Quantum Computing

Daniel Wärnå

Kandidatavhandling

Datavetenskap vid Åbo Akademi

Handledare: Ion Petre

Abstrakt

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id ante in neque ullamcorper facilisis ac vel leo. Aliquam ullamcorper, felis sed tincidunt mattis, nulla nunc imperdiet ligula, nec auctor enim sapien hendrerit mauris. Curabitur diam diam, dignissim non aliquam vitae, mollis eget nibh. Vivamus dignissim vehicula velit ac varius. Nulla nec fermentum arcu. Nam adipiscing, massa scelerisque consequat vehicula, felis elit congue dolor, id vehicula lorem lorem at

TODO:

Nyckelord: kvantdator

ordningen på innehålle e ännu oklar, kom ihåg

Introduktion

1975 förutspådde Gordon E. Moore [källa] att antalet transistorer i en microprosessor kommer att dubblas ungefär vartannat år. Detta påstående gäller ännu idag. I och med att man fått plats för flera transistorer i en processor har också datorers prestanda ökat i ungefär samma takt**(källa)**.

Det finns dock problem som är väldigt svåra att lösa på datorer,

Kvantfysik är ett område inom fysiken som fokuserar sig på att beskriva subatomära partiklar. Dessa partiklars beteende skiljer sig väldigt mycket från vad vi är vana med i den makroskopiska världen.

Dagens datorer utför stuff genom att modifierar bitar som har två olika tillstånd 0 och 1. I en kvantdator utförs operationer på så kallade kvant bitar, qbitar, som kan vara i ett supervärde av 0 och 1, d.v.s. den kan anta båda värdena på samma gång. Den här egenskapen gör att vissa beräkningar (va för beräkningar)kan utföras väldigt mycket snabbare än med en konventionell dator.

I denna avhandling kommer jag att försöka ge läsaren en grundläggande bild av vad en kvantdator är, hur den fungerar och vad som kan göras med en. För att kunna presentera detta kommer jag att börja med att förklara den matematiken den matematiken som behövs till att beskriva kvantmekanik.

Presentera en algoritm, Shor's algorithm, effektiv algoritm för att utföra primtalsfaktorisering, implikationer för RSA. Nånting om kvantprogrammering om det finns utrymme. Till sist kommer jag att ta upp programmering av kvantmaskiner och vad hur läget ser ut idag gällande fullt funktionerande kvantdatorer.

introduktionen sku kunna vara lite längre

Komplexa tal

De komplexa talen står som grund för den matematiska och fysiska beskrivningen av kvantsystem.

Komplexa tal används för att hitta lösningar till funktioner som inte har reella lösningar. Ta t.ex. funktionen x^2 + 1 = 0, man kan genast se att ingen reell lösning existerar eftersom x^2 >= 0.

Ett imaginärt tal betecknas vanligtvis med *i* och är definierat enligt följande: $i^2 = -1$ eller $i = \sqrt{-1}$ Genom att kombinerar ett imaginärt tal och ett reellt tal får vi det som kallas ett komplext tal som betecknas med $\mathrm{c} = \mathrm{a} + \mathrm{b}i$ där a är den reella delen och b är den imaginära.

Komplexa tal har följande operationer:

Addition: $c_1 + c_2 = (a_1 + a_2) + i(b_1 + b_2)$
Subtraktion: $c_1 - c_2 = (a_1 - a_2) + i(b_1 - b_2)$
Multiplikation: $c_1 \times c_2 = (a_1 a_2 - b_1 b_2) + i(b_1 a_2 + b_2 a_1)$
Division: $\frac{c_1}{c_2} = \frac{(a_1 a_2 + b_1 b_2) + i(a_1 b_2 - a_2 b_1)}{a_2^2 + b_2^2}$

Absoluta beloppet: $|c| = +\sqrt{a^2 + b^2}$
Konjugat: $= a + bi => \overline{c} = a - bi$

Annars så berörs komplexa tal av samma räkne regler som reella tal, t.ex. är addition associativ och division är definierat för alla tal förutom noll. C kan alltså definieras som en kropp (*eng field*). De komplexa talen är också algebraiskt slutna d.v.s. att alla ekvationer innehållande ett värde i C har en lösning som tillhör C. (Lang, 1969)

