SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 69420

Simulacija kvantnog računala

Dominik Matić

Zagreb, svibanj 2021.

SADRŽAJ

1. Uvod

Neprestanim razvojem znanosti i tehnologije, kvantna računala postaju sve bitnijim dijelom računarstva. Predstavljaju sasvim drugačiji način računanja koji otvara vrata rješavanju mnogih problema koje klasično računalo, zbog same prirode problema, ili rješava znatno sporije ili uopće ne može riješiti u razumnom vremenu. Takvi problemi javljaju se u područjima kriptografije, umjetne inteligencije i strojnog učenja, računalne biologije, financija, simulacije kvantnih sustava, kao i u ostalim područjima računarstva kao što su algoritmi pretraživanja.

Područje kriptografije je posebno zanimljivo jer se većina današnjih sigurnosnih mehanizama interneta temelji na matematičkim problemima za koje se smatra da su teško izračunljivi. To su primarno problem faktorizacije velikih brojeva te problem diskretnog logaritma. Kvantno računalo efikasno rješava takve probleme, što je još devedesetih godina demonstrirao Peter Shor[6]. Naravno, iz toga slijedi da će kvantna računala biti velika prijetnja sigurnosti na internetu, no već su se počeli razvijati mehanizmi koji će biti otporni na napade kvantnim računalom[citati?] što samo govori o tome koliko je kvantno računalo blizu da postane velikim dijelom računarstva.

Potencijal kvantnog računala poznat je desetljećima te su već smišljeni i detaljno opisani mnogi od algoritama prikladni za takav način računanja. Sve što je preostalo je izgraditi računalo koja će moći izvoditi te algoritme. Danas, IBM posjeduje kvantno računalo s najvećim brojem kvantnih bitova, njih čak 65, no ni to još nije dovoljno da bude korisno. Naime, problem nastaje s pojavom kvantne dekoherencije. Dekoherencija predstavlja gubitak informacije kvantog sustava zbog interakcije s okolinom što onemogućava precizno ili čak bilo kakvo računanje. No, kvantno računarstvo je trenutno veliki predmet istraživanja te se napretci ostvaruju skoro svaki dan. Mogućnost izgradnje kvantnog računala sa dovoljno velikim brojem kvantnih bitova je sve izglednija, kao primjer, IBM ~~obećava~~ do 2023. godine izgraditi kvantno računalo sa 1000 kvantnih bitova[3].

S ciljem demonstracije nekih od mogućnosti kvantnog računala, ovaj rad se u provm dijelu bavi osnovnim načelima kvantnog računala, odnosno matematičkim mo-

delom kojim se opisuju kvantnomehaničke pojave koje omogućuju drugačiji način računanja. Nakon toga rad opisuje neke od najpoznatijih algoritama namijenjenih izvedbi na kvantnim računalima koji će zatim neki od njih biti implementirani i demonstrirani u samostalno izgrađenom simulatoru.

2. Kvantno računalo

2.1. Polarizacija svjetlosti

Klasična fizika shvaća svjetlost kao transverzalni elektromagnetski val koji može biti poraliziran na različite načine. Polarizacija svjetlosti određena je njenom električnom komponentom te je svjetlost koju emitiraju prirodni izvori svjetlosti uglavnom nepolarizirana. Tek kada se snop svjetlosti pusti kroz polarizator dobije se linearno polariziran val svjetlosti — val koji ima stalni smjer širenja okomit na smjer titranja. Njegova električna komponenta dana je jednadžbom:

$$E = E_0 \hat{p} e^{i\omega t} \tag{2.1}$$

gdje je $\hat{p}$ jedinični vektor u smjeru polarizacije vala. Intenzitet ovako polariziranog vala biti će upola manji od intenziteta početnog nepolariziranog vala jer će točno toliko intenziteta polarizator apsorbirati. Kada ovakav val pustimo kroz još jedan polarizator (tzv. analizator), val koji ćemo dobiti jest:

$$E' = (E \cdot \hat{n})\hat{n} = E_0(\hat{p} \cdot \hat{n})\hat{n} e^{i\omega t} = E_0 \cos\alpha \tag{2.2}$$

gdje je $\hat{n}$ jedinični vektor u smjeru polarizacije analizatora, a α kut između vektora $\hat{p}$ i $\hat{n}$. Intenzitet novog vala dobivamo po Malusovom zakonu:

$$I' = I \cos^2\alpha \tag{2.3}$$

gdje je I intenzitet prethodno polariziranog vala. Drugim riječima, polarizacijom vala dobivamo njegovu projekciju na ravninu određenu kutom polarizatora.

U kvantnoj mehanici, svjetlost je definirana drugačije — kao niz fotona, nedjeljivih elementarnih čestica. Kod takve interpretacije postavlja se pitanje što se događa kada individualni fotoni prolaze kroz polarizator. Ako polarizator propušta projekciju, koja je uvijek manja ili jednaka ulaznom valu, što će se dogoditi fotonu, ako je on nedjeljiv? Eksperimenti su pokazali da će foton sa točno određenom šansom ili cijeli proći kroz

polarizator sa novim smjerom polarizacije ili biti apsorbiran. Ključan element takvih eksperimenata je činjenica da je nemoguće *niti u načelu* odrediti što će se dogoditi sa fotonom. To znači da čak uz poznavanje svih varijabli nekog sustava, nemoguće je deterministički odrediti ishod.

Prijelaze kvantnih sustava iz jednog stanja u drugo modeliramo takozvanom amplitudom vjerojatnosti:

$$a(\Phi \to \Psi) \tag{2.4}$$

koja je element skupa kompleksnih brojeva, a vjerojatnost da neki sustav koji je u stanju Φ bude izmjeren u stanju Ψ dobivamo na način:

$$p(\Phi \to \Psi) = |a(\Phi \to \Psi)|^2 \tag{2.5}$$

U slučaju polarizatora, vjerojatnost ~~smo mogli~~ izračunati pomoću Malusovog zakona što ~~nam~~ daje vjerojatnost prolaska fotona kroz polarizator jednaku

$$p(\Phi \to \Psi) = \cos^2 \alpha \tag{2.6}$$

iz čega je vidljiva amplituda vjerojatnosti:

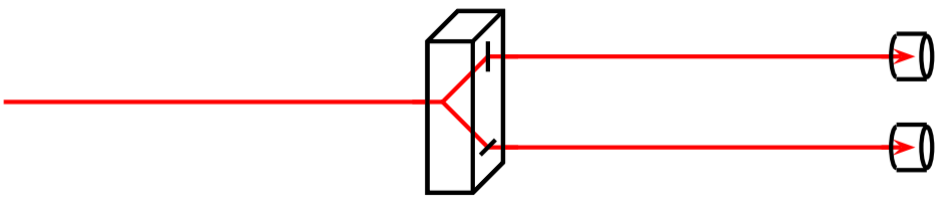
$$|a(\Phi \to \Psi)|^2 = \cos^2 \alpha \Rightarrow a(\Phi \to \Psi) = \cos \alpha \tag{2.7}$$

gdje je α kut koji zatvaraju vektori polarizacije fotona i polarizatora, Φ stanje fotona prije prolaska kroz polarizator, a Ψ stanje u kojemu foton nije apsorbiran.

2.2. Kvantni bit

2.2.1. Definicija

Dvolomac je vrsta polarizatora koji dijeli svjetlost na dva ortagonalno polarizirana snopa. Kombinirajući dvolomac sa dva detektora uvijek je moguće odrediti u koje je stanje prešao foton pušten kroz dvolomac.



Slika 2.1: Dvolomac s dva detektora

Nakon detekcije, foton je apsorbiran i ne može se više mjeriti. Takav kvantnomehanički sustav, koji je moguće samo jednom izmjeriti u samo jednom od dva moguća stanja nazivamo **kvantnim bitom** ili **qubitom**. Kvantni bit prije mjerenja može biti u beskonačno mnogo različitih stanja te se općenito jedno takvo stanje naziva **superpozicijom** dvaju stanja. Ta dva stanja se odnose na stanja baze mjerenja koja mogu biti proizvoljna, ali su uvijek međusobno ortogonalna. Pojam superpozicije se može primijeniti i u klasičnom smislu. U slučaju svjetlosti, uzimajući dva linearno polarizirana vala s okomitim smjerovima polarizacije kao bazu sustava, općenito stanje polarizacije vala može se prikazati kao:

$$E = E_x\hat{x}e^{i(\omega t+\phi_x)} + E_y\hat{y}e^{i(\omega t+\phi_y)} = E_0(\lambda\hat{x} + \mu\hat{y})e^{i\omega t} \tag{2.8}$$

gdje vrijedi:

$$E_0 = \sqrt{(E_x^2 + E_y^2)} \qquad \lambda = \frac{E_x}{E_0}e^{i\phi_x} \qquad \mu = \frac{E_y}{E_0}e^{i\phi_y} \qquad |\lambda|^2 + |\mu|^2 = 1 \tag{2.9}$$

Bitno je uočiti da su λ i μ kompleksni brojevi kao i da dio izraza $\lambda\hat{x} + \mu\hat{y}$ sadrži informaciju o stanju polarizacije vala te da takvih stanja ima beskonačno mnogo.

2.2.2. Diracova notacija

Stanja kvantnih bitova u praksi se prikazuje ket simbolima koji su dio Diracove ili bra-ket notacije:

$$\text{bra:} \qquad \langle\Phi|$$

$$\text{ket:} \qquad |\Phi\rangle$$

$$\text{braket:} \qquad \langle\Phi|\Phi\rangle$$

S pretpostavkom da su $|x\rangle$ i $|y\rangle$ dva stanja baze mjerenja sustava, kao npr. stanje fotona polariziranog u smjeru x i stanje fotona polariziranog u smjeru y, općenito stanje polarizacije fotona možemo prikazati superpozicijom ta dva stanja:

$$|\Phi\rangle = \lambda\,|x\rangle + \mu\,|y\rangle$$

gdje su λ i μ kompleksni brojevi te vrijedi $|\lambda|^2 + |\mu|^2 = 1$.

2.2.3. Svojstva Hilbertovog prostora

U matematičkom smislu, stanja kvantnih bitova prikazanih ketovima su zapravo vektori dvodimenzionalnog kompleksnog Hilbertovog prostora. Za takav prostor vrijede sljedeća svojstva.

Skalarni umnožak:

$$\langle \Phi | \Psi \rangle = \langle \Psi | \Phi \rangle^* \in \mathbb{C}$$

$$\langle \Phi | \Phi \rangle \geq 0$$

Norma vektora $|\Phi\rangle$:

$$||\Phi|| = \sqrt{\langle \Phi | \Phi \rangle}$$

Vektori stanja baze moraju biti ortonormirani što znači da mora vrijediti:

$$\langle x | y \rangle = 0 \qquad \langle x | x \rangle = \langle y | y \rangle = 1$$

Za računanje skalarnog umnoška potrebno je izračunati bra nekog stanja koji se dobiva kompleksnom konjugacijom keta. Općenito, izračun $\langle \Psi | \Phi \rangle$ dobiva se:

$$|\Phi\rangle = \lambda |x\rangle + \mu |y\rangle$$

$$|\Psi\rangle = \nu |x\rangle + \sigma |y\rangle \Rightarrow \langle \Psi | = \nu^* \langle x | + \sigma^* \langle y |$$

$$\langle \Psi | \Phi \rangle = \nu^* \lambda \langle x | x \rangle + \nu^* \mu \langle x | y \rangle + \sigma^* \lambda \langle y | x \rangle + \sigma^* \mu \langle y | y \rangle = \nu^* \lambda + \sigma^* \mu = \langle \Phi | \Psi \rangle^*$$

Norma vektora se računa na analogan način.

