2. seminarska naloga pri predmetu ONT: Kvantno procesiranje

Martin Kranjc, Blaž Parkelj
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko,
Ljubljana, Slovenija

Povzetek

Seminarska naloga obravnava možne implementacije nekaterih osnovnih vezij klasičnega računalništva s kvantnimi vrati. Pri načrtovanju sva uporabljala samo najpogosteje uporabljana kvantna vrata. Vsa vezja so bila narejena s programom jQuantum.

Ključne besede: kvantno procesiranje, jQuantum, kvantna vrata

Kazalo vsebine

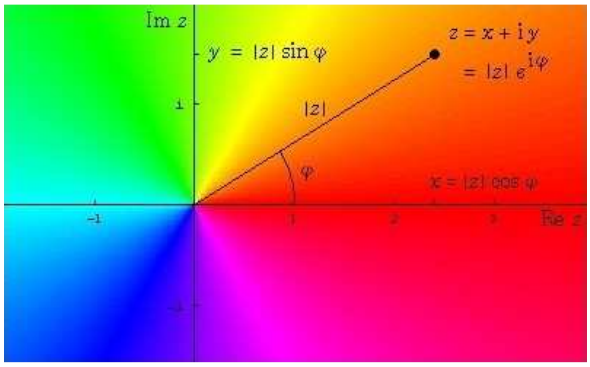
1. Uvod

Kvantni računalnik je naprava, ki za procesiranje podatkov izkorišča fenomene kvantne fizike, kot je na primer princip superpozicije. Čeprav je kvantno procesiranje še vedno novost v računalništvu, je bilo izvedenih že veliko eksperimentov na tem področju. Ti eksperimenti so bili izvedeni z majhnim številom qubitov. V primeru, da bi bili sposobni zgraditi kvantni računalnik večjih pomnilniških zmogljivosti, bi bil takšen računalnik sposoben rešiti določene primere veliko hitreje kot klasični računalnik. Narejenih je tudi že nekaj algoritmov, ki eksponentno zmanjšajo potreben čas za rešitev problema. Tipičen primer bi bil Shorov algoritem, ki vrne delitelje nekega celega števila.
V klasičnih računalnikih je osnovna enota za pomnjenje informacije bit, v kvantnih računalnikih pa qubit (quantum bit). Razlika med njima je ta, da se lahko bit nahaja v enem od dveh osnovnih stanj: ˝0˝ ali ˝1˝, qubit pa se lahko nahaja v osnovnem stanju: ˝0˝ ali ˝1˝ ali v kateremkoli superpozicioniranem stanju, ki je kombinacija obeh stanj. To stanje qubita $|q\rangle$ lahko opišemo z $|q\rangle = a * |0\rangle + b * |1\rangle$, kjer sta a in b imaginarni števili in za kateri velja $|a|^2 + |b|^2 = 1$.
Dva qubita se lahko nahajata v katerikoli kvantni superpoziciji štirih stanj in trije qubiti se lahko nahajajo v katerikoli kvantni superpoziciji osmih stanj. Torej računalnik z n qubiti se lahko nahaja v 2^n stanjih hkrati. To je povsem drugačen koncept od klasičnih računalnikov, ki so lahko naenkrat samo v enem od 2^n stanjih. Samo kvantno procesiranje poteka tako, da kvantni računalnik upravlja z qubiti v skladu z zaporedjem kvantnih vrat. Zaporedju kvantnih vrat pravimo tudi kvantni algoritem.
V tej seminarski bova predstavila implementacije nekaj klasičnih vezij. Za implementacijo sva uporabila program jQuantum, ker je javno dostopen in ima vgrajena osnovna kvantna vrata.