De komplexa talen kan också tolkas grafiskt där den reella delen motsvara x axeln och den imaginära delen y-axeln. Till exempel det komplexa talet 3 + 4i kan beskrivas i ett kordinatsystem med koordinaterna (3,4) som i figur 1. Detta kallas den kartekesiska formen

figur 1

En annan metod för att beskriva ett komplext tal är den så kallade polära formen. Då beskrivs talet av en *längd* ρ som är avståndet mellan en punkt P och origo samt vinkeln θ som är vinkeln på linjen som går genom origo och P. θ beräknas med hjälp trigonometri där

$$\theta = \tan^{-1}\left(\frac{b}{a}\right)$$

och ρ beräknas med absoluta beloppet av talet. För att komma tillbaka till den kartekesiska formen används trigonometri:$a = \rho\cos(\theta) \quad b = \rho\sin(\theta)$

En tredje metod för att beskriva ett komplext tal är Euler formel (Euler, 1748)

:

$$e^{i\theta} = \cos(\theta) + i\sin(\theta)$$

Komplexa vektorrum

Ett komplext vektorrum kan definieras som set av vektorer som är slutna under operationerna; addition, negation och multiplikation av en skalär. Och det har en speciell vektor som kallas nollvektorn. Det komplexa vektorrummet betecknas med $\mathbb{C}^n$ där *n* står för vektorrummets dimension.

Tensor product

Tensor produkten är en metod för att kombinera linjära rum. Om V beskriver ett vektorrum och V' ett annat så beskriver tensor produkten bägge tillstånden. Räknas ut enligt föjande:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \otimes \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} a_0 * \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \\ a_1 * \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \\ a_2 * \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \\ a_2 b_0 \\ a_2 b_1 \end{bmatrix}$$

Hermiteska och unitära matriser

Hermiteska och unitära matriser är specialfall av en *n* gånger *n* matris som definieras med hjälp av det hermiteska konjugatet som betecknas med symbolen †. Det hermiteska konjugatet av A betecknas alltså med $A^{\dagger}$ och är en kombination av transponatet, $A^{T}[j,k] = A[k,j]$ och konjugatet $\bar{A}$ som är det komplexa konjugatet av alla element i A, $A^{\dagger}$ kan alltså definieras med $A^{\dagger}[j,k] = \overline{A[k,j]}$

En hermitesk matris kan sedan defineras som en matris vars hermiteska konjugat är lika med matriser, alltså $A = A^{\dagger}$. Och en matris är unitär om $A^{\dagger} * A = A * A^{\dagger}) = I_n$ där I_n är identitetsmatrisen.

Kvantfysik

Kort allmän introduktion om kvantfysik

Inom kvantfysiken används vanligtvis Dirac notationen, eller bra-ket notationen som den också kallas, för att beskriva ett kvantsystems läge(eng state). Dirac notationen består av en **bra** $< |$ (som representerar vårt slutgiltiga tillstånd) och en **ket** $| >$ (som beskriver vad vi vet nu). Ett bra-ket par $< \Phi | \Psi >$ beskriver sannolikheten(amplituden) av att Ψ hittas i tillstånd Φ efter en mätning.

Ett kvantsystem som beskriver en partikel på en linje kan representeras av följande ket där x_i är en position på linen.

$$|\psi> = c_0|x_0> + c_1|x_1> + \cdots + c_{n-1}|x_{n-1}>$$

Samma system kan också beskrivas med hjälp av en kolumnvektor:

$$|\psi\rangle \longmapsto [c_0, c_1, \ldots, c_{0n-1}]^T$$

Där x_i är positioner på linjen och c_i är sannolikheten eller den komplexa amplituden av att partikeln finns vid just den positionen. $|\Psi>$ Är i en superposition av de olika tillstånden x. Partikeln är alltså i en blandning av alla tillstånd x på samma gång och c_i berättar hur blandningen av tillstånd ser ut. När

vi sedan mäter $|\Psi>$ får vi resultatet att partikeln befinner sig på endast en position, detta kallas att partikeln kollapsar från en superposition till endast en entydig position. Vi kan beräkna sannolikheten av att en partikel befinner sig i ett visst tillstånd *efter* en mätning på följande sätt:

$$p(x_i) = \frac{|c_i|^2}{||\Psi\rangle|^2} = \frac{|c_i|^2}{\sum_j |c_j|^2}$$

Där $p(x_i)$ alltså är sannolikheten, $0 < p(x_i) < 1$, att vi hittar partikeln i position x_i efter att en mätning utförts.