2.2.4. Amplituda vjerojatnosti i vjerojatnost

Amplituda vjerojatnosti modelira prijelaz jednog kvantnog stanja u drugo koji se računa skalarnim umnoškom vektora stanja.

$$a(\Phi \to \Psi) = \langle \Psi | \Phi \rangle$$

Odgovarajuća vjerojatnost mjerenja kvantnog bita u stanju Ψ ako se prethodno nalazio u stanju Φ računa se:

$$p(\Phi \to \Psi) = |a(\Phi \to \Psi)|^2 = |\langle \Psi | \Phi \rangle|^2$$

2.2.5. Vektorski prikaz

Za lakšu implementaciju kvantnog bita na klasičnom računalu, korisno je prikazati kvantne bitove vektorski. Klasična nula i jedinica bi imale vektorski oblik:

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

Ti vektori odgovaraju ujedno i vektorima stanja baze mjerenja nekog sustava koji se mogu označiti kao $|0\rangle$ i $|1\rangle$. Sva ostala stanja mogu se dobiti superpozicijom tih vektora:

$$|\Phi\rangle = \lambda\,|0\rangle + \mu\,|1\rangle = \lambda \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \mu \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda \\ \mu \end{bmatrix}$$

Bra se dobiva kompleksnim konjugiranjem i transponiranjem keta:

$$\langle\Psi| = \begin{bmatrix} \nu \\ \sigma \end{bmatrix}^{\dagger} = \begin{bmatrix} \nu^* & \sigma^* \end{bmatrix}$$

Računanje amplitude vjerojatnosti se tada svodi na jednostavno matrično množenje:

$$\langle\Psi|\Phi\rangle = \begin{bmatrix} \nu^* & \sigma^* \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = \nu^*\lambda + \sigma^*\mu$$

2.3. Sustav kvantnih bitova

2.3.1. Dva ili više kvantnih bitova

Kao što se jedan kvantni bit prikazuje kao vektor u dvodimenzionalnom kompleksnom Hilbertovom prostoru, sustav od n kvantnih bitova može se prikazati kao vektor u 2^n-dimenzionalnom kompleksnom Hilbertovom prostoru.

Za n stanja oblika $|\Phi_i\rangle, i \in \{1, 2, \ldots, n\}$, njihov zajednički vektor stanja $|\Psi\rangle$ dobiva se **tenzorskim produktom** svih stanja:

$$|\Psi\rangle = |\Phi_1\rangle \otimes |\Phi_2\rangle \otimes \ldots \otimes |\Phi_n\rangle$$

Za dva kvantna bita, tenzorski produkti vektora stanja baze mjerenja sustava označavaju se kao:

$$|0\rangle \otimes |0\rangle = |00\rangle \qquad |0\rangle \otimes |1\rangle = |01\rangle \qquad |1\rangle \otimes |0\rangle = |10\rangle \qquad |1\rangle \otimes |1\rangle = |11\rangle$$

U vektorskom obliku te baze se prikazuju kao:

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Općenito, za dva kvantna bita dana u obliku $|\Phi_i\rangle = \lambda_i\,|0\rangle + \mu_i\,|1\rangle, i \in \{1, 2\}$, tenzorski produkt jednak je:

$$\begin{aligned} |\Phi_1\rangle \otimes |\Phi_2\rangle &= (\lambda_1\,|0\rangle + \mu_1\,|1\rangle) \otimes (\lambda_2\,|0\rangle + \mu_2\,|1\rangle) \\ &= \lambda_1\lambda_2\,|00\rangle + \lambda_1\mu_2\,|01\rangle + \mu_1\lambda_2\,|10\rangle + \mu_1\mu_2\,|11\rangle \end{aligned} \tag{2.10}$$

U vektorskom obliku:

$$|\Phi_1\rangle \otimes |\Phi_2\rangle = \begin{bmatrix} \lambda_1 \\ \mu_1 \end{bmatrix} \otimes \begin{bmatrix} \lambda_2 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 \begin{bmatrix} \lambda_2 \\ \mu_2 \end{bmatrix} \\ \mu_1 \begin{bmatrix} \lambda_2 \\ \mu_2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \lambda_1\lambda_2 \\ \lambda_1\mu_2 \\ \mu_1\lambda_2 \\ \mu_1\mu_2 \end{bmatrix}$$

Analogno se dobiju vektori 3 ili više kvantnih bitova.

Naravno, za svaki vektor koji opisuje sustav od n kvantnih bitova oblika

$$\lambda_1 |00\ldots00\rangle + \lambda_2 |00\ldots01\rangle + \ldots + \lambda_{2^n} |11\ldots11\rangle$$

vrijedi:

$$|\lambda_1|^2 + |\lambda_2|^2 + \ldots + |\lambda_{2^n}|^2 = 1 \tag{2.11}$$

2.3.2. Spregnutost

Spregnutost (engl. *entaglement*) opisuje sustav kvantnih bitova koji se ne može prikazati tenzorskim produktom vektora stanja pojedinih kvantnih bitova. Bitno je uočiti da iz jednadžbe 2.10 vrijedi da je umnožak koeficijenata uz $|00\rangle$ i $|11\rangle$ jednak umnošku koeficijenata uz $|01\rangle$ i $|10\rangle$.

$$\lambda_1\lambda_2 \cdot \mu_1\mu_2 = \lambda_1\mu_2 \cdot \mu_1\lambda_2$$

Stanja u kojima vrijedi prethodna jednadžba nazivaju se separabilnima, te zauzimaju samo podskup Hilbertovog prostora u kojemu se nalaze. Vektori stanja koji ne zadovoljavaju takvu jednakost, uz nužan uvjet da zadovoljavaju jednadžbu 2.11 nazivaju se spregnutim stanjima. Jedno takvo stanje jest:

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

Mjerenje takvog sustava rezultiralo bi detekcijom stanja $|00\rangle$ ili $|11\rangle$, ali nikada stanja $|01\rangle$ ili $|10\rangle$. Drugim riječima, mjerenjem samo jednog kvantnog bita u sustavu, odmah je poznata i vrijednost drugoga.

Spregnutost se ne odnosi samo na sustave s dva kvantna bita, analogno se ista logika može primijeniti i na sustave s 3 ili više kvantnih bitova. Primjer jednog spregnutog stanja s tri kvantnih bitova:

$$|\Phi\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$$

2.4. Kvantni operatori

2.4.1. Svojstva

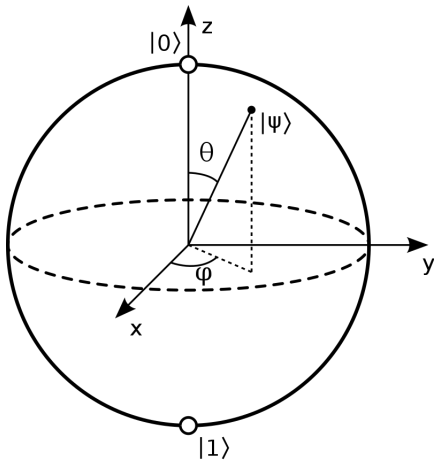
Kako bi bilo kakav izračun s kvantnim bitovima bio moguć, potrebno je moći manipulirati s njima. S klasičnim bitovima to rade operatori kao što su AND, OR, XOR i sl. Kvantni operatori su nešto drugačiji. Naime, ~~pošto~~ je jedna od glavnih značajki kvantne mehanike reverzibilnost sustava, kvantni operatori moraju biti **reverzibilni**, što je od klasičnih operatora samo NOT. ~~Pošto~~ stanja kvantnih bitova moraju uvijek ostati normirana, operatori su također **unitarni**. Uz unitarnost, kvantni operatori su često i **hermitski**.

2.4.2. Blochova sfera

Radi lakše interpretacije nekih od operatora, korisno je poznavati reprezentaciju kvantnog bita na Blochovoj sferi. Svako stanje[1] kvantnog bita može se prikazati točkom na površini Blochove sfere.

Ta činjenica se čini zbunjujućom jer svaki kvantni bit u potpunosti određuju dva kompleksna broja što rezultira s četiri stupnja slobode, no uzimajući u obzir da kvantni bit mora biti normiran i da množenje kvantnog bita s faznim faktorom nema nikakvog fizičkog utjecaja na njegovo stanje, dobiju se samo dva stupnja slobode. Na temelju toga, općenito stanje kvantnog bita se može izraziti kao:

$$|\Psi\rangle = e^{-i\frac{\varphi}{2}} \cos\frac{\theta}{2}|0\rangle + e^{i\frac{\varphi}{2}} \sin\frac{\theta}{2}|1\rangle$$



Slika 2.2: Blochova sfera

[1]Odnosi se na čista stanja (engl. *pure states*), ali postoje i takozvana mješovita stanja(engl. *mixed states*) koja se mogu prikazati točkama unutar sfere, ali takva stanja nisu bitna u sklopu ovog rada

Bitno je primjetiti da vektori koji su u Hilbertovom prostoru ortogonalni su na Blochovoj sferi suprotni. U točki gdje os x probada sferu nalazi se stanje $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, a suprotno njemu $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Na isti način u smjeru osi y nalazi se stanje $\frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle)$ te suprotno njemu $\frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle)$. Navedeni vektori označavaju se kao $|x\rangle$ i $|y\rangle$, odnosno $|R\rangle$ i $|L\rangle$.

2.4.3. Operator projekcije

Skalarnim umnoškom dva vektora stanja kvantnih bitova dobije se skalar koji se može interpretirati kao koliko prvi vektor 'ide' u smjeru drugoga. Ako se taj skalar pomnoži drugim vektorom, dobije se vektor projekcije prvoga na drugi.

$$|\Omega\rangle = |\Psi\rangle \langle\Psi|\Phi\rangle$$

Dio izraza $|\Psi\rangle \langle\Psi|$ možemo shvatiti kao operator projekcije na stanje $|\Psi\rangle$ te se takav operator naziva projektorom. Vektori dobiveni projekcijom općenito nisu normirani te kao takvi ne predstavljaju neko stanje kvantnog bita. Projektori se koriste kako bi se konstruirali drugi operatori.

Matrični prikaz projektora na vektore stanja $|0\rangle$ i $|1\rangle$:

$$|0\rangle \langle 0| = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad |1\rangle \langle 1| = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

2.4.4. Hermitski operator

Svaki operator u Hilbertovom prostoru koji je jednak sebi kada se kompleksno konjugira i transponira naziva se Hermitskim operatorom.

$$A^\dagger = A$$

Hermitski operator se može prikazati kao linearna kombinacija projektora ortonormirane baze N-dimenzionalnog prostora pomnoženih *realnim* koeficijentima:

$$A = \sum_{i=1}^{N} a_i \,|i\rangle \langle i| \qquad a_i \in \mathbb{R}$$

Vektori $|i\rangle$ su svojstveni vektori Hermitskog operatora sa odgovarajućim svojstvenim vrijednostima a_i.

Hermitski operatori se mogu koristiti za izračune vezane uz fizikalne veličine pripisane baznim stanjima nekog kvantnog sustava, no u sklopu kvantnog računarstva koriste se za manipulaciju stanjima.

2.4.5. Paulijeve matrice

Paulijeve matrice su Hermitske i unitarne matrice koje su često korišteni operatori u kvantnim logičkim krugovima, označavaju se kao X, Y i Z, odnosno σ_x, σ_y i σ_z. Svojstvene vrijednosti svake Paulijeve matrice su ± 1, dok su svojstveni vektori:

$$\begin{aligned}
\sigma_x: &\qquad |x\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad |y\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\
\sigma_y: &\qquad |R\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle) \qquad |L\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle) \\
\sigma_z: &\qquad |x\rangle \qquad |y\rangle
\end{aligned}$$

Njihove vrijednosti iznosee:

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

Svaka Paulijeva matrica vrši operaciju rotacije vektora stanja kvantnog bita za π radijana na Blochovoj sferi oko osi po kojoj je nazvana.

Paulijeve matrice su također važne jer linearnom kombinacijom njih i jedinične matrice moguće je konstruirati bilo koji Hermitski operator:

$$A = \lambda_0 I + \sum_{i=1}^{3} \lambda_i \sigma_i \tag{2.12}$$

gdje su svi λ koeficijenti realni.

2.4.6. Hadamardov operator

Hadamardov operator vrši operaciju rotacije od π radijana oko osi $\frac{\hat{x}+\hat{z}}{\sqrt{2}}$. Primarno se koristi kako bi se kvantni bit postavio u stanje superpozicije. Njegova vrijednost je:

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

Primjer Hadamardovog operatora nad kvantnim bitom $|1\rangle$:

$$H\,|1\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Naravno, zbog reverzibilnosti operatora, ponovnom primjenom Hadamardovog operatora kvantni bit bi se vratio u stanje $|1\rangle$.

