Messbasis dem Zufall überlässt. Er erzeugt also seinerseits ein Zufallsbit b_H das entscheidet in welcher Basis er seine Messung an $|x>$ vornimmt. Ist $b_H = 0$ misst er in der Standardbasis, für $b_H = 1$ misst er in der Hadamard Basis. Bob's gemessenes Ergebnis ist b*:*

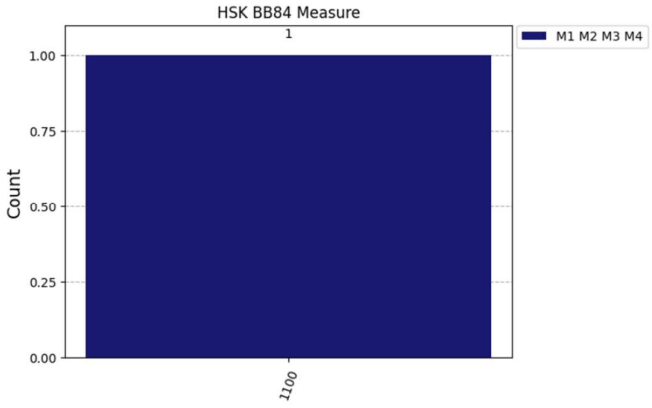
|x >	|0 > |1 >	|+> |−>
|0 >	0	0/1
|1 >	1	0/1
|+>	0/1	0
|−>	0/1	1

Das bedeutet wenn Bob und Alice verschiedene Basen verwenden ist Bob's Messung nur ein Zufallswert. Er teilt daher Alice mit in welcher Basis er gemessen hat indem er b_H über einen klassischen Kanal an sie übermittelt. Alice kann also entscheiden ob die verwendete Basis identisch war und Bob mitteilen ob das von ihm gemessene Bit für den Schlüssel verwendbar ist oder nicht. Statistisch ist zu erwarten dass in 50% der Fälle ein benutzbares Bit erzeugt wird.

Der entsprechende Quantenschaltkreis sieht dann so aus:



Die singulären Hadamard Gates sind für die Erzeugung der Zufallsbits verantwortlich. Alice erzeugt das zu übermittelnde Zufallsbit a_{init} zunächst auf q_0 und dann das über die Basis entscheidende Zufallsbit a_H auf q_1. Das controlled Hadamard Gate setzt dann die entsprechende Basis. Dann wird der durch q_0 repräsentierte Zustand |x > an Bob übergeben. Er erzeugt durch das Hadamard Gate auf q_2 das Zufallsbit b_H welches nach der Messung darüber entscheidet in welcher Basis er q_0 messen wird. Die 4 klassischen Bits die gemessen werden sind dann für eine Beispielmessung wie folgt belegt:



In diesem Fall ist das Ergebnis im grauen Feld für Bit1 abgebildet. Weitere Messungen ergaben die folgenden Bits:

	B1	B2	B3	B4	B5	B6	B7	B8
a_{init}	0	1	1	0	1	1	0	0

a_H	0	0	0	0	1	1	0	0
b_H	1	1	0	1	1	0	1	0
b	1	0	1	1	1	0	0	0
Res	**X**	**X**	**1**	**X**	**1**	**X**	**X**	**0**

Man sieht dass für die 8 Messungen 3 Fälle entstanden sind für die $a_H = b_H$ gilt. In diesem Fall haben Alice und Bob ein verwendbares Bit erhalten. Statistisch ist zu erwarten dass 50% der Bits verwendbar sind.

Neben Alice und Bob wird Eve als fiktiver Angreifer bezeichnet. Eve kennt das Verfahren das Alice und Bob zur Verschlüsselung ihrer Nachricht wie es oben geschildert wurde und kann sowohl den Quantenkanal als auch den klassischen Kommunikationskanal abhören. Über das Abhören des klassischen Kanals kennst sie sowohl a_H als auch b_H nachdem Bob $|\mathrm{x}>$ gemessen hat. Da die 4 möglichen Zustände von $|\mathrm{x}>$ nicht orthogonal sind kann Eve wegen des No Cloning Theorems keine Kopie von $|\mathrm{x}>$ erzeugen. Sie kann also das qBit nur messen und es dann an Bob weiterschicken oder alternativ mittels des erhaltenen Ergebnisses von $|\mathrm{x}>$ ein neues qBit erzeugen welches sie dann an Bob übermittelt. Wenn Eve die von Alice verwendete Basis kennen würde könnte sie dies unbemerkt tun. Da sie aber bezüglich der Messung in der selben Situation wie Bob steckt muss sie zur Wahl der Basis ein zufälliges Bit e_H annehmen mit dem sie statistisch in 50% der Fälle die richtige Wahl treffen wird. Für $e_H = 1$ misst sie dann in der Hadamard Basis und für $e_H = 0$ in der Standardbasis. Sie kann bei zufällig richtiger Wahl der Basis immerhin die Hälfte der verwendbaren Bits unbemerkt abhören. Für den Fall dass sie die falsche Basis wählt wird Alice und Bob jedoch bemerken dass das übertragene qBit $|\mathrm{x}>$ verändert wurde. Denn durch die Messung wird $|\mathrm{x}>$ auf den Zustand projeziert der dem Messergebnis in der falschen Basis entspricht. Hat also Alice zum Beispiel a_{init}= 0 und a_H= 1 gewählt schickt sie einen $|+>$ Zustand an Bob. Wählt Eve nun $e_H = 1$ misst sie in der Hadamard Basis und folgert dass Alice's a_{init}= 0 war und schickt dann wieder einen $|+>$ Zustand an Bob weiter weil sie $e_H = 1$ gewählt hatte. Der Vorgang bleibt unbemerkt und sie hat ein Bit des Schlüssels ausgehorcht. Wählt sie allerdings $e_H = 0$ misst sie in der Standardbasis und erhält zufällig 0 oder 1 und sie schickt entsprechend ihres Ergebnisses einen $|0>$ oder $|1>$ Zustand an Bob. Für den Fall dass Alice und Bob die Basis gleich gewählt haben müssen ihre Messergebnisse von $|\mathrm{x}>$ identisch sein. Ist dies nicht der Fall so wissen sie dass $|\mathrm{x}>$ manipuliert wurde. Dieser Fall tritt mit einer Wahrscheinlichkeit von 25% auf. Man erhält also folgende Kombinationen:

$\boldsymbol{a_{init}}$	$\boldsymbol{a_H = e_H}$ $\boldsymbol{a_H = b_H}$	$\boldsymbol{a_H \neq e_H}$ $\boldsymbol{a_H \neq b_H}$	$\boldsymbol{a_H \neq e_H}$ $\boldsymbol{a_H = b_H}$	$\boldsymbol{a_H = e_H}$ $\boldsymbol{a_H \neq b_H}$
	a->e->b	**a->e->b**	**a->e->b**	**a->e->b**
0	0->0->0	0->0/1->0/1	0->0/1->0/1	0->0->0/1
1	1->1->1	1->0/1->0/1	1->0/1->0/1	1->1->0/1
Erfolgsbeitrag	25%	0%	12.5%	0%

Im Fall der grau unterlegten Felder können Alice und Bob mit jeweils 50% Wahrscheinlichkeit feststellen ob sie abgehört wurden da Eve aufgrund von $a_H \neq e_H$ das qBit $|\mathrm{x}>$ in der falschen Basis weitergibt. Für die einzelnen Fälle lässt sich feststellen:

- Spalte 1:
  Der Lauschangriff findet unbemerkt statt und führt zum richtigen Ergebnis für Alice, Bob und Eve

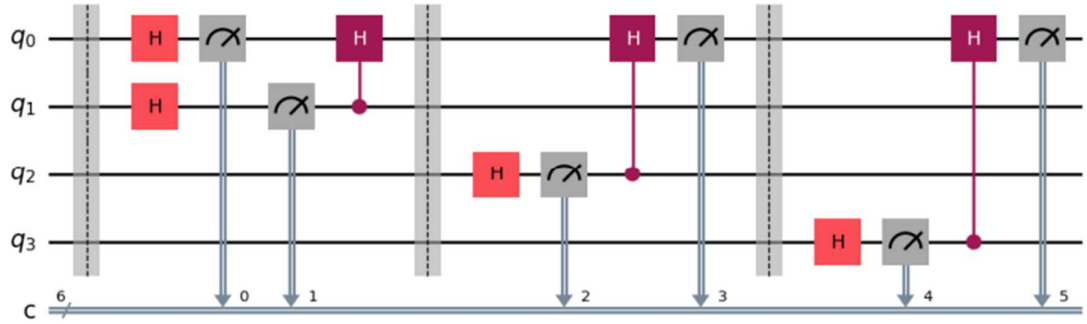
- Spalte 2:
  Der Lauschangriff findet unbemerkt statt und führt zu einem Zufallsergebnis

- Spalte 3:
  Der Lauschangriff wird zu 50% bemerkt und führt zu einem Zufallsergebnis

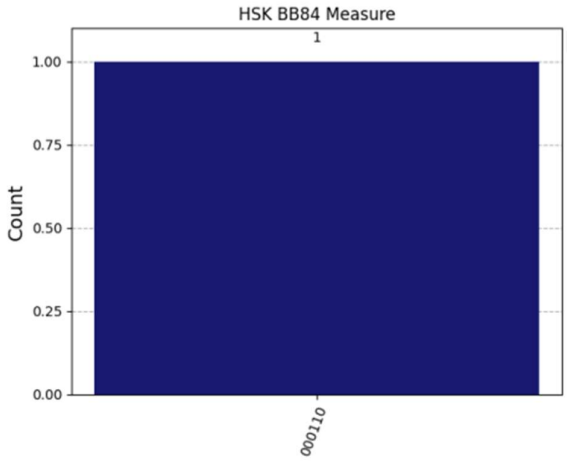
- Spalte 4:
  Der Lauschangriff findet unbemerkt statt und führt zu einem Zufallsergebnis

Das bedeutet insgesamt dass Alice und Bob bei der diskutierten Übermittlung damit rechnen müssen dass die korrekt übertragenen Bits 50% betragen wenn sie nicht abgehört werden weil $a_H = b_H$. Im Falle dass sie abgehört werden wird jedoch aufgrund der Intervention von Eve die Erfolgsrate der Fälle in denen Eve bezüglich der Basis von Alice falsch liegt noch einmal halbiert. Dadurch reduzieren sich die korrekt übertragenen Bits in diesem Fall auf 37.5%. Tauschen Alice und Bob also m Bits aus und werten dann ihre Erfolgsrate aus dann wissen sie das sie abgehört wurden wenn diese signifikant unter 50% liegt.

Folgender Quantenschaltkreis spiegelt das Verhalten von Eve wieder:



Eve erzeugt also im mittleren Teil ihrerseits ein Zufallsbit auf q_2 welches entscheidet in welcher Basis sie misst. Wie oben ergeben sich dann folgende Messungen:



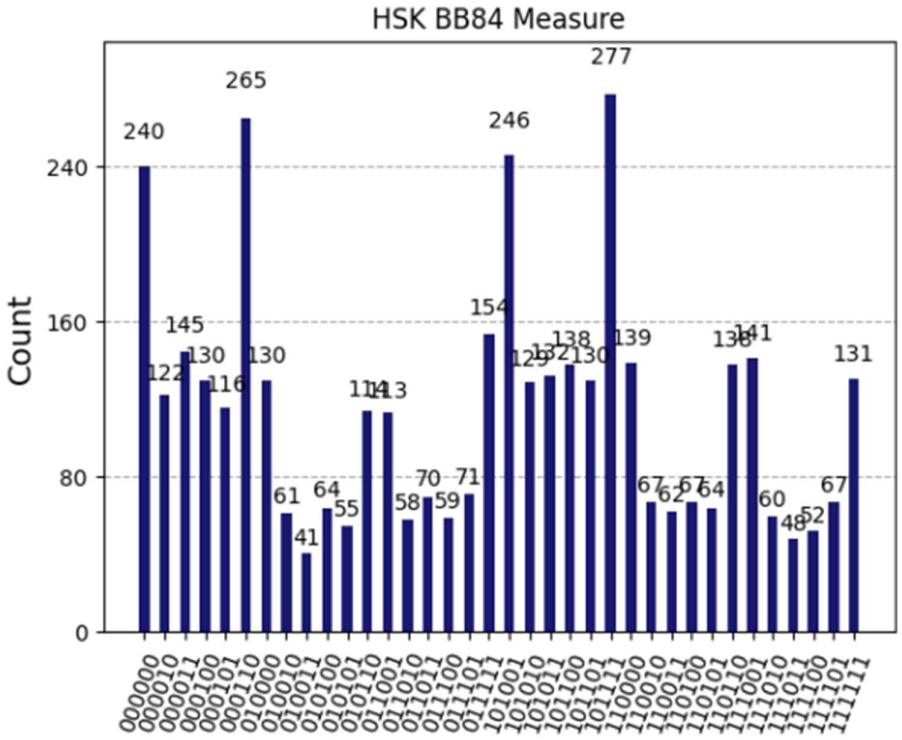
Mit den entsprechenden Ergebnissen:

	B1	B2	B3	B4	B5	B6	B7	B8
a_{init}	0	0	1	0	1	0	0	0
a_H	1	0	0	1	0	0	0	1
e_H	1	1	0	1	0	1	0	1
e	0	1	1	0	1	0	0	0
b_H	0	0	0	0	0	0	1	0
b	0	1	1	0	1	0	1	0
Res	**X**	**L**	**1**	**X**	**1**	**0**	**X**	**X**

Beurteilt man nun die einzelnen Ergebnisse sieht man:

- B1
  $a_H \neq b_H$ die Messung wird von Alice und Bob nicht beachtet
- B2
  $a_H = b_H$ die Messung wäre also richtig, da aber $a_{init} \neq b$ ist klar dass abgehört wurde
- B3
  $a_H = b_H$ die Messung wird von Alice und Bob akzeptiert, da gleichzeitig auch $a_H = e_H$ bleibt der Lauschangriff unbemerkt
- B4
  $a_H \neq b_H$ die Messung wird von Alice und Bob nicht beachtet
- B5
  $a_H = b_H$ die Messung wird von Alice und Bob akzeptiert, da gleichzeitig auch $a_H = e_H$ bleibt der Lauschangriff unbemerkt
- B6
  $a_H = b_H$ die Messung wird von Alice und Bob als richtig eingeschätzt. Obwohl $a_H \neq e_H$ bekommt Eve das korrekte e. Der Lauschangriff bleibt unbemerkt und liefert für Eve das richtige Ergebnis
- B7
  $a_H \neq b_H$ die Messung wird von Alice und Bob nicht beachtet
- B8
  $a_H \neq b_H$ die Messung wird von Alice und Bob nicht beachtet

Macht man nun sehr viele Messungen so ergibt sich eine solche Statistik:



Man kann dann alle Fälle für die gilt $a_H = b_H$ und $a_{init} = b$ identifizieren und sieht an der Statistik ob abgehört wurde oder nicht. Schaltet man Eve wie im ersten Quantenschaltkreis gezeigt aus beziehungsweise wie im zweiten Quantenschaltkreis ein so verändert sich die Statistik signifikant wie vorhergesagt. Es ergibt sich folgende Statistik:

Eve eingeschaltet:
shot_count :4096
loop_count :36
stat_count :1265
percentage :0.308837890625

Schaltet man hingegen Eve aus ergibt sich:

shot_count :4096
loop_count :12
stat_count :2056
percentage :0.501953125

Man sieht also deutlich dass die Erfolgsrate sich signifikant reduziert. Für Eve sind ausser der hier vorgestellten Lauschstrategie auch andere Vorgehensweisen denkbar indem sie zum Beispiel weniger oft misst oder eine beliebig verdrehte Basis nutzt. Es zeigt sich jedoch dass es keine Lauschstrategie gibt mit der Eve die volle Information über den Schlüsselcode erhält und gleichzeitig unbemerkt bleibt.

Neben dem BB84 Protokoll wird auch das E91 Protokoll zum Schlüsselaustausch eingesetzt. Es basiert auf der Verwendung verschränkter qBit Paare, die wir zum Beispiel als Bell States kennengelernt haben. Wie wir gesehen haben gibt es Bell States die bezüglich beliebiger Basen

perfekt korriliert $|\phi^+>$ beziehungsweise antikorreliert $|\psi^->$ sind. Misst nun Alice ihr qBit bezüglich einer zufällig gewählten Basis so wird Bob's qBit bei perfekter Korrelation in den selben Zustand übergehen den Alice gemessen hat. Tauschen sie sich bezüglich der gewählten Basis aus so können sie im Falle gleichgewählter Basen das gemessene Bit für den Schlüssel verwenden. Werden sie allerdings abgehört so werden sie an der Auswertung der Statistik der Messwerte sehen dass keine perfekte Korrelation zwischen den Messergebnissen vorliegt.

Klassische Kommunikation

Um Nachrichten auf klassische Weise vor unerlaubtem Zugriff zu schützen wird vor allem die sogenannte RSA Kryptographie eingesetzt. Die Idee dieses Verfahrens beruht auf dem Umstand dass es sehr aufwendig ist eine große Zahl in ihre Primfaktoren zu zerlegen. Zum Verständnis der RSA Kryptographie benötigen wir einige Grundbegriffe aus der Zahlentheorie.

Grundlagen

Die Menge der ganzen Zahlen wird mit $\mathbb{Z}$ bezeichnet. Für $a, b \in \mathbb{Z}$ gilt: a ist ein Teiler von b wenn es eine Zahl $k \in \mathbb{Z}$ gibt für die gilt:

$$b = k \cdot a$$

Man sagt dann a teilt b. Jede Zahl $b \in \mathbb{Z}$ wird von 1 und sich selbst geteilt. Gilt $a \neq b$ so nennt man a einen echten Teiler von b. Eine Zahl $p \in \mathbb{Z}$ heisst Primzahl wenn 1 und p ihre einzigen Teiler sind. Für beliebige ganze Zahlen $a, b \in \mathbb{Z}$ gilt: Ist p ein Teiler von $a \cdot b$ dann ist p ein Teiler von a oder ein Teiler von b.

Der größte gemeinsame Teiler ggT zweier Zahlen $m, n \in \mathbb{Z}$ ist die grösste ganze Zahl a die sowohl m als auch n teilt:

$$a = ggT(m, n)$$

Jeder gemeinsame Teiler von m und n teilt auch a. Der ggT lässt sich mit dem Euklidschen Algorithmus bestimmen. Für $m > n$ ist die Laufzeit des Algorithmus $\mathcal{O}(log(m)^3)$. Sind m, n teilerfremd so ist 1 ihr größter gemeinsamer Teiler. Haben m und n keinen weiteren gemeinsamen Teiler außer der 1 nennt man sie teilerfremd.

Teilt man eine natürliche Zahl m durch eine natürliche Zahl n so erhält man einen Rest r. Die Modulo Funktion beschreibt dies:

$$m \; \boldsymbol{mod} \; n = r \qquad \text{mit} \qquad 0 \leq r \leq (n-1)$$

Für die Modulofunktion gelten die folgenden Rechenregeln:

$$(a + b) \; mod \; m = \big((a \; mod \; m) + (b \; mod \; m)\big) mod \; m$$

$$(a - b) \; mod \; m = \big((a \; mod \; m) - (b \; mod \; m)\big) mod \; m$$

$$(a \cdot b) \; mod \; m = \big((a \; mod \; m) \cdot (b \; mod \; m)\big) mod \; m$$

$$(a^k) \; mod \; m = \big((a \; mod \; m)^k\big) mod \; m$$

Man kann die Menge der ganzen Zahlen $\mathbb{Z}$ mittels der Modulofunktion in sogenannte Restklassen einteilen. Alle ganzen Zahlen $n \in \mathbb{Z}$ die bezüglich eines Teilers a den selben Rest b erzeugen sind dann in der Klasse $[b]_a$:

$$[b]_a = \{n \in \mathbb{Z} \mid \exists k \in \mathbb{Z}: \; n = (k \cdot a) + b\} = \{n \mid n \; mod \; a = \; b \; mod \; a\}$$

Beispiele:

$[0]_2 = \{0,2,4,6,8 \dots \dots \dots \dots\}$

$[1]_3 = \{1,4,7,10,13 \dots \dots \dots \dots\}$

$[3]_5 = \{3,8,13,18,23 \dots \dots .\}$

Auch auf den Restklassen eines gleichen Teilers lassen sich eine Multiplikation und eine Addition einführen. Diese werden durch Tabellen dargestellt. So gilt zum Beispiel für den Teiler 4:

$$[1]_4 + [2]_4 = [3]_4$$

Wird die Summe grösser als der Teiler ergibt sich:

$$[2]_4 + [3]_4 = [5]_4 = [1]_4$$

Dies gilt auch für die Multiplikation:

$$[2]_4 \cdot [3]_4 = [6]_4 = [2]_4$$

In Tabellenform ergibt sich dann für die Restklassen zum Teiler 4:

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

·	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Weiterhin lässt sich mittels der Modulo Funktion das multiplikative Inverse einer ganzen Zahl definieren. Da für jedes $a \in \mathbb{Z}$ gilt $a \cdot 1 = a$ bezeichnet man 1 als das neutrale Element der Multiplikation. Als inverses Element bezeichnet man eine Zahl b für die gilt $a \cdot b = 1$. Für ganze Zahlen existiert kein solches Element, da der Kehrwert einer ganzen Zahl keine ganze Zahl ist. Anstatt dessen wird aus mathematisch strukturellen Gründen das multiplikative Inverse modulo einer Zahl m definiert. Es gilt dann für eine ganze Zahl a: b ist das multiplikative Inverse modulo m wenn:

$$(a \cdot b)\ mod\ m = 1$$

Dabei müssen a und m natürlich teilerfremd sein denn sonst gilt für jede Zahl $b \in \mathbb{Z}$ immer $(a \cdot b)\ mod\ m = 0$. Mit diesen Grundlagen können wir uns nun der mit klassischen Methoden realisierbaren RSA Verschlüsselung zuwenden. Grundlage für die Ermittlung des multiplikativen Inversen modulo m ist die Bestimmung des $ggT(a, m)$ der wie oben erwähnt mittels des Euklidschen Algorithmus leicht auf einem Computer ausgewertet werden kann.

RSA Verschlüsselung

Wir betrachten wieder das Scenario dass Alice eine Nachricht an Bob senden möchte die für Eve nicht zu entschlüsseln ist. Die RSA Methode, die dies ermöglicht funktioniert folgendermaßen.

Bob wählt zwei unterschiedlich Primzahlen p und q und bildet daraus:

$$n = p \cdot q$$

und

$$m = (p-1) \cdot (q-1)$$

Dann wählt er eine Zahl a die zu m teilerfremd ist. Dies bedeutet a und m haben keinen gemeinsamen Teiler ausser der 1. Bob, der die Nachricht empfangen und später entschlüsseln soll gibt n und a als public key bekannt.

Die Nachricht von Alice an Bob ist eine Zahl x , die kleiner als n ist. Alice verschlüsselt nun ihre Zahl x nach folgendem Schema:

$$y = x^a \bmod n$$

Diese Zahl y sendet Alice an Bob. Bob berechnet b als das zu a multiplikative Inverse modulo m und kann mit b die Nachricht y mit der folgenden Formel entschlüsseln:

$$x = y^b \bmod n$$

Man kann mittels des Satzes von Euler-Fermat beweisen dass die oben angegebenen Identitäten allgemein gelten, dass also:

$$x = y^b \bmod n = (x^a \bmod n)^b \bmod n$$

wenn a und b entsprechend den angegebenen Regeln gebildet wird.

Wir rechnen ein einfaches Beispiel:

Schritt	Alice	Bob
1		$p = 5$ $q = 11$ $n = 55$ $m = 40$ $a = 7$
2	$x = 8$ $y = x^a \bmod n = 2$	
3		$(a \cdot b) \bmod m = 1$ $(7 \cdot b) \bmod 40 = 1$ $(7 \cdot 23) = 161$ $161 \bmod 40 = 1$ $b = 23$ $x = y^b \bmod n$ $x = 2^{23} \bmod 55$ $x = 8$

Der public key besteht aus den Zahlen n und a und wird von Bob bekanntgegeben. Mittels dieser beiden Zahlen kann Alice das zu übermittelnde x verschlüsseln und bekommt y. Damit Bob y wieder entschlüsseln kann benötigt er a und m mit welchen er dann b berechnen kann. Dann kann er mittels n und b das erhaltene y entschlüsseln und er erhält x . Entscheidend ist also dass m unter Verschluss bleibt um zu verhindern dass y auch durch einen Angreifer entschlüsselt werden kann. Da die Rechenregel $m = (p-1) \cdot (q-1)$ bekannt und einfach zu berechnen ist wäre das Verfahren leicht anzugreifen wenn man p und q aus n bestimmen könnte. Denn n ist ja Teil des public key den Bob bekannt gibt. Für kleine Primzahlen erscheint es einfach aus n auf p und q zu schliessen. Es gibt jedoch kein bekanntes Verfahren mit dem man die Primfaktorenzerlegung auf klassische Weise

durchführen kann außer alle Kombinationen an Primzahlen durchzuprobieren. Für große Zahlen ist dieser Aufwand jedoch extrem groß und kann auch von den leistungsstärksten Computern nur innerhalb einer sehr langen Rechenzeit realisiert werden.

