Univerza v Ljubljani

Optične in nanotehnologije
4.l. UNI-RI
2. seminarska naloga

Kvantno procesiranje

Dejan Štimpfelj, 63040162
Matjaž Okretič, 63040118
Miha Gorše, 63020046
Anže Leban, 63030312

Ljubljana, 12.1.2009

Kazalo

1 Uvod....3
2 Osnove kvantnega procesiranja....4
2.1 Uvod....4
2.2 Vektorji stanja in Bra/Ket notacija....4
2.3 Kvantni bit – qubit....5
2.4 Kvantni register....5
2.5 Kvantna logična vrata....6
3 Orodje jQuantum....7
3.1 Uvod....7
3.2 Zgradba delovnega okna....7
3.3 Inicializacija kvantnega registra....8
3.4 Načrtovanje vezja....8
3.5 Upravljanje algoritma....9
3.6 Logične strukture v jQuantum....9
3.6.1 Inicializacija....9
3.6.2 Hadamardova vrata....10
3.6.3 c-NOT vrata....10
3.6.4 Vrata X....11
3.6.5 Vrata Y....11
3.6.6 Vrata Z....11
3.6.7 Vrata S....11
3.6.8 Vrata S+....12
3.6.9 Vrata T....12
3.6.10 Vrata √X....12
3.6.11 Toffolijeva vrata....13
3.6.12 Fouriereva vrata....13
3.6.13 Inverzna Fouriereva vrata....13
3.6.14 Funkcijska vrata....13
3.6.15 Rotacija....14
3.6.16 Groverjev operator....14
3.6.17 Meritev....14
3.6.18 Brisanje....14
4 Predstavitev osnovnih algoritmov....15
4.1 Klasična vezja....15
4.1.1 AND vrata....15
4.1.2 OR vrata....16
4.1.3 Seštevalnik....16
4.2 Osnovna neklasična vezja – algoritmi....17
4.2.1 Gooverjev algoritem....17
4.2.2 Shorov algoritem za iskanje prafaktorjev števil....18
5 Zaključek....20
6 Literatura....21
7 Viri....21

1 Uvod

Kvantna fizika se je začela razvijati ob prelomu 20. stoletja kot odgovor na nezmožnost klasične fizike, da razloži rezultate eksperimentov s svetlobo in osnovnimi delci. Kljub težavam in neodobravanju s strani pristašev klasične fizike, so kvantni fiziki z eksperimenti dokazovali svoje teorije, ne glede na to kako neverjetne so se takrat le te zdele. Kljub dolgoletnemu delu na področju kvantne fizike, je to še vedno v veliki meri neraziskano. Težava pri razumevanju kvantne fizike je, da je potrebno „pozabiti“ veliko večino tega, kar vemo o naravi v realnosti, saj so pojavi v kvantni fiziki in razlage le teh velikokrat v nasprotju z „zdravo pametjo“. Kljub vsem težavam je kvantna fizika še vedno najboljše orodje za razumevanje pojavov v mikro svetu in s tem delovanje kvantnih računalnikov.

Znanstveniki so že v 80. letih 20. stoletja ugotovili, da se določeni kvantni pojavi ne morejo simulirati na klasičnih računalnikih. Ta opažanja so privedla do ideje, da je možno z uporabo kvantnih pojavov pohitriti računanje in zato so se kmalu začeli razvijati kvantni računalniki. To so računalniki, ki za svoje delovanje uporabljajo kvante pojave. Ker pa je bil razvoj takih računalnikov zelo kompleksen in ker ni bilo osnovne ideje, kako točno bi pospešili računanje z uporabo kvantih pojavov, je področje razvoja kvantnih računalnikov napredovalo zelo počasi. Tako je bilo vse do leta 1994, ko je Peter Shor predstavil kvantni algoritem za faktorizacijo (iskanje prafaktorjev števil). Odkritje je bilo prelomno, saj je vzpodbudilo razvoj na področju razvoja kvantnih računalnikov ter iskanju novih kvantnih algoritmov. Do sedaj je trud na tem področju obrodil veliko sadov. Najbolj znani so predstavitev distribucija kvantnega ključa, kvantne teleportacije ter tri-bitnega kvantnega računalnika.

V tej seminarski nalogi bomo predstavili osnove kvantnega procesiranja ter uporabo teh konceptov v simulatorju jQuantum, ki ga bomo podrobneje predstavili. Na koncu bomo predstavili tudi nekaj osnovnih algoritmov, ki so značilni za procesiranje na kvantnih računalnikih.

2 Osnove kvantnega procesiranja

2.1 Uvod

Pri kvantnem procesiranju se za povečanje hitrosti računanja napram klasičnem procesiranju uporablja kvantne pojave. Le ti omogočajo visok nivo paralelizma pri računanju v kvantnih sistemih.
V splošnem lahko rečemo, da se čas reševanja nekega problema lahko zmanjša z uporabo paralelnega procesiranja. Pri klasičnih računalnikih to pomeni uporabo več procesorjev, ki so povezani v računalniški sistem. Za eksponentno zmanjšanje časa reševanja problema, bi tako potrebovali tudi eksponentno število procesorjev in posledično tudi eksponentno porabo fizičnega prostora.
Pri kvantnih računalnikih pa stopnja paralelizma narašča s samo velikostjo sistema. To pomeni, da za eksponentno zmanjšanje časa reševanja problema potrebujemo linearno porabo fizičnega prostora. To omogoča dejstvo, da se v kvantnem sistemu operacija hkrati izvede nad vsemi možnimi stanji, ki jih sistem lahko hrani, in ne le nad enim stanjem, kot je to pri klasičnih računalnikih. Ta efekt je poznan pod imenom kvantni paralelizem.

Toda pri kvantnih računalnikih ni vse tako lepo kot zgleda. Res je, da le ti lahko izvajajo masovne paralelne izračune, toda težava se pojavi pri samem branju rezultata. Dostop do rezultatov računanja je ekvivalenten meritvi, ki zmoti kvantno stanje. To pomeni, da lahko hkrati beremo rezultate le ene vzporedne niti in ker je merjenje problematično, ne moremo točno izbrati, katero bomo prebrali. Poleg tega se po branju spremeni kvantno stanje kvantnega sistema.