En ket kan adderas med en annan ket och multipliceras med en komplex skalär(?).

$$2\,|\psi\rangle = [2c_0 + 2c_1 + \cdots + 2c_{n-1}]^T$$

Multiplikationen har vi nytta av i och med att en ket representerar samma fysiska tillstånd oberoende av dess längd.

$$p(x_i) = \frac{|2c_i|^2}{\sum_j |2c_j|^2} = \frac{2^2|c_i|^2}{2^2 \sum_j |c_j|^2} = \frac{|c_i|^2}{\sum_j |c_j|^2}$$

Detta möjliggör normalisering av ketar det vill säga multiplicera eller dividera en ket så att dess längd blir 1. Och som vi såg tidigare så representerar den ändå samma tillstånd. Och det simplifierar också vissa räkneoperationer, till exempel $p(x_i)$ för en normaliserad ket är då $|c_i|^2$ i och med att dess längd $\sum_j |c_j|^2 = 1$

Den andra delen av bra-ket notation, alltså bra:n är definierad på följande sätt:

$$\langle\psi| = |\psi\rangle^\dagger$$

Beteckningen $\langle\psi'|\psi\rangle^\dagger$ kan alltså skrivas om enligt följande $[\overline{c_0'}, \overline{c_1'}, \dots, \overline{c_{n-1}'}] \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix}$ och vi ser att det är samma sak som inre produkten vilket togs upp tidigare.

Mätningar

En mätning är en observation på ett fysiskt system, det kan ses som att man ställer en fråga, t.ex ”i vilken position befinner sig partikel x i” och får ett svar, till exempel ”X befinner sig i position y.

Mätningar av kvantpartiklar skiljer sig dock från mätningar av ”vanliga” fysiska objekts som vi är vana vid. Då en mätning utförs inom den klassiska fysiken kan vi anta följande

- En mätning förändra i princip inte på systemets tillstånd
- En mätning är förutsägbar, det vill säga om vi vet tillräckligt om systemet så kan vi förutse mätningens resultat.

Inom kvantfysiken gäller inget av dessa antaganden. En mätning orsakar förändringar i systemet och mätresultatet är inte deterministiskt utan snarare probabilistiskt.

Om flera mätningar utförs på systemet så spelar ordningen i vilken mätningarna utförs en roll. Exemple maybe?

Kets and bras både spin och position eller kanske bara position får om de ryms

Kvantdatorer

Bitar och Qbitar

På samma vis som bitar fungerar som kärnan i klassiska datorer fungerar kvantbitar, eller qbitar, som kärna för kvant datorer. En bit kan beskrivas som ett system med två olika tillstånd, till exempel är en knapp av eller på.

En klassisk bit betecknad med värdet 0 betecknad med bra-ket notationen set ut enligt följande

$$|0\rangle = \begin{matrix}\mathbf{0}\\\mathbf{1}\end{matrix}\begin{bmatrix}1\\0\end{bmatrix}$$

och en bit med värdet 1 betecknas med

$$|1\rangle = \begin{matrix}\mathbf{0}\\\mathbf{1}\end{matrix}\begin{bmatrix}0\\1\end{bmatrix}$$

Av detta förstås att en bit i läget 0 med 100% sannolikhet är i läget 0.

En qbit fungerar på samma sätt som en bit, den kan efter en mätning anta antingen värdet 1 eller värdet 0. Men före mätningen kan den vara i en *båda* lägena samtidigt. Den kan betecknas med

$$|\psi> = \begin{bmatrix}c_0\\c_1\end{bmatrix} = c_0 * \begin{bmatrix}1\\0\end{bmatrix} + c_1 * \begin{bmatrix}0\\1\end{bmatrix} = c_0|0\rangle + c_1|1\rangle$$

Där $|c_0|^2$ är sannolikheten att qbiten finns i läge 0 efter en mätning.