2. Metode

2.1. Program jQuantum

jQuantum je program za simulacijo kvantnega računalnika na klasičnem računalniku in ima preprost grafični uporabniški vmesnik. Nudi nam dva kvantna registra x in y, ki sta med seboj popolnoma ločena. Posameznih qubitov iz obeh registrov ne moremo hkrati uporabiti v istih vratih, kot so na primer Toffolijeva vrata. Za načrtovanje vezja moramo najprej določiti število qubitov oziroma velikost obeh kvantnih registrov. Tu bi opozoril, da se vsakič, ko ju nastavimo, vsa vrata zbrišejo in da program nima vgrajene opcije »razveljavi« (Undo), s katero bi razveljavili zadnji ukaz. Nato lahko poljubno dodajamo vrata, pri čemer moramo določiti qubit ali qubite (če delujejo nad več kot enim), nad katerimi naj se izvedejo. Nekatera vrata delujejo samo nad enim qubitom, druga nad večimi.
Kvantni register dolžine n ima 2^n osnovnih stanj. Vsako osnovno stanje je v programu predstavljeno s svojim kvadratkom in barve kvadratkov določajo kompleksna števila $a, b, \ldots$ *Slika 1* nam kaže pomen vseh barv razen črne. Črna barva pa pomeni, da je vrednost imaginarnega števila enaka 0.



Slika 1: Koordinatni sistem kompleksnih števil, ki nam kaže pomen barv

V nadaljevanju bova predstavila tista vrata, ki sva jih uporabljala pri načrtovanju vezij, vendar to še zdaleč niso vsa. Bralec si lahko v različni literaturi pogleda še ostala najbolj uporabljana vrata. Kdor se bo pa s tem več ukvarjal, pa si lahko definira tudi svoje.

2.2. Toffolijeva vrata

Toffolijeva vrata delujejo nad tremi biti ali qubiti. Ta vrata so enaka tako v kvantnem kot v klasičnem procesiranju. Definirana so z naslednjim izrazom: $|a, b, c\rangle = |a, b, c \oplus ab\rangle$. Lahko pa jih predstavimo tudi z pravilnostno tabelo:

Vhod			Izhod		
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Tabela 1: Pravilnostna tabela Toffolijevih vrat

Tu velja opozoriti, da jQuantum ne prepreči, da na primer za vse tri vhode določimo isti qubit. V tem primeru se lahko spreminjajo tudi vrednosti sosednih qubitov. Na to mora uporabnik sam paziti, da do tega ne pride.

2.3. Vrata CNOT

CNOT vrata delujejo nad dvema qubitoma. Prvi qubit služi za kontrolo: če je postavljen se drugi negira, če ni, potem drugi qubit ohrani vrednost. Definirana so z naslednjo matriko:

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

2.4. Vrata X

Vrata X se imenujejo tudi Paulijeva X vrata. Delujejo nad enih qubitom in vršijo funkcijo negacije. Definirana so z naslednjo matriko:

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

2.5. Hadamardova vrata

Hadamardova vrata delujejo nad enih qubitom. Stanje $|0\rangle$ preslikajo v $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ in $|1\rangle$ v $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Definirana so z naslednjo matriko:

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

2.6. Vrata Z

Vrata X se imenujejo tudi Paulijeva Z vrata in delujejo nad enih qubitom. Definirana so z naslednjo matriko:

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

2.7. Measure vrata

S temi vrati opravimo meritev vrednosti celega kvantnega registra ali pa posameznega qubita. Sama meritev je invazivna (in ireverzibilna), ki iz množice vseh rezultatov naključno vrne samo enega. Uporabnik nima vpliva na to, kateri rezultat vrne. To pomeni, da lahko dobimo rezultat, ki ni rezultat operacije nad vhodnimi podatki. Zaradi tega dejstva teh vrat nisva uporabljala in zato sva vezja implementirala tako, da ne vsebujejo superpozicioniranih stanj.

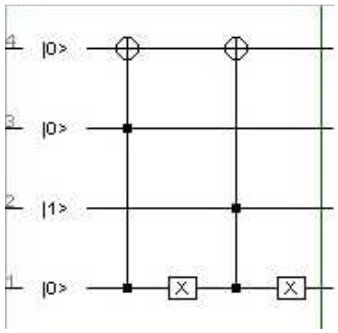
3. Rezultati

3.1. Multiplekser

Multiplekser je naprava, ki izbere enega od podatkovnih vhodov in vrednost tega vhoda prenese na izhod. Tipično ima 2^n podatkovnih vhodov, n kontrolnih signalov in en izhod. Kateri vhod se prenese na izhod je določeno s kontrolnimi signali.