2.4.7. Ostali unarni operatori

Svi do sada navedeni operatori djeluju nad samo jednim kvantnim bitom, pa se takvi operatori nazivaju unarnim. Postoje i drugi operatori kao što su $\sqrt{\text{NOT}}$ i operator faznog pomaka $P[\phi]$:

$$\sqrt{\text{NOT}} = \sqrt{\sigma_x} = \frac{1}{1+i}\begin{bmatrix} 1 & i \\ i & 1 \end{bmatrix} \qquad P[\phi] = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$$

No, njih kao i sve ostale unitarne Hermitske operatore koje je moguće dobiti na način dan jednadžbom 2.12, nije potrebno posebno razraditi u sklopu ovoga rada.

2.4.8. CU operatori

CU operatori ili control-U operatori djeluju na 2 ili više kvantnih bitova od kojih je barem jedan bit **upravljački**. Općeniti izgled CU operatora za dva kvantna bita u *blok-matričnom* prikazu je:

$$CU = \begin{bmatrix} I & 0 \\ 0 & U \end{bmatrix} \tag{2.13}$$

gdje je U bilo koji unitarni operator, a I jedinična matrica. U načelu, operator provodi operaciju U nad drugim kvantnim bitom ako je prvi u jedinici, inače ga ne mijenja.

Najkorišteniji CU operator jest CX ili CNOT čija se funkcionalnost može opisati kao:

$$CNOT\,|xy\rangle = |x, y \oplus x\rangle$$

gdje je x upravljački bit, a $\oplus$ operator zbrajanja modulo 2. Vrijedi:

$$CNOT\,|00\rangle = |00\rangle$$

$$CNOT\,|01\rangle = |01\rangle$$

$$CNOT\,|10\rangle = |11\rangle$$

$$CNOT\,|11\rangle = |10\rangle$$

s odgovarajućom matricom:

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

Na analogan način se dobiju matrice za CY, CZ i CH unarnih operatora σ_y, σ_z, odnosno H.

Ovakva konstrukcija CU operatora uvijek pretpostavlja da će prvi kvantni bit biti kontrolni, a onaj nad kojim se vrši operacija drugi. Pošto je na kvantnom računalu moguće da bilo koja dva para kvantnih bitova budu operandi, korisno je, za svrhu izgradnje kvantnog simulatora, na drugačiji način prikazati konstrukciju CU operatora. Sljedeća jednadžba opisuje CU operator već spomenutog oblika 2.13, gdje je prvi bit kontrolni.

$$CU_{1,2} = |0\rangle \langle 0| \otimes I_2 + |1\rangle \langle 1| \otimes U$$

Ukoliko je potrebno izgraditi matricu CU operatora gdje je kontrolni bit drugi, a ne prvi, u prethodnoj jednadžbi potrebno je samo promijeniti redoslijed varijabli na način:

$$CU_{2,1} = I_2 \otimes |0\rangle \langle 0| + U \otimes |1\rangle \langle 1|$$

2.4.9. SWAP operator

SWAP operator zamjenjuje stanja dvaju kvantnih bitova. Moguće ga je dobiti kombinacijom CNOT operatora:

$$\mathit{SWAP} = CNOT_{1,2} \cdot CNOT_{2,1} \cdot CNOT_{1,2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.4.10. Toffolijeva i Fredkinova vrata

Toffolijeva vrata ili CCNOT operator jest operator koji djeluje na tri kvantna bita od kojih su dva kontrolna. Oba kontrolna bita moraju biti u jedinici kako bi se obavila operacija NOT nad trećim bitom. Matrična forma CCNOT operatora gdje su prvi i drugi bit kontrolni jest:

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

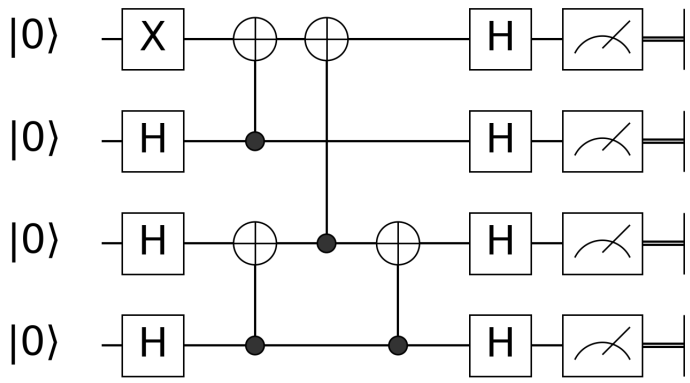
Fredkinova vrata ili drugim imenom CSWAP operator radi točno što mu ime sugerira. Ako je kontrolni bit u jedinici, obavi operaciju SWAP nad ostala dva bita, inače ne radi ništa. Njegova matrica jest:

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

2.5. Kvantni logički kurg

2.5.1. Topografija logičkog kruga

Kvantni logički krug sastoji se od kvantnih bitova i kvantnih logičkih vrata kroz koja prolaze određeni kvantni bitovi. Bitovi su reprezentirani kao paralelne linije na koje se s lijeva na desno primjenjuju odgovarajuća logička vrata. Na kraju logičkog kruga vrši se mjerenje[2].



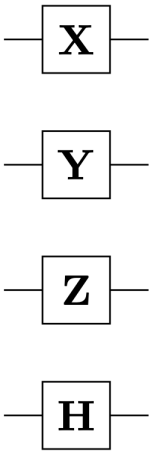
Slika 2.3: Primjer kvantnog logičkog kruga

[2]Po načelu odgođenog mjerenja (engl. *Deferred Measurement Principle*) odgađanje mjerenja do kraja logičkog kruga ne utječe na raspodjelu vjerojatnosti ishoda, pa se iz tog razloga mjerenje uvijek smješta na kraj, iako je sasvim moguće mjeriti pojedine kvantne bitove kada se zna da više neće prolaziti kroz neka logička vrata.

Također, standardno je da su svi kvantni bitovi inicijalizirani u stanje $|0\rangle$.

2.5.2. Reprezentacija logičkih operatora

Svaki do sada spomenuti logički operator ima svoju reprezentaciju u kvantnom logičkom krugu. Operatori σ_x, σ_y, σ_z i Hadamard:

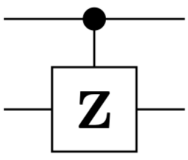


Slika 2.4: Reprezentacija unarnih logičkih operatora

Pošto je σ_x često korišten ima i drugi oblik:

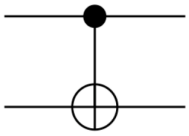
Slika 2.5: Alternativni σ_x

CU operatori imaju crnu točku na kontrolnom bitu spojenu vertikalnom crtom sa unarnim operatorom na drugom bitu:



Slika 2.6: CZ operator

No, kao i σ_x, CNOT ima drugačiji prikaz koji se puno češće koristi:

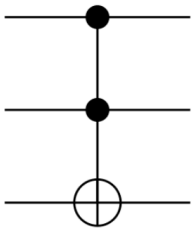


Slika 2.7: CNOT operator

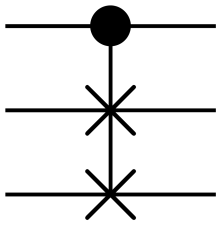
SWAP operator također ima dvije reprezentacije:

Slika 2.8: SWAP operator

Toffolijeva i Fredkinova vrata:



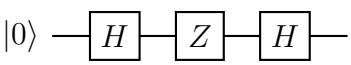
Slika 2.9: Toffolijeva vrata



Slika 2.10: Fredkinova vrata

2.5.3. Matrični prikaz

Svaki kvantni logički krug može se prikazati jednom unitarnom transformacijom. U slučaju kvantnog logičkog kruga s jednim kvantnim bitom, matrica cijelog logičkog kruga dobiva se jednostavnim matričnim množenjem operatora.

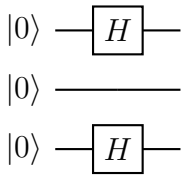


$$U = HZH = \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Vektor stanja na kraju logičkog kruga:

$$U |0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

Matrica nezavisnih paralelnih operatora dobiva se *tenzorskim produktom* pojedinih operatora, odnosno matricom identiteta ako nema operatora.



$$\begin{aligned} U = H \otimes I \otimes H &= \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} I & I \\ I & -I \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} H & 0 & H & 0 \\ 0 & H & 0 & H \\ H & 0 & -H & 0 \\ 0 & H & 0 & -H \end{bmatrix} \end{aligned}$$

Stanje sustava na kraju logičkog kruga je:

$$U |000\rangle = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 & -1 & -1 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \\ 0 \\ \frac{1}{2} \\ \frac{1}{2} \\ 0 \\ 0 \end{bmatrix}$$

Bitovi 0 i 2 su postavljeni u stanje superpozicije dok je bit 1 ostao u stanju $|0\rangle$ rezultirajući da se mjerenjem sustava uvijek dobije stanje koje na drugom bitu ima nulu. Iz

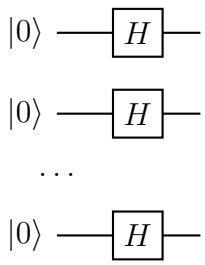
vektora se vidi da se mjerenjem mogu dobiti samo stanja $|000\rangle$, $|001\rangle$, $|100\rangle$ i $|101\rangle$ što odgovara očekivanjima.

Kombinirajući tenzorski produkt i matrično množenje može se izračunati matrična reprezentacija svakog kvantnog logičkog kruga što je u suštini točno ono što izrađeni simulator kvantnog računala i radi.

3. Kvantni algoritmi

3.1. Kvantni paralelizam

Kvantni paralelizam je svojstvo kvantnih računala da izvrše neku operaciju nad više mogućih ulaza odjednom. To svojstvo proizlazi iz prirode kvantnih bitova koja im omogućava da se nalaze u superpoziciji stanja. Za višestruku evaluaciju neke funkcije f, klasično računalo mora evaluirati f više puta za različite ulaze, no kvantno računalo može tu istu funkciju evaluirati samo jednom i dobiti vektor stanja koji je težinska superpozicija svih mogućih izlaza. Takvo svojstvo se možda na prvi pogled ne čini previše korisnim, ali postoje algoritmi i situacije gdje se takvo svojstvo pokazalo iznimno korisnim, ponajviše kada je bitno neko općenito svojstvo funkcije.



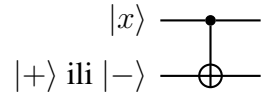
Slika 3.1: Postavljanje bitova u stanje superpozicije

Iz tog razloga, veliki broj kvantnih algoritama kao prvi korak ima postavljanje svih kvantnih bitova u stanje superpozicije korištenjem Hadamardovih operatora što se često označava operatorom $H^{\otimes N}$ gdje je N broj kvantnih bitova.

3.2. Prevrtanje faze

Prevrtanje faze je pojava kada ciljni bit CU operatora utječe na upravljački bit mijenjajući mu fazu. Javlja se kada je ciljni bit postavljen u svojstveno stanje unarnog

operatora i kada je upravljački bit u jedinici. U takvoj situaciji, upravljački bit će primiti fazu od ciljnog bita, tj. pomnožiti će se svojstvenom vrijednosti unarnog operatora koja odgovara svojstvenom stanju ciljnog bita. Pošto je unarni operator unitaran, njegove svojstvene vrijednosti će uvijek biti oblika $e^{i\phi}$ što utječe samo na fazu.



Slika 3.2: Prevrtanje faze

Na primjeru operatora CX, svojstvena stanja operatora X su $|+\rangle$ i $|-\rangle$ uz svojstvene vrijednosti ± 1. Jednostavnim izračunom se dobije:

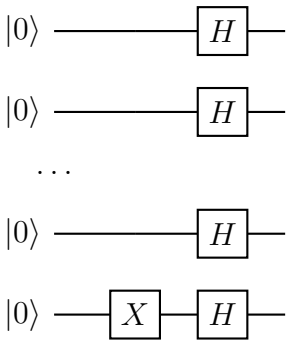
$$CNOT\,|0+\rangle = |0+\rangle \qquad CNOT\,|0-\rangle = |0-\rangle$$

$$CNOT\,|1+\rangle = 1 \cdot |1+\rangle \qquad CNOT\,|1-\rangle = -1 \cdot |1-\rangle$$

$$CNOT\,|++\rangle = |++\rangle \qquad CNOT\,|+-\rangle = |--\rangle$$

$$CNOT\,|-+\rangle = |-+\rangle \qquad CNOT\,|--\rangle = |+-\rangle$$

Prevrtanje faze je svojstvo koje se često koristi u kvantnim algoritmima zbog kojeg se neki bitovi inicijaliziraju u $|1\rangle$ prije primjene Hadamardovog operatora.