En maskin med endast en bit är inte särskilt intressant, för att utföra beräkningar behöver vi flera qbitar. En byte, alltså 8 bitar till exempel 01001101 kan beskrivas med:

$$\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}0\\1\end{bmatrix},\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}0\\1\end{bmatrix},\begin{bmatrix}0\\1\end{bmatrix},\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}0\\1\end{bmatrix}$$

För att kombinera dess bitar använder vi tensor produkten vilket ger:

$$|0\rangle\otimes|1\rangle\otimes|0\rangle\otimes|0\rangle\otimes|1\rangle\otimes|1\rangle\otimes|0\rangle\otimes|1\rangle$$

$$\begin{matrix}00000000\\00000001\\\vdots\\01101011\\\vdots\\11111110\\11111111\end{matrix}\begin{bmatrix}c_0\\c_1\\\vdots\\c_{107}\\\vdots\\c_{254}\\c_{255}\end{bmatrix}$$

En qbyte kan alltså beskrivas med $2^8 = 256$ komplexa tal. Vi ser att mängden data som krävs för att kunna beskriva en superposition växer exponentiellt.

Logiska grindar

Logiska grindar är en ”sak” som utför Boolska logiska operationer på ett värde, i klassiska datorer används dessa till att manipulera bitar. En klassisk logisk grind som opererar på ett värde kan beskrivas med matriserna:

$$(2^m gånger 2^n) * (2^n gånger 1) = (2^m gånger\ 1)$$

där den logiska grinden beskrivs av en $2^m gånger 2^n$ matris och indatan beskrivs av en kolumnvektor med n element. En NOT grind, som tar in ett värde och omvandlar en 0 till en 1 och en 1 till en 0, kan alltså beskrivas med matrisen:

$$NOT = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

en NOT grind som utför en operation på biten 0 kan då betecknas med:

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

d.v.s. NOT (0) = 1

En AND grind tar till skillnad från NOT grinden 2 stycken insignaler och ger ett resultat. En AND grind ger resultatet 1 ifall båda in värdena är 1 och resultatet 0 i andra fall. AND grinden beskrivs av en 2 gånger 4 matris i och med att den har två inputbitar.

$$AND = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

NAND grinden är viktig inom klassiska datorer i och med att den är universal, det vill säga det går att implementera alla andra logiska grindar med hjälp av ett antal NAND grindar. NAND grinden kan beskrivas med följande matris;

$$NAND = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

En NAND grind är egentligen en AND grind som efterföljs av en NOT grind. Så en NAND operation kan utföras genom att först beräkna AND och sedan NOT. Detta kan betecknas med matriser på följande sätt:

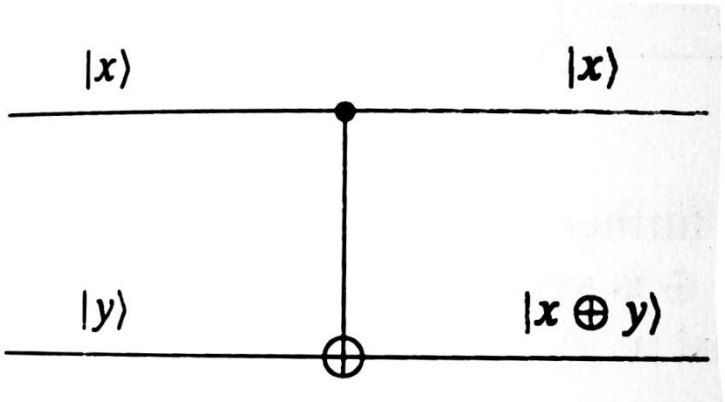
$$NOT * AND = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} = NAND$$

(Dandamudi, 2002)

Reversibla grindar

Kvantgrindar skiljer sig från klassiska logiska grindar genom att deras operationer bör vara unitära och reversibla. Jämför med en AND grind, som inte är reversibel i och med att det är omöjligt att bestämma vad invärdena var enbart på basis av resultatet. Orsaken till detta är att fysikens lagar är reversibla och en kvantdator hela tiden bör upprätthålla kvantsystemets superpositioner, till skillnad från klassiska datorer som avger värme vilket möjliggör oreversibla beräkningar. På grund av detta är klassiska beräkningar opraktiska på en kvantdator. (Shor, 2005)