3.1.1. MUX 2/1

Multiplekser 2/1 je vezje, ki ima en izhod, dva podatkovna vhoda in en kontrolni signal, s katerim se izbere en od podatkovnih vhodov, ki gre na izhod. Ena od možnih realizacij je prikazana na *Slika 2*. Prvi (spodnji) qubit je kontrolni vhod, s katerim izbiramo med drugim in tretjim qubitom. Četrti qubit je izhod, ki mora na začetku imeti vrednost 0.

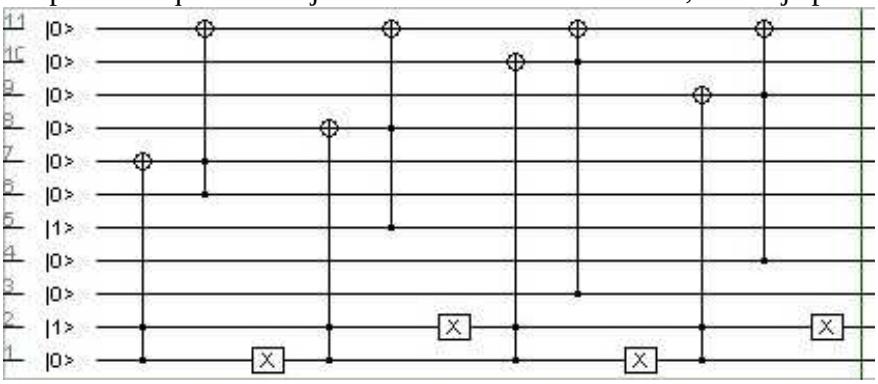


Slika 2: MUX 2/1

V splošnem multiplekser deluje po enačbi $Z = A\bar{S} \vee BS$, kjer sta A in B podatkovna vhoda, Z izhod in S kontrolni signal. Ampak prvi in drugi člen ne bosta nikoli hkrati imela vrednost 1. Zato ga lahko opišemo tudi z enačbo $Z = A\bar{S} \oplus BS$, po kateri je bilo to vezje tudi narejeno.

3.1.2. MUX 4/1

Multiplekser 4/1 je vezje, ki ima en izhod, štiri podatkovne vhode in dva kontrolna vhoda, s katerima se izbere podatkovni vhod, ki gre na izhod. Ena od možnih realizacij je na Slika 3. Prvi in drugi (spodnja dva) qubit sta kontrolna vhoda, s katerima izbiramo med podatkovnimi qubiti, ki so na pozicijah od 3 do 6. Od sedmega do desetega qubita so pomožni qubiti in enajsti qubit je izhod. Izhodni in pomožni qubiti morajo imeti na začetku vrednost 0, da vezje pravilno deluje.