Slika 3.3: Česta inicijalizacija kvantnog logičkog kruga

No, u praksi stanje kvantnog sustava na početku kvantnog logičkog kruga uvijek bude inicijalizirano u $|00\ldots00\rangle$, pa je potrebno samo primijeniti operator X na kvantni bit koji treba biti u jedinici.

3.3. Deutschev algoritam

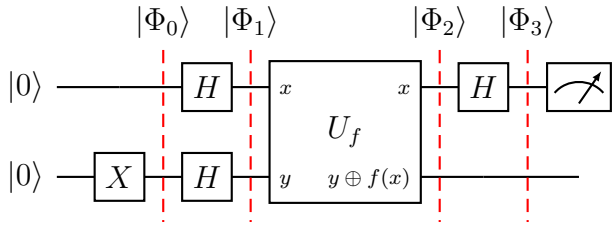
3.3.1. Opis

Deutschev algoritam jedan je od najjednostavnijih primjera kvantnog paralelizma koji demonstrira kvantnu nadmoć nad klasičnim računalom. Problem koji Deutschev algoritam rješava jest određivanje je li neka funkcija crne kutije oblika $f : \{0,1\} \to \{0,1\}$ *uravnotežena* ili *konstantna*. Postoje četiri takve funkcije:

$$f(x) = 0 \qquad f(x) = 1 \qquad f(x) = x \qquad f(x) = \neg x$$

gdje su prve dvije konstantne, a druge dvije uravnotežene. Za rješavanje ovog problema, klasično računalo treba evaluirati funkciju barem dva puta, dok na kvantnom računalu funkciju je dovoljno evaluirati samo jednom.

Kvantni logički krug Deutschevog algoritma prikazan je kao:



Slika 3.4: Kvantni logički krug Deutschevog algoritma

U_f je kvantna implementacija funkcije f za koju vrijedi:

$$U_f \left|x \otimes y\right\rangle = \left|x \otimes (y \oplus f(x))\right\rangle \qquad x, y \in \{0,1\}$$

Na kraju logičkog kruga, izmjerena vrijednost prvog kvantnog bita iznosi 0 za konstantne funkcije, a 1 za uravnotežene.

3.3.2. Analiza toka algoritma

Razlog takvom ishodu može se pronaći analizirajući tok algoritma. Prije primjene Hadamardovih vrata sustav se nalazi u stanju:

$$|\Phi_0\rangle = |0 \otimes 1\rangle$$

Nakon Hadamardovih vrata:

$$|\Phi_1\rangle = |+-\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |-\rangle = \frac{1}{\sqrt{2}}(|0-\rangle + |1-\rangle)$$

Primjenom U_f na stanje $|x-\rangle$ gdje je $x = \{0, 1\}$ dobiva se:

$$\begin{aligned} U_f |x-\rangle &= \frac{1}{\sqrt{2}}(U_f |x0\rangle - U_f |x1\rangle) \\ &= \frac{1}{\sqrt{2}}(|x\rangle \otimes |f(x)\rangle - |x\rangle \otimes |f(x) \oplus 1\rangle) \end{aligned}$$

Uvrštavanjem 0 i 1 umjesto $f(x)$ dobiva se:

$$U_f |x-\rangle = \begin{cases} \frac{1}{\sqrt{2}}(|x0\rangle - |x1\rangle) = |x-\rangle & \text{za} f(x) = 0 \\ \frac{1}{\sqrt{2}}(|x1\rangle - |x0\rangle) = -|x-\rangle & \text{za} f(x) = 1 \end{cases}$$

odnosno:

$$U_f |x-\rangle = (-1)^{f(x)} |x-\rangle$$

Sada, primjenom U_f na stanje $|\Phi_1\rangle$ dobije se $|\Phi_2\rangle$:

$$\begin{aligned} |\Phi_2\rangle = U_f |\Phi_1\rangle &= \frac{1}{\sqrt{2}}(U_f |0-\rangle + U_f |1-\rangle) \\ &= \frac{1}{\sqrt{2}}((-1)^{f(0)} |0-\rangle + (-1)^{f(1)} |1-\rangle) \\ &= \frac{(-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle}{\sqrt{2}} \otimes |-\rangle \end{aligned}$$

Očigledno je da su stanja separabilna, stoga se prvi bit može promatrati samostalno. Primjenom Hadamardovih vrata na prvi kvantni bit dobiva se konačno stanje:

$$\begin{aligned} |\Phi_3\rangle &= H \cdot \frac{(-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle}{\sqrt{2}} \\ &= \frac{(-1)^{f(0)} + (-1)^{f(1)}}{2} |0\rangle + \frac{(-1)^{f(0)} - (-1)^{f(1)}}{2} |1\rangle \end{aligned}$$

Iz jednadžbe se vidi da za konstantne funkcije vrijedi,

$$|\Phi_3\rangle = \pm |0\rangle$$

a za uravnotežene

$$|\Phi_3\rangle = \pm |1\rangle$$

Pošto faza nema utjecaja ne rezultate mjerenja, za konstantne funkcije rezultat mjerenja uvijek bude 0, a za uravnotežene 1.

3.3.3. Deutsch-Joszin algoritam

Postoji generalizacija ovoga algoritma pod nazivom Deutsch-Jozsin algoritam koji rješava isti problem, ali sa ulazom proizvoljnog broja bitova. U njemu je također potrebno

evaluirati funkciju samo jednom gdje će izlazni registar biti u nulama ako je funkcija konstantna, a bilo što drugo ako je uravnotežena. U njega ovaj rad neće ulaziti, ali je sličan i može se pogledati u [5]

Deutschev i Deutsch-Joszin algoritam dobro demonstriraju situaciju gdje je kvantno računalo puno efikasnije od klasičnog, ali za sada ne postoje neke korisne primjene tih algoritama.

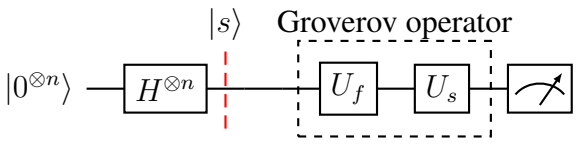
3.4. Groverov algoritam

3.4.1. Opis

Groverov algoritam[4] jedan je od algoritama koji je definirao principe kvantnih algoritama pretraživanja. U načelu, problem koji rješava jest pretraživanje nestrukturirane baze podataka. Klasično računalo taj problem rješava u prosječno $N/2$ koraka, dok kvantno računalo implementirajući Groverov algoritam pronađe rješenje s visokom vjerojatnošću u samo $\sqrt{N}$ koraka.

Definirati Groverov algoritam kao algoritam pretraživanja nestrukturirane baze podataka malo je zavaravajuće jer se u praktičnom smislu nikada ne bi mogao koristiti za točno to. Prikladnija primjena Groverovog algoritma bila bi pronalaženje inverza funkcije što čak i zvuči puno zanimljivijim i korisnijim. Jedna takva funkcija jest kriptografska funkcija sažetka. Groverov algoritam može pronaći kolizije takve funkcije u $\sqrt{N}$ koraka gdje je N veličina domene funkcije, što klasično računalo u načelu može izračunati samo grubom silom[1].

Kvantni logički krug Groverovog algoritma može se prikazati kao:



Slika 3.5: Kvantni logički krug Groverovog algoritma

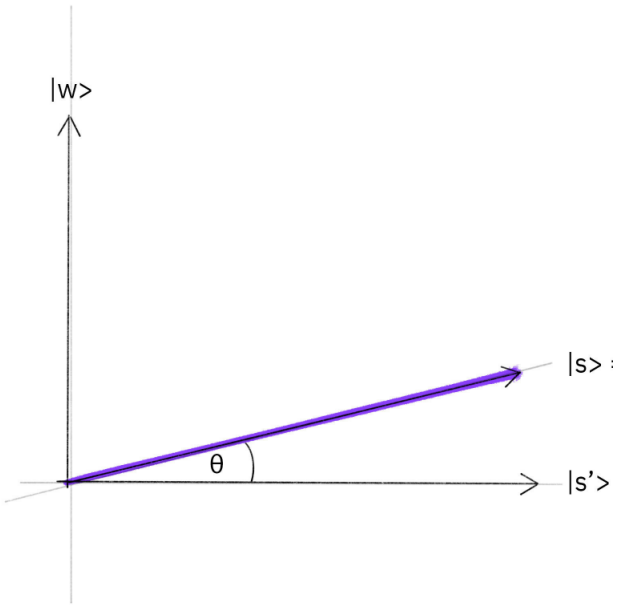
Groverov operator potrebno je primijeniti $\sqrt{N}$ puta gdje je $N = 2^n$, a n broj kvantnih bitova. Sastoji se od kvantnog proroka (engl. *quantum oracle*) U_f i difuzera (engl. *diffuser*) U_s. Kvantni prorok je jedinična matrica koja na mjestu jednog ili više elemenata kojeg tražimo umjesto jedinice ima -1. Difuzer je operator koji provodi

[1] Groverov algoritam se na neki način može interpretirati kao algoritam grube sile kvantnog računala, ali je kao takav i dalje eksponencionalno brži od klasičnog

operaciju $2\left|s\right\rangle\left\langle s\right| - I_{2^n}$. Uzastopnim primjenjivanjem ovih operatora, stanje sustava se približava ciljnom stanju $|w\rangle$ koje želimo pronaći. Mjerenjem sustava nakon $\sqrt{N}$ primjena Groverovog operatora dobiva se traženi element s dovoljno velikom vjerojatnošću.

3.4.2. Analiza toka algoritma

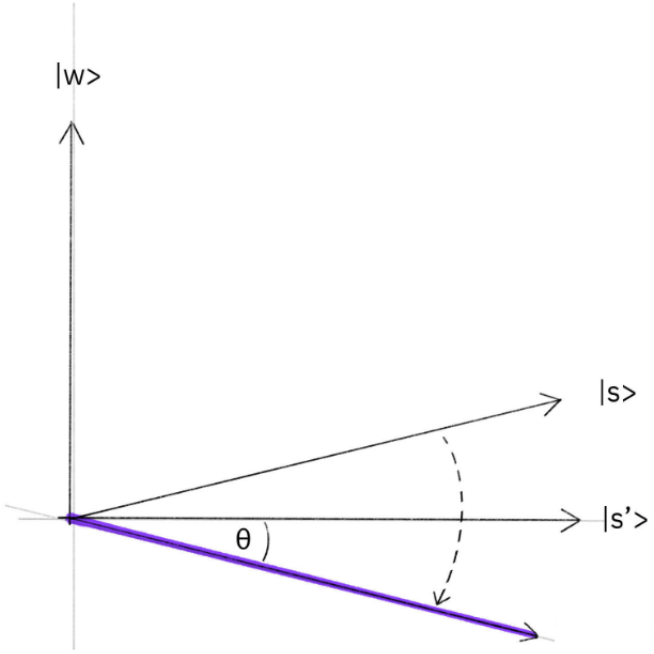
Algoritam započinje kao i mnogi drugi, postavljanjem svih bitova u stanje superpozicije primjenom Hadamardovih operatora čime dobivamo stanje $|s\rangle$. Neka se traženo stanje zove $|w\rangle$ koji može biti ciljno stanje ili superpozicija ciljnih stanja ako ih ima više. Neka se vektor stanja koji je okomit na stanje $|w\rangle$ zove $|s'\rangle$. Takvo stanje može se dobiti oduzimanjem stanja $|w\rangle$ od stanja $|s\rangle$ i normiranjem. Stanja $|w\rangle$, $|s\rangle$ i $|s'\rangle$ mogu se nacrtati u dvodimenzioalnom prostoru:



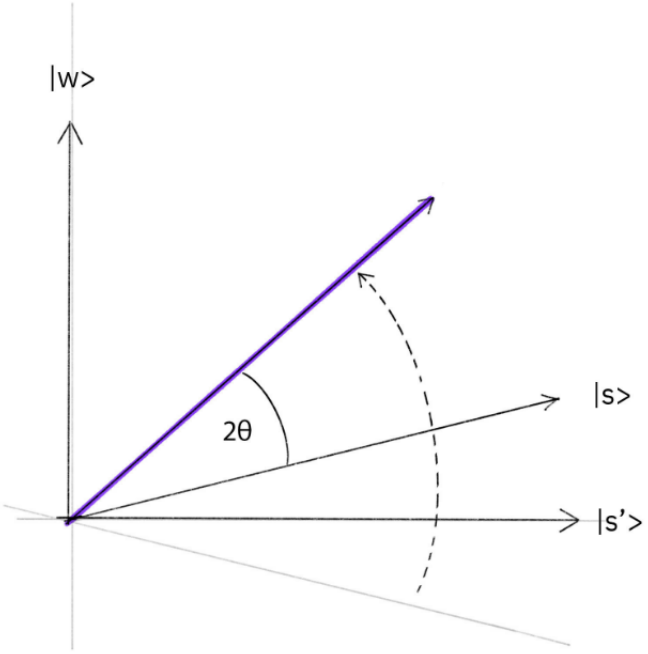
gdje kut θ opisuje koliko je stanje $|s\rangle$ zakrenuto prema $|w\rangle$. Za njega vrijedi:

$$\langle s|w\rangle = \frac{1}{\sqrt{N}} = \sin\theta \approx \theta$$

Idući korak algoritma primjenjuje kvantni prorok U_f na stanje $|s\rangle$. Sve što U_f radi jest refleksiju oko stanja $|s'\rangle$:



Zatim se primjenjuje difuzer U_s koji radi refleksiju oko stanja $|s\rangle$:



Dakle, nakon jedne primjene Groverovog operatora, stanje $|s\rangle$ se zaokrenulo za dodatnih 2θ prema ciljnom stanju $|w\rangle$. Nakon k primjena Groverovog operatora, stanje $|s\rangle$ biti će za $(2k+1)\theta$ zaokrenuto prema stanju $|w\rangle$. Cilj je da to bude što bliže stanju $|w\rangle$, dakle da vrijedi:

$$(2k+1)\theta \approx \frac{\pi}{2}$$

Rješavajući jednadžbu za k, dobiva se:

$$k \approx \frac{\pi}{4\theta} \approx \frac{\pi}{4}\sqrt{N} \approx \sqrt{N}$$

što znači da je potrebno primijeniti Groverov operator približno $\sqrt{N}$ puta kako bi vjerojatnost mjerenja stanja $|w\rangle$ bila najveća.

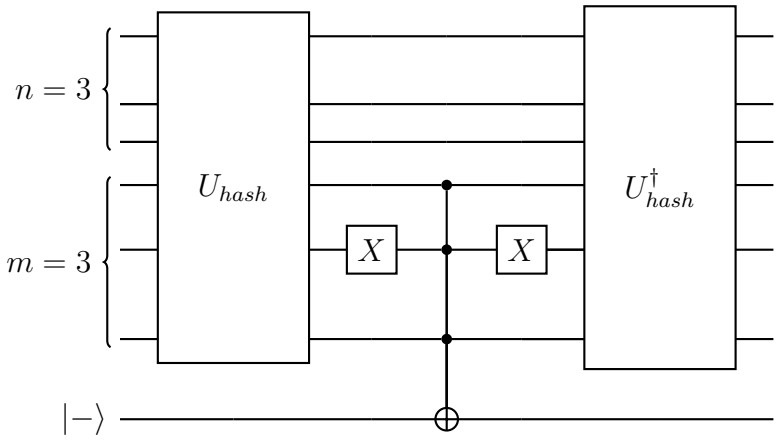
3.4.3. Kvantni prorok

Neka je U_f kvantni prorok Groverovog algoritma koji traži stanje $|10\rangle$. Njegova vrijednost iznosi:

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

Čini se kao da je za samu izgradnju logičkog kruga odnosno proroka potrebno poznavati rješenje koje tražimo što poništava cijeli smisao Groverovog algoritma. To je donekle i istina; poznavanjem matrične reprezentacije proroka, moguće je odrediti ciljna stanja, ili obratno, za konstrukciju proroka na ovakav način, potrebno je poznavati ciljna stanja. Iz toga slijedi da je Groverov algoritam koristan jedino kada se prorok tretira kao crna kutija ili ako se prorok konstruira na način gdje njegova vrijednost nije očita, ali da i dalje daje ispravan rezultat. Takva konstrukcija proroka postiže se koristeći prevrtanje faze.

Neka je f kriptografska funkcija sažetka za koju vrijedi $f : \{0,1\}^n \to \{0,1\}^m$ te neka je njoj potrebno pronaći inverz.



Slika 3.6: Primjer kvantnog proroka za pronalaženje inverza od $|101\rangle$

Za implementaciju kvantnog proroka potrebno je $n + m + 1$ kvantnih bitova od kojih su $m + 1$ pomoćni. Prvih n bitova se očekuje da su u stanju superpozicije, idućih m bitova se postavlja u stanje $|0\rangle$, dok se zadnji bit postavlja u stanje $|-\rangle$. Zatim je potrebno u kvantnom logičkom krugu implementirati samu funkciju f na način da joj je ulazni registar prvih n bitova, a izlazni idućih m bitova. Koristeći Toffolijeva vrata sa m upravljačkih bitova, može se odabrati izlaz funkcije kojemu je potrebno pronaći inverz (na slici je to $f(x) = 101$). Ciljni bit Toffolijevih vrata je poslijednji bit koji je u stanju $|-\rangle$. Svrha Toffolijevih vrata je svakoj komponenti stanja koja rezultira željenim izlazom promijeniti predznak. To radi pomoću prevrtanja faze. Naime, vrijedi:

$$CNOT\,|0-\rangle = |0-\rangle \qquad CNOT\,|1-\rangle = -\,|1-\rangle$$

Pošto se prorok koristi iterativno u algoritmu, potrebno je vratiti svih m izlaznih kvantnih bitova u stanje nule. To se može postići primjenom operatora $f^{\dagger}$, koji često zna biti jednak f.

3.5. Shorov algoritam

3.5.1. Opis problema

Shorov algoritam rješava problem faktorizacije velikih brojeva. To je problem na čiju se tešku izračunljivost oslanja veliki dio modernih kriptografskih mehanizama. Dobar primjer toga jest RSA kriptosustav čija se tajnost privatnog ključa temelji na težini računanja Eulerove funkcije koja se može svesti na problem faktorizacije brojeva. Iz algoritma generiranja ključeva i funkcija enkripcije i dekripcije može se vidjeti zašto je to tako.

Algoritam generiranja ključeva:

1. Generiranje broja $N = p \cdot q$, gdje su p i q veliki slučajno odabrani prosti brojevi
2. Računanje Eulerove funkcije[2] $\varphi(N) = (p-1)(q-1)$
3. Odabir proizvoljnog broja e iz reduciranog sustava ostataka modulo $\varphi(N)$
4. Računanje $d = e^{-1}$ u reduciranom sustavu ostataka modulo $\varphi(N)$
5. Javni ključ: pk = (e, N)

[2]Eulerova funkcija računa broj koji opisuje koliko relativno prostih faktora ima neki broj, tj. veličinu raduciranog sustava ostataka, no ovdje važnije svojstvo Eulerove funkcije jest da ako vrijedi $nzd(a, N) = 1$ onda isto vrijedi $a^{\varphi(N)} \equiv 1 \mod N$

6. Privatni ključ: sk = (d, N)

Funkcija enkripcije:

$$E(m,(e,N)) = m^e \mod N$$

Funkcija dekripcije:

$$D(c,(d,N)) = c^d \mod N$$

Ovo funkcionira jer vrijedi $e \cdot d = 1 \mod \varphi(N)$, tj.

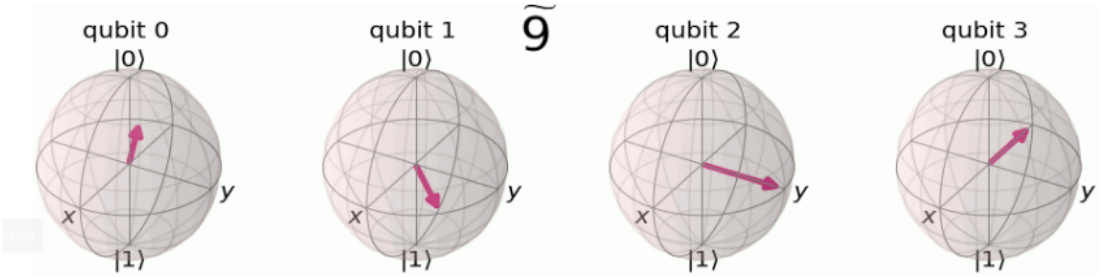
$$D(m^e,(d,N)) = m^{e \cdot d} \mod N = m \mod N$$

Dakle, kako bi se narušila sigurnost ovakvog sustava potrebno je moći efikasno izračunati proste faktore, odnosno $\varphi(N)$.

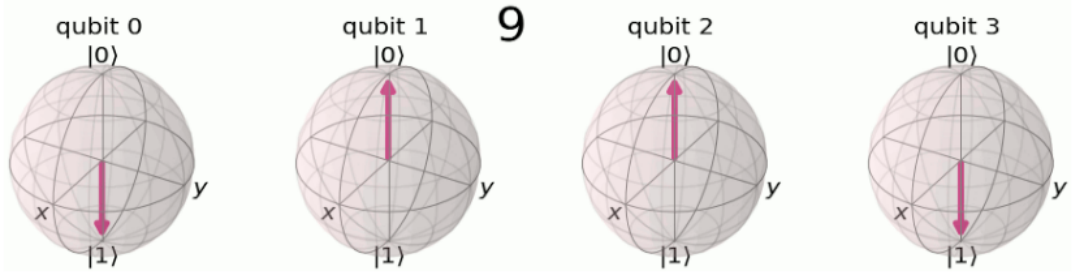
3.5.2. Kvantna Fourierova transformacija

Kvantna Fourierova transformacija vrši transformaciju baze računanja iz Z-baze (nazvane po osima Blochove sfere) u X-bazu koja se često zove Fourierovom bazom. Svaki broj koji se može prikazati stanjima $|0\rangle$ i $|1\rangle$ ima svoju reprezentaciju u Fourierovoj bazi zapisanu pomoću različitih rotacija oko osi Z.

Broj 0 u Fourierovoj bazi ima sve kvantne bitove postavljene u $|+\rangle$. Za svaki idući broj, najniži bit se rotira oko osi Z za $\frac{1}{2^n} \cdot 2\pi$ radijana, drugi najniži za $\frac{2}{2^n} \cdot 2\pi$ radijana, treći za $\frac{4}{2^n} \cdot 2\pi$ radijana itd. Na primjer, broj 9 ($|1001\rangle$ u Z-bazi) u Fourierovoj bazi izgleda:



Slika 3.7: Broj 9 u Fourierovoj bazi s 4 kvantna bita



Slika 3.8: Broj 9 u Z-bazi s 4 kvantna bita

Vidi se da je najniži kvantni bit (qubit 0) zaokrenut oko osi Z za $\frac{9}{16} \cdot 2\pi$ radijana, drugi najniži (qubit 1) za $\frac{18}{16} \cdot 2\pi = \frac{2}{16} \cdot 2\pi$ radijana, treći za $\frac{36}{16} \cdot 2\pi = \frac{4}{16} \cdot 2\pi$ radijana i poslijednji za $\frac{72}{16} \cdot 2\pi = \frac{8}{16} \cdot 2\pi$ radijana.

U matematičkom smislu, kvantna Fourierova transformacija vrši transformaciju vektora stanja $\sum_{i=0}^{N-1} x_i \left|i\right\rangle$ u vektor stanja $\sum_{i=0}^{N-1} y_i \left|i\right\rangle$ gdje vrijedi:

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

Unitarna matrica kvantne Fourierove transformacije može se prikazati kao:

$$U_{QFT} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \omega_N^{xy} \left|y\right\rangle \left\langle x\right| = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

gdje je $\omega_N^{jk} = e^{\frac{2\pi i}{n} \cdot jk}$. Hadamardov operator je najmanji U_{QFT}.

Općenito, može se izračunati:

$$\begin{aligned} U_{QFT} \left|x\right\rangle &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega_N^{xy} \left|y\right\rangle \\ &= \ldots = \frac{1}{\sqrt{N}} (\left|0\right\rangle + e^{\frac{2\pi i}{2} x} \left|1\right\rangle) \otimes (\left|0\right\rangle + e^{\frac{2\pi i}{2^2} x} \left|1\right\rangle) \otimes \ldots \otimes (\left|0\right\rangle + e^{\frac{2\pi i}{2^n} x} \left|1\right\rangle) \end{aligned}$$

Zbog unitarnosti operatora U_{QFT}, za transformaciju iz Fourierove baze u Z-bazu, potrebno je samo primijeniti operator $U_{QFT}^{\dagger}$.