Ett exempel på en reversibel grind är NOT grinden, om resultatet $|1\rangle$ vet vi att invärdet var $|0\rangle$. En variant av NOT grinden är den kontrollerade NOT-grinden



Figur 1 kontrollerad NOT grind

Den kontrollerade NOT grinden tar in två värden $|x\rangle$ och $|y\rangle$. Om $|x\rangle = 0$ kommer utsignalen at vara $|y\rangle$, ifall $|x\rangle = 1$ är utsignalen det motsatta. Grindens utvärden blir alltså $|x, x \oplus y\rangle$ där $\oplus$ är den binära *"exclusive-or"* operationen,

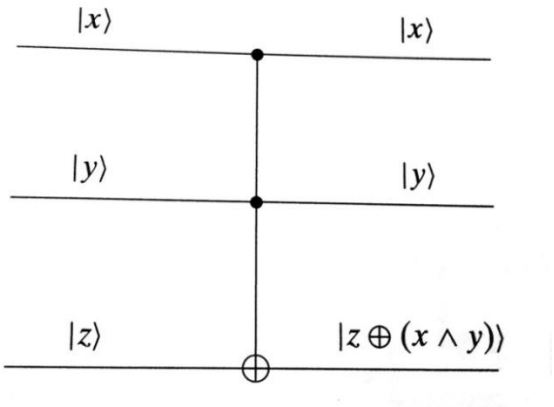
$|x\rangle$ värdet styr alltså om grinden ändrar det andra värdet eller inte. Om $|x\rangle$ är 1 så kommer $|y\rangle$ värdet att ändras som i NOT grinden.

Denna grind kan beskrivas med följande matris:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Vi kan enkel visa att CNOT grinden är reversibel genom att sätta två CNOT grindar efter varandra. CNOT av $|x, y\rangle$ blir alltså $|x, x \oplus y\rangle$ och tar vi CNOT av detta får vi $|x, x \oplus (x \oplus y)\rangle$. I och med att $\oplus$ är associativ kan detta kan skrivas om till $|x, (x \oplus x) \oplus y)\rangle$. Här ser vi att $x \oplus x = 0$ och det kan således förkortas bort. Kvar blir $|x, y\rangle$ vilket alltså är värdet vi började med. CNOT är alltså reversibel.

Troffoli grinden är en annan reversibel grind. Den liknar den kontrollerade NOT grinden men har två stycken kontrollbitar istället för en.

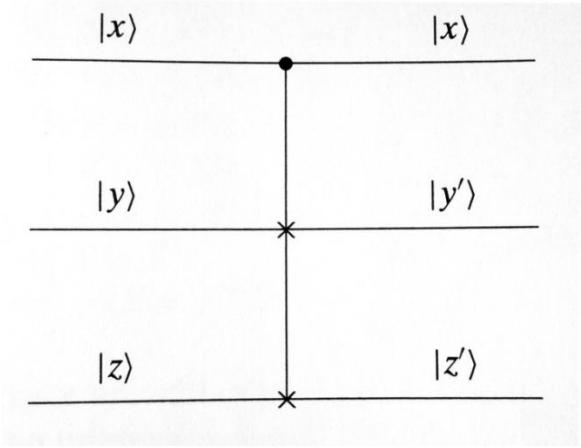


Figur 2 Troffoli grind

Troffoli grinden kan alltså, likt CNOT grinden, *exclusive-or* operationen, ”flippa” en bit, och detta sker om båda kontrollbitarna är satta till $|1\rangle$. Troffoli grinden kan också användas till att konstruera en AND grind genom att sätta $|z\rangle$ till 0, då har vi $(x \wedge y)$kvar. I och med att både en AND och en NOT grind kan konstrueras med en Troffoli grind kan vi också konstatera att Troffoli grinden likt NAND är universal.

troffoli matrix

Den sista reversibla grinden vi tittar på är Fredkin grinden.



