Quantencomputer – Simulation und experimentelle Umsetzungen von Algorithmen

Marcel Schmittfull, Jörg Metzner

Februar 2006

Inhaltsverzeichnis

1 Einleitung

Das Ziel der Arbeit ist mit Hilfe einer leistungsfähigen und effizienten Simulation von Quantencomputern eine Grundlage für die Weiterentwicklung von Quantenalgorithmen zu schaffen, und möglicherweise sogar zu einer Neuentwicklung beizutragen. Bereits gefundene Quantenalgorithmen sollen simuliert werden und anschaulich in einer grafischen Ausgabe dargestellt werden. Desweiteren sollen neue experimentelle Umsetzungen vorhandener und weiterentwickelter Algorithmen entwickelt und aufgebaut werden, die wegen ihres vergleichsweise einfachen Aufbaus zum Einsatz im Unterricht und zur Erklärung der Funktionsweise eines Quantencomputers in der Schule geeignet sind (siehe Abschnitt 4.2).[1] Um hierauf eingehen zu können, muss auf die allgemeine Funktionsweise eines Quantencomputers aufgebaut werden (siehe folgender Abschnitt). Nützliche mathematische und theoretische Grundlagen sind im Anhang aufgelistet.

Im Gebiet der experimentellen Umsetzung von Quantencomputern, als auch auf dem Gebiet der theoretischen Behandlung von Quantengattern und Qubits, wird momentan weltweit intensiv geforscht. Bereits Anfang der 1990er Jahre wurden wichtige theoretische Grundlagen für Quantencomputing geschaffen (z.B. Universellität von nur vier Quantengattern, Algorithmen mit Effizienzvorteil gegenüber klassischen Algorithmen). In den darauf folgenden Jahren bis heute wurden viele experimentelle Umsetzungen eines Quantencomputers diskutiert und aufgebaut. Für wenige Qubits (bis zu acht) konnten solche Aufbauten bereits einfache Quantenalgorithmen erfolgreich implementieren. Die Entwicklung zu Experimenten mit mehr Qubits erfolgt in verschiedenen Forschungszentren, z.B. in Innsbruck, Oxford oder am Caltech und anderen amerikanischen Universitäten (siehe z.B. `http://www.qubit.org`). Auch wenn die experimentelle Umsetzung eines Quantencomputers, der so viele Qubits besitzt, dass er bestimmte Probleme effizienter als ein klassischer Computer lösen kann, vermutlich noch einige Jahre in Anspruch nehmen wird, zeigt die enorm gestiegene Anzahl neuer Forschungsartikel in den vergangenen Jahren auf z.B. `http://xxx.lanl.gov` in der Rubrik *quant-ph* zum Thema Quantum Computing, dass viele Forscher Quantencomputern ein großes Potential zusprechen und die Entwicklung experimenteller Umsetzungen und Algorithmen sehr intensiv betrieben wird. So erscheinen im Moment nahezu monatlich neue wichtige Ergebnisse von Forschungsgruppen zur Theorie und Umsetzung eines Quantencomputers.

2 Funktionsweise und Simulation eines Quantencomputers

Im folgenden Kapitel wird beschrieben wie ein Quantencomputer auf einem klassichen Computer simuliert werden kann, um neue Algorithmen zu testen und zu entwickeln. Dabei wird auf die allgemeine Funktionsweise von Quantencomputern eingegangen.

2.1 Komponenten eines Quantencomputers

Ein Quantencomputer besteht aus drei grundlegenden Komponenten.

1. Qubits als kleinste Informationseinheit (ähnlich zu klassischen Bits).
2. Register als Ansammlung von Qubits und Speicher von Information.
3. Gatter als Möglichkeit, Operationen auf die Qubits und das Register ausführen zu können.

Auf diese Komponenten wird in den folgenden Abschnitten ausführlich eingegangen.

2.2 Ziele der Simulation

Um komplexe Algorithmen mit vielen Qubits entwickeln und untersuchen zu können, ist es notwendig, Quantencomputer einfach simulieren zu können. Die Aufgaben einer solchen Simulation sind dabei:

- Möglichst jeder Algorithmus, der auf einem Quantencomputer liefe, soll implementierbar sein.
- Die quantenmechanischen Gesetzmäßigkeiten sollen möglichst genau abgebildet werden.
- Die Ergebnisse und Vorgehensweise der Simulation sollen in Echtzeit möglichst anschaulich nachvollziebar sein, so dass Untersuchungen und Analysen von Algorithmen möglich sind.
- Die Performance sollte möglichst hoch sein, so dass auch größere Probleme in einem akzeptablen Zeitrahmen untersucht werden können.

[1] Dies ist ein eher langfristiges Ziel zur Erweiterung des Projekts, der bisherige Stand der Arbeit konzentriert sich hauptsächlich auf die ersten beiden genannten Ziele.

2.3 Qubits

Ein Bit eines klassischen Computers ist entweder gesetzt (1) oder nicht gesetzt (0). Somit stellt es die kleinste Informationseinheit eines klassischen Computers dar. Bei Quantencomputern wird die kleinste Informationseinheit *Qubit* genannt. Ein solches Qubit befindet sich im Gegensatz zum klassischen Bit jedoch nicht in entweder dem Zustand 0 oder dem Zustand 1, sondern in einer Superposition (Überlagerung) zweier Zustände. Formal lässt sich ein Qubit also wie folgt beschreiben.

Ein zweidimensionales Quantensystem habe die beiden Basiszustände

$$|0\rangle \equiv \begin{pmatrix}1\\0\end{pmatrix} \quad \text{und} \quad |1\rangle \equiv \begin{pmatrix}0\\1\end{pmatrix}.$$

Ein Qubit ist dann ein Quantenzustand

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix}, \tag{2.1}$$

wobei für die komplexen Amplituden $\alpha, \beta \in \mathbb{C}$ die Normierung $|\alpha|^2 + |\beta|^2 = 1$ gelten muss (vgl. Anhang). Die Betragsquadrate der Amplituden $|\alpha|^2$ bzw. $|\beta|^2$ sind die Wahrscheinlichkeiten, das System nach einer Messung im Basiszustand $|0\rangle$ bzw. $|1\rangle$ vorzufinden. $(\alpha, \beta)^T$ stellt die Koordinatendarstellung des Zustands bzgl. der beiden Basisvektoren dar.

Der Zustand eines Qubits kann also *jede* Linearkombination der beiden Basiszustände $|0\rangle$ und $|1\rangle$ sein. Ein solches Qubit in der Realität herzustellen ist auf verschiedene Arten möglich, z.B. durch Polarisation von Photonen, Spin-$\frac{1}{2}$-Systeme oder der Weg eines Teilchens beim Doppelspaltexperiment.

2.4 Register

2.4.1 Funktion des Speichers von Information

Das *Register* (auch: *Speicher*) eines Quantencomputers dient der Speicherung von Information und besteht aus einer Ansammlung mehrerer Qubits. Der Zustand $|\psi\rangle$ eines z.B. 2-Qubit-Registers ist deshalb eine Linearkombination der Basiszustände $|0_a 0_b\rangle, |1_a 0_b\rangle, |0_a 1_b\rangle$ und $|1_a 1_b\rangle$ der beiden Qubits *a* und *b*, also

$$|\psi\rangle = \alpha|0_a 0_b\rangle + \beta|1_a 0_b\rangle + \gamma|0_a 1_b\rangle + \delta|1_a 1_b\rangle,$$

wobei $|x_a y_b\rangle$ für den Basiszustand steht, in dem das Bit *a* im Zustand $|x\rangle$ und das Bit *b* im Zustand $|y\rangle$ ist. Die hier eingeführte Binärschreibweise der Basiszustände wird auf ein *n*-Qubit-Register verallgemeinert, d.h. die Basis des zu einem *n*-Qubit-Register gehörenden Hilbert-Raums wird mit

$$\mathcal{B} = \{|i\rangle \mid i \in \{0,1\}^n\}$$

bezeichnet. Daraus folgt, dass es genau 2^n Basiszustände gibt und der Zustand eines *n*-Qubit-Registers durch einen Vektor im 2^n-dimensionalen Hilbert-Raum $\mathcal{H}_{2^n}$ repräsentiert werden kann. Synonym zur Binärschreibweise der Basiszustände wird die Dezimalschreibweise verwendet, d.h.

$$|i_0 i_1 \dots i_{n-1}\rangle = \left|i_0 2^0 + i_1 2^1 + \dots + i_{n-1} 2^{n-1}\right\rangle.$$

Der Zustand eines *n*-Qubit-Register kann dann also allg. als

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i_0 i_1 \dots i_{n-1}\rangle = \sum_{i=0}^{2^n-1} \alpha_i \left|\sum_{j=0}^{n-1} i_j 2^j\right\rangle \quad \text{mit } i_j \in \{0,1\}, \alpha_i \in \mathbb{C} \text{ und } \sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$$

geschrieben werden.

Hervorzuheben ist nochmals, dass die Anzahl der Basiszustände exponentiell, nämlich wie 2^n, mit der Anzahl *n* der Qubits wächst. D.h. schon für $n = 250$ Qubits ist die Anzahl der Basiszustände eines Registers bereits $2^{250} \approx 0,18 \cdot 10^{78}$, was immerhin etwa der Größenordnung der geschätzten Anzahl der Atome im Universum entspricht.[1]

2.4.2 Simulation

Das Register wird in der Simulation durch ein Array mit 2^n komplexen Amplituden für die Basiszustände dargestelt. Da so bei einer Simulation von zu vielen Qubits Speicherprobleme auftreten, ist eine Optimierungsmöglichkeit für eine große Qubit-Anzahl das Array der Amplituden in eine verkettete Liste zu überführen, die nur Amplituden verschieden von 0 und die Basiszustandsbezeichnung speichert. Eine solche Lösungsmethode ist sehr sinnvoll, da bei Algorithmen mit vielen Qubits oft ein großer Teil der Qubits nur für Zwischen- und Endergebnisse benötigt wird, deshalb nicht am Anfang in einen Superpositionszustand gebracht werden muss und somit auch keine von 0 verschiedene Amplitude besitzt. Die verkettete Liste, die den Speicher simulieren soll, hat demnach den folgenden Aufbau:

[1] Siehe `http://www.harri-deutsch.de/verlag/hades/clp/kap09/cd236b.htm`

- Anzahl n der Qubits.
- Anzahl N der Basiszustände $|i\rangle$ mit $\alpha_i \neq 0$.
- 1. Basiszustand mit $\alpha \neq 0$: Amplitude α, Bezeichnung.
- 2. Basiszustand mit $\alpha \neq 0$: Amplitude α, Bezeichnung.

  $\vdots$
- N-ter Basiszustand mit $\alpha \neq 0$: Amplitude α, Bezeichnung.

Als Bezeichnung der Basiszustände wird hierbei eine Zahl zwischen 0 und $2^n - 1$ gewählt, die nach obiger Formel in die Binärschreibweise umgewandelt werden kann, um die Zustände der Einzelbits in diesem Basiszustand zu erhalten.

Als Programmiersprache zur Implementierung wurde Java wegen seiner übersichtlichen Objektorientierung, guten Unterstützung komplexer Zahlen (mittels eines Zusatzpakets), Plattformunabhängigkeit und guten grafischen Ausgabemöglicheit gewählt. So ist es möglich das Register auf dem Bildschirm zu visualisieren. Dabei wird die Amplitude jedes Basiszustands im Register zum einen als komplexer Zahlenwert ausgegeben, zum anderen wird jede Phase der komplexen Amplitude durch eine bestimmte Farbe repräsentiert. Es ist desweiteren denkbar, die Gattertransformation als Vektorverschiebung bzw. -drehung im $\mathbb{C}^n$ darzustellen und im Ausgabefenster anzuzeigen. Für $n \leq 3$ könnte man $\mathbb{C}^n$ durch eine Bloch-Kugel darstellen, z.B. ein Phasenverschiebungsgatter U_R würde dann Rotation des Vektors in einer Koordinatenebene bedeuten.

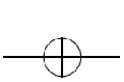
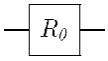
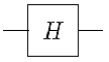
Bei der Implementierung der Simulation wird in der Klasse `Register` das Array oder die verkettete Liste (je nach Qubit Anzahl) initialisiert, das oder die die Amplituden der Registerbasisvektoren speichern sollen. Die Quantengatter werden als Unterfunktionen der Register-Klasse implementiert, sodass z.B. der Aufruf `reg.gate_hadamard(0)` zur Anwendung eines Hadamard-Gatters auf das erste (nullte) Qubit des Registers `reg` führt. Die Funktionsweise von Gattern und ihre genaue Implementierung wird im folgenden Kapitel beschrieben.

2.5 Gatter

2.5.1 Funktion: Operationen auf Qubits

Die Aufgabe von *Gattern* (engl. *Gates*) besteht darin, den Zustand des Registers zu verändern. Ein wichtiges Postulat der Quantenmechanik sagt aus, dass jede Veränderung eines Quantensystems durch einen linearen Operator dargestellt werden kann. Wenn also $|\psi\rangle$ der Zustand zum Zeitpunkt t und $|\psi'\rangle$ der Zustand zum Zeitpunkt t' ist, dann gibt es einen unitären Operator[2] U, so dass $|\psi'\rangle = U|\psi\rangle$. Die Unitarität von U wird gefordert, damit U reversibel bzw. invertierbar ist, d.h. $U^+U = I \Rightarrow U^+ = U^{-1}$, also $|\psi\rangle = U^{-1}|\psi'\rangle = U^+|\psi'\rangle$. Somit ist der Übergang von $|\psi\rangle$ zu $|\psi'\rangle$ reversibel, d.h. keine Information geht verloren. Umgedreht kann auch jede unitäre Matrix als Gatter aufgefasst werden.

In [1] wird gezeigt, dass sich jedes Gatter durch Verknüpfung von Gattern aus der Menge der sogenannten *universellen* Gatter ergibt. Diese vier universellen Gatter sind in folgender Tabelle aufgelistet.

Gatter	NOT	Controlled-NOT	Phasenverschiebung	Hadamard
Abkürzung	$U_\oplus$	$U_{\bullet\atop\oplus}$	U_R	U_H
Symbol				
Matrix	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{matrix} \quad\|00\rangle\|10\rangle\|01\rangle\|11\rangle \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{matrix} \|00\rangle \\ \|10\rangle \\ \|01\rangle \\ \|11\rangle \end{matrix} \end{matrix}$	$\begin{pmatrix} e^{i\theta} & 0 \\ 0 & e^{-i\theta} \end{pmatrix}$	$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

In der Symbolschreibweise steht jede horizontale Linie für ein Qubit. Das NOT-Gatter $U_\oplus$, das Rotations-Gatter U_R und das Hadamard-Gatter U_H modifizieren also nur ein Qubit, während das CNOT Gatter $U_{\bullet\atop\oplus}$ zwei Qubits bearbeitet.