2.2 Vektorji stanja in Bra/Ket notacija

V kvantni fiziki je stanje kvantnega sistema definirano z vektorjem v Hilbertovem prostoru. To pomeni, da je predstavitev fizičnega stanja odvisna od neodvisnih koordinat, ki si jih lahko predstavljamo kot komplet definiranih osi, kjer posamična os predstavlja možno stanje, v katerem se lahko kvantni sistem nahaja. Projekcija vektorja na osi pokaže prispevek posamičnega možnega stanja h končnemu stanju. Analogijo temu lahko najdemo v komponentah klasičnega vektorja v Evklidskem prostoru. Pomembna razlika je v tem, da je Hilbertov prostor prostor kompleksnih linearnih vektorjev, zato so dolžine vektorjev v Hilbertovem prostoru kompleksna števila. Vektorji stanja (ang. state vector) kvantnega sistema so običajno zapisani v posebni, t.i. *ket* notaciji $|\psi\rangle$, ki označuje stolpčni vektor. *Bra* notacija $\langle\psi|$ pa predstavlja inverz vektorja $|\psi\rangle$. Produkt stolpčnega in vrstičnega vektorja torej lahko zapišemo kot $\langle\psi \mid \psi\rangle$ in tako dobimo *bracket* notacijo. Razlog za vpeljavo te notacije je, da je veliko formul v kvantni fiziki sestavljenih iz množenj stolpčnih in vrstičnih vektorjev in s to notacijo tako dosežemo kompaktni zapis enačb. Podobno kot pri klasičnih vektorjih, je tudi vektor stanja definiran z izborom baznih vektorjev, ki predstavljajo možna stanja in nizom kompleksih števil, ki določajo prispevek posamičnega možnega stanja h končnemu. Preprosti 2-stanjski sistem je lahko po definiciji v enem od 2 stanj. Posledično ima tak sistem 2 možni stanji in vektor stanj ima 2 komponenti. Kočno stanje lahko tako zapišemo kot:

$$|\psi\rangle = a^*|\psi_0\rangle + b^*|\psi_1\rangle = \begin{pmatrix} a \\ b \end{pmatrix}$$

kjer uteži a in b predstavljata kompleksni števili in možni stanji ψ_i tvorita popolno ortagonalno bazo za vektor stanja $|\psi\rangle$. Z možnimi stanji ψ_i je lahko predstavljen vsak vektor stanj v Hilberovem prostoru, možna stanja pa sestavljajo sistem osi v Hilbertovem prostoru.
Ko enkrat poznamo vektor stanja $|\psi\rangle$, lahko izračunamo pričakovano vrednost kateregakoli opazovanega atributa v sistemu. To pomeni, da vektor stanj vsebuje popolno informacijo o sistemu, katerega del je.

2.3 Kvantni bit – qubit

V klasičnih računalnikih je osnovna enota pomnjenja bit. Le ta lahko zaseda 2 logični vrednosti (logično 0 in logično 1), sama vrednost pa se izraža z napetostnim nivojem. Če združimo n bitov, dobimo n-bitni register, v katerega lahko shranimo 2^n možnih vrednosti. Klasičen register lahko v nekem trenutku hrani le eno vrednost.

V kvantnih računalnikih je osnovna enota pomnjenja kvantni bit oziroma qubit. Le ta se lahko nahaja v dveh osnovnih kvantnih stanjih, ki ju označimo z $|0\rangle$ ter $|1\rangle$ in sta ekvivalentna logični 0 in logični 1 v klasičnih računalnikih, ali v t.i. stanju superpozicije (v obeh osnovnih kvantnih stanjih hkrati). Osnovni stanji $|0\rangle$ter $|1\rangle$ tvorita računsko osnovo, tako da lahko vsa stabilna stanja, v katerih je lahko qubit, zapišemo v superpoziciji:

$|q\rangle = a|0\rangle + b|1\rangle$ za vse a ter b, za katere velja $|a|^2 + |b|^2 = 1$

$a = x_0 + i*y_0$

$b = x_1 + i*y_1$

Vrednosti qubitov so v današnjih implementacijah predstavljeni z nivojem naboja elektrona, spinom elektrona ali pa polarizacijo fotona.

2.4 Kvantni register

Če združimo n qubitov, potem dobimo n-bitni kvantni register, v katerega lahko shranimo 2^n možnih vrednosti. Kvantni register lahko v nekem trenutku hrani 2^n vrednosti. To mu daje veliko prednost pred klasičnim registrom.

Če predvidevamo, da je informacija v kvantnem registru zapisana v binarni obliki, potem lahko na primer število 6 zapišemo kot $|1\rangle ¤ |1\rangle ¤ |0\rangle$, oziroma število 7 kot $|1\rangle ¤ |1\rangle ¤ |1\rangle$. Zaradi bolj kompaktne notacije lahko rečemo, da je $|q\rangle$ enak tenzorskemu produktu osnovnih stanj q_i, kjer je $q_i \in \{0,1\}$.

$$|q\rangle = |q_{n-1}\rangle ¤ |q_{n-2}\rangle ¤ \ldots\ldots ¤ |q_1\rangle ¤ |q_0\rangle$$

S to enačbo je predstavljen kvantni register, v katerem je zapisana vrednost:

$$q = 2^0a_0 + 2^1a_1 + \ldots. 2^{n-2}a_{n-2} + 2^{n-1}a_{n-1}$$

V kvantni register lahko shranimo 2^n možnih stanj, ki predstavljajo binarno prezentacijo števil od 0 do 2^n-1. Na primer kvantni register velikosti 3 lahko shrani 8 posamičnih števil v obliki, ki je navedena zgoraj (zapis števil 6 in 7), lahko pa shrani obe števili istočasno. To je možno z uporabo superpozicije. Za to vzamemo prvi qubit in namesto, da bi ga nastavili na $|0\rangle$ ali $|1\rangle$, uporabimo superpozicijo $\frac{1}{\sqrt{2}} * (|0\rangle + |1\rangle)$ in dobimo:

$$\frac{1}{\sqrt{2}} * (|0\rangle + |1\rangle) \text{ ¤ } |1\rangle \text{ ¤ } |1\rangle = \frac{1}{\sqrt{2}} (|011\rangle + |111\rangle).$$

Lahko pa register uporabimo tako, da naenkrat hrani vseh 8 možnih vrednosti:

$$\frac{1}{\sqrt{2}} * (|0\rangle + |1\rangle) \text{ ¤ } \frac{1}{\sqrt{2}} *(|0\rangle + |1\rangle) \text{ ¤ } \frac{1}{\sqrt{2}} *(|0\rangle + |1\rangle) ,$$

kar lahko binarno zapišemo kot:

$$|000\rangle + |001\rangle + |010\rangle + |010\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle.$$

2.5 Kvantna logična vrata

Kvantna logična operacija je unitarna preslikava, kar pomeni, da je tudi reverzibilna (na podlagi izhoda lahko točno vemo kakšen je bil vhod). Posledica tega je tudi, da so kvantna logična vrata reverzibilna. Recimo, da unitarna matrika U opisuje transformacijo v kompleksnem vektorskem prostoru.