Periodizität

Wir haben gesehen dass der entscheidende Punkt des RSA Verfahrens ist dass sich die Primfaktoren p und q der Zahl n für grosse Zahlen nicht einfach bestimmen lassen. Prinzipiell erscheint diese Aufgabe machbar. Man kann sich vorstellen alle Kombinationen von p und q die durch Multiplikation die Zahl n ergeben zu identifizieren. Dann nachzuprüfen für welche der Kombinationen p eine Primzahl ist und dann zu überprüfen ob auch q eine Primzahl ist. Sei $p = 7$ und $q = 13$ und somit $n = 81$:

p	q		n
3	27		81
7	13		81
9	9		81
27	3		81

Man kann die Reihe an dieser Stelle abbrechen da es ausreicht alle Kombinationen zu identifizieren für die $p < q$ gilt. In diesem Fall sind nur die markierten Zahlen beide Primzahlen und damit die Lösung der Faktorisierung. Mittels dieser einfachen Verfahrensweise wird der rechnerische Aufwand für große n jedoch massiv anwachsen so dass es nicht möglich ist in endlicher Zeit eine Lösung zu finden. Um erfolgreich zu sein müssen also andere Verfahren gefunden werden, die vor allem dazu in der Lage sind große Zahlen effizient zu handhaben.

Ein erster Schritt eine Zahl n zu faktorisieren wäre einen echten Teiler von n zu finden so dass:

$$n = t \cdot m$$

Bei dieser Aufgabe spielt die Periodizität einer Funktion eine interessante Rolle. Eine Funktion hat die Periodizität p wenn gilt:

$$f(x) = f(x + p) \qquad \forall\, x \in \mathbb{D}_f$$

Bekannte Funktionen sind zum Beispiel die Sinus und die Cosinusfunktion für deren Periodizität gilt $p = 2\pi$. Interessanterweise lassen sich mittels der Modulo Funktion periodische Funktionen erzeugen. So ergibt zum Beispiel die Funktion:

$$f(x) = a^x \bmod n$$

Für $a = 3$ und $n = 13$ die folgenden Funktionswerte:

x: 1 f(x): 3

x: 2 f(x): 9

x: 3 f(x): 1

x: 4 f(x): 3

x: 5 f(x): 9

x: 6 f(x): 1

x: 7 f(x): 3

x: 8 f(x): 9

x: 9 f(x): 1

Man sieht dass $p = 3$ weil zum Beispiel $f(1) = f(1+3) = f(1+6)$ gilt. Es ist im allgemeinen nicht einfach die Periode einer Funktion zu finden. Für die oben angegebene Funktion $f(x) = a^x \ mod \ n$ lässt sich die Periode jedoch berechnen.

Wir definieren die Ordnung einer Funktion f folgendermassen:

Die Funktion f sei gegeben durch $f(x) = a^x \ mod \ n$ mit $a \in \mathbb{N}$ und $ggT(a,n) = 1$. Das heisst a und n sind teilerfremd. Das kleinste $r \in \mathbb{N}$ für welches gilt $a^r \ mod \ n = 1$ heißt Ordnung von a modulo n . Die Funktion f ist dann r periodisch.

Das bedeutet für a^r :

$$\exists \, k \in \mathbb{N}: \quad a^r = kn + 1$$

$$(a^r - 1) = kn$$

n ist also ein Teiler von $(a^r - 1)$ und für gerade r gilt:

$$(a^r - 1) = \left(a^{r/2} + 1\right)\left(a^{r/2} - 1\right)$$

somit:

$$\left(a^{r/2} + 1\right)\left(a^{r/2} - 1\right) = kn$$

Dies ist eine Gleichung der Form:

$$kn = gh$$

damit ist:

$$g = \frac{k}{h} \cdot n \qquad \text{beziehungsweise} \qquad h = \frac{k}{g} \cdot n$$

und es gilt:

$$ggT(n,g) = ggT(n, \frac{k}{h} \cdot n\,) \neq 1 \qquad \text{beziehungsweise} \qquad ggT(n,h) = ggT(n, \frac{k}{h} \cdot n\,) \neq 1$$

Da entweder g oder h nicht teilerfremd mit n sind muss einer der beiden gemeinsame Primfaktoren mit n besitzen. Also müssen die Primfaktoren von n in den beiden Termen $\left(a^{r/2} + 1\right)$ und $\left(a^{r/2} - 1\right)$ enthalten sein. Wären alle Primfaktoren in $\left(a^{r/2} - 1\right)$ enthalten würde sich folgender Widerspruch ergeben. Dann wäre $\left(a^{r/2} - 1\right) mod \ n = 0$ und somit $a^{r/2} \ mod \ n = 1$. Da aber r die kleinste Zahl ist für die diese Bedingung gilt entsteht ein Widerspruch. Falls nicht alle Primfaktoren in $\left(a^{r/2} + 1\right)$ enthalten sind verteilen sie sich auf beide Terme. Durch die Berechnung des $ggT\left(\left(a^{r/2} - 1\right), n\right)$ erhält man einen echten Teiler von n.

Wir betrachten das folgende Beispiel: Sei $p = 7$ und $q = 13$ und somit $n = 91$. Wir wählen $a = 4$ und finden durch Ausprobieren Periode $r = 6$:

x: 1	f(x): 4
x: 2	f(x): 16
x: 3	f(x): 64
x: 4	f(x): 74
x: 5	f(x): 23
x: 6	f(x): 1
x: 7	f(x): 4
x: 8	f(x): 16
x: 9	f(x): 64
x: 10	f(x): 74
x: 11	f(x): 23
x: 12	f(x): 1
x: 13	f(x): 4

Wir bilden:

$$ggT\left(\left(a^{r/2}-1\right),n\right) = ggT\left((4^3-1),91\right) = ggT(63{,}91) = 7$$

$$ggT\left(\left(a^{r/2}+1\right),n\right) = ggT\left((4^3+1),91\right) = ggT(65{,}91) = 13$$

Das heißt wenn wir die Periode r für ein zufällig gewähltes a bezüglich des gegebenen n bestimmen können lässt sich n faktorisieren. Unser Verfahren erfordert dass r gerade ist. Es lässt sich im Rahmen der Zahlentheorie zeigen:

Für n ungerade und keine Primpotenz gilt für mindestens die Hälfte aller Zahlen $a \epsilon \{0, \dots, n-1\}$ mit $ggT(a,n) = 1$, also a und n teilerfremd:

- Periode r der Funktion $f(x) = a^x \, mod \, n$ ist gerade
- n ist kein Teiler von $\left(a^{r/2}+1\right)$

Man hat also ein Verfahren das unter bestimmten Bedingungen erfolgreich ist falls man die Periodizität der Funktion $f(x) = a^x \, mod \, n$ bestimmen kann. Sei die Ermittlung der Periode gegeben dann gilt für dieses Verfahren dass im Kern Shor's Algorithmus ausdrückt $O((log \ n)^3)$.

FFT

Fast Fourier Transformation ist ein bekanntes klassisches Verfahren um Periodizitäten in Datenreihen zu prozessieren. Sie basiert auf der diskreten Fouriertransformation. Man kann eine Folge von N Zahlen gegeben durch $a = (a_0, \dots \dots, a_{N-1})$ transformieren in eine Folge von N Koeffizienten die die harmonischen Anteile der durch a gegebenen Folge darstellen. Die transformierte Folge ist dann $\overline{a} = (\overline{a_0}, \dots \dots, \overline{a_{N-1}})$ mit den Koeffizienten:

$$\overline{a_k} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j \cdot e^{i2\pi \cdot jk/} \qquad\qquad k = 0, \dots \dots, N-1$$

dies lässt sich auch in Vektorschreibweise darstellen:

$$\bar{a} = W \, a$$

Die sogenannten Einheitswurzeln sind die komplexen Lösungen der Gleichung $x^n = 1$ für $x \in \mathbb{C}$. Sie sind definiert durch:
$\omega \in \mathbb{C}$ heisst N-te Einheitswurzel wenn $\omega^N = 1$ und primitive N-te Einheitswurzel wenn $\omega^N = 1$ aber $\omega^k \neq 1$ für alle $k = 1, \dots\dots, N-1$

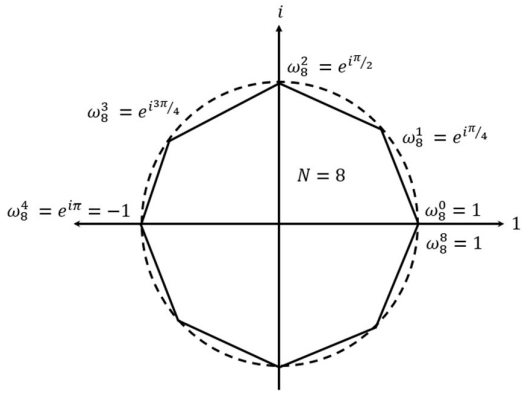
Für ω_N gilt:

$$\omega_N = e^{i\cdot\frac{2\pi}{N}}$$

Es ergeben sich genau N komplexe Einheitswurzeln:

$$\omega_N^k = \left(e^{i2\pi/N}\right)^k = e^{ik2\pi/N} \quad \text{für} \quad k = 0, \dots\dots, N-1$$

Mann kann sich die Werte der Einheitswurzeln vorstellen als die Eckpunkte eines N Eckes welches in den komplexen Einheitskreis einbeschrieben ist:



Sie haben folgende Eigenschaften:
Ihre Potenzen sind wiederum Einheitswurzeln. Durch Quadrieren ergibt sich speziell:

$$(\omega_N{}^0)^2, \dots., (\omega_N{}^{N-1})^2 = \omega_{N/2}{}^0, \dots\dots\dots, \omega_{N/2}{}^{N/2-1}$$

Das heißt die Anzahl der Einheitswurzeln halbiert sich und bestimmte Einheitswurzeln liegen dann paarweise aufeinander. Ist N gerade so gilt für jede primitive Einheitswurzel:

$$\omega_N^{N/2} = -1$$

Die Fourier Matrix W für N Elemente lässt sich mittels der Definition der Einheitswurzeln dann schreiben als:

$$W_{kj} = \omega_N{}^{kj} = e^{i2\pi\cdot kj/}$$

W ist eine symetrische $N \times N$ Matrix .

Die Idee des Verfahrens der schnellen Fouriertransformation ist, die einzelnen Berechnungen der Matrix-Vektor-Multiplikation $\bar{a} = W\ a$ in einer speziellen Reihenfolge auszuführen, sodass jeweils auf schon berechnete Zwischenergebnisse zurückgegriffen werden kann. Dabei werden spezielle Eigenschaften der Einheitswurzeln ausgenutzt. Zum einen dass $\omega_N^{N/2} = -1$ ist und dass das Quadrat der primitiven *N*-ten Einheitswurzel die primitive *n*/2-te Einheitswurzel ist. Das Verfahren setzt voraus, dass N eine Zweierpotenz ist. Bei der Berechnung wird zwischen den geraden und den ungeraden Koeffizienten von $\bar{a}$ unterschieden.

Für gerade Indizes gilt:

$$k \in \{0, \dots, (N/2 - 1)\} \qquad \bar{a}_k^g = \overline{a_{2k}} = \sum_{j=0}^{N-1} a_j \cdot {\omega_N}^{kj}$$

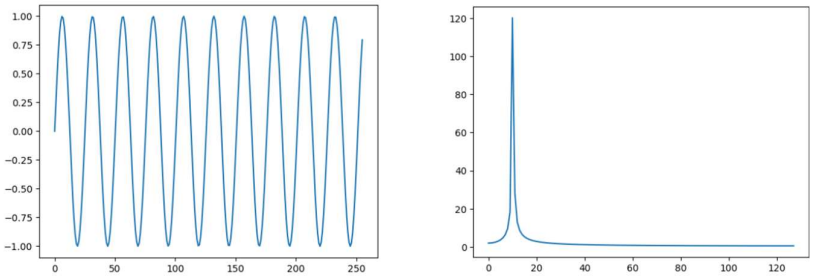
Sei $m = N/2$ und $v = {\omega_N}^2$ wobei v die primitive m te Einheitswurzel ist, dann gilt:

$$\bar{a}_k^g = \sum_{j=0}^{m-1} \left(a_j + a_{j+m}\right) \cdot v^{kj}$$

Für ungerade Indizes gilt mit der selben Terminologie:

$$\bar{a}_k^u = \sum_{j=0}^{m-1} {\omega_N}^j \left(a_j - a_{j+m}\right) \cdot v^{kj}$$

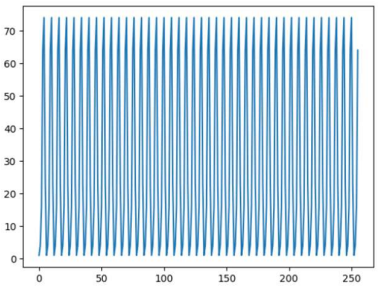
Folgendes Beispiel zeigt wie eine Zahlenreihe durch die FFT transformiert wird:

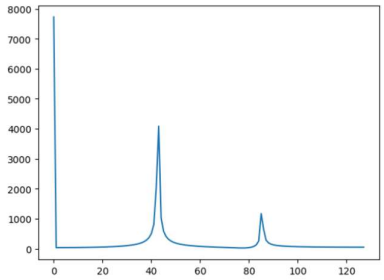


Dargestellt ist die Sinusfunktion $f = \sin(i/4)$ für $i = 0, \dots, 255$ und ihre FFT. Der deutliche Peak ergibt sich für $i = 10$. Wir wissen dass die sinus Funktion 2π periodisch ist. Da unser Argument $i/4$ ist gilt also dass sich f nach $2 \cdot \pi \cdot 4 \approx 25$ Punkten wiederholt. Das heisst bezogen auf unsere 256 Datenpunkte besitz sie die Frequenz 10 :

$$f(i) = \sin(i/4) \approx f = \sin((i/4) + 25) = f(i + 25)$$

Damit ist also die Periodizität $p = Anzahl\ Punkte \div FFT\ Frequenz$. Wir können nun eine Funktion der Form $f(x) = a^x\ mod\ n$ mittels der FFT betrachten. Wählen wir dasselbe Beispiel wie oben so ergibt sich für $n = 91$ und $a = 4$ wiederum für unsere 256 Datenpunkte $x = 0, \dots, 255$:





Die beiden Peaks liegen bei $i = 43$ und $i = 85$. Der zweite Peak entspricht der ersten Oberschwingung. Unsere Funktion f wiederholt sich in diesem Fall also nach etwa $256 \div 43 = 5{,}9$ Datenpunkten. Was sehr nahe bei unserer beobachteten Periodizität von 6 liegt. Erhöht man die Zahl der Datenpunkte so erhöht sich auch die Genauigkeit. Man erhält zum Beispiel für 1024 Datenpunkte bereits einen Wert von $1024 \div 171 = 5{,}98$. Allerdings steigt der Rechenaufwand mit der Zahl der Datenpunkte immens.