Figur 3 Fredkin grind

Fredkin grinden tar likt har likt Troffoli grinden 3 stycken in värden. $|x\rangle$ fungerar som en kontroll bit, och kommer inte att ändras. Om $|x\rangle$ är $|0\rangle$ leder det till att $|y\rangle = |y'\rangle$ och $|z\rangle = |z'\rangle$, det vill säga grinden ger ut samma värden som den tar in. Om kontrollbiten = $|1\rangle$ kommer utvärdena att omvändas; $|y'\rangle = |z\rangle$ och $|z'\rangle = |y\rangle$. Generellt kan vi beteckna det på följande sätt $|0, y, z\rangle \rightarrow |0, y, z\rangle$ och $|1, y, z\rangle \rightarrow |0, z, y\rangle$.

fredkin matrix

Fredkin grinden är också universal. Om vi sätter $|y\rangle = |0\rangle$ så får vi en AND grind

picture

NOT grinden kan skapas genom att sätta $|y\rangle = |1\rangle$ och $|z\rangle = |0\rangle$.

Genom att studera matriserna som motsvarar Troffoli och Fredkin grindarna kan vi konstatera att båda två är unitära. De här grindarna kan alltså också användasför att utföra kvantberäkningar.

Kvantgrindar

En kvantgrind kan definieras om en operator som utför operationer på en eller flera kvantbitar Denna operator kan beskrivas av en unitär matris och operationen

bör också vara reversibel. Vi konstaterade redan att Troffoli och Fredkin grindarna uppfyller kraven för att kunna användas till kvantberäkningar. I och med att de är universella betyder det att en kvantdator kan utföra klassiska beräkningar.

För att underlätta illusterande av en kvant grind som opererar på en enskild qbit används en så kallad Bloch sfär som är en tre dimensionell sfär med radien 1. En qbits läge beskrivs som bekant med $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$. Med hjälp av några omskrivningar kan samma qbit representeras av $|\psi\rangle = \cos(\theta)|0\rangle + e^{i\phi}\sin(\theta)|1\rangle$ där kvantbiten representeras av endast två variabler θ och ϕ.

figur

I Bloch sfären är θ och ϕ vinklar som beskriver en vektor med längd 1 som börjar från origo i Bloch sfären, sfären kan ses som ett jordklot där polerna motsvarar läget $|0\rangle$ respektive $|1\rangle$ och vinklarna kan ses som latitude(θ) respektive longitude(ϕ). Sannolikheten att qbiten kollapsar till endera värden vid en mätning beskrivs av avståndet från vektorns ända till polerna, detta avstånd är beroende av vinkeln θ. Till exempel om vektorn pekar på sfärens ”ekvator” är sannolikheten att qbiten kollapsar till ettdera resultatet 50 %. Pekar vektorn på en av polerna kollapsar den alltid till det polens värde.

Om den andra vinkeln ϕ ändras så ändras inte sannolikheten för vilket värde qbiten kollapsar till utan vektorn kommer bara att rotera runt z-axeln. En sådan ändring kallas en fasändring. Grinden för att utföra en fasändring kallas $R(\theta)$ och beskrivs med matrisen: $R(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\theta} \end{bmatrix}$

Några andra vanliga kvantgrindar är Paulimatriserna:

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

I Bloche sfären kan Paulimatriserna tolkas som om de roterar sfären $180°$ längs x, y eller z axeln. $\sigma_x$är samma operation som en NOT-grind och när sfären roteras $180°$ längs x-axeln kommer polerna att byta plats, det vill säga om vektorn i sfären

pekar på $|0\rangle$ så kommer den att peka på $|1\rangle$ efter rotationen. Detta överensstämmer med hur NOT-grinden är definierad.

Hadamard grinden är en kvantgrind som opererar på en qbit som ofta används till att sätta ett antal qbitar i en superposition. Grinden utför följande operation:

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

Hadamard grinden sätter alltså en qbit i en superposition som vid en mätning får värdet 0 med 50 % sannolikhet och värdet 1 med samma sannolikhet. Om två Hadamard grindar tillämpas i följd kommer grindarna att ta ut varandra så kvantbitens tillstånd kommer inte att ändras.