Man denke sich zur Veranschaulichung der 1-Qubit-Gatter ein 1-Qubit Register im Zustand $|\psi\rangle$. Bezeichnet man mit $|0\rangle$ und $|1\rangle$ die beiden messbaren Basiszustände des Qubits, so lässt sich $|\psi\rangle$ wie oben gezeigt durch

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha\begin{pmatrix}1\\0\end{pmatrix} + \beta\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}\alpha\\\beta\end{pmatrix}$$

[2] Ein unitärer Operator ist analog zu einer unitären Matrix, vgl. Def. A.5, definiert, d.h. es gilt $U^+U = I$. Vgl. hierzu auch den Anhang.

darstellen. Anwendung der Gatter auf $|\psi\rangle$ ergibt:

$$\begin{aligned} U_{\oplus}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} &= \begin{pmatrix}\beta\\ \alpha\end{pmatrix} \\ U_R(\theta)\begin{pmatrix}\alpha\\ \beta\end{pmatrix} &= \begin{pmatrix}\alpha\cdot e^{i\theta}\\ \beta\cdot e^{-i\theta}\end{pmatrix} \\ U_H\begin{pmatrix}\alpha\\ \beta\end{pmatrix} &= \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix}. \end{aligned}$$

Das NOT-Gatter $U_{\oplus}$ vertauscht also die Amplituden α und β der Zustände $|0\rangle$ und $|1\rangle$. Wiederholtes Anwenden von NOT bzw. Controlled-NOT ermöglich es später beispielsweise die Reihenfolge der Amplituden umzukehren (vgl. Abschnitt 3.5). Das Rotations-Gatter U_R verändert die Phase zweier Amplituden. Wichtig ist hierbei, dass die Wahrscheinlichkeit $p_{|0\rangle} = |\alpha|^2 = \alpha\alpha^*$, den Basiszustand $|0\rangle$ zu messen, nach Anwendung von U_R auf $|0\rangle$ wegen $(\alpha e^{i\theta})(\alpha e^{i\theta})^* = (\alpha e^{i\theta})(\alpha^* e^{-i\theta}) = \alpha\alpha^* = p_{|0\rangle}$ unverändert bleibt. Das Hadamard-Gatter U_H dient schließlich der Erzeugung eines Superpositionszustandes von $|0\rangle$ und $|1\rangle$.

Zur Veranschaulichung des 2-Qubit-Gatters Controlled-NOT stelle man sich ein 2-Qubit-Register im Zustand $|\psi\rangle$ vor. $|\psi\rangle$ ist dann eine Linearkombination der vier Basiszustände $\{|0\rangle,\ldots,|3\rangle\}$ bzw. $\{|00\rangle, |10\rangle, |01\rangle, |11\rangle\}$ d.h.

$$|\psi\rangle = \alpha|00\rangle + \beta|10\rangle + \gamma|01\rangle + \delta|11\rangle = \begin{pmatrix}\alpha & \beta & \gamma & \delta\end{pmatrix}^T.$$

Anwendung des CNOT-Gatters ergibt dann

$$U_{\overset{\bullet}{\oplus}}|\psi\rangle = \begin{pmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\\ \gamma\\ \delta\end{pmatrix} = \begin{pmatrix}\alpha\\ \delta\\ \gamma\\ \beta\end{pmatrix}.$$

D.h. $U_{\overset{\bullet}{\oplus}}|x,y\rangle = |x, y\oplus x\rangle$, wenn $\oplus$ für Addition mod 2 bzw. XOR steht.

2.5.2 Simulation

Wie in Abschnitt 2.5.1 gezeigt wurde, lässt sich die von einem Gatter bewirkte Modifikation der Qubits als Matrixprodukt der Matrix U des Gatters mit dem Zustand $|\psi\rangle$ des Registers schreiben, $|\psi'\rangle = U|\psi\rangle$. Die Implementierung der Gatter unter Verwendung dieses Matrixprodukts ist jedoch sehr ineffizient und kostenintensiv, da bei großer Qubit Anzahl eine große Zahl an Multiplikationen und Summationen nötig ist. Eine wesentlich effizientere Simulationsmöglichkeit der Anwendung von Gattern wird durch die geschickte Bezeichnung der Basiszustände des Registers in der Binärschreibweise erreicht, d.h.

$$|i_0 i_1 \ldots i_{n-1}\rangle = \left|i_0 2^0 + i_1 2^1 + \cdots + i_{n-1}2^{n-1}\right\rangle.$$

Auf Grund der speziellen Form der universellen Gatter, kann die Simulation von Gattern so auf wenige Vertauschungen und wenige Additionen zurückgeführt werden.

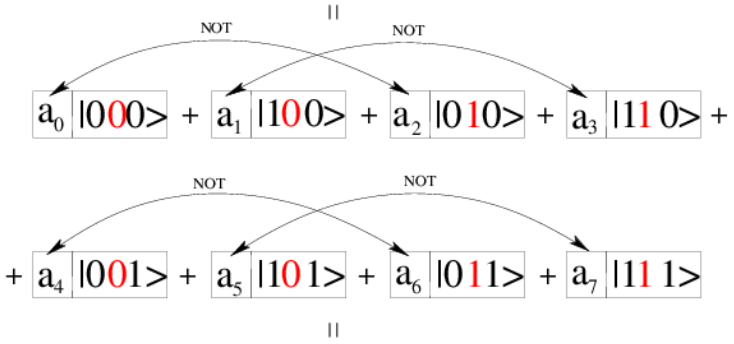
So lässt sich zunächst das *NOT-Gatter* sehr einfach simulieren. Wendet man dieses nämlich auf das *i*-te Bit des Registers an, so tauschen genau die Basiszustände ihre Amplituden, deren Bits bis auf das *i*-te Bit gleich sind. In der Simulation wird für einen bestimmten Basiszustand der „Tauschpartner“ gefunden, indem das *i*-te Bit des Basiszustands invertiert wird (0 zu 1 oder 1 zu 0). Wendet man also beispielsweise das NOT-Gatter auf das mittlere Bit eines 3-Bit-Register an, dann werden die Amplituden wie in Abb. 1 vertauscht.

Das *CNOT-Gatter* wird analog zum NOT-Gatter simuliert, mit der Zusatzbedinung, dass nur dann die Amplituden getauscht werden, wenn das Control-Bit 1 ist.Das *Rotationsgatter* kann simuliert werden, indem alle Amplituden des Registers mit $e^{i\theta}$ bzw. $e^{-i\theta}$ multipliziert werden.

Um die Anwendung des *Hadamard-Gatters* auf das *i*-te Bit zu simulieren, muss zunächst zu jedem Basiszustand $|a\rangle$ der Basiszustand $|b\rangle$ gesucht werden, der sich von $|a\rangle$ nur im *i*-ten Bit unterscheidet. Verwendet man wieder obiges Bezeichnungsschema, so erhält man den zu $|a\rangle$ gehörenden Basiszustand $|b\rangle$, indem man in der Binärschreibweise von *a* das *i*-te Bit von links invertiert. Wendet man also z.B. das Hadamard-Gatter auf das dritte Bit eines 6-Bit-Registers an, so gehört beispielsweise $|0\rangle = |000000\rangle$ zu $|8\rangle = |000100\rangle$, $|45\rangle = |101101\rangle$ zu $|101001\rangle = |37\rangle$. Somit ist also für jeden Basiszustand $|a\rangle$ der zu $|a\rangle$ gehörende Basiszustand $|b\rangle$ bekannt. Die Amplituden nach Anwendung des Hadamard-Gatters sind dann $\frac{\alpha+\beta}{\sqrt{2}}$ für $|a\rangle$ und $\frac{\alpha-\beta}{\sqrt{2}}$ für $|b\rangle$, wenn α bzw. β die zu $|a\rangle$ bzw. $|b\rangle$ gehörenden Amplituden sind. Für die Anwendung eines Hadamard Gatters auf das mittlere Qubit eines 3-Qubit-Registers ist dieser Simulationsalgorithmus in Abbildung 2 nochmals gezeigt.

Durch Kombination dieser vier Gatter kann jede andere unitäre Qubit-Transformation dargestellt werden, da die vier Gatter NOT-, CNOT-, Rotations- und Hadamard-Gatter universell sind. Rein formal kann also jede unitäre Matrix als Produkt der universellen Matrizen geschrieben werden (vgl. z.B. [1], [10], [7]). Weiß

$$U_\oplus |\psi\rangle = U_\oplus \Big[a_0 |000\rangle + a_1 |100\rangle + a_2 |010\rangle + a_3 |110\rangle + \\ + \; a_4 |001\rangle + a_5 |101\rangle + a_6 |011\rangle + a_7 |111\rangle \Big]$$



$$a_2 |000\rangle + a_3 |100\rangle + a_0 |010\rangle + a_1 |110\rangle + \\ + \; a_6 |001\rangle + a_7 |101\rangle + a_4 |011\rangle + a_5 |111\rangle .$$

Abb. 1: Simulation des NOT Gatters durch Amplitudenvertauschung.

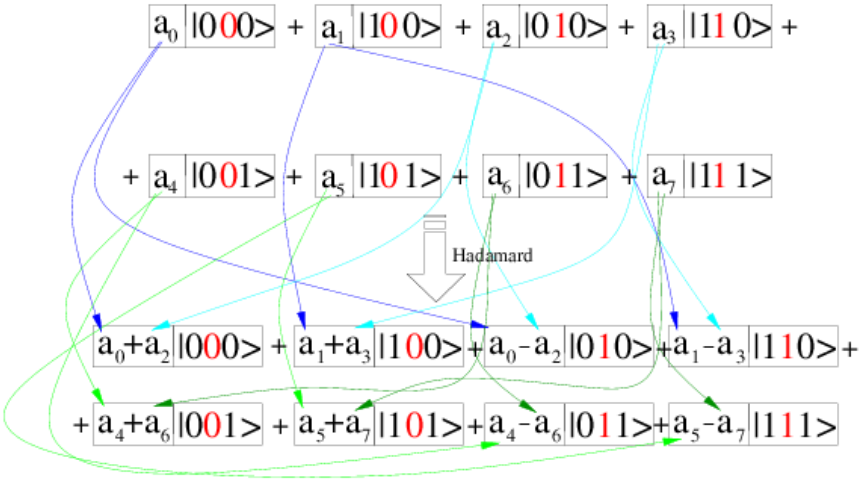


Abb. 2: Simulation des Hadamard-Gatters durch Amplitudenvertauschung und gegebenenfalls Multiplikation mit -1.

man für eine beliebige unitäre Matrix wie man sie durch Multiplikation von universellen Gattern erhält, so kann man die Multiplikation der universellen Gatter einfach als Hintereinanderausführung der Gatteroperationen auf das Register simulieren. Sollte die Darstellung einer unitären Matrix durch die universellen Gatter allerdings zu kompliziert oder nicht bekannt sein, so kann mit der Simulation auch jede beliebige unitäre Transformation auf Qubits durchgeführt werden, indem die Matrix direkt mit dem Registerzustand multipliziert wird.

2.6 Messungen

2.6.1 Bedeutung für Quantencomputer

Eine Messung des Registers ist notwendig, damit der Anwender in der klassischen Welt ein Ergebnis aus einer Berechnung ablesen kann. Während der Anwendung von Gattern auf das Register befindet sich dieses in einem Quantenzustand, der von den Gattern manipuliert wird. Ein Quantenalgorithmus überführt nun den Anfangs-Quantenzustand $|\psi\rangle$ des Registers in einen End-Quantenzustand $|\psi'\rangle$. Dieser End-Quantenzustand selbst ist jedoch noch nicht von großem Nutzen für den Anwender, da dieser nur Basis- bzw. Eigenzustände bzgl. einer bestimmter Basis bzw. eines bestimmten Operators (der einer physikalischen Größe entspricht, z.B. Impuls) und niemals einen Superpositionszustand selbst messen kann. Jede Messung führt also zum Verlust der Superposition, d.h. jedes Qubit befindet sich nach der Messung entweder im Zustand 0 oder 1. Das Register „springt“ [6] also durch eine Messung von einem beliebigen Superpositionszustand in einen seiner 2^n Basiszustände. Welchen der 2^n Basiszustände man bei einer Messung erhält ist abhängig von den Amplituden α_i jedes Basiszustands. Die Wahrscheinlichkeit den Basiszustand $|i\rangle$ zu messen entspricht dabei

dem Amplitudenquadrat dieses Basiszustands $P(|i\rangle) = |\alpha_i|^2 = \alpha_i \alpha_i^*$. Bei Quantencomputer-Algorithmen muss der Endzustand deshalb so konstruiert sein, dass aus den Werten eines einzigen durch Messung erhaltenen Basiszustand das Ergebnis der Berechnung mit absoluter oder zumindest hoher Sicherheit abgelesen werden kann. Z.B. kann das Ergebnis einer bestimmten Berechnung im ersten Qubit gespeichert werden. Dann ist es entscheidend, dass bei der Messung das erste Qubit mit Wahrscheinlichkeit 1 in den Zustand mit dem berechneten Wert „springt“. Wenn das Ergebnis einer Berechnung z.B. 1 ist, dann müsste also der Endzustand von der Form $|\psi-\rangle = (0 \cdot |0_a\rangle + 1 \cdot |1_a\rangle)(\alpha_2 |0_b\rangle + \alpha_3 |1_b\rangle) \cdots$ sein, da dann mit Wahrscheinlichkeit 1 beim ersten Qubit (a-Qubit) der Wert 1 gemessen wird und das Ergebnis mit Sicherheit angegeben werden kann.

2.6.2 Simulation

Eine Messung kann simuliert werden, indem von einer gleichverteilten Zufallszahl q zwischen 0 und 1 zunächst die Wahrscheinlichkeit $|\alpha_0|^2$, dann die Wahrscheinlichkeit $|\alpha_1|^2$ usw. abgezogen wird bis q kleiner als ein $|\alpha_i|^2$ ist. In Pseudocode kann man dies wie folgt formulieren:

1. Ermittle gleichverteilte Zufallszahl q, $0 \leq q < 1$.
2. `for` $(i = 0;\ \|\alpha_i\|^2 > q;\ i$++$)$ q-=$\|\alpha_i\|^2$.
3. Der gemessene Zustand ist $\|i\rangle$.

Die Wahrscheinlichkeitsverteilung der Basiszustände $|i\rangle$ ist nun durch die Amplitudenquadrate $|\alpha_i|^2$ bestimmt. Sei beispielsweise bei einem 2-Qubit-Register $|\alpha_0|^2 = 0.2, |\alpha_1|^2 = 0.3, |\alpha_2|^2 = 0.4$ und $|\alpha_3|^2 = 0.1$. Dann ist die Wahrscheinlichkeit $|0\rangle$ als Ergebnis des Messalgorithmus zu erhalten 0.2, da nur in 20% aller Fälle die Zufallszahl q minus 0.2 kleiner als 0 ist, nämlich genau dann wenn q zwischen 0 und 0.2 liegt. Die Wahrscheinlichkeit für den Output von $|1\rangle$ ist 0.3, da $|1\rangle$ nur ausgegeben wird, wenn die Zufallszahl q zwischen 0.2 und 0.5 lag, was mit einer Wahrscheinlichkeit von 0.3 eintritt.

3 Quanten-Algorithmen

3.1 Ziele

Quanten-Algorithmen sind Netzwerke (circuits) von Quantengattern. Die Hauptaufgaben solcher Quanten-Algorithmen lassen sich in folgende Gebiete unterteilen.

1. Entwicklung von Quanten-Algorithmen, die bestimmte Probleme schneller lösen können als klassische Algorithmen.
2. Experimentelle Umsetzung leichter Quanten-Algorithmen.
3. Abstraktion bestimmter Methoden, die für die Entwicklung effizienter Quanten-Algorithmen benötigt werden, z.B. Quanten-Fourier-Transformation.