QFT

Die Quanten Fourier Transformation für einen Zustand der Ordnung N ist mit der Definition der Potenzen k derEinheitswurzeln zu N: $\omega_N^k = e^{i2k\pi/N}$ gegeben durch die unitäre $N \times N$ Matrix:

$$QFT_N = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots & \cdots & 1 \\ 1 & \omega_N^1 & \omega_N^2 & \omega_N^3 & \cdots & \omega_N^{(N-1)} \\ 1 & \omega_N^2 & \omega_N^4 & \omega_N^6 & \cdots & \omega_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \omega_N^{(N-1)} & \omega_N^{2(N-1)} & \omega_N^{3(N-1)} & \cdots & \omega_N^{(N-1)(N-1)} \end{pmatrix}$$

Ist $|\mathrm{j}>$ ein Basisvektor der Normalbasis $|0> \cdots |\mathrm{N}-1>$ so gilt:

$$QFT_N\ |\mathrm{j}> = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} \cdot\ |\mathrm{k}>$$

Damit wirkt die QFT als eine Basistransformation , mit:

$$|0> \cdots |\mathrm{N}-1> \quad \xrightarrow{QFT_N} \quad QFT_N\ |0> \cdots QFT_N\ |\mathrm{N}-1>$$

Betrachtet man einen beliebigen Zustandsvektor:

$$|\mathrm{v}> = \sum_{j=0}^{N-1} \alpha_j \cdot |\mathrm{j}>$$

So gilt:

$$QFT_N\ |\mathrm{v}> = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \left(\sum_{j=0}^{N-1} \alpha_j \cdot \omega_N^{jk} \right) \cdot\ |\mathrm{k}>$$

Es stellt sich nun also die Frage wie sich die QFT durch die zur Verfügung stehenden Quantengatter realisieren lässt. Um dies zu erreichen werden Basis Vektoren der Dimension N-1 als ganze Zahlen interpretiert. Das bedeutet:

$$|q_1 \ldots . . q_N > = q_1 \cdot 2^{N-1} + \cdots + q_N \cdot 2^0$$

$$|00 \ldots . .00 > = 0$$

$$|00 \ldots . .01 > = 1$$

$$|00 \ldots . .10 > = 2$$

$$|00 \ldots . .11 > = 3$$

$$|00 \ldots .100 > = 4$$

$$|00. \ldots 101 > = 5$$

Es lässt sich zeigen dass für einen beliebigen Basiszustand $|\mathrm{x}>$ mit der oben gezeigten Zuordnung bei n qBits, $N = 2^n$ gilt:

$$x = 2^{N-1} x_{N-1} \cdots + 4x_2 + 2x_1 + x_0$$

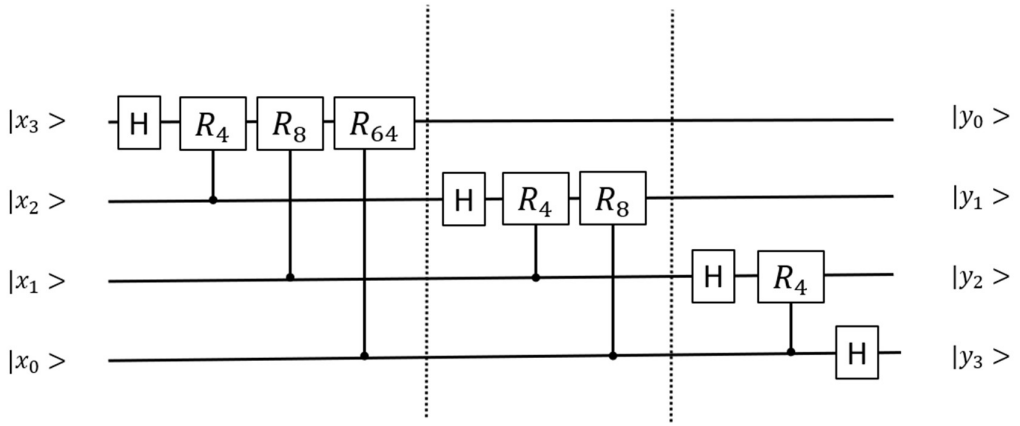
$$QFT_N \, |\mathrm{x}> = {}^{1}\!/_{\sqrt{N}} \, (|0> + \cdot \, \omega_2^x \, |1>) \cdot \, (|0> + \cdot \, \omega_4^x \, |1>) \cdot \cdots (|0> + \cdot \, \omega_N^x \, |1>)$$

Um ein Quantenregister wie beschrieben zu transformieren benutzt man die folgende unitäre Transformation R_m :

$$R_m = \begin{pmatrix} 1 & 0 \\ 0 & \omega_m \end{pmatrix} \qquad\qquad \omega_m = \, e^{i \cdot \frac{2\pi}{m}}$$

Sie bewirkt eine Drehung des Phasenwinkels um $2\pi/m$ falls sich das qBit im Zustand $|1>$ befindet. Für ein Quantenregister mit n qBits, $N = 2^n$ lässt sich QFT_N dann durch ein Hadamard Gate und eine Sequenz an R_m Gattern die auf die einzelnen qBits angewendet werden realisieren. Ist $|x_i>$ das i te qBit werden i verschiedene R_m Gatter in aufsteigender Reihenfolge angewendet. Diese R_m Gatter werden von den qBits $|x_0> \ldots\ldots |x_{i-1}>$ kontrolliert und es ergibt sich für $j = 1, \ldots . . , i-1$ das entsprechende m zu $m = \, 2^{i-j+1}$

Für $N = 16$ ist $n = 4$:

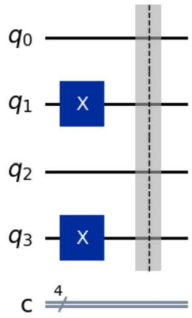


Für $N = 2^n$ lässt sich QFT_N mit $\frac{n(n+1)}{2} = \mathcal{O}(n^2)$ Gattern ausführen. Die klassische FFT skaliert mit $N(\log N)$, während die QFT mit $(log\ N)^2$ skaliert. Shor's Algorithmus nutzt diesen exponentiellen Vorteil aus.

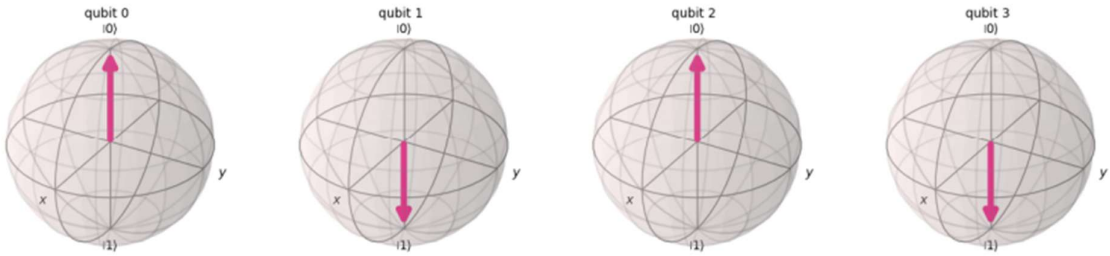
Zur Veranschaulichung kann man sich die QFT einfach als die Darstellung einer Zahl in einem anderen Zahlenschema vorstellen. In der binären Basis bestimmen die Werte der einzelnen qBits eines Registers einfach den Beitrag der einzelnen 2er Potenzen zu einer Zahl:

$$|0101> = 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5$$

Transformiert man das Register wird die selbe Zahl in einer anderen Basis dargestellt. Für die Normalbasis lässt sich $k = 5$ durch den folgenden Schaltkreis in die qBits 0…3 encodieren:



Wir erhalten dann in der Blochdarstellung folgende Kombination:



qBit3 als least significant bit für 2^0 und qBit1 für 2^2 sind jeweils 1.

Wir haben bereits erwähnt dass der |+> und der |−> Zustand in der Blochkugel auf der Äquatorlinie der xy Ebene liegen. Man kann diese Zustände durch Rotationen um die z Achse modifizieren. Haben wir wie zum Beispiel $N = 4$ qBits so ist lässt sich ein Kreis teilen in $2^4 = 16$ Teilsegmente mit dem Winkel $2\pi/16$. Die Zahl $k = 5$ aus obigem Beispiel kann dann durch folgende Einzelrotationen encodiert werden:

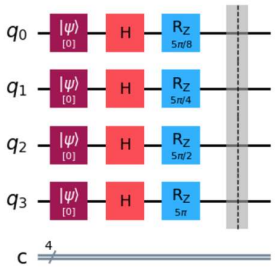
qBit 0: $\frac{5}{16} \cdot 2\pi = \frac{1 \cdot k}{16} \cdot 2\pi$

qBit 1: $\frac{10}{16} \cdot 2\pi = \frac{2 \cdot k}{16} \cdot 2\pi$

qBit 2: $\frac{20}{16} \cdot 2\pi = \frac{4 \cdot k}{16} \cdot 2\pi$

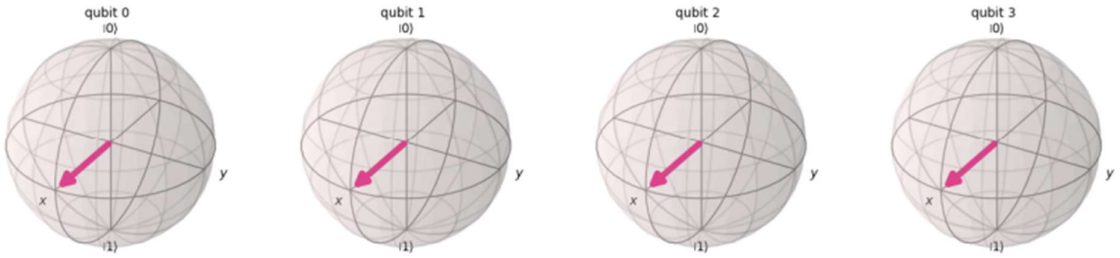
qBit 3: $\frac{40}{16} \cdot 2\pi = \frac{8 \cdot k}{16} \cdot 2\pi$

Man sieht also dass das qBit 3 am schnellsten rotiert und bei jeder nächsten Zahl um π hin und herflipt, während qBit 0 sich erst nach jeder 16-ten Zahl einmal um die z Achse gedreht hat. Der folgende Quantenschaltkreis demonstriert diese Betrachtungsweise in diesem Fall für $k = 5$:



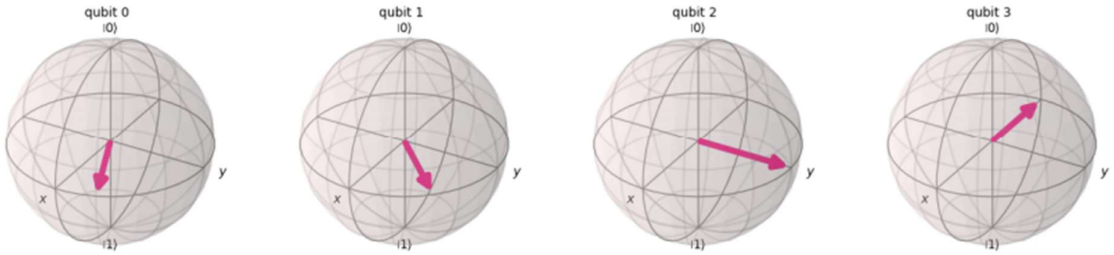
Im folgenden sind die Blochkugeln für ansteigende Zahlen k dargestellt:

$k = 0$:



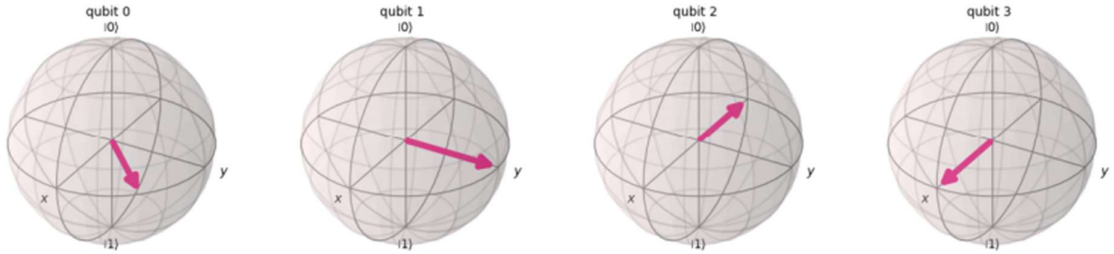
Für $k = 0$ bleiben alle qBits in ihrer Ausgangsstellung

$k = 1$:



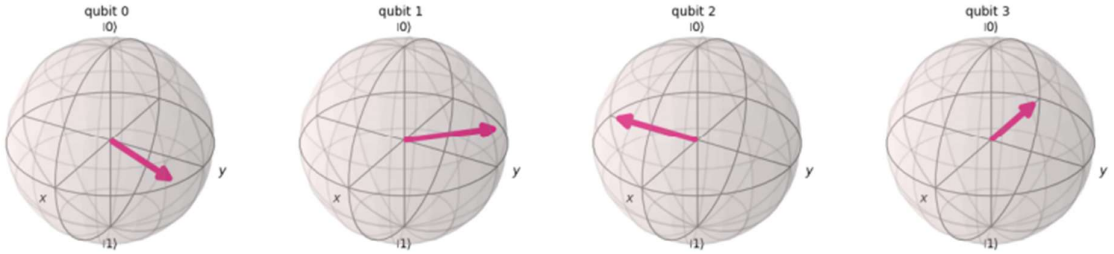
Für $k = 1$ hat sich qBit3 um π gedreht, qBit2 um $\pi/2$, qBit1 um $\pi/4$ und qBit0 um $\pi/8$

$k = 2$:

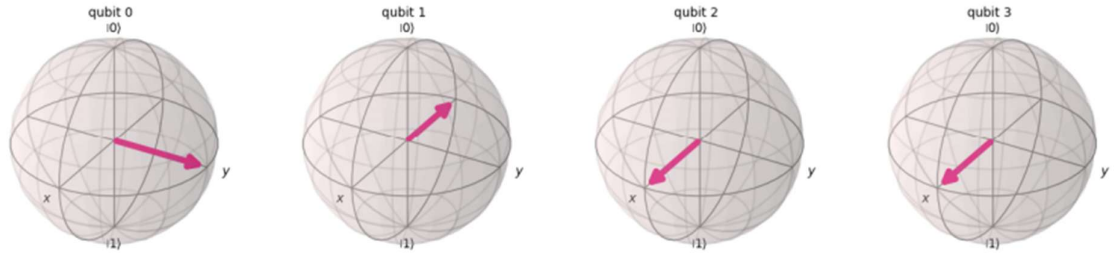


Man sieht an der fortlaufenden Entwicklung wie die unterschiedlichen Drehgeschwindigkeiten sind

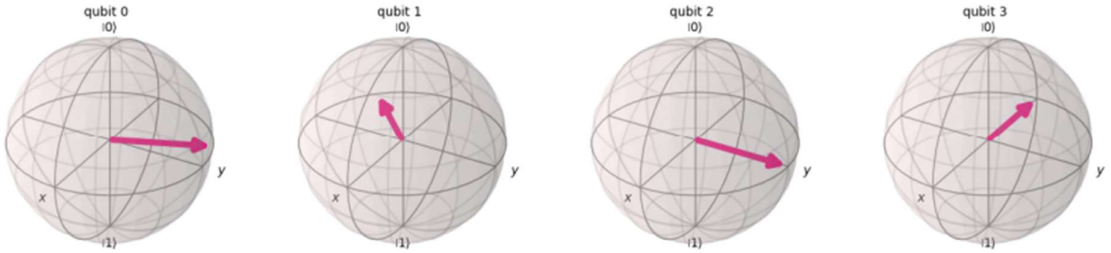
$k = 3$:



$k = 4$:



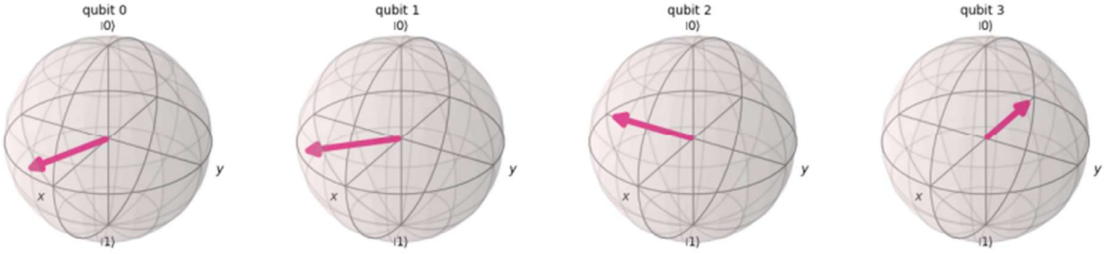
$k = 5$:



Dies ist der Fall für den obigen Schaltkreis.

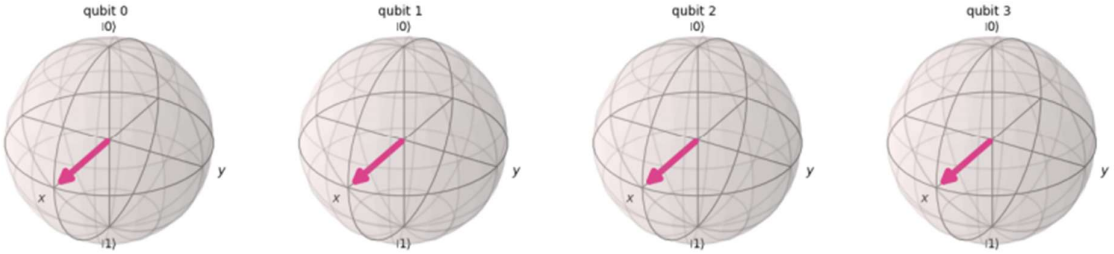
⋮

$k = 15$:



Man sieht dass alle qBits in ihrer Rotation kurz davor sind den Vollkreis abzuschliessen

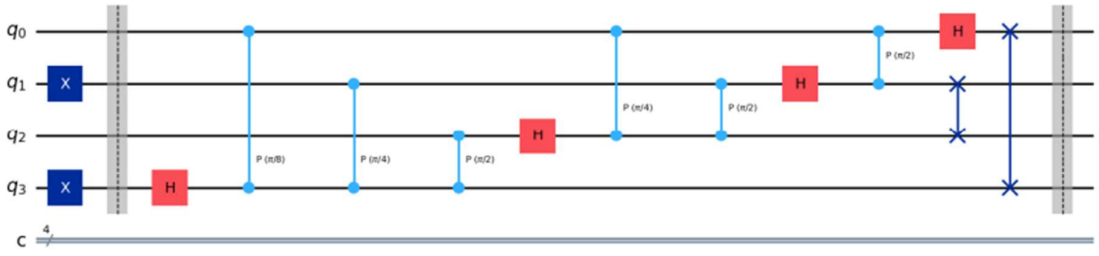
$k = 16$:



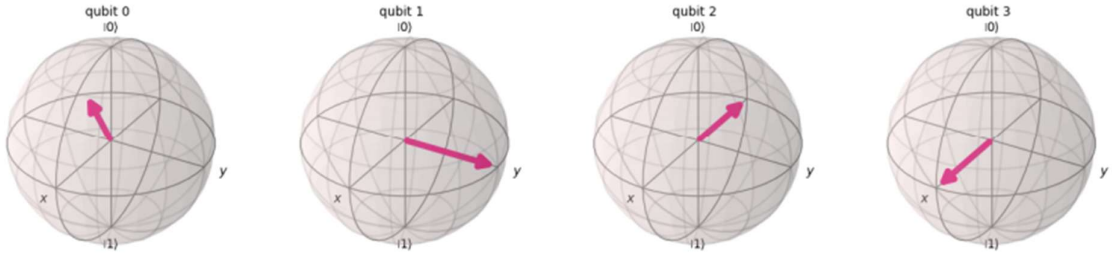
Um dann schließlich im Ausgangszustand anzukommen. Dabei ist qBit0 einmal um die z Achse rotiert, qBit1 2-mal, qBit2 4-mal und qBit3 8-mal.

Ist also der Drehwinkel des qBit0 für eine bestimmte Zahl n ein Vielfaches von 2π so gilt $n\ modulo\ 16 = 0$ und dies gilt auch für die anderen qBits. Ist also qBit1 für eine bestimmte Zahl n ein Vielfaches von 2π so gilt $n\ modulo\ 8 = 0$. Das heisst am Rotationswinkel der einzelnen qBits lässt sich ablesen ob in einer Zahl k ein ganzzahliges Vielfaches einer bestimmten 2er Potenz enthalten ist. Wir haben an diesem Beispiel gesehen wie sich Zahlen mittels entsprechender Rotationen in qBits encodieren lassen und haben ein wenig spekuliert welche Information sich anhand der Encodierung ableiten lässt.

Ganz ähnlich lässt sich auch die QFT betrachten. Wendet man die QFT in einem qiskit Program an und encodiert mittels X Gates eine Zahl, in diesem Fall wieder $k = 5$ (Big Endian) , in die qBits so ergibt sich folgender Schaltkreis:

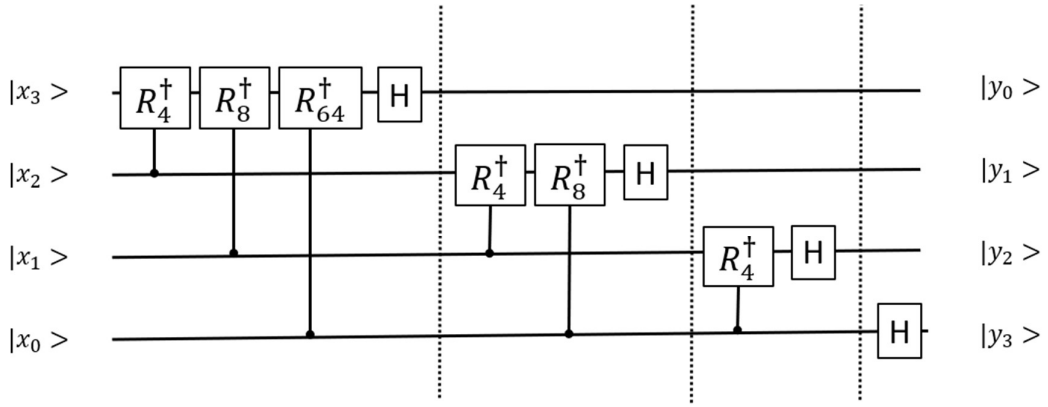


Aufgrund der Big Endian/Little Endian Konvention und der Anwendung von Swap Operatoren ist er bezüglich des oben angegebenen Schemas gespiegelt. Er stellt jedoch die selbe Funktionsweise dar. qBit3 ist also das least significant Bit und steht für 2^0und qBit0 das most significant bit und steht für 2^3. In der Bloch Darstellung erhalten wir nach der QFT:

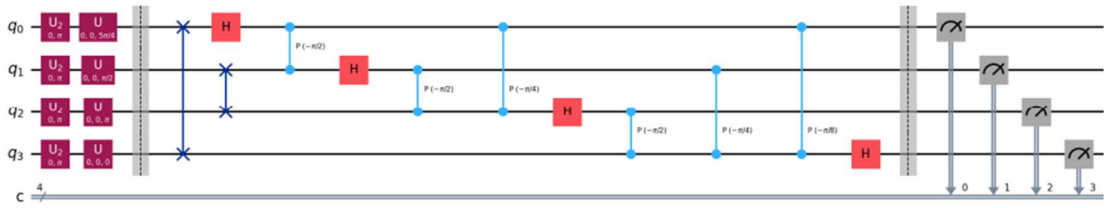


Das heißt die Information der Zahl k ist in die einzelnen qBits bezüglich der Normalbasis encodiert wie wir es im obigen Beispiel gezeigt haben und der QFT Schaltkreis erzeugt daraus die dazugehörige Fouriertransformierte deren Information in die Rotationswinkel der einzelnen qBits encodiert ist.