Slika 3: MUX 4/1

3.2. Pomikalnik

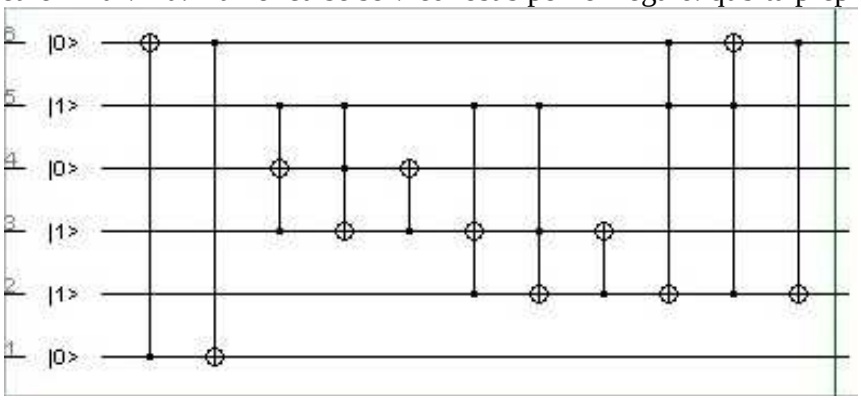
Imenovan tudi ˝Shifter˝ in je ena osnovnih funkcij vsakega sodobnega procesorja oziroma njihovih ALE enot. Gre za pomikanje bitov na sosednje ležeče po en korak. Funkcija, ki deluje nad enim registrom, je običajno realizirana na več načinov. Poznamo pomikanje v levo in desno, za vsako od teh dveh pa obstajata po dve možnosti. Ena je ta, da na prosto začetno mesto vstavimo bit, ki je izpadel iz konca registra, kar se imenuje tudi rotacijski način, pri drugi pa na začetno sproščeno mesto vstavljamo ničle. Pomikanja z vstavljanjem ničel so pogosto uporabljene pri funkciji množenja (pomik levo) in deljenja (pomik desno) za faktor 2 pri premiku za 1 bit.

V seminarski nalogi sva realizirala obe možnosti pomikanja v eni smeri, saj je obrnjena smer ekvivalentna. Gre za 4-bitni pomikalnik, za katerega sva porabila poleg štirih podatkovnih qubitov (biti od 1 do 4), še dva dodatna (5 in 6). Peti qubit, ki mora biti postavljen na logično vrednost 1, služi za izvajanje funkcije Toffolijevih vrat pri prenosu informacije iz ene linije na

njeno sosednjo. Šesti pa služi za shranitev vrednosti prvega qubita. Podatkovni qubiti sprejemajo vrednosti zaporedno: najprej najvišje ležeči prevzame vrednost njegovega spodnjega soseda, ta ponovno vrednost svojega soseda in tako naprej do predzadnjega.

3.2.1. Pomikalnik z vstavljanjem ničel

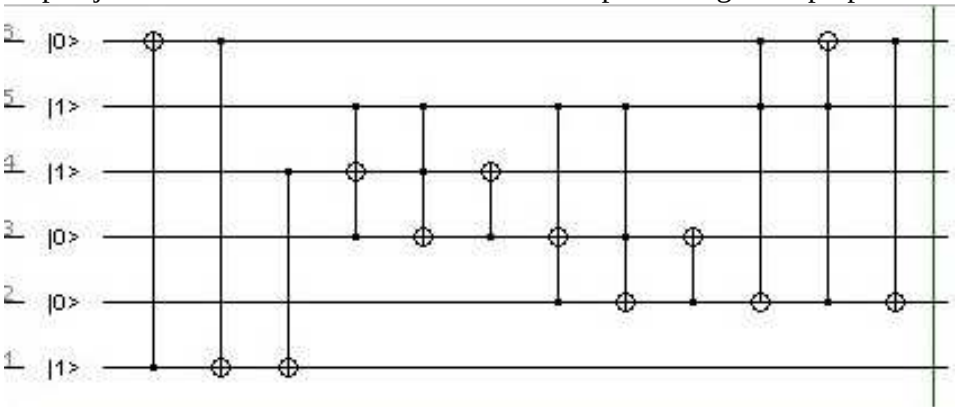
Pri tej izvedbi sta na začetku najprej dva C-NOT operatorja, ki preneseta vrednost iz 1. na 6. qubit, hkrati pa se vrednost qubita 1 postavi na ˝0˝. Zatem sledi le pomikanje qubitov v smeri navzgor, z začetkom na vrhu. Na koncu se še vrednost s pomožnega 6. qubita prepiše na 2. qubit.



Slika 4: Vezje pomikalnika z vstavljanjem ničel

3.2.2. Pomikalnik z rotacijskim načinom

Začetna dva operatorja sta enaka kot v prejšnjem primeru. Edina razlika v realizaciji so dodatna vrinjena C-NOT vrata na tretje mesto, ki informacijo s 4. qubita prepišejo na spodnji 1. qubit. Celoten postopek je enak in tudi tu se na koncu vrednost s pomožnega bita prepiše na 2. qubit.