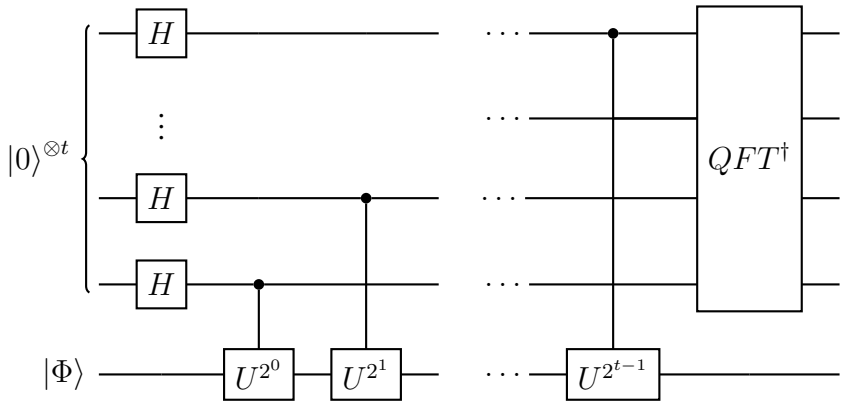
Postoji točno određen način kako se konstruira kvantni logički krug kvantne Fourierove transformacije, ali u sklopu izgradnje simulatora, dovoljno je poznavati samo njezin matrični oblik.

3.5.3. Kvantna procjena faze

Kvantna procjena faze je algoritam sam za sebe, ali je samo dio Shorovog algoritma. Neka je U unitarni operator sa svojstvenim vektorom $|\Phi\rangle$ i odgovarajućom svojstvenom vrijednosti $e^{2\pi i\theta}$.

$$U|\Phi\rangle = e^{2\pi i\theta}|\Phi\rangle$$

Algoritam procjene faze procjenjuje θ za dani U.



Slika 3.9: Kvantni logički krug kvantne procjene faze

Princip algoritma je zapisati fazu operatora U na prvih t bitova u Fourierovoj bazi koja se onda transformira u Z-bazu korištenjem inverza kvantne Fourierove transformacije što omogućuje njeno mjerenje. To se postiže prevrtanjem faze, tj. korištenjem upravljačkih U vrata i inicijalizacijom stanja $|\Phi\rangle$ u svojstveno stanje operatora U.

Pošto je zapis vrijednosti a u Fourierovoj bazi takav da k-ti bit napravi $\frac{a\cdot 2^k}{2^t}$ rotacija oko osi Z, potrebno je primijeniti točno t control-U^{2^k} operacija gdje $k \in \{0, 1, \ldots t-1\}$ kao što je prikazano na slici 3.9.

$$U^{2^k}|\Phi\rangle = U^{2k-1}U|\Phi\rangle = U^{2k-1}e^{2\pi i\theta} = \ldots = e^{2\pi i 2^k\theta}|\Phi\rangle$$

Nakon primjene svih k control-U operatora, stanje sustava jest:

$$|\Phi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\theta 2^0}|1\rangle) \otimes (|0\rangle + e^{2\pi i\theta 2^1}|1\rangle) \otimes \ldots \otimes (|0\rangle + e^{2\pi i\theta 2^{t-1}}|1\rangle) \otimes |\Phi\rangle$$

koje odgovara procjeni θ u Fourierovoj bazi tenzorski pomnoženo s $|\Phi\rangle$. Nakon toga se primjenjuje $QFT^\dagger$ i izvršava se mjerenje.

Ovaj algoritam samo procjenjuje fazu jer će dobiveni rezultat x označavati fazu $\frac{x}{2^t}$ što znači da će u pravilu procjena biti točnija što je t veći, ali je, naravno, za neke faze moguće dobiti potpuno precizan rezultat.

3.5.4. Opis algoritma

Shorov algoritam se temelji na činjenici da ako je moguće efikasno pronaći period od

$$f(x) = a^x \mod N$$

onda je moguće efikasno faktorizirati N. Iz tog razloga Shorov algoritam procjenjuje period unitarnog operatora U koji vrši operaciju:

$$U|y\rangle = |ay \mod N\rangle$$

Svaki svojstveni vektor operatora U može se zapisati kao:

$$|\Phi_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i s k}{r}} |a^k \mod N\rangle$$

Primjenom operatora U na jedno takvo svojstveno stanje dobije se:

$$U|\Phi_s\rangle = e^{\frac{2\pi i s}{r}} |\Phi_s\rangle$$

Koristeći algoritam procjene faze sa svojstvenim stanjem $|\Phi_s\rangle$ dobije se procjena faze $\frac{s}{r}$. No, konstrukcija svojstvenog stanja $|\Phi_s\rangle$ zahtjeva poznavanje r. Srećom, superpozicijom svih svojstvenih stanja $|\Phi_s\rangle$ dobiva se stanje $|1\rangle$.

$$\frac{1}{\sqrt{r}} \sum_{s=1}^{r-1} |\Phi_s\rangle = |1\rangle$$

Primjenjujući algoritam procjene faze sa svojstvenim stanjem $|1\rangle$, zapisana faza biti će superpozicija svih faza oblika $\frac{s}{r}$ te će se mjerenjem izmjeriti samo jedna od njih. Ovo je odlično svojstvo jer ono što je zapravo bitno jest r za koji vrijedi:

$$a^r \mod N = 1 \qquad r \mid \varphi(N)$$

Algoritam se sastoji od dijela gdje se operacije mogu izvoditi na klasičnom računalu, i kvantnog dijela koji je samo opisana kvantna procjena faze funkcije. Algoritam je kako slijedi:

1. Odabire se nasumičan broj $1 < a < N$
2. Provjerava se je li a relativno prost s N, ako nije, pronađen je faktor, algoritam završava
3. Koristi se kvantna procjena faze unitarnog operatora za kojega vrijedi $U|y\rangle = |ay \mod N\rangle$

4. Dobivena faza $\frac{x}{2^t}$ se reducira (aproksimira) na $\frac{s}{r}$ gdje je $r < N$ i cijeli broj. Ako se odmah ne dobije r za koji vrijedi $a^r \mod N = 1$, ovaj korak treba ponoviti nekoliko puta. Ako se i dalje ne dobije zadovoljavajući r, onda treba ponovno početi od koraka 1

5. Ako je r neparan ili $a^{r/2} \equiv -1(\mod N)$, povratak na korak 1

6. Inače, $nzd(a^{\frac{r}{2}} + 1, N)$ i $nzd(a^{\frac{r}{2}} - 1, N)$ su netrivijalni faktori od N

Funkcija $nzd(a, b)$ označava najmanjeg zajedničkog djelitelja od a i b.

4. Simulacija kvantnog računala

4.1. Postojeći simulatori kvantnog računala

Danas postoje biblioteke i *toolkits* za simulaciju kvantnih računala kao što su Qiskit, QuTiP, staq ili neki od brojnih drugih od kojih se veliki broj može pronaći na [2]. Mnogi nude razne funkcionalnosti kao što su analiza tijeka izvođenja logičkog kruga ili razne načine vizualizacije stanja sustava, ali isto tako znaju imati zbunjujuću dokumentaciju i neintuitivan način korištenja.

Qiskit, s druge strane, ima vrhunsku dokumentaciju i uglavnom vrlo praktično sučelje. Čak dopušta programiranje i izvršavanje kvantnih logičkih krugova na pravim IBM-ovim kvantnim računalima za koje je često potrebno čekati u redu za korištenje. Također nudi razne vizualizacije kvantnog logičkog kruga i stanja sustava, a uz korištenje nekih drugih Python biblioteka kao što je matplotlib, moguće je dodatno vizualizirati rezultate izvođenja.

Samostalno izrađen simulator u sklopu ovoga rada ima sučelje slično Qiskitu, no napisan je u jeziku C++ te je dizajniran da što jednostavnije omogući demonstraciju nekih od prethodno opisanih pojava i algoritama u ovom radu.

4.2. Izrada simulatora kvantnog računala

4.2.1. SQS

Simple Quantum Simulator ili **SQS** osmišljen je kao *header-only* biblioteka u jeziku C++. Kao takav je neovisan o platformi uz kompromis duljeg vremena prevođenja. Za operacije s vektorima i matricama SQS koristi biblioteku Eigen [1] koja se također sastoji od isključivo *header* datoteka što dodatno produljuje vrijeme prevođenja, ali je ono i dalje prihvatljivo. Za korištenje biblioteke potrebno je u zaglavlje programa staviti *include* datoteke *sqs.h*.

4.2.2. Struktura

SQS se temelji na tri glavne komponente koje se koriste za konstrukciju kvantnih logičkih krugova. To su **QOperator**, **QComponent** i **QCircuit**.

QOperator

Razred QOperator predstavlja kvantni operator nad jednim ili više kvantnih bitova. Enkapsulira matricu operatora i sadrži informaciju o tipu operatora koji pomaže objektu tipa QComponent integrirati ga u logički krug. Korisnik ne mora brinuti o tipu operatora. Za instanciranje QOperatora korisniku su na raspolaganju konstante i funkcije za konstruiranje često korištenih kvantnih operatora. Konstante su redom: *Eye*, *Hadamard*, *PauliX*, *PauliY*, *PauliZ*, *CX*, *CY*, *CZ*, *Toffoli* i *SWAP*. Od funkcija to su:

```
QOperator CU(size_t controls, QOperator unitary);
QOperator Phase(double phase);
QOperator QFT(unsigned int qubits);
QOperator QFTDagger(unsigned int qubits);
```

Funkcija *CU* prima dva argumenta: koliko ima upravljačkih bitova te unitarni operator kojim upravljaju. *Phase* prima fazu operatora, a *QFT* i *QFTDagger* primaju nad koliko kvantnih bitova djeluju. Sve navedene konstante i funkcije nalaze se u datoteci *ops.hpp* biblioteke. Matrice konstanti se nalaze u datoteci *eigenconsts.hpp*, ali su donekle sakrivene od korisnika koristeći namespace *sqs::Private*.

Korisnik također može stvoriti vlastiti QOperator inicijalizirajući ga s Eigen matricom u konstruktoru ili koristeći linearnu kombinaciju postojećih i vlastitih QOperatora i Eigen matrica. QOperator ne brine o svojoj unitarnosti te tu odgovornost ostavlja korisniku.

QComponent

QComponent je odgovoran za prvi korak integracije kvantnih operatora u kvantni logički krug. Funkcionira na način da se u njega dodavaju objekti tipa QOperator ili QComponent uz navedene indekse kvantnih bitova nad kojim djeluju.

```
void add(QOperator qop, std::vector<unsigned int> qubitPos);
void add(QOperator qop, unsigned int qPos);
void add(QOperator qop, unsigned int qPos1, unsigned int qPos2);
void add(QOperator qop, unsigned int qPos1, unsigned int qPos2,
    unsigned int qPos3);
```

```
void add(QComponent qcomp);
```

Ukoliko se radi o operatoru s upravljačkim bitovima, potrebno je njih prve navesti, neovisno kojim redoslijedom.

QComponent nastoji što više operatora staviti u paralelu što smanjuje količinu matrica koje je kasnije potrebno izračunati u koracima simulacije. Od ostalih funkcionalnosti, QComponent nudi:

```
void setIterations(unsigned int times);
std::vector<unsigned int> getQubitRange();
const MX& calculateMatrix();
```

setIterations postavlja koliko puta se komponenta treba ponoviti u logičkom krugu što je korisno za neke algoritme poput Groverovog, *getQubitRange* računa koje kvantne bitove komponenta koristi, a *calculateMatrix* računa matrični prikaz komponente. Zadnje dvije funkcije koristi QCircuit kako bi uspješno simulirao logički krug. *MX* i ostali nestandardni tipovi koji se koriste su definirani u datoteci *eigenconsts.hpp* i predstavljaju samo kraći zapis tipova Eigen matrica.