Bevor nun auf einige Beispiele dieser Aufgaben von Quanten-Algorithmen eingegangen wird, soll noch kurz ein für viele Quanten-Algorithmen sehr wichtiges Phänomen besprochen werden.

3.2 Parallelismus

Die Anwendung eines unitären Operators A auf einen Zustand

$$|\psi\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle$$

ergibt

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

d.h. durch die Anwendung *eines* Operators A auf *ein* Register werden 2^n Zustände verändert. Dieser sogenannte *Quanten-Parallelismus* resultiert letztlich aus der Linearität von A und ist von großer Wichtigkeit für viele Quanten-Algorithmen, weil er es ermöglicht, bestimmte Probleme mit einem Quanten-Algorithmus schneller zu lösen als mit klassischen Algorithmen.

Um die Funktionsweise von Quanten-Algorithmen zu verstehen, sollen in den folgenden Kapitel exemplarisch an zwei wichtigen Beispielalgorithmen ausführlich erklärt werden, dem XOR-Problem von Deutsch und der Quanten-Fourier-Transformation. Anschließend werden weitere Quantenalgorithmen kurz vorgestellt, die mit Hilfe der Simulation simuliert werden können.

3.3 Münzproblem von Deutsch

Das Problem von Deutsch besteht schlichtweg darin, die Echtheit einer Münze zu testen, d.h. zu testen, ob eine Seite „Kopf“ und die andere „Zahl“ ist. Die Frage ist hierbei, wie oft man auf die Münze sehen muss, um sicher sagen zu können, ob es sich um eine echte oder falsche Münze handelt. In der klassischen Welt ist das Ergebnis klar – man muss die Münze von jeder Seite ansehen, also zwei Mal. In der Quantenmechanik genügt es jedoch, nur ein Mal auf die Münze, nämlich auf eine Superposition von $|\text{Kopf}\rangle$ und $|\text{Zahl}\rangle$, zu sehen.

Dieses Problem lässt sich mathematisch durch eine „Black Box“ oder ein „Orakel“ [9] beschreiben, welche(s) eine Funktion $f : \{1,\ldots,n\} \to \{0,1\}$ berechnet und das Ergebnis ausgibt.[1] Bei dem beschriebenen Münz-Problem ist $f : \{0,1\} \to \{0,1\}$ zu wählen. Die Aufgabe besteht darin, festzustellen, ob die Ergebnisse von $f(0)$ und $f(1)$ gleich oder verschieden sind, d.h. ob $f(0) \oplus f(1) = 0$ oder 1 gilt, wobei $\oplus$ für die Addition modulo 2 bzw. XOR steht. Während es in der klassischen Welt unmöglich ist, nach einer Anwendung von f festzustellen, ob die Ausgaben von f verschieden oder gleich sind, ist genau dies in der Quantenmechanik durch Verwendung eines Superpositionszustandes von f möglich.

Ein Algorithmus, der diese Möglichkeit demonstriert, ist in Abbildung 3 skizziert (vgl. z.B. [7], [10]. H

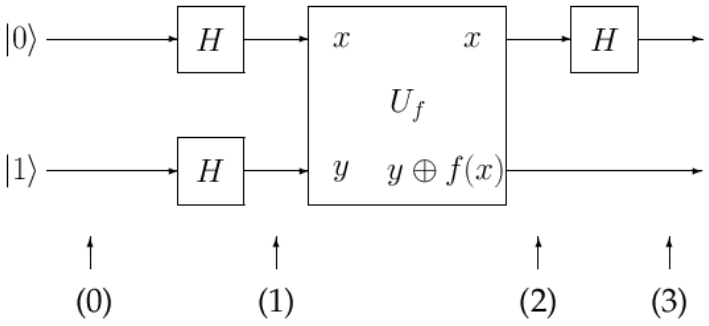


Abb. 3: Algorithmus zur Lösung des Münzproblems von Deutsch.

steht für das Hadamard-Gatter und U_f für das f-controlled-NOT Gatter, das durch

$$U_f\,|x,y\rangle := |x, y \oplus f(x)\rangle \tag{3.1}$$

definiert ist. Dieses f-controlled-NOT Gatter soll die „Berechnung“ von f darstellen, wobei das erste Qubit für den Input und das zweite Qubit für den Output von f steht. Ist nämlich das zweite Qubit 0, so überführt U_f den Zustand $|x,0\rangle$ in $|x, f(x)\rangle$, d.h. es weist dem zweiten Qubit den Output $f(x)$ für den Input x zu. Damit das Gatter reversibel und durch eine unitäre Matrix darstellbar ist, wird es in (3.1) für alle Zustände des zweiten Qubits, also auch $y = 1$ definiert.

Der Algorithmus lautet nun wie folgt. Im Anfangszustand (0) befinde sich das Bit a im Zustand 0, das Bit b im Zustand 1, der Gesamtzustand kann also mit $|0_a\rangle\,|1_b\rangle$ bezeichnet werden. Anwenden des Hadamard-Gatters auf Bit a dieses Zustands führt wegen

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

zu einem Zustand der Form[2] $|0_a\rangle + |1_a\rangle$. Anwenden des Hadamard-Gatters auf Bit b führt zu einem Zustand $|0_b\rangle - |1_b\rangle$. Der Gesamtzustand bei (1) ist also $(|0_a\rangle + |1_a\rangle)(|0_b\rangle - |1_b\rangle)$

Nun ist für jedes $x \in \{0,1\}$

$$U_f\,|x_a\rangle\,(|0_b\rangle - |1_b\rangle) = |x_a\rangle\,(|0_b \oplus f(x)\rangle - |1_b \oplus f(x)\rangle) = (-1)^{f(x)}\,|x_a\rangle\,(|0_b\rangle - |1_b\rangle),$$

weil

$$|0_b \oplus f(x)\rangle - |1_b \oplus f(x)\rangle = \begin{cases} |0_b \oplus 0\rangle - |1_b \oplus 0\rangle = (-1)^0(|0_b\rangle - |1_b\rangle) & \text{falls } f(x) = 0, \\ |0_b \oplus 1\rangle - |1_b \oplus 1\rangle = |1_b\rangle - |0_b\rangle = (-1)^1(|0_b\rangle - |1_b\rangle) & \text{falls } f(x) = 1. \end{cases}$$

Wendet man jetzt das f-controlled-NOT Gatters auf den obigen Zustand der Form $(|0_a\rangle + |1_a\rangle)(|0_b\rangle - |1_b\rangle)$ an, so folgt für den Gesamtzustand am Punkt (2)

$$U_f\,(|0_a\rangle + |1_a\rangle)(|0_b\rangle - |1_b\rangle) = ((-1)^{f(0)}\,|0_a\rangle + (-1)^{f(1)}\,|1_a\rangle)(|0_b\rangle - |1_b\rangle).$$

[1] D.h. von der Funktion f weiß man nichts mehr als die einzelnen Ausgaben des Orakels.

[2] Die Normierungsfaktoren sind bei der Erklärung des Algorithmus ohne Bedeutung und werden daher in diesem Abschnitt zur besseren Übersichtlichkeit weggelassen.

Der Zustand von Bit a in (2) kann auch als

$$(-1)^{f(0)}(|0_a\rangle + (-1)^{f(0)\oplus f(1)}|1_a\rangle)$$

geschrieben werden. Wendet man auf diesen Zustand von a das zweite Hadamard-Gatter an, so wird der Zustand in (3) zu

$$(-1)^{f(0)}|f(0)\oplus f(1)\rangle.$$

Somit ist das Bit a schließlich im Zustand $|0_a\rangle$, falls f konstant ist (d.h. $f(0)\oplus f(1)=0 \Leftrightarrow f(0)=f(1)$), und im Zustand $|1_a\rangle$, falls f verschieden ist (d.h. $f(0)\oplus f(1)=1 \Leftrightarrow f(0)\neq f(1)$).

Wegen $|(-1)^x|^2=1$ lässt sich nach einer Messung mit *Sicherheit* sagen, ob f konstant oder verschieden ist. Faszinierenderweise ist zu dieser Feststellung lediglich *eine* Anwendung des U_f-Gatters, d.h. nur *eine einzige* Befragung des Orakels nötig! Fasst man f wie am Anfang des Abschnittes wieder als Münze auf, so ermöglicht der beschriebene Algorithmus durch lediglich *einmaliges* „Hinsehen“ festzustellen, ob es sich bei der Münze um ein falsches oder ein echtes Exemplar handelt. Bemerkenswerter Weise wurde dieser Algorithmus Ende 1998 von Jones und Mosca experimentell mit Hilfe der kernmagnetischen Resonanz (NMR) umgesetzt.[3]

Simulation

Die Simulation des Deutsch-Algorithmus kann durch Simulation der universellen Gatter und eines Registers einfach simuliert werden. In Pseudocode lautet der Deutsch-Algorithmus

```
reg.gate_hadamard(0)
reg.gate_hadamard(1)
reg.gate_f_controlled_NOT(0,1,f)
reg.gate_hadamard(0)
```

`reg` ist dabei ein Objekt der Klasse `Register`, `gate_...` sind die Gatter, die beim Deutsch-Algorithmus verwendet werden. Als Argumente werden an die Gatterfunktionen die Qubits übergeben, auf die das Gatter angewandt werden soll. Das f-controlled-Gatter besitzt zudem als weiteres Argument die Funktion f.

Die grafische Ausgabe der Simulation des Deutsch-Algorithmus ist in Abb. 9 bis Abb. 12 im Anhang abgebildet. Durch Klicken auf den „next step“-Button kann man den Registerzustand an jeder Stelle des Algorithmus betrachten. Die Phasen der komplexen Amplituden der Basiszustände werden durch Farben repräsentiert, die zugehörige Wahrscheinlichkeit für das Messen des jeweiligen Basiszustands mit einer Grauskala. Die Funktion f kann rechts durch Anklicken von Boxen eingestellt werden. Die Funktionsweise des Deutsch-Algorithmus wird auf diese Weise sehr anschaulich dargestellt.

3.4 Verallgemeinerung des Münzproblems: Deutsch-Jozsa-Algorithmus

Eine Verallgemeinerung des XOR-Algorithmus ist der Deutsch-Jozsa-Algorithmus, der entscheiden kann, welche der beiden möglichen Alternativen der Eingabefunktion $f:\{0,1\}^n\to\{0,1\}$ vorliegt:

- Entweder ist f *konstant* (alle Eingaben werden auf die gleiche Ausgabe 0 oder 1 abgebildet),
- oder f ist *balanciert* (die Hälfte der Eingaben wird auf 0, die andere Hälfte auf 1 abgebildet).

Das Register besteht aus zwei Teilregistern, das erste $|x_0,\ldots,x_{n-1}\rangle$ mit n Qubits, das zweite $|y\rangle$ mit einem Qubit. Der Schaltkreis für den Algorithmus ist in Abb. 4 gezeigt. Im Anfangszustand des Registers sind die ersten n Qubits $|0\rangle$, das Qubit im zweiten Teilregister $|y\rangle=|1\rangle$ (analog zum Deutsch-Algorithmus für $n=2$). Der Algorithmus wendet zunächst auf alle $n+1$ Qubits ein Hadamard-Gatter an, sodass das Register von $|\psi_1\rangle=|0\ldots0\rangle|1\rangle$ in den Zustand

$$|\psi_2\rangle=\frac{1}{\sqrt{2^{n+1}}}(|0\rangle+|1\rangle)\cdots(|0\rangle+|1\rangle)\cdot(|0\rangle-|1\rangle)=\frac{1}{\sqrt{2^{n+1}}}(|0\rangle+|1\rangle+|2\rangle+\cdots+|2^n-1\rangle)\cdot(|0\rangle-|1\rangle)$$

überführt wird. Anschließend wird das U_f-Orakel

$$U_f|x_0,\ldots,x_{n-1}\rangle|y\rangle=|x_0,\ldots,x_{n-1}\rangle|y\oplus f(x_0,\ldots,x_{n-1})\rangle$$

auf das Register angewandt. Dies führt zu

$$|\psi_3\rangle=U_f|\psi_2\rangle=\frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle(|f(x)\rangle-|1\oplus f(x)\rangle)=\frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle(|0\rangle-|1\rangle)$$

[3] Vgl. [7] S. 312f.

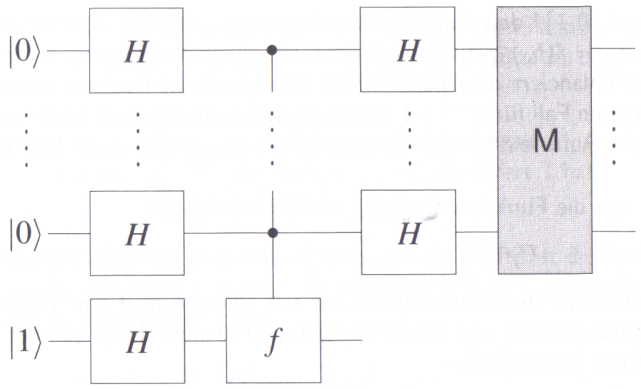


Abb. 4: Schaltkreis des Algorithmus von Deutsch-Jozsa zur Entscheidung, ob die Funktion f balanciert oder konstant ist.

analog zur Wirkung von U_f im Deutsch-Algorithmus für zwei Bits. Nun werden auf das erste Teilregister n Hadamard-Gatter angewandt, um die Superposition in Basiszustände zu transformieren aus denen das Ergebnis ablesbar ist. Die Anwendung der n Hadamard-Gatter führt zu[4]

$$|\psi_4\rangle = H_n|\psi_3\rangle = \frac{1}{\sqrt{2^{2n+1}}}\left(\sum_{z=0}^{2^n-1}\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot z}|z\rangle\right)\cdot(|0\rangle - |1\rangle).$$

Nun wird das erste Teilregister gemessen. Falls $f(x)$ konstant ist, dann ist der Faktor $(-1)^{f(x)}$ konstant. Für ein beliebiges $z \neq 0$ ist der Faktor $(-1)^{x\cdot z}$ für die Hälfte der Summanden 1, für die andere Hälfte -1, d.h. die Amplitude von $|z\rangle \neq |0\rangle$ ist 0. Für $z = 0$ ist $(-1)^{x\cdot z}$ immer 1, d.h. die Amplitude von $|0\rangle$ ist 1. Im Fall f konstant ist das Ergebnis der Messung also $|0\rangle$. Falls f balanciert ist, ist für $z = 0$ der Faktor $(-1)^{x\cdot z}$ immer 1, aber der Faktor $(-1)^{f(x)}$ ist für die Hälfte der Summanden 1, für die andere Hälfte -1, sodass die Amplitude von $|0\rangle$ 0 ist. Die Amplitude aller Zustände $|z\rangle \neq |0\rangle$ ist verschieden von 0, das der Faktor $(-1)^{x\cdot z}$ nicht konstant für alle x ist. Ist f also balanciert, dann ist das Ergebnis der Messung von $|0\rangle$ verschieden.

Screenshots von der Simulation des Deutsch-Jozsa-Algorithmus für $n = 4$ und ein balanciertes f befinden sich im Anhang (Abb. 13 bis 18. Die Hadamard-Gatter sowie das U_f-Gatter können wie oben beschrieben durch Amplituden-Vertauschungen simuliert werden.