$|0\rangle = a\,|0\rangle + b\,|1\rangle$

$|1\rangle = c\,|0\rangle + d\,|1\rangle$ $\qquad U = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

Za U velja $U*U^T = U^T*U = I$

Reverzibilnost je tudi velika prednost napram klasičnim vratom, ki po večini niso reverzibilna. Klasičen primer nereverzibilnih klasičnih vrat sta OR in AND vrata.
Zanimiva lastnost računalnikov, ki so sestavljeni iz nereverzibilnih vrat je, da ob svojem delovanju trošijo energijo. Ta energija se sprosti v obliki toplote, ko se v nereverzibilnih vratih „uničuje“ informacija. Te sproščene energije pri reverzibilnih vratih ni, saj se v le teh informacija vedno ohranja. Znanstveno je dokazano, da se lahko z uporabo reverzibilnih vrat zgradi računalnik, ki ne bi imel toplotnih izgub.
V poglavju 4 bodo predstavljena osnovna kvantna logična vrata, ki se uporabljajo v kvantnih sistemih.

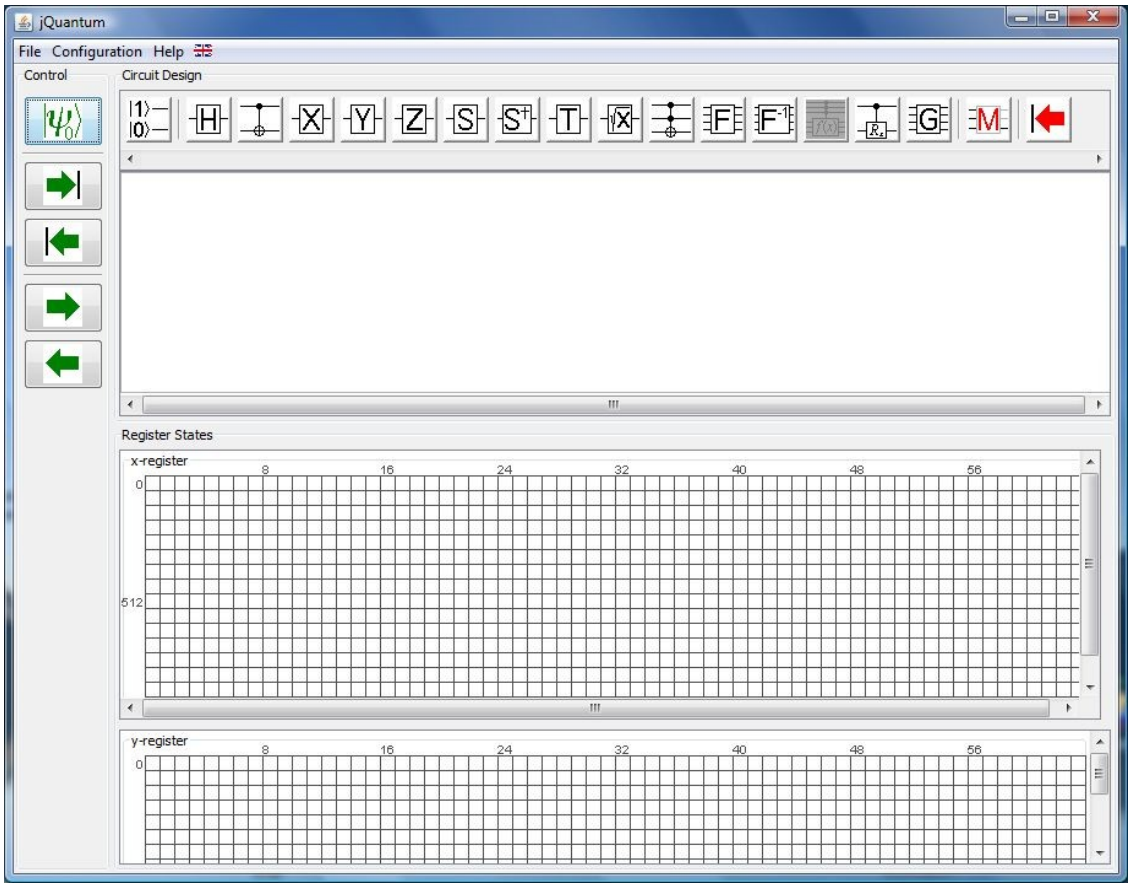
3 Orodje jQuantum

3.1 Uvod

jQuantum je program za simuliranje kvantnih algoritmov. Simulira kvantno vezje, rezultat pa prikaže v kvantnem registru velikosti do 15 qubitov. Glavni namen simulatorja je pomoč pri učenju in razumevanju kvantnih vezij ter kvantnih algoritmov.

3.2 Zgradba delovnega okna

Po zagonu programa vidimo delovno okno (slika 3.1).