Slika 5: Vezje pomikalnika z rotacijskim načinom

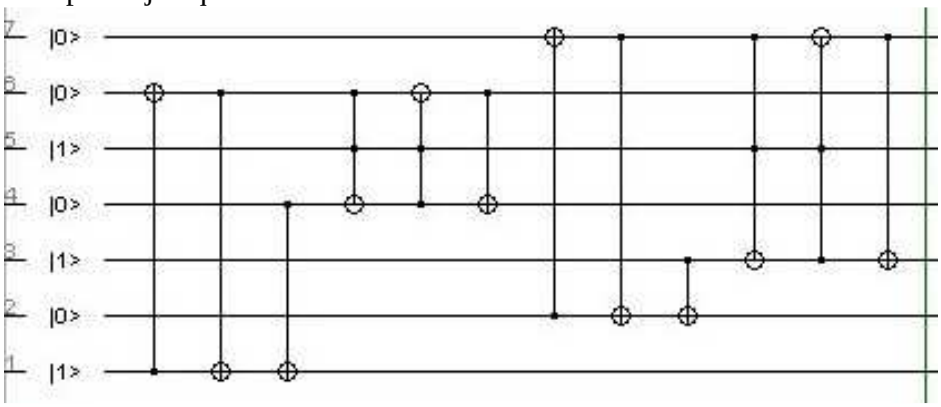
3.3. SWAP preslikava

Funkcija SWAP oziroma preslikava bitov ˝preko sredine˝ je v računalništvu bolj redko uporabljena, je pa tudi ta tipičen primer funkcije ki ima enako število vhodnih in izhodnih podatkovnih signalov. Realizirala sva jo na dva načina.

3.3.1. 4-bitni SWAP

To je bil prvi poizkus realizacije te funkcije. Za izvajanje so poleg štirih podatkovnih signalov potrebni še trije kontrolni, razdeljeni v dve skupini. 5. qubit ima vedno vrednost ˝1˝ in služi kot

pri pomikalniku za premikanje qubitov iz linije na linijo, 6. in 7. qubit pa imata nalogo shranitve informacije za zamenjavo, enako kot 6. qubitu pri pomikalniku, s tem da je po en qubit (6. ali 7.) za vsak par podatkovnih bitov. Na primer qubit 6 prevzame vrednost qubita 1 in ga hkrati postavi na ˝0˝, zatem pa prenesemo vrednost s četrtega na prvi qubit, na koncu pa še vrednost s pomožnega 6. na četrti qubit. Enako se zgodi pri zamenjavi 2. in 3. qubita, le da je tu kot pomožni qubit uporabljen qubit 7.

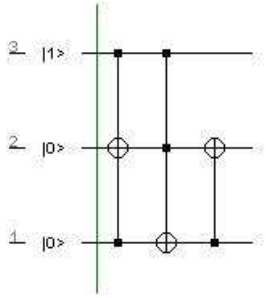


Slika 6: 4-bitni SWAP

Slabost te realizacije je njena nepotrebna kompleksnost, veliko število vrat in veliko število linij glede na število podatkovnih bitov, saj za vsak par potrebujemo svoj pomožni bit. Ta slabost je spodbujala k iskanju boljše rešitve, ki sva jo kaj kmalu našla z malo poizkušanja in opazovanja že podanih realizacij. Tako sva realizirala še 8-bitno verzijo te funkcije.