3.5 Quanten-Fourier-Transformation (QFT)

Analog zu einer klassischen Fourier-Transformation, die z.B. bei einem Audiosignal einen Basiswechsel zwischen Zeit- und Frequenzbereich bewirkt, kann die Quanten-Fourier-Transformation (QFT) abstrakt gesehen als Basistransformation eines Registerzustands von der Standardbasis $\{|0\rangle,\ldots,|N-1\rangle\}$ in die Basis $\{\mathrm{QFT}|0\rangle,\ldots,\mathrm{QFT}|N-1\rangle\}$ aufgefasst werden.[5] Ihre Notwendigkeit und ihre Einsatzmöglichkeiten für Quantenalgorithmen werden weiter unten bei der Beschreibung des Shor-Algorithmus deutlich werden, in der die QFT eine Superposition von Zuständen so transformiert, dass durch eine Messung des Zustands nach Anwendung der QFT die Information über das Ergebnis des Algorithmus gewonnen werden kann.

Rein formal transformiert die QFT einen Inputvektor $(x_0,\ldots,x_{N-1})$ komplexer Zahlen x_k in einen Outputvektor $(y_0,\ldots,y_{N-1})$ komplexer Zahlen y_k, wobei der Outputvektor definiert ist durch

$$y_k := \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} x_j e^{2\pi i/N\cdot jk}.$$

Wendet man die QFT auf einen Standardbasisvektor $|j\rangle \in \{|0\rangle,\ldots,|N-1\rangle\}$ an, so erhält man eine Summe über alle Basisvektoren:

$$QFT\,|j\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi i/N\cdot jk}|k\rangle\,. \tag{3.2}$$

Die Multiplikation mit $e^{2\pi i/N\cdot jk}$ entspricht wegen der Eulerschen Formel $e^{i\varphi} = \cos\varphi + i\sin\varphi$ einer Drehung der Phase um den Winkel $\varphi = 2\pi/N\cdot jk$, d.h. um das $j\cdot k$-fache eines $2\pi/N$-Kreissektors[6]. Das heißt die

[4] Die Auswirkung von H_n erhält man durch sukzessive Anwendung des Hadamard-Gatters. Z.B. zwei Hadamard-Gatter auf den Zwei-Bit-Zustand $|01\rangle$: $|0_x1_y\rangle \overset{H_x}{\to} \frac{1}{\sqrt{2}}(|0_x1_y\rangle + |1_x1_y\rangle) \overset{H_y}{\to} \frac{1}{\sqrt{2^2}}(|0_x0_y\rangle + |1_x0_y\rangle - |0_x1_y\rangle - |1_x1_y\rangle)$. Hier erkennt man, dass $H_2|x\rangle = \frac{1}{2}\sum_{z=0}^{3}(-1)^{x\cdot z}|z\rangle$ gilt mit $x\cdot y = \bigoplus_{i=1}^{n} x_i y_i$, denn $01\cdot 00 = 0, 01\cdot 10 = 0, 01\cdot 01 = 1, 01\cdot 11 = 1$. So lässt sich allgemein für die Operation von n Hadamard-Gattern auf n Qubits herleiten: $H_n|x\rangle = \frac{1}{\sqrt{2^n}}\sum_z(-1)^{x\cdot z}|z\rangle$.

[5] Die QFT ist also nicht eine klassische Fourier-Transformation, die effizienter ausgeführt werden kann. Sie gleicht der klassischen Fourier-Transformation lediglich in dem abstrakten Prinzip einen Basiswechsel auszuführen, der einem Informationsgewinn dient.

[6] Auch bezeichnet als „komplexe N-te Einheitswurzel“, da $e^{2\pi i/N}$ Lösung von $x^N - 1 = 0$ ist.

QFT transformiert einen Basiszustand in eine Superposition aller Basiszustände, deren Amplituden zwar unterschiedliche Phasen aber die gleiche Länge $\frac{1}{\sqrt{N}}|e^{i\varphi}| = \frac{1}{\sqrt{N}}$ besitzen. Insofern ist die QFT ähnlich zu N aufeinanderfolgenden Hadamard-Gattern, die einen Basiszustand ebenfalls in eine Superposition über alle Basiszustände transformiert, allerdings ohne die Phasenverschiebung der QFT. $\mathrm{QFT}\,|0\rangle, \ldots, \mathrm{QFT}\,|N-1\rangle$ unterscheiden sich voneinander dadurch, dass die Amplituden der Superposition unterschiedliche Vielfache des $2\pi/N$-Sektors sind. Wie diese Eigenschaft ausgenutzt werden kann wird im Kapitel über Shors Algorithmus gezeigt.

Im Folgenden wird gezeigt wie sich die QFT als Quantenschaltkreis umsetzen lässt. In [3] wird gezeigt, dass sich die QFT in einer Produktform angegeben lässt. Hierzu wähle man $N = 2^n$, $n \in \mathbb{N}$ und $|0\rangle, \ldots, |2^n - 1\rangle$ als orthonormale Basis eines n-Qubit Registers. Diese Basisvektoren bezeichnet man nach dem dyadischen System, d.h. $|j\rangle = \left|j_1 2^{n-1} + j_2 2^{n-2} + \cdots + j_n 2^0\right\rangle = |j_1 j_2 \ldots j_n\rangle$. Zudem wird die Zahl $j_l/2 + j_{l+1}/4 + \cdots + j_m/2^{m-l+1}$ mit $0.j_l j_{l+1} \ldots j_m$ bezeichnet. Dann lautet die zu Gleichung (3.2) äquivalente Produktform

$$QFT\,|j_1, \ldots, j_n\rangle = \frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{2\pi i 0.j_n}\,|1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_{n-1} j_n}\,|1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}\,|1\rangle\right). \tag{3.3}$$

Mit Hilfe dieser Produktform ist es möglich, die QFT mit dem in Abb. 1 dargestellten Quanten-Algorithmus durchzuführen. Hierzu benötigt man das in Abschnitt 2.5.1 beschriebene Hadamard-Gatter H und das Gatter R_k, welches durch

$$R_k := \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$$

definiert ist. Der Input in den Algorithmus ist der Zustand $|j\rangle = |j_1 \ldots j_n\rangle$. Wendet man das Hadamard-Gatter

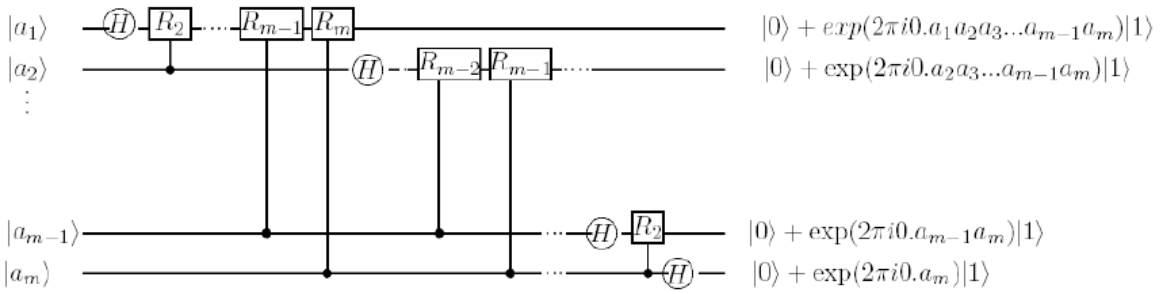


Abb. 5: Implementierung der QFT nach Cleve [3].

H auf das erste Bit an, dann

$$H\,|j\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i 0.j_1}\,|1\rangle\right) |j_2 \ldots j_n\rangle, \quad \text{weil } e^{2\pi i 0.j_1} = \begin{cases} -1 & \text{falls } j_1 = 1, \\ +1 & \text{sonst.} \end{cases}$$

Anwendung des Controlled-R_2-Gatters[7] auf $|j_2\rangle$ als Controll-Bit und $|j_1\rangle$ als Operations-Bit führt zu dem Zustand

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2}\,|1\rangle\right) |j_2 \ldots j_n\rangle.$$

Wendet man nun nacheinander die Gatter Controlled-R_3, -R_4, …, bis -R_n an, so wird durch jedes dieser Gatter ein weiteres Bit zur Phase des Koeffizienten von $|1\rangle$ hinzugefügt und man gelangt schließlich auf den Zustand

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}\,|1\rangle\right) |j_2 \ldots j_n\rangle.$$

Nun wird dasselbe Schema auf das 2. Bit angewendet. Das Hadamard-Gatter ergibt den Zustand

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}\,|1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_2}\,|1\rangle\right) |j_3 \ldots j_n\rangle.$$

Die Gatter Controlled-R_2 bis R_{n-1} führen zum Zustand

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}\,|1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_2 \ldots j_n}\,|1\rangle\right) |j_3 \ldots j_n\rangle.$$

[7] Der Controlled-Zusatz bedeutet analog zum Controlled-NOT-Gatter, dass das R_2-Gatter nur ausgeführt wird, falls das Controll-Bit (hier $|j_2\rangle$) gesetzt ist.

Setzt man dieses Schema für jedes Qubit fort, so erhält man schließlich den Endzustand

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2 \cdots j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right).$$

Dreht man die Reihenfolge der Qubits um, so ist der Zustand gleich der Produktform (3.3). Dieses Umdrehen der Reihenfolge kann durch wiederholtes Anwenden des sogenannten Swap-Circuits

$$\underbrace{\mathrm{CNOT}(|a\rangle,|b\rangle)}_{1}\ \underbrace{\mathrm{CNOT}(|b\rangle,|a\rangle)}_{2}\ \underbrace{\mathrm{CNOT}(|a\rangle,|b\rangle)}_{3}$$

erreicht werden, da

$$|a,b\rangle \xrightarrow{1} |a,a\oplus b\rangle \xrightarrow{2} |a\oplus(a\oplus b),a\oplus b\rangle = |b,a\oplus b\rangle \xrightarrow{3} |b,(a\oplus b)\oplus b\rangle = |b,a\rangle,$$

wenn $\oplus$ für die Addition modulo 2 steht, die äquivalent zur Invertierung eines Bits ist. Somit ist die QFT mit Hilfe des beschriebenen Algorithmus durchführbar.

Die Simulation kann für nicht zu große m durchgeführt werden, da sowohl das Hadamard-Gatter als auch das Rotationsgatter R_k wie weiter oben beschrieben simuliert werden können. Analog zum Deutsch-Algorithmus lässt sich die Funktionsweise der QFT dann durch eine Simulation gut veranschaulichen.

3.6 Faktorisierungsalgorithmus von Shor

Der *Faktorisierungs-Algorithmus von Shor* [13] ermöglicht es, die Zerlegung einer natürlichen Zahl n in ihre Primfaktoren $p,q,\ldots$ mit nicht-exponentiellem Aufwand zu lösen. Da das Problem der Primfaktorzerlegung mit großer Wahrscheinlichkeit NP-vollständig ist (z.B. beruht die RSA-Verschlüsselung auf dieser Annahme), ist ein nicht-exponentieller Lösungsalgorithmus zur Primfaktorzerlegung mittels klassischer Computer vermutlich nicht möglich. Im Folgenden soll die allgemeine Funktionsweise des Algorithmus kurz aufgezeigt werden und an einem konkreten Beispiel verdeutlicht werden. Anschließend soll der Algorithmus mit Hilfe der Simulation simuliert werden.

Sei n die zu faktorisierende natürliche Zahl. O.B.d.A. kann angenommen werden, dass n aus genau zwei Primfaktoren besteht, da im Fall von mehr als zwei Primfaktoren diese nach und nach abgespaltet werden können. Zunächst wird eine sogenannte modulare Exponentiation durchgeführt, d.h. es wird die Funktion $f:\mathbb{N}\to\mathbb{N}, f(x)=a^x \bmod n$ gebildet, wobei mod für die Modulo-Funktion steht (Rest beim Teilen durch n) und a eine natürliche Zahl ist, die teilerfremd mit n ist und für die $1<a<n$ gilt. Man kann nun zeigen, dass diese Funktion periodisch in x ist, d.h. es gibt eine *Periode* $r\in\mathbb{N}$ mit $f(x+r)=f(x)$ für alle $x\in\mathbb{N}$. Weiter lässt sich zeigen, dass man mit Hilfe dieser Periode r einen Primfaktor von n berechnen kann: Da die Funktion f unendlich viele Eingabewerte, aber nur n (endlich viele) Ausgabewerte hat, gibt es zwei natürliche Zahlen $x,y\in\mathbb{N}$ mit $f(x)=f(y)$. Daraus folgt

$$a^x \bmod n = a^y \bmod n \ \Rightarrow\ (a^x-a^y)\bmod n = a^x\cdot(1-a^{y-x})\bmod n = 0 \ \Rightarrow\ a^{y-x}-1 \bmod n = 0.$$

Der letzte Schritt folgt daraus, dass a und n teilerfremd sind (d.h. $a^x \neq 0 \bmod n$). Wegen $f(x)=f(y)$ ist $y-x$ die Periode r von f, d.h. es gilt $a^r = 1 \bmod n$. Deshalb gilt für beliebige $z\in\mathbb{N}$

$$f(z+r) = a^{z+r} \bmod n = (a^z \bmod n \cdot a^r \bmod n)\bmod n = a^z \bmod n = f(z).$$

Falls die Periode r gerade ist, gilt die binomische Formel und aus $a^r-1\bmod n=0$ folgt $(a^{r/2}+1)\cdot(a^{r/2}-1)\bmod n=0$. Nun sind die größten gemeinsamen Teiler $\mathrm{ggT}(a^{r/2}-1,n)$ und $\mathrm{ggT}(a^{r/2}11,n)$ echte Teiler von n, außer $a^{r/2}-1$ ist mit n teilerfremd und $a^{r/2}+1$ ist ein Vielfaches von n, was nach [8] jedoch unwahrscheinlich ist. Zusammengefasst lautet der Algorithmus für eine Input-Zahl $n\in\mathbb{N}$ also:

1. Wähle zufällig ein $a\in\{2,\ldots,n-1\}$ mit $\mathrm{ggT}(a,n)=1$,
2. ermittle die Periode von $f(x)=a^x \bmod n$,
3. Falls r ungerade ist gehe zu Schritt 1,
4. Berechne $\mathrm{ggT}(a^{r/2}-1,n)$ und $\mathrm{ggT}(a^{r/2}11,n)$. Wenn sich kein echter Teiler ergibt, gehe zu 1. Schritt. Sonst gebe echten Teiler aus.

Mit Hilfe der Zahlentheorie lässt sich zeigen, dass die Wahrscheinlichkeit, dass dieser Algorithmus einen echten Teiler liefert, sehr groß ist (vgl. z.B. [8] S. 198ff).

Die Funktionsweise soll am Beispiel von $n=15=3\cdot5$ gezeigt werden. Zunächst wird ein zufälliges a zwischen 1 und 15 gewählt, das mit 15 teilerfremd ist. In Frage kommen alle Zahlen zwischen 1 und 15 außer 3 und 5, z.B. $a=7$. Nun wird $f(x)=7^x \bmod n$ gebildet und die Periode von f bestimmt:

x	0	1	2	3	4	5	6	7
$f(x)$	1	7	4	13	1	7	4	13

Die Periode von f ist also $r = 4$. Da die Periode gerade ist, werden die beiden ggTs berechnet, $\text{ggT}(15, 49+1) = 5$ und $\text{ggT}(15, 49-1) = 3$. Dies sind die beiden Primfaktoren von $n = 15 = 3 \cdot 5$.