Likt i klassiska datorer så finns det kvantgrindar som är universella, det vill säga grindar som kan utföra alla möjliga kvantoperationer. Med en kombination av grindarna $\left\{H, CNOT, R\left(\cos^{-1}\left(\frac{3}{5}\right)\right)\right\}$, alltså Hadamard, kontrollerade-NOT och fas grinden med värdet $\cos^{-1}\left(\frac{3}{5}\right)$

No Cloning theorem

Shor's faktoriseringsalgoritm

Shor's faktoriseringsalgoritm är en kvant algoritm som utför primtalsfaktorisering och den formulerades av Peter Shor år 1994. Shor's algoritm har polynomisk komplexitet, närmare bestämt $O = (n^2 \log n \log\log n)$. Den effektivaste metoden för primtalsfaktorisering på en klassisk dator har en komplexitet på $\exp(c(\log\ n)^{1/3}(\log\log n)^{2/3}$ (K.;W.;Manesse;& Pollard, 1990). Shors algoritm är alltså exponentiellt snabbare än den klassiska varianten.

Algoritmens in värde är alltså är ett tal N som inte är ett primtal och dess resultat blir då en faktor av N, som inte är +-1 eller +-N. Den kan delas in i två stycken

huvudsteg. Det första steget utförs på en klassisk dator och det andra på en kvantmaskin.

I det klassiska steget väljer vi slumpmässigt ett värde *a* som är mindre än *N.* Sedan beräknas den största gemensamma nämnaren (förkortat SGN) av N och a, SGN kan effektivt beräknas med till exempel Euklides Algoritm. Om SGD(N,a) != 1 har vi hittat en faktor till N och är klara. Annars, om SGD(N, a) = 1, är N och a relativt prima, det vill säga, de saknar en gemensam delare som är olik 1.

När ett passligt ha hittats använder vi det till att lösa funktionen $f_{a,N}(x) = a^x \; Mod \; N$ för olika värden på *x*. Om man till exempel väljer a till 13 och N till 15 kommer de första resultaten av funktionen att se ut på följande sätt:

image

Resultatet kan också illustreras med hjälp av en graf:

graf

Man ser ur grafen att funktionens resultat börjar upprepa sig efter ett visst antal steg, det vill säga funktionen är periodisk. Och det är alltså periodens längd som vi vill hitta, alltså hitta ett *r* så att funktionen $f_{a,N}(r) = a^r Mod \; N = 1$. Denna funktion kommer alltid att få resultatet 1 för något $r < N$. Detta kan betecknas med:

$$f_{a,N}(r + s) = f_{a,N}(s)$$

Med små värden är det relativt enkelt att räkna ut perioden för $f_{a,N}(r)$ men med väldigt stora värden på N blir omöjligt att lösa detta med en klassisk dator. Det är i det här steget som kvantdatorn kommer till nytta, i och med att den möjliggör att $f_{a,N}(r)$ för alla *r* kan beräknas med *en* operation (Pittenger, 2000)

För att kunna lösa ekvationen behövs två kvantregister *m* och *n* där register *n* har storleken $\log_2 N$ bitar och register *m* har storleken *2n*. Beräkningen av $f_{a,N}(r)$ kan illustreras med en figur:

Beräkningen börjar med två kvantregister *m* och *n* som initialiseras till värdet 0.

$$|\varphi_0\rangle = |0_m, 0_n\rangle$$

Sedan sätts *m* i en jämn superposition genom att tillämpa Hadmard grinden på dem

$$|\varphi_1\rangle = \frac{\sum_{\boldsymbol{x}\in\{1,0\}^m} |\boldsymbol{x}, 0_n\rangle}{\sqrt{2^m}}$$

Följande steg är att tillämpa kvantkretsen $U_{f_{a,N}}$ som betecknas med $|x, y\rangle \rightarrow \left|x, y \ominus f_{a,N}(x)\right\rangle = |x, y \oplus a^x \text{Mod } N\rangle$

$$|\varphi_2\rangle = \frac{\sum_{x\in\{0,1\}^m} \left|x, f_{a,N}(x)\right\rangle}{\sqrt{2^m}} = \frac{\sum_{x\in\{0,1\}^m} |x, a^x \text{ Mod } N\rangle}{\sqrt{2^m}}$$

Placeholder

_____

Shor's algoritm bidrog också till att intresset för kvantdatorer ökade i och med att algoritmen kan användas till att knäcka RSA kryptot, som är baserat på att primtalsfaktorisering av stora tal är så gått som omöjligt på en klassisk dator. RSA är ett "public-key" krypto som bland annat brukar användas för att kryptera meddelanden som skickas över Internet.