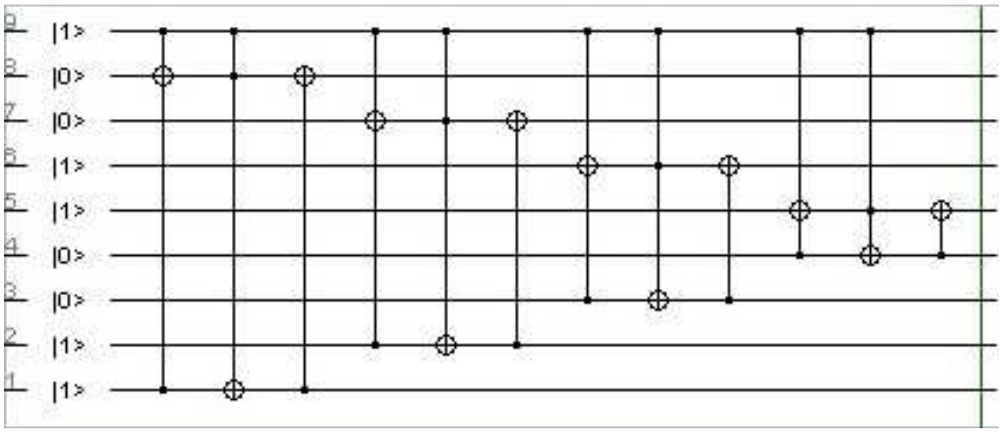
3.3.2. 8-bitni SWAP:

Pri opazovanju zaporedja operatorjev Toffoli + Toffoli + C-NOT, ki je bilo uporabljeno v obeh tipih pomikalnikov in 4-bitnemu SWAP, sva opazila, da se pri določenem vhodu vrednosti dveh bitov samodejno zamenjata.



Slika 7: Vezje, ki zamenja vrednosti prvega in drugega qubita

Tako sva realizirala SWAP funkcijo z osmimi biti in samo enim kontrolnim signalom:

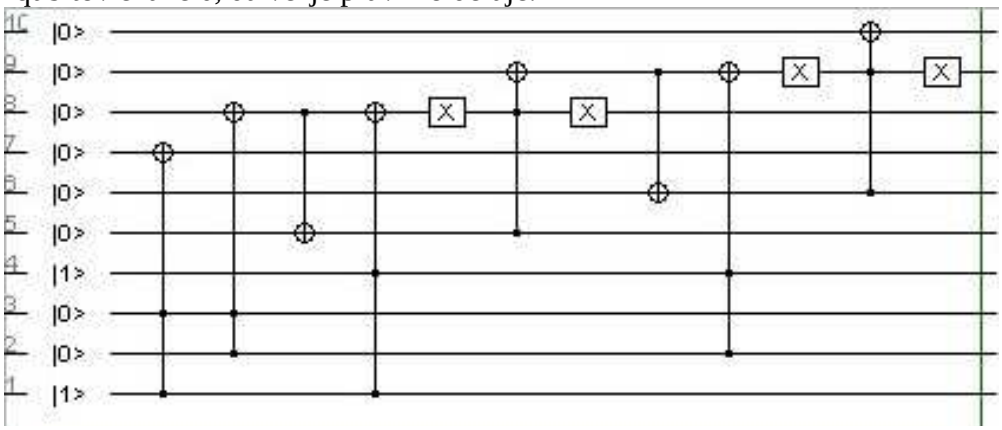


Slika 8: 8-bitni SWAP

Ta realizacija je že na pogled bolj enostavna in tudi iz *Slika 8* si lahko predstavljamo delovanje funkcije same.

3.4. Množilnik

Kot zadnje klasično vezje bova predstavila še množilnik dveh dvo-bitnih števil. Ker so vhodni števili dvo-bitni, je tudi vezje majhno in enostavno. Že pri tro-bitnih vhodnih številih, bi potrebovala kvantni register velikosti do 20 (6 qubitov za vhodni števili, 6 za rezultat in najmanj 6 za pomožne rezultate). Pri tako velikem registru pregledovanje rezultatov postane težko, ker se program pri premikanju drsnika začne odzivati zelo počasi.
Vezje na *Slika 9* ima vhodni števili na 1. in 2. qubitu oziroma na 3. in 4., rezultat dobimo na qubitih od 7 do 10, 5. in 6. pa sta pomožna qubita. Razen vhodnih števil morajo biti začetne vrednosti qubitov enake 0, da vezje pravilno deluje.