In dem Algorithmus sind alle Schritte auf einem klassischen Rechner in nicht-exponentieller Zeit ausführbar bis auf den Schritt in dem die Periode r der Funktion f bestimmt wird. Um nun einen vollständig nicht-exponentiellen Algorithmus zu erhalten, muss die Periode von f in nicht-exponentieller Zeit bestimmbar sein. Da dies mit klassischen Rechnern sehr wahrscheinlich nicht möglich ist, muss für diese Aufgabe ein Quantencomputer verwendet werden. Der gesamte Algorithmus besteht dann aus dem oben beschriebenen klassischen Teil und dem nicht-klassischen Teil zur Periodenbestimmung, den ein Quantencomputer ausführen muss.

Gesucht ist nun also ein Quantenalgorithmus, dessen Eingabe eine Funktion $f : \mathbb{N} \to \{0, \ldots, n-1\}$ mit Periode r ist und dessen Ausgabe der Wert der Periode r sein soll. Um die Anzahl der Qubits endlich zu halten und dennoch mit großer Wahrscheinlichkeit ein Ergebnis zu erhalten, wird der Definitionsbereich von f eingeschränkt auf die Menge $\{0, \ldots, n^2\}$. Die Funktion f wird dem Algorithmus dann analog zum Deutsch-Algorithmus durch ein f-controlled-Gatter $U_f : |a, b\rangle \to |a, b \oplus f(a)\rangle$ zur Verfügung gestellt.

Der Algorithmus zur Bestimmung der Periode von f ist in Abb. 6 gezeigt. Das Register wird in zwei

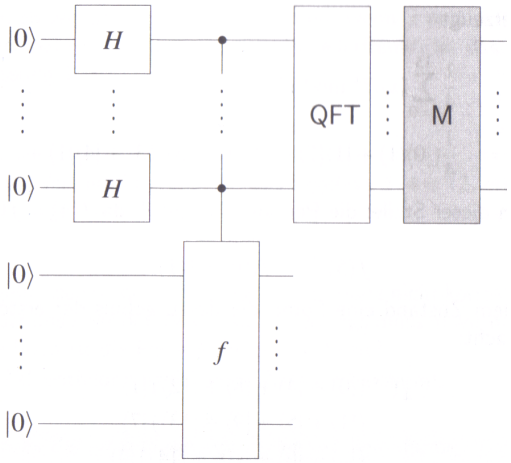


Abb. 6: Schaltkreis des Algorithmus von Shor zur Bestimmung der Periode r der Funktion f (vgl. [8]).

Teilregister $|a\rangle$ und $|b\rangle$ aufgeteilt, wobei das erste aus $|a| = \lceil \log_2 n^2 \rceil$ und das zweite aus $|b| = \lceil \log_2 n \rceil$ Qubits besteht. Alle Qubits werden mit $|0\rangle$ initialisiert. Im ersten Schritt wird eine größtmögliche Superposition des ersten Teilregisters $|a\rangle$ durch Anwenden des Hadamard-Gatters auf alle Qubits von $|a\rangle$ erzeugt. Anschließend wird das U_f-Gatter auf $|a\rangle$ und $|b\rangle$ angewandt. Dabei werden alle Funktionswerte von $a^x \bmod n$ gleichzeitig durch Parallelisierung berechnet. Die erhaltenen Zustände des $|a\rangle$-Registers werden durch die QFT in eine andere Basis transformiert, sodass eine anschließende Messung von $|a\rangle$ die Bestimmung der Periode von f erlaubt.

Die genaue Funktionsweise des Algorithmus soll exemplarisch für das obige Beispiel $n = 15$ und $a = 7$ dargelegt werden. Das Register $|a\rangle$ sollte aus $|a| = \lceil \log_2 15^2 \rceil = 8$ Qubits, $|b\rangle$ aus $|b| = 4$ Qubits bestehen. Zur besseren Übersichtlichkeit nehmen wir in diesem Beispiel $|a| = 4$ an, was lediglich zu einer Vergrößerung der Fehlerwahrscheinlichkeit führt, die Funktionsweise aber nicht verändert. Der Anfangszustand der beiden Register ist demnach $|a\rangle|b\rangle = |0000\rangle|0000\rangle = |0\rangle|0\rangle$. Die Anwendung von vier Hadamard-Gattern auf Register $|a\rangle$ ergibt

$$|a\rangle = \frac{1}{\sqrt{2^4}}(|0\rangle + |1\rangle + |2\rangle + \cdots + |15\rangle), \quad |b\rangle = |0\rangle.$$

Nun wird $U_f : |a\rangle|b\rangle \to |a\rangle|b\rangle \oplus f(a)$ angewandt und das Register in den Zustand

$$|a\rangle|b\rangle = \frac{1}{4}\sum_{x=0}^{15} |x\rangle|7^x \bmod 15\rangle = \frac{1}{4}(|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle + |4\rangle|1\rangle + |5\rangle|7\rangle + \cdots)$$

versetzt (vgl. Tabelle oben). Nun wird Register $|b\rangle$ gemessen, sodass $|a\rangle$ in einen der Zustände

$$\frac{1}{2}(|0\rangle + |4\rangle + |8\rangle + |12\rangle),\ \frac{1}{2}(|1\rangle + |5\rangle + |9\rangle + |13\rangle),\ \frac{1}{2}(|2\rangle + |6\rangle + |10\rangle + |14\rangle),\ \frac{1}{2}(|3\rangle + |7\rangle + |11\rangle + |15\rangle)$$

geht (die Superposition der Qubits im $|b\rangle$-Register wird zerstört, $|b\rangle \in \{|1\rangle, |7\rangle, |4\rangle, |13\rangle\}$). Könnte man die Superposition von $|a\rangle$ als Ergebnis ausgeben, so könnte man direkt die Periode r ablesen. Leider zerstört eine Messung jedoch die Superposition und man erhält nur einen Summand als Messergebnis, z.B. $|0\rangle$ oder

$|11\rangle$, woraus man die Periode nicht ablesen kann. Die Lösung dieses Problems liefert die oben beschriebene Fouriertransformation, die auf $|a\rangle$ angewandt zu einem der Zustände

$$\frac{1}{2}(|0\rangle+|4\rangle+|8\rangle+|12\rangle),\ \frac{1}{2}(|0\rangle+i|4\rangle-|8\rangle-i|12\rangle),\ \frac{1}{2}(|0\rangle-|4\rangle+|8\rangle-|12\rangle),\ \frac{1}{2}(|0\rangle-i|4\rangle-|8\rangle+i|12\rangle)$$

führt.

Nun wird $|a\rangle$ gemessen, sodass wir eine ganze Zahl $y \in \{0,\ldots,15\}$ als Ergebnis erhalten. Der Wert von y gibt noch nicht direkt die Periode r an. r lässt sich jedoch mit klassischen Rechnern effizient aus y bestimmen, denn y ist entweder ein Vielfaches von n/r oder liegt so nah bei einem Vielfachen von n/r, dass mit hoher Wahrscheinlichkeit durch das Verfahren der wiederholten Kettenbrüche (vgl. z.B. [11], [13]) die Periode r bestimmt werden kann. (Im Beispiel oben liegen die möglichen Werte von y nahe bei den Vielfachen von $n/r = 15/4$, sodass man mit wiederholten Kettenbrüchen $r = 4$ berechnen kann.)

Die *Simulation* des Shor-Algorithmus konzentriert sich auf den nicht-klassischen Teil der Periodenbestimmung. Alle benötigten Gatter (Hadamard-, U_f-Gatter und QFT) wurden bereits oben behandelt und können direkt simuliert werden. Durch die Visualisierung der Phasen der Basiszustände erkennt man in der Simulation die Periode der Funktion f bereits nach dem U_f-Gatter. Wie bereits oben erklärt lässt sich durch Messung des Registers nach dem U_f-Gatter die Periode jedoch nicht bestimmen, da die Superposition zerstört wird. Durch Anwenden der QFT in der Simulation erkennt man sehr gut, dass die QFT den störenden Offset der Superposition[8] beseitigt. Die Erklärung für das Beseitigen des störenden Offsets durch die QFT ist wie folgt: Ein Zustand $|\hat{x}+j\cdot r\rangle$ mit störendem Offset $\hat{x}$, der die Periode r „versteckt“, ($j \in \mathbb{N}$) wird durch die QFT nach Gleichung (3.2) zu

$$\frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{2\pi i/N\cdot(\hat{x}+j\cdot r)y}\,|y\rangle\,,$$

wobei $N = 2^{|a|}$ die Anzahl der Basiszustände des $|a\rangle$-Registers ist, auf das die QFT angewandt wird. Wendet man die QFT auf eine Superposition $\frac{1}{\sqrt{A}}(|\hat{x}\rangle+|\hat{x}+r\rangle+\cdots+|\hat{x}+(A-1)r\rangle)$ von A Zuständen der Form[9] $|\hat{x}+j\cdot r\rangle$ ($j \in \{0,1,\ldots,A-1\}$) an, so erhält man also

$$\frac{1}{\sqrt{A\cdot N}}\sum_{y=0}^{N-1}\sum_{j=0}^{A-1} e^{2\pi i/N\cdot(\hat{x}+j\cdot r)y}\,|y\rangle = \frac{1}{\sqrt{A\cdot N}}\sum_{y=0}^{N-1} e^{2\pi i/N\cdot\hat{x}\cdot y}\sum_{j=0}^{A-1} e^{2\pi i/N\cdot j\cdot r\cdot y}\,|y\rangle\,.$$

Die Wahrscheinlichkeit den Zustand $|y\rangle$ zu messen ist somit

$$\frac{1}{NA}\left|e^{2\pi i/N\cdot\hat{x}\cdot y}\sum_{j=0}^{A-1} e^{2\pi i/N\cdot j\cdot r\cdot y}\right|^2 = \frac{1}{NA}\left|\sum_{j=0}^{A-1} e^{2\pi i/N\cdot j\cdot r\cdot y}\right|^2,$$

da $\left|e^{2\pi i/N\cdot\hat{x}\cdot y}\right| = 1$ gilt. Das bedeutet, dass nach der QFT der störende Offset $\hat{x}$ keinen Einfluss auf die Wahrscheinlichkeitsverteilung der messbaren Zustände $|y\rangle$ hat! Die Wahrscheinlichkeiten für $|y\rangle$ hängen nur von der Anzahl der Basiszustände N und von der Periode r ab. Die der QFT vom Offset $\hat{x}$ wird in der Simulation durch die Rotation der jeweiligen Basisvektoren sehr gut veranschaulicht. Die Analogie zur klassischen Fouriertransformation wird hier deutlich, da sie angewandt auf eine periodische Funktion als Ausgabe ebenfalls die Periode (bzw. Frequenz) dieser Funktion liefert.

3.7 Suchalgorithmus von Grover

Der *Suchalgorithmus von Grover* ermöglicht es mit nicht-exponentiellem Aufwand nach einem bestimmten Eintrag in einer Datenbank zu suchen, z.B. den Namen zu einer Telefonnummer in einem Telefonbuch. Da diese Suche wiederum ein NP-vollständiges Problem ist, ist auch dieser Algorithmus effizienter als jeder klassische Algorithmus. Als Input dient eine Funktion $f : \{0,\ldots,N-1\}$, die für den gesuchten Wert x^* $f(x^*) = 1$ ist und für alle anderen Werte 0. Dies entspricht einer Datenbank mit N Einträgen von denen genau einer gesucht ist. Ziel des Algorithmus ist es den Wert des gesuchten Datenbankeintrags x^* auszugeben.

Das Register für den Algorithmus besteht aus $n+1$ Qubits, wenn die Größe der Datenbank als Zweierpotenz $N = 2^n$ gewählt wird. Das Register ist aufgeteilt in ein Register $|a\rangle$ mit n Qubits und Anfangszustand $|0\rangle$ und ein Register $|b\rangle$ mit einem Qubit und Anfangszustand $|1\rangle$. Zunächst wird das Register $|a\rangle$ in eine Superposition aller Datenbankeinträge gebracht, d.h. man wendet nacheinander das Hadamard-Gatter auf alle Qubits des $|a\rangle$-Registers an. Anschließend wendet man das Hadamard-Gatter auch auf $|b\rangle$ an. Nach Anwendung des U_f-Gatters auf diesen Zustand erkennt man an der Superposition den Wert von x^*, da diese Amplitude als einzige einen Vorzeichenwechsel vollzieht. Durch Messung wird diese Superposition allerdings zerstört, sodass

[8] Gemeint ist, dass z.B. $\frac{1}{2}(|2\rangle+|6\rangle+|10\rangle+|14\rangle)$ um einen störenden Offset von 2 verschoben ist. Schafft man es diesen Offset zu beseitigen, d.h. den Zustand in eine Superposition von $|2-2=0\rangle, |6-2=4\rangle, |10-2=8\rangle$ und $|14-2=12\rangle$ zu überführen, so ist es möglich dem gemessenen Zustand die Periode abzulesen.

[9] Die Zustände sind jeweils modulo n zu betrachten, d.h. $|(\hat{x}+j\cdot r)\bmod n\rangle$. Zur besseren Übersichtlichkeit wird dies im Text nicht explizit notiert.

der Zustand vor der Messung so transformiert werden muss, dass aus einer Messung x^* ablesbar ist. Beim Deutsch-Algorithmus wurden zum gleichen Zweck Hadamard-Gatter verwendet, beim Shor-Algorithmus die QFT. Beim Suchalgorithmus von Grover spiegelt man die Amplituden an ihrem Mittelwert, um die Amplitude des Zustands $|x^*\rangle$ mit dem negativen Vorzeichen zu vergrößern und alle anderen Amplituden zu verkleinern. Seien z.B. $x^* = 3$, die Amplitude von $|3\rangle$ gleich $-\frac{1}{2}$, die Amplituden von $|0\rangle, |1\rangle$ und $|2\rangle$ gleich $+\frac{1}{2}$. Der Mittelwert der Amplituden ist dann $\frac{1}{4}$. Spiegelung an diesem Wert überführt $+\frac{1}{2}$ zu 0 und $-\frac{1}{2}$ zu 1, sodass man $|x^*\rangle$ messen kann.

Die Funktionsweise dieses Algorithmus ist beispielsweise in [10], oder [7] genauer beschrieben. Durch die Simulation der Gatter und Register kann die Funktionsweise des Algorithmus gut simuliert werden.

4 Experimentelle Umsetzungen mittels Quantenhardware

Im Jahr 1949 prognostizierte das Magazin *Popular Mechanics* „Die Computer der Zukunft könnten weniger als anderthalb Tonnen wiegen." Die große Fülle an Ansätzen einen Quantencomputer experimentell aufzubauen und die Tatsache, dass bereits Algorithmen mit bis zu 8 Qubits aufgebaut und getetestet werden konnten, lässt hoffen, dass in den nächsten Jahrzehnten auf dem Gebiet der Quantenhardware ein ähnlicher Fortschritt erzielt wird wie in den vergangenen Jahrzenten beim klassischen PC. Im Folgenden soll ein Ansatz zur experimentellen Umsetzung von Quantencomputern mittels linearer Optik vorgestellt werden, der sich verhältnismäßig leicht aufbauen lässt und in dem die Funktionsweise von Quantenalgorithmen sehr gut physikalisch erklärbar ist. Zuvor ist es jedoch zweckmäßig die Anforderungen an einen Quantencomputer kurz zu nennen.