QCircuit

QCircuit predstavlja kvantni logički krug u kojeg se ugrađuju komponente koje nastoji što je više moguće paralelizirati. Pri stvaranju objekta tipa QCircuit, u konstruktoru je potrebno navesti broj kvantnih bitova. Funkcionalnosti koje QCircuit nudi su:

```
void add(QComponent qcomp);
void add(QOperator qop, std::vector<unsigned int> qubitPos);
void add(QOperator qop, unsigned int qPos1);
void add(QOperator qop, unsigned int qPos1, unsigned int qPos2);
void add(QOperator qop, unsigned int qPos1, unsigned int qPos2,
    unsigned int qPos3);
void execute();
void resetQubits();
void clearCircuit();
VX getStateVector();
std::.vector<double> probabilityVector();
std::map<unsigned int, unsigned int> measure(unsigned int times,
    unsigned int bits);
std::map<unsigned int, unsigned int> measure(unsigned int times);
```

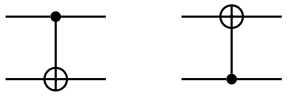
```
void measureAndDisplay(unsigned int times, unsigned int bits);
void measureAndDisplay(unsigned int times);
```

execute pokreće simulaciju kvantnog logičkog kruga, odnosno računa sve potrebne matrice te ih množi vektorom stanja koji je na početku inicijaliziran u $|0\rangle$. *resetQubits* postavlja vektor stanja u početno stanje. *clearCircuit* uklanja sve komponente iz logičkog kruga. *getStateVector* vraća trenutni vektor stanja logičkog kruga, dok *probabilityVector* vraća vektor vjerojatnosti. *measure* vrši mjerenje onoliko puta koliko je navedeno prvim argumentom nad svim bitovima ili nad prvih onoliko bitova koliko je navedeno u drugom argumentu te vraća rezultate mjerenja kao mapu. *measureAndDisplay* radi isto što i *measure*, samo što ne vraća mapu nego odmah prikazuje rezultate mjerenja na standardnom izlazu.

4.2.3. Izazovi pri implementaciji

U suštini, simulator se može svesti na računanje tenzorskog produkta i matrično množenje, no situacija ipak nije toliko jednostavna. Sve matrične reprezentacije kvantnih operatora koji djeluju na dva ili više kvantnih bitova pretpostavljaju točno određen raspored bitova. Na primjer CNOT i Toffolijeva vrata pretpostavljaju da su svi bitovi jedan uz drugog i da su upravljački bitovi iznad ciljnog. To je vrlo ograničavajuće te je bilo potrebno pronaći način konstrukcije operatora koji djeluje na proizvoljno raspoređenim bitovima.

Za CU operator sa jednim upravljačkim bitom i jednim ciljnim bitom, to je uvijek moguće postići jer se on može rastaviti na zbroj tenzorskih produkata. Na primjer, CNOT operatori:



Slika 4.1: $CNOT_{1,2}$ i $CNOT_{2,1}$ operatori

mogu se prikazati kao:

$$CNOT_{1,2} = |0\rangle\langle 0| \otimes I_2 + |1\rangle\langle 1| \otimes \sigma_x \qquad CNOT_{2,1} = I_2 \otimes |0\rangle\langle 0| + \sigma_x \otimes |1\rangle\langle 1|$$

Općenito, ako se između upravljačkog i ciljnog bita nalazi n drugih operatora koji ne utječu na upravljački ili ciljni bit, zajednička matrica se može dobiti na način:

$$U = |0\rangle\langle 0| \otimes U_1 \otimes \ldots \otimes U_n \otimes I_2 + |1\rangle\langle 1| \otimes U_1 \otimes \ldots \otimes U_n \otimes \sigma_x$$

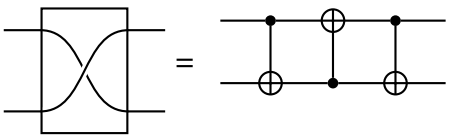
ili

$$U = I_2 \otimes U_1 \otimes \ldots \otimes U_n \otimes |0\rangle \langle 0| + \sigma_x \otimes U_1 \otimes \ldots \otimes U_n \otimes |1\rangle \langle 1|$$

gdje U_i mogu biti bilo koji unitarni operatori, a I_2 ako nema operatora.

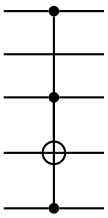
Ovo naizgled rješava samo dio problema, ali je zapravo jedina stvar koja je potrebna za konstrukciju svih ostalih operatora koji djeluju na proizvoljno raspoređenim bitovima.

Razlog tome je što se SWAP vrata mogu konstruirati od CNOT vrata. Vrijedi



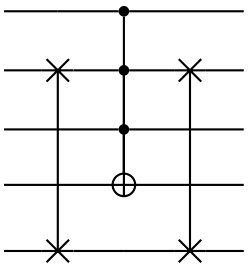
Slika 4.2: SWAP vrata prikazana pomoću CNOT vrata

Posljedica ove činjenice jest da je moguće zamijeniti bilo koja dva kvantna bita u logičkom krugu. Dakle, prije svake primjene višebitnog operatora, simulator napravi potrebne zamjene kako bi bitovi odgovarali ulazima operatora, bez da ikako mijenja matricu samog operatora. Takav način omogućuje korištenje logičkih vrata kao što su višeupravljačka Toffolijeva vrata:



Slika 4.3: Višeupravljačka Toffolijeva vrata

Općenito matricu višeupravljačkog operatora je lagano konstruirati: svi elementi na dijagonali se postave u jedinicu, a u donji desni kut se postavi na matricu ciljnog operatora. Dakle, simulator bi vrata 4.3 konstruirao na način:

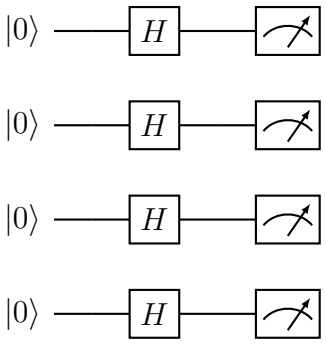


Slika 4.4: Realizacija vrata sa slike 4.3 u simulatoru

4.3. Primjeri simulacije kvantnih logičkih krugova

4.3.1. Superpozicija

Radi lakšeg upoznavanja sa simulatorom, prvih par primjera će biti jednostavni. Simulator započinje svoj rad u stanju $|0\rangle$, stoga za postavljanje bitova u superpoziciju potrebno je primijeniti Hadamardov operator na sve bitove. Mjerenje sustava trebalo bi rezultirati približno jednakom raspodjelom svih vrijednosti. Logički krug koji treba simulirati:



i njemu odgovarajući kod:

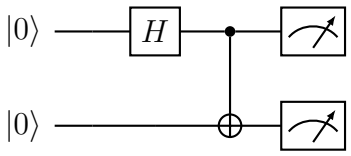
```
#include "sqs/sqs.h"

using namespace sqs;

int main() {
    /* stvaranje kvantnog logickog kruga */
    QCircuit qc(4);

    /* dodavanje Hadamardovog operatora na sve bitove,
        moguce napraviti i u for petlji */
    qc.add(Hadamard, {0, 1, 2, 3});

    /* pokretanje simulatora i mjerenje stanja na kraju 1000 puta */
    qc.execute();
    qc.measureAndDisplay(1000);

    return 0;
}
```

Dobiveni ispis:

```
|0000>: 63
|0001>: 56
|0010>: 64
|0011>: 62
|0100>: 61
|0101>: 68
|0110>: 59
|0111>: 68
|1000>: 55
|1001>: 72
|1010>: 50
|1011>: 82
|1100>: 59
|1101>: 63
|1110>: 62
|1111>: 56
```

4.3.2. Spregnutost

Spregnutost označava neseparabilno stanje kvantnih bitova te ga je iznenađujuće jednostavno dobiti; potrebna su samo dva operatora. Logički krug za dobivanje spregnutog sustava dva kvantna bita jest:



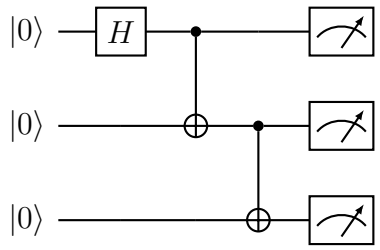

```
#include "sqs/sqs.h"

using namespace sqs;

int main() {

    QCircuit qc(2);
```

```
    /* dodavanje Hadamardovog operatora na qubit 0 */
    qc.add(Hadamard, 0);

    /* dodavanje CNOT operatora gdje je qubit 0 upravljacki,
       a 1 ciljni */
    qc.add(CX, 0, 1);

    qc.execute();
    qc.measureAndDisplay(1000);

    return 0;
}
```

Dobiveni ispis:

```
|00>: 498
|11>: 502
```

Za tri kvantna bita, primjer je sličan:



uz odgovarajući kod

```
    QCircuit qc(3);

    qc.add(Hadamard, 0);

    qc.add(CX, 0, 1);
    qc.add(CX, 1, 2);

    qc.execute();
    qc.measureAndDisplay(1000);
```

Dobiveni ispis:

```
|000>:  487
|111>:  513
```

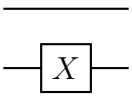
Idući primjeri izvornog koda uglavnom neće sadržavati elemente zaglavlja i funkcije main, nego samo relevantne dijelove.

4.3.3. Deutschev algoritam

Za Deutschev algoritam potrebno je implementirati funkciju oblika $f : \{0,1\} \rightarrow \{0,1\}$ te odrediti je li ona uravnotežena ili konstantna.

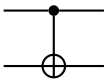
Za konstante funkcije, implementacija je takva da postaje jasno zašto se uvijek dobiva $|0\rangle$ kada se izmjeri prvi bit, odnosno prvih n bitova u Deutsch-Josza generalizaciji algoritma. Za funkciju $f(x) = 0$ implementacija se sastoji od praznih žica, odnosno jediničnih matrica, dok $f(x) = 1$ ima σ_x operator na izlaznom bitu. Očigledno je da izlazni bit uopće ne interagira sa ulazom što rezultira da ulaz na kraju evaluacije crne kutije ostaje nepromijenjen.

Slika 4.5: $f(x) = 0$

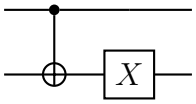


Slika 4.6: $f(x) = 1$

Implementacije uravnoteženih funkcija također nisu kompliciranе:



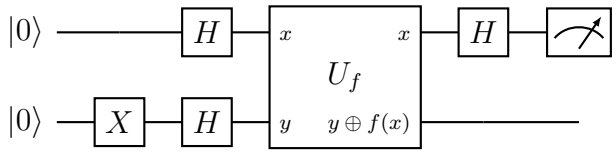
Slika 4.7: $f(x) = x$



Slika 4.8: $f(x) = \neg x$

Ovdje se vidi da će prevrtanje faze imati ključnu ulogu pošto je izlazni bit inicijaliziran u $|-\rangle$, a ulazni u $|+\rangle$. Primjenom CNOT operatora, izlazni bit će ulaznom bitu promijeniti fazu rezultirajući da će se ulazni bit pronaći u stanju $|-\rangle$ koje prolaskom kroz Hadamardov operator postaje stanjem $|1\rangle$.

Kvantni logički krug Deutschevog algoritma:



gdje U_f odgovara jednoj od spomenutih implementacija. Sve funkcije možemo testirati na način:

```
#include "sqs/sqs.h"

using namespace sqs;

void Deutsch(int i) {
    QCircuit qc(2);

    qc.add(PauliX, 1);
    qc.add(Hadamard, 0, 1);

    switch(i) {
        case 0: // f(x) = 0
            break;
        case 1: // f(x) = 1
            qc.add(PauliX, 1);
            break;
        case 2: // f(x) = x
            qc.add(CX, 0, 1);
            break;
        case 3: // f(x) = ~x
            qc.add(CX, 0, 1);
            qc.add(PauliX, 1);
            break;
        default:
            return;
    }

    qc.add(Hadamard, 0);

    qc.execute();
    qc.measureAndDisplay(1000);
}
```

```
int main() {

    std::string funcs[] = { "f(x) = 0", "f(x) = 1",
                            "f(x) = x", "f(x) = ~x" };

    for(int i = 0; i < 4; ++i) {
        std::cout << funcs[i] << std::endl;
        Deutsch(i);
    }

    return 0;
}
```

Dobije se očekivani rezultat:

```
f(x) = 0
|00>: 497
|10>: 503
f(x) = 1
|00>: 487
|10>: 513
f(x) = x
|01>: 508
|11>: 492
f(x) = ~x
|01>: 482
|11>: 518
```

Napomena: desni bit je rezultat algoritma; u skici kvantnog logičkog kruga, bit koji je najviši je u ispisu bit najmanje težine.