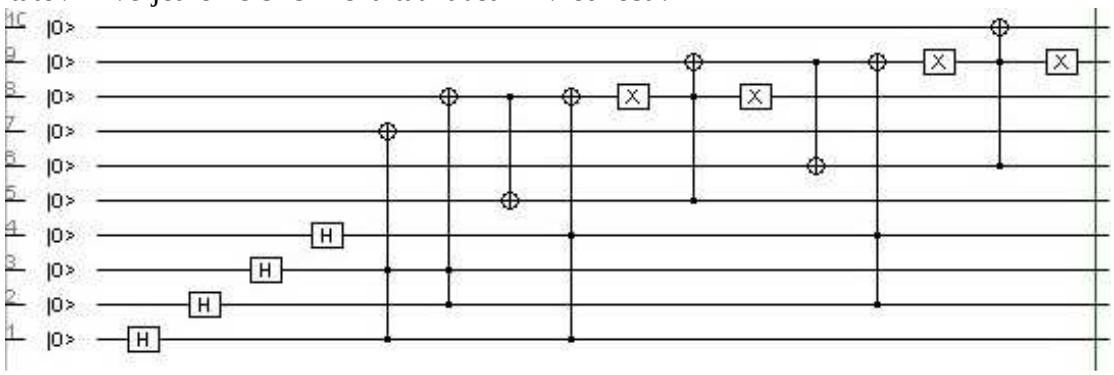
Slika 9: Množilnik dvo-bitnih števil

Vhod						Izhod				
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0
0	0	0	2	1	0	0	0	0	0	0
0	0	0	3	1	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0

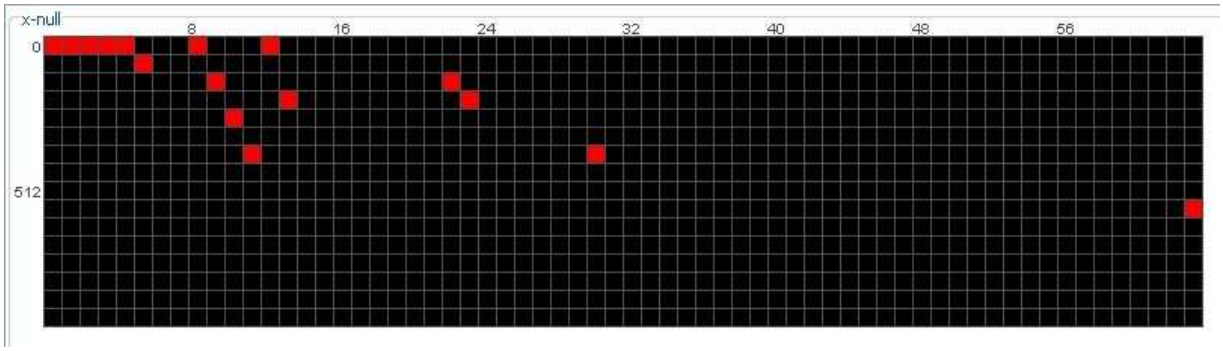
1	0	1	1	0	1	1	0	0	0	1
1	0	1	2	1	0	2	0	0	1	0
1	0	1	3	1	1	3	0	0	1	1
2	1	0	0	0	0	0	0	0	0	0
2	1	0	1	0	1	2	0	0	1	0
2	1	0	2	1	0	4	0	1	0	0
2	1	0	3	1	1	6	0	1	1	0
3	1	1	0	0	0	0	0	0	0	0
3	1	1	1	0	1	3	0	0	1	1
3	1	1	2	1	0	6	0	1	1	0
3	1	1	3	1	1	9	1	0	0	1

Tabela 2: Pravilonostna tabela množilnika

To vezje zna izračunati produkt dveh števil. Če pa vezje rahlo dopolnimo, pa lahko dobimo rezultate vseh možnih produktov dveh dvobitnih števil. To naredimo tako, da na začetek vezja dodamo vsem vhodnim qubitom Hadamardova vrata, ostalo vezje ostane popolnoma enako. Na ta način osnovno stanje pretvorimo v superpozicionirano stanje, torej bo tudi rezultat v superpozicioniranem stanju. Tu bi lahko opravili meritev, ampak bi dobili naključno enega od rezultatov in verjetno ne bi bil rezultat začetnih vrednosti.