4.1 Anforderungen an die experimentelle Umsetzung von Quantencomputern

Notwendige Eigenschaften eines experimentell umgesetzten Quantencomputers [8]: Ein Quantencomputer besteht aus mehreren Qubits,

1. die in einen Anfangszustand versetzt werden können,
2. die Information robust speichern (d.h. in einem zeitlich stabilen Zustand),
3. auf die (universelle) Quantengatter anwendbar sind,
4. die gemessen werden können.

4.2 Optische Repräsentation von Gattern

Qubits und Gatter lassen sich optisch repräsentieren, indem man Qubits durch Photoneneigenschaften darstellt (z.B. Polarisation oder Ort) und Gatter durch lineare optische Geräte, z.B. Strahlteiler, Phasenschieber, Spiegel (vgl. insbesondere [2]).

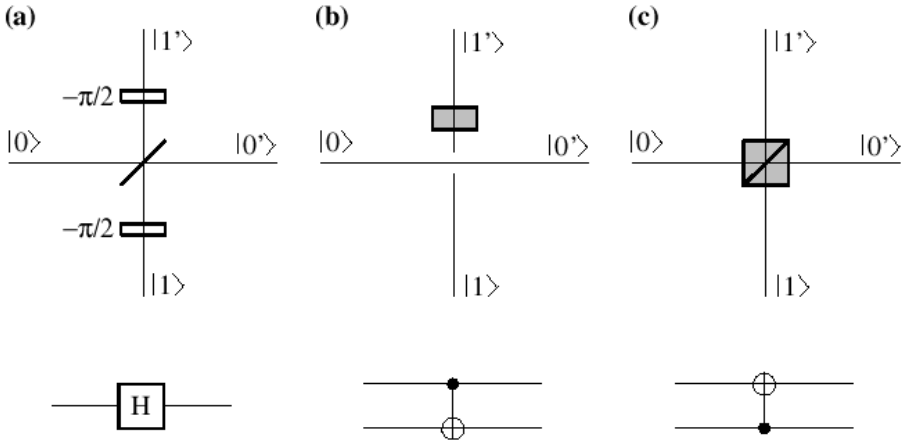


Abb. 7: Repräsentation von Qubits und Gatter durch Einzelphoton und lineare optische Geräte nach [2].

In Abb. 7 wird die optische Repräsentation gezeigt. In Bild a) wird ein Hadamard-Gatter durch zwei Phasenverschieber um $-\pi/2$ und einen 50-50-Strahlteiler umgesetzt. Das Input-Photon kann von links (Qubit in Zustand $|0\rangle$) oder von unten (Qubit in Zustand $|1\rangle$) oder in einer Superposition beider Wege auf den Strahlteiler treffen. Dort wird es reflektiert und gespiegelt und befindet sich nach Verlassen des Strahlteilers in den beiden Output-Wegen $|0'\rangle$ und $|1'\rangle$. Diese Wege unterscheiden sich durch einen zusätzlichen Phasenunterschied von $\pi/2$ (vgl. z.B. [14]). Für einen beliebigen Input-Zustand $\alpha|0\rangle + \beta|1\rangle$ ist der Output-Zustand gegeben durch $(\alpha+\beta)|0'\rangle + (\alpha-\beta)|1'\rangle$, da in der komplexen Zahlenebene ein Phasensprung um φ durch Multiplikation mit $e^{i\varphi}$ dargestellt werden kann, und deshalb einem Phasensprung von π die Multiplikation der Amplitude mit $e^{i\pi} = -1$ entspricht. Diese Operation entspricht exakt der eines Hadamard-Gatters, siehe oben.

In Abb. b) ist ein 2-Bit-Gatter, das CNOT-Gatter abgebildet. Der graue Kasten entspricht dabei einer Drehung der Polarisation um 90°. Ein Bit wird durch „welcher Weg“ dargestellt (nach oben 1, nach rechts 0), das zweite Bit durch Polarisation des Photons (horizontal 0, vertikal 1). Wenn das Polarisations-Bit beliebig ist, so wird es genau dann geshiftet (von 0 zu 1 oder von 1 zu 0), wenn das Welcher-Weg-Bit 1 ist (das Photon also nach oben geht), d.h. wenn das Controll-Bit 1 ist. Dies entspricht also einem Controlled-Not-Gatter.

In Bild c) befindet sich in der Mitte ein Strahlteiler, der vertikal polarisierte Photonen reflektiert, horizontal polarisierte jedoch transmittiert. Genau dann, wenn das Polarisations-Bit 1 ist (vertikal polarisiert), werden die Wege des Photons ausgetauscht, d.h. 0 wird zu 1 und 1 wird zu 0. Dies entspricht wiederum einem CNOT Gatter, hier allerdings mit dem Polarisations-Bit als Controll-Bit.

Die oben genannten Anforderungen werden alle recht gut erfüllt: Die Qubits sind leicht in einen klassischen Anfangszustand zu setzen (Quelle des Photons von links bedeutet Welcher-Weg-Qubit ist 0). Durch Strahlteiler mit unterschiedlicher Reflektivitätsrate und Phasenschieber können beliebige Superpositionen eines Bits hergestellt werden und als Anfangszustand für eine weitere Berechnung verwendet werden. Die Information der Qubits geht nicht verloren, kann also robust gespeichert werden, wenn die Absorptionsraten der optischen Geräte sehr klein gehalten wird. Die universellen Gatter sind alle anwendbar. Eine Messung der Qubits kann durch einen Photonendetektor erfolgen.

4.3 Optische Umsetzung des Deutsch-Algorithmus

Verwendet man die oben beschriebene Repräsentation von Gattern und Qubits durch optische Geräte, so ist es möglich den Deutsch-Algorithmus durch einen optischen Aufbau zu repräsentieren, siehe Abb. 8. Der

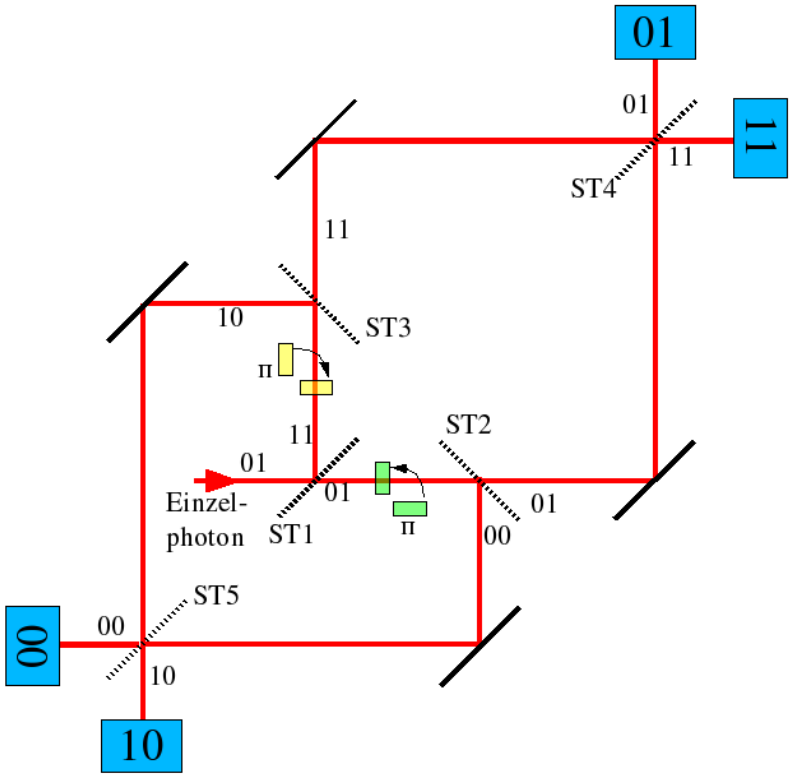


Abb. 8: Repräsentation des Deutsch-Algorithmus mit Strahlteilern (gestrichelt), Phasenverschiebern und Spiegeln.

Anfangszustand $|01\rangle$ wird durch ein von links kommendes Photon repräsentiert. Das erste Hadamard-Gatter wird durch einen Strahlteiler (ST1) umgesetzt, der das Photon in eine Superposition der beiden Wege $|01\rangle$ und $|11\rangle$ bringt. Auf diese Superposition wird noch ein Hadamard-Gatter angewandt, indem in beide Wege erneut ein Strahlteiler gesetzt wird (ST2 und ST3). Das Photon befindet sich in einer Superposition der Wege $|01\rangle, |00\rangle, |11\rangle$ und $|10\rangle$. Nun wird das U_f-Gatter angewandt. Wie oben gezeigt wurde und in den Screenshots im Anhang deutlich wird, entspricht die Anwendung von U_f einem Vorzeichenwechsel bestimmter Amplituden, der durch einklappbare π- bzw. $\lambda/2$-Plättchen (grün bzw. gelb in Abb. 8) realisiert werden kann. Im Fall von $f(0) = f(1) = 0$ sind beide Plättchen nicht im Strahlengang, also ausgeklappt. Im Fall von $f(0) = f(1) = 1$ sind beide Plättchen eingeklappt, da alle vier Amplituden ihr Vorzeichen wechseln. Im Fall $f(0) = 0$, $f(1) = 1$ vertauschen $|01\rangle$ und $|00\rangle$ ihr Vorzeichen, d.h. nur das grüne Plättchen ist eingeklappt. Im Fall $f(0) = 1$, $f(1) = 0$ vertauschen $|10\rangle$ und $|11\rangle$ ihr Vorzeichen, d.h. nur das gelbe Plättchen ist eingeklappt. In den ersten beiden Fällen (f konstant) wird zu beiden Wegen die gleiche Phasendifferenz hinzugefügt. In den letzten beiden Fällen (f balanciert) wird jedoch eine Phasendifferenz von π hinzugefügt, die das Interfe-

renzbild vertauscht, d.h. destruktive Interferenz wird zu konstruktiver und umgekehrt. Zuletzt wird das letzte Hadamard-Gatter auf das erste Qubit durch zwei Strahlteiler ST4 und ST5 angewandt.

Nun kann gemessen werden an welchem Ausgang das Photon austritt und hieraus auf die Echtheit der Münze bzw. $f(0) \oplus f(1)$ geschlossen werden. Stellt man die Weglängen der einzelnen Arme nämlich so ein, dass ohne eingeklappte Plättchen an den Ausgängen $|00\rangle$ und $|01\rangle$ konstruktive Interferenz (Wahrscheinlichkeit für Photonankunft jeweils 50%) und an den Ausgängen $|10\rangle$ und $|11\rangle$ destruktive Interferenz auftritt (Wahrscheinlichkeit für Photon jeweils 0%), so misst man bei konstantem f für das erste Qubit den Wert 0 ($|01\rangle$ oder $|00\rangle$). Klappt man nun genau ein Plättchen ein (d.h. f balanciert), so ist die Wahrscheinlichkeit für das Photon die Ausgänge $|01\rangle$ oder $|00\rangle$ zu erreichen jeweils 0, die Wahrscheinlichkeit $|10\rangle$ oder $|11\rangle$ zu erreichen aber jeweils 50%, d.h. eine Messung des ersten Qubits ergibt mit Sicherheit den Wert 1. So kann entschieden werden, ob f balanciert oder konstant ist.

Da die Erzeugung von Einzelphotonen sehr schwierig ist, wird es voraussichtlich nur möglich sein den Deutsch-Algorithmus zu *simulieren*, z.B. mit einem Laser. Das Messergebnis mit einem Laser wird allerdings wegen analoger Interferenz wie mit Einzelphotonen das gleiche sein, sodass die Funktionsweise des Deutsch-Algorithmus experimentell gezeigt werden kann.

5 Ausblick und Danksagung

Zur Zeit konzentriert sich die Arbeit auf die Implementierung der beschriebenen Simulation eines Quantencomputers und der dazugehörenden Visualisierung der Abläufe in einem Quantencomputer. Nach den in der Arbeit beschriebenen Algorithmen soll der Suchalgorithmus von Grover genauer untersucht und implementiert werden. Dann ist eine Untersuchung NP-vollständiger Probleme (siehe v.a. [10], z.B. Problem der kürzesten Rundreise) sicherlich sehr interessant, da Quantenalgorithmen auf diesem Gebiet enorme Vorteile gegenüber klassischen Algorithmen vorweisen können. Ebenso könnte auf die Quantenkryptographie (ebenfalls [10]), die durch den Shor-Algorithmus bereits gestreift wird, genauer eingegangen werden.

Die Visualisierung von Vektoren im Hilbert-Raum durch Bloch-Kugeln in dem Programm ist geplant, die Dreidimensionalität stellt allerdings implementiertechnische Schwierigkeiten dar. Zudem soll die optische Umsetzung des Deutsch-Algorithmus genauer untersucht und möglicherweise im Experiment umgesetzt werden.

Unser Dank gilt allen, die uns bei diesem Projekt unterstützt haben. Ganz besonders sind wir Christian Hoffmann für die aufschlussreichen Erklärungen und interessanten Diskussionen dankbar. Ebenso möchten wir uns bei Wolfgang Kinzel und Peter Krahmer für das durch sie geweckte Interesse für Quantenmechanik bedanken.

Literatur

[1] ADRIANO BARENCO, CHARLES H. BENNET, PETER SHOR ET AL.: *Elementary Gates for quantum computation*. Physical Review A, March 22, 1995 (AC5710).

[2] C. ADAMI, N. J. CERF: *Quantum Computation With Linear Optics*. arXiv:quant-ph/9806048[1], 1998.

[3] CLEVE, R.: *Quantum Algorithms Revisited*. arXiv:quant-ph/9708016, 1997.

[4] COHEN, DAVID W.: *An Introduction to Hilbert Space and Quantum Logic*. New York, 1989.

[5] CRISTIAN S. CALUDE, GHEORGHE PAUN: *Computing with cells and atoms*. London, 2000.

[6] DIRAC, PAUL A. M.: *The Principles of Quantum Mechanics*. Oxford, 1988.

[7] GRUSKA, JOZEF: *Quantum Computing*. London, 1999.

[8] HOMEISTER, MATTHIAS: *Quantum Computing verstehen*. Wiesbaden, 2005.

[9] JOZSA, RICHARD: *Quantum Algorithms and the Fourier Transform*. arXiv:quant-ph/9707033, 1997.

[10] NIELSEN, MICHAEL A.: *Quantum computation and quantum information*. Cambridge, 2000.

[11] PATRICK KILIAN, ALEX IVASCENKO: *Faktorisierung: Quanten machen's leichter (Seminar Quanteninformation)*. `http://theorie.physik.uni-wuerzburg.de/~hinrichsen/Lehre/Vorlesungen/Quanteninfo/Seminar/shor.pdf`, 2006.

[12] SCHMITTFULL, MARCEL: *JavaPsi Quantenmechanik Simulation*. `htt://javapsi.sf.net`, 2003.

[13] SHOR, PETER: *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. arXiv:quant-ph/9508027, 1995.

[14] ZEILINGER, ANTON: *General properties of lossless beamsplitters in interferometry*. `http://www.quantum.univie.ac.at/publications/pdffiles/1981-03.pdf`, 1981.

[1] Siehe `http://www.arxiv.org/quant-ph/9806048`, URLs für die restlichen arXiv-Artikel werden analog gebildet: „`http://www.arxiv.org/`“ + „`quant-ph/...`“.