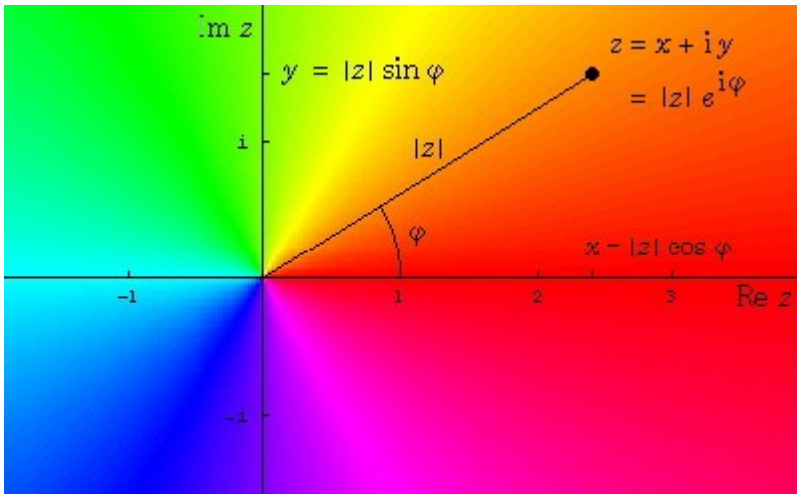
Slika 3.1 - delovno okno

Delovno okno je razdeljeno sledeče: na levem delu je nadzorna plošča, kjer so gumbi za inicializacijo in delovanje kvantnega algoritma. Zgornje vodoravno okno je sestavljeno iz gumbov za dodajanje kvantnih vrat vezju. V spodnjem oknu je predstavljen kvantni register. Razdeljen je na del, kjer dobimo izhod – x register, ter na del, ki služi hrambi začasnih qubitov – y register.

Na zgornjem delu okna imamo meni *File* za shranjevanje oziroma odpiranje algoritmov. Meni *Configuration* nam ponuja možnost barvanja glede na razdaljo. V meniju *Help* najdemo koordinatni sistem kompleksnih števil, ki nam pokaže, kaj pomenijo posamezne barve kvadratkov, ter podatke o verziji in avtorjih simulatorja.

3.3 Inicializacija kvantnega registra

Najprej je potrebno klikniti na zgornji levi gumb v nadzorni plošči, ki je označen z Ψ_0. S tem se nam odpre okno, kjer določimo število qubitov za vsak register (x in y). Vsak qubit je predstavljen kot linija, ki gre v vezje. V simulatorju ne smemo izbrati več kot 15 qubitov.



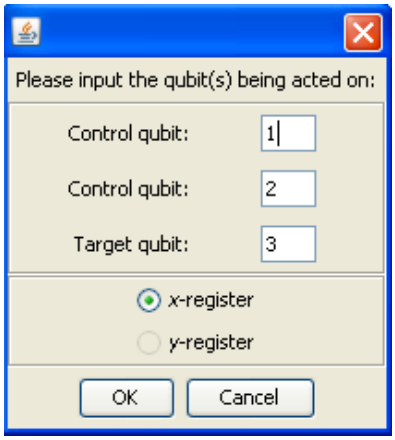
Slika 3.2 - koordinatni sistem kompleksnih števil, ki nam pokaže kaj pomenijo posamezne barve kvadratkov

Qubitni register velikosti n je določen z 2^n osnovnimi stanji. V oknu registra je vsako osnovno stanje predstavljeno s svojim kvadratkom, kjer je amplituda *z* označena z barvo, kot vidimo na sliki 3.2 . Če ima qubit |j› realno amplitudo a ∈ (0,1], potem je j-ti kvadratek rdeče barve. Tako je ponavadi inicializirana vrednost qubita, ko je nastavljena. Qubiti, ki imajo amplitudo z=0 so črne barve.

Nastavljena inicializirana vrednost qubita je |0 … 0_i› . Za spremembo začetne vrednosti in tudi za ponovno inicializacijo kvantnega vezja med ali po pogonu simulacije pritisnemo gumb |1⟩ |0⟩.

3.4 Načrtovanje vezja

Po inicializaciji kvantnega registra lahko gradimo vezje s kliki na ikone za dodajanje kvantnih vrat. Določiti je potrebno tudi qubit, ki ga simulator priključi na izbrana vrata. Podrobnosti so odvisne od vrat.



Slika 3.3 – primer nastavitve podrobnosti Toffolijevih vrat

Poleg gumbov za vrata imamo v zgornji orodni vrstici tudi gumba  ter  . Prvi je za določanje mesta meritve, drugi pa za brisanje posameznih vrat. Več o ostalih gumbih pa v naslednjem razdelku.

3.5 Upravljanje algoritma

Imamo dve možnosti, kako poženemo algoritem, ki ga implementiramo v kvantno vezje. Delovanje lahko simuliramo po korakih tako, da pritiskamo zelena gumba *blokiranih* puščic ( , ) ali pa zaženemo celoten algoritem s pritiskom na gumb  .

Trenutni položaj simulacije v vezju je predstavljen z zeleno navpično črto. Trenutno stanje kvantnega registra je predstavljeno v registrskem oknu v spodnjem delu delovnega okna.

Za ponovni zagon algoritma je potrebno pritisniti gumb  . Z gumbom lahko tud spremenimo inicializacijo qubitov.

3.6 Logične strukture v jQuantum

V okvirju za načrtovanje vezja nam orodje jQuantum nudi nekaj osnovnih gradnikov za delo s kvantnim procesiranjem. V tem razdelku bomo le-te natančneje opisali.

3.6.1 Inicializacija

Prvi nam ponuja inicializacijo qubitov. Možnost določitve začetnih vrednosti nam je na voljo šele po inicializaciji registrov. Vsakemu qubitu v x oziroma y registru lahko določimo logično vrednost 0 ali 1. To so hkrati tudi začetne vrednosti pri kasnejšem procesiranju. Nastavljene vrednosti vidimo v okencu s stanji registrov x in y.
Če na primer inicializiramo 3-bitni register x in qubite nastavimo na 110, nam slika registra x pokaže vrednost 6, kot prikazuje slika 3.4.

Slika 3.4

Poleg qubitov registra x lahko seveda poljubno nastavimo začetno vrednost qubitom registra y.

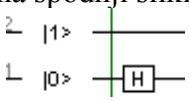
3.6.2 Hadamardova vrata

Ponuja nam možnost postavitve Hadamardovih vrat na posamezen qubit x ali y registra. Hadamardova vrata nam naredijo sledečo transformacijo na posameznem qubitu:

$$|0\rangle \rightarrow \frac{1}{\sqrt{2}} \left(|0\rangle + |1\rangle \right)$$

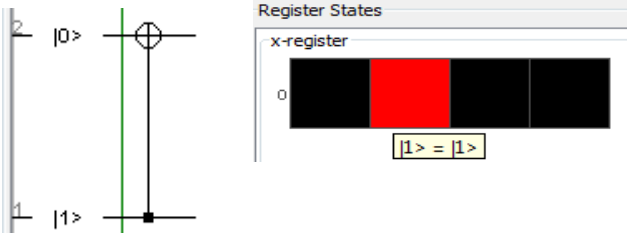
$$|1\rangle \rightarrow \frac{1}{\sqrt{2}} \left(|0\rangle - |1\rangle \right)$$

Primer Hadamardovih vrat v jQuantum je na spodnji sliki.