Slika 10: Množilnik dvobitnih števil, ki jih pretvori v superpozicionirano stanje

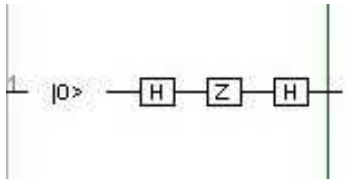


Slika 11: Rezultati vseh produktov dveh dvobitnih števil

3.5. Negator

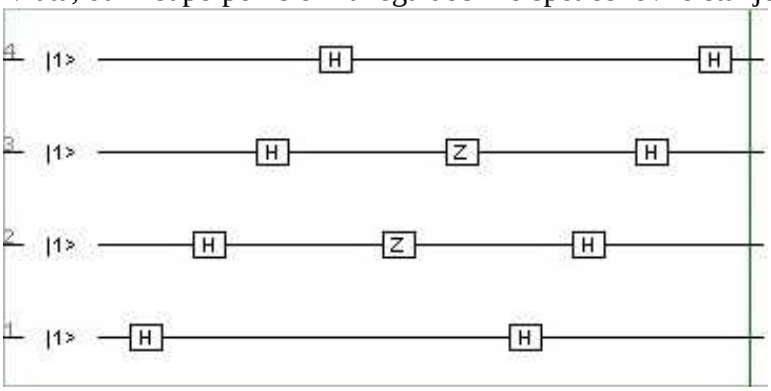
Kot primer neklasičnega vezja bova predstavila še vezje, ki vrši funkcijo negacije. To vezje je zelo enostavno in je sestavljeno iz treh zaporednih vrat: Hadardova, Z in Hadamardova vrata.

Hadamardova vrata služijo temu, da osnovno stanje qubita pretvorijo v superpozicionirano stanje in obratno.



Slika 12: Negator kot primer neklasičnega vezja

Vezje je mogoče posplošiti tudi na več qubitov. Na vsak qubit postavimo Hadamardova vrata in nato tistim qubitom, ki jih želimo negirati, dodamo Z vrata. Na koncu postavimo na vsak qubit še ena Hadamardova vrata, da iz superpozicioniranega dobimo spet osnovno stanje.



Slika 13: Negacija dveh od štirih qubitov

4. Zaključek

Kvantno procesiranje je zelo zahtevno področje, ki se šele razvija. Raziskav je vsako leto več, kar kaže med drugim tudi na to, kako perspektivno je to področje. Že samo dejstvo, da omogoča reševanje problemov, ki imajo na klasičnih računalnikih eksponentno časovno zahtevnostjo, je dovolj, da so nekatera podjetja začela vlagati sredstva v razvoj te tehnologije. Po drugi strani pa je to tako zahtevno področje, da sva se pri tej seminarski nalogi seznanila samo z osnovami. Sicer pa je bolj malo programov, ki bi omogočali toliko kot jQuantum. Edini, ki je primerljiv je program QCad, ampak ta ne omogoča simulacije po korakih temveč samo v celoti. Obstaja sicer še veliko drugih programov in knjižnic, ki simulirajo kvantni računalnik. Toda velika večina jih še vedno deluje samo v konzoli, kjer mora uporabnik upravljati z vektorji in matrikami. Ti programi so bolj namenjeni raziskovalcem, ki načrtujejo svoja kvantna vrata.

Literatura

- http://en.wikipedia.org/wiki/Quantum_computer
- http://en.wikipedia.org/wiki/Quantum_gate
- http://jquantum.sourceforge.net/
- http://www.quantiki.org/wiki/index.php/List_of_QC_simulators
- http://apollon.cc.u-tokyo.ac.jp/~watanabe/qcad/