(Shor, 2005)

Hårdvara

Hittills har vi pratat om endast diskuterat hur en teoretisk kvantdator fungerar. För att kunna dra någon nytta av detta vid beräkningar krävs det att vi har en riktig kvantdator att utföra dem på. För att uppnå detta behöver vi en tillräcklig mängd individuellt adresserbara qbitar som går att initialisera till ett väldefinierat tillstånd, t.ex alla qbitar har värdet 1. Sedan bör vi kunna utföra operationer med kvant grindar med både en och två qbits grindar, på qbitarna med en tillräckligt låg

felmarginal. Slutligen vill vi kunna avläsa resultatet av dessa operationer exakt. Det krävs också att qbitarnas tillstånd endast ändras då en operation utförs. För att kunna implementera en kvantdator förutsätts det att man använder sig av ett kvantsystem med 2 dimensioner för att representera qbitens värde. (DiVincenzo, 2000)

Flera olika metoder för att lösa detta

Jonfällor

En jon är en atom med en positiv eller negativ elektiskt laddning, katjon/anjon. I och med att en jon har en elektrisk laddning kan man påverka den med ett elektromagnetiskt fält. Detta möjliggör att vi kan stänga in en jon inom ett bestämt område med hjälp av ett elektromagnetiskt fält, vilket kallas en jonfälla (Paul, 1990) .

Kvantbitens läge kan lagras i atom i och med att atom kan vara i ett exciterat tillstånd ($|1\rangle$)eller i ett grund tillstånd ($|0\rangle$). I exciterat tillstånd har en elektron hoppat högre elektronorbital. En atom kan sättas i ett högre tillstånd genom att tillföra en foton. När atomen avger en foton återgår den till sitt grundtillstånd. Genom att beskjuta jonerna med en laser kan vi med stor sannolikhet initialisera dem till ”exited state”.

Qbitens tillstånd kan sedan manipuleras genom att på olika sätt beskjuta dem med en laser.

Genom att försiktigt beskjuta jonen kan man få den till ett tredje kortlivat tillstånd $|s\rangle$. Vilket kan ses som ett tillstånd mellan $|0\rangle$ och $|1\rangle$. Om jonen är i grundtillståndet och dess till ändras till $|s\rangle$ kommer den att återgå till grundtillståndet och avge en foton. Om jonen är i exciterat tillstånd kommer detta inte att ske. Genom att detektera fotonen kan vi bestämma vilket tillstånd jonen var i.

Fördelem med detta system är relativ lång samstämmighet av kvanttillståndet, väldigt pålitliga mätningar, och det är möjligt att transportera runt qbitar i datorn(kvanttillstånd kan inte kopieras). Negativt med metoden är att den är svårt att utöka antalet grindar och att grindar för detta system är relativt långsamma.

Linjär optik

En annan metod för att implementera kvantdatorer är med hjälp linjär optik och ljus alltså fotoner. Kvantbiten kan implementeras i ljusets polarisering, alltså i vilket plan ljuset vibrerar. Ljus som passerar genom ett polariseringsfilter kommer alltså att vibrera i ett bestämt plan.

Andra metoder

Var är vi idag på hårvarufronten?

Slutsatser

I dagsläge är kvantberäkningar

Bibliography

Dandamudi, S. P. (2002). *Fundamentals of Computer Organization and Design.* School of Computer Science Carleton University.

DiVincenzo, D. P. (2000). *The Physical Implementation of Quantum Computation.* IBM T.J. Watson Research Center.

Hirvensalo, M. (2001). *Quantum Computing.* Springer.

K., L. A., W., L. H., Manesse, M. S., & Pollard, J. M. (1990). The number field sieve. *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 564-572.

McMahon, D. (2007). *Quantum computing explained.* John Wiley & Sons.

Meglicki, Z. (2008). *Quantum Computing without magic devices.* MIT.

Paul, W. (1990). Electromagnetic traps for charged and neutral particles. *RevModPhys.62*, 531--540.

Shor, P. W. (2005). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*.

Yanofsky, N. S., & Manunucci, M. A. (2008). *Quantum computing for computer scientists.* Cambridge University Press.

http://www.physics.sc.edu/~knight/502s08/spin.pdf

http://www.tp.umu.se/~dion/atmol/dirac.pdf