3.6.3 c-NOT vrata

To so kontrolirana negacijska vrata. Delujejo pa tako, da spremenijo logično vrednost drugega qubita, če je prvi bit po logični vrednosti 1, sicer pa ohranja vrednost drugega qubita. Vnesemo kontrolni in ciljni qubit.



V primeru na zgornji sliki bo qubit 2 spremenil svojo logično vrednost na 1, ko bo qubit 1 po logični vrednosti 1.

Po enem koraku je torej izhod 011 (vrednost 3).

3.6.4 Vrata X

Vrata X nam predstavljajo logično negacijo. Torej

X: $|0\rangle \rightarrow |1\rangle$

$|1\rangle \rightarrow |0\rangle$

V simulatorju izberemo qubit, ki ga želimo negirati.

3.6.5 Vrata Y

Y vrata nam predstavljajo transformacijo

Y: $|0\rangle \rightarrow -|1\rangle$

$|1\rangle \rightarrow |0\rangle$, kjer velja Y=ZX (Z vrata so opisana v nadaljevanju).

3.6.6 Vrata Z

Vrata Z imajo funkcijo zamika faze.

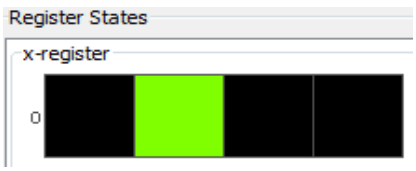
Z: $|0\rangle \rightarrow |0\rangle$

$|1\rangle \rightarrow -|1\rangle$

V simulatorju se nam na qubitu, ki smo ga zbrali, vrata X, Y ter Z prikažejo kot črka na posameznem qubitu.

3.6.7 Vrata S

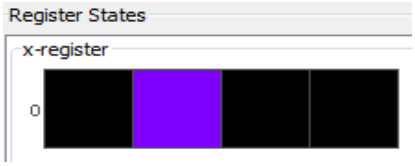
Vrata S izvedejo funkcijo zamenjave (*swap*).



Delujejo po formuli:

$$|0_y\rangle = HS\,|0\rangle = \frac{|0\rangle + \mathrm{i}\,|1\rangle}{\sqrt{2}} \qquad |1_y\rangle = HS\,|1\rangle = \frac{|0\rangle - \mathrm{i}\,|1\rangle}{\sqrt{2}}$$

3.6.8 Vrata S⁺



podobno kot vrata S, le da je formula:

$$|0\rangle = S^\dagger H\,|0_y\rangle = \frac{|0_y\rangle + |1_y\rangle}{\sqrt{2}} \qquad |1\rangle = S^\dagger H\,|1_y\rangle = \frac{|0_y\rangle - |1_y\rangle}{\sqrt{2}\,\mathrm{i}}$$

3.6.9 Vrata T

Kaj točno T vrata počnejo žal nismo našli ne v dokumentaciji in ne v člankih. Za razlago bi morali kontaktirati avtorja.

3.6.10 Vrata √X

Imenujemo jih tudi $\sqrt{NOT}$ vrata. Izvedejo operacijo po formuli:

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

3.6.11 Toffolijeva vrata

S toffolijevimi vrati lahko zgradimo konjunkcijo in negacijo ter s tem dobimo poln funkcijski nabor.
V simulatorju izberemo dva kontrolna qubita ter en ciljni qubit.
Tako lahko realiziramo tudi razne seštevalnike in podobne strukture, kot jih poznamo pri klasičnih vezjih.
Posamezna vrata tvorimo na sledeč način:

T| 1,1,x ⟩ = |1, 1, ¬x⟩

T| x, y, 0 ⟩ = | $x, y, x \wedge y$ ⟩

3.6.12 Fouriereva vrata

Kvantna Fouriereva vrata. Izvedejo Fourierevo transformacijo na bitih registra x ali y, s katero pripravimo superpozicijo.

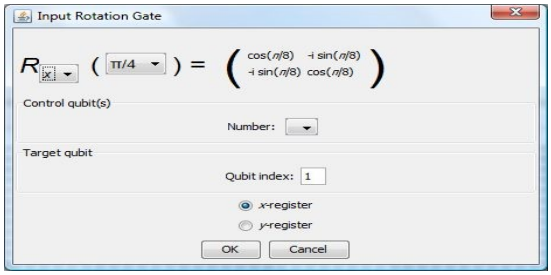
3.6.13 Inverzna Fouriereva vrata

Inverzna Fouriereva transformacija nad registrom x ali y.

3.6.14 Funkcijska vrata

Funkcijska vrata nam nudijo možnost zapisa lastne funkcije. Na primer f(x)=z^x mod 15.

3.6.15 Rotacija



Vrata zarotirajo vrednost qubita za izbrani kot.

3.6.16 Groverjev operator

Groverjev operator je uporaben za realizacijo Groverjevega iskalnega algoritma. Potrebno je podati vrednost, ki jo iščemo ter število iteracij, ki se bodo izvedle nad registrom x oziroma y.

3.6.17 Meritev

Določimo, kje v algoritmu naj se meritev izvede.

3.6.18 Brisanje

Ikona nam služi za brisanje posameznega dela algoritma oziroma vrat.