4.3.4. Groverov algoritam

Groverovim algoritmom moguće je rješavati SAT probleme. Problem koji je potrebno riješiti programira se unutar kvantnog proroka algoritma.

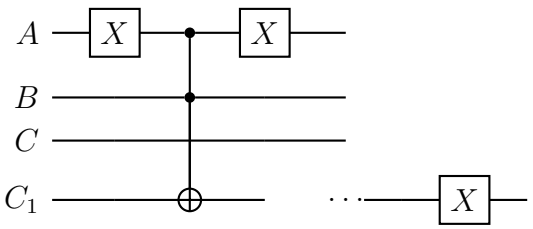
Neka je taj problem dan kao:

$$(A \vee \neg B) \wedge (B \vee C) \wedge (\neg A)$$

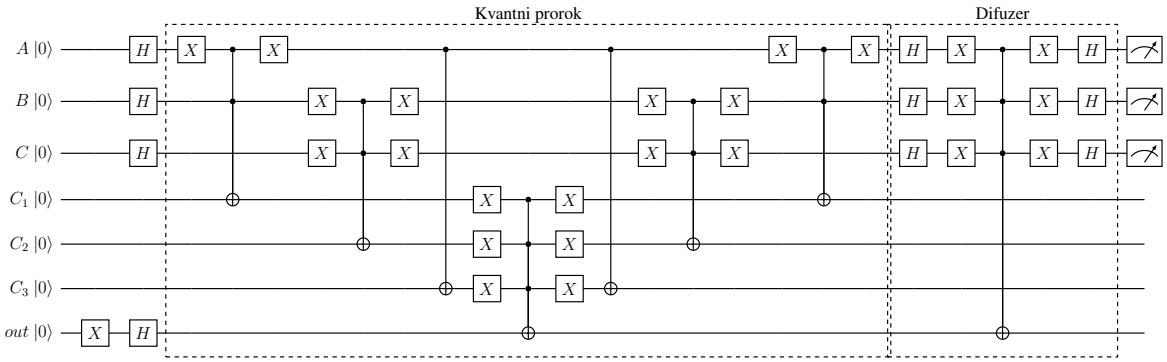
Korisno je da se taj izraz negira dva puta kako bi se dobio oblik sa isključivo operatorima logičkog I. Na taj način prorok je moguće izgraditi pomoću Toffolijevih vrata.

$$\neg(\neg A \wedge B) \wedge \neg(\neg B \wedge C) \wedge \neg(A)$$

Jedna ovakva klauzula može se prikazati kao:



Dakle, za svaku klauzulu potrebno je imati jedan dodatni kvantni bit kao i još jedan koji će provjeravati jesu li sve klauzule zadovoljene. Ta provjera služi kako bi se pomoću prevrtanja faze negirala sva ona stanja koja su rezultirala rješenjem. Nakon provjere potrebno je vratiti sve dodatne kvantne bitove u početno stanje, odnosno treba ponoviti sve klauzule obrnutim redoslijedom. Cijeli kvantni logički krug izgleda ovako:



Difuzer neće biti objašnjen, ali kao što je prethodno spomenuto, on izvodi operaciju $2\left|s\right\rangle\left\langle s\right| - I_{2^n}$ koja je zapravo samo refleksija vektora stanja oko uravnotežene superpozicije svih stanja $\left|s\right\rangle$. Izvedba kvantnog logičkog kruga u simulatoru:

```
QCircuit qc(7);

qc.add(PauliX, 6);
qc.add(Hadamard, {0, 1, 2, 6});

QComponent groverOperator;
```

```
QComponent oracle;
QComponent diffuser;

/* A or ~B */
oracle.add(PauliX, 0);
oracle.add(Toffoli, 0, 1, 3);
oracle.add(PauliX, 0);

/* B or C*/
oracle.add(PauliX, 1, 2);
oracle.add(Toffoli, 1, 2, 4);
oracle.add(PauliX, 1, 2);

/* ~A */
oracle.add(CX, 0, 5);

oracle.add(PauliX, 3, 4, 5);
oracle.add(CU(3, PauliX), {3, 4, 5, 6});
oracle.add(PauliX, 3, 4, 5);

/* obrnuti smjer */
oracle.add(CX, 0, 5);
oracle.add(PauliX, 1, 2);
oracle.add(Toffoli, 1, 2, 4);
oracle.add(PauliX, 1, 2);
oracle.add(PauliX, 0);
oracle.add(Toffoli, 0, 1, 3);
oracle.add(PauliX, 0);


/* difuzer */
diffuser.add(Hadamard, 0, 1, 2);
diffuser.add(PauliX, 0, 1, 2);
diffuser.add(CU(3, PauliX), {0, 1, 2, 6});
diffuser.add(PauliX, 0, 1, 2);
diffuser.add(Hadamard, 0, 1, 2);
```

```
groverOperator.add(oracle);
groverOperator.add(diffuser);

qc.add(groverOperator);
qc.execute();
qc.measureAndDisplay(1000, 3);
```

Groverov operator se u ovom primjeru primijenio samo jednom te je dobiveni rezultat:

```
|000>: 71
|001>: 81
|010>: 73
|011>: 68
|100>: 486
|101>: 68
|110>: 74
|111>: 79
```

Vidi se da rezultat $|100\rangle$ ima najveću vjerojatnost mjerenja, te odgovara rješenju

$$A = False \qquad B = False \qquad C = True$$

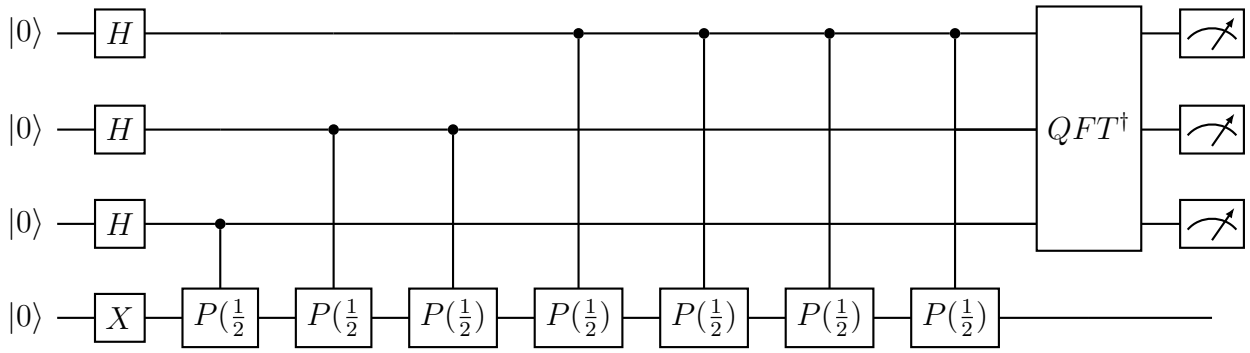
Komponentu groverOperator moguće je primijeniti više puta pozivom funkcije *QComponent::setIterations(unsigned int times)* te bi se tada dobila veća vjerojatnost mjerenja točnog rezultata.

4.3.5. Kvantna estimacija faze

Ključan dio Shorovog algoritma je algoritam kvantne estimacije faze. U ovom konkretnom primjeru, estimirati će se faza samog operatora faze P. Operator faze ima oblik:

$$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix}$$

stoga će kvantna estimacija faze izračunati $\frac{\varphi}{2}$. Za primjer, uzeti će se $\varphi = \frac{1}{2}$. Kvantni logički krug:



Odgovarajući kod:

```
QCircuit qc(4);
qc.add(PauliX, 3);
qc.add(Hadamard, {0, 1, 2});

QOperator ph = Phase(1.0 / 2.0); /* pi/2 = 2pi * (1/4) */
QOperator cPh = CU(1, ph);

qc.add(cPh, 0, 3);
qc.add(cPh, 0, 3);
qc.add(cPh, 0, 3);
qc.add(cPh, 0, 3);
qc.add(cPh, 1, 3);
qc.add(cPh, 1, 3);
qc.add(cPh, 2, 3);

qc.add(QFTDagger(3), {0, 1, 2});
qc.execute();

auto m = qc.measure(1000, 3);
for(auto a : m) {
    std::cout << a.first << "/8"
        << "\tp: " << a.second / 10.0 << "%\n";
}
```

Dobiveni rezultat:

```
2/8     p: 100%
```

odgovara očekivanjima, naime $\frac{2}{8} = \frac{1}{4} = \frac{1/2}{2} = \frac{\varphi}{2}$.

4.4. Prednosti i mane simulatora kvantnog računala

Bilo koji simulator kvantnog računala nikada neće moći nadomjestiti pravo kvantno računalo što je posljedica same njegove prirode. Sustav od n kvantnih bitova klasično računalo prikazuje vektorom dimenzije 2^n, a svaki operator koji djeluje na taj vektor matricom dimenzije $2^n \times 2^n$. Uz problem pohrane takvih podataka za veliki broj kvantnih bitova, još je veći problem vrijeme izvršavanja operacija koje uključuju takve operande. Postoji fizička granica kada klasična računala više ne mogu u razumnom vremenu računati što se odvija u nekom kvantnom sustavu[1].

No, simulatori su daleko od toga da su beskorisni. Sama činjenica što su simulatori, a ne kvantni sustavi može pomoći pri analizi određenih svojstava kvantnih sustava. Simulator je u svakom trenutku moguće zaustaviti i analizirati stanje sustava bez da se naruši to stanje. Postoje neka ograničenja kvantnih računala koja ne postoji na klasičnim, kao što je na primjer nemogućnost kloniranja kvantnog bita ili nemogućnost višestrukog mjerenja sustava. Također je bitno spomenuti da u simulatorima ne postoji[2] pojava dekoherencije koja smanjuje preciznost računanja u pravim kvantnim računalima.

U svakom slučaju, simulatori su odlični alati za eksperimentiranje, analizu i bolje shvaćanje kvantnih sustava, pogotovo ako pravo kvantno računalo nije lako dostupno.

[1] Pojam kvantna nadmoć (engl. *quantum supremacy*) se odnosi točno na ovu granicu.

[2] Osim ako to namjerno nije simulirano

5. Zaključak

Potencijal kvantnog računala se tek nedavno počeo fizički ostvarivati i kao takav postaje sve bitnijim u području računarstva. Vrlo je vjerojatno da će taj potencijal donijeti mnoge promjene i napretke u svijet kao što se to dogodilo s klasičnim računalom, ali kvantno računalo nije tu zamijeniti klasično, već ponuditi nešto sasvim novo. Područje kvantnog računarstva još je uvijek relativno mlado i potrebno je uložiti puno truda u istraživanje i eksperimentiranje s novom tehnologijom i mogućnostima koje se otvaraju. Veliku pozornost treba posvetiti područjima umjetne inteligencije i strojnog učenja kojima se ovaj rad nije bavio, ali ih je bitno spomenuti.

Izgrađeni simulator u sklopu ovoga rada definitivno je moguće još unaprijediti i optimizirati; osmišljen je samo kao alat za demonstraciju nekih pojmova i algoritama i kao takav ima potencijala biti puno boljim te autor planira nastaviti raditi na njemu. S tim na umu, ovaj rad se bavio samo osnovama kvantnog računarstva te je za dublje shvaćanje potrebno dodatnog istraživanja.

LITERATURA

[1] Eigen. URL https://eigen.tuxfamily.org/.

[2] List of QC simulators | Quantiki. URL https://quantiki.org/wiki/list-qc-simulators.

[3] Adrian Cho. IBM promises 1000-qubit quantum computer—a milestone—by 2023, September 2020. URL https://www.sciencemag.org/news/2020/09/ibm-promises-1000-qubit-quantum-computer-milestone-2023.

[4] Lov K. Grover. A fast quantum mechanical algorithm for database search. U *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, stranica 212–219, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917855. doi: 10.1145/237814.237866. URL https://doi.org/10.1145/237814.237866.

[5] M.A. Nielsen i I.L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. ISBN 9781139495486.

[6] Peter W. Shor. Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Sci. Statist. Comput.*, 26:1484, 1997. doi: 10.1137/S0097539795293172.

Simulacija kvantnog računala

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

Quantum computer simulation

Abstract

Abstract.

Keywords: Keywords.