A Theoretische Grundlagen

A.1 Quantenmechanik Grundlagen

Zunächst soll eine kurze Einführung in die Grundlagen der Quantenmechanik gegeben werden. Weil die klassische Mechanik in der mikroskopischen Welt, d.h. im Bereich der Größe von Elektronen, versagt, wurde die Quantenmechanik Anfang des 20. Jahrhunderts zu dem Zweck entwickelt, auch das Verhalten sehr kleiner Teilchen, wie beispielsweise Elektronen oder Photonen, möglichst exakt zu beschreiben.

Einige wichtige Prinzipien der Quantenmechanik, die für die weiter unten folgende Erklärung der Funktionsweise eines Quantencomputers benötigt werden, sollen in folgender Liste kurz aufgeführt werden.

(a) Als *Zustand* $|\psi\rangle$ bezeichnet man in der Quantenmechanik die vollständige Beschreibung eines Quantensystems bzgl. bestimmter physikalischer Größen, z.B. Ort oder Impuls.

(b) Unter einem *Ereignis* versteht man ein Paar (Endzustand, Anfangszustand).

(c) Die *Wahrscheinlichkeit p*, dass ein Ereignis E eintritt, wird durch $p = |\alpha|^2$ ausgedrückt, wobei $\alpha \in \mathbb{C}$ *Wahrscheinlichkeitsamplitude*, oder einfach *Amplitude*, des Ereignisses E genannt wird.

(d) Die Amplitude eines Ereignisses E mit dem Anfangszustand AZ und dem Endzustand EZ wird in der DIRACschen Bra-Ket Notation [6] durch $\alpha_E = \langle EZ|AZ\rangle$ dargestellt.

Diese Teilaspekte der Quantenmechanik reichen bereits aus, um die Funktionsweise von Quantencomputern und Quantenalgorithmen zu erklären. (Für detailliertere Quantenmechanik Abhandlungen sei auf Lehrbücher oder auch [12] verwiesen.)

A.2 Mathematische Darstellung

Ein grundlegendes Prinzip der Quantenmechanik besagt, dass die möglichen Zustände jedes abgeschlossenes Quantensystem, z.B. auch eines Quantencomputers, durch Elemente eines *Hilbert-Raums* $\mathcal{H}$ dargestellt werden können [6]. Im Folgenden soll deshalb auf die Definition und einige wichtige Eigenschaften von Hilbert Räumen eingegangen werden.[1] (Vgl. z.B. [4] S.14ff.)

Definition A.1 Das *Skalarprodukt* ist eine Funktion $\langle\cdot|\cdot\rangle : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{C}$, definiert durch

$$\langle x|y\rangle := \sum_{i=0}^{n} x_i^* y_i,$$

wenn $x = (x_1, \ldots, x_n) \in \mathbb{C}^n$ und $y = (y_1, \ldots, y_n) \in \mathbb{C}^n$.

Es lassen sich folgende Eigenschaften für $x, y, z \in \mathbb{C}^n$, $\lambda \in \mathbb{C}$ zeigen.

(a) $\langle x|y\rangle = \langle y|x\rangle^*$ (* ist komplex konjugiert),

(b) $\langle x+y|z\rangle = \langle x|z\rangle + \langle y|z\rangle$ und $\langle x|y+z\rangle = \langle x|y\rangle + \langle x|z\rangle$,

(c) $\langle \lambda x|y\rangle = \lambda^* \langle x|y\rangle = \langle x|\lambda y\rangle$,

(d) $\langle x|x\rangle \geq 0$ und $\langle x|x\rangle = 0$ genau dann, wenn $x = \mathbf{0}$.

In der Quantenmechanik ist $\langle x|y\rangle \in \mathbb{C}$ die Amplitude für ein Ereignis, das das Quantensystem vom Anfangszustand y in den Endzustand x überführt.

Definition A.2 $\mathcal{V} = (V, +, \cdot, \langle\cdot|\cdot\rangle)$ ist ein *Skalarprodukt-Raum*, falls $(V, +, \cdot)$ ein Vektorraum (über dem Körper $\mathbb{C}$) ist und das Skalarprodukt $\langle\cdot|\cdot\rangle$ wie in Def. A.1 definiert ist. Ein *Hilbert-Raum* $\mathcal{H}$ ist ein vollständiger[2] Skalarprodukt-Raum.

Ein Vektor $|\cdot\rangle$ eines Hilbert-Raums $\mathcal{H}$ dient der Repräsentation eines Quantenzustands. Anschaulich kann man sich für Zwecke des Quantencomputing $\mathcal{H}$ als Vektorraum $\mathbb{C}^n$ vorstellen, auf dem zusätzlich ein Skalarprodukt nach Def. A.1 definiert ist.

Definition A.3 Zwei Vektoren $|x\rangle, |y\rangle \in \mathcal{H}$ heißen *orthogonal* oder senkrecht aufeinander, falls $\langle x|y\rangle = 0$ gilt.[3] Haben $|x\rangle, |y\rangle$ zusätzlich beide die Norm 1, so heißen sie *orthonormal*. Eine Menge $\mathcal{B}$ von Vektoren eines Hilbert-Raums $\mathcal{H}$ ist eine *orthonormale Basis* von $\mathcal{H}$, wenn alle Vektoren aus $\mathcal{B}$ paarweise orthonormal sind und sich jeder Vektor $|\psi\rangle \in \mathcal{H}$ eindeutig als Linearkombination von Basisvektoren $|\psi\rangle = \sum_{|i\rangle \in \mathcal{B}} (\alpha_i \cdot |i\rangle)$ mit $\alpha_i \in \mathbb{C}$ darstellen lässt. Die Mächtigkeit n der Menge $\mathcal{B}$ heißt *Dimension* von $\mathcal{H}$ und ist die maximale Anzahl linear unabhängiger (Basis-)Zustände, die den Hilbert-Raum $\mathcal{H}$ aufspannen, der ein Quantensystem mit n unterschiedlichen messbaren (Eigen-)Zuständen repräsentieren soll.

---

[1] Die im Folgenden als Definitionen angegeben Textstellen sind nicht immer reine Definitionen, sondern enthalten zum Teil auch Erläuterungen und Folgerungen der eigentlichen Definitionen.

[2] V ist vollständig genau dann, wenn jede Cauchy-Folge in V konvergiert. Für die Konvergenz benötigt man eine Norm auf V, hier: $|||x\rangle|| := \sqrt{\langle x|x\rangle}$.

[3] Orthogonalität ist deshalb von großer Bedeutung für Quantencomputing, weil bei jeder Messung eines Zustands dieser Zustand in einen der zueinander orthogonalen Basiszustände (Eigenzustände) „springt“ und somit die Information über andere Basiszustände verloren geht. Ziel eines Quantenalgorithmus ist es die Output-Zustände so zu konstruieren, dass durch Messung solcher Output-Zustände das Ergebnis abgelesen werden kann, d.h. dass die Information eines einzelnen Basiszustands ausreicht.

Betrachten wir zur Veranschaulichung ein System mit $n = 3$ messbaren Größen. Dann ist die Dimension von $\mathcal{H}$ 3, d.h. $\mathcal{H}_3$ kann für Quantencomputingzwecke als $\mathbb{C}^3$ aufgefasst werden. Eine einfache orthonormale Basis von $\mathcal{H}_3$ ist dann die Standardbasis des $\mathbb{C}^3$ $\{|0\rangle, |1\rangle, |2\rangle\} = \{(1,0,0)^T, (0,1,0)^T, (0,0,1)^T\}$. Ein beliebiger Zustand $|\psi\rangle \in \mathcal{H}_3$ kann als Linearkombination von Basisvektoren $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \alpha_2 |2\rangle$ dargestellt werden. Die komplexen Amplituden α_i sind die Amplituden der ortogonalen Basiszustände $|i\rangle$. Da $|\alpha_i|^2 = \alpha_i^* \alpha_i$ die Wahrscheinlichkeit für das Messen des Basiszustands $|i\rangle$ ist, müssen die α_i der Normierung $\sum_i |\alpha_i|^2 = 1$ genügen.

Zur Verdeutlichung des Orthogonalitätsbegriffs können wir in $\mathbb{C}^3$ z.B.

$$\langle 0|0\rangle = \left\langle (1,0,0)^T | (1,0,0)^T \right\rangle = 1^* \cdot 1 + 0 + 0 = 1$$

betrachten.[4] Dies kann so interpretiert werden, dass die Wahrscheinlichkeit, dass bei der Messung des reinen Basiszustands (Eigenzustands) $|0\rangle$ der Basiszustand $|0\rangle$ gemessen wird, 1 ist. Ein Basiszustand kann durch eine Messung also nur in sich selbst überführt werden und nicht in einen anderen Basiszustand. Zudem ist das Skalarprodukt zweier unterschiedlicher Basiszustände stets 0, z.B. ist $\left\langle (1,0,0)^T | (0,1,0)^T \right\rangle = 1^* \cdot 0 + 0 \cdot 1^* + 0 = 0$. Das bestätigt, dass die Wahrscheinlichkeit bei Messung eines Basiszustands $|1\rangle$ den Basiszustand $|0\rangle$ zu erhalten 0 ist. Deshalb sind zwei messbare Basiszustände immer zueinander orthogonal.

Definition A.4 Unter einem *linearen Operator* eines Hilbert-Raums $\mathcal{H}$ versteht man eine lineare Abbildung[5] $A : \mathcal{H}_n \to \mathcal{H}_n$.

Lineare Operatoren werden in der Quantenmechanik z.B. dazu benötigt, um Basistransformationen durchführen zu können, z.B. von der Orts- in die Impulsdarstellung. Beim Quantencomputing repräsentieren diese linearen Operatoren bestimmte Qubit-Gatter.

Da jede lineare Abbildung durch eine Abbildungsmatrix dargestellt werden kann, können lineare Operatoren A durch eine $n \times n$-Matrix dargestellt werden, indem jedem Input-Wert ein Output-Wert zugewiesen wird. Für Quantencomputing-Gatter ist es notwendig, dass sie durch eine sogenannte unitäre Matrix darstellbar sein müssen.

Definition A.5 Eine komplexe Matrix A ist genau dann *unitär*, falls $A^+ \cdot A = I$ gilt, wobei I die Einheitsmatrix und A^+ die adjungierte Matrix von A ist, die man durch Transponieren und Komplex-Konjugieren von A erhält.

Die Unitarität der Quantengatter-Matrizen muss gefordert werden, um für jede Qubit-Transformation die logische Reversibilität zu garantieren, d.h. die Möglicheit den Input-Zustand eindeutig aus dem Output-Zustand feststellen zu können (z.B. ist ein logisches *not* reversibel, da man das Input-Bit durch nochmalige Anwendung von *not* erhält, während ein logisches *und* nicht reversibel ist, da bei einem Ergebnis 0 die Input Bits 01, 10, oder 00 sein konnten). Ist eine Bit-Operation nicht logisch reversibel, so ist sie nach dem Prinzip von Landauer[6] auch nicht physikalisch reversibel. Da die Entropie eines geschlossenen Systems nach dem 2. Hauptsatz der Thermodynamik nicht kleiner werden kann, bedeutet physikalische Irreversibilität bzw. Informationsverlust, dass Entropie und somit Energie an die Umgebung abgegeben werden muss. Pro verlorenem Bit muss nach John von Neumann[7] eine Energie $E \geq kT \ln 2$ (k Boltzmann-Konstante, T Temperatur) an die Umgebung abgegeben werden. Während diese Energiezufuhr an die Computerhardware bei momentanen klassischen Rechnern noch nicht zu Problemen führt, werden in 10-15 Jahren wohl auch klassiche Computer reversible Operationen ausführen müssen, um eine zu große Erhitzung zu vermeiden. Quantencomputer müssen in jedem Fall reversible Operationen ausführen, da sonst nach einer bestimmten Anzahl angewandter irreversibler Gatter die Energie der Qubits (egal in welcher Darstellung, z.B. Photonen als Qubits) verloren gehen würde.[8]

[4] Natürlich kann an Stelle von $(1,0,0)$ auch jeder andere Vektor $(a,b,c)^T \in \mathbb{C}^3 \setminus \{0\}$ als erster Basisvektor verwendet werden. Da $(a,b,c)^T$ normiert ist, gilt dann analog $\left\langle (a,b,c)^T | (a,b,c)^T \right\rangle = 1$, z.B. $\left\langle (0,1/\sqrt{2},1/\sqrt{2})^T | (0,1/\sqrt{2},1/\sqrt{2})^T \right\rangle = 1/2 + 1/2 = 1$.

[5] Eine Abbildung T ist linear genau dann, wenn $T(\alpha\phi + \beta\psi) = \alpha T(\phi) + \beta T(\psi)$ gilt.

[6] Siehe z.B. `http://en.wikipedia.org/wiki/Reversible_computing`.

[7] Vgl. `http://en.wikipedia.org/wiki/Landauer%27s_Principle`.

[8] Eine genauere Diskussion dieses Themas befindet sich beispielsweise in [10] S. 153ff, [5] S. 181ff und `http://www.cs.ucf.edu/~dcm/Teaching/QuantumComputing/Fall2004Class-QC/QCV1.pdf`. Hier wird zudem gezeigt, dass durch reversible Gatter kein Effizienzverlust gegenüber irreversiblen Gattern erfolgt. Zudem wird auf das interessante Problem von „Maxwell's Dämon" eingegangen.

B Screenshots der Simulation

B.1 Deutsch-Algorithmus mit 2 Qubits

Screenshots der grafischen Ausgabe der Simulation des Deutsch-Algorithmus für $f(0) = 0$ und $f(1) = 1$ (echte Münze):

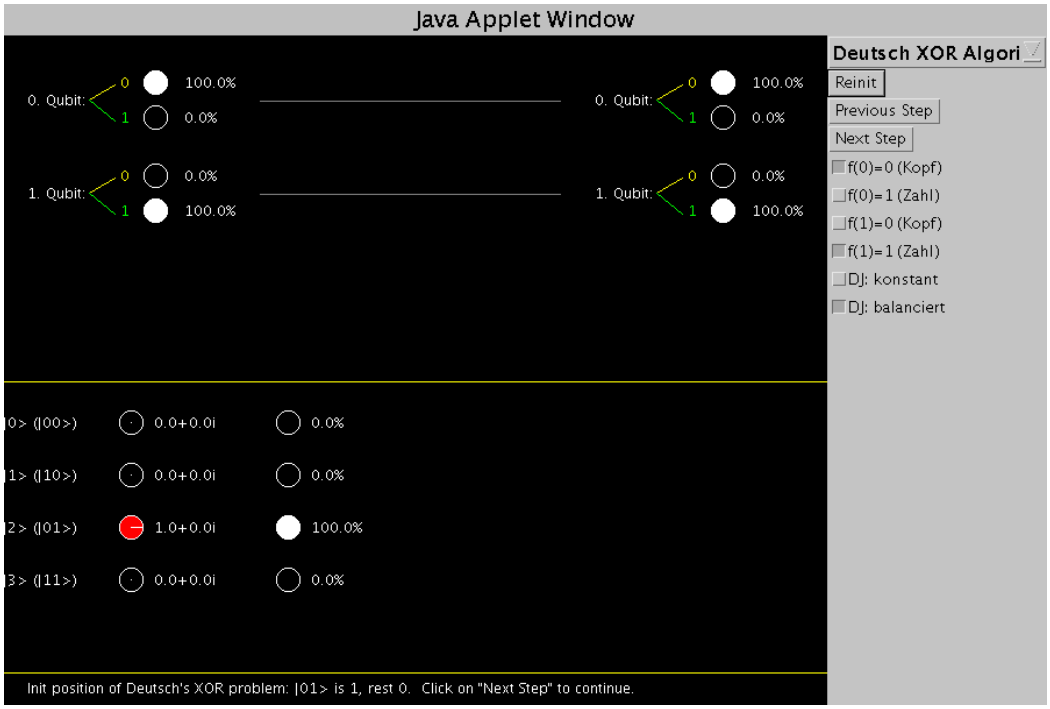


Abb. 9: Anfangszustand mit 0. Qubit auf 0 gesetzt, 1. Qubit auf 1 (d.h. Register ist im Basiszustand $|01\rangle$). Oben sind die Zustände der Qubits gezeigt, unten die Amplituden der Basiszustände mit Visualisierung der Amplituden durch Vektor, Farbe und Helligkeit.