4 Predstavitev osnovnih algoritmov

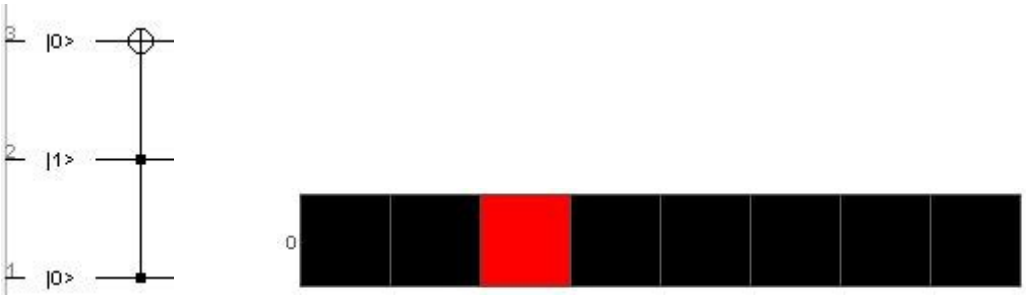
V tem poglavju bomo predstavili nekaj osnovnih algoritmov, ki bodo lepo prikazali delovanje kvantnega procesiranja. Najprej bomo poskušali s pomočjo kvantnih operatorjev v jQuantum simulatorju realizirati klasične logične funkcije: OR, AND in seštevalnik.
Zgoraj naštete funkcije, kot tudi druga klasična vrata, pa za kvantno procesiranje pravzaprav niso pomembna, saj je namen kvantnega procesiranja usmerjen drugam. To bomo spoznali s pomočjo kvantnih algoritmov. Med njimi so osnovni za razumevanje moči kvantnega procesiranja prav zagotovo Shorov algoritem iskanja prafaktorjev danega števila in Groverjev algoritem iskanja.

4.1 Klasična vezja

4.1.1 AND vrata

Na spodnji sliki lahko vidimo realizacijo klasičnih AND vrat s Toffolijevimi kvantnimi vrati. Kvantni register sestoji iz treh qubitov. Qubita 1 (x_1) in 2 (x_2) sta vhodna, izhod pa se pokaže na qubitu 3 (x_3). Po Toffolijevi operaciji se vhodni stanji ohranita. Spodaj sta prikazana dva primera. Prvič, ko sta vhoda $x_1 = 1$ in $x_2 = 1$ in drugič, ko je $x_1 = 0$ in $x_2 = 1$. Inicializacija $x_3 = 0$.

Slika 4.1 : Levo vidimo realizacijo AND vrat z Toffolijevimi kvantnimi vrati. Če sta qubit 1 in qubit 2 enaka 1, potem je tudi izhod (qubit 3) enak 1, kar je tudi pravilen rezultat AND vrat. Rezultat je razviden iz registra na desni strani slike. Register se začne na levi s stanjem 000 in konča na desni s stanjem 111. Na začetku je bilo aktivno stanje 011 (ker je $x_3 = 0$, $x_2 = 1$ in $x_1 = 1$), na koncu pa v x_3 dobimo rezultat 1, zato se aktivira stanje 111.

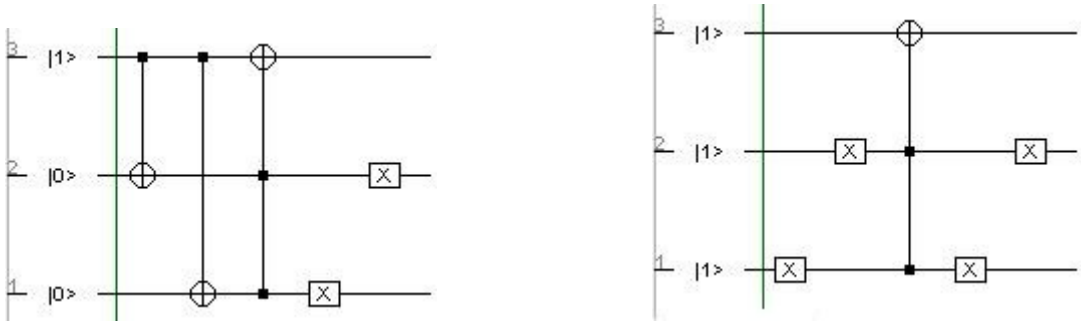


Slika 4.2 : Zgornja slika prikazuje isto funkcijo, le da ima različna vhoda x_1 in x_2.

4.1.2 OR vrata

OR vrata so realizirana z dvojnim kontroliranim NOT operatorjem (c-NOT) in enimi Toffolijevimi vrati. Na koncu so na vhodih uporabljena še Pauli-X vrata. Tudi v tem primeru qubit 3 (x_3) hrani rezultat OR operacije. Na začetku mora biti vrednost x_3 enaka 1; to pa zato, ker naš algoritem deluje na sledeč način:

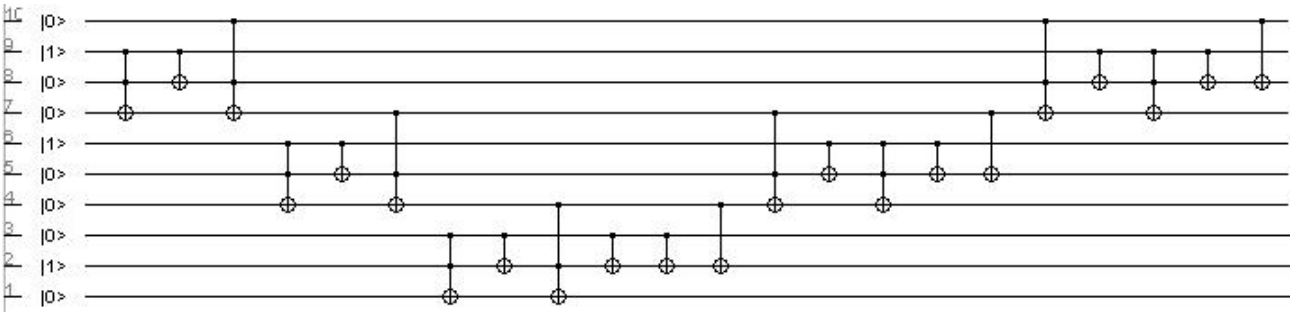
1. Najprej obe vrednosti vhodnih spremenljivk negiramo (uporabimo c-NOT, lahko bi uporabili tudi Pauli-X vrata).
2. Ker je vrednost $x_3 = 1$, se le ta spremeni le če je negirana vrednost obeh vhodnih spremenljivk enaka 1. To pa je le v primeru, ko je $x_1 = 0$ in $x_2 = 0$. Takrat je tudi izhod OR po vrednosti 0.
3. S pomočjo Pauli-X vrat vhodni vrednosti spet „odnegiramo“ tako, da imamo na izhodu enake vrednosti kot na izhodu.