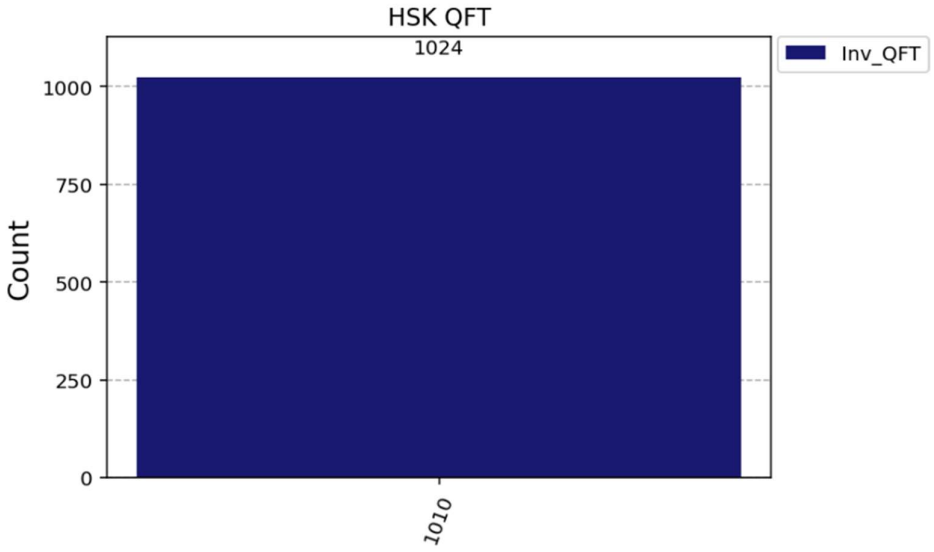
Diverse Anwendungen sind neben der QFT auf die Invertierung der QFT, die so genannte iQFT angewiesen. Da es sich bei R und H um unitäre Operatoren handelt und der Hadamard Operator zudem selbstadjungiert ist ergibt sich für die IQFT der folgende Quantenschaltkreis:



Encodiert man die durch die QFT erhaltenen Phasenwinkel in quiskit entsprechend in die einzelnen qBits so erhält man für $k = 5$ den skizzierten Quantenschaltkreis:



Hier gilt die selbe Anmerkung wie im obigen Beispiel bzgl. der Big Endian/Little Endian Konvention und der Anwendung von Swap Operatoren. Die Messungen zeigen dann das zu erwartende Ergebnis:



QPE

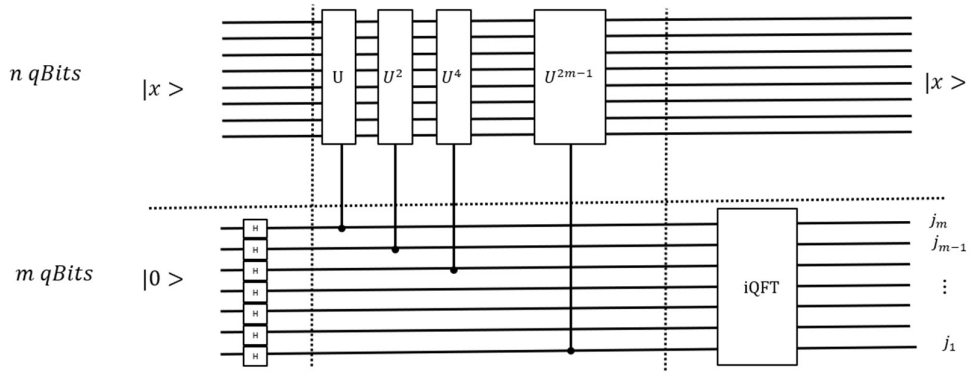
Die Quantum Phase Estimation kann nicht als abgeschlossener Quanten Algorithmus betrachtet werden. Sie stellt ein Modul dar, welches in Kombination mit anderen Modulen relevante Rechenoperationen realisieren kann. Die QPE dient zur Abschätzung des Phasenwinkels der folgenden Eigenwertgleichung.

Sei U ein unitärer Operator und $|x>$ ein Eigenvektor mit dem Eigenwert λ :

$$U\,|x> = \lambda\ |x> \qquad \text{mit} \qquad \lambda = e^{2\pi i\,\varphi} \qquad 0 \leq \varphi < 1$$

Das Ziel der QPE ist es nun φ abzuschätzen. Um dieses Ziel zu erreichen benötigt die QPE zwei Register. Dabei enthält das erste der beiden Register m qBits die alle mit 0 initialisiert werden. Die Anzahl der qBits beeinflusst sowohl die Genauigkeit der Abschätzung als auch die Wahrscheinlichkeit

dass die Abschätzung überhaupt richtig ist. Das zweite Register stellt $|x>$ dar und enthält n qBits. Die QPE wird in 2 Phasen in einem Schaltkreis mit $m+n$ qBits durchgeführt:



Da φ eine Dezimalzahl ist kann man sie binär annähern durch den Ansatz:

$$\varphi = \sum_{k=1}^{m} \frac{\varphi_k}{2^k} \qquad \text{mit} \quad \varphi_k \in \{0,1\}$$

Die φ_k sind also binär und ihre Anzahl entspricht der Anzahl der verwendeten qBits des obigen Quantumcircuits.

Aus der Eigenwertgleichung wissen wir:

$$U^l |x> = \lambda^l |x> = e^{2\pi i\, l\varphi} |x>$$

Man kann nun den Ausgangszustand, der gegeben ist durch:

$$|+ + + \cdots +> |x>$$

schrittweise berechnen zu:

$$\frac{1}{\sqrt{2^m}}(|0> + e^{2\pi i\, 0.\varphi_1 \ldots \varphi_m} |1>) \otimes (|0> + e^{2\pi i\, 0.\varphi_2 \ldots \varphi_m} |1>) \ldots \otimes (|0> + e^{2\pi i\, 0.\varphi_m} |1>) \otimes |x>$$

Dieser Ausdruck entspricht im ersten Teil der QFT von $|\varphi_1 \ldots \varphi_m>$. Wendet man also wie oben die iQFT an so ergibt sich:

$$|\varphi_1 \ldots \varphi_m > |x>$$

Und aus den gemessenen $\varphi_1 \ldots . \varphi_m$, die sich als Binärkombination ergeben lässt sich dann φ berechnen zu:

$$\varphi = \frac{\varphi_1}{2} + \frac{\varphi_2}{4} + \cdots + \frac{\varphi_m}{2^m} \qquad \text{mit dem Eigenwert} \qquad \lambda = e^{2\pi i \varphi}$$

Man sieht am obigen Ausdruck wie m die Genauigkeit der Abschätzung beeinflusst.

Diese verfahren lässt sich auch anwenden wenn U mehrere Eigenvektoren besitzt. Präpariert man den Vektor $|x>$ als Superposition dieser Vektoren so erhält man die Eigenwertergebnisse entsprechend der Wahrscheinlichkeitsverteilung der Superposition.

Shor Algorithmus

Wie wir im Abschnitt über die Periodizität einer Funktion gesehen haben lässt sich eine aus dem Produkt zweier Primzahlen gebildete Zahl n faktorisieren wenn wir die Periode r der Funktion

$$f(x) = a^x \, mod \, n$$

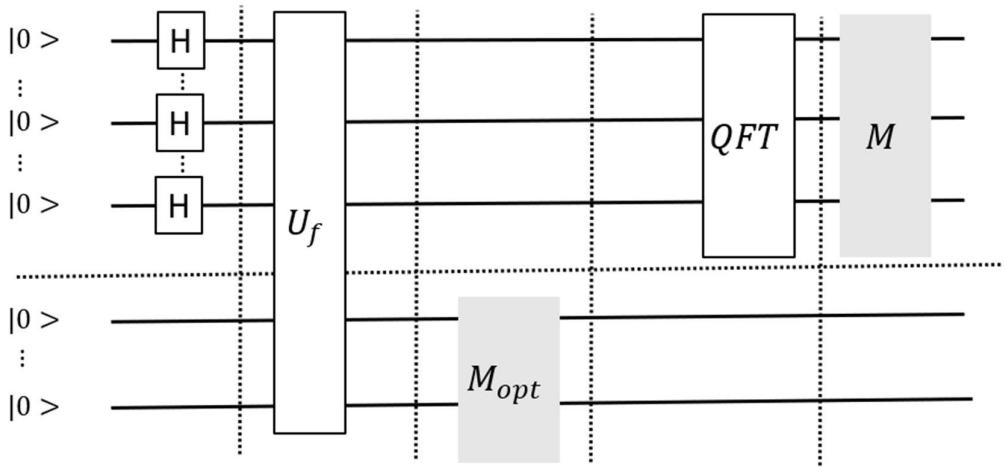
für ein zufällig gewähltes a mit $ggT(a,n) = 1$ bestimmen können. Zur Bestimmung von r haben wir in den obigen Abschnitten die QFT_N diskutiert, die von der klassischen FFT abgeleitet wurde.

Schaltkreis

Gegeben ist also n und a mit $a < n$ und $ggT(a,n) = 1$. Ziel ist es die Periode r der Funktion f zu bestimmen. Um dies zu bewerkstelligen wählt man ein n_x und ein n_y so dass:

$$N = 2^{n_x} \approx n^2 \quad \text{und} \quad 2^{n_y} > n$$

Man beginnt mit einem Zustand $|\sigma_0>$ der aus $n_x + n_y$ qBits besteht. Die zu n_x gehörenden qBits bilden das x Register und die zu n_y gehörenden qBits bilden das y Register. Beide Register werden zu 0 initialisiert. Der folgende Schaltkreis ermöglicht die Bestimmung der Periode r:



Analyse

Zunächst stellen die Hadamard Transformationen im x Register eine gleichmässige Superposition über alle 2^{n_x} Zustände her und der Gesamtzustand geht über in $|\sigma_1>$. In früheren Algorithmen haben wir bereits die Funktion U_f kennengelernt, die hier die entsprechende Quantenversion der Funktion $f(x) = a^x \, mod \, n$ darstellt:

$$U_f: |\mathrm{x}> \otimes \ |\mathrm{y}> \rightarrow |\mathrm{x}> \ \otimes \ |\mathrm{y} \oplus \mathrm{f}(x)>$$

Die Bildung der Funktion f ist theoretisch gewährleistet da jeder klassische Boole'sche Term durch entsprechende Quanten Gatter dargestellt werden kann. Der Gesamtzustand geht über in $|\sigma_2>$ und aufgrund der 0 Initialisierung des y Registers gilt:

$$|\mathrm{x}> \otimes |\mathrm{y} \oplus \mathrm{f}(x)> = |\mathrm{x}> \otimes |\mathrm{f}(x)> \quad \text{für} \quad x = 0, \dots, N-1$$

Man kann nun optional das y Register messen. Man erhält dabei einen Funktionswert $f(x_0)$ für ein bestimmtes x_0 . Da f periodisch ist kollabiert dann der Gesamtzustand so dass das x Register zu einer gleichmässigen Überlagerung all derjenigen Zustände wird für die gilt $f(x) = f(x_0)$. Betrachtet man $|\sigma_3>$ als solch einen kollabierten Zustand und wendet die QFT auf das x Register an so entsteht der Zustand $|\sigma_4>$ bei dem lediglich die Amplituden der Basiszustände die nahe bei einem Vielfachen von N/r liegen signifikant von 0 verschieden sind. Dies gilt auch wenn man die Messung des y Registers nicht durchführt. Es gilt dann nach Messung des x Registers:

$$x \approx k \cdot \frac{N}{r}$$

Man kann nun durch klassische Nachbearbeitung mittels der sogenannten Kettenbruch Entwicklung die Periode r mit grosser Wahrscheinlichkeit herausfinden.

HHL Algorithmus

Der HHL Algorithmus wird in erster Linie unter dem Aspekt der Lösung eines linearen Gleichungssystems betrachtet:

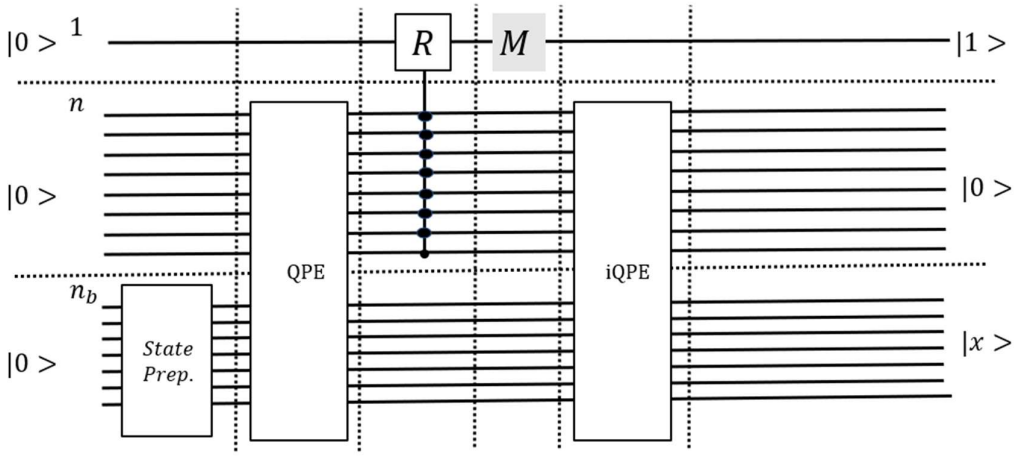
$$Ax = b \qquad \text{wobei} \qquad x, b \in \mathbb{C}^N \text{ und } A \in \mathbb{C}^{N \times N}$$

A ist eine hermitesche Matrix, was bedeutet dass : $A^\dagger = A$. Wir erinnern uns an die Definition der unitären Matrix: $U^\dagger = (U^*)^T = U^{-1}$. Des weiteren wird angenommen dass $N = 2^{n_b}$ also eine Zweierpotenz ist. Wir werden n_b qBits betrachten aus denen wir die N unbekannten Zahlen die wir für die Lösung suchen erzeugen. Da A und b bekannt sind ergibt sich die Lösung zu:

$$x = A^{-1}b$$

Basis

Schematisch sieht der HHL Algorithmus wie folgt aus:



Der HHL Algorithmus besteht aus 5 unterteilten Phasen:

- State Preperation
- Quantum Phase Estimation
- Ancilla Bit Rotation
- Messung des Ancilla Bit
- inverse Quantum Phase Estimation

Wir benutzen die sogenannte little endian Notation. Das heißt als Binärzahl beziffert das bit ganz rechts das sogenannte least significant bit (LSB) also die Zahl 2^0. Wie man oben sieht gibt es mehrere Register an qBits. Das LSB ist in der obigen schematischen Darstellung jeweils das oberste qBit der beiden Register mit n_b qBits beziehungsweise n qBits. Die Register haben unterschiedliche Funktionen. Zunächst gibt es das Register b , das aus n_b qBits besteht. Seine qBits stellen den Vektor b beziehungsweise x dar. Das Register c besteht aus n qBits und wird clock Register genannt. Wie wir später sehen werden benötigt man die clock qBits bei der Quantum Phase Estimation und die Genauigkeit der Phase Estimation ist abhängig von ihrer gewählten Anzahl. Außerdem gibt es ein einzelnes qBit das man Ancilla Bit nennt welches seinem Namen entsprechend als Hilfsbit betrachtet werden kann.