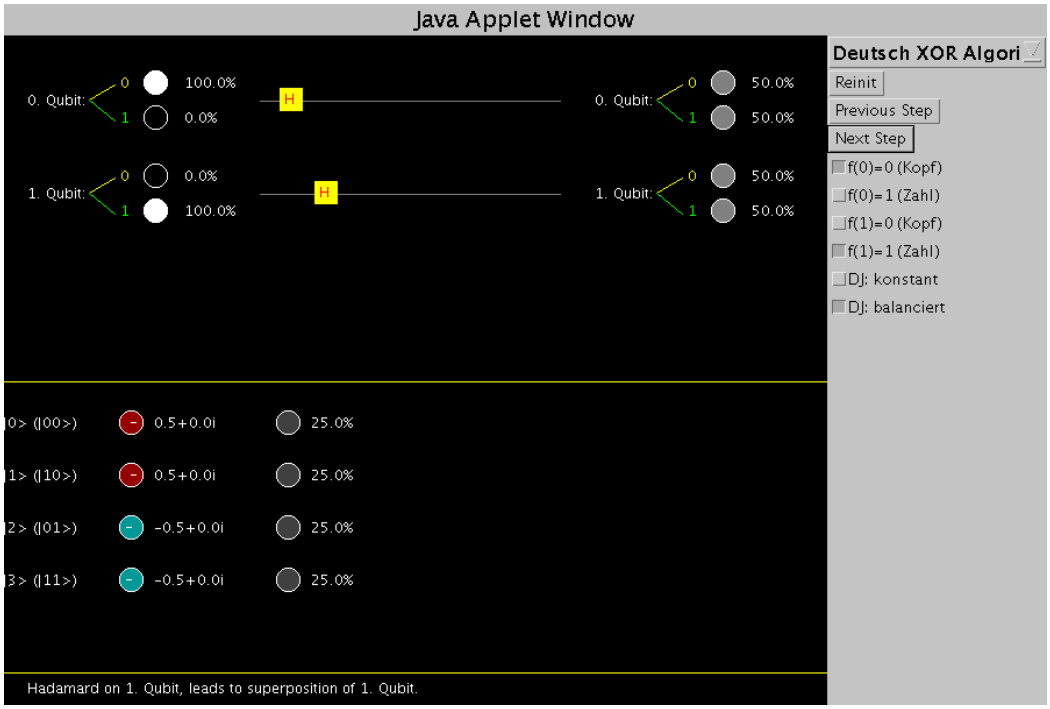


Abb. 10: Hadamard-Gatter auf beide Qubits zur Erzeugung einer gleichverteilten Superposition.

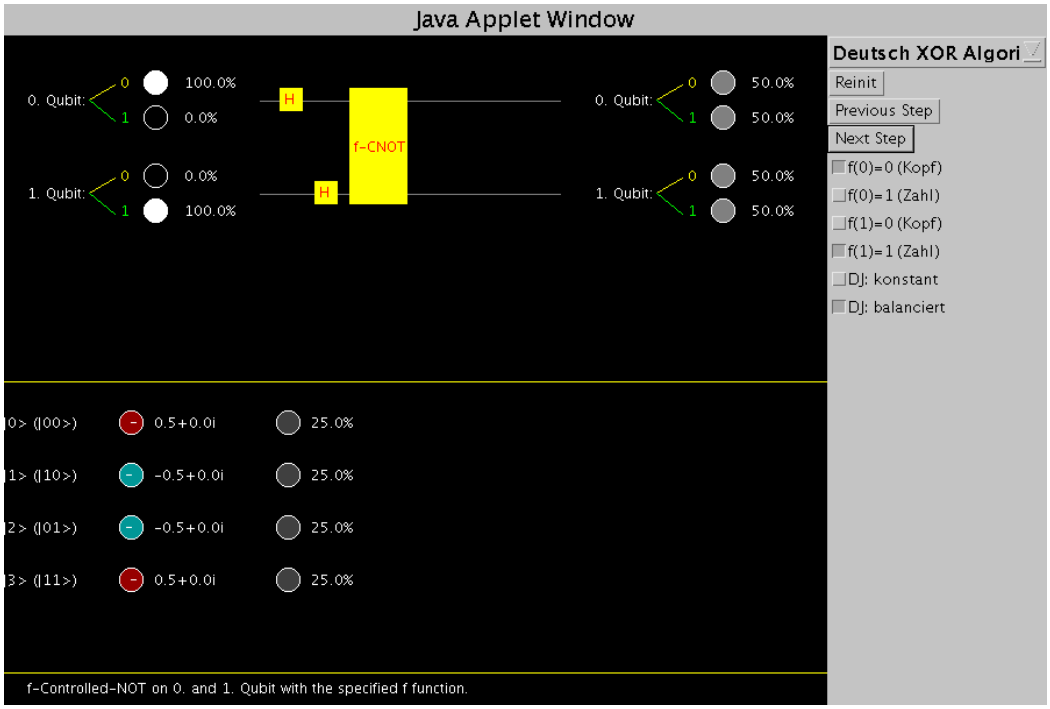


Abb. 11: U_f-Gatter zur Vertauschung der Vorzeichen der Amplituden von $|10\rangle$ und $|11\rangle$.

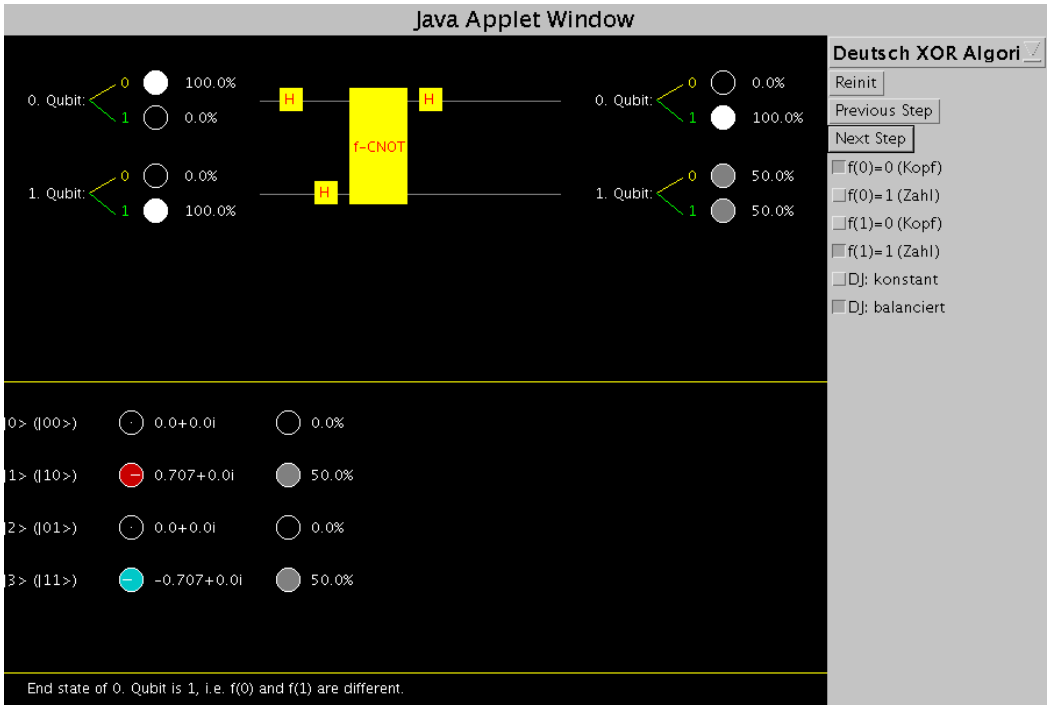


Abb. 12: Das Hadamard-Gatter hebt die Superposition des 0. Qubits wieder auf (Mischzustand wird zu Reinzustand) und führt zum Endzustand. Messung des 0. Qubits ist mit Sicherheit Zustand 1, d.h. die Münze ist echt, also $f(0) \neq f(1)$. Würde die Messung das 0. Qubits 0 ergeben, so wäre die Münze unecht, d.h. $f(0) = f(1)$.

B.2 Deutsch-Jozsa-Algorithmus mit 4 Qubits

Grafische Ausgabe der Simulation des Deutsch-Jozsa-Algorithmus für balanciertes $f(x) : \{0,1\}^4 \rightarrow \{0,1\}$ mit $n = 4$ Qubits:

Abb. 13: Anfangszustand $|0001\rangle$ des Deutsch-Jozsa-Algorithmus.

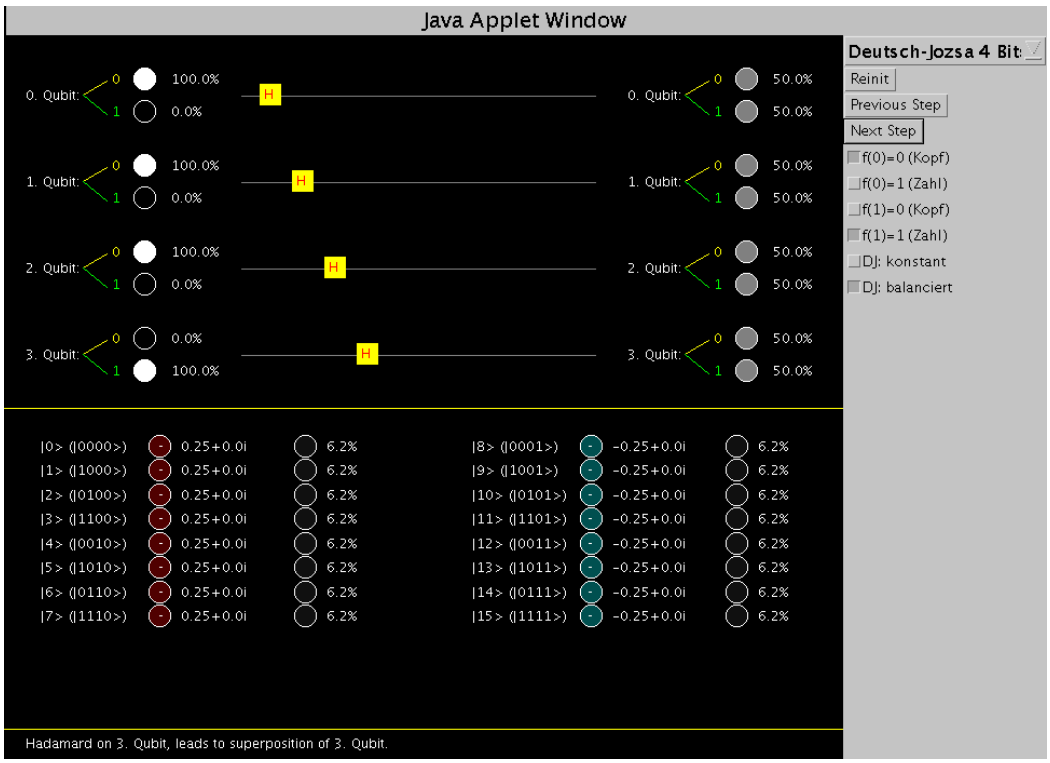


Abb. 14: Hadamard-Gatter auf alle Qubits zur Herstellung einer gleichverteilten Superposition.

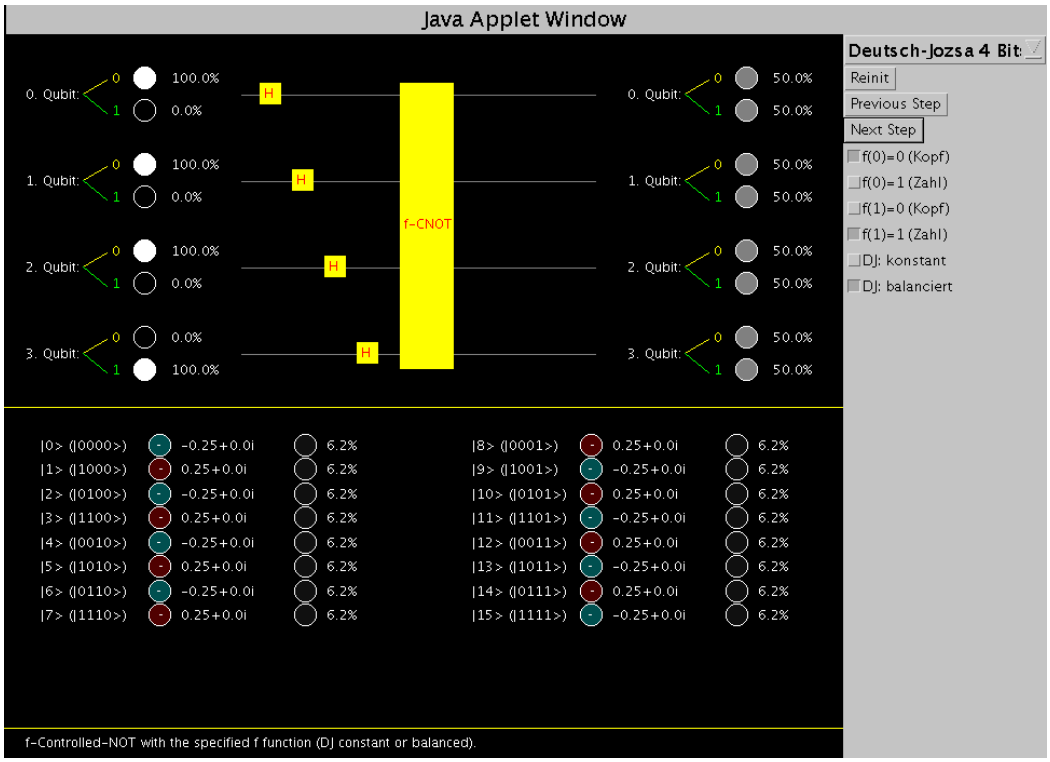


Abb. 15: U_f-Gatter zur Vertauschung der Vorzeichen bestimmter Amplituden.

Abb. 16: Hadamard-Gatter auf Qubit 0 löst die Superposition des 0. Qubits auf.

Abb. 17: Hadamard-Gatter auf Qubit 1 löst die Superposition des 1. Qubits auf.

Abb. 18: Hadamard-Gatter auf Qubit 2 führt zum Endzustand. Messung der Qubits 0 bis 2 ergibt mit Sicherheit den Basiszustand $|100\rangle$, d.h. $f(x)$ ist balanciert (wäre der Endzustand $|000\rangle$, dann wäre $f(x)$ konstant).