Slika 4.3 : Primer različnih realizacij OR vrat s kvantnimi operatorji. V prvem primeru (levo) sta uporabljena c-NOT operatorja, Toffolijeva vrata in Pauli-X vrata. Desno pa sta namesto c-NOT uporabljena Pauli-X vrata.

4.1.3 Seštevalnik

Kot zadnje klasično vezje bomo predstavili še n-bitni seštevalnik (v našem primeru bo sešteval dve tri-bitni števili). Začetno stanje registra vsebuje naslednje zaporedje vhodov: 0, a_0, b_0, 0, a_1, b_1, 0, a_2, b_2, 0. Ničle so začetni parametri, ki so potrebni za izvajanje algoritma (carry vhodi). Najprej poračunamo vse prenose (carry) do zadnjega, ki nam da qubit, ki je potreben za nadaljnje računanje. Nato vsa računanja prenosov postopoma razveljavimo in računamo vsoto. Izhod nam na prvih n izhodih (če izvzamemo 0 na istih mestih kot so podane na vhodih) da vrednost začetnega števila a, na naslednjih n mestih (spet izvzamemo ničle in vhoda) pa nam vrne vsoto.



Slika 4.4 : Seštevalnik dveh tri-bitnih števil s pomočjo Toffolijevih vrat in cNOT operatorjev.

Slika 4.5 : Levo je samostojno prikazana ena ponovitev prenosa(carry), desno pa vsote (sum)

4.2 Osnovna neklasična vezja – algoritmi

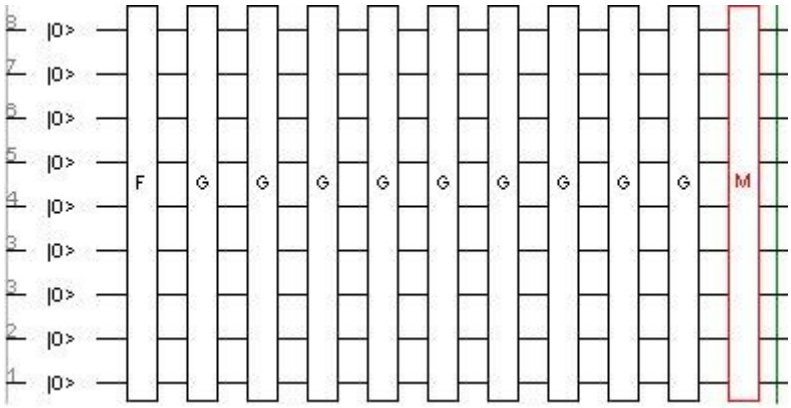
4.2.1 Gooverjev algoritem

Grooverjev algoritem je metoda za iskanje elementa **y**, ki pripada množici dvobitnih celih števil velikosti n, ki ustreza pogoju f(y) = 1. To naredimo z amplitudnim ojačanjem pravega števila v več operacijah. Amplituda pa nam pove, kolikšna je verjetnost, da nam bo meritev vrnila pravilno rešitev.

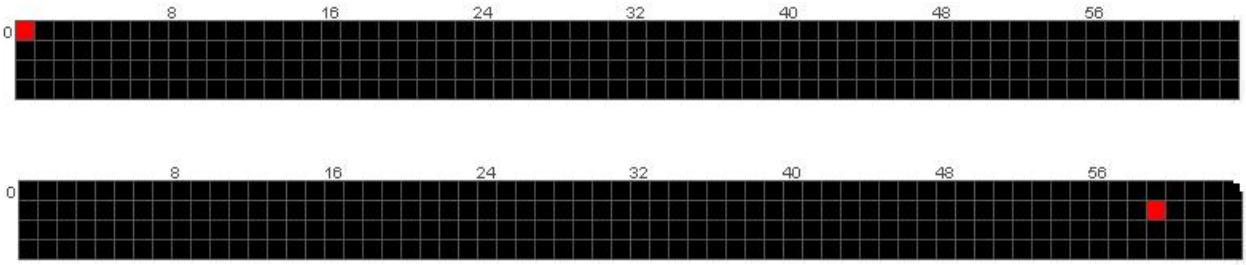
Predpostavimo, da imamo 12 bitni register. S klasičnim procesiranjem bi v najslabšem primeru morali narediti 2^{12} primerjav, s kvantnim procesiranjem pa s pomočjo Grooverjevega operatorja naredimo kvečjemu $2^{(n-4)/2}n$ operacij (iteracij). Slednje nam omogoča lastnost superpozicije, saj kvantno procesiranje algoritem izvaja nad vsemi možnimi vrednostmi registra hkrati.

Algoritem deluje tako, da v vsaki iteraciji ojača amplituda iskanega števila in oslabi amplitudo ostalih števil. Na začetku se nad registrom naredi še Fourierева transformacija, ki nam da register v superpozicijo tako, da se operacija izvaja nad vsemi možnimi vrednostmi. Nad registrom opravimo meritev in s tem dobimo najmočnejšo vrednost, ki je hkrati tudi naša rešitev. Spodaj je prikazan algoritem, v katerem iščemo število 123.

Časovna zahtevnost algoritma je O($\sqrt{(2^n)}$).



Slika 4.6 : Realizacija Grooverjevega iskalnega algoritma v orodju jQuantum.



Slika 4.7 : Na začetku mora biti register postavljen na 0, da z Fourierevo transformacijo dobimo superpozicijo registra. Po končani meritvi dobimo rezultat 123

V osnovi bi lahko bil zunanji vir kakšen kriptografski algoritem, ki bi s pomočjo znanega čistopisa uporabi neznani ključ na šifropis in ojačal njegovo vrednost, če se čistopis in tajnopis ujemata, sicer pa vrednost zmanjšal. Ker kvantni računalnik deluje paralelno nad vsemi vrednostmi registra, bi lahko neznani ključ ugotovili veliko prej kot po klasičnih poteh.