Sind $|u_i>$ die Eigenvektoren von A und λ_i die dazugehörigen Eigenwerte dann kann man A darstellen als:

$$A = \sum_{i=0}^{2^{n_b}-1} \lambda_i \cdot |u_i><u_i|$$

Drückt man A in der Basis seiner Eigenvektoren aus ist A diagonal und damit lässt sich A^{-1} leicht bilden:

$$A^{-1} = \sum_{i=0}^{2^{n_b}-1} {\lambda_i}^{-1} \cdot |u_i><u_i|$$

Ebenso lässt sich b in der Basis der Eigenvektoren von A ausdrücken:

$$|b>= \sum_{j=0}^{2^{n_b}-1} b_j \cdot |u_j>$$

Damit gilt wegen $<u_i|u_j>= \delta_{ij}$:

$$|x>= A^{-1}|b>= \sum_{i=0}^{2^{n_b}-1} {\lambda_i}^{-1} \cdot b_i \cdot |u_i>$$

Dies ist der prinzipielle mathematische Lösungsweg um x zu finden. Schauen wir uns nun die einzelnen Phasen des Algorithmus etwas genauer an.

State Preperation

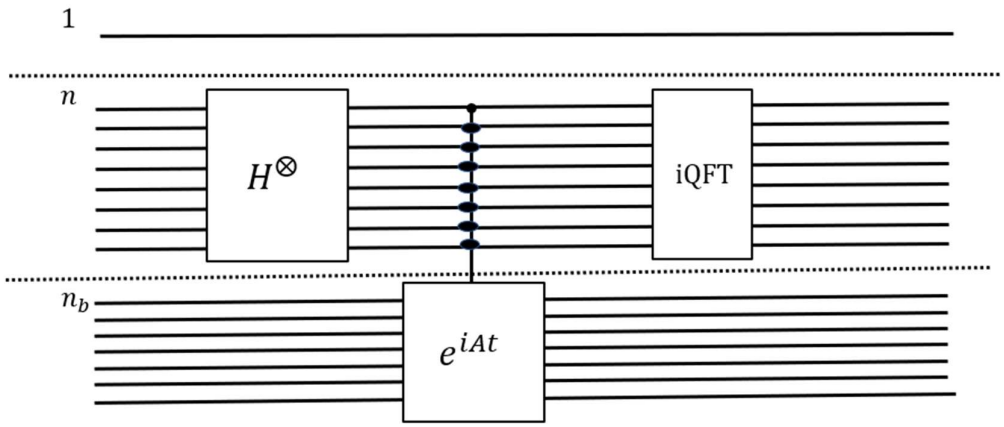
Grundsätzlich werden alle qBits zunächst 0 initialisiert. Nun gilt es den Vektor b in das Register b zu encodieren. Dabei werden die einzelnen qBits entsprechend der Amplituden des Vektors b in ihrer Phase rotiert. Das heisst die Phasenauflösung ist entscheidend für die Genauigkeit der Encodierung von b. Nach der State Preperation gilt also:

$$|b> \otimes |c> \otimes |Anc> = |b_{enc}> \otimes |0> \otimes |0>$$

Wobei b_{enc} bedeutet dass der Vektor b in das Register b encodiert wurde.

Quantum Phase Estimation

Die Quantum Phase Estimation QPE kann als ein Algorithmus zur Abschätzung von Eigenwerten betrachtet werden. Im vorherigen Kapitel haben wir die Quantum Fourier Transformation diskutiert. Ihre Invertierung, also die umgekehrte Transformation wird iQFT genannt. Sie lässt sich durch Quantengatter ähnlich implementieren wie die QFT und spielt eine entscheidende Rolle bei der QPE. Schematisch sieht die QPE folgendermaßen aus:



Nach der QPE gilt:

$$|b> \otimes |c> \otimes |A> = |b_{enc}> \otimes |A_{enc}> \otimes |0>$$

Rotation and Measurement Ancilla Bit

Wir haben im vorigen Schritt die Eigenwerte der Matrix A in das Register c enkodiert. Die qBits dieses Registers kontrollieren nun die Rotation des Ancilla Bits, das entscheidet ob die Messung verwertbar ist. Nach kontrollierter Rotation und Messung gilt:

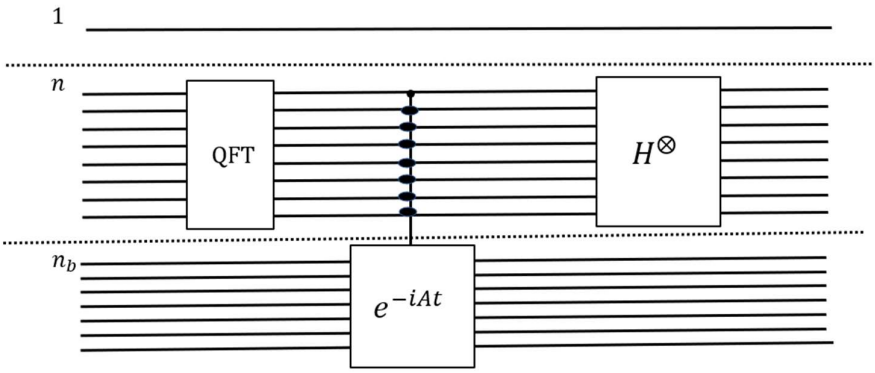
$$|b> \otimes |c> \otimes |A> = |b_{enc}> \otimes |A_{enc}> \otimes (\alpha|0> + \beta|1>)$$

Die Koeffizienten α und β entscheiden ob die Messung des Ancilla Bit 0 oder 1 ergibt. Ergibt die Messung 0 wird das Ergebnis ignoriert. Ist das Ergebnis 1 werden die Register b und c weiter prozessiert. Die kontrollierte Rotation beurteilt im Prinzip ob die Eigenwerte ausreichend aufgelöst werden konnten. Sind ihre Abstände zu klein reicht die Zahl der qBits im Register c nicht aus um sie ausreichend zu separieren.

Inverse Quantum Phase Estimation

Hat die oben beschriebene Messung 1 ergeben muss das gesuchte Ergebnis x aus dem Register b gewonnen werden. Da die qBits des Register b allerdings mit dem Register c entangled sind lassen sie sich nicht in ein Register b und ein Register c faktorisieren. Und damit lässt sich Register b nicht einfach in der Normalbasis darstellen. Man muss das erhaltene Ergebnis also nachprozessieren.

Hierbei wird die iQPE angewendet die aus ganz ähnlichen Elementen besteht wie die QPE und sich der QFT bedient:



Nach iQPE gilt:

$$|b> \otimes |c> \otimes |A> = |x> \otimes |0>^{\otimes n} \otimes |1>$$

Damit lässt sich die Lösung x aus dem Register b auslesen.

Fehlerkorrektur

Bisher haben wir angenommen dass physikalisch existierende qBits den bislang diskutierten Formalismus exakt umsetzen. Es existieren allerdings bislang keine Implementierungen von qBits die dies erfüllen können. Ein Quantenobjekt müsste dazu von seiner Umwelt komplett abgetrennt sein. Das bedeutet der Zustand jedes individuellen qBits unterliegt eines gewissen Fehlers der sich in einer Berechnung auswirkt und sich entsprechend fortpflanzt.

Fehlerfortpflanzung

Kann sich ein kleiner Fehler im Laufe einer Berechnung zu einem größeren Fehler fortpflanzen. Nehmen wir an der fehlerfreie Zustand wäre $|\mathrm{x}>$ der tatsächliche Zustand jedoch $|x'>$ so ergibt sich die Fehlergrösse zu:

$$\varepsilon = \| \ |x> \ - |x'> \ \|$$

Da unitäre Transformationen längenerhaltend sind gilt:

$$\varepsilon = \| \ |x> \ - |x'> \ \| = \| \ U|x> \ - \mathrm{U}|x'> \ \| = \varepsilon$$

Das bedeutet dass U einen bestehenden Fehler nicht verstärkt. Es stellt sich die Frage wie sich eine fehlerbehaftete unitäre Transformation auswirkt. Seien U_1 und U_2 zwei unitäre Transformationen und U_1' und U_2' die zu ihnen gehörenden fehlerbehafteten unitären Transformationen. Dann gilt für die hintereinander ausgeführten Transformationen mit:

$$\varepsilon_i = \left\| \ U_i|x> - \mathrm{U}_i'|x> \ \right\|$$

$$\|U_2' \otimes U_1'|x> - U_2 \otimes U_1|x>\|$$

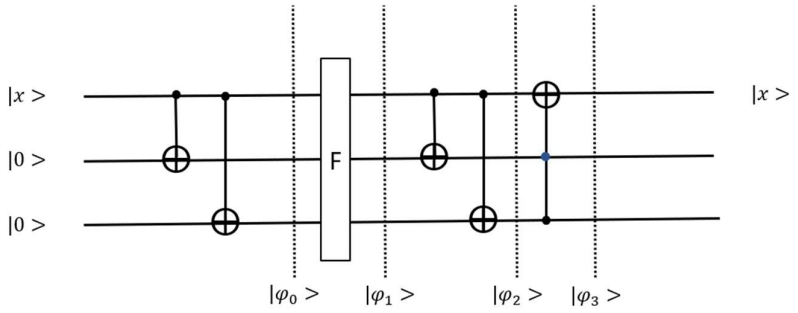
$$= \|(U_1' - U_1)|x>\| + \|(U_2' - U_2)(U_1|x>)\|$$

$$\leqq \varepsilon_1 + \varepsilon_2$$

Das heißt die Einzelfehler addieren sich auf und somit ist der Gesamtfehler beschränkt durch die Summe der Einzelfehler der unitären Transformationen.

Bitflip

Da qBits nicht einfach kopiert werden können lassen sich klassische Fehlerkorrekturverfahren nicht direkt umsetzen. Es gibt jedoch alternative Verfahren die angewendet werden können. Ein Bitflip liegt vor wenn ein qBit fälschlicherweise aus dem Zustand $\alpha|0\rangle + \beta|1\rangle$ in den Zustand $\alpha|1\rangle + \beta|0\rangle$ übergeht. Dieser Übergang entspricht einer σ_x Transformation. Folgender Schaltkreis erkennt und korrigiert einen potentiellen Bitflip auf $|x>$:



Die Fehlerfunktion F ist definiert als eine Transformation die maximal einen oder aber keinen Bitflip auf einem der drei qBits bewirkt. Man nennt den ersten Teil bei dem $|x>$ durch 2 CNOT Gates auf zwei weitere Hilfsbits ausgedehnt wird die Codierung. Dabei wird der Zustand von $|x>$ durch Verschränkung auf 3 qBits übertragen die dann als sogenanntes logisches qBit bezeichnet wird.