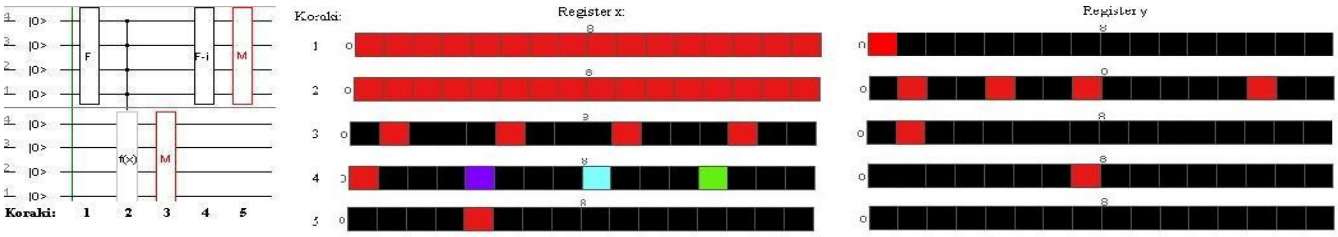
4.2.2 Shorov algoritem za iskanje prafaktorjev števil

Dandanes je zelo razširjeno prepričanje, da za iskanje prafaktorjev števila (imamo število n = p*q, najti moramo praštevili p in q) ni nobenega algoritma, ki bi problem rešil v zglednem času. Ta problem ima veliko vlogo v današnjem kriptografski domeni. Ravno RSA kriptosistem z javnim ključem temelji na domnevi, da ni uspešnega algoritma, ki bi rešil faktorizacijo.

Peter W. Shor je leta 1994 predstavil verjetnostni kvantni faktorizacijski algoritem, ki zahteva polinomski čas. Algoritem temelji na faktorizaciji s pomočjo zaporedij v množici celih števil. Za uspeh tega algoritma pa moramo izračunati periodo zaporedja. Ta izračun pa je računsko zelo zahteven. Dejstvo je, da se s pomočjo Fouriereve transformacije da izločiti informacijo o periodičnosti funkcije.

Celoten algoritem (tudi deli, ki ne vsebujejo kvantnega računanja) poteka takole: najprej izberemo naključno a, ki je element množice celih števil v intervalu od 1 do n-1 (kjer je n število, katerega prafaktorje iščemo). Nato izračunamo največji skupi delitelj a in n (s pomočjo Evklidovega algoritma). Če je le ta večji od 1, potem d ni trivialen faktor števila n in naše delo je končano. V nasprotnem primeru pa s kvantnim algoritmom izračunamo periodo zaporedja r. Na koncu še izračunamo največji skupni delitelj števila n in $a^{r/2}$+/- 1 z Evklidovim algoritmom. Po zadnjem koraku je rešitev že znana.

Sedaj pa poglejmo še del algoritma, kjer se izvaja kvantno računanje: najprej pripravimo superpozicijo registra x, nato izračunamo $f(x) = a^x \pmod n$. Rezultate nato shranimo v y register in nad njim naredimo meritev. Ta nam v registru x da vsa stanja, ki so povezana s stanji v registru y. Nato izvedemo meritev še nad registrom x. Tako dobimo vrednosti, ki nam pomagajo pri izračunu prafaktorjev števila n. Na koncu še izračunamo ugnezden ulomek p_i/q_i in r (perioda zaporedja), ki je enak najmanjšemu q_i, kjer velja $a^{qi} \equiv 1 \pmod n$.



Slika 4.8 : Realizacija kvantnega dela Shorovega algoritma v simulatorju jQuantum (levo) in vrednosti registrov po vseh petih korakih (rešitev prikazana v korakih 3, 4 in 5 ima verjetnost ¼)

Zgornja slika nam prikazuje primer, ko iščemo prafaktorje števila 15. Izbran je a = 7. Kot smo že omenili, najprej pripravimo superpozicijo s pomočjo Fouriereve transformacije. V registru y dobimo v naslednjem koraku rešitve funkcije $7^x \pmod{15}$ za vsa števila v registru x. V našem konkretnem primeru so te vrednosti naslednje:
¼ ((|0> + |4> + |8> + |12>)*|1> + (|1> + |5> + |9> + |13>)*|7> +
(|2> + |6> + |10> + |14>)*|4> + (|3> + |7> + |11> + |15>)*|13>).

Z meritvijo y registra se nam v registru x pokažejo vrednosti, za katere velja f(x) = y. Inverz Fouriereve transformacije nam razkrije amplitudne maksimume. V našem primeru dobimo štiri vrednosti: |0>, |4>, |8>, |12>. Vsako od teh vrednosti nam lahko zadnja meritev vrne z verjetnostjo ¼. Iz dobljenih vrednosti nato z lahkoto izračunamo periodo zaporedja po prej omenjenem postopku.

Celoten algoritem se izvede v času $O(n^3)$. Verjetnost uspeha algoritma pa je $\Omega(1/\log n)$. Torej moramo algoritem pognati O(log(n))-krat, če hočemo dobiti našo pravilno, netrivialno rešitev. Rešitev dobimo torej v času $O(n^3 \log n)$.

5 Zaključek

Skozi pisanje seminarske naloge smo prišli do spoznanja, da je področje kvantnega procesiranja zelo obsežno in zahtevno področje, za katerega popolno razumevanje bi potrebovali veliko več časa ter predznanja kvantne fizike in mehanike. V seminarski nalogi smo se dotaknili le osnov, ki smo jih prenesli v opis delovanja simulatorja jQuantum. Preko dela z njim, smo simulator dodobra spoznali. Na žalost je dokumentacija zelo skopa in ne obsega razlage vrat, ki so na voljo za delo v samem simulatorju. Za pomoč smo se obrnili k knjigam in člankom, ki so objavljeni na internetu. Precejšen del funkcionalnosti pa smo ugotovili z poizkušanjem ter logičnim sklepanjem.
Na koncu lahko rečemo, da je kvantno procesiranje zanimivo področje, ki definitivno ima perspektivo na področju računalništva.

6 Literatura

- Eleanor Rieffel: An Introduction to Quantum Computing for Non Physicists
- Hirvensalo, M.: Quantum Computing. New York, Springer, 2001.
- Pittenger O. A.: An introduction to Quantum Computing Algorithms. Birkhäuser, Boston, 2000.
- Colin P. Williams, Scott H. Clearwater: Explorations in QUANTUM COMPUTING, New York, Springer, 1998

7 Viri

- http://jquantum.sourceforge.net/
- http://www.quantiki.org/wiki/index.php/Basic_concepts_in_quantum_computation
- http://staff.cs.utu.fi/~mikhirve/Lectures/
- http://xxx.lanl.gov/abs/quant-ph/9802065
- http://www.scribd.com/doc/7300246/jQuantumdoc
- http://physics.aps.org/articles/v1/35