Sei $|x> = \alpha|0\rangle + \beta|1\rangle$ dann kann man für den obigen Schaltkreis verschiedene Fälle unterscheiden die in den einzelnen Gesamtzuständen $|\varphi_i>$ resultieren:

Bitflip	$\|\varphi_0>$	$\|\varphi_1>$	$\|\varphi_2>$	$\|\varphi_3>$
none	$\alpha\|000\rangle + \beta\|111\rangle$	$\alpha\|000\rangle + \beta\|111\rangle$	$\alpha\|000\rangle + \beta\|100\rangle$	$(\alpha\|0\rangle + \beta\|1\rangle)\otimes\|00>$
q_0	$\alpha\|000\rangle + \beta\|111\rangle$	$\alpha\|100\rangle + \beta\|011\rangle$	$\alpha\|111\rangle + \beta\|011\rangle$	$(\alpha\|0\rangle + \beta\|1\rangle)\otimes\|11>$
q_1	$\alpha\|000\rangle + \beta\|111\rangle$	$\alpha\|010\rangle + \beta\|101\rangle$	$\alpha\|010\rangle + \beta\|110\rangle$	$(\alpha\|0\rangle + \beta\|1\rangle)\otimes\|10>$
q_2	$\alpha\|000\rangle + \beta\|111\rangle$	$\alpha\|001\rangle + \beta\|110\rangle$	$\alpha\|001\rangle + \beta\|101\rangle$	$(\alpha\|0\rangle + \beta\|1\rangle)\otimes\|01>$

Man sieht dass in der letzten Spalte unabhängig von den Hilfsbits gilt $|x> = \alpha|0\rangle + \beta|1\rangle$ und es liegt kein Entanglement zwischen den Hilfsbits und $|x>$ vor. Man kann aus der Kombination der qBits 1 und 2 außerdem ablesen welches der qBit geflipped ist.

So wie sich ein Bitflip σ_x korrigieren lässt gibt es auch Korrekturverfahren für einen Phasenflip der einer σ_z Transformation entspricht. Wir werden im nächsten Abschnitt sehen das Bit Flips und Phase Flips als identische Flips in unterschiedlichen Basen gesehen werden können. Mittels solcher Korrekturverfahren lassen sich Schaltkreise entsprechend gegen Fehlereinflüsse schützen. Man sieht allerdings dass dies jeweils eine bestimmte Anzahl an Hilfsbits und entsprechende Quantengatter erfordert.

Shor 9-qBit Code

Wir haben gesehen dass ein qBit sowohl durch einen Bitflip als auch durch einen Phasenflip beeinflusst werden kann. Es ergeben sich also 4 Fälle die den Zustand $|x>= \alpha|0\rangle + \beta|1\rangle$ modifizieren können:

Fehlereinfluss	Zustand	Operation
none	$\alpha\|0\rangle + \beta\|1\rangle$	I
Bitflip	$\alpha\|1\rangle + \beta\|0\rangle$	σ_x
Phaseflip	$\alpha\|0\rangle - \beta\|1\rangle$	σ_z
both	$\alpha\|1\rangle - \beta\|0\rangle$	$\sigma_x \cdot \sigma_z$

Ein Verfahren welches alle 4 Fälle erkennen und entsprechend korrigieren könnte würde also ein gesichertes qBit liefern. Zur Erörterung eines solchen Verfahrens brauchen wir zunächst einige Hilfsmittel. Wir haben bereits gesehen dass wir den Einfluss auf ein qBit durch Matrizen beschreiben können:

Kein Fehler:

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

mit:

$$I\,|0>=|0> \qquad I\,|1>=|1> \qquad I\,|+>=|+> \qquad I\,|->=|->$$

Bit Flip:

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

mit:

$$X\,|0>=|1> \qquad X\,|1>=|0> \qquad X\,|+>=|+> \qquad X\,|->=-|->$$

Phase Flip:

$$\sigma_z = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

mit:

$$Z\,|0>=|0> \qquad Z\,|1>=-|1> \qquad Z\,|+>=|-> \qquad Z\,|->=|+>$$

Beides:

$$\sigma_x\sigma_z = XZ = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

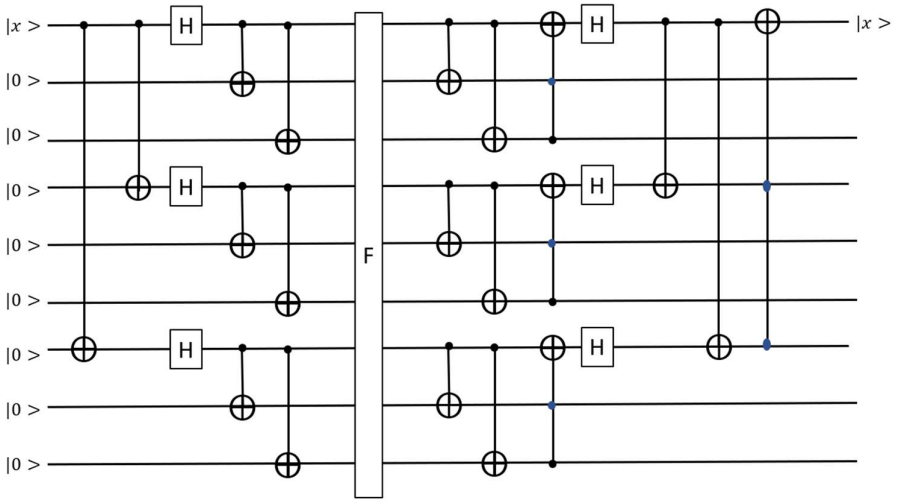
mit:

$Y\,|0> = i|1>$ $\quad$ $Y\,|1> = -i\,|0>$ $\quad$ $Y\,|+> = -i\,|->$ $\quad$ $Z\,|-> = i\,|+>$

X, Y, Z sind die sogenannten Pauli Matrizen, die wir bereits zu Beginn kennengelernt haben. Man sieht dass X und Z dahingehend ähnlich sind dass X die bits in der Normalbasis flipped, während Z den flip in der Hadamard Basis vollzieht. Dies bedeutet:

$$Z = HXH \quad \text{und} \quad X = HZH$$

Fehler die man durch die Pauli Matrizen beschreibt sind komplett. Das heißt man kann zeigen dass damit alle denkbaren singulären Fehler abgedeckt sind. Um nun alle Fehler zu korrigieren benutzt man die bereits oben erwähnte Codierung. Und zwar in verschiedenen Basen. Für die Korrektur der Bit Flips in der Normalbasis und für die Korrektur der Phase Flips in der Hadamard Basis. Es wird dann 1 qBit in 9 qBits encodiert:



Im ersten Schritt werden die Phase Flips adressiert. Man beginnt mit dem Ausgangszustand:

$$|\varphi> = \alpha|000\rangle|000\rangle|000\rangle + \beta|111\rangle|111\rangle|111\rangle$$

Die CNOT Gatter und die H Gatter ergeben:

$$|\varphi> = \alpha|+00\rangle|+00\rangle|+00\rangle + \beta|-00\rangle|-00\rangle|-00\rangle$$

Die qBits 1,4 und 7 sind verschränkt. Sie bilden diese Codierung des Phase Flip. Die restlichen qBits sind alle auf 0 gesetzt und sind unabhängig von 1,4 und 7. Sie werden nach dem oben bereits besprochenen Schema für die Codierung der Bit Flips verwendet. Betrachtet man einen der Dreierblöcke so ergibt sich für diesen nach Ausführung der zweiten Gruppe an CNOT Gates:

$$|+00\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |100\rangle) \rightarrow \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) =: |c^+\rangle$$

beziehungsweise

$$|-00\rangle = \frac{1}{\sqrt{2}}(|000\rangle - |100\rangle) \rightarrow \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle) =: |c^-\rangle$$

Der Gesamtzustand nach der Anwendung der zweiten CNOT Gruppe lässt sich mit dieser Notation schreiben als:

$$|\varphi> = \alpha|c^+c^+c^+\rangle + \beta|c^-c^-c^-\rangle$$

Dies ist die aus $|x>= \alpha|0\rangle + \beta|1\rangle$ entstandene Codierung. Mit dieser Codierung können nun entsprechend den obigen Betrachtungen die verschiedensten Fehlerkombinationen betrachtet werden und danach mit den entsprechenden Transformationen des obigen Schaltkreises wieder zurücktransformiert werden. Es wird sich für alle erlaubten Kombinationen zeigen dass das Ausgangsbit sich wieder im Zustand $|x>= \alpha|0\rangle + \beta|1\rangle$ befindet.

Hardware

In unseren bisherigen Betrachtungen haben sind wir von einer sehr klaren Vorstellung ausgegangen was ein qBit ist beziehungsweise wie es sich verhält. In der Praxis ist es jedoch sehr schwierig physikalische Implementierungen zu realisieren die diese Annahme bestätigen. Es gibt eine Vielzahl von unterschiedlichen Implementierungen von denen wir nur einige wenige kurz diskutieren werden.

Ion Traps

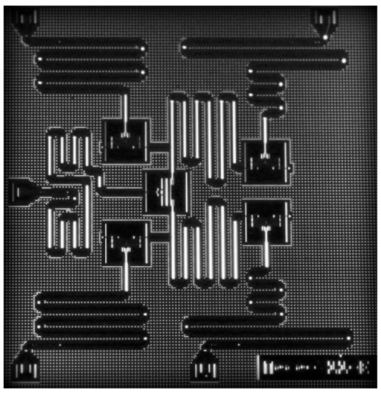
In sogenannten Ionenfallen ist es möglich durch eine geeignete Anordnung von elektrischen und magnetischen Feldern einzelne Moleküle in einem Vakuum von der Außenwelt zu separieren. Durch die Wechselwirkung mit Photonen, die durch einen Laser gezielt appliziert werden können dann unterschiedliche Energieniveaus des Moleküls als qBit Basiszustände manipuliert werden.

Quantum Dots

Durch die fortschreitende Miniaturisierung von Halbleiterelementen sind sehr hoch auflösende Lithografieverfahren möglich die es erlauben Quantenobjekte auf Halbleiteroberflächen zu designen. Man kann mit diesen Methodiken Halbleiterstrukturen schaffen bei denen einzelne Elektronen in unterscheidbare Spinzustände gebracht werden können die als Basiszustände eines qBits betrachtet werden können.

Superconducting Circuits

Hierbei handelt es sich um spezielle auf Halbleitern realisierte Schwingkreise die auf Josephson Kontakten basieren. Sie ermöglichen bei extrem niedrigen Temperaturen in Supraleitung elektromagnetische Schwingungen auf unterschiedlichen Energieniveaus und können durch elektromagnetische Felder und Mikrowellenresonatoren entsprechend manipuliert werden. Die Abbildung zeigt einen 5qBit Quanten Prozessor der Firma IBM:



Photonen

Photonen stellen a priori Quantenobjekte dar. Man kann ihre verschiedenen Eigenschaften, wie zum Beispiel ihre Polarisationsrichtung nutzen um zwei Zustände zu unterscheiden. Da sie sich in Luft oder in Glasfasern sehr unproblematisch und mit hoher Geschwindigkeit ausbreiten werden sie bevorzugt bei Experimenten zur Quantenkommunikation und Quantenkryptographie eingesetzt.

Neben den aufgelisteten Technologien wird weltweit an weiteren Alternativen geforscht. Die grosse Herausforderung besteht darin eine Methodik zu finden bei der die Abschottung von der Umwelt keine extremen Aufwände erfordert gleichzeitig aber garantiert dass die benötigten Quantenzustände möglichst langlebig und fehlerfrei existieren. Des weiteren müssen die durch die Technologie bereitgestellten Quantenzustände dazu in der Lage sein die zur Realisierung bestimmter Quantengatter erforderlichen Manipulationen zu ermöglichen. Die Details hierzu werden durch die sogenannte Dekohärenzzeit beziehungsweise durch die DiVincenzo Kriterien beschrieben.

Dekohärenzzeit

Wie wir bereits mehrfach erwähnt werden Quantenobjekte in ihrem Zustand sehr leicht durch ihre Umgebung gestört. Die stille Annahme bei unseren bisherigen Betrachtungen war stets dass die für die Realisierung eines Algorithmus angestrebten unitären Transformationen zu definierten Zustandsänderungen führen die dann als Ergebnis gedeutet werden können. Wird dieser Vorgang von außen beeinflusst stellt es die Qualität der erhaltenen Ergebnisse in Frage. Die Dekohärenz beschreibt diesen Vorgang, also die Tatsache dass Quantenobjekte nur innerhalb eines sehr kleinen Zeitfensters als ein ungestörter Zustand betrachtet werden kann. Bis also eine Wechselwirkung mit einem anderen Quantenobjekt aus der Umgebung stattgefunden hat. Um die Dekohärenzzeit zu verlängern müssen Experimente daher bei sehr tiefen Temperaturen und im Ultrahochvakuum stattfinden. Je länger die Dekohärenzzeit eines Quanten Computers ist umso grösser ist die Anzahl der anwendbaren Gatter beziehungsweise die Qualität eines verschränkten Zustandes.

DiVincenzo Kriterien

Die DiVincenzo Kriterien definieren Anforderungen an Implementierungen von qBits die notwendig sind um das Konzept des Quantum Computings überhaupt zu erfüllen. Sie lauten:

- Es existiert ein skalierbares System aus gut charakterisierten qBits
- Die qBits können in einen definierten Anfangszustand versetzt werden
- Elementare Quantengatter erlauben universelle Rechenoperationen

- Die qBits können gemessen werden
- Die Dekohärenzzeit ist wesentlich länger als die Operationszeit eines Gatters

Nicht alle Implementierungen erlauben es diese Kriterien umfassend zu erfüllen. Die folgende Abbildung zeigt die Entwicklung der Dekohärenzzeit für Quantenrechner früherer